aws

Panduan Developer

Amazon Simple Notification Service

Amazon Simple Notification Service: Panduan Developer

Table of Contents

Apa itu Amazon SNS? 1
Fitur dan kemampuan 3
Layanan terkait 4
Mengakses Amazon SNS 5
Harga Amazon SNS 6
Skenario Amazon SNS umum 6
Integrasi aplikasi 6
Pemberitahuan aplikasi 7
Notifikasi pengguna 7
Notifikasi push seluler 7
Bekerja dengan AWS SDK 8
Sumber kejadian dan tujuan Amazon SNS 10
Sumber kejadian 10
Analitik 10
Integrasi aplikasi 11
Manajemen penagihan dan biaya 11
Aplikasi bisnis 12
Komputasi 12
Kontainer 13
Keterlibatan pelanggan 14
Basis data 15
Alat developer 16
Web & seluler front-end 17
Pengembangan permainan 17
Internet untuk Segala (IoT) 18
Machine learning 18
Manajemen & tata kelola 19
Media 21
Migrasi & transfer 22
Jaringan & pengiriman konten 23
Keamanan, identitas, & kepatuhan 24
Nirserver 25
Penyimpanan 25
Sumber kejadian tambahan 27

Tujuan kejadian 28
Tujuan A2A 28
Tujuan A2P 29
Penyiapan 32
Buat akun dan pengguna IAM 32
Mendaftar untuk Akun AWS 32
Buat pengguna dengan akses administratif 33
Langkah selanjutnya 34
Memulai 35
Prasyarat 35
Langkah 1: Buat topik 35
Langkah 2: Buat langganan topik 35
Langkah 3: Publikasikan pesan ke topik 36
Langkah 4: Hapus langganan dan topik 37
Langkah selanjutnya 37
Mengonfigurasi Amazon SNS 38
Membuat topik 38
AWS Management Console 39
AWS SDK 41
Berlangganan topik 55
Untuk berlangganan endpoint topik Amazon SNS 55
Menghapus langganan dan topik 57
AWS Management Console 57
AWS SDK 58
Penandaan 67
Penandaan untuk alokasi biaya 68
Penandaan untuk kontrol akses 68
Penandaan untuk pencarian dan penyaringan sumber daya 70
Mengonfigurasi tag 71
Pengurutan pesan dan deduplikasi (topik FIFO) 77
Kasus penggunaan topik FIFO 77
Detail pemesanan pesan 79
Grup pesan 82
Mendistribusikan data dengan ID grup pesan untuk meningkatkan kinerja 83
Pengiriman pesan 84
Pemfilteran pesan 85

Deduplikasi pesan ... 86
Keamanan pesan ... 88
Daya tahan pesan ... 89
Pengarsipan dan pemutaran ulang pesan ... 91
Apa itu pengarsipan dan pemutaran ulang pesan ... 92
Untuk pemilik topik ... 93
Untuk pelanggan topik ... 98
Contoh-contoh kode ... 102
Contoh FIFO (AWSSDK) ... 102
Contoh FIFO (AWS CloudFormation) ... 115
Penerbitan pesan ... 120
AWS Management Console ... 120
AWS SDK ... 121
Muatan pesan large ... 144
Perpustakaan Klien yang Diperluas untuk Java ... 144
Perpustakaan Klien Diperpanjang untuk Python ... 149
Atribut pesan ... 152
Pesan atribut item dan validasi ... 153
Jenis Data ... 154
Atribut pesan yang dicadangkan untuk notifikasi push seluler ... 155
Batching pesan ... 157
Apa itu batching pesan? ... 157
Bagaimana cara kerja batching pesan? ... 157
Contoh ... 157
Pemfilteran pesan ... 161
Cakupan kebijakan filter ... 161
Kebijakan filter langganan ... 162
Contoh kebijakan filter ... 163
Kendala kebijakan filter ... 166
Logika DAN/ATAU ... 170
Pencocokan kunci ... 175
Pencocokan nilai numerik ... 177
Pencocokan nilai string ... 179
Menerapkan kebijakan filter langganan ... 186
AWS Management Console ... 186
AWS CLI ... 187

AWS SDK 188
API Amazon SNS 192
AWS CloudFormation 193
Menghapus kebijakan filter langganan 193
AWS Management Console 193
AWS CLI 194
API Amazon SNS 194
Perlindungan data pesan 195
Apa itu perlindungan data pesan 195
Mengapa menggunakan perlindungan data pesan 196
Kebijakan perlindungan data 196
Apa itu kebijakan perlindungan data? 197
Ikhtisar struktur kebijakan perlindungan data 197
Bagaimana cara menentukan prinsip IAM 200
Operasi kebijakan perlindungan data 200
Contoh kebijakan perlindungan data 209
Membuat kebijakan perlindungan data 216
Menghapus kebijakan perlindungan data 225
Pengidentifikasi data 226
Pengidentifikasi data terkelola 227
Pengidentifikasi data khusus 265
Pengiriman pesan 268
Pengiriman pesan mentah 268
Mengaktifkan pengiriman pesan mentah menggunakan AWS Management Console 269
Contoh format pesan 269
Atribut pesan dan pengiriman pesan mentah untuk langganan Amazon SQS 270
Pengiriman lintas akun 270
Pemilik antrean membuat langganan 271
Pengguna yang tidak memiliki antrian membuat langganan 273
Bagaimana cara memaksa langganan untuk meminta otentikasi pada permintaan berhenti berlangganan? 276
Pengiriman lintas wilayah 276
Wilayah Keikutsertaan 276
Status pengiriman pesan 279
Mengkonfigurasi pencatatan status pengiriman menggunakan AWS Management Console 280

Mengonfigurasi pencatatan status pengiriman menggunakan SDK AWS 281
AWS Contoh SDK untuk mengonfigurasi atribut topik 283
Mengkonfigurasi pencatatan status pengiriman menggunakan AWS CloudFormation 291
Pengiriman ulang pesan 293
Protokol dan kebijakan pengiriman 293
Tahap kebijakan pengiriman 294
Membuat kebijakan pengiriman HTTP/S 295
Antrean surat mati (DLQs) 302
Mengapa pengiriman pesan gagal? 303
Bagaimana cara kerja antrean surat mati? 304
Bagaimana pesan dipindahkan ke antrean surat mati? 304
Bagaimana cara memindahkan pesan dari antrean surat mati? 304
Bagaimana saya bisa memantau dan mencatat antrean surat mati? 305
Mengonfigurasi antrean surat mati 305
Pengarsipan dan analitik pesan 311
Pesan A pplication-to-application (A2A) 312
Aliran pengiriman Fanout ke Firehose 312
Prasyarat 313
Berlangganan aliran pengiriman untuk topik 315
Tujuan aliran pengiriman 316
Contoh kasus penggunaan 330
Fanout untuk fungsi Lambda 342
Prasyarat 342
Berlangganan fungsi ke topik 343
Fanout ke antrean Amazon SQS 344
Berlangganan antrean ke topik 344
Contoh (AWS CloudFormation) 352
Fanout ke HTTP (S) titik akhir 360
Melangganken titik akhir ke topik 362
Memverifikasi tanda tangan pesan 370
Menguraikan format pesan 374
Fanout ke Alur Fork Peristiwa AWS 384
Bagaimana Alur Fork Peristiwa AWS bekerja 385
Men-deploy Alur Fork Peristiwa AWS 389
Men-deploy dan menguji Alur Fork Peristiwa AWS 390
Berlangganan alur peristiwa untuk topik 400

Menggunakan EventBridge Scheduler 410
Mengatur peran eksekusi 410
Buat jadwal 411
Sumber daya terkait 416
Olahpesan aplikasi-ke-orang (application-to-person/A2P) 417
Olahpesan teks seluler (SMS) 417
Sandbox SMS 418
Identitas asal 423
Meminta dukungan SMS 508
Mengatur preferensi SMS 524
Mengirim pesan SMS 531
Memantau aktivitas SMS 554
Mengelola langganan SMS 564
Negara dan wilayah yang didukung 595
Praktik terbaik SMS 615
Notifikasi push seluler 632
Cara kerja notifikasi pengguna 632
Gambaran umum proses notifikasi pengguna 633
Menyiapkan aplikasi seluler 634
Mengirim notifikasi push seluler 653
Atribut aplikasi seluler 668
Peristiwa aplikasi seluler 672
Tindakan API push seluler 675
Kesalahan API push seluler 677
TTL push seluler 688
Wilayah yang Didukung 691
Praktik terbaik pemberitahuan push seluler 692
Pemberitahuan email 693
AWS Management Console 693
AWS SDK 694
Contoh kode 725
Tindakan 735
`CheckIfPhoneNumberIsOptedOut` 736
`ConfirmSubscription` 743
`CreateTopic` 749
`DeleteTopic` 762

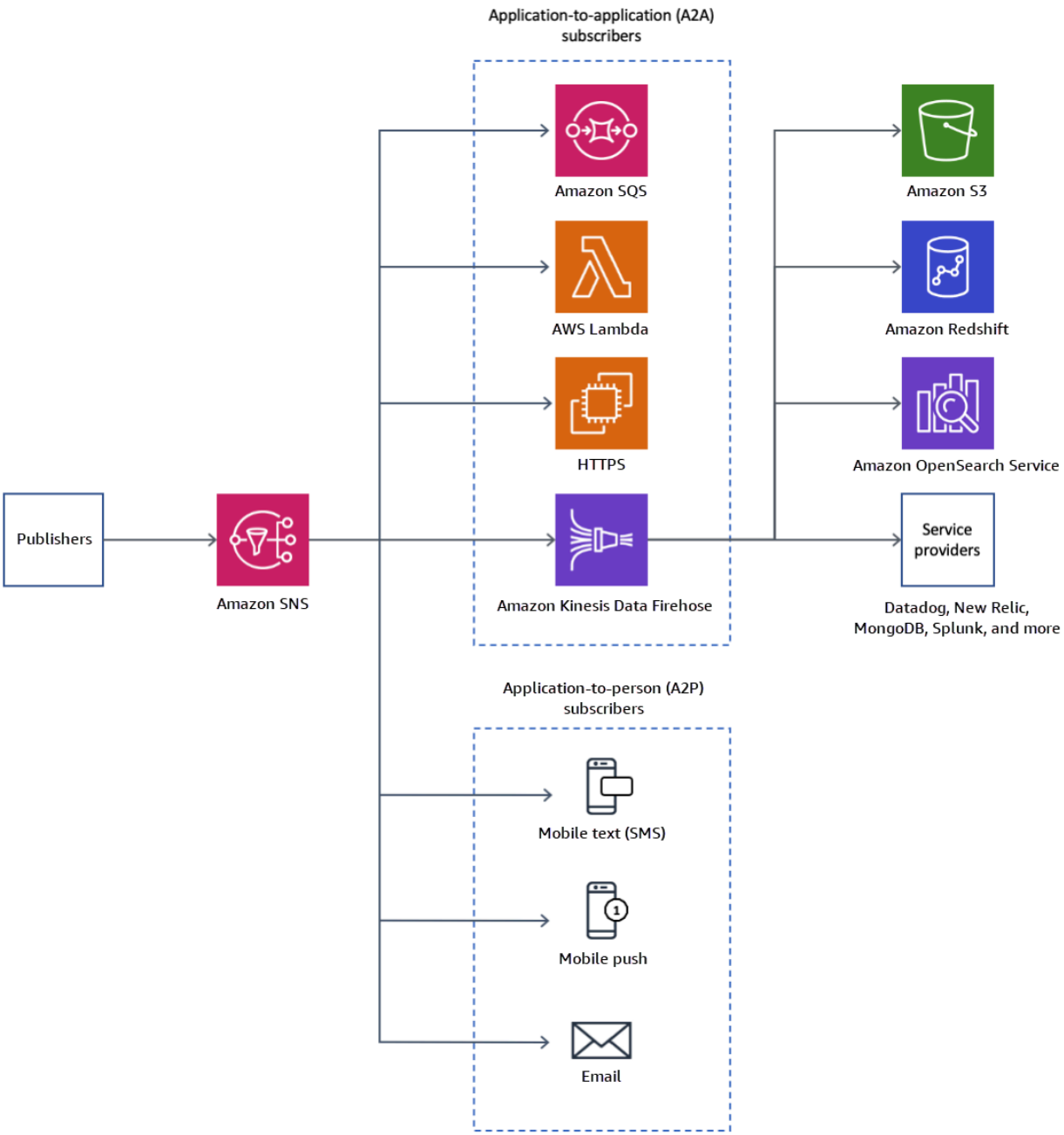
GetSMSAttributes ... 772
GetTopicAttributes ... 779
ListPhoneNumbersOptedOut ... 789
ListSubscriptions ... 792
ListTopics ... 804
Publish ... 817
SetSMSAttributes ... 839
SetSubscriptionAttributes ... 845
SetSubscriptionAttributesRedrivePolicy ... 849
SetTopicAttributes ... 850
Subscribe ... 859
TagResource ... 888
Unsubscribe ... 892
Skenario ... 901
Buat titik akhir platform untuk pemberitahuan push ... 901
Membuat dan mempublikasikan ke topik FIFO ... 904
Publikasikan pesan SMS ke suatu topik ... 916
Publikasikan pesan besar ... 923
Publikasikan pesan teks SMS ... 926
Publikasikan pesan ke antrian ... 934
Contoh nirserver ... 997
Memanggil fungsi Lambda dari pemicu Amazon SNS ... 998
Contoh lintas layanan ... 1007
Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB ... 1008
Membangun aplikasi Amazon SNS ... 1009
Membuat aplikasi nirserver untuk mengelola foto ... 1011
Membuat aplikasi penjelajah Amazon Textract ... 1015
Mendeteksi orang dan objek dalam video ... 1016
Publikasikan pesan ke antrian ... 1017
Menggunakan API Gateway untuk menginvokasi fungsi Lambda ... 1018
Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda ... 1020
Keamanan ... 1022
Perlindungan data ... 1022
Enkripsi data ... 1024
Privasi lalu lintas jaringan Internet ... 1042
Keamanan Perlindungan Data Pesan ... 1058

Identity and access management 1059
Audiens 1059
Mengautentikasi dengan identitas 1060
Mengelola akses menggunakan kebijakan 1064
Pengendalian akses 1066
Gambaran umum 1067
Bagaimana Amazon Simple Notification Service bekerja dengan IAM 1087
Tindakan kebijakan 1088
Sumber daya kebijakan 1089
Kunci persyaratan kebijakan 1089
ACL 1090
ABAC 1091
Kredensial sementara 1091
Izin-izin pengguna utama 1092
Peran layanan 1092
Peran terkait layanan 1093
Contoh kebijakan berbasis identitas 1093
Kebijakan berbasis identitas 1097
Kebijakan berbasis sumber daya 1098
Menggunakan kebijakan berbasis identitas 1098
Menggunakan kredensial sementara 1106
Referensi izin API 1107
Pencatatan dan pemantauan 1111
Logging panggilan API menggunakan CloudTrail 1112
Memantau topik menggunakan CloudWatch 1121
Validasi Kepatuhan 1137
Ketangguhan 1138
Keamanan infrastruktur 1138
Praktik terbaik 1139
Praktik terbaik pencegahan 1139
Pemecahan Masalah 1144
Memecahkan masalah topik menggunakan X-Ray 1144
Penelusuran aktif 1144
Izin 1145
Mengaktifkan penelusuran aktif 1145
Mengaktifkan penelusuran aktif pada topik Amazon SNS (SDK) AWS 1146

Mengaktifkan penelusuran aktif pada topik Amazon SNS (CLI) AWS 1147
Mengaktifkan penelusuran aktif pada topik Amazon SNS () AWS CloudFormation 1147
Memverifikasi penelusuran aktif diaktifkan 1147
Pengujian 1148
Riwayat dokumentasi 1150
AWSGlosarium 1159
..... mclx

Apa itu Amazon SNS?

Amazon Simple Notification Service (Amazon SNS) 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 topik SNS dan menerima pesan yang dipublikasikan menggunakan jenis endpoint yang didukung, seperti Amazon Data Firehose, Amazon SQS, HTTP, email AWS Lambda, notifikasi push seluler, dan pesan teks seluler (SMS).



Topik

- Fitur dan kemampuan
- Layanan terkait
- Mengakses Amazon SNS

- Harga Amazon SNS
- Skenario Amazon SNS umum
- Menggunakan Amazon SNS dengan SDK AWS

Fitur dan kemampuan

Amazon SNS menyediakan fitur dan kemampuan berikut:

- Sebuah pplication-to-application pesan

  pplication-to-application Pesan mendukung pelanggan seperti aliran pengiriman Amazon Data Firehose, fungsi Lambda, antrian Amazon SQS, titik akhir HTTP/S, dan Pipelines Event Fork. AWS Untuk informasi selengkapnya, lihat Pesan A pplication-to-application (A2A).

- pplication-to-person Pemberitahuan

  pplication-to-person Notifikasi memberikan pemberitahuan pengguna kepada pelanggan seperti aplikasi seluler, nomor ponsel, dan alamat email. Untuk informasi selengkapnya, lihat Olahpesan aplikasi-ke-orang (application-to-person/A2P).

- Topik standar dan FIFO

  Gunakan topik FIFO untuk memastikan urutan pesan yang ketat, menentukan grup pesan, dan mencegah duplikasi pesan. Anda dapat menggunakan FIFO dan antrian standar untuk berlangganan topik FIFO. Untuk informasi selengkapnya, lihat Pengurutan pesan dan deduplikasi (topik FIFO).

  Gunakan topik standar bila urutan pengiriman pesan dan kemungkinan duplikasi pesan tidak kritis. Semua protokol pengiriman yang didukung dapat berlangganan topik standar.

- 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 langganan tidak tersedia, Amazon SNS menjalankan kebijakan pengiriman ulang.
  - 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 SNS dengan berbagai cara termasuk berlangganan aliran pengiriman Firehose ke topik SNS, 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, topik Amazon SNS FIFO mendukung pengarsipan dan pemutaran ulang pesan 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 pesan dan pemutaran ulang untuk topik FIFO.

- Atribut pesan

  Atribut pesan memungkinkan Anda memberikan metadata arbitrer tentang pesan tersebut. the section called “Atribut pesan”.

- 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 topik Amazon SNS, menggunakan kunci enkripsi yang disediakan oleh. AWS KMS Untuk informasi selengkapnya, lihat the section called “Enkripsi diam”.

  Anda juga dapat membangun koneksi privat antara Amazon SNS dan virtual private cloud (VPC) Anda. Untuk informasi selengkapnya, lihat the section called “Privasi lalu lintas jaringan Internet”.

Layanan terkait

Anda dapat menggunakan layanan berikut dengan Amazon SNS:

- Amazon SQS menawarkan antrean yang di-hosting yang aman, tahan lama, dan tersedia yang memungkinkan Anda untuk mengintegrasikan dan memisahkan sistem dan komponen perangkat lunak terdistribusi. Amazon SQS terkait dengan Amazon SNS dengan cara berikut:
  - Amazon SNS menyediakan antrean surat mati yang didukung oleh Amazon SQS untuk pesan yang tidak dapat terkirim.
  - Anda dapat berlangganan antrian Amazon SQS ke topik Amazon SNS.
  - Anda dapat berlangganan antrian Amazon SQS FIFO atau antrianstandar ke topik Amazon SNS FIFO. Hanya antrian Amazon SQS FIFO 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 topik SNS.
- AWS Identity and Access Management (IAM) membantu Anda mengontrol akses ke AWS sumber daya untuk pengguna Anda dengan aman. Gunakan IAM untuk mengontrol siapa saja yang dapat menggunakan topik Amazon SNS (autentikasi), 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 template yang menjelaskan AWS sumber daya yang Anda inginkan, termasuk topik dan langganan Amazon SNS. AWS CloudFormation mengurus penyediaan dan konfigurasi sumber daya tersebut untuk Anda. Untuk informasi selengkapnya, lihat Panduan Pengguna AWS CloudFormation.

Mengakses Amazon SNS

Anda dapat mengonfigurasi dan mengelola topik dan langganan SNS menggunakan konsol Amazon SNS, alat baris perintah, atau SDK. AWS

- Konsol Amazon SNS menyediakan antarmuka pengguna yang nyaman untuk membuat topik dan langganan, mengirim dan menerima pesan, dan 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 SDK dalam berbagai bahasa. Untuk informasi selengkapnya, lihat SDK dan Kit Alat.

Harga Amazon SNS

Tidak ada biaya yang harus dibayar di muka di Amazon SNS. Anda membayar berdasarkan jumlah pesan yang Anda terbitkan, jumlah notifikasi yang Anda kirimkan, dan panggilan API tambahan untuk mengelola topik dan langganan. Harga pengiriman bervariasi berdasarkan jenis titik akhir. Anda dapat memulai secara gratis dengan Amazon SNS tingkat gratis.

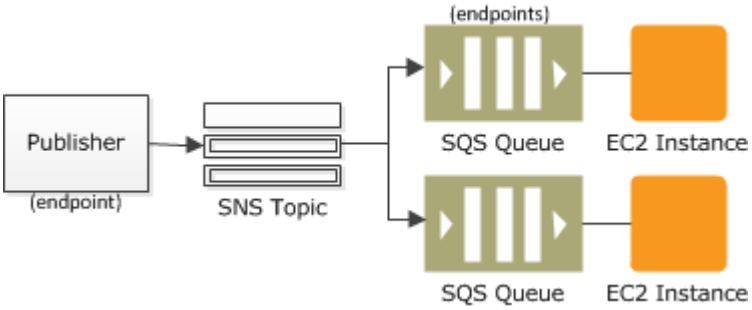
Untuk informasi lebih lanjut, lihat harga Amazon SNS.

Skenario Amazon SNS umum

Integrasi aplikasi

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

Misalnya, Anda dapat mengembangkan aplikasi yang menerbitkan pesan ke topik SNS setiap kali pesanan produk dibuat. Kemudian, antrean SQS yang berlangganan ke topik SNS menerima notifikasi identik untuk pesanan baru. Instans server Amazon Elastic Compute Cloud (Amazon EC2) yang melekat pada salah satu antrean SQS dapat menangani pemrosesan atau pemenuhan pesanan. Dan Anda dapat melampirkan instans server Amazon EC2 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 antrean SQS lain ke topik SNS yang sama untuk pesanan masuk baru. Kemudian, dengan melampirkan antrean SQS baru ini untuk 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
- Fanout untuk fungsi Lambda
- Fanout ke antrean Amazon SQS
- Fanout ke HTTP (S) titik akhir
- Komputasi Berbasis Acara dengan Amazon SNS AWS dan Layanan 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 notifikasi ini ke pengguna tertentu melalui SMS dan email. Misalnya, Anda dapat menerima pemberitahuan langsung saat peristiwa terjadi, seperti perubahan spesifik pada grup Auto Scaling Amazon EC2, file baru yang diunggah ke bucket Amazon S3, atau ambang metrik yang dilanggar di Amazon. CloudWatch Untuk informasi selengkapnya, lihat Menyiapkan notifikasi Amazon SNS di CloudWatch Panduan Pengguna Amazon.

Notifikasi pengguna

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

Notifikasi push seluler

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

Pesan notifikasi dapat menyertakan tautan untuk mengunduh dan menginstal pembaruan. Untuk informasi selengkapnya tentang penggunaan Amazon SNS untuk mengirim pesan notifikasi, lihat Notifikasi push seluler.

Menggunakan Amazon SNS dengan SDK AWS

AWS kit pengembangan perangkat lunak (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	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 Amazon SNS, lihat. Contoh kode untuk Amazon SNS menggunakan SDK AWS

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

Sumber kejadian dan tujuan Amazon SNS

Amazon SNS dapat menerima pemberitahuan berbasis-peristiwa dari banyak sumber AWS dan mengirimkan pemberitahuan ke tujuan aplikasi-ke-aplikasi (A2A) dan aplikasi-ke-orang (A2P). Bagian ini mencantumkan sumber kejadian dan tujuan yang didukung, dan menyediakan tautan untuk informasi lebih lanjut.

Topik

- Sumber kejadian Amazon SNS
- Tujuan kejadian Amazon SNS

Sumber kejadian Amazon SNS

Halaman ini berisi daftar layanan AWS yang dapat mempublikasikan kejadian ke topik Amazon SNS, yang dikelompokkan berdasarkan Kategori produk AWS.

> **Note**
>
> Amazon SNS memperkenalkan Topik FIFO pada bulan Oktober, 2020. Saat ini, sebagian besar layanan AWS mendukung pengiriman kejadian ke topik standar saja.

Layanan analitik

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Athena – Memungkinkan Anda untuk menganalisis data di Amazon S3 menggunakan SQL standar.	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 lebih lanjut, lihat SnsAlarm dalam Panduan PengembangAWS Data Pipeline.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
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

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.
AWS Step Functions – Memungkinkan Anda menggabungkan fungsi AWS Lambda dan layanan AWS lainnya untuk membangun aplikasi bisnis penting.	Menerima notifikasi kejadian Step Functions . Untuk informasi selengkapnya, lihat Panggil Amazon SNS dengan Step Functions di Panduan Developer AWS Step Functions.

Layanan manajemen penagihan & biaya

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 topik Amazon SNS untuk pemberitahuan anggaran

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	• Menyiapkan notifikasi • Mendeteksi pengeluaran yang tidak biasa dengan Deteksi Anomali AWS Biaya

Layanan aplikasi bisnis

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 kejadian Amazon Chime SDK di Panduan Developer Amazon Chime.

Layanan komputasi

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon EC2 Auto Scaling – Membantu Anda mendapatkan jumlah instans Amazon Elastic Compute Cloud (Amazon EC2) yang tersedia dengan benar untuk menangani beban aplikasi Anda.	Menerima notifikasi saat Auto Scaling meluncurkan atau mengakhiri instans Amazon EC2 di grup Auto Scaling. Untuk informasi selengkapnya, lihat Mendapatkan notifikasi Amazon SNS ketika grup Auto Scaling Anda menskalakan di Panduan Pengguna Amazon EC2 Auto Scaling.
EC2 Image Builder - Membantu mengotoma tiskan pembuatan, pengelolaan, dan penyebara n gambar yang disesuaikan, aman, up-to-date dan server yang telah diinstal sebelumnya dan dikonfigurasi sebelumnya dengan perangkat	Menerima notifikasi saat membangun selesai. Untuk informasi selengkapnya, lihat Pelacakan citra server terbaru dalam alur EC2 Image Builder pada Blog Komputasi AWS.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
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 Notifikasi lingkungan Elastic Beanstalk dengan Amazon SNS di Panduan Developer AWS Elastic Beanstalk.
AWS Lambda – Memungkinkan Anda menjalankan kode tanpa penyediaan atau pengelolaan server.	Menerima data output fungsi dengan mengatur topik SNS sebagai antrean surat mati Lambda atau tujuan Lambda. Untuk informasi selengkap nya, lihat Invokasi asinkron di Panduan Developer AWS Lambda.
Amazon Lightsail – Membantu developer mulai menggunakan AWS 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 selengkap nya, lihat Menambahkan kontak notifikasi di Amazon Lightsail di Panduan Developer Amazon Lightsail.

Layanan kontainer

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon EKS Distro – Memungkinkan Anda membuat klaster yang handal dan aman di mana pun aplikasi Anda di-deploy.	Lacak pembaruan dan patch keamanan untuk klaster yang dibuat dengan Amazon EKS Distro. Untuk informasi selengkapnya, lihat Memperkenalkan Amazon EKS Distro - distribusi Kubernetes sumber terbuka yang digunakan oleh Amazon EKS.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Elastic Container Service (Amazon ECS) – Memungkinkan Anda untuk menjalankan, menghentikan, dan mengelola kontainer pada sebuah klaster.	Menerima notifikasi ketika AMI baru yang dioptimalkan oleh Amazon ECS-tersedia. Untuk informasi selengkapnya, lihat Berlangganan notifikasi pembaruan AMI yang dioptimalkan oleh Amazon ECS di Panduan Developer Amazon Elastic Container Service.

Layanan keterlibatan pelanggan

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 dengan Amazon Connect di Panduan Administrator Amazon Connect.
Amazon Pinpoint – Membantu Anda berinteraksi dengan pelanggan Anda dengan mengirimkan email, SMS dan pesan suara, serta notifikasi push kepada pelanggan.	Konfigurasikan SMS dua arah, yang memungkinkan Anda menerima pesan dari pelanggan Anda. Untuk informasi selengkapnya, lihat Menggunakan olahpesan SMS dua arah di Amazon Pinpoint di Panduan Pengguna Amazon Pinpoint.
Amazon Simple Email Service (Amazon SES) – Menyediakan cara yang 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 notifikasi Amazon SNS untuk Amazon SES di Panduan Developer Amazon Simple Email Service.

Layanan basis data

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Database Migration Service – Memigrasi data dari basis data on-premise ke dalam AWS Cloud.	Menerima notifikasi saat peristiwa AWS DMS terjadi; misalnya, ketika instans 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 – Menyediakan performa yang cepat dan dapat diprediksi dengan skalabilitas tanpa hambatan dalam layanan basis data NoSQL terkelola penuh ini.	Menerima notifikasi ketika kejadian pemelihar aan terjadi. Untuk informasi selengkapnya, lihat Mengkustomisasi pengaturan klaster DAX di Panduan Developer Amazon DynamoDB.
Amazon ElastiCache - Menyediakan cache dalam memori berkinerja tinggi, dapat diubah ukurannya, dan hemat biaya, sekaligus menghilangkan kompleksitas yang terkait dengan penerapan dan pengelolaan lingkungan cache terdistribusi.	Menerima notifikasi ketika kejadian penting terjadi. Untuk informasi selengkapnya, lihat Pemberitahuan acara dan Amazon SNS di Panduan Pengguna Amazon ElastiCache untuk Memcache.
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 notifikas i peristiwa Amazon Redshift di Panduan Manajemen Pergeseran Merah Amazon.
Amazon Relational Database Service – Memudahkan pengaturan, pengoperasian, dan penskalaan basis data relasional di AWS Cloud.	Menerima notifikasi kejadian Amazon RDS. Untuk informasi selengkapnya, lihat Menggunakan notifikasi kejadian Amazon RDS di Panduan Pengguna Amazon RDS.

Layanan alat developer

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 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: Buat pemicu AWS CodeCommit untuk topik Amazon SNS di Panduan Pengguna AWS CodeCommit.
AWS CodeDeploy – Mengotomatisasi deployment aplikasi ke instans Amazon EC2, instans on-premises, fungsi Lambda nirserver, 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 performa 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 diAWS.	Menerima notifikasi tentang kejadian yang terjadi di sumber daya yang Anda gunakan. Untuk informasi selengkapnya, lihat Mengkonfi

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	gurasi topik Amazon SNS untuk notifikasi di Panduan Pengguna Konsol Alat Developer.

Layanan web & seluler front-end

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Pinpoint – Membantu Anda berinteraksi dengan pelanggan Anda dengan mengirimkan email, SMS dan pesan suara, serta notifikasi push kepada pelanggan.	Konfigurasikan SMS dua arah, yang memungkinkan Anda menerima pesan dari pelanggan Anda. Untuk informasi selengkapnya, lihat Menggunakan olahpesan SMS dua arah di Amazon Pinpoint di Panduan Pengguna Amazon Pinpoint.

Layanan pengembangan game

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 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

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS IoT Core – Menyediakan layanan cloud yang menghubungkan perangkat IoT Anda ke perangkat lain dan layanan AWS Cloud.	Menerima notifikasi dari kejadian AWS IoT Core. Untuk informasi selengkapnya, lihat Membuat aturan Amazon SNS di Panduan Developer AWS IoT.
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 deteksi AWS IoT Device Defender di Panduan Developer AWS IoT.
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 notifikasi dari kejadian AWS IoT Events. Untuk informasi selengkapnya, lihat Amazon Simple Notification Service di Panduan Developer AWS IoT Events.
AWS IoT Greengrass – Memperluas AWS ke perangkat fisik sehingga mereka dapat bertindak secara lokal pada data yang mereka hasilkan, sementara masih menggunakan cloud untuk manajemen, analitik, dan penyimpanan yang tahan lama.	Menerima notifikasi dari kejadian AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Konektor SNS dalam Panduan Developer AWS IoT Greengrass Version 1.

Layanan machine learning

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon CodeGuru — Mengumpulkan data kinerja runtime dari aplikasi live Anda, dan memberikan rekomendasi yang dapat	Menerima notifikasi saat anomali terjadi. Untuk informasi selengkapnya, lihat Bekerja dengan anomali dan laporan rekomendasi di CodeGuru Panduan Pengguna Amazon.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
membantu Anda menyempurnakan performa aplikasi.	
Amazon DevOps Guru — Menghasilkan wawasan operasional menggunakan pembelajaran mesin untuk membantu Anda meningkatkan kinerja aplikasi operasional Anda.	Wawasan dan konfirmasi ke depan. 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.
Amazon Lookout for Metrics – Menemukan anomali dalam data Anda, menentukan akar penyebabnya, dan memungkinkan Anda untuk segera mengambil tindakan.	Menerima notifikasi anomali. Untuk informasi selengkapnya, lihat Menggunakan Amazon SNS dengan Lookout for Metrics di Panduan Developer 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

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 Cloud. AWS	Mengirimkan notifikasi ke ruang obrolan. Untuk informasi selengkapnya, lihat Menyiapkan AWS Chatbot di Panduan Administrator AWS Chatbot.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS CloudFormation – Memungkinkan Anda untuk membuat dan menyediakan deploymen t infrastruktur AWS dengan cara yang dapat diprediksi dan berulang kali.	Menerima notifikasi saat tumpukan dibuat dan diperbarui. Untuk informasi selengkapnya, lihat Pengaturan opsi tumpukan AWS CloudForm ation di Panduan Pengguna AWS CloudForm ation.
AWS CloudTrail – Menyediakan riwayat kejadian dari kegiatan Akun AWS Anda.	Terima pemberitahuan saat CloudTrail memublikasikan file log baru ke bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi notifikasi Amazon SNS di CloudTrail Panduan Pengguna AWS CloudTrai l.
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 – Menyediakan tampilan detail dari konfigurasi sumber daya AWS dalam Akun AWS Anda.	Menerima notifikasi saat sumber daya diperbarui, atau saat AWS Config mengevalu asi aturan kustom atau terkelola dengan sumber daya Anda. Untuk informasi selengkap nya, lihat Notifikasi yang dikirim oleh AWS Config ke topik SNS dan Contoh notifikas i perubahan item konfigurasi di Panduan Developer AWS Config.
AWS Control Tower – Memungkinkan Anda untuk menyiapkan dan menata lingkungan AWS multi-akun yang aman dan patuh.	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 Anda dari vendor perangkat lunak secara terpusat di seluruh AWS dan lingkungan on-premises Anda.	Menerima notifikasi dan pemberitahuan License Manager. Untuk informasi selengkapnya, 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 menampilkan dan mengontrol infrastruktur Anda di AWS.	Menerima notifikasi tentang status perintah. Untuk informasi selengkapnya, lihat Pemantauan perubahan status Systems Manager menggunakan notifikasi Amazon SNS di Panduan Pengguna AWS Systems Manager.

Layanan media

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Elastic Transcoder – Memungkinkan Anda mengkonversi file media yang Anda simpan di Amazon S3 ke file media dalam format yang dibutuhkan oleh perangkat pemutaran konsumen.	Menerima notifikasi saat status tugas berubah. Untuk informasi selengkapnya, lihat Notifikasi status tugas di Panduan Developer Amazon Elastic Transcoder.

Lyanan migrasi & transfer

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 on-premises Anda.	Menerima notifikasi kejadian melalui AWS CloudTrail. Untuk informasi selengkapnya, lihat Mencatat Log panggilan API Application Discovery Service dengan AWS CloudTrail di Panduan Pengguna Application Discovery Service.
AWS Database Migration Service – Memigrasi data dari basis data on-premise ke dalam AWS Cloud.	Menerima notifikasi saat peristiwa AWS DMS terjadi; misalnya, ketika instans 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 panduan berikut: • Notifikasi Snowball di Panduan Pengguna AWS Snowball • Langkah 5: Pilih preferensi notifikasi Anda di Panduan Developer Snowball Edge AWS • Langkah 5: Pilih preferensi notifikasi Anda di Panduan Pengguna Snowcone AWS

Layanan jaringan dan pengiriman konten

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon API Gateway — Memungkinkan Anda membuat dan menerapkan REST dan WebSocket API Anda sendiri dalam skala apa pun.	Menerima pesan yang dikirim ke titik akhir API Gateway. Untuk informasi selengkapnya, lihat Tutorial: Membangun API REST API Gateway dengan Integrasi AWS di Panduan Developer API Gateway.
Amazon CloudFront - Mempercepat distribusi konten web statis dan dinamis Anda, seperti.html, .css, .php, gambar, dan file media.	Menerima pemberitahuan saat alarm berdasarkan CloudFront metrik yang ditentukan terjadi. Untuk informasi selengkapnya, lihat Menyetel alarm untuk menerima notifikasi di Panduan CloudFront Pengembang Amazon.
AWS Direct Connect – Menautkan jaringan internal Anda ke lokasi AWS Direct Connect 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 – Mendistribusikan lalu lintas masuk secara otomatis di beberapa target, seperti instans Amazon EC2, kontainer, dan alamat IP, di beberapa 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.
Amazon Route 53 – Menyediakan pendaftaran domain, perutean DNS, dan pemeriksaan kondisi.	Menerima notifikasi ketika status pemeriksaan kondisi tidak sehat. Untuk informasi selengkapnya, lihat Untuk menerima notifikasi Amazon SNS ketika status pemeriksaan kondisi tidak sehat (konsol) di Panduan Developer Amazon Route 53.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Virtual Private Cloud (Amazon VPC) – Memungkinkan Anda untuk meluncurkan sumber daya AWS ke dalam 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 Pengguna Amazon VPC.

Layanan keamanan, identitas, dan kepatuhan

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Directory Service – Menyediakan beberapa cara untuk menggunakan Direktori Aktif (AD) Microsoft dengan layanan AWS.	Menerima pesan email atau teks (SMS) ketika status direktori Anda berubah. Untuk informasi selengkapnya, lihat Mengkonfigurasi notifikasi status direktori di Panduan Administrasi AWS Directory Service.
Amazon GuardDuty - Menyediakan pemantauan keamanan berkelanjutan untuk membantu mengidentifikasi aktivitas yang tidak terduga dan berpotensi tidak sah atau berbahaya di AWS lingkungan Anda.	Menerima notifikasi tentang jenis temuan yang baru dirilis, pembaruan untuk jenis temuan yang ada, dan perubahan fungsionalitas lainnya. Untuk informasi selengkapnya, lihat Berlangganan GuardDuty pengumuman topik SNS di Panduan Pengguna Amazon GuardDuty .
Amazon Inspector – Menguji aksesibilitas jaringan instans Amazon EC2 Anda dan status keamanan aplikasi Anda yang berjalan pada instans tersebut.	Menerima notifikasi untuk kejadian Amazon Inspector. Untuk informasi selengkapnya, lihat Menyiapkan topik SNS 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

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	an, diedit, atau dihentikan. Untuk informasi selengkapnya, lihat Berlangganan AWS Security Hub pengumuman dengan Amazon SNS.

Layanan nirserver

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon DynamoDB – Menyediakan performa yang cepat dan dapat diprediksi dengan skalabilitas tanpa hambatan dalam layanan basis data NoSQL terkelola penuh ini.	Menerima notifikasi ketika kejadian pemelihar aan terjadi. Untuk informasi selengkapnya, lihat Mengkustomisasi pengaturan klaster DAX di Panduan Developer Amazon DynamoDB.
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.
AWS Lambda – Memungkinkan Anda menjalankan kode tanpa penyediaan atau pengelolaan server.	Menerima data output fungsi dengan mengatur topik SNS sebagai antrean surat mati Lambda atau tujuan Lambda. Untuk informasi selengkap nya, lihat Invokasi asinkron di Panduan Developer AWS Lambda.

Layanan penyimpanan

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Backup – Membantu Anda memusatka n dan mengotomatisasi pencadangan data di seluruh layanan AWS di cloud dan on premise	Menerima notifikasi dari kejadian AWS Backup. Untuk informasi selengkapnya, lihat Menggunakan Amazon SNS untuk melacak

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	kejadian AWS Backup di Panduan Developer AWS Backup.
Amazon Elastic File System – Menyediakan penyimpanan file untuk instans Amazon EC2 Anda.	Menerima notifikasi alarm yang telah Anda buat untuk kejadian Amazon EFS. Untuk informasi selengkapnya, lihat Alat pemantauan otomatisasi dalam Panduan Pengguna Amazon Elastic File System.
Amazon S3 Glacier – Menyediakan penyimpanan untuk data yang jarang digunakan.	Mengatur konfigurasi notifikasi di vault sehingga ketika tugas selesai, pesan akan 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 (topik SNS atau antrean SQS) dan Memantau kemajuan dengan metrik replikasi dan notifikasi peristiwa Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
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 panduan berikut: • Notifikasi Snowball di Panduan Pengguna AWS Snowball • Langkah 5: Pilih preferensi notifikasi Anda di Panduan Developer Snowball Edge AWS • Langkah 5: Pilih preferensi notifikasi Anda di Panduan Pengguna Snowcone AWS

Sumber kejadian tambahan

Sumber	Manfaat menggunakan dengan Amazon SNS
AWSPembaruan Fitur Harian	Terima informasi terperinci tepat waktu tentang rilis dan pembaruan AWS melalui topik Amazon SNS. Rilis ini mencakupWilayah AWS,Layanan AWS, titik akhir Amazon VPC, Layanan AWS terintegrasi dengan AWS Service Quotas, jenis instans Amazon EC2, jenis instans Amazon, jenis instans Amazon SageMaker Nimble Studio, versi mesin database Amazon RDS, dan versi Amazon MSK Apache Kafka. Untuk informasi selengkapnya, lihat Berlangganan Pembaruan Fitur AWS Harian melalui Amazon SNS di Blog AWSBerita.
AWSRentang alamat IP	Terima pemberitahuan perubahan rentang AWS IP melalui topik Amazon SNS. Untuk informasi selengkapnya, lihat pemberitahuan rentang alamat AWS IP di Referensi Umum

Sumber	Manfaat menggunakan dengan Amazon SNS
	Amazon Web Services, dan Berlangganan Perubahan Alamat IP AWS Publik melalui Amazon SNS di Blog AWSBerita.

Untuk informasi selengkapnya tentang komputasi berbasis kejadian, lihat sumber berikut:

- Apa itu Arsitektur Event-Driven?
- Komputasi Berbasis Kejadian dengan Amazon SNS dan Komputasi, Penyimpanan, Basis Data, dan Layanan Jaringan AWS pada Blog Komputasi AWS
- Memperkaya Arsitektur Berbasis Kejadian dengan Alur Event Fork AWS pada Blog Komputasi AWS

Tujuan kejadian Amazon SNS

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 Topik FIFO pada bulan Oktober, 2020. Saat ini, sebagian besar layanan AWS mendukung penerimaan kejadian dari topik standar SNS saja. Amazon SQS mendukung penerimaan kejadian baik dari standar SNS dan topik FIFO.

Tujuan A2A

Tujuan kejadian	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

Tujuan kejadian	Manfaat menggunakan dengan Amazon SNS
	Redshift, dan OpenSearch Amazon Service OpenSearch (Service), atau ke tujuan pihak 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 Fanout untuk fungsi Lambda.
Amazon SQS	Mengirimkan kejadian ke antrean untuk tujuan integrasi aplikasi. Untuk informasi selengkap nya, lihat Fanout ke antrean Amazon SQS.
AWS Event Fork Pipelines	Mengirimkan kejadian ke pencadangan dan penyimpanan kejadian, pencarian dan analitik kejadian, atau alur putar ulang kejadian. Untuk informasi selengkapnya, lihat Fanout ke Alur Fork Peristiwa AWS.
HTTP/S	Mengirimkan kejadian ke webhooks eksternal. Untuk informasi selengkapnya, lihat Fanout ke HTTP (S) titik akhir.

Tujuan A2P

Tujuan kejadian	Manfaat menggunakan dengan Amazon SNS
SMS	Mengirimkan kejadian ke ponsel sebagai pesan teks. Untuk informasi selengkapnya, lihat Olahpesan teks seluler (SMS).

Tujuan kejadian	Manfaat menggunakan dengan Amazon SNS
Email	Mengirimkan kejadian ke kotak masuk sebagai pesan email. Untuk informasi selengkapnya, lihat Pemberitahuan email.
Titik akhir platform	Mengirimkan kejadian ke ponsel sebagai notifikasi push asli. Untuk informasi selengkap nya, lihat Notifikasi push seluler.
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 baik kejadian AWS asli maupun kejadian kustom ke aplikasi obrolan:
>
> • Kejadian AWS asli – Anda dapat menggunakan AWS Chatbot untuk mengirim kejadian AWS asli, melalui topik Amazon SNS, ke Amazon Chime dan Slack. Kumpulan AWS acara

asli yang didukung mencakup acara dariAWS Billing and Cost Management,,AWS Health, Amazon AWS CloudFormation CloudWatch, dan banyak lagi. Untuk informasi lebih lanjut, lihat Menggunakan AWS Chatbot dengan layanan lain di Panduan Administrator AWS Chatbot.

- Kejadian kustom – Anda juga dapat mengirim kejadian kustom Anda, melalui topik Amazon SNS, ke Amazon Chime, Slack, dan Microsoft Teams. Untuk melakukannya, Anda perlu mempublikasikan kejadian kustom ke topik SNS, yang mengirimkan kejadian ke fungsi Lambda berlangganan. Fungsi Lambda kemudian menggunakan webhook aplikasi obrolan untuk mengirimkan kejadian ke penerima. Untuk informasi lebih lanjut, lihat Bagaimana cara saya menggunakan webhooks untuk mempublikasikan pesan Amazon SNS ke Amazon Chime, Slack, atau Microsoft Teams?

Menyiapkan akses untuk Amazon SNS

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

Topik

- Langkah 1: Buat Akun AWS dan pengguna IAM
- Langkah selanjutnya

Langkah 1: Buat Akun AWS dan pengguna IAM

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/signup.
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. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ 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. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

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

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, 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 Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

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

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, 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 Amazon SNS, mulailah dengan membuat topik, membuat langganan untuk topik, menerbitkan pesan ke topik, dan menghapus langganan dan topik.

Memulai dengan Amazon SNS

Bagian ini membantu Anda membiasakan diri dengan Amazon SNS dengan menunjukkan cara mengelola topik, langganan, dan pesan menggunakan konsol Amazon SNS.

Topik

- Prasyarat
- Langkah 1: Buat topik
- Langkah 2: Buat langganan topik
- Langkah 3: Publikasikan pesan ke topik
- Langkah 4: Hapus langganan dan topik
- Langkah selanjutnya

Prasyarat

Sebelum memulai, selesaikan langkah-langkah di Menyiapkan akses untuk Amazon SNS.

Langkah 1: Buat topik

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi kiri, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih Create topic (Buat topik).
4. Secara default, konsol membuat topik FIFO. Pilih Standard (Standar).
5. Di bagian Detail, masukkan Nama untuk topik tersebut, seperti *`MyTopic`*.
6. Gulir ke akhir formulir dan pilih Create topic (Buat topik).

   Konsol membuka halaman Detail topik baru.

Langkah 2: Buat langganan topik

1. Di panel navigasi kiri, pilih Subscriptions (Langganan).
2. Di halaman Subscriptions (Langganan), pilih Create subscription (Buat langganan).

3. Di halaman Create subscription (Buat langganan), pilih bidang Topic ARN (ARN topik) untuk melihat daftar topik di Akun AWS Anda.
4. Pilih topik yang Anda buat di langkah sebelumnya.
5. Untuk Protocol (Protokol), pilih Email.
6. Untuk Titik Akhir, masukkan alamat email yang dapat Anda gunakan untuk menerima pemberitahuan.
7. Pilih Create Subscription (Buat langganan).

   Konsol membuka halaman Detail langganan baru.
8. Periksa kotak masuk email Anda dan pilih Konfirmasi langganan dalam email dari Notifikasi AWS. ID pengirim biasanya "no-reply@sns.amazonaws.com".
9. Amazon SNS membuka browser web Anda dan menampilkan konfirmasi berlangganan dengan ID langganan Anda.

Langkah 3: Publikasikan pesan ke topik

1. Di panel navigasi kiri, pilih Topics (Topik).
2. Pada halaman Topics (Topik), pilih topik yang Anda buat sebelumnya, dan kemudian pilih Publish message (Publikasikan pesan).

   Konsol membuka halaman Publikasikan pesan ke topik.
3. (Opsional) Di bagian Detail pesan, masukkan Subjek, seperti:

   ```
   Hello from Amazon SNS!
   ```
4. Di bagian Isi pesan, pilih Muatan identik untuk semua protokol pengiriman, lalu masukkan isi pesan, seperti:

   ```
   Publishing a message to an SNS topic.
   ```
5. Pilih Publish message (Publikasikan pesan).

   Pesan diterbitkan ke topik, dan konsol membuka halaman Detail topik.
6. Periksa kotak masuk email Anda dan verifikasi bahwa Anda menerima email dari Amazon SNS dengan pesan yang dipublikasikan.

Langkah 4: Hapus langganan dan topik

1. Di panel navigasi, pilih Subscriptions (Langganan).
2. Pada halaman Subscriptions (Langganan), pilih confirmed (dikonfirmasi) dan kemudian pilih Delete (Hapus).

   > **Note**
   >
   > Anda tidak dapat menghapus konfirmasi yang tertunda. Setelah 48 jam, Amazon SNS menghapusnya secara otomatis.

3. Di kotak dialog Delete subscription (Hapus langganan), pilih Delete (Hapus).

   Langganan dihapus.
4. Di panel navigasi, pilih Topics (Topik).
5. Pada halaman Topics (Topik), pilih topik dan kemudian pilih Delete (Hapus).

   > **Important**
   >
   > Saat menghapus topik, Anda juga menghapus semua langganan topik tersebut.

6. Pada kotak *`MyTopic`*dialog Hapus topik, masukkan `delete me` lalu pilih Hapus.

   Topik dihapus.

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 SQS) terenkripsi yang telah berlangganan.
- Berlangganan AWSEvent Fork Pipelines ke suatu topik.

Mengonfigurasi Amazon SNS

Gunakan Konsol Amazon SNS untuk membuat dan mengkonfigurasi topik dan langganan Amazon SNS. Untuk informasi lebih lanjut tentang Amazon SNS, lihat Apa itu Amazon SNS?

Topik

- Membuat topik Amazon SNS
- Berlangganan topik Amazon SNS
- Menghapus topik dan langganan Amazon SNS
- Penandaan topik Amazon SNS

Membuat topik Amazon SNS

Topik Amazon SNS adalah titik akses logis yang bertindak sebagai saluran komunikasi. Topik memungkinkan Anda mengelompokkan beberapa titik akhir (seperti AWS Lambda, Amazon SQS, 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.

Tugas Amazon SNS yang 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 atau FIFO) dan menamai 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
- Untuk membuat topik menggunakan AWS SDK

Untuk membuat topik menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
2. Lakukan salah satu hal berikut:
   - 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 Type (Jenis), pilih jenis topik (Standar atau FIFO).
   b. Masukkan Nama untuk topik. Untuk topik FIFO, tambahkan .fifo di akhir nama.
   c. (Opsional) Masukkan Nama tampilan untuk topik.

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

   d. (Opsional) Untuk topik FIFO, Anda dapat memilih deduplikasi pesan berbasis konten untuk mengaktifkan deduplikasi pesan default. Untuk informasi selengkapnya, lihat Deduplikasi pesan untuk topik FIFO.
5. (Opsional) Perluas bagian Encryption (Enkripsi) dan lakukan hal-hal berikut ini. Untuk informasi selengkapnya, lihat Enkripsi diam.
   a. Pilih Enable encryption (Aktifkan enkripsi).

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

Untuk setiap jenis KMS, Deskripsi, Akun, dan KMS ARN ditampilkan.

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

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

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

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

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

- Untuk menggunakan ARN KMS khusus dari akun AWS Anda atau dari akun AWS lain, masukkan ke bidang kunci KMS.

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, lihat Identity and access management di Amazon SNS dan Contoh kasus untuk pengendalian akses Amazon SNS.

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

7. (Opsional) Untuk mengkonfigurasi bagaimana Amazon SNS mencoba ulang upaya pengiriman pesan yang gagal, perluas bagian Kebijakan pengiriman ulang (HTTP/S). Untuk informasi selengkapnya, lihat Pengiriman ulang pesan Amazon SNS.
8. (Opsional) Untuk mengonfigurasi cara Amazon SNS mencatat pengiriman pesan ke CloudWatch, perluas bagian Pencatatan status pengiriman. Untuk informasi selengkapnya, lihat Status pengiriman pesan Amazon SNS.
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 topik Amazon SNS.
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 mengonfigurasinya dengan kredensyal Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensyal bersama di Panduan Referensi AWS SDK 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 mengatur 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 FIFO dan de-duplikasi tertentu.

```
    /// <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 detail API, lihat CreateTopicdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk membuat topik SNS

`create-topic`Contoh berikut membuat topik SNS 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 detail API, lihat CreateTopicdi Referensi AWS CLI Perintah.

Go

SDK for Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
```

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

 return topicArn, err
}
```

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

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat CreateTopicdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi AWS SDK untuk referensi API Kotlin.

PHP

SDK for PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi Referensi AWS SDK for PHP API.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi Referensi AWS SDK for Ruby API.

Rust

SDK for Rust

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicreferensi AWS SDK for Rust API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi AWS SDK untuk referensi SAP ABAP API.

Berlangganan topik Amazon SNS

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**
>
> Endpoint HTTP(S), alamat email, dan sumber daya AWS di Akun AWS lain memerlukan konfirmasi langganan sebelum mereka dapat menerima olahpesan.

Untuk berlangganan endpoint topik Amazon SNS

1. Masuk ke konsol Amazon SNS.
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 Topic ARN (ARN topik), pilih Amazon Resource Name (ARN) dari topik. Nilai ini adalah AWS ARN yang dihasilkan saat Anda membuat topik Amazon SNS, misalnya. `arn: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 Topik SNS FIFO, pilih opsi ini.

      - AWS Lambda
      - Titik akhir aplikasi platform
      - SMS

   c. Untuk Endpoint, masukkan nilai endpoint, seperti alamat email atau ARN antrean Amazon SQS.

   d. Hanya titik akhir Firehose: Untuk ARN peran Langganan, tentukan ARN dari peran 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 Firehose, Amazon SQS, HTTP/S endpoint, Anda juga dapat mengaktifkan pengiriman pesan mentah. Untuk informasi selengkapnya, lihat Pengiriman pesan mentah Amazon SNS.

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

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

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 Antrean surat mati Amazon SNS (DLQs).

i. Pilih Create subscription (Buat langganan).

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

Menghapus topik dan langganan Amazon SNS

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

Untuk menghapus topik atau langganan Amazon SNS menggunakan AWS Management Console

Untuk menghapus topik menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
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 konsol Amazon SNS.
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 mengonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi AWS SDK 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

Hapus topik berdasarkan topiknya ARN.

```
/// <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 detail API, lihat DeleteTopicdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat DeleteTopicdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk menghapus topik SNS

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

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

Perintah ini tidak menghasilkan output.

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

Go

SDK untuk Go V2

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

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

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

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat DeleteTopicdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DeleteTopicdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat DeleteTopicdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat DeleteTopicdi Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat DeleteTopicdi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

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

```
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 detail API, lihat DeleteTopicdi AWS SDK untuk referensi SAP ABAP API.

Penandaan topik Amazon SNS

Amazon SNS mendukung penandaan topik Amazon SNS. Hal ini dapat membantu Anda melacak dan mengelola biaya yang terkait dengan topik Anda, memberikan keamanan yang lebih baik di kebijakanAWS Identity and Access Management (IAM) Anda, serta memungkinkan Anda mencari atau memfilter ribuan topik dengan mudah. Penandaan memungkinkan Anda mengelola topik Amazon SNS Anda menggunakanAWS Resource Groups. Untuk informasi lebih lanjut tetang Resource Groups, lihat Panduan PenggunaAWS Resource Groups.

Topik

- Penandaan untuk alokasi biaya
- Penandaan untuk kontrol akses
- Penandaan untuk pencarian dan penyaringan sumber daya
- Mengonfigurasi tag topik Amazon SNS

Penandaan untuk alokasi biaya

Untuk mengelola dan mengidentifikasi topik Amazon SNS untuk alokasi biaya, Anda dapat menambahkan tanda yang mengidentifikasi tujuan topik. Ini sangat berguna ketika Anda memiliki banyak topik. Anda juga dapat menggunakan tanda untuk mengatur tagihan AWS Anda untuk merefleksikan struktur biaya Anda sendiri. Untuk melakukannya, daftar untuk membuat tagihan akun AWS Anda menyertakan kunci dan nilai tanda. Untuk informasi selengkapnya, lihat Menyiapkan Laporan Alokasi Biaya Bulanan dalam Panduan PenggunaAWS Billing and Cost Management.

Misalnya, Anda dapat menambahkan tanda yang mewakili pusat biaya dan tujuan topik Amazon SNS Anda, sebagai berikut:

Resource	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 mengizinkan Anda untuk mengelompokkan dua topik yang melakukan tugas terkait di pusat biaya yang sama, ketika menandai aktivitas yang tidak terkait dengan tanda alokasi biaya yang berbeda.

Penandaan untuk kontrol akses

AWS Identity and Access Managementmendukung akses pengendalian ke sumber daya berdasarkan tanda. Setelah menandai sumber daya Anda, berikan informasi tentang tanda sumber daya Anda di elemen kondisi kebijakan IAM untuk mengelola akses berbasis tanda. Untuk informasi tentang cara menandai sumber daya Anda menggunakan konsol Amazon SNS atau AWSSDK, lihat Mengonfigurasi tag.

Anda dapat membatasi akses untuk identitas IAM. Misalnya, Anda dapat membatasi`Publish` dan`PublishBatch` mengakses semua topik Amazon SNS yang menyertakan tag dengan kunci`environment` dan nilainya`production`, sekaligus memungkinkan akses ke semua topik Amazon SNS lainnya. Pada contoh di bawah ini, kebijakan membatasi kemampuan untuk mempublikasikan pesan ke topik yang ditandai dengan`production`, sementara memungkinkan pesan untuk dipublikasikan ke topik yang ditandai dengan`development`. Untuk informasi selengkapnya, lihat Mengendalikan Akses Menggunakan Tanda di Panduan Pengguna IAM.

> **Note**
>
> Mengatur izin IAM 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 penyaringan sumber daya

AWSAkun dapat memiliki puluhan ribu topik Amazon SNS (lihat Kuota Amazon SNS untuk detailnya). Dengan menandai topik Anda, Anda dapat menyederhanakan proses pencarian melalui atau menyaring topik.

Misalnya, Anda mungkin memiliki ratusan topik yang terkait dengan lingkungan produksi Anda. Daripada harus secara manual mencari topik ini, Anda dapat query untuk 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 topik Amazon SNS

Halaman ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console, AWS SDK, dan AWS CLI untuk mengonfigurasi tag untuk topik Amazon SNS.

> **Important**
>
> Jangan menambahkan informasi pengenal 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 topik Amazon SNS menggunakan AWS Management Console
- Menambahkan tag ke topik menggunakan AWS SDK
- Mengelola tag dengan tindakan API Amazon SNS
- Tindakan API yang mendukung ABAC

Membuat daftar, menambahkan, dan menghapus tag untuk topik Amazon SNS menggunakan AWS Management Console

1. Masuk ke Konsol Amazon SNS.
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 mengonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi AWS SDK 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 detail API, lihat TagResourcedi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat TagResourcedi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat TagResourcedi AWS SDK untuk referensi API Kotlin.

Mengelola tag dengan tindakan API Amazon SNS

Untuk mengelola tag menggunakan Amazon SNS API, gunakan tindakan API berikut:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

Tindakan API yang mendukung ABAC

Berikut ini adalah daftar tindakan API yang mendukung kontrol akses berbasis atribut (ABAC). Untuk detail lebih lanjut tentang ABAC, lihat Untuk apa ABAC? AWS di Panduan Pengguna IAM.

- `AddPermission`
- `ConfirmSubscription`
- `DeleteTopic`

- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

Pengurutan pesan dan deduplikasi (topik FIFO)

Anda dapat menggunakan topik Amazon SNS FIFO (masuk pertama, keluar pertama) dengan antrian Amazon SQS FIFO untuk memberikan pemesanan pesan yang ketat dan deduplikasi pesan. Kemampuan FIFO dari masing-masing layanan ini bekerja sama untuk bertindak sebagai layanan yang terkelola sepenuhnya untuk mengintegrasikan aplikasi terdistribusi yang memerlukan konsistensi data dalam waktu dekat. Berlangganan antrian standar Amazon SQS ke topik Amazon SNS FIFO memberikan pemesanan dengan upaya terbaik dan setidaknya sekali pengiriman.

Topik

- Contoh kasus penggunaan topik FIFO
- Detail pemesanan pesan untuk topik FIFO
- Grup pesan untuk topik FIFO
- Pengiriman pesan untuk topik FIFO
- Pemfilteran pesan untuk topik FIFO
- Deduplikasi pesan untuk topik FIFO
- Keamanan pesan untuk topik FIFO
- Daya tahan pesan untuk topik FIFO
- Pengarsipan pesan dan pemutaran ulang untuk topik FIFO
- Contoh kode untuk topik FIFO

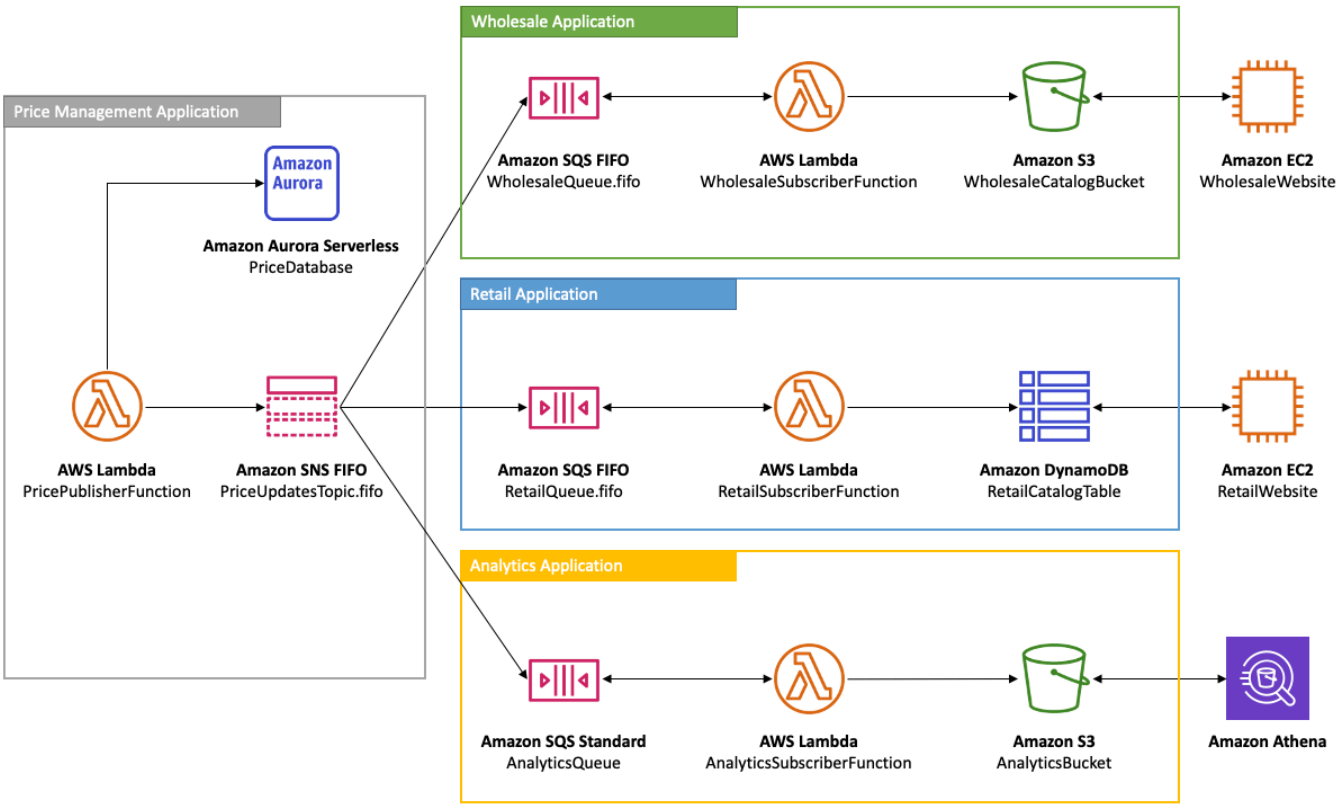
Contoh kasus penggunaan topik FIFO

Contoh berikut menjelaskan platform e-niaga yang dibangun oleh produsen suku cadang mobil menggunakan topik Amazon SNS FIFO dan antrian Amazon SQS. 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 topik Amazon SNS FIFO 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 antrian Amazon SQS FIFO ke topik Amazon SNS FIFO aplikasi manajemen harga.

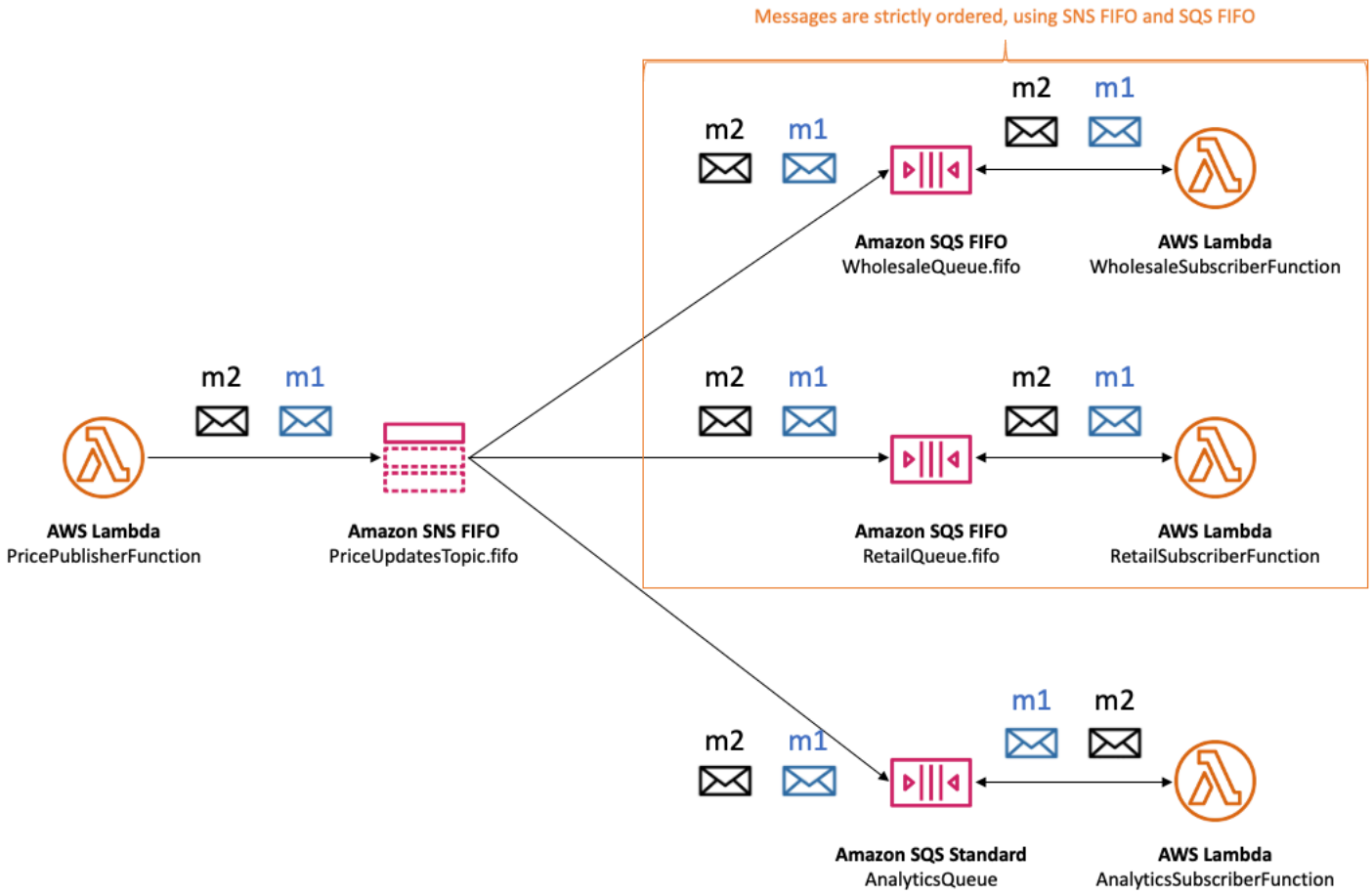
- Sebuah aplikasi ritel menyediakan backend untuk situs web lain tempat pemilik mobil dan penggemar penyetelan mobil dapat membeli suku cadang mobil individu untuk kendaraan mereka. Untuk mendapatkan pemberitahuan perubahan harga, aplikasi ritel juga berlangganan antrian Amazon SQS FIFO ke topik Amazon 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 standar Amazon SQS ke topik Amazon SNS FIFO aplikasi manajemen harga. Berbeda dengan aplikasi lain, analitik tidak memerlukan pembaruan harga untuk dipesan secara ketat.



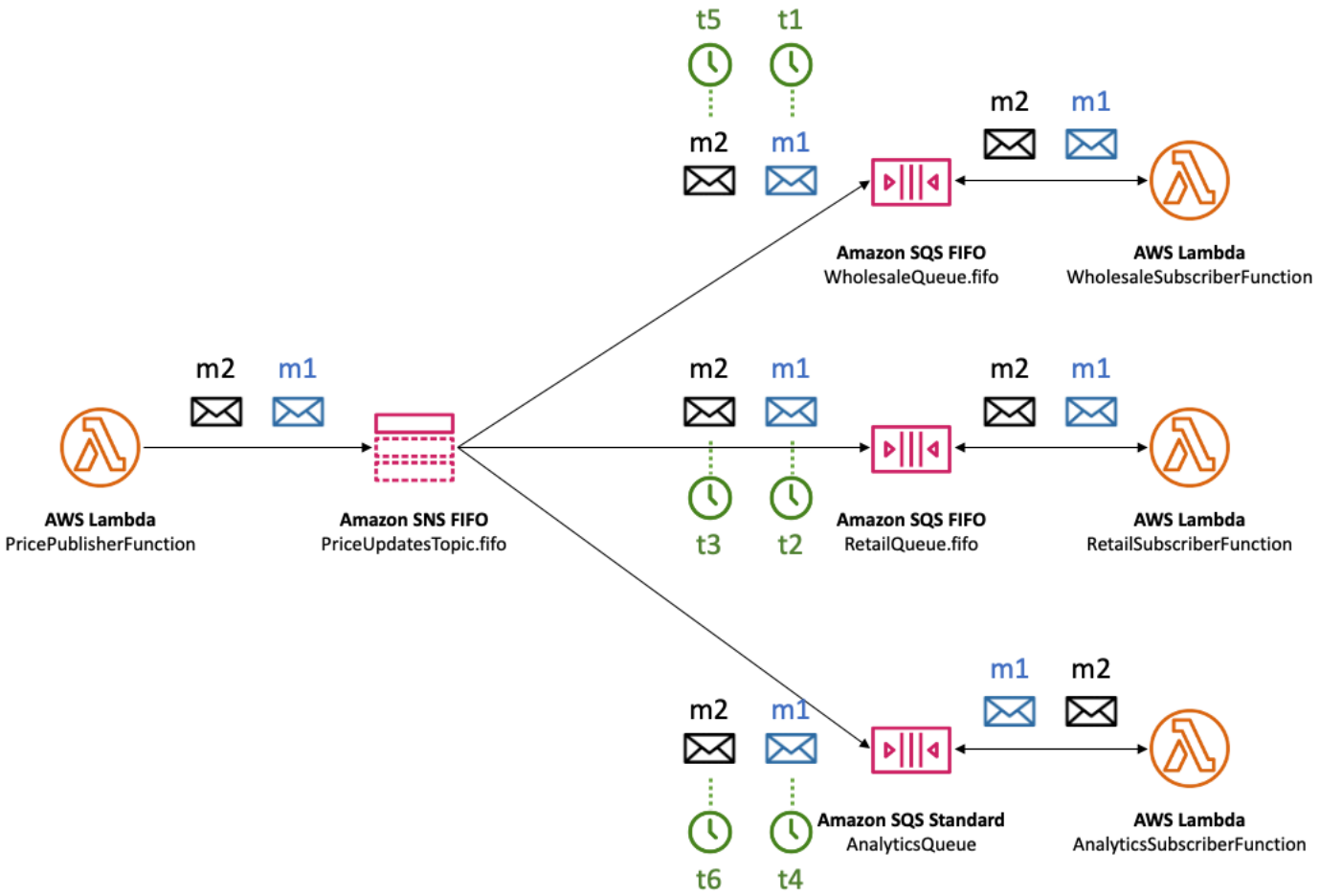
Agar aplikasi grosir dan ritel menerima pembaruan harga dalam urutan yang benar, aplikasi manajemen harga harus menggunakan sistem distribusi pesan yang diurutkan secara ketat. Menggunakan topik Amazon SNS FIFO dan antrian Amazon SQS FIFO memungkinkan pemrosesan pesan secara berurutan dan tanpa duplikasi.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

Detail pemesanan pesan untuk topik FIFO

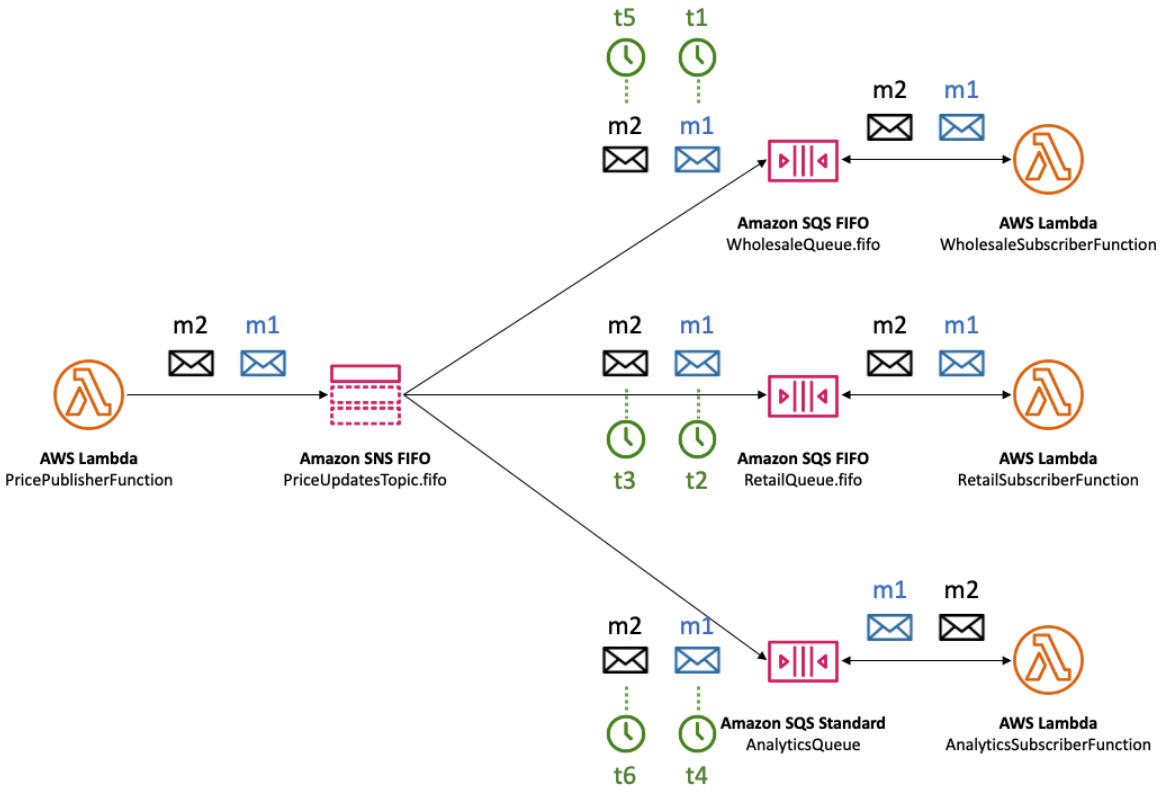
Topik Amazon SNS FIFO selalu mengirimkan pesan ke antrian Amazon SQS berlangganan dalam urutan yang tepat di mana pesan dipublikasikan ke topik, dan hanya sekali. Dengan antrian Amazon SQS FIFO berlangganan, konsumen antrian menerima pesan dalam urutan yang tepat di mana pesan dikirim ke antrian, dan tidak ada duplikat. Namun, dengan antrean standar Amazon SQS 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 topik FIFO



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 pelanggan Amazon SQS FIFO. Setiap pelanggan dianggap terpisah dari pelanggan lain.

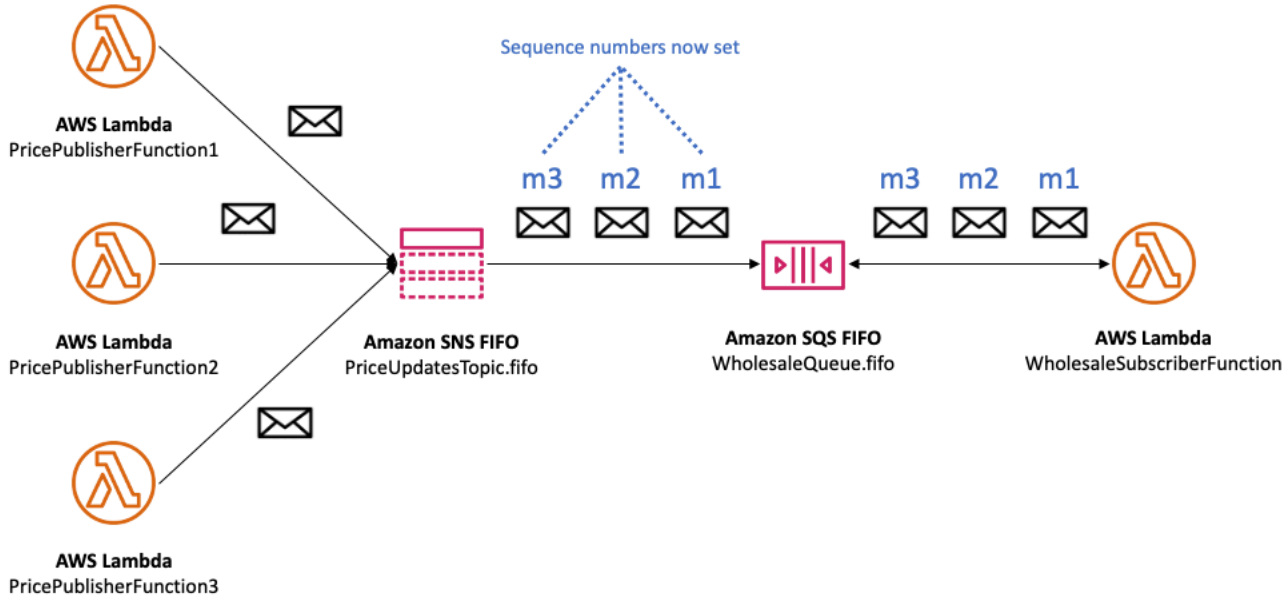


Jika pelanggan antrian Amazon SQS menjadi tidak dapat dijangkau, itu bisa keluar dari sinkronisasi. Sebagai contoh, mengatakan pemilik antrean aplikasi grosir keliru mengubah kebijakan antrean Amazon SQS dengan cara yang mencegah layanan utama Amazon SNS mengirimkan pesan ke antrean. Dalam hal ini, pengiriman pembaruan harga ke antrian grosir gagal, sedangkan antrian ritel dan analitik berhasil, menyebabkan pelanggan tidak sinkron. Ketika pemilik antrian aplikasi grosir mengoreksi kebijakan antreannya, Amazon SNS 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 thread dalam aplikasi yang sama) menerbitkan pesan ke topik SNS FIFO secara paralel. Ketika Anda melakukan ini, Anda secara efektif mendelegasikan urutan pesan ke layanan Amazon SNS. Untuk menentukan urutan pesan yang telah ditetapkan, Anda dapat memeriksa nomor urutannya.

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



Topik Amazon SNS FIFO mendefinisikan pemesanan dalam konteks grup pesan. Untuk informasi selengkapnya, lihat Grup pesan untuk topik FIFO.

Grup pesan untuk topik FIFO

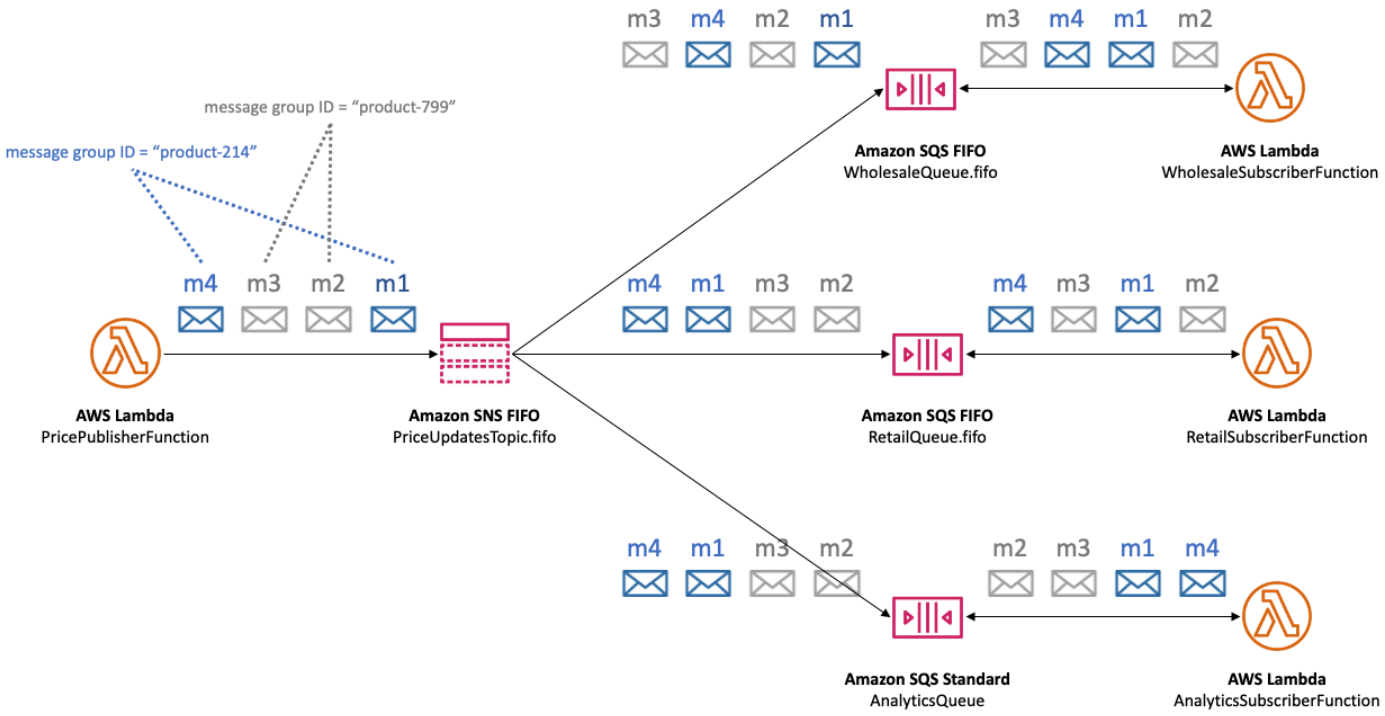
Pesan yang termasuk dalam grup yang sama diproses satu per satu, dalam urutan yang ketat relatif terhadap grup.

Ketika Anda mempublikasikan pesan ke topik Amazon SNS FIFO, Anda menetapkan ID grup pesan. ID grup adalah token wajib yang menentukan bahwa pesan milik grup pesan tertentu. Topik SNS FIFO melewati ID grup untuk antrean Amazon SQS FIFO berlangganan. Tidak ada batas untuk jumlah ID grup di SNS FIFO topik atau antrean SQS FIFO. ID grup pesan tidak diteruskan ke antrian standar Amazon SQS.

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 pesan untuk topik FIFO dan Pemfilteran pesan untuk topik FIFO.

Di bagian auto manajemen harga contoh kasus penggunaan, ada grup pesan khusus untuk setiap produk yang dijual di platform. Topik Amazon SNS FIFO 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

m1 diproses sebelum pesan m4. Urutan ini dipertahankan di seluruh alur kerja yang menggunakan Amazon SNS FIFO ke Amazon SQS FIFO. Demikian juga, untuk produk yang ID grup pesannya adalah product-799, pesan m2 diproses sebelum pesan m3, selama alur kerjanya menggunakan Amazon SNS FIFO dan Amazon SQS FIFO. Namun, saat menggunakan antrian standar Amazon SQS, 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 ID grup pesan untuk meningkatkan kinerja

Untuk mengoptimalkan throughput pengiriman, topik Amazon SNS FIFO 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 ID grup pesan yang berbeda. Dengan memanfaatkan kumpulan grup pesan yang beragam, topik Amazon SNS FIFO secara otomatis mendistribusikan pesan di sejumlah besar partisi paralel.

> **Note**
>
> Topik Amazon SNS FIFO dioptimalkan untuk distribusi pesan yang seragam di seluruh ID grup pesan, terlepas dari jumlah grup. AWSmerekomendasikan agar Anda menggunakan sejumlah besar ID grup pesan yang berbeda untuk kinerja yang dioptimalkan.

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

Pengiriman pesan untuk topik FIFO

Topik Amazon SNS FIFO (pertama masuk, keluar pertama) mendukung pengiriman ke standar Amazon SQS dan antrian FIFO 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 antrian Amazon SQS FIFO berlangganan karena titik akhir pengiriman menyediakan 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 Amazon SQS ke topik Amazon SNS FIFO memberikan kemampuan untuk menurunkan biaya, selain berbagi antrian di seluruh beban kerja yang tidak menggunakan FIFO.

> **Note**
>
> Untuk mengirimkan pesan dari topik FIFO Amazon SNS ke fungsi AWS Lambda, perlu 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 FIFO SQS sebagai sumber peristiwa di Blog Komputasi AWS.

Topik FIFO SNS tidak dapat mengirimkan pesan ke titik akhir yang dikelola pelanggan, seperti alamat email, aplikasi seluler, nomor telepon untuk olahpesan 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 topik FIFO SNS mengakibatkan kesalahan.

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

Pemfilteran pesan untuk topik FIFO

Topik FIFO Amazon SNS 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 topik SNS FIFO, 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 filter Amazon SNS 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.

Topik FIFO SNS mendukung berbagai operator yang cocok, termasuk nilai string atribut, nilai numerik atribut, dan kunci atribut. Untuk informasi selengkapnya, lihat Pemfilteran pesan Amazon SNS.

Topik FIFO SNS tidak mengirimkan pesan duplikat ke titik akhir langganan. Untuk informasi selengkapnya, lihat Deduplikasi pesan untuk topik FIFO.

Deduplikasi pesan untuk topik FIFO

Topik FIFO Amazon SNS dan antrean FIFO Amazon SQS mendukung deduplikasi pesan, yang menyediakan pengiriman dan pemrosesan pesan tepat satu kali selama syarat berikut terpenuhi:

- Antrian Amazon SQS FIFO berlangganan ada dan memiliki izin yang memungkinkan kepala layanan Amazon SNS untuk mengirimkan pesan ke antrian.
- Konsumen antrian Amazon SQS FIFO memproses pesan dan menghapusnya dari antrian sebelum batas waktu visibilitas berakhir.
- Topik langganan Amazon SNS tidak memiliki pemfIlteran pesan. Saat Anda mengonfigurasi pemfilteran pesan, topik Amazon SNS FIFO at-most-once mendukung 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 topik Amazon SNS FIFO, bukan untuk grup pesan individual.

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

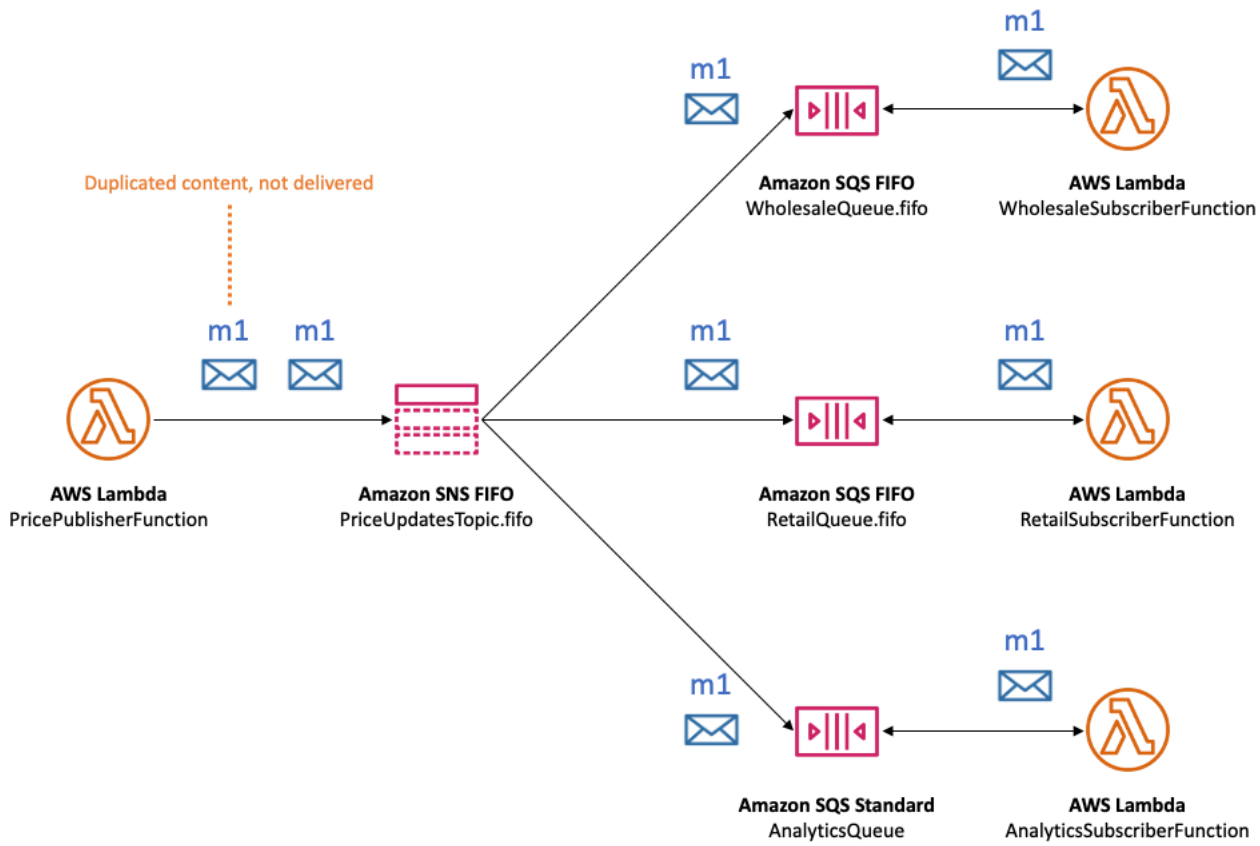
Jika pesan dengan ID deduplikasi tertentu berhasil dipublikasikan ke topik Amazon SNS FIFO, pesan apa pun yang diterbitkan dengan ID deduplikasi yang sama, dalam interval deduplikasi lima menit, diterima tetapi tidak dikirim. Topik Amazon SNS FIFO terus melacak ID deduplikasi pesan, bahkan setelah pesan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

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

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

Saat deduplikasi berbasis konten diaktifkan untuk topik Amazon SNS FIFO, 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 topik Amazon SNS FIFO dan antrian FIFO Amazon SQS berlangganan.



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

Keamanan pesan untuk topik FIFO

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

> **Note**
>
> Menambahkan enkripsi ke topik FIFO atau antrean tidak mengenkripsi pesan yang disimpan kembali, dan menghapus enkripsi dari topik atau antrean meninggalkan pesan yang tersimpan terenkripsi.

Topik SNS FIFO mendekripsi pesan segera sebelum mengirimkannya ke titik akhir berlangganan. Antrean SQS FIFO mendekripsi pesan sebelum mengembalikan mereka ke aplikasi konsumen. Untuk informasi selengkapnya, lihat Enkripsi data dan Mengenkripsi pesan yang dipublikasikan ke Amazon SNS dengan posting AWS KMS di AWS Komputasi Blog.

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

Topik SNS FIFO juga mendukung antrean surat mati dan penyimpanan pesan di seluruh Availability Zones. Untuk informasi selengkapnya, lihat Daya tahan pesan untuk topik FIFO.

Daya tahan pesan untuk topik FIFO

Topik Amazon SNS FIFO dan antrian Amazon SQS 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 Amazon SNS, pengiriman pesan gagal ketika topik Amazon SNS tidak dapat mengakses antrean Amazon SQS langganan karena kesalahan sisi klien atau sisi server:

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

- Mengubah kebijakan antrean dengan cara yang mencegah perwakilan layanan Amazon SNS mengirimkan pesan ke sana.

Amazon SNS tidak mencoba lagi mengirimkan pesan yang gagal karena kesalahan sisi klien.

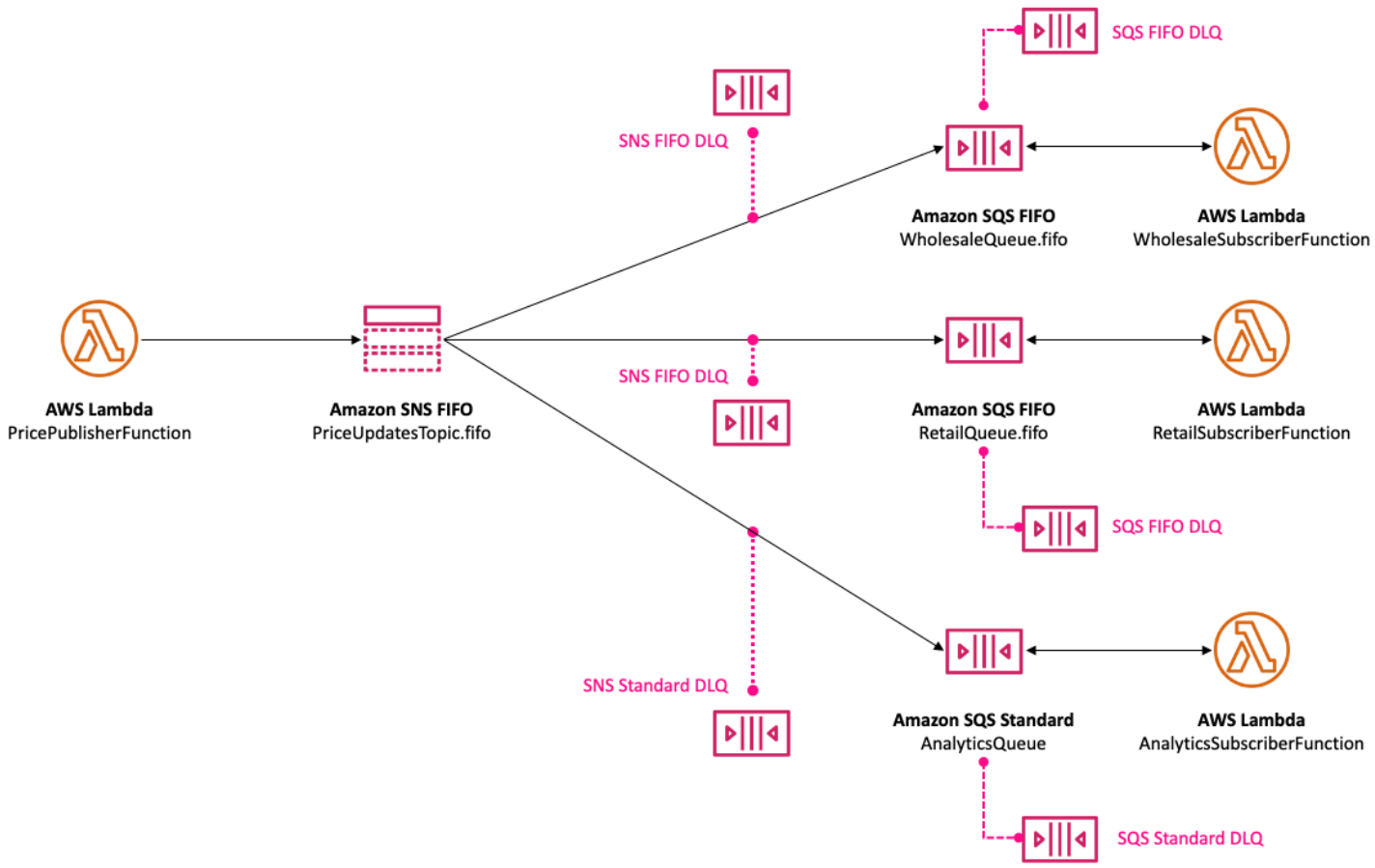
- Kesalahan sisi server dapat terjadi dalam situasi ini:
  - Layanan Amazon SQS tidak tersedia.
  - Amazon SQS gagal memproses permintaan yang valid dari layanan Amazon SNS.

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

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

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

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



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

Untuk ketahanan yang lebih lama untuk membantu pemulihan dari kegagalan hilir, pemilik topik juga dapat menggunakan topik FIFO 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 pesan dan pemutaran ulang untuk topik FIFO.

Pengarsipan pesan dan pemutaran ulang untuk topik FIFO

Topik

- Apa itu pengarsipan dan pemutaran ulang pesan?
- Pengarsipan pesan untuk pemilik topik FIFO
- Putar ulang pesan untuk pelanggan topik FIFO

Apa itu pengarsipan dan pemutaran ulang pesan?

Pengarsipan dan pemutaran ulang pesan Amazon SNS adalah arsip pesan tanpa kode di tempat 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, yang dapat digunakan untuk:

- Pulihkan pesan yang mungkin hilang karena kegagalan dalam aplikasi hilir.
- Replikasi status aplikasi yang ada ke aplikasi baru dengan berlangganan endpoint baru, dan pilih stempel waktu yang diinginkan untuk direplikasi.

Anda dapat menggunakan pengarsipan dan pemutaran ulang pesan dengan AWS API, SDK, AWS CloudFormation dan. AWS Management Console

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

Pengarsipan dan pemutaran ulang pesan terdiri dari dua komponen utama:

1. Pengarsipan pesan - Pemilik topik mengaktifkan fitur pengarsipan dan pemutaran ulang pada suatu topik, dan menetapkan periode penyimpanan pesan (hingga 365 hari). Pemilik topik juga dapat memantau pesan yang diarsipkan menggunakan Amazon CloudWatch metrik. Untuk lebih lanjut, lihatPengarsipan pesan untuk pemilik topik FIFO.
2. Putar ulang pesan - Pelanggan topik memulai pemutaran ulang untuk serangkaian pesan dari topik ke titik akhir berlangganan mereka. Untuk lebih lanjut lihat,Putar ulang pesan untuk pelanggan topik FIFO.

Pengarsipan pesan untuk pemilik topik FIFO

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 harga Amazon SNS.

Topik

- Membuat kebijakan arsip pesan menggunakan AWS Management Console
- Membuat kebijakan arsip pesan menggunakan API
- Membuat kebijakan arsip pesan menggunakan SDK
- Membuat kebijakan arsip pesan menggunakan AWS CloudFormation
- Berikan akses ke arsip terenkripsi
- Pantau metrik arsip pesan menggunakan Amazon CloudWatch

Membuat kebijakan arsip pesan menggunakan AWS Management Console

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

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

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

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 JSON kosong.
  - Periode penyimpanan pesan — Jumlah hari yang ditentukan untuk penyimpanan pesan.
  - Tanggal mulai arsip — Tanggal dari mana pelanggan dapat memutar ulang pesan.
  - Pratinjau JSON - Pratinjau JSON dari 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 akan 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 menggunakan API, Anda perlu menambahkan atribut `ArchivePolicy` ke topik Anda. Anda dapat mengatur `ArchivePolicy` menggunakan tindakan API `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 yang 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 mengonfigurasinya dengan kredensil Anda. Untuk informasi selengkapnya, lihat Berbagi `config` dan `credentials` file di Panduan Referensi AWS SDK dan Alat.

Contoh kode berikut menunjukkan cara mengatur topik Amazon SNS `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 `Decrypt` disediakan akses ke kunci KMS yang digunakan untuk mengenkripsi pesan dalam arsip.

1. Saat Anda mengenkripsi pesan dengan kunci KMS 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 Amazon SNS. Untuk lebih lanjut, lihatMengonfigurasi izin AWS KMS.

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

Berikan izin dekripsi ke Amazon SNS

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

Berikut ini adalah contoh kebijakan yang diperlukan untuk mengizinkan prinsipal layanan Amazon SNS 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.

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

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

```
  ...
}
```

Putar ulang pesan untuk pelanggan topik FIFO

Pemutaran ulang Amazon SNS 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.

Topik

- Membuat kebijakan pemutaran ulang pesan menggunakan AWS Management Console
- Menambahkan kebijakan pemutaran ulang ke langganan menggunakan API
- Menambahkan kebijakan pemutaran ulang ke langganan menggunakan SDK
- Filter pesan yang diputar ulang
- Pantau metrik pemutaran ulang pesan menggunakan Amazon CloudWatch

Membuat kebijakan pemutaran ulang pesan menggunakan AWS Management Console

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

1. Masuk ke Konsol Amazon SNS.
2. Pilih langganan topik atau buat yang baru. Untuk mempelajari lebih lanjut tentang membuat langganan, lihatBerlangganan topik Amazon SNS.
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 (format YYYY/MM/DD) 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 (format YYYY/MM/DD) 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.
  - Pending - 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.

Menambahkan kebijakan pemutaran ulang ke langganan menggunakan API

Untuk memutar ulang pesan yang diarsipkan, gunakan atribut. `ReplayPolicy` `ReplayPolicy`dapat digunakan dengan tindakan `SetSubscriptionAttributes` API `Subscribe` dan. 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 selama dua jam pada 1 Oktober 2023, gunakan tindakan `SetSubscriptionAttributes` API 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 dipublikasikan ke topik Anda, gunakan tindakan `SetSubscriptionAttributes` API untuk menyetel 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.

Menambahkan kebijakan pemutaran ulang ke langganan menggunakan SDK

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

Contoh kode berikut menunjukkan cara mengatur langganan untuk mengirim ulang pesan dari arsip topik Amazon SNS FIFO 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);
```

Filter pesan yang diputar ulang

Pemfilteran pesan Amazon SNS memungkinkan Anda mengontrol pesan yang diputar ulang yang diputar ulang Amazon SNS 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 langganan Amazon SNS.

Pantau metrik pemutaran ulang pesan menggunakan Amazon CloudWatch

Anda dapat memantau pesan pemutaran ulang 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 kode untuk topik FIFO

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

Topik

- Menggunakan AWS SDK
- Menggunakan AWS CloudFormation

Menggunakan AWS SDK

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

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

- Topik Amazon SNS FIFO yang mendistribusikan pembaruan harga
- Antrian Amazon SQS FIFO yang menyediakan pembaruan ini untuk aplikasi grosir dan eceran
- Antrian standar Amazon SQS untuk aplikasi analitik yang menyimpan catatan, yang dapat ditanyakan untuk intelijen bisnis (BI)
- Langganan Amazon SNS FIFO 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 pesan untuk topik FIFO.

Java

SDK untuk 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 topik Amazon SNS FIFO, dua antrian FIFO Amazon SQS, 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 detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x.
  - CreateTopic
  - Publikasikan
  - Berlangganan

Python

SDK untuk Python (Boto3)

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

Buat topik Amazon SNS FIFO, berlangganan Amazon SQS FIFO dan antrian standar ke topik tersebut, 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 detail API, lihat topik berikut di Referensi API AWS SDK for Python (Boto3).
  - CreateTopic
  - Publikasikan
  - Berlangganan

SAP ABAP

SDK for SAP ABAP

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

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

```
    " 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 mengetahui hal detail mengenai API, silakan lihat topik-topik berikut di SDK AWS untuk referensi API ABAP SAP.
  - CreateTopic
  - Publikasikan
  - Berlangganan

Menerima pesan dari langganan FIFO

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

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

Menggunakan AWS CloudFormation

AWS CloudFormationmemungkinkan 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:

- Topik Amazon SNS FIFO yang mendistribusikan pembaruan harga
- Antrian Amazon SQS FIFO yang menyediakan pembaruan ini untuk aplikasi grosir dan eceran
- Antrian standar Amazon SQS untuk aplikasi analitik yang menyimpan catatan, yang dapat ditanyakan untuk intelijen bisnis (BI)
- Langganan Amazon SNS FIFO 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 lebih lanjut tentang men-deploy sumber daya AWS menggunakan templat AWS CloudFormation, lihat Memulai di AWS CloudFormation Panduan Pengguna.

Penerbitan pesan Amazon SNS

Setelah Anda membuat topik Amazon SNS dan berlangganan titik akhir ke topik tersebut, Anda dapat menerbitkan pesan ke topik. Ketika pesan diterbitkan, Amazon SNS mencoba untuk mengirimkan pesan ke titik akhir berlangganan.

Topik

- Cara menerbitkan pesan ke topik Amazon SNS menggunakan AWS Management Console
- Untuk memublikasikan pesan ke topik menggunakan AWS SDK
- Penerbitan pesan large dengan Amazon SNS dan Amazon S3
- Atribut pesan Amazon SNS
- Batching pesan Amazon SNS

Cara menerbitkan pesan ke topik Amazon SNS menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
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 topik FIFO, masukkan ID grup pesan. Pesan-pesan dalam grup pesan yang sama akan dikirim sesuai urutan yang penerbitannya.

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

   d. (Opsional) Untuk mobile push notifications (notifikasi push seluler), masukkan nilai Waktu untuk tayang (TTL) dalam detik. Ini adalah jumlah waktu yang dibutuhkan 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 Custom payload for each delivery protocol (Muatan kustom untuk setiap protokol pengiriman), lalu masukkan objek JSON untuk menentukan pesan yang akan dikirim untuk setiap protokol pengiriman.

Untuk informasi selengkapnya, lihat Penerbitan dengan muatan khusus platform.

6. Di bagian Message attributes (Atribut pesan), tambahkan atribut yang Anda inginkan agar Amazon SNS cocok dengan `FilterPolicy` atribut langganan untuk memutuskan apakah titik akhir langganan tertarik dengan pesan yang diterbitkan.

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 pesan Amazon SNS.

7. Pilih Publish message (Terbitkan pesan).

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

Untuk memublikasikan pesan ke topik menggunakan AWS SDK

Untuk menggunakan AWS SDK, Anda harus mengonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi AWS SDK 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 mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for C++ API.

CLI

AWS CLI

Contoh 1: Untuk mempublikasikan pesan ke topik

`publish`Contoh berikut menerbitkan pesan yang ditentukan ke topik SNS 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 pesan SMS 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 detail API, lihat Menerbitkan di Referensi AWS CLI Perintah.

Go

SDK for Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

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

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, deduplicationId, choices;

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

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

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

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

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

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

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

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat Publikasikan di Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Menerbitkan di AWS SDK untuk referensi API Kotlin.

PHP

SDK for PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for PHP API.

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 detail API, lihat Menerbitkan di Referensi AWS Tools for PowerShell Cmdlet.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Menerbitkan di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for Ruby API.

Rust

SDK for Rust

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Menerbitkan di AWS SDK untuk referensi API Rust.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di AWS SDK untuk referensi API SAP ABAP.

Penerbitan pesan large dengan Amazon SNS dan Amazon S3

Untuk mempublikasikan pesan Amazon SNS yang besar, Anda dapat menggunakan Amazon SNS Extended Client Library for 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. Berlangganan Amazon SQS antrian 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 topik standar dan FIFO.

Topik

- Perpustakaan Klien yang Diperluas untuk Java
- Perpustakaan Klien Diperpanjang untuk Python

Perpustakaan Klien yang Diperluas untuk Java

Topik

- Prasyarat
- Mengkonfigurasi penyimpanan pesan
- Contoh: Penerbitan pesan ke Amazon SNS dengan muatan yang disimpan di Amazon S3
- Protokol titik akhir lainnya

Prasyarat

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

- Sebuah AWS SDK.

  Contoh pada halaman ini menggunakan AWS Java SDK. Untuk menginstal dan mengatur SDK, lihat Menyiapkan AWS SDK for Java di AWS SDK for Java Panduan Developer.

- 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);
```

- Kunci kustom KMS - Kunci yang digunakan untuk enkripsi sisi server dalam bucket Amazon S3 Anda.
- Nama bucket - Nama bucket Amazon S3 untuk menyimpan muatan pesan.

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

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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.

SDK for Java 1.x

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

Kedua perpustakaan Amazon SNS dan Amazon SQS menggunakan Muatan Pembongkar Java Common Library untuk AWS untuk menyimpan dan mengambil muatan pesan dengan Amazon S3. Titik akhir Java diaktifkan (misalnya, titik akhir HTTPS yang diterapkan di Java) dapat menggunakan perpustakaan yang sama untuk menghilangkan referensi isi 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"
  }
]
```

Perpustakaan Klien Diperpanjang 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 mengatur SDK, lihat dokumentasi AWS SDK untuk 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 kredensial, lihat Kredensial di SDK AWS for Python Developer Guide.
- 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 Klien, Topik, PlatformEndpointdan objek Amazon SNS 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 untuk melakukan:

- Buat contoh topik Amazon SNS dan antrean Amazon SQS.
- 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 pesan Amazon SNS

Amazon SNS mendukung pengiriman atribut pesan, yang memungkinkan Anda menyediakan item metadata terstruktur (seperti cap waktu, data geospasial, tanda tangan, dan pengidentifikasi) tentang pesan. Untuk langganan SQS, 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 Pengiriman Pesan Mentah mengaktifkan langganan Amazon SQS 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.

Selengkapnya tentang mengirim pesan dengan atribut menggunakan AWS Management Console atau AWS SDK for Java, lihat tutorial Cara menerbitkan pesan ke topik Amazon SNS menggunakan AWS Management Console.

> **Note**
>
> Atribut pesan dikirim hanya ketika struktur pesan String, bukan JSON.

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 endpoint saat mengirim pesan dengan atribut pesan ke Amazon SQS endpoint.

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 Pemfilteran pesan Amazon SNS.

> **Note**
>
> Ketika atribut pesan digunakan untuk pemfilteran, nilainya harus berupa string JSON 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. Jenis data peka huruf besar/kecil, dan panjangnya bisa mencapai 256 byte. 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 Publish (Publikasikan) di Amazon Simple Notification Service API Reference (Referensi API Amazon Simple Notification Service).

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

Jenis data atribut pesan mengidentifikasi bagaimana nilai atribut pesan ditangani oleh Amazon SNS. Sebagai contoh, jika jenis adalah angka, Amazon SNS memvalidasi bahwa itu adalah angka.

Amazon SNS mendukung jenis data logis berikut untuk semua endpoint kecuali seperti yang dicatat:

- String — String adalah Unicode dengan pengkodean biner UTF-8. 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.

  Jenis data ini tidak didukung untuk langganan AWS Lambda. Jika Anda menentukan jenis data ini untuk endpoint Lambda, ini akan diteruskan sebagai jenis data `String` dalam muatan JSON yang dikirimkan Amazon SNS 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.

Jenis data ini tidak didukung untuk langganan AWS Lambda. Jika Anda menentukan jenis data ini untuk endpoint Lambda, ini akan diteruskan sebagai jenis data `String` dalam muatan JSON yang dikirimkan Amazon SNS 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`
APN 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 notifikasi Amazon SNS jika atribut pesan tidak memenuhi persyaratannya. Untuk detail tambahan, lihat Mengirim Permintaan Pemberitahuan ke APN di situs web Pengembang Apple.

Batching pesan Amazon SNS

Apa itu batching pesan?

Alternatif untuk memublikasikan pesan ke topik Standar atau FIFO dalam permintaan`Publish` API individual, adalah menggunakan Amazon SNS`PublishBatch` API untuk mempublikasikan hingga 10 pesan dalam satu permintaan API. 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 publikasikan ke topik per detik berdasarkan wilayah tempat Anda beroperasi. Lihat halaman endpoint dan kuota Amazon SNS di Referensi Umum AWSpanduan untuk informasi selengkapnya tentang kuota API.

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

Bagaimana cara kerja batching pesan?

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

Contoh

Menerbitkan sejumlah 10 pesan ke topik Standar

```
// 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 sejumlah 10 pesan ke topik FIFO

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

Pemfilteran pesan Amazon SNS

Secara default, pelanggan topik Amazon SNS menerima setiap pesan yang diterbitkan untuk topik. Untuk hanya menerima subset pesan, pelanggan harus menetapkan kebijakan filter untuk langganan topik.

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

Jika langganan tidak memiliki kebijakan filter, pelanggan akan menerima setiap pesan yang diterbitkan untuk totopiknya. Ketika Anda mempublikasikan pesan untuk topik dengan kebijakan filter, Amazon SNS membandingkan atribut pesan untuk atribut dalam kebijakan filter untuk masing-masing topik langganan. Jika salah satu atribut pesan atau properti isi pesan cocok, Amazon SNS mengirimkan pesan ke pelanggan. Jika tidak, Amazon SNS tidak mengirim pesan ke pelanggan tersebut.

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

Topik

- Cakupan kebijakan filter langganan Amazon SNS
- Kebijakan filter langganan Amazon SNS
- Menerapkan kebijakan filter langganan
- Menghapus kebijakan filter langganan

Cakupan kebijakan filter langganan Amazon SNS

Atribut `FilterPolicyScope` langganan memungkinkan Anda memilih cakupan pemfilteran dengan menetapkan salah satu nilai berikut:

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

> **Note**
>
> Jika tidak ada cakupan kebijakan filter yang didefinisikan untuk kebijakan filter yang ada, cakupan default ke. `MessageAttributes`

Kebijakan filter langganan Amazon SNS

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

Saat Amazon SNS mengevaluasi atribut pesan atau properti isi pesan terhadap kebijakan filter langganan, Amazon SNS 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:

- Logika DAN
- 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

Contoh kebijakan filter

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

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

- Minat pelanggan
- Nama penyimpanan
- State kejadian
- Harga beli dalam 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 pesan Amazon SNS.

```
{
   "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 ada properti tunggal dalam kebijakan ini yang 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"
   ]
```

```
}
```

Kendala kebijakan filter

Saat Anda membuat kebijakan filter untuk langganan Amazon SNS, penting untuk memahami bagaimana kunci dalam kebijakan dihitung. Aspek kunci yang perlu diingat adalah:

1. Parent Keys — Kunci induk adalah kunci tingkat atas dalam kebijakan filter. Ini adalah kunci yang Anda tentukan nilai atau kendala.
2. Nama Atribut — Kunci induk dianggap sebagai nama atribut dalam kebijakan filter. Nilai atau kendala yang Anda tentukan untuk kunci ini akan diterapkan ke atribut yang sesuai dalam payload pesan.
3. Nilai yang valid - Nilai yang ditentukan untuk kunci induk harus berupa string, array string, atau angka. Jika nilainya adalah objek (misalnya, objek JSON), nilai tersebut tidak akan dihitung sebagai kunci yang valid dalam kebijakan filter.

Mari kita pertimbangkan contoh kebijakan filter berikut:

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

Dalam contoh ini, kunci berikut dihitung sebagai bagian dari kebijakan filter:

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

Temuan kunci tidak dihitung, karena berisi objek JSON sebagai nilainya, bukan string, array string, atau angka.

Contoh lain:

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

Dalam hal ini, hanya kunci `key_d` dan `key_f` dihitung sebagai bagian dari kebijakan filter, karena mereka memiliki nilai yang ditetapkan untuk mereka yang baik string atau array string. Kunci indukkey_a,key_b, dan tidak `key_c` dihitung, karena berisi objek JSON bersarang sebagai nilainya.

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"]
}
```

Dalam kebijakan ini:

- Array pertama memiliki 3 nilai
- Array kedua memiliki 1 nilai
- Array ketiga memiliki 2 nilai

Kombinasi total dihitung sebagai berikut:

- 3 x 1 x 2 = 6

Filter sintaks kebijakan

JSON kebijakan filter dapat berisi:

- String terlampir dalam tanda kutip
- Nomor
- Kata kunci`true`,`false`, dan`null`, tanpa tanda kutip

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

Filter batas kebijakan

- 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 antrian Amazon SQS dibuat dengan API. `Subscribe` Namun, itu akan gagal saat Anda melampirkan kebijakan filter dalam panggilan `Subscribe` API (atau panggilan `SetSubscriptionAttributes` API).
- 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 olahpesan dengan tipe data `Binary`.
- 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"]
    }
}
```

Dalam kebijakan ini:

- Array pertama memiliki empat nilai dalam kunci bersarang tiga tingkat.
- Yang kedua memiliki tiga nilai dalam kunci bersarang dua tingkat.

Kombinasi total dihitung sebagai berikut:

- 4 x 3 x 3 x 2 = 72

Batas kebijakan

Kebijakan filter dapat memiliki maksimal lima kunci induk (kunci tingkat atas). Untuk kebijakan bersarang, hanya kunci induk yang dihitung terhadap batas lima kunci.

Rentang numerik

Untuk pencocokan numerik dalam kebijakan filter, nilainya dapat berkisar dari -10^9 hingga 10^9 (-1 miliar hingga 1 miliar), dengan lima digit akurasi setelah titik desimal.

Beralih ke pemfilteran berbasis muatan

Untuk beralih dari pemfilteran berbasis atribut (default) ke pemfilteran berbasis muatan, Anda harus menyetel ke dalam `FilterPolicyScope` langganan. `MessageBody`

Logika DAN/ATAU

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

Topik

- Logika DAN
- Logika ATAU
- ATAU operator

Logika DAN

Anda dapat menerapkan logika AND 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 logika AND ke 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 mengenali `"$or"` hubungan ketika kebijakan telah memenuhi semua persyaratan berikut. Ketika semua kondisi ini tidak terpenuhi, `"$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

Standar0R:

```
{
  "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
```

`metric`Name`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 apa pun yang berisi kunci `store` atribut, seperti berikut ini:

```
"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-nilai numerik tidak dikutip 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-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
- Apa saja tapi tidak cocok

- Menggunakan prefiks dengan operator anything-but
- quals-ignore-case Pencocokan E
- Pencocokan alamat IP
- Pencocokan prefiks
- Pencocokan akhiran

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 atautennis):

```
{
   "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"
}
```

quals-ignore-case Pencocokan E

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

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

> **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 konsol Amazon SNS.
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 editor JSON, berikan isi JSON dari kebijakan filter Anda.
7. Pilih Simpan perubahan.

   Amazon SNS menerapkan kebijakan filter Anda untuk berlangganan.

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 kebijakan JSON Anda.

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

Untuk memberikan JSON yang valid untuk kebijakan Anda, lampirkan nama atribut dan nilai-nilai 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 nama dan nilai JSON, 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 SDK

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributes`.

> **Important**
>
> Jika Anda menggunakan contoh SDK for Java 2.x, `SNSMessageFilterPolicy` kelas tidak 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 langganan SQS.

```
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 langganan SQS.

```
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 langganan SQS.

```
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 detail API, lihat SetSubscriptionAttributesdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat SetSubscriptionAttributesdi Referensi AWS SDK for Java 2.x API.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat SetSubscriptionAttributesdi AWS SDK for Python (Boto3) Referensi API.

API Amazon SNS

Untuk menerapkan kebijakan filter dengan API Amazon SNS, buat permintaan ke tindakan `SetSubscriptionAttributes`. Atur parameter `AttributeName` ke `FilterPolicy`, dan mengatur parameter `AttributeValue` ke kebijakan filter JSON.

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 template JSON atau YAMAL untuk membuat tumpukan. AWS CloudFormation 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.
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 titik akhir HTTP untuk berlangganan topik Anda.

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

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

Menghapus kebijakan filter langganan

Untuk menghentikan penyaringan pesan yang dikirim ke langganan, hapus kebijakan filter langganan dengan menimpanya dengan isi JSON kosong. Setelah Anda menghapus kebijakan, langganan akan menerima setiap pesan yang dipublikasikan.

AWS Management Console

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi, pilih Berlangganan.
3. Pilih langganan dan kemudian pilih Edit.

4. Pada Edit halaman ***Example1-23BC-4567-D890-EF12G3Hij456***, memperluas bagian kebijakan filter langganan.
5. Di bidang editor JSON, menyediakan isi JSON kosong untuk kebijakan filter Anda: `{}`.
6. Pilih Simpan perubahan.

   Amazon SNS menerapkan kebijakan filter Anda untuk berlangganan.

AWS CLI

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

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

API Amazon SNS

Untuk menghapus kebijakan filter dengan API Amazon SNS, buat permintaan ke tindakan `SetSubscriptionAttributes`. Mengatur parameter `AttributeName` ke `FilterPolicy`, dan menyediakan isi JSON kosong untuk parameter `AttributeValue`.

Perlindungan data pesan

Topik

- Apa itu perlindungan data pesan?
- Mengapa saya harus menggunakan perlindungan data pesan?
- Memahami kebijakan perlindungan data
- Pengidentifikasi data

Apa itu perlindungan data pesan?

Perlindungan data pesan melindungi data yang dipublikasikan ke topik Amazon SNS 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 identitas pribadi (PII) dan informasi kesehatan yang dilindungi (PHI) menggunakan pengidentifikasi data. Anda dapat memilih untuk menggunakan pengidentifikasi data yang telah ditentukan (atau dikelola Amazon SNS) (misalnya, nama, alamat, nomor kartu kredit, dan kode obat resep), atau Anda dapat membuat pengidentifikasi data khusus 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 topik Amazon SNS. 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 hanya untuk topik standar Amazon SNS.

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 seperti HIPAA, GDPR, PCI, 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 hanya untuk topik standar Amazon SNS.

Memahami kebijakan perlindungan data

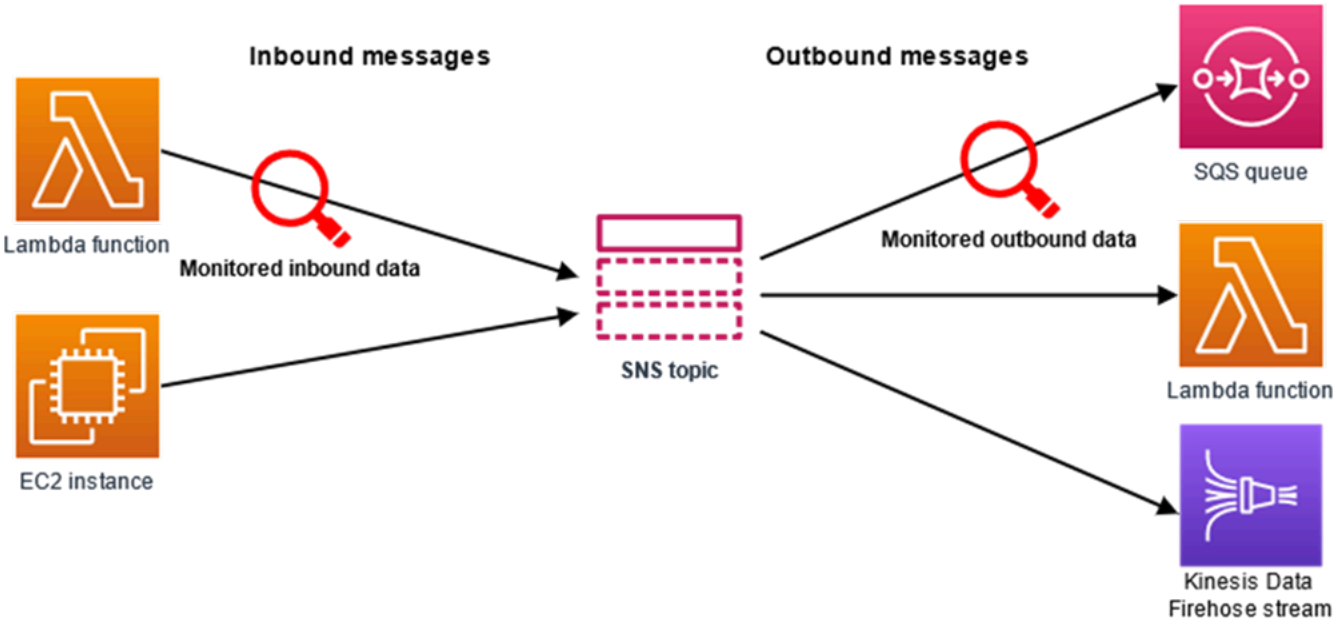
Topik

- Apa itu kebijakan perlindungan data?
- Bagaimana kebijakan perlindungan data terstruktur?
- Bagaimana cara menentukan prinsip IAM untuk kebijakan perlindungan data saya?
- Operasi kebijakan perlindungan data
- Contoh kebijakan perlindungan data
- Membuat kebijakan perlindungan data

- 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 topik Amazon SNS Anda. Untuk memilih data sensitif yang menarik, Anda menggunakan pengidentifikasi data. Perlindungan data pesan Amazon SNS kemudian mendeteksi data sensitif dengan menggunakan pembelajaran mesin dan pencocokan pola. Untuk menindaklanjuti 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.

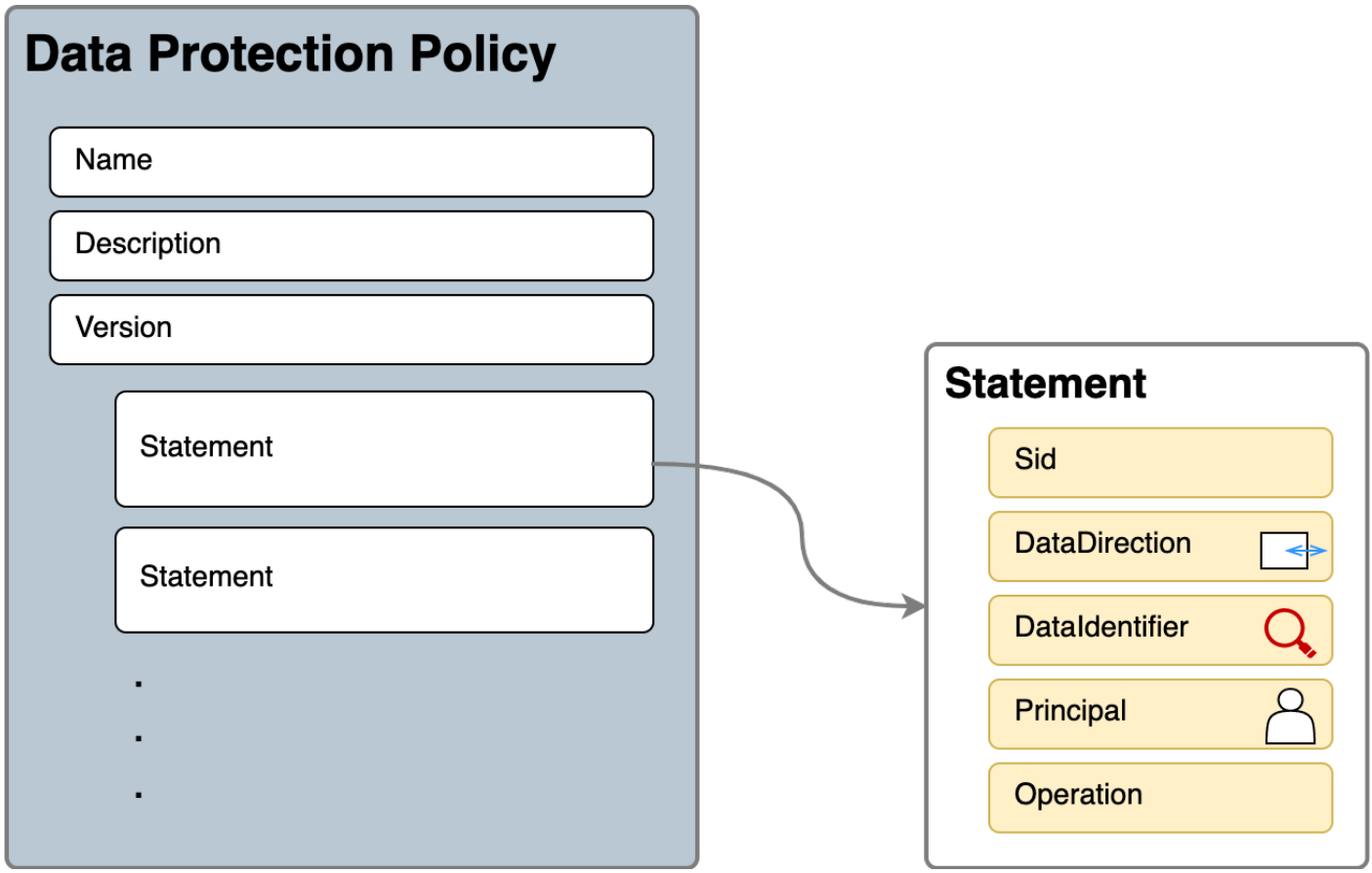


Bagaimana kebijakan perlindungan data terstruktur?

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

Properti JSON 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": [
        ...
  ]
}
```

Properti JSON untuk pernyataan kebijakan

Pernyataan kebijakan menetapkan konteks deteksi untuk operasi perlindungan data.

- Sid (Opsional) - Pengidentifikasi pernyataan.
- DataDirection— Inbound (untuk permintaan Publish API) atau Outbound (untuk pengiriman notifikasi) sehubungan dengan topik Amazon SNS.
- DataIdentifier— Data sensitif yang harus dipindai oleh topik Amazon SNS. Misalnya, nama, alamat, atau nomor telepon.
- Principal — Kepala IAM yang diterbitkan untuk topik, atau kepala IAM yang berlangganan topik.
- Operasi — Tindakan tindak lanjut, baik Audit, De-identify (mask atau redact), atau Deny (block), yang dijalankan oleh topik Amazon SNS 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": {
        ...
    }
}
```

Properti JSON 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 publikasi Amazon SNS atau gagal pengiriman pesan.

Bagaimana cara menentukan prinsip IAM untuk kebijakan perlindungan data saya?

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

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

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

```
{
  "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

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 blog Amazon SNS.

Topik

- Operasi audit

- De-identifikasi operasi
- Tolak operasi

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 topik Amazon SNS menemukan data sensitif di payload.
2. NoFindingsDestination— Tujuan pencatatan saat topik Amazon SNS 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 enkripsi SSE-KMS 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 identitas IAM dari prinsipal IAM yang memanggil Amazon SNS `PutDataProtectionPolicy` API, atau API dengan parameter. `CreateTopic --data-protection-policy`

Tujuan audit	Izin IAM
Default	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups

Tujuan audit	Izin IAM
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Tindakan tambahan diperlukan untuk menggunakan bucket Amazon S3 dengan enkripsi SSE-KMS 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 (SSE-KMS). AWS KMS keys 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

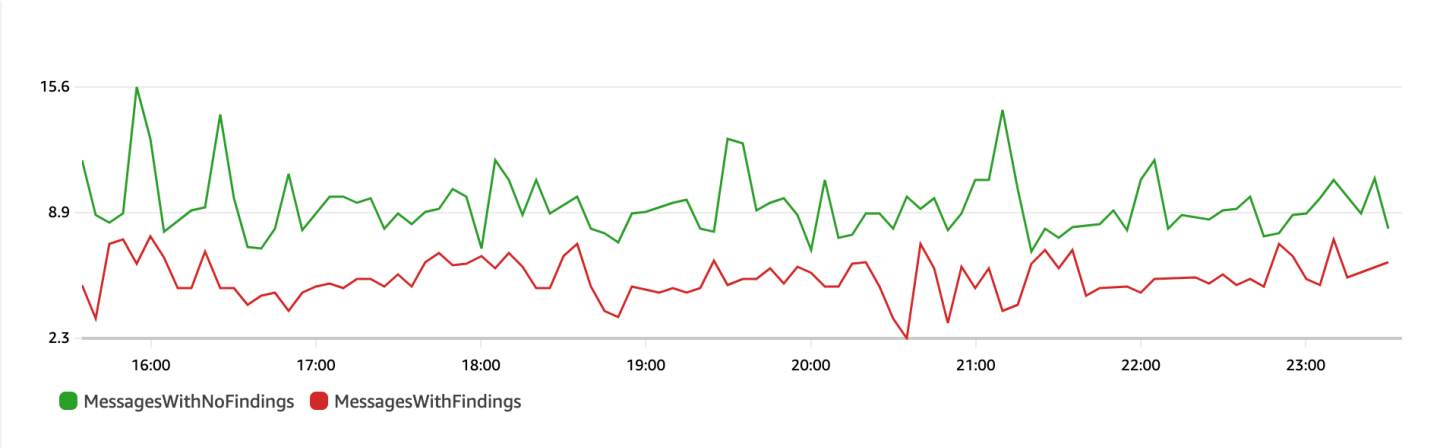
Contoh log tujuan audit

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

```
{
  "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 pemanggil `SNS:Publish` API menerima kesalahan parameter tidak valid berikut ketika seluruh pesan sensitif.

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

Tolak operasi

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

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

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

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

Pada pesan keluar, topik Amazon SNS 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 data

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 blog Amazon SNS.

Topik

- Contoh kebijakan untuk audit
- Contoh kebijakan dengan pernyataan topeng de-identifikasi masuk
- 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 masuk

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. Payload yang ditolak dalam respons API 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

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

Topik

- Membuat kebijakan perlindungan data untuk mengamankan data pesan (API)
- Membuat kebijakan perlindungan data untuk mengamankan data pesan (CLI)
- Membuat kebijakan perlindungan data untuk mengamankan data pesan (CloudFormation)
- Membuat kebijakan perlindungan data untuk mengamankan data pesan (Konsol)
- Membuat kebijakan perlindungan data untuk mengamankan data pesan (SDK)

Membuat kebijakan perlindungan data untuk mengamankan data pesan (API)

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

Membuat kebijakan perlindungan data (AWSAPI)

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

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

Gunakan `DataProtectionPolicy` properti topik Amazon SNS standar:

- `CreateTopic`

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

Panggil salah satu operasi berikut ini:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Membuat kebijakan perlindungan data untuk mengamankan data pesan (CLI)

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

Membuat kebijakan perlindungan data (AWS CLI)

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

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

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

- buat-topik

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

Panggil salah satu operasi berikut ini:

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

Membuat kebijakan perlindungan data untuk mengamankan data pesan (CloudFormation)

Jumlah dan ukuran sumber daya Amazon SNS 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 data Amazon SNS menggunakan. AWS CloudFormation

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

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

- AWS::SNS::Topic

Membuat kebijakan perlindungan data untuk mengamankan data pesan (Konsol)

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

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

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

1. Masuk ke Konsol Amazon SNS.
2. Pilih topik atau buat yang baru. Untuk detail selengkapnya tentang membuat topik, lihat Membuat topik Amazon SNS.
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 khusus menggunakan JSON.

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 dalam panggilan API. 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 prinsip IAM tempat pernyataan de-identifikasi ini berlaku. Anda dapat menerapkannya ke semua AWS akun, atau ke AWSakun tertentu atau entitas IAM (akar akun, peran, atau pengguna) yang menggunakan ID akun atau ARN entitas IAM. Pisahkan beberapa ID atau ARN menggunakan koma (,).

Prinsipal IAM berikut didukung:

- Prinsipal akun IAM — Misalnya,. `arn:aws:iam::AWS-account-ID:root`
- Prinsip peran IAM — Misalnya,. `arn:aws:iam::AWS-account-ID:role/role-name`

- Prinsip pengguna IAM — 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 prinsip IAM yang berlaku untuk pernyataan penolakan ini. Anda dapat menerapkannya ke semua AWS akun, ke entitas tertentu Akun AWS, atau IAM (misalnya, akar akun, peran, atau pengguna) yang menggunakan ID akun atau ARN entitas IAM. Pisahkan beberapa ID atau ARN menggunakan koma (,). Prinsipal IAM berikut didukung:

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

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

Membuat kebijakan perlindungan data untuk mengamankan data pesan (SDK)

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

Membuat kebijakan perlindungan data (AWSSDK)

Anda dapat membuat kebijakan perlindungan data Amazon SNS menggunakan SDKAWS.

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

Gunakan opsi berikut untuk membuat kebijakan perlindungan data baru bersama dengan topik Amazon SNS 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 topik AWS Amazon SNS (SDK) yang ada

Gunakan opsi berikut untuk membuat atau mengambil kebijakan perlindungan data baru bersama dengan topik Amazon SNS 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 data Amazon SNS menggunakan AWS API,, AWS CLIAWS CloudFormation, atau. AWS Management Console

Untuk informasi umum tentang kebijakan perlindungan data Amazon SNS, lihat. Memahami kebijakan perlindungan data

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

Topik

- Menghapus kebijakan perlindungan data (Konsol)
- Menghapus kebijakan perlindungan data menggunakan string JSON kosong
- Menghapus kebijakan perlindungan data menggunakan AWS CLI

Menghapus kebijakan perlindungan data (Konsol)

Untuk menghapus kebijakan perlindungan data terkelola (Konsol)

1. Masuk ke Konsol Amazon SNS.
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 Save changes (Simpan perubahan).

Menghapus kebijakan perlindungan data menggunakan string JSON kosong

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

Menghapus kebijakan perlindungan data menggunakan AWS CLI

Anda dapat menghapus kebijakan perlindungan data menggunakanAWS CLI.

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

Pengidentifikasi data

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 terkelola Amazon SNS 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?
- Tipe data sensitif: Kredensyal
- Tipe data sensitif: Perangkat
- Tipe data sensitif: Keuangan
- Jenis data sensitif: Informasi kesehatan yang dilindungi (PHI)
- Tipe data sensitif: Informasi identitas pribadi (PII)

Apa itu pengidentifikasi data terkelola?

Pengidentifikasi data terkelola Amazon SNS 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 guna 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 alamat MAC
- Informasi keuangan, seperti nomor kartu kredit
- Informasi Kesehatan, untuk PHI seperti asuransi kesehatan atau nomor identifikasi medis
- Informasi pribadi, untuk 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 yang berdekatan dengan 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.

Pengidentifikasi data terkelola Amazon SNS untuk tipe data sensitif

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

Pengidentifikasi data yang bergantung pada wilayah memerlukan nama pengenal dengan tanda hubung, dan kode dua huruf (ISO 3166-1 alpha-2). Misalnya, DriversLicense -US.

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

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

Pengenal yang Didukung yang independen bahasa/wilayah

Pengidentifikasi	Kategori
Alamat	Personal
AwsSecretKey	Kredensial
CreditCardExpiration	Keuangan
CreditCardNumber	Keuangan
CreditCardSecurityCode	Keuangan
EmailAddress	Personal
IpAddress	Personal
LatLong	Personal
Nama	Personal
OpenSshPrivateKey	Kredensial
PgpPrivateKey	Kredensial

Pengidentifikasi	Kategori
PkcsPrivateKey	Kredensial
PuttyPrivateKey	Kredensial
VehicleIdentificationNumber	Personal

Tipe data sensitif: Kredensyal

Tabel berikut mencantumkan dan menjelaskan jenis kredensil yang dapat dideteksi Amazon SNS menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
AWS secret access key	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Sebarang
Kunci pribadi OpenSSH	OpenSshPrivateKey	Tidak	Sebarang
Kunci pribadi PGP	PgpPrivateKey	Tidak	Sebarang
Kunci pribadi Standar Kriptografi Kunci Publik (PKCS)	PkcsPrivateKey	Tidak	Sebarang
Kunci pribadi PuTTY	PuttyPrivateKey	Tidak	Sebarang

ARN pengenal data untuk tipe data kredensyal

Berikut ini mencantumkan Nama Sumber Daya Amazon (ARN) untuk pengidentifikasi data yang dapat Anda tambahkan ke kebijakan perlindungan data Anda.

ARN pengidentifikasi data kredensyal
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

Tipe data sensitif: Perangkat

Tabel berikut mencantumkan dan menjelaskan jenis pengidentifikasi perangkat yang dapat dideteksi Amazon SNS menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
Alamat IP	IpAddress	Tidak	Sebarang

ARN pengenal data untuk tipe data perangkat

Berikut ini mencantumkan Nama Sumber Daya Amazon (ARN) untuk pengidentifikasi data yang dapat Anda tambahkan ke kebijakan perlindungan data Anda.

Pengidentifikasi data perangkat ARN
arn:aws:dataprotection: :aws:data-identifier/ IpAddress

Tipe data sensitif: Keuangan

Tabel berikut mencantumkan dan menjelaskan jenis informasi keuangan yang dapat dideteksi Amazon SNS 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 (IBAN) 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	–	Sebarang
Data strip magnetik kartu kredit	CreditCardMagneticStripe	Ya, termasuk: data kartu, iso7813, mag, magstripe, stripe, swipe.	Hal ini mencakup lintasan 1 dan 2.	Sebarang
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	Sebarang

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	–	Sebarang

1. Amazon SNS tidak melaporkan kemunculan 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 (IBAN) 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			

ARN pengenal data untuk tipe data keuangan

Berikut ini mencantumkan Nama Sumber Daya Amazon (ARN) untuk pengidentifikasi data yang dapat Anda tambahkan ke kebijakan perlindungan data Anda.

ARN pengidentifikasi data keuangan
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

ARN pengidentifikasi data keuangan
arn:aws:dataprotection: :aws:data-identifier/ CreditCardNumber
arn:aws:dataprotection: :aws:data-identifier/ CreditCardSecurityCode

Jenis data sensitif: Informasi kesehatan yang dilindungi (PHI)

Tabel berikut mencantumkan dan menjelaskan jenis informasi kesehatan yang dilindungi (PHI) yang dapat dideteksi Amazon SNS menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
Nomor Pendaftaran Drug Enforcement Agency (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 sjukförsäkrsärsskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione number, versicherungsnummer	
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 Pemeliharaan 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 Layanan 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 memerlukan kata kunci untuk berada di dekat angka-angka tersebut. Ini termasuk nomor European Health Insurance Card (EU, Finland), nomor asuransi kesehatan (France), Medicare Beneficiary Identifiers (US), nomor Asuransi Nasional (UK), nomor NHS (UK), dan Nomor Kesehatan Pribadi (Canada).

Tabel berikut mencantumkan kata kunci yang diakui Amazon SNS 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

ARN pengidentifikasi data untuk tipe data informasi kesehatan yang dilindungi (PHI)

Berikut ini mencantumkan pengenal data Amazon Resource Names (ARN) yang dapat digunakan dalam kebijakan perlindungan data PHI.

ARN pengidentifikasi data PHI
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

Tipe data sensitif: Informasi identitas pribadi (PII)

Tabel berikut mencantumkan dan menjelaskan jenis informasi identitas pribadi (PII) yang dapat dideteksi oleh 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 (-).	Sebarang
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 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 individu	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 (DNI) (Spanyol), kode fiscale 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 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	–	Canada

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	–	Canada
Registrasi Geral (RG)	RgNumber	Ya, lihatKata kunci untuk nomor induk kependudukan.	–	Brazil
Nomor Pokok Wajib Pajak (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Canada

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, socialsecurityno #, 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); Steueridentifikationsnummer (Jerman); CIF (Spanyol); dan TRN, UTR (Inggris).	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	–	Sebarang

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Koordinat Global Positioning System (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS dapat mendeteksi koordinat GPS 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 Degrees Decimal Minutes (DDM), misalnya format 41° 56.9168'N 87° 39.3187'W, atau Derajat, Menit, Detik (DMS), misalnya 41° 56'55.0104 “N 87° 39'19.119 6"W.	Sebarang

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nama lengkap	Name	No	Amazon SNS hanya dapat mendeteksi nama lengkap. Dukungan terbatas pada set karakter Latin.	Sebarang
Nomor identifikasi Mesin (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 VIN yang terdiri dari urutan 17 karakter dan mematuhi standar ISO 3779 dan 3780. Standar ini dirancang untuk penggunaan di seluruh dunia.	Sebarang

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 diakui Amazon SNS 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 memerlukan kata kunci untuk berada di dekat angka. Hal ini termasuk pengenal Documento Nacional de Identidad (DNI) (Spain), kode French National Institute for Statistics and Economic Studies (INSEE), nomor German National Identity Card, dan nomor Registro Geral (RG) (Brazil).

Tabel berikut mencantumkan kata kunci yang diakui Amazon SNS 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 nomor. Tabel berikut mencantumkan kata kunci yang diakui Amazon SNS 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 diakui Amazon SNS 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.

ARN pengidentifikasi data untuk informasi identitas pribadi (PII)

Tabel berikut mencantumkan Nama Sumber Daya Amazon (ARN) untuk pengidentifikasi data yang dapat ditambahkan ke kebijakan perlindungan data Anda.

ARN pengidentifikasi data PII
arn:aws:dataprotection: :AWS:data-identifier/alamat
arn:aws:dataprotection: :aws:data-identifier/ -BR CepCode
arn:aws:dataprotection: :AWS:Pengenal data/CNPJ-BR

ARN pengidentifikasi data PII
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

ARN pengidentifikasi data PII
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

ARN pengidentifikasi data PII
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

ARN pengidentifikasi data PII
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

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 (CDI) 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 ID karyawan khusus perusahaan. Pengidentifikasi data khusus dapat digunakan bersama dengan pengidentifikasi data terkelola.

Menggunakan pengidentifikasi data khusus dalam kebijakan perlindungan data Anda

Kebijakan perlindungan data berikut menginstruksikan topik Amazon SNS untuk mendeteksi muatan yang membawa ID karyawan khusus perusahaan, lalu menutupi ID ini 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 khusus Amazon SNS 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: ('_' | '-')
- RegExmemiliki panjang maksimum 200 karakter. Karakter berikut didukung:
  - Alfanumerik: (A-za-Z0-9)
  - Simbol: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | '')
  - RegExkarakter 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 topik Amazon SNS.

Pengiriman pesan Amazon SNS

Bagian ini menjelaskan cara kerja pengiriman pesan.

Topik

- Pengiriman pesan mentah Amazon SNS
- Mengirim olahpesan Amazon SNS ke antrean Amazon SQS di akun yang berbeda
- Mengirim pesan Amazon SNS ke antrean Amazon SQS atau fungsi AWS Lambda di Wilayah yang berbeda
- Status pengiriman pesan Amazon SNS
- Pengiriman ulang pesan Amazon SNS
- Antrean surat mati Amazon SNS (DLQs)

Pengiriman pesan mentah Amazon SNS

Untuk menghindari Amazon Data Firehose, Amazon SQS, dan titik akhir HTTP/S memproses pemformatan pesan JSON, Amazon SNS memungkinkan pengiriman pesan mentah:

- Saat Anda mengaktifkan pengiriman pesan mentah untuk Amazon Data Firehose atau titik akhir Amazon SQS, metadata Amazon SNS apa pun akan dilucuti dari pesan yang dipublikasikan dan pesan dikirim apa adanya.
- Ketika Anda mengaktifkan pengiriman pesan mentah untuk titik akhir HTTP/S, header HTTP `x-amz-sns-rawdelivery` dengan nilainya diatur ke `true` ditambahkan ke pesan, menunjukkan bahwa pesan telah diterbitkan tanpa format JSON.
- Saat Anda mengaktifkan pengiriman pesan mentah untuk titik akhir HTTP/S, badan 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 tindakan `SetSubscriptionAttribute` API dan menetapkan nilai `RawMessageDelivery` atribut ke`true`.

Mengaktifkan pengiriman pesan mentah menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi, pilih Topik.
3. Pada halaman Topik, pilih topik yang berlangganan Firehose, Amazon SQS, atau titik akhir HTTP/S.
4. Pada ***MyTopic***halaman, di bagian Berlangganan, pilih langganan dan pilih Edit.
5. Di halaman Edit ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***, di bagian Details (Detail), pilih Enable raw message delivery (Aktifkan pengiriman pesan mentah).
6. Pilih Save changes (Simpan perubahan).

Contoh format pesan

Dalam contoh berikut, pesan yang sama dikirim ke antrean Amazon SQS 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 langganan Amazon SQS

Amazon SNS mendukung pengiriman atribut pesan, yang memungkinkan Anda menyediakan item metadata terstruktur, seperti stempel waktu, data geospasial, tanda tangan, dan pengidentifikasi, tentang pesan. Untuk langganan Amazon SQS 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 langganan Amazon SQS dengan Pengiriman Pesan Mentah diaktifkan, memperlakukannya sebagai kesalahan sisi klien.

Mengirim olahpesan Amazon SNS ke antrean Amazon SQS di akun yang berbeda

Dokumen ini menjelaskan cara mempublikasikan pemberitahuan ke topik Amazon SNS dengan satu atau beberapa langganan ke antrian Amazon SQS di akun lain. Atur topik dan antrean dengan cara yang sama jika mereka berada di akun yang sama (lihat Fanout ke antrean Amazon SQS). 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 antrian Amazon SQS 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 antrean Amazon SQS adalah pemilik antrean. 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 konsol Amazon SNS.
2. Di panel navigasi, pilih Topics (Topik).
3. Pilih topik, kemudian pilih Edit (Edit).
4. Pada *`MyTopic`*halaman Edit, 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 pengguna/peran IAM mereka. Hanya akun root atau pengguna administrator yang diizinkan untuk meneleponsns:Subscribe. Pengguna/peran IAM 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 antrean Amazon SQS ke topik di akun Akun AWS lainnya menggunakan AWS Management Console

Sebelum memulai, pastikan Anda memiliki ARN untuk topik dan antrian Anda, dan bahwa Anda telah memberikan izin untuk topik untuk mengirim pesan ke antrian.

1. Masuk ke konsol Amazon SQS.
2. Pada panel navigasi, pilih Antrian.
3. Dari daftar antrian, pilih antrian untuk berlangganan topik Amazon SNS.
4. Pilih Berlangganan topik Amazon SNS.
5. Dari Tentukan topik Amazon SNS yang tersedia untuk menu antrian ini, pilih topik Amazon SNS untuk antrian Anda.
6. Pilih Masukkan topik Amazon SNS ARN dan kemudian masukkan nama sumber daya Amazon (ARN) topik.
7. Pilih Save (Simpan).

> **Note**
>
> - Untuk dapat berkomunikasi dengan layanan, antrean 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.

Ketika Anda menggunakan tindakan `Subscribe`, Amazon SNS mengirimkan konfirmasi langganan ke antrean. Langganan ditampilkan di konsol Amazon SNS, dengan ID langganan diatur ke Pending Confirmation (Menunggu Konfirmasi).

Untuk mengonfirmasi langganan, pengguna dengan izin untuk membaca pesan dari antrian harus mengambil URL konfirmasi langganan, dan pemilik langganan harus mengonfirmasi langganan menggunakan URL konfirmasi langganan. Tidak ada notifikasi yang dipublikasikan ke topik yang dikirim ke antrean sampai langganan dikonfirmasi. Untuk mengonfirmasi langganan, Anda dapat menggunakan konsol Amazon SQS atau tindakan `ReceiveMessage`.

> **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 untuk topik Amazon SNS untuk mengirim pesan ke antrean Amazon SQS.

Langkah 1: Untuk menambahkan langganan antrian Amazon SQS ke topik lain menggunakan Akun AWSAWS Management Console

Sebelum memulai, pastikan Anda memiliki ARN untuk topik dan antrian Anda, dan bahwa Anda telah memberikan izin untuk topik untuk mengirim pesan ke antrian.

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi, pilih Berlangganan.
3. Di halaman Subscriptions (Langganan), pilih Create subscription (Buat langganan).
4. Di halaman Create subscription (Buat langganan), di bagian Details (Detail), lakukan:
   a. Untuk ARN Topik, masukkan ARN topik.
   b. Untuk Protokol, pilih Amazon SQS.
   c. Untuk Endpoint, masukkan ARN antrean.

d. Pilih Create subscription (Buat langganan).

> **Note**
>
> - Untuk dapat berkomunikasi dengan layanan, antrean harus memiliki izin untuk Amazon SNS.

Berikut ini adalah contoh pernyataan kebijakan yang memungkinkan topik Amazon SNS mengirim pesan ke antrian Amazon SQS.

```
{
   "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 konsol Amazon SQS.
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 Rincian Pesan, temukan dan catat nilai SubscribeUrl. Ini adalah tautan langganan Anda (contoh di bawah). Untuk detail tambahan tentang validasi token API, lihat `ConfirmSubscription`di Referensi API Amazon SNS.

```
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. URL harus diteruskan dari pemilik antrian ke pemilik langganan. Pemilik langganan harus memasukkan URL ke konsol Amazon SNS.

5. Masuk sebagai pemilik langganan ke konsol Amazon SNS 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.

    Respons XML 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 topik langganan di konsol Amazon SNS, Anda dapat melihat bahwa olahpesan Pending Confirmation (Menunggu Konfirmasi) telah digantikan oleh langganan ARN di kolom Subscription ID (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 disetel ke true saat pemilik antrian membuat langganan.
- `AuthenticateOnUnsubscribe`tidak dapat disetel ke true saat tautan konfirmasi langganan dinavigasi tanpa autentikasi.

Mengirim pesan Amazon SNS ke antrean Amazon SQS atau fungsi AWS Lambda di Wilayah yang berbeda

Amazon SNS mendukung pengiriman lintas wilayah, baik untuk Wilayah yang diaktifkan secara default dan untuk Wilayah keikutsertaan. Untuk daftar AWS Wilayah saat ini yang didukung Amazon SNS, termasuk Wilayah keikutsertaan, lihat titik akhir dan kuota Layanan Pemberitahuan Sederhana Amazon di.Referensi Umum Amazon Web Services

Amazon SNS mendukung pengiriman lintas wilayah notifikasi ke antrean Amazon SQS dan ke fungsi AWS Lambda . Ketika salah satu Wilayah adalah wilayah keikutsertaan, Anda harus menentukan layanan utama Amazon SNS yang berbeda dalam kebijakan sumber daya berlangganan.

Perintah langganan Amazon SNS harus dijalankan di akun target di Wilayah tempat Amazon SNS di-host. Misalnya, jika Amazon SNS ada di akun “A” di wilayah us-timur-1, dan fungsi Lambda ada di akun “B” di wilayah us-timur-2, perintah CLI langganan harus dijalankan di akun “A” di wilayah us-timur-1.

Wilayah Keikutsertaan

Amazon SNS mendukung berikut Wilayah keikutsertaan:

Nama wilayah	Wilayah
Wilayah Afrika (Cape Town)	af-south-1
Wilayah Asia Pasifik (Hong Kong)	ap-east-1

Nama wilayah	Wilayah
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
Wilayah Timur Tengah (UEA)	me-central-1

Untuk informasi tentang mengaktifkan Wilayah keikutsertaan, lihat Mengelola AWS Wilayah di.Referensi Umum Amazon Web Services

Ketika Anda menggunakan Amazon SNS untuk mengirim pesan dari Wilayah keikutsertaan untuk 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 antrean Amazon SQS di US East (Virginia N.) ke topik Amazon SNS 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 topik Amazon SNS di Wilayah keikutsertaan (af-selatan-1) untuk dikirimkan ke antrean Amazon SQS di Wilayah (us-timur-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 US East (Virginia N.) ke topik Amazon SNS di Asia Pasifik (Hong Kong), ubah prinsip kebijakan fungsi AWS Lambda 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 pesan Amazon SNS

Amazon SNS menyediakan dukungan untuk mencatat status pengiriman pesan notifikasi dikirim ke topik dengan titik akhir Amazon SNS 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. Pencatatan status pengiriman pesan membantu memberikan wawasan operasional yang lebih baik, seperti berikut ini:

- Mengetahui apakah pesan dikirim ke titik akhir Amazon SNS.
- Mengidentifikasi respon yang dikirim dari titik akhir Amazon SNS ke Amazon SNS.
- Menentukan waktu tinggal pesan (waktu antara mempublikasikan timestamp 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 (SDK), API kueri, atau. AWS CloudFormation

Topik

- Mengkonfigurasi pencatatan status pengiriman menggunakan AWS Management Console
- Mengonfigurasi pencatatan status pengiriman menggunakan SDK AWS
- AWS Contoh SDK untuk mengonfigurasi atribut topik
- Mengkonfigurasi pencatatan status pengiriman menggunakan AWS CloudFormation

Mengkonfigurasi pencatatan status pengiriman menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih topik dan kemudian pilih Edit (Edit).
4. Pada *MyTopic*halaman Edit, perluas bagian Pencatatan status pengiriman.
5. Pilih protokol yang ingin Anda catat status pengiriman; misalnya AWS Lambda.

6. Masukkan tingkat sampel Sukses (persentase pesan sukses yang ingin Anda terima CloudWatch Log).
7. Di bagian IAM role, lakukan salah satu hal berikut:

   - Untuk memilih peran layanan yang ada dari akun Anda, pilih Gunakan peran layanan yang ada dan kemudian menentukan IAM role untuk pengiriman sukses dan gagal.

   - Untuk membuat peran layanan baru di akun Anda, pilih Buat peran layanan baru, pilih Buat peran baru untuk menentukan peran IAM untuk pengiriman sukses dan gagal di konsol IAM.

     Untuk memberikan akses tulis Amazon SNS 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.

Mengonfigurasi pencatatan status pengiriman menggunakan SDK AWS

AWS SDK menyediakan API dalam beberapa bahasa untuk menggunakan atribut status pengiriman pesan dengan Amazon SNS.

Atribut topik

Anda dapat menggunakan nilai nama atribut topik berikut untuk status pengiriman pesan:

HTTP

- `HTTPSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk topik Amazon SNS yang berlangganan titik akhir HTTP.
- `HTTPSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk topik Amazon SNS yang berlangganan titik akhir HTTP.
- `HTTPFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan gagal untuk topik Amazon SNS yang berlangganan titik akhir HTTP.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk topik Amazon SNS yang berlangganan titik akhir Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk diambil sampel untuk topik Amazon SNS yang berlangganan titik akhir Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang gagal untuk topik Amazon SNS yang berlangganan titik akhir Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk topik Amazon SNS yang berlangganan titik akhir Lambda.
- `LambdaSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk topik Amazon SNS yang berlangganan titik akhir Lambda.
- `LambdaFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan gagal untuk topik Amazon SNS yang berlangganan titik akhir Lambda.

Titik akhir aplikasi platform

- `ApplicationSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk topik Amazon SNS yang berlangganan titik akhir aplikasi AWS .
- `ApplicationSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk topik Amazon SNS yang berlangganan titik akhir aplikasi AWS .
- `ApplicationFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang gagal untuk topik Amazon SNS yang berlangganan titik akhir aplikasi AWS .

> **Note**
>
> Selain mampu mengkonfigurasi atribut topik untuk status pengiriman pesan notifikasi dikirim ke titk akhir Amazon SNS aplikasi, Anda juga dapat mengkonfigurasi atribut aplikasi untuk status pengiriman pesan notifikasi push dikirim ke layanan notifikasi push. Untuk informasi selengkapnya, lihat Menggunakan Atribut Aplikasi Amazon SNS untuk Status Pengiriman Pesan.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk topik Amazon SNS yang berlangganan titik akhir Amazon SQS.
- `SQSSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk topik Amazon SNS yang berlangganan titik akhir Amazon SQS.
- `SQSFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan gagal untuk topik Amazon SNS yang berlangganan titik akhir Amazon SQS.

> **Note**
>
> `<ENDPOINT>FailureFeedbackRoleArn`Atribut `<ENDPOINT>SuccessFeedbackRoleArn` dan digunakan untuk memberikan akses tulis Amazon SNS 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 Contoh SDK untuk mengonfigurasi 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 detail API, lihat SetTopicAttributesdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat SetTopicAttributesdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat SetTopicAttributesdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat SetTopicAttributesdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat SetTopicAttributesdi Referensi AWS SDK for PHP API.

Ruby

SDK untuk Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat SetTopicAttributesdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

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

```
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 detail API, lihat SetTopicAttributesdi AWS SDK untuk referensi SAP ABAP API.

Mengkonfigurasi pencatatan status pengiriman menggunakan AWS CloudFormation

Untuk mengonfigurasi `DeliveryStatusLogging` penggunaan AWS CloudFormation, gunakan template JSON atau YAMAL untuk membuat tumpukan. AWS CloudFormation Untuk informasi selengkapnya, lihat `DeliveryStatusLogging` properti `AWS::SNS::Topic` sumber daya di Panduan AWS CloudFormation Pengguna. Di bawah ini adalah contoh AWS CloudFormation template di JSON dan YAMAL untuk membuat topik baru atau memperbarui topik yang ada dengan semua `DeliveryStatusLogging` atribut untuk protokol Amazon SQS.

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
```

Pengiriman ulang pesan Amazon SNS

Amazon SNS menentukan delivery policy (kebijakan pengiriman) untuk setiap protokol pengiriman. Kebijakan pengiriman menentukan cara Amazon SNS mencoba ulang pengiriman pesan ketika kesalahan sisi server terjadi (ketika sistem yang menghosting endpoint berlangganan menjadi tidak tersedia). Ketika kebijakan pengiriman habis, Amazon SNS berhenti untuk mengirimkan ulang dan membuang pesan-kecuali antrean surat mati dilampirkan ke langganan. Untuk informasi selengkapnya, lihat Antrean surat mati Amazon SNS (DLQs).

Topik

- Protokol dan kebijakan pengiriman
- Tahap kebijakan pengiriman
- Membuat kebijakan pengiriman HTTP/S

Protokol dan kebijakan pengiriman

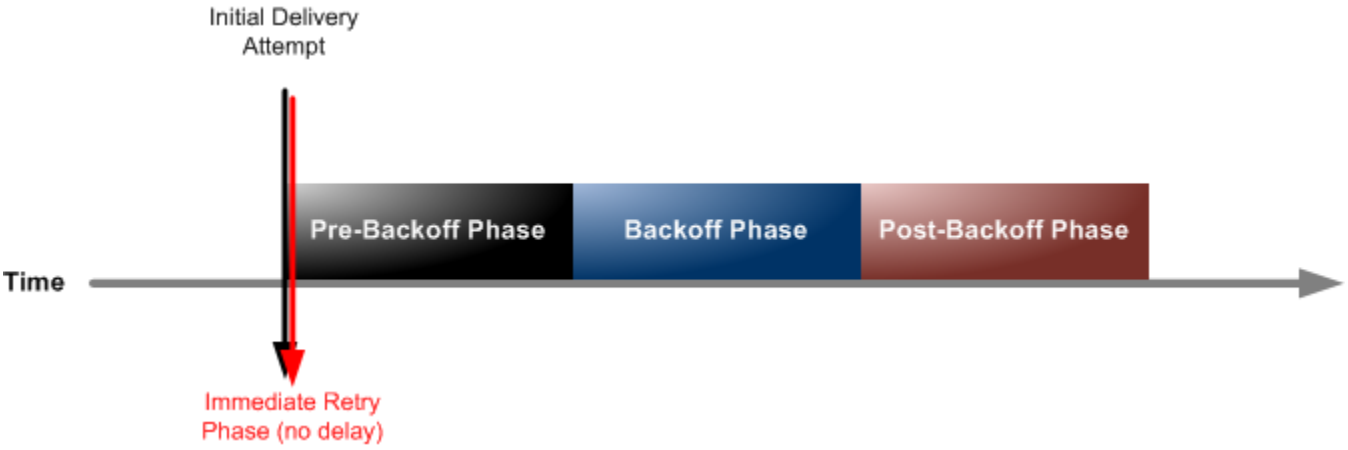
> **Note**
>
> - Dengan pengecualian HTTP/S, Anda tidak dapat mengubah kebijakan pengiriman yang ditentukan Amazon SNS. Hanya HTTP/S mendukung kebijakan kustom. Lihat Membuat kebijakan pengiriman HTTP/S.
> - Amazon SNS menerapkan jittering untuk pengiriman ulang pesan. Untuk informasi selengkapnya, lihat posting Exponential Backoff and Jitter (Backoff Eksponensial dan Jitter) di AWS Architecture Blog (Blog Arsitektur).
> - Total waktu percobaan 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 dikelola pelanggan	SMTP	0 kali, tanpa penundaan	2 kali, 10 detik terpisah	10 kali, dengan backoff eksponensial, dari 10 detik hingga 600 detik (10 menit)	38 kali, 600 detik (10 menit) terpisah	50 percobaan, lebih dari 6 jam
	SMS					
	Push seluler					

[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 pengiriman ulang pesan 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 fase untuk menentukan cara Amazon SNS mencoba ulang pengiriman pesan untuk HTTP/S endpoint. Amazon SNS memungkinkan Anda mengganti kebijakan coba ulang default untuk titik akhir HTTP ketika Anda mungkin, misalnya, ingin menyesuaikan kebijakan berdasarkan kapasitas server HTTP Anda.

Anda dapat mengatur kebijakan pengiriman HTTP/S Anda sebagai objek JSON di tingkat langganan atau topik. Saat menentukan kebijakan di tingkat topik, kebijakan itu berlaku untuk semua langganan HTTP/S yang terkait dengan topik. Untuk menetapkan kebijakan pengiriman di tingkat langganan, Anda dapat menggunakan tindakan `Subscribe`atau `SetSubscriptionAttributes`API. Untuk menyetel kebijakan pengiriman di tingkat topik, Anda dapat menggunakan tindakan `CreateTopic`atau `SetTopicAttributes`API. Atau, Anda juga dapat menggunakan AWS::SNS::Subscriptionsumber daya di AWS CloudFormation template Anda.

Anda harus menyesuaikan kebijakan pengiriman Anda sesuai dengan kapasitas server HTTP/S Anda. Anda dapat mengatur kebijakan sebagai atribut topik atau atribut langganan. Jika semua langganan HTTP/S di topik Anda menargetkan server HTTP/S yang sama, kami sarankan Anda menetapkan kebijakan pengiriman sebagai atribut topik, sehingga tetap berlaku untuk semua HTTP/S langganan dalam topik. Jika tidak, Anda harus membuat kebijakan pengiriman untuk setiap langganan HTTP/S dalam topik Anda, sesuai kapasitas HTTP/S server yang ditargetkan oleh kebijakan.

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 ke. `text/plain; charset=UTF-8` Amazon SNS memungkinkan Anda mengganti kebijakan permintaan default. Lihat tabel di bawah ini untuk dukungan `headerContentType`dan pengekangan.

Objek JSON berikut merupakan kebijakan pengiriman yang menginstruksikan Amazon SNS untuk mencoba lagi 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 membuat total 50 percobaan sebelum menghapus pesan. Untuk menyimpan pesan setelah pengiriman ulang yang ditentukan dalam kebijakan pengiriman habis, konfigurasikan langganan Anda untuk memindahkan pesan yang tidak terkirim ke antrean surat mati (DLQ). Untuk informasi selengkapnya, lihat Antrean surat mati Amazon SNS (DLQs).

> **Note**
>
> Kebijakan pengiriman ini juga menginstruksikan Amazon SNS untuk membatasi (throttle) pengiriman hingga tidak lebih dari 10 per detik, menggunakan properti `maxReceivesPerSecond`. Tingkat self-throttling ini dapat menghasilkan lebih banyak pesan yang diterbitkan (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 mengganti jenis 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
`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

Kebijakan	Deskripsi	Kendala
		• 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.

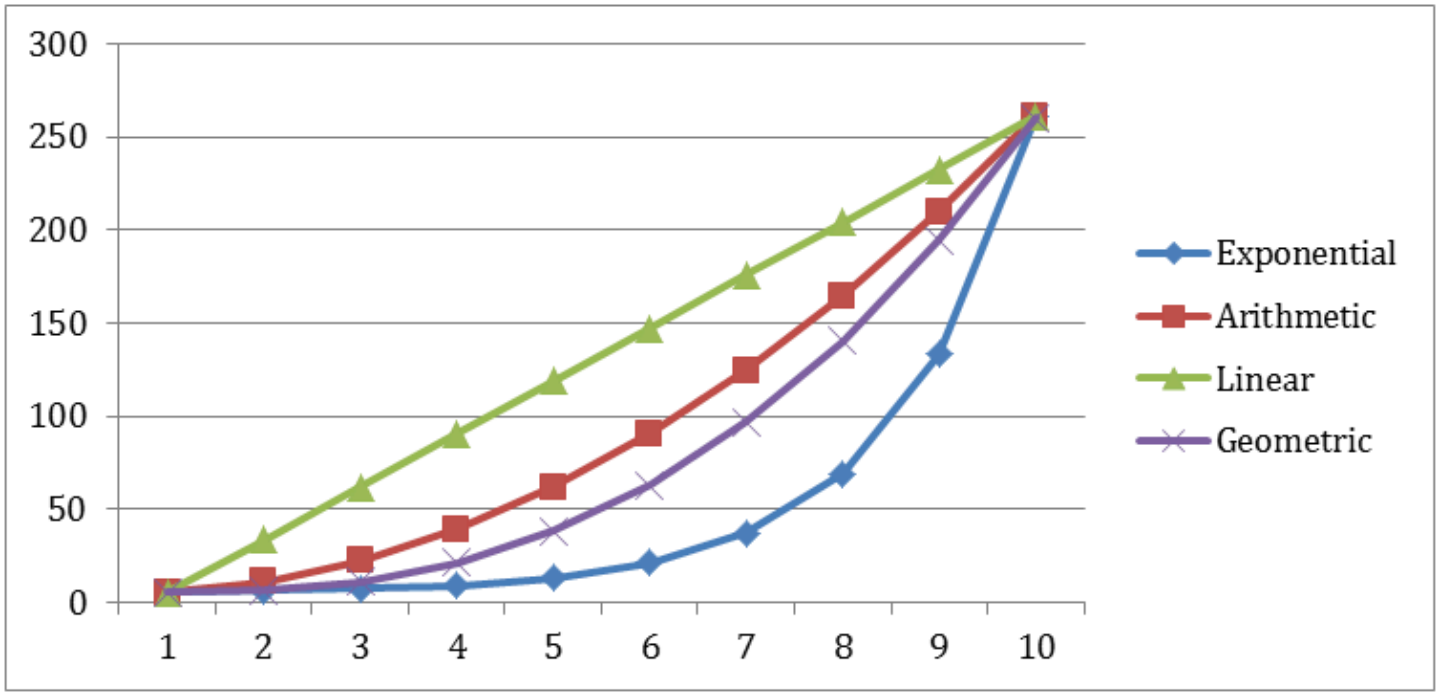
Amazon SNS menggunakan rumus berikut untuk menghitung jumlah pengiriman ulang dalam fase backoff:

```
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` — Menentukan algoritme yang digunakan Amazon SNS untuk menghitung penundaan yang terkait dengan semua pengiriman ulang antara pengiriman ulang pertama dan terakhir dalam 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.



Antrean surat mati Amazon SNS (DLQs)

Antrean surat mati adalah antrean Amazon SQS yang langganan Amazon SNS dapat menargetkan pesan yang tidak dapat dikirim ke pelanggan dengan suskes. 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 lebih lanjut, lihat Mengkonfigurasi antrean surat mati Amazon SNS untuk berlangganan dan Pengiriman ulang pesan Amazon SNS.

Note

- Langganan Amazon SNS dan antrean Amazon SQS harus di bawah akun AWS yang sama dan Wilayah.
- Untuk topik FIFO, Anda dapat menggunakan antrean Amazon SQS sebagai antrian surat mati untuk langganan Amazon SNS. Langganan topik FIFO menggunakan antrian FIFO, dan langganan topik standar menggunakan antrian standar.
- Untuk menggunakan antrean Amazon SQS terenkripsi sebagai antrian huruf mati, Anda harus menggunakan KMS kustom dengan kebijakan kunci yang memberikan akses utama layanan Amazon SNS ke tindakan API. AWS KMS Untuk informasi selengkapnya, lihat

Enkripsi diam di panduan dan Melindungi Data Amazon SQS Menggunakan Server-Side Encryption (SSE) dan AWS KMS di Panduan Developer Amazon Simple Queue Service.

Topik

- Mengapa pengiriman pesan gagal?
- Bagaimana cara kerja antrean surat mati?
- Bagaimana pesan dipindahkan ke antrean surat mati?
- Bagaimana cara memindahkan pesan dari antrean surat mati?
- Bagaimana saya bisa memantau dan mencatat antrean surat mati?
- Mengkonfigurasi antrean surat mati Amazon SNS untuk berlangganan

Mengapa pengiriman pesan gagal?

Secara umum, pengiriman pesan gagal ketika Amazon SNS tidak dapat mengakses titik akhir berlangganan karena sisi klien atau kesalahan sisi server. Ketika Amazon SNS menerima galat sisi klien, atau terus menerima galat sisi server untuk pesan di luar jumlah coba lagi yang ditentukan oleh kebijakan coba lagi yang sesuai, Amazon SNS membuang pesan-kecuali antrean surat mati dilampirkan ke langganan. Pengiriman yang gagal tidak mengubah status langganan Anda. Untuk informasi selengkapnya, lihat Pengiriman ulang pesan Amazon SNS.

Kesalahan sisi klien

Kesalahan sisi klien dapat terjadi ketika Amazon SNS memiliki basi langganan metadata. Kesalahan ini sering terjadi ketika pemilik menghapus endpoint (misalnya, fungsi Lambda berlangganan topik Amazon SNS) atau ketika pemilik perubahan kebijakan yang melekat pada titik akhir berlangganan dengan cara yang mencegah Amazon SNS mengirimkan pesan ke titik akhir. Amazon SNS tidak coba lagi pengiriman pesan yang gagal sebagai akibat dari kesalahan sisi klien.

Kesalahan sisi server

Kesalahan sisi server dapat terjadi ketika sistem yang bertanggung jawab untuk titik akhir berlangganan menjadi tidak tersedia atau mengembalikan pengecualian yang menunjukkan bahwa ia tidak dapat memproses permintaan yang valid dari Amazon SNS. Ketika kesalahan sisi server terjadi, Amazon SNS mencoba kembali pengiriman gagal menggunakan fungsi linear atau backoff eksponensial. Untuk kesalahan sisi server yang disebabkan oleh AWS dikelola titik akhir didukung

oleh Amazon SQS atau AWS Lambda, Amazon SNS mencoba kembali pengiriman hingga 100,015 kali, lebih 23 hari.

Pelanggan dikelola titik akhir (seperti HTTP, SMTP, SMS, atau mobile push) juga dapat menyebabkan kesalahan sisi server. Amazon SNS mencoba kembali pengiriman ke jenis titik akhir juga. Sementara titik akhir HTTP mendukung kebijakan coba lagi yang ditetapkan pelanggan, Amazon SNS menetapkan kebijakan coba lagi pengiriman internal untuk 50 kali lebih dari 6 jam, untuk SMTP, SMS, dan titik akhir mobile push.

Bagaimana cara kerja antrean surat mati?

Antrean surat mati melekat berlangganan Amazon SNS (bukan topik) karena pengiriman pesan terjadi di tingkat berlangganan. Hal ini memungkinkan Anda mengidentifikasi titik akhir target asli untuk setiap pesan dengan lebih mudah.

Sebuah antrean surat mati terkait dengan berlangganan Amazon SNS adalah antrean Amazon SQS biasa. 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 Amazon SQS tindakan `SetQueueAttributes` API. 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 objek JSON yang mengacu pada ARN antrean surat mati. Atribut `deadLetterTargetArn` menentukan ARN tersebut. ARN harus menunjuk ke antrean Amazon SQS di Akun AWS dan Wilayah yang sama sebagai langganan Amazon SNS Anda. Untuk informasi selengkapnya, lihat Mengkonfigurasi antrean surat mati Amazon SNS untuk berlangganan.

Objek JSON berikut adalah kebijakan redrive sampel, dilampirkan ke berlangganan SNS.

```
{
  "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 konsumen Amazon SQS – Atur antrean surat mati Anda sebagai sumber acara untuk fungsi Lambda untuk menguras antrean surat mati Anda.
- Menulis logika konsumen Amazon SQS – Gunakan API Amazon SQS,AWS SDK, atau AWS CLI untuk menulis logika konsumen kustom untuk polling, pengolahan, dan menghapus pesan dalam antrean surat 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 SNS Anda. Semua antrian Amazon SQS memancarkan CloudWatch metrik dengan interval satu menit. Untuk informasi selengkapnya, lihat CloudWatch Metrik yang Tersedia untuk Amazon SQS di Panduan Pengembang Layanan Antrian Sederhana Amazon. Semua langganan Amazon SNS dengan antrian huruf mati juga memancarkan metrik. CloudWatch Untuk informasi selengkapnya, lihat Memantau topik Amazon SNS menggunakan CloudWatch.

Untuk mengetahui aktivitas dalam antrian surat mati, Anda dapat menggunakan metrik dan alarm. CloudWatch Misalnya, ketika Anda mengharapkan antrian huruf mati selalu kosong, Anda dapat membuat CloudWatch alarm untuk metrik. `NumberOfMessagesSent` Anda dapat mengatur ambang batas alarm ke 0 dan menentukan topik Amazon SNS untuk diberitahu ketika alarm berbunyi. Topik Amazon SNS ini dapat mengirimkan notifikasi alarm Anda ke jenis titik akhir apa pun (seperti alamat email, nomor telepon, atau aplikasi halaman seluler).

Anda dapat menggunakan CloudWatch Log untuk menyelidiki pengecualian yang menyebabkan pengiriman Amazon SNS gagal dan pesan dikirim ke antrian surat mati. Amazon SNS dapat mencatat pengiriman yang berhasil dan gagal. CloudWatch Untuk informasi selengkapnya, lihat Status pengiriman pesan Amazon SNS.

Mengkonfigurasi antrean surat mati Amazon SNS untuk berlangganan

Antrean surat mati adalah antrean Amazon SQS yang langganan Amazon SNS dapat menargetkan pesan yang tidak dapat dikirim ke pelanggan dengan suskes. 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 lebih lanjut, lihat Antrean surat mati Amazon SNS (DLQs) dan Pengiriman ulang pesan Amazon SNS.

Halaman ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console, AWS SDK AWS CLI, dan AWS CloudFormation untuk mengonfigurasi antrian huruf mati untuk langganan Amazon SNS.

> **Note**
>
> Untuk topik FIFO, Anda dapat menggunakan antrean Amazon SQS sebagai antrian surat mati untuk langganan Amazon SNS. Langganan topik FIFO menggunakan antrian FIFO, dan langganan topik standar menggunakan antrian standar.

Prasyarat

Sebelum Anda mengonfigurasi antrean surat mati, selesaikan prasyarat berikut:

1. Buat topik Amazon SNS bernama `MyTopic`.
2. Buat antrean Amazon SQS bernama `MyEndpoint`, yang akan digunakan sebagai titik akhir untuk berlangganan Amazon SNS.
3. (Lewati untuk AWS CloudFormation) Berlangganan antrian ke topik.
4. Buat antrean Amazon SQS bernama `MyDeadLetterQueue`, untuk digunakan sebagai antrean surat mati untuk berlangganan Amazon SNS.
5. Untuk memberikan Amazon SNS akses utama ke tindakan Amazon SQS API, mengatur 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 huruf mati untuk langganan Amazon SNS menggunakan SDK AWS
- Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS CLI
- Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan 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 konsol Amazon SQS.
2. Buat antrean Amazon SQS atau menggunakan antrean yang ada dan perhatikan ARN antrean pada tab Detail antrean, misalnya:

   ```
   arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
   ```

3. Masuk ke konsol Amazon SNS.
4. Di panel navigasi, pilih Berlangganan.
5. Pada halaman Berlangganan, pilih langganan yang ada, lalu pilih Edit.
6. Pada Edit halaman ***1234a567-bc89-012d-3e45-6fg7h890123i***, memperluas Bagian kebijakan redrive (antrean surat mati), dan kemudian lakukan hal berikut:

   a. Pilih Diaktifkan.

   b. Tentukan ARN antrean Amazon SQS.
7. Pilih Simpan perubahan.

   Langganan Anda dikonfigurasi untuk menggunakan antrean surat mati.

Untuk mengonfigurasi antrian huruf mati untuk langganan Amazon SNS menggunakan SDK AWS

Sebelum Anda menjalankan contoh ini, pastikan Anda menyelesaikan prasyarat.

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

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK for Java 1.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 AWS Command Line Interface Panduan Pengguna.
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 SNS menggunakan AWS CloudFormation

Sebelum memulai tutorial ini, pastikan Anda menyelesaikan prasyarat.

1. Salin kode JSON 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 AWS CloudFormation konsol.
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 status CREATE_COMPLETE.

Pengarsipan, pemutaran ulang, dan analitik pesan Amazon SNS

Topik standar Amazon SNS 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.

Topik Amazon SNS FIFO 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 pesan dan pemutaran ulang untuk topik FIFO.

Fitur	Topik Standar	Topik FIFO
Pengarsipan pesan	Aliran pengiriman Fanout ke Firehose	Pengarsipan pesan untuk pemilik topik FIFO
Putar ulang pesan	Putar ulang untuk topik standar bukanlah fitur bawaan. Banyak pelanggan membangun sendiri berdasarkan arsip pesan mereka.	Putar ulang pesan untuk pelanggan topik FIFO

Menggunakan Amazon SNS untuk pengiriman pesan application-to-application (A2A)

Bagian ini memberikan informasi tentang penggunaan Amazon SNS untuk application-to-application pengiriman pesan dengan pelanggan.

Topik

- Aliran pengiriman Fanout ke Firehose
- Fanout untuk fungsi Lambda
- Fanout ke antrean Amazon SQS
- Fanout ke HTTP (S) titik akhir
- Fanout ke Alur Fork Peristiwa AWS
- Menggunakan Amazon EventBridge Scheduler dengan Amazon SNS

Aliran pengiriman Fanout ke Firehose

Anda dapat berlangganan aliran pengiriman Amazon Data Firehose ke topik Amazon SNS, yang memungkinkan Anda mengirim pemberitahuan ke titik akhir penyimpanan dan analitik tambahan. Pesan yang dipublikasikan ke topik Amazon SNS 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 mengirimkan notifikasi Amazon SNS ke Amazon Simple Storage Service (Amazon S3), Amazon Redshift, OpenSearch Amazon Service (Layanan), dan ke penyedia layanan pihak OpenSearch 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 topik Amazon SNS, buat aliran pengiriman dengan tujuan indeks OpenSearch Layanan. Anda kemudian dapat berlangganan aliran pengiriman Firehose ke topik Amazon SNS.

Amazon SNS juga mendukung pencatatan status pengiriman pesan untuk pemberitahuan yang dikirim ke titik akhir Firehose. Lihat informasi yang lebih lengkap di Status pengiriman pesan Amazon SNS.

Topik

- Prasyarat untuk berlangganan aliran pengiriman Firehose ke topik Amazon SNS
- Berlangganan aliran pengiriman Firehose ke topik Amazon SNS
- Bekerja dengan tujuan aliran pengiriman
- Contoh kasus penggunaan untuk pengarsipan pesan dan analitik

Prasyarat untuk berlangganan aliran pengiriman Firehose ke topik Amazon SNS

Untuk berlangganan aliran pengiriman Amazon Data Firehose ke topik SNS, Anda Akun AWS harus memiliki:

- Topik SNS standar. Untuk informasi selengkapnya, lihat Membuat topik Amazon SNS.
- 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.
- AWS Identity and Access Management (IAM role) yang mempercayai prinsip layanan Amazon SNS 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 prinsip layanan Amazon SNS. Sebagai contoh:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Untuk informasi selengkapnya tentang cara membuat peran, lihat Membuat peran untuk mendelegasikan izin untuk layanan AWSdalam Panduan Pengguna IAM.

Setelah menyelesaikan persyaratan tersebut, Anda dapat berlangganan aliran pengiriman untuk topik SNS.

Berlangganan aliran pengiriman Firehose ke topik Amazon SNS

Untuk mengirimkan notifikasi Amazon SNS 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 konsol Amazon SNS.
2. Di panel navigasi, pilih Subscriptions (Langganan).
3. Pada halaman Berlangganan, pilih Buat berlangganan.
4. Pada halaman Buat berlangganan, di bagian Detail, lakukan hal berikut ini:
    a. Untuk ARN topik, pilih Amazon Resource Name (ARN) dari topik standar.
    b. Untuk Protokol, pilih Firehose.
    c. Untuk Endpoint, pilih ARN aliran pengiriman Firehose yang dapat menerima notifikasi dari Amazon SNS.
    d. Untuk peran Langganan ARN, tentukan ARN dari 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 metadata Amazon SNS dari pesan yang diterbitkan, pilih Aktifkan pengiriman pesan mentah. Untuk informasi selengkapnya, lihat Pengiriman pesan mentah Amazon SNS.
5. (Opsional) Untuk mengkonfigurasi kebijakan filter, perluas bagian Subscription filter policy (Kebijakan filter langganan). Untuk informasi selengkapnya, lihat Kebijakan filter langganan Amazon SNS.
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 Antrean surat mati Amazon SNS (DLQs).
7. Pilih Create subscription (Buat langganan).

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

Bekerja dengan tujuan aliran pengiriman

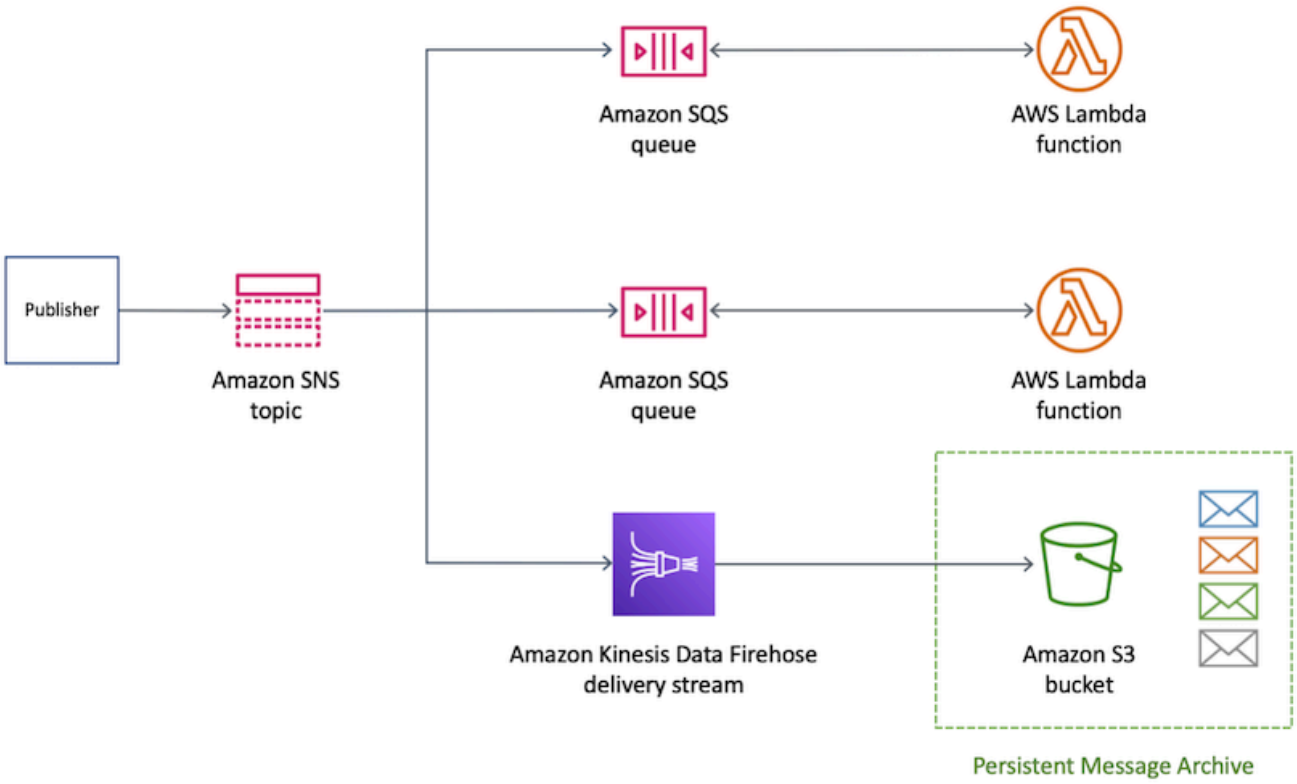
Melalui aliran pengiriman Amazon Data Firehose, Anda dapat mengirim pesan ke titik akhir tambahan. Bagian ini menjelaskan cara untuk bekerja dengan tujuan yang didukung.

Topik

- Tujuan Amazon S3
- OpenSearch Tujuan layanan
- Tujuan Amazon Redshift
- Tujuan HTTP

Tujuan Amazon S3

Bagian ini memberikan informasi tentang aliran pengiriman Amazon Data Firehose yang mempublikasikan data ke Amazon Simple Storage Service (Amazon S3).



Topik

- Format pesan yang diarsipkan untuk tujuan Amazon S3

- Menganalisis pesan untuk tujuan Amazon S3

Format pesan yang diarsipkan untuk tujuan Amazon S3

Contoh berikut ini menunjukkan notifikasi Amazon SNS yang dikirim ke bucket Amazon Simple Storage Service (Amazon S3), menggunakan indentasi untuk kemudahan pembacaan.

> **Note**
>
> Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Ketika pengiriman pesan mentah dinonaktifkan, Amazon SNS menambahkan metadata JSON ke pesan, termasuk properti tersebut:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan mentah Amazon SNS.

```
{
    "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 pesan SNS 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 pesan untuk tujuan Amazon S3

Halaman ini menjelaskan cara menganalisis pesan Amazon SNS yang dikirim melalui aliran pengiriman Amazon Data Firehose ke tujuan Amazon Simple Storage Service (Amazon S3).

Untuk menganalisis pesan SNS 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 kueri objek Amazon S3 menggunakan SQL standar. Untuk informasi selanjutnya, lihat Memulai dalam Panduan Pengguna Amazon Athena.

Kueri contoh

Untuk kueri contoh ini, asumsikan berikut ini:

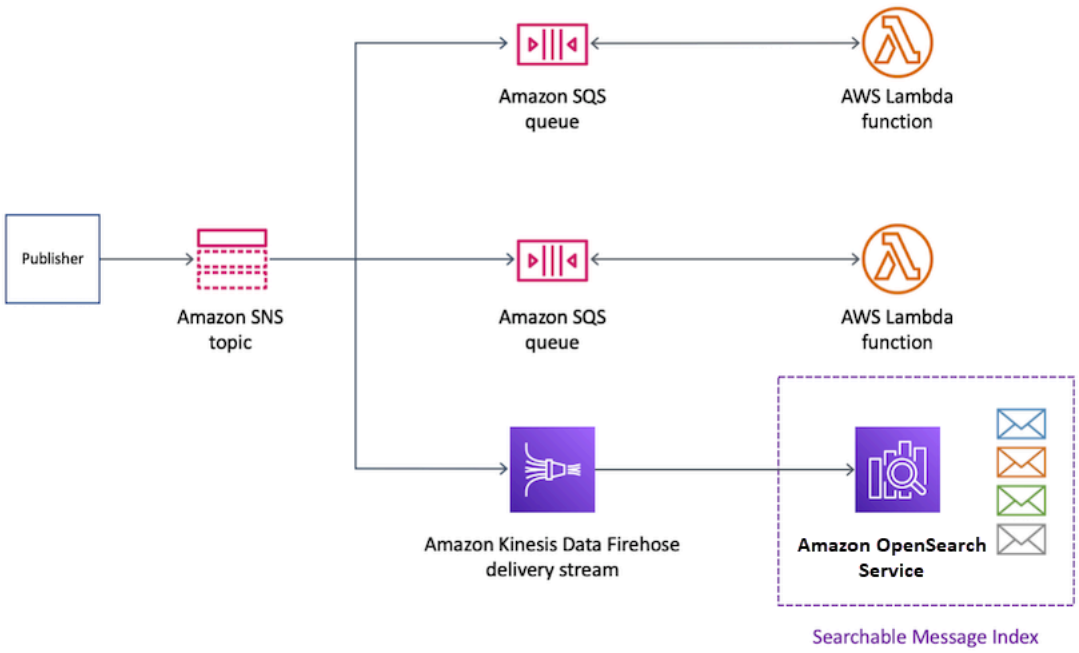
- Pesan disimpan dalam tabel `notifications` di skema `default`.
- Tabel `notifications` mencakup kolom `timestamp` dengan jenis `string`.

Kueri berikut ini mengembalikan semua pesan SNS 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';
```

OpenSearch Tujuan layanan

Bagian ini memberikan informasi tentang aliran pengiriman Amazon Data Firehose yang mempublikasikan data ke OpenSearch Layanan Amazon (Layanan)OpenSearch .



Topik

- Format pesan yang diarsipkan dalam indeks OpenSearch Service
- Menganalisis pesan untuk tujuan OpenSearch Layanan

Format pesan yang diarsipkan dalam indeks OpenSearch Service

Contoh berikut ini menunjukkan notifikasi Amazon SNS yang dikirim ke indeks Amazon OpenSearch Service (OpenSearch Service) yang bernama`my-index`. Indeks ini memiliki bidang filter waktu pada bidang `Timestamp`. Notifikasi SNS ditempatkan di properti `_source` dari muatan.

> **Note**
>
> Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Ketika pengiriman pesan mentah dinonaktifkan, Amazon SNS menambahkan metadata JSON ke pesan, termasuk properti tersebut:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`

- `UnsubscribeURL`
- `MessageAttributes`

Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan mentah Amazon SNS.


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "4961310096311132320325040540219328379477388655098593280 2.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

Menganalisis pesan untuk tujuan OpenSearch Layanan

Halaman ini menjelaskan cara menganalisis pesan Amazon SNS yang dikirim melalui aliran pengiriman Amazon Data Firehose ke tujuan OpenSearch Layanan Amazon (Layanan). OpenSearch

Untuk menganalisis pesan SNS yang dikirim melalui aliran OpenSearch pengiriman Firehose ke tujuan Layanan

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.

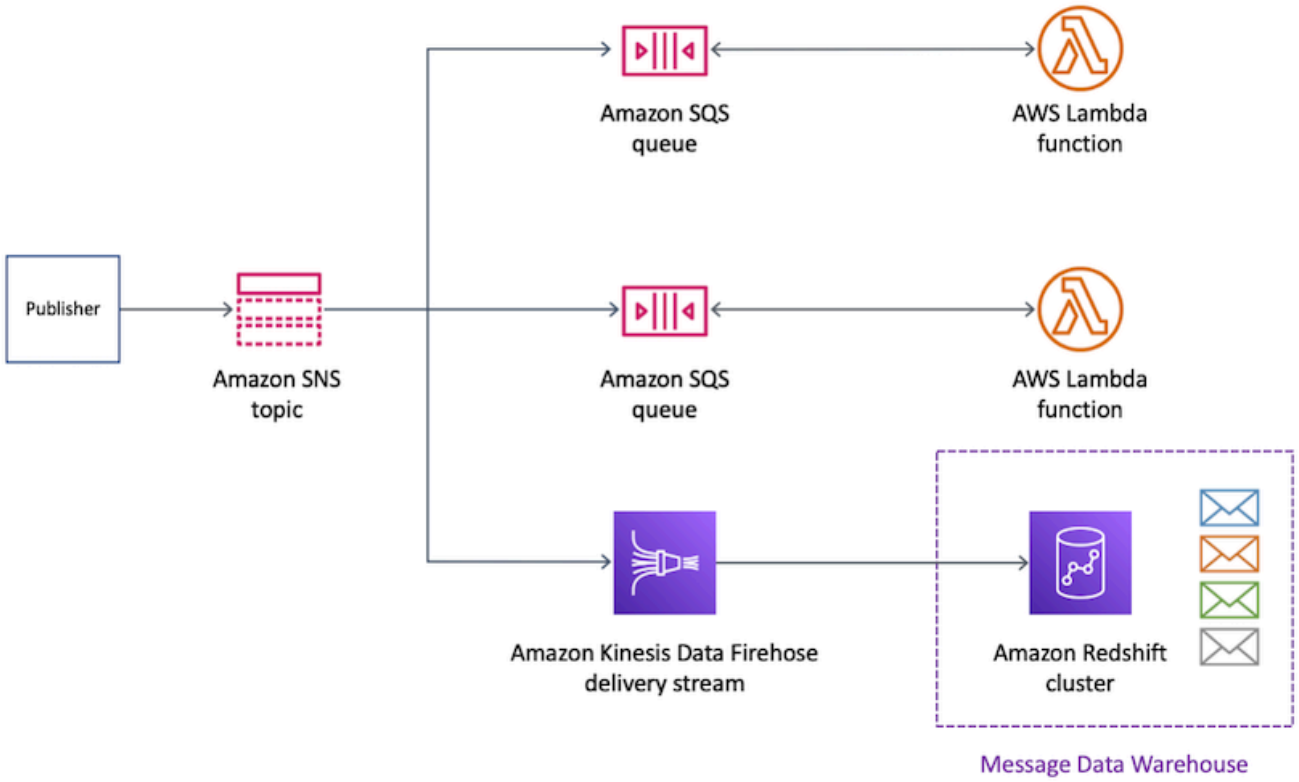
Kueri contoh

Contoh berikut ini membuat kueri untuk indeks `my-index` untuk semua pesan SNS 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"
            }
          }
        }
      ]
    }
  }
}
```

Tujuan Amazon Redshift

Bagian ini menjelaskan cara menyebarkan notifikasi Amazon SNS ke aliran pengiriman Amazon Data Firehose yang menerbitkan data ke Amazon Redshift. Dengan konfigurasi ini, Anda dapat terhubung ke basis data Amazon Redshift dan menggunakan alat kueri SQL untuk membuat kueri basis data untuk pesan Amazon SNS yang memenuhi kriteria tertentu.



Topik

- Struktur tabel arsip untuk tujuan Amazon Redshift
- Menganalisis pesan untuk tujuan Amazon Redshift

Struktur tabel arsip untuk tujuan Amazon Redshift

Untuk titik akhir Amazon Redshift, pesan Amazon SNS yang diterbitkan diarsipkan sebagai baris dalam tabel. Berikut ini adalah contoh.

Note

Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Ketika pengiriman pesan mentah dinonaktifkan, Amazon SNS menambahkan metadata JSON ke pesan, termasuk properti tersebut:

- `Type`
- `MessageId`
- `TopicArn`
- `Subject`
- `Message`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan mentah Amazon SNS.
Meskipun Amazon SNS menambahkan properti ke pesan menggunakan kapitalisasi yang ditampilkan dalam daftar ini, nama kolom di tabel Amazon Redshift muncul dalam semua karakter huruf kecil. Untuk mengubah metadata JSON untuk titik akhir Amazon Redshift, Anda dapat menggunakan perintah `COPY` SQL. Untuk informasi selengkapnya, lihat Salin dari contoh JSON dan Muat dari data JSON menggunakan opsi 'auto ignorecase' di Panduan Developer Basis Data 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=U	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

jenis	messageid	topicarn	subjek	pesan	timestamp	unsubscribeurl	messageattributes
						nsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	

jenis	messageid	topicarn	subjek	pesan	timestamp	unsubscribeurl	messageattributes
Notifikasi	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Subjek sampel 2	Pesan sampel 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

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=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic: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 Tujuan Amazon Redshift.

Menganalisis pesan untuk tujuan Amazon Redshift

Halaman ini menjelaskan cara menganalisis pesan Amazon SNS yang dikirim melalui aliran pengiriman Amazon Data Firehose ke tujuan Amazon Redshift.

Untuk menganalisis pesan SNS 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.
- Properti `Timestamp` dari pesan SNS disimpan dalam kolom `timestamp` tabel dengan jenis data kolom `timestamptz`.

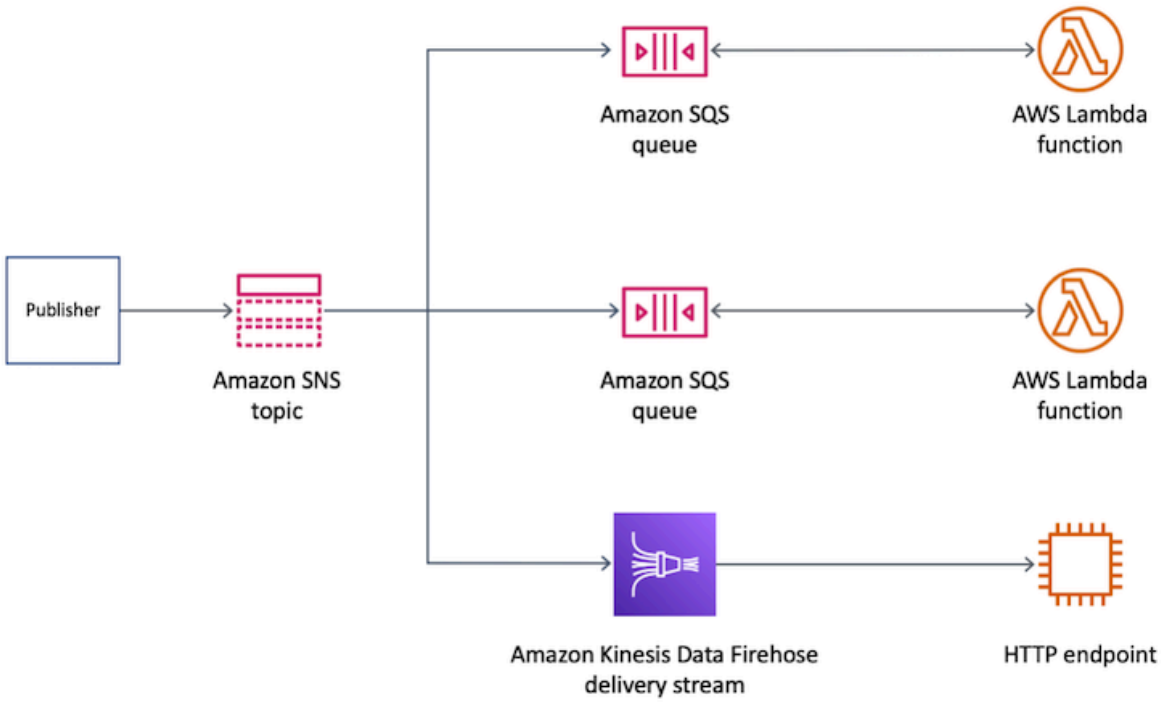
> **Note**
>
> Untuk mengubah metadata JSON untuk titik akhir Amazon Redshift, Anda dapat menggunakan perintah `COPY` SQL. Untuk informasi selengkapnya, lihat Salin dari contoh JSON dan Muat dari data JSON menggunakan opsi 'auto ignorecase' di Panduan Developer Basis Data Amazon Redshift.

Kueri berikut ini mengembalikan semua pesan SNS 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';
```

Tujuan HTTP

Bagian ini memberikan informasi tentang aliran pengiriman Amazon Data Firehose yang mempublikasikan data ke titik akhir HTTP.



Topik

- Format pesan yang dikirim untuk tujuan HTTP

Format pesan yang dikirim untuk tujuan HTTP

Berikut ini adalah contoh badan permintaan HTTP POST dari Amazon SNS yang dapat dikirim oleh aliran pengiriman Amazon Data Firehose ke titik akhir HTTP. Notifikasi SNS dienkodekan sebagai muatan 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 mentah Amazon SNS.

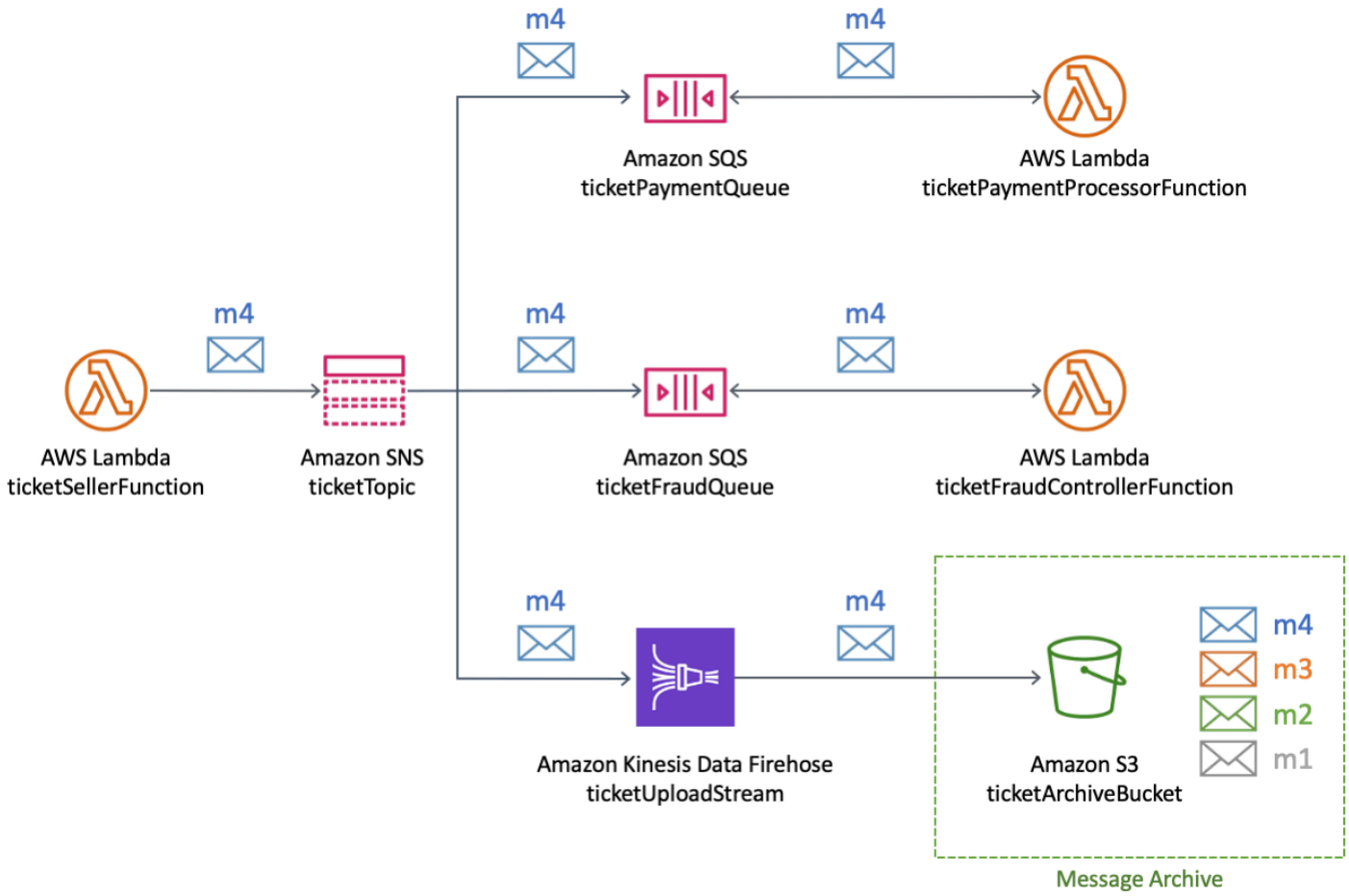
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
```

```
    "records": [
      {
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```

Contoh kasus penggunaan untuk pengarsipan pesan dan analitik

Bagian ini menyediakan tutorial dari kasus penggunaan umum untuk pengarsipan dan menganalisis pesan Amazon SNS.

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 SNS yang ada. Tujuan untuk aliran pengiriman adalah bucket Amazon Simple Storage Service (Amazon S3). Dengan konfigurasi ini, semua peristiwa yang diterbitkan untuk topik SNS diarsipkan dalam bucket Amazon S3. Diagram berikut ini menunjukkan arsitektur konfigurasi ini:



Untuk menjalankan analitik dan mendapatkan wawasan tentang penjualan tiket, perusahaan menjalankan kueri SQL menggunakan Amazon Athena. Sebagai contoh, perusahaan dapat membuat kueri untuk mempelajari tentang tujuan yang paling populer dan selebaran yang paling sering muncul.

Untuk membuat sumber daya AWS untuk kasus penggunaan ini, Anda dapat menggunakan AWS Management Console atau templat AWS CloudFormation.

Topik

- Membuat sumber daya awal
- Membuat aliran pengiriman Firehose
- Berlangganan aliran pengiriman Firehose ke topik Amazon SNS
- Menguji dan membuat kueri konfigurasi
- Menggunakan templat AWS CloudFormation

Membuat sumber daya awal

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 antrean Amazon Simple Queue Service (Amazon SQS)
- Topik Amazon SNS
- Dua langganan Amazon SQS untuk 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 lebih lanjut tentang bucket Amazon S3, lihatMembuat bucketdiPanduan Pengguna Amazon Simple Storage ServicedanBekerja dengan Bucket Amazon S3diPanduan Pengguna Amazon Simple Storage Service.

2. Buat dua antrean Amazon SQS:

   a. Buka konsol Amazon SQS.

   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. Dalam kotak kebijakan JSON, tempel kebijakan berikut ini:

```
{
  "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 angka Akun AWS (*123456789012*) dengan angka Anda sendiri, dan ubah Wilayah AWS (*us-east-1*) secara sesuai.

g. Pilih Buat antrean.

h. Ulangi langkah tersebut untuk membuat antrean SQS kedua bernama **ticketFraudQueue**.

Untuk informasi selengkapnya tentang cara membuat antrean SQS, lihat Membuat antrean Amazon SQS (konsol) di Panduan Developer Amazon Simple Queue Service.

3. Buat topik SNS:

   a. Buka Halaman topik dari konsol Amazon SNS.

   b. Pilih Buat topik.

   c. Di bawah Detail, untuk Jenis, pilih Standar.

   d. Untuk Nama, masukkan **ticketTopic**.

   e. Pilih Buat topik.

   Untuk informasi selengkapnya mengenai cara membuat topik SNS, lihat Membuat topik Amazon SNS.

4. Berlangganan kedua antrean SQS untuk topik SNS:

   a. Di konsol Amazon SNS, pada halaman detail topik ticketTopic, pilih Buat berlangganan.

   b. Di bawah Detail, untuk Protokol, pilih Amazon SQS.

c. Untuk Titik akhir, pilih Amazon Resource Name (ARN) dari antrean ticketPaymentQueue.

d. Pilih Buat berlangganan.

e. Ulangi langkah-langkah tersebut untuk membuat langganan kedua menggunakan ARN dari antrean TicketFraudQueue.

Untuk informasi selengkapnya tentang berlangganan topik SNS, lihat Berlangganan topik Amazon SNS. Anda juga dapat berlangganan antrean SQS untuk topik SNS dari konsol Amazon SQS. Untuk informasi selengkapnya, lihat Berlangganan antrean Amazon SQS untuk topik Amazon SNS (konsol) di Panduan Developer Amazon Simple Queue Service.

Anda telah membuat sumber daya awal untuk kasus penggunaan contoh ini. Untuk melanjutkan, lihat Membuat aliran pengiriman Firehose.

Membuat aliran pengiriman Firehose

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 AWS Identity and Access Management (IAM) role 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, catat Amazon Resource Name (ARN) pengaliran 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 cara membuat peran, lihat Membuat peran untuk mendelegasikan izin ke layanan AWS dalam Panduan Pengguna IAM.

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 langganan Amazon SNS untuk menaruh catatan di aliran pengiriman Amazon Data Firehose
- Aliran pengiriman Firehose berlangganan ke topik SNS

Untuk membuat IAM role untuk berlangganan Amazon SNS

1. Buka halaman Peran dari konsol IAM.
2. Pilih Buat peran.
3. Untuk Pilih jenis 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 tab JSON, dan kemudian tempel kebijakan berikut ini 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 angka Akun AWS (*`123456789012`*) dengan angka Anda sendiri, dan ubah Wilayah AWS (*`us-east-1`*) secara sesuai.
11. Pilih Tinjau kebijakan.
12. Pada halaman Tinjau kebijakan, untuk Nama, masukkan **`FirehoseSnsPolicy`**. Kemudian pilih Buat kebijakan.
13. Pada halaman Ringkasan peran, catat ARN Peran untuk nanti.

Untuk informasi selengkapnya tentang cara membuat peran, lihat Membuat peran untuk mendelegasikan izin ke layanan AWS dalam Panduan Pengguna IAM.

Untuk berlangganan aliran pengiriman Firehose ke topik SNS

1. Buka Halaman topik dari konsol Amazon SNS.
2. Pada tab Berlangganan, pilih Buat berlangganan.
3. Di bawah Detail, untuk Protokol, pilih Amazon Data Firehose.
4. Untuk Endpoint, masukkan Nama Sumber Daya Amazon (ARN) dari aliran pengiriman ticketUploadStreamyang Anda buat sebelumnya. Misalnya, masukkan **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Untuk peran Langganan ARN, masukkan ARN dari peran ticketUploadStreamSubscriptionRoleIAM yang Anda buat sebelumnya. Sebagai contoh, masukkan **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Pilih kotak centang Aktifkan pengiriman pesan mentah.
7. Pilih Buat berlangganan.

Anda telah membuat IAM role dan berlangganan topik SNS. Untuk melanjutkan, lihat Menguji dan membuat kueri konfigurasi.

Menguji dan membuat kueri konfigurasi

Halaman ini menjelaskan cara untuk menguji pengarsipan pesan dan kasus penggunaan contoh analitik dengan menerbitkan pesan untuk topik Amazon SNS. Instruksi termasuk kueri contoh yang dapat Anda jalankan dan menyesuaikan dengan kebutuhan Anda sendiri.

Untuk menguji konfigurasi Anda

1. Buka Halaman topik dari konsol Amazon SNS.
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 Penerbitan pesan Amazon SNS.

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:

- Berlangganan Amazon SNS
- Topik Amazon SNS
- Antrean Amazon Simple Queue Service (Amazon SQS)
- Bucket Amazon S3
- Aliran pengiriman Amazon Data Firehose
- Kebijakan dan peran AWS Identity and Access Management (IAM)

Menggunakan templat AWS CloudFormation

Untuk mengotomatisasi deployment pengarsipan pesan dan kasus penggunaan contoh analitik Amazon SNS, Anda dapat menggunakan templat YAML berikut ini:

```
---
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
```

Fanout untuk fungsi Lambda

Amazon SNS dan AWS Lambda terintegrasi sehingga Anda dapat memanggil fungsi Lambda dengan pemberitahuan Amazon SNS. Ketika pesan diterbitkan untuk topik SNS yang memiliki fungsi Lambda berlangganan, fungsi Lambda dipanggil dengan muatan pesan yang dipublikasikan. Fungsi Lambda menerima muatan pesan sebagai parameter input dan dapat memanipulasi informasi dalam pesan, mempublikasikan pesan ke topik SNS lainnya, atau mengirim pesan ke layanan AWS lain.

Amazon SNS juga mendukung atribut status pengiriman pesan untuk pemberitahuan pesan yang dikirim ke titik akhir Lambda. Untuk informasi selengkapnya, lihat Status pengiriman pesan Amazon SNS.

Prasyarat

Untuk mengaktifkan fungsi Lambda menggunakan pemberitahuan Amazon SNS, anda memerlukan berikut ini:

- Fungsi Lambda
- Topik Amazon SNS

Untuk informasi tentang cara membuat fungsi Lambda untuk digunakan dengan Amazon SNS, lihat Menggunakan Lambda dengan Amazon SNS. Untuk informasi tentang cara membuat topik Amazon SNS, lihat Membuat topik.

Ketika Anda menggunakan Amazon SNS untuk menyampaikan pesan dari opt-in daerah ke daerah yang diaktifkan secara default, Anda harus mengubah kebijakan yang dibuat dalam fungsi AWS Lambda dengan mengganti prinsipal `sns.amazonaws.com` dengan `sns.<opt-in-region>.amazonaws.com`.

Misalnya, jika Anda ingin berlangganan fungsi Lambda di US East (N. Virginia) untuk topik SNS di Asia Pacific (Hong Kong), mengubah prinsipal dalam kebijakan fungsi AWS Lambda ke `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 pesan SNS lintas wilayah dari wilayah penyertaan ke wilayah penyertaan lainnya tidak didukung.

Berlangganan fungsi ke topik

1. Masuk ke Konsol Amazon SNS.
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 ARN yang dipilih.
    b. Untuk Protocol (Protokol) pilih AWS Lambda.
    c. Untuk Endpoint (Titik akhir) masukkan ARN suatu fungsi.
    d. Pilih Create Subsription (Buat langganan).

Ketika pesan diterbitkan untuk topik SNS yang memiliki fungsi Lambda berlangganan, fungsi Lambda dipanggil dengan muatan pesan yang dipublikasikan. Untuk informasi tentang cara menggunakan AWS Lambda dengan Amazon SNS, termasuk tutorial, lihat Menggunakan AWS Lambda dengan Amazon SNS.

Fanout ke antrean Amazon SQS

Amazon SNS Bekerja sama dengan Amazon Simple Queue Service (Amazon SQS). Layanan ini memberikan manfaat yang berbeda bagi developer. Amazon SNS memungkinkan aplikasi untuk mengirim pesan waktu-kritis ke beberapa pelanggan melalui mekanisme “push”, menghilangkan kebutuhan untuk secara berkala memeriksa atau “poll” pembaruan. Amazon SQS adalah layanan antrean pesan yang digunakan oleh aplikasi terdistribusi untuk bertukar pesan melalui model polling, dan dapat digunakan untuk memisahkan pengiriman dan menerima komponen—tanpa memerlukan setiap komponen yang akan 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 antrean Amazon SQS untuk aplikasi lain untuk memproses di lain waktu.

Ketika Anda berlangganan antrean Amazon SQS untuk topik Amazon SNS, Anda dapat mempublikasikan pesan ke topik dan Amazon SNS mengirimkan pesan Amazon SQS ke antrian berlangganan. Pesan Amazon SQS berisi subjek dan pesan yang dipublikasikan ke topik bersama dengan metadata tentang pesan dalam dokumen JSON. Pesan Amazon SQS akan terlihat serupa dengan dokumen JSON 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 antrean Amazon SQS ke topik Amazon SNS

Untuk mengaktifkan topik Amazon SNS untuk mengirim pesan ke antrean Amazon SQS, lakukan salah satu dari berikut ini:

- Gunakan Konsol Amazon SQS, yang menyederhanakan prosesnya. Untuk informasi selengkapnya, lihat Berlangganan antrean Amazon SQS ke topik Amazon SNS dalam Panduan Developer Amazon Simple Queue Service.
- Ikuti langkah-langkah ini:
  1. Dapatkan Amazon Resource Name (ARN) dari antrean yang ingin Anda kirimi pesan dan topik tempat Anda ingin berlantrean.
  2. Memberikan`sqs:SendMessage` izin untuk topik Amazon SNS SNsehingga dapat mengirim pesan ke antrean.
  3. Berantrean topik Amazon SNS
  4. Memberikan pengguna IAM atauAkun AWS izin yang sesuai untuk mempublikasikan topik Amazon SNS
  5. Uji dengan menerbitkan pesan ke topik dan membaca pesan dari antrean.

Untuk mempelajari tentang cara menyiapkan topik untuk mengirim pesan ke antrean yang ada di antreanAWS yang berbeda, lihatMengirim olahpesan Amazon SNS ke antrean Amazon SQS di akun yang berbeda.

Untuk melihat templat AWS CloudFormation yang membuat topik yang mengirim pesan ke dua antrean, lihat Menggunakan templat AWS CloudFormation untuk membuat topik yang mengirim pesan ke antrean Amazon SQS.

Langkah 1: Dapatkan ARN dari antrean dan topik

Saat berlangganan antrean ke topik Anda, Anda memerlukan salinan ARN untuk antrean. Demikian pula, ketika memberikan izin untuk topik untuk mengirim pesan ke antrean, Anda akan memerlukan salinan ARN untuk topik tersebut.

Untuk mendapatkan antrean, Anda dapat menggunakan konsol Amazon SQS atau Tindakan GetQueueAttributesAPI API.

Untuk mendapatkan ARN antrean dari konsol Amazon SQS

1. Masuk ke AWS Management Console dan buka konsol Amazon SQS di https://console.aws.amazon.com/sqs/.
2. Pilih kotak untuk antrean yang ARN ingin Anda dapatkan.
3. Dari bagian Rincian, salin nilai ARN sehingga Anda dapat menggunakannya untuk berlangganan ke topik Amazon SNS.

Untuk mendapatkan topik ARN, Anda dapat menggunakan konsol Amazon SNS, perintah `sns-get-topic-attributes`, atau Tindakan API `GetQueueAttributes`.

Untuk mendapatkan topik ARN dari konsol Amazon SNS

1. Masuk ke Konsol Amazon SNS.
2. Pada panel navigasi, pilih topik yang ARN-nya ingin anda dapatkan.
3. Dari bagian Rincian, salin nilai ARN sehingga Anda dapat menggunakannya untuk memberikan izin untuk topik Amazon SNS untuk mengirim pesan ke antrean.

Langkah 2: Berikan izin untuk topik Amazon SNS untuk mengirim pesan ke antrean Amazon SQS

Untuk topik Amazon SNS untuk dapat mengirim pesan ke antrean, Anda harus menetapkan kebijakan antrian yang memungkinkan topik Amazon SNS untuk 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 antrean di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Untuk menetapkan kebijakan antrean, Anda dapat menggunakan konsol Amazon SQS atau Tindakan SetQueueAttributesAPI API. Sebelum memulai, pastikan Anda memiliki ARN untuk topik yang ingin Anda izinkan untuk mengirim pesan ke antrean. Jika Anda berlangganan antrean ke beberapa topik, kebijakan Anda harus berisi satu`Statement` elemen untuk setiap topik.

Untuk menetapkan SendMessage kebijakan antrean menggunakan konsol Amazon SQS SQ

1. Masuk ke AWS Management Console dan buka konsol Amazon SQS 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.
   - Atur `Principal` untuk menjadi layanan Amazon SNS, seperti yang ditunjukkan pada contoh di bawah ini.

- Gunakan kunci kondisi `aws:SourceAccount`global `aws:SourceArn`atau untuk melindungi dari skenario wakil yang bingung. Untuk menggunakan kunci kondisi ini, tetapkan nilai ke topik ARN. Jika antrean Anda berlangganan beberapa topik, Anda dapat`aws:SourceAccount` menggunakannya.

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 antrean topik Amazon SNS

Untuk mengirim pesan ke antrean melalui topik, Anda harus berlangganan antrean ke topik Amazon SNS. Anda menentukan antrean dengan ARN nya. Untuk berlangganan topik, Anda dapat menggunakan konsol Amazon SNS, perintah CLI `sns-subscribe`, atau Tindakan API `Subscribe`. Sebelum Anda mulai, pastikan Anda memiliki ARN untuk antrean yang Anda ingin berlangganan.

1. Masuk ke Konsol Amazon SNS.
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 ARN topik.
b. Untuk Protocol (Protokol), pilih Amazon SQS.
c. Untuk Endpoint (Titik akhir), masukkan ARN antrean Amazon SQS.
d. Pilih Create subscription (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 olahpesan Amazon SNS ke antrean Amazon SQS 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 memublikasikan topik Amazon SNS dan membaca/menghapus pesan dari antrean Amazon SQS. Untuk informasi selengkapnya tentang cara mengendalikan tindakan pada topik dan antrean untuk pengguna IAM, lihat Menggunakan kebijakan berbasis identitas dengan Amazon SNS, dan Identity and Access Management di Amazon SQS Dalam Panduan Developer Amazon Simple Queue Service.

Ada dua cara untuk mengontrol akses ke topik atau antrean:

- Menambahkan kebijakan ke pengguna atau grup IAM. 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 akun AWS lain, satu-satunya cara Anda dapat melakukannya adalah dengan menambahkan kebijakan yang memiliki prinsipal yang Akun AWS ingin Anda beri 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 pengguna atau grup IAM

Jika Anda menambahkan kebijakan berikut ke pengguna atau grup IAM, Anda akan memberikan pengguna atau anggota grup tersebut izin untuk melakukan`sns:Publish` tindakan pada topik itu MyTopic.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Jika Anda menambahkan kebijakan berikut ke antrean atau grup IAM, Anda akan memberikan pengguna atau anggota grup tersebut izin untuk melakukan`sqs:DeleteMessage` tindakan`sqs:ReceiveMessage` dan tindakan pada antrean 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**
>
> Bila Anda memberikan Akun AWS yang lain akses ke sumber daya di akun Anda, Anda juga memberikan pengguna IAM yang memiliki izin akses tingkat admin (akses wildcard) ke sumber daya tersebut. Semua pengguna IAM lain di akun lain secara otomatis ditolak akses ke sumber daya Anda. Jika Anda ingin memberikan pengguna IAM tertentu dalam Akun AWS akses ke sumber daya Anda, akun atau pengguna IAM dengan akses tingkat admin harus mendelegasikan izin untuk sumber daya untuk pengguna IAM tersebut. Untuk informasi selengkapnya tentang pendelegasian lintas akun, lihat Mengaktifkan akses Lintas Akun dalam Menggunakan Panduan IAM.

Jika Anda menambahkan kebijakan berikut ke topik MyTopic di akun 123456789012, Anda akan memberikan akun 111122223333 izin untuk melakukan`sns:Publish` tindakan pada topik itu.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Jika Anda menambahkan kebijakan berikut ke antrean MyQueue di akun 123456789012, Anda akan memberikan akun 111122223333 izin untuk melakukan`sqs:DeleteMessage` tindakan`sqs:ReceiveMessage` dan tindakan pada antrean itu.

```
{
  "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 topik menggunakan konsol Amazon SNS

1. Menggunakan kredensials dari Akun AWS atau pengguna IAM dengan izin untuk mempublikasikan ke topik, masuk ke AWS Management Console dan buka konsol Amazon SNS 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 konsol Amazon SQS

1. Menggunakan kredensials dari Akun AWS atau pengguna IAM dengan izin untuk melihat pesan dalam antrean, masuk ke AWS Management Console dan buka konsol Amazon SQS dihttps://console.aws.amazon.com/sqs/.
2. Pilih antrean yang berlangganan topik.
3. Pilih Kirim dan terima pesan, lalu pilih Polling untuk pesan. Sebuah pesan dengan jenis Pemberitahuan akan muncul.

4. Di kolom Body (Isi), pilih More Details (Detail lebih lanjut). Parameter kotak Message Details (Detail Pesan) berisi dokumen JSON yang berisi subjek dan pesan yang Anda diterbitkan untuk topik. Pesan terlihat serupa dengan dokumen JSON 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.

Menggunakan templat AWS CloudFormation untuk membuat topik yang mengirim pesan ke antrean Amazon SQS

AWS CloudFormation memungkinkan Anda untuk menggunakan file templat untuk membuat dan mengkonfigurasi koleksi sumber daya AWS bersama-sama sebagai satu unit. Bagian ini memiliki contoh templat yang membuatnya mudah untuk menyebarkan topik yang mempublikasikan ke antrean. Templat mengurus langkah-langkah pengaturan untuk Anda dengan membuat dua antrean, membuat topik dengan langganan antrean, menambahkan kebijakan ke antrean sehingga topik dapat mengirim pesan ke antrean, dan membuat pengguna dan grup IAM untuk mengontrol akses ke sumber daya tersebut.

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

Menggunakan AWS CloudFormation untuk mengatur topik dan antrean dalam sebuah Akun AWS

Templat contoh membuat topik Amazon SNS yang dapat mengirim pesan ke dua antrean Amazon SQS dengan izin yang sesuai bagi anggota satu grup IAM untuk memublikasikan ke topik dan yang lain untuk membaca pesan dari antrean. Templat juga menciptakan pengguna IAM yang ditambahkan ke setiap kelompok.

Anda menyalin isi templat ke dalam file. Anda juga dapat mengunduh template dari AWSHalaman templat CloudFormation. Pada halaman templat, pilih Jelajahi contoh templat dengan layanan AWSlalu pilih Amazon Simple Queue Service.

MySNSTopic diatur untuk mempublikasikan ke dua titik akhir berlangganan, yaitu dua antrean Amazon SQS (MyQueue1 dan MyQueue2). MyPublishTopicGroup adalah grup IAM yang anggotanya memiliki izin untuk mempublikasikan ke MySNSTopic menggunakan Tindakan API Publikasikan atau perintah sns-publish. Templat menciptakan pengguna IAM 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. Templat menciptakan kunci akses untuk dua pengguna IAM dengan MyPublishUserKey dan MyQueueUserKey. AddUserToMyPublishTopicGroup menambahkan MyPublishUser ke MyPublishTopicGroup sehingga pengguna akan memiliki izin yang ditetapkan untuk grup.

MyRDMessageQueueGroup adalah grup IAM yang anggotanya memiliki izin untuk membaca dan menghapus pesan dari antrean dua Amazon SQS menggunakan Tindakan API ReceiveMessage dan DeleteMessage. AddUsertoMyQueueGroup menambahkan MyQueueUser ke MyRDMessageQueueGroup sehingga pengguna akan memiliki izin yang ditetapkan untuk grup. MyQueuePolicy memberikan izin untuk MySNSTopic untuk mempublikasikan pemberitahuan untuk dua antrean.

Daftar berikut menunjukkan templat isi AWS CloudFormation.

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

```
        ]
      }
    }
  }
}
```

Fanout ke HTTP (S) titik akhir

Anda dapat menggunakan Amazon SNS untuk mengirimkan pesan notifikasi ke satu atau lebih titik akhir HTTP atau HTTPS. Saat Anda melangganken titik akhir ke topik, Anda dapat memublikasikan notifikasi ke topik dan Amazon SNS mengirimkan permintaan HTTP POST yang mengirimkan isi notifikasi ke titik akhir yang dilanggankan. Saat Anda melangganken titik akhir, Anda memilih apakah Amazon SNS menggunakan HTTP atau HTTPS untuk mengirimkan permintaan POST ke titik akhir. Jika Anda menggunakan HTTPS, maka Anda dapat mengambil keuntungan dari dukungan di Amazon SNS untuk hal berikut ini:

- Indikasi Nama Server (SNI)—Hal ini mengizinkan Amazon SNS untuk mendukung titik akhir HTTPS yang memerlukan SNI, seperti server yang memerlukan beberapa sertifikat untuk meng-host beberapa domain. Untuk informasi lebih lanjut tentang SNI, lihat Indikasi Nama Server.

- Autentikasi Akses Dasar dan Digest—Hal ini mengizinkan Anda untuk menentukan nama pengguna dan kata sandi di URL HTTPS untuk permintaan HTTP POST, seperti `https://user:password@domain.com` atau `https://user@domain.com`. Nama pengguna dan kata sandi dienkripsi melalui koneksi SSL yang dibuat saat menggunakan HTTPS. Hanya nama domain yang dikirim dalam plaintext. Untuk informasi selengkapnya tentang Autentikasi Akses Dasar dan Digest, lihat RFC-2617.

  > **Important**
  >
  > Amazon SNS saat ini tidak mendukung titik akhir HTTP (S) pribadi.
  > URL HTTPS hanya dapat diambil dari tindakan `GetSubscriptionAttributes` API Amazon SNS, 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 untuk topik bersama dengan metadata tentang notifikasi dalam dokumen JSON. Permintaan akan terlihat serupa dengan permintaan HTTP POST berikut. Untuk detail tentang header HTTP dan format JSON dari isi permintaan, lihat Header HTTP/HTTPS dan Format JSON notifikasi HTTP/HTTPS.

```
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

- Melanggankan titik akhir HTTP/S ke topik
- Memverifikasi tanda tangan pesan Amazon SNS
- Menguraikan format pesan

Melanggankan titik akhir HTTP/S ke topik

Halaman-halaman di bagian ini menjelaskan cara melanggankan titik akhir HTTP/S ke topik Amazon SNS.

Topik

- Langkah 1: Pastikan titik akhir Anda siap untuk memproses pesan Amazon SNS
- Langkah 2: Melanggankan titik akhir HTTP/HTTPS ke topik Amazon SNS
- Langkah 3: Konfirmasi langganan.
- Langkah 4: Tetapkan kebijakan pengiriman untuk langganan (opsional)
- Langkah 5: Memberikan pengguna izin untuk memublikasikan ke topik (opsional)
- Langkah 6: Kirim pesan ke titik akhir HTTP/HTTPS

Langkah 1: Pastikan titik akhir Anda siap untuk memproses pesan Amazon SNS

Sebelum Anda melanggankan titik akhir HTTP atau HTTPS ke topik, Anda harus memastikan bahwa titik akhir HTTP atau HTTPS memiliki kemampuan untuk menangani permintaan HTTP POST yang digunakan Amazon SNS untuk mengirimkan konfirmasi berlangganan dan pesan notifikasi. Biasanya, hal ini berarti membuat dan men-deploy aplikasi web (misalnya, servlet Java jika host titik akhir Anda menjalankan Linux dengan Apache dan Tomcat) yang memproses permintaan HTTP dari Amazon SNS. Saat Anda melanggankan titik akhir HTTP, Amazon SNS mengirimkan 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 waktu itu. Amazon SNS tidak akan mengirimkan notifikasi ke titik akhir sampai Anda mengonfirmasi langganan. Setelah Anda mengonfirmasi langganan, Amazon SNS akan mengirimkan notifikasi ke titik akhir saat tindakan memublikasikan dilakukan pada topik yang dilanggan.

Menyiapkan titik akhir Anda untuk memproses konfirmasi berlangganan dan pesan notifikasi

1. Kode Anda harus membaca header HTTP dari permintaan HTTP POST yang Amazon SNS kirim ke titik akhir Anda. Kode Anda harus mencari kolom header `x-amz-sns-message-type`, yang memberi tahu Anda jenis pesan yang telah dikirim Amazon SNS kepada Anda. Dengan melihat header, Anda dapat menentukan jenis pesan tanpa harus mengurai isi permintaan HTTP. Ada dua jenis yang perlu Anda tangani: `SubscriptionConfirmation` dan `Notification`. Pesan `UnsubscribeConfirmation` hanya digunakan saat langganan dihapus dari topik.

Untuk detail tentang header HTTP, lihat Header HTTP/HTTPS. Permintaan HTTP POST 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 dokumen JSON di badan permintaan HTTP POST dan teks/polos tipe konten untuk membaca pasangan nama-nilai yang membentuk pesan Amazon SNS. Gunakan parser JSON yang menangani konversi representasi yang di-escape dari karakter kontrol kembali ke nilai-nilai karakter ASCII-nya (misalnya, mengonversi \n ke karakter baris baru). Anda dapat menggunakan parser JSON yang ada seperti Jackson JSON Processor atau menulisnya sendiri. Untuk mengirimkan teks di bidang subjek dan pesan sebagai JSON yang valid, Amazon SNS harus mengonversi beberapa karakter kontrol untuk representasi yang di-escape yang dapat dimasukkan dalam dokumen JSON. Saat Anda menerima dokumen JSON di isi permintaan POST yang dikirim ke titik akhir Anda, Anda harus mengonversi karakter yang di-escape kembali ke nilai karakter aslinya jika Anda menginginkan representasi yang tepat dari

subjek asli dan pesan yang dipublikasikan ke topik. 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 notifikasi, konfirmasi berlangganan, atau pesan konfirmasi berhenti berlangganan yang dikirim oleh Amazon SNS. Dengan menggunakan informasi yang terdapat dalam pesan Amazon SNS, titik akhir Anda dapat menciptakan kembali tanda tangan sehingga Anda dapat memverifikasi isi pesan dengan mencocokkan tanda tangan Anda dengan tanda tangan yang dikirim Amazon SNS bersama dengan pesan. Untuk informasi selengkapnya tentang memverifikasi tanda tangan pesan, lihat Memverifikasi tanda tangan pesan Amazon SNS.
4. Berdasarkan jenis yang ditentukan oleh kolom header `x-amz-sns-message-type`, kode Anda harus membaca dokumen JSON yang terkandung dalam isi permintaan HTTP dan memproses pesan. Berikut adalah panduan untuk menangani dua jenis utama pesan:

   SubscriptionConfirmation

   Baca nilai `SubscribeURL` dan kunjungi URL tersebut. Untuk mengonfirmasi langganan dan mulai menerima notifikasi di titik akhir, Anda harus mengunjungi URL `SubscribeURL` (misalnya, dengan mengirimkan permintaan HTTP GET ke URL). Lihat contoh permintaan HTTP pada langkah sebelumnya untuk melihat seperti apa `SubscribeURL` itu. Untuk informasi lebih lanjut tentang format `SubscriptionConfirmation`, lihat Format JSON konfirmasi berlangganan HTTP/HTTPS. Saat Anda mengunjungi URL, Anda akan mendapatkan kembali respon yang terlihat seperti dokumen XML berikut. Dokumen mengembalikan ARN langganan untuk titik akhir dalam elemen `ConfirmSubscriptionResult`.

```
<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 Header HTTP/HTTPS. Permintaan HTTP POST berikut adalah contoh pesan notifikasi yang dikirim ke titik akhir 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 bahwa titik akhir Anda menanggapi pesan HTTP POST dari Amazon SNS dengan kode status yang sesuai. Waktu koneksi akan habis dalam 15 detik. Jika titik akhir Anda tidak merespons sebelum waktu koneksi habis atau jika titik akhir Anda mengembalikan kode status di luar jangkauan 200—4xx, Amazon SNS akan mempertimbangkan pengiriman pesan sebagai upaya yang gagal.
6. Pastikan bahwa kode Anda dapat menangani percobaan kembali pengiriman pesan dari Amazon SNS. Jika tidak menerima respons yang berhasil dari titik akhir Anda, Amazon SNS mencoba untuk 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 kembali dengan penundaan antara upaya yang gagal ditetapkan pada 20 detik.

   > **Note**
   >
   > Waktu permintaan pesan habis setelah 15 detik. Hal ini berarti, jika kegagalan pengiriman pesan disebabkan oleh waktu habis, 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 Amazon SNS. Dengan membandingkan ID pesan yang telah diproses dengan pesan masuk, Anda dapat menentukan apakah pesan tersebut merupakan upaya percobaan kembali.
7. Jika Anda melanggankan titik akhir HTTPS, pastikan bahwa titik akhir Anda memiliki sertifikat server dari Otoritas Sertifikasi (CA) terpercaya. Amazon SNS hanya akan mengirimkan pesan ke titik akhir HTTPS yang memiliki sertifikat server yang ditandatangani oleh CA yang dipercaya oleh Amazon SNS.
8. Deploy kode yang telah Anda buat untuk menerima pesan Amazon SNS. Saat Anda melanggankan titik akhir, titik akhir harus siap untuk menerima setidaknya pesan konfirmasi berlangganan.

Langkah 2: Melanggankan titik akhir HTTP/HTTPS ke topik Amazon SNS

Untuk mengirimkan pesan ke titik akhir HTTP atau HTTPS melalui topik, Anda harus melanggankan titik akhir ke topik Amazon SNS. Anda menentukan titik akhir menggunakan URL-nya. Untuk berlangganan ke topik, Anda dapat menggunakan konsol Amazon SNS, perintah sns-command,

atau tindakan API Langganan. Sebelum memulai, pastikan Anda memiliki URL untuk titik akhir yang ingin Anda langgan dan titik akhir Anda siap menerima konfirmasi dan notifikasi pesan seperti yang dijelaskan pada Langkah 1.

Untuk melangganakan titik akhir HTTP atau HTTPS ke topik menggunakan konsol Amazon SNS

1. Masuk ke Konsol Amazon SNS.
2. Di panel navigasi, pilih Topik.
3. Pilih langganan Buat.
4. Di daftar menurun Protokol, pilih HTTP atau HTTPS.
5. Di kotak Titik akhir, tempelkan URL titik akhir yang Anda inginkan untuk dikirimkan pesan oleh topik dan kemudian pilih Buat langganan.
6. Pesan konfirmasi ditampilkan. Pilih Close (Tutup).

   ID Langganan baru Anda ditampilkan PendingConfirmation. Saat Anda mengonfirmasi langganan, ID langganan akan menampilkan ID langganan.

Langkah 3: Konfirmasi langganan.

Setelah Anda berlangganan titik akhir Anda, Amazon SNS akan mengirimkan pesan konfirmasi berlangganan ke titik akhir. Kode yang melakukan tindakan yang dijelaskan di Langkah 1 seharusnya sudah di-deploy ke titik akhir Anda. Secara khusus, kode pada titik akhir harus mengambil nilai `SubscribeURL` dari pesan konfirmasi berlangganan dan mengunjungi lokasi yang ditentukan oleh `SubscribeURL` sendiri atau membuatnya tersedia untuk Anda sehingga Anda dapat secara manual mengunjungi `SubscribeURL`, misalnya menggunakan peramban web. Amazon SNS tidak akan mengirimkan pesan ke titik akhir sampai langganan telah dikonfirmasi. Saat Anda mengunjungi`SubscribeURL`, respon akan berisi dokumen XML yang berisi elemen `SubscriptionArn` yang menentukan ARN untuk berlangganan. Anda juga dapat menggunakan konsol Amazon SNS untuk memverifikasi bahwa langganan dikonfirmasi: ID Langganan akan menampilkan ARN untuk berlangganan, bukan nilai `PendingConfirmation` yang Anda lihat saat pertama kali menambahkan langganan.

Langkah 4: Tetapkan kebijakan pengiriman untuk langganan (opsional)

Secara default, jika pengiriman awal pesan gagal, Amazon SNS mencoba hingga tiga percobaan kembali dengan penundaan antara upaya yang gagal ditetapkan pada 20 detik. Sebagaimana dibahas dalam Langkah 1, titik akhir Anda harus memiliki kode yang dapat menangani pesan yang

dicoba kembali. Dengan menetapkan kebijakan pengiriman pada topik atau langganan, Anda dapat mengontrol frekuensi dan interval percobaan kembali Amazon SNS untuk pesan 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: Memberikan pengguna izin untuk memublikasikan ke topik (opsional)

Secara default, pemilik topik memiliki izin untuk memublikasikan ke topik. Untuk mengaktifkan pengguna atau aplikasi lain untuk memublikasikan ke topik, Anda harus menggunakan AWS Identity and Access Management (IAM) untuk memberikan izin memublikasikan ke topik. Untuk informasi lebih lanjut tentang memberikan izin untuk tindakan Amazon SNS kepada pengguna IAM, lihat Menggunakan kebijakan berbasis identitas dengan Amazon SNS.

Ada dua cara untuk mengontrol akses ke topik:

- Menambahkan kebijakan ke pengguna atau grup IAM. 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 pengguna atau grup IAM, Anda akan memberikan izin kepada pengguna atau anggota grup tersebut untuk melakukan `sns:Publish` tindakan pada topik MyTopic tersebut.

```
{
  "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 lain akses ke sumber daya di akun Anda, Anda juga memberikan pengguna IAM yang memiliki izin akses tingkat admin (akses wildcard) ke sumber daya tersebut. Semua pengguna IAM lain di akun lain secara otomatis ditolak akses ke sumber daya Anda. Jika Anda ingin memberikan pengguna IAM tertentu dalam Akun AWS akses ke sumber daya Anda, akun atau pengguna IAM dengan akses tingkat admin harus mendelegasikan izin untuk sumber daya untuk pengguna IAM tersebut. Untuk informasi selengkapnya tentang pendelegasian lintas akun, lihat Mengaktifkan akses Lintas Akun dalam Menggunakan Panduan 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 pesan ke titik akhir HTTP/HTTPS

Anda dapat mengirimkan pesan ke langganan topik dengan memublikasikan ke topik. Untuk memublikasikan ke topik, Anda dapat menggunakan konsol Amazon SNS, perintah CLI `sns-publish`, atau API `Publish`.

Jika Anda mengikuti Langkah 1, kode yang Anda deploy di titik akhir Anda harus memproses notifikasi.

Untuk memublikasikan ke topik menggunakan konsol Amazon SNS

1. Dengan menggunakan kredensial Akun AWS atau pengguna IAM dengan izin untuk memublikasikan ke topik, masuk ke AWS Management Console dan buka konsol Amazon SNS 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 titik akhir HTTP Anda oleh Amazon SNS, Anda dapat memverifikasi tanda tangan pesan. Ada dua kasus di mana kami merekomendasikan untuk memverifikasi keaslian pesan. Pertama, ketika Amazon SNS mengirim pesan ke titik akhir HTTP Anda bahwa Anda berlangganan topik. Kedua, saat Amazon SNS mengirimi Anda pesan konfirmasi ke titik akhir HTTP Anda setelah eksekusi `Subscribe` atau tindakan API. `Unsubscribe`

Anda harus melakukan hal berikut saat memverifikasi pesan yang dikirim oleh Amazon SNS:

- Selalu gunakan HTTPS saat mendapatkan sertifikat dari Amazon SNS.
- Validasi keaslian sertifikat.
- Verifikasi bahwa sertifikat diterima dari Amazon SNS.
- Saat memungkinkan, gunakan salah satu dari SDK AWS yang didukung untuk Amazon SNS untuk memvalidasi dan memverifikasi pesan.
- Validasi bahwa pesan Amazon SNS diterima dari yang Anda inginkan. `TopicArn`

Amazon SNS mendukung dua versi tanda tangan pesan:

- `SignatureVersion1`: Amazon SNS membuat tanda tangan berdasarkan hash SHA1 pesan.

- `SignatureVersion`2: Amazon SNS membuat tanda tangan berdasarkan hash SHA256 pesan.

Untuk mengonfigurasi versi tanda tangan pesan pada topik Amazon SNS

Secara default, topik Amazon SNS menggunakan `SignatureVersion` 1. Untuk memilih algoritma hashing pada topik Amazon SNS Anda, `SignatureVersion` baik 1 (SHA1) `SignatureVersion` atau 2 (SHA256), Anda dapat menggunakan tindakan API. `SetTopicAttributes`

Contoh kode berikut menunjukkan cara mengatur atribut topik `SignatureVersion` menggunakanAWS 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 pesan Amazon SNS saat menggunakan permintaan berbasis kueri HTTP

1. Ekstraksi pasangan nama-nilai dari dokumen JSON dalam isi permintaan HTTP POST yang dikirim Amazon SNS ke titik akhir Anda. Anda akan menggunakan nilai-nilai dari beberapa pasangan nama-nilai untuk membuat string-to-sign. Saat Anda memverifikasi tanda tangan pesan Amazon SNS, sangat penting bahwa Anda mengonversi karakter kontrol yang di-escape 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 dokumen JSON, lihat Langkah 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 digunakan Amazon SNS 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 yang diturunkan dari pesan Amazon SNS. Kirim pesan Amazon SNS, dalam format kanonik, ke algoritma hash yang sama yang digunakan untuk menghasilkan tanda tangan.
   a. Jika 1, gunakan SHA1 sebagai algoritma hash. `SignatureVersion`
   b. Jika 2, gunakan SHA256 sebagai algoritma hash. `SignatureVersion`
8. Hasilkan nilai hash yang dinyatakan dari pesan Amazon SNS. Nilai hash yang dinyatakan adalah hasil dari menggunakan nilai kunci publik (dari langkah 3) untuk mendekripsi tanda tangan yang dikirimkan dengan pesan Amazon SNS.

9. Verifikasi keaslian dan integritas pesan Amazon SNS. Bandingkan nilai hash yang diturunkan (dari langkah 7) dengan nilai hash yang dinyatakan (dari langkah 8). Jika nilai-nilai tersebut identik, maka penerima diyakinkan bahwa pesan belum diubah saat dalam transit dan pesan pasti berasal dari Amazon SNS. Jika nilai-nilai tersebut tidak identik, pesan tidak boleh dipercaya oleh penerima.

Menguraikan format pesan

Amazon SNS menggunakan format berikut.

Topik

- Header HTTP/HTTPS
- Format JSON konfirmasi berlangganan HTTP/HTTPS
- Format JSON notifikasi HTTP/HTTPS
- Format JSON konfirmasi berhenti berlangganan HTTP/HTTPS
- SetSubscriptionAttributesFormat JSON
- SetTopicAttributes Format JSON

Header HTTP/HTTPS

Saat mengirimkan konfirmasi berlangganan, notifikasi, atau pesan konfirmasi berhenti berlangganan untuk titik akhir HTTP/HTTPS, Amazon SNS mengirimkan pesan POST dengan sejumlah nilai header yang spesifik Amazon SNS. Anda dapat menggunakan nilai-nilai header untuk tugas-tugas seperti mengidentifikasi jenis pesan tanpa harus mengurai isi JSON untuk membaca`Type` nilai. 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` diMembuat kebijakan pengiriman HTTP/S.

`x-amz-sns-message-type`

Jenis pesan. Nilai yang mungkin adalah `SubscriptionConfirmation`, `Notification`, dan `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Pengidentifikasi Unik Universal, yang unik untuk setiap pesan yang dipublikasikan. Untuk notifikasi bahwa Amazon SNS mengirimkan ulang selama percobaan kembali, ID pesan dari pesan asli digunakan.

`x-amz-sns-topic-arn`

Amazon Resource Name (ARN) untuk topik tempat pesan ini diterbitkan.

`x-amz-sns-subscription-arn`

ARN untuk langganan ke titik akhir ini.

Header HTTP POST berikut adalah contoh header untuk sebuah`Notification` pesan ke titik akhir HTTP.

```
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
```

Format JSON konfirmasi berlangganan HTTP/HTTPS

Setelah Anda melanggankan HTTP/HTTPS titik akhir, Amazon SNS mengirimkan pesan konfirmasi berlangganan ke titik akhir HTTP/HTTPS. Pesan ini berisi`SubscribeURL` nilai yang harus Anda kunjungi untuk mengonfirmasi langganan (sebagai alternatif, Anda dapat menggunakan`Token` nilai dengan [`ConfirmSubscription`]).

> **Note**
>
> Amazon SNS

Pesan konfirmasi berlangganan adalah pesan POST dengan isi pesan yang berisi dokumen JSON dengan pasangan nama-nilai berikut.

`Type`

Jenis pesan. Untuk konfirmasi berlangganan, tipenya adalah `SubscriptionConfirmation`.

`MessageId`

Pengidentifikasi Unik Universal, yang unik untuk setiap pesan yang dipublikasikan. Untuk pesan yang dikirim ulang Amazon SNS selama mencoba lagi, 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 dilanggan titik akhir 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`

URL yang harus Anda kunjungi untuk mengonfirmasi langganan. Sebagai alternatif, Anda dapat menggunakan `ConfirmSubscription`tindakan untuk mengonfirmasi langganan.`Token`

`Timestamp`

Waktu (GMT) saat konfirmasi berlangganan dikirim.

`SignatureVersion`

Versi tanda tangan Amazon SNS yang digunakan.

- Jika`SignatureVersion` is 1,`Signature` adalah`SHA1withRSA` tanda tangan Base64 dikodekan dari`Message`,`MessageId`,`Type`,`Timestamp`, dan`TopicArn` nilai-nilai.

- JikaSignatureVersion adalah 2,Signature adalahSHA256withRSA tanda tangan Base64 dikodekan dariMessage,MessageId,Type,Timestamp, danTopicArn nilai-nilai.

Signature

Base64 dikodekanSHA1withRSA atauSHA256withRSA tanda tangan dariMessageMessageId,Type,Timestamp,, danTopicArn nilai-nilai.

SigningCertURL

URL untuk sertifikat yang digunakan untuk menandatangani pesan.

Pesan HTTP POST berikut adalah contohSubscriptionConfirmation pesan ke titik akhir HTTP.

```
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"
}
```

Format JSON notifikasi HTTP/HTTPS

Saat Amazon SNS mengirimkan notifikasi ke titik akhir HTTP atau HTTPS langganan, pesan POST yang dikirim ke titik akhir memiliki isi pesan yang berisi dokumen JSON dengan pasangan nama-nilai berikut.

`Type`

Jenis pesan. Untuk notifikasi, jenisnya adalah `Notification`.

`MessageId`

Pengidentifikasi Unik Universal, yang unik untuk setiap pesan yang dipublikasikan. Untuk notifikasi bahwa Amazon SNS mengirimkan ulang selama percobaan kembali, ID pesan dari pesan asli digunakan.

`TopicArn`

Amazon Resource Name (ARN) untuk topik tempat pesan ini diterbitkan.

`Subject`

`Subject`Parameter yang ditentukan saat notifikasi dipublikasikan ke topik.

> **Note**
>
> Ini adalah parameter opsional. Jika tidak`Subject` ada yang ditentukan, maka pasangan nama-nilai ini tidak muncul dalam dokumen JSON ini.

`Message`

`Message`Nilai yang ditentukan saat notifikasi dipublikasikan ke topik.

`Timestamp`

Waktu (GMT) saat notifikasi dipublikasikan.

`SignatureVersion`

Versi tanda tangan Amazon SNS yang digunakan.

- Jika`SignatureVersion` is 1,`Signature` adalah`SHA1withRSA` tanda tangan Base64 dikodekan dari`Message`,`MessageId`,`Subject` (jika ada),`Type`,`Timestamp`, dan`TopicArn` nilai-nilai.

- JikaSignatureVersion adalah 2,Signature adalahSHA256withRSA tanda tangan Base64 dikodekan dariMessage,MessageId,Subject (jika ada),Type,Timestamp, danTopicArn nilai-nilai.

Signature

Base64 dikodekanSHA1withRSA atauSHA256withRSA tanda tangan dariMessageMessageId,,Subject (jika ada)Type,Timestamp,, danTopicArn nilai-nilai.

SigningCertURL

URL untuk sertifikat yang digunakan untuk menandatangani pesan.

UnsubscribeURL

URL yang dapat Anda gunakan untuk berhenti melanggankan titik akhir dari topik ini. Jika Anda mengunjungi URL ini, Amazon SNS berhenti melanggankan titik akhir dan berhenti mengirimkan notifikasi ke titik akhir ini.

Pesan HTTP POST berikut adalah contohNotification pesan ke titik akhir HTTP.

```
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"
  }
}
```

Format JSON konfirmasi berhenti berlangganan HTTP/HTTPS

Setelah titik akhir HTTP/HTTPS berhenti dilangganken dari topik, Amazon SNS mengirimkan pesan konfirmasi berhenti berlangganan ke titik akhir.

Pesan konfirmasi berhenti berlangganan adalah pesan POST dengan isi pesan yang berisi dokumen JSON dengan pasangan nama-nilai berikut.

`Type`

Jenis pesan. Untuk konfirmasi berhenti berlangganan, jenisnya adalah `UnsubscribeConfirmation`.

`MessageId`

Pengidentifikasi Unik Universal, yang unik untuk setiap pesan yang dipublikasikan. Untuk pesan yang dikirim ulang Amazon SNS selama mencoba lagi, 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 yang telah berhenti dilanggan titik akhir.

`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`

URL yang harus Anda kunjungi untuk mengonfirmasi ulang langganan. Sebagai alternatif, Anda dapat menggunakan `ConfirmSubscription`tindakan untuk mengonfirmasi ulang langganan.`Token`

`Timestamp`

Waktu (GMT) saat konfirmasi berhenti berlangganan dikirim.

`SignatureVersion`

Versi tanda tangan Amazon SNS yang digunakan.

- Jika`SignatureVersion` is 1,`Signature` adalah`SHA1withRSA` tanda tangan Base64 dikodekan dari`Message`,`MessageId`,`Type`,`Timestamp`, dan`TopicArn` nilai-nilai.
- Jika`SignatureVersion` adalah 2,`Signature` adalah`SHA256withRSA` tanda tangan Base64 dikodekan dari`Message`,`MessageId`,`Type`,`Timestamp`, dan`TopicArn` nilai-nilai.

`Signature`

Base64 dikodekan`SHA1withRSA` atau`SHA256withRSA` tanda tangan dari`Message``MessageId`,`Type`,`Timestamp`,, dan`TopicArn` nilai-nilai.

`SigningCertURL`

URL untuk sertifikat yang digunakan untuk menandatangani pesan.

Pesan HTTP POST berikut adalah contoh`UnsubscribeConfirmation` pesan ke titik akhir HTTP.

```
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"
}
```

SetSubscriptionAttributesFormat JSON

Jika Anda mengirimkan permintaan ke`SetSubscriptionAttributes` tindakan dan mengatur`AttributeName` parameter ke nilai`DeliveryPolicy`, nilai`AttributeValue` parameter harus objek JSON 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 Format JSON 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, kunjungi SetSubscriptionAttributesdi Amazon Simple Notification Service API Reference. Untuk informasi selengkapnya tentang header tipe konten HTTP yang didukung, lihatMembuat kebijakan pengiriman HTTP/S.

SetTopicAttributes Format JSON

Jika Anda mengirimkan permintaan ke`SetTopicAttributes` tindakan dan mengatur`AttributeName` parameter ke nilai`DeliveryPolicy`, nilai`AttributeValue` parameter harus objek JSON 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 Format JSON 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, kunjungi SetTopicAttributesdi Amazon Simple Notification Service API Reference. Untuk informasi selengkapnya tentang header tipe konten HTTP yang didukung, lihatMembuat kebijakan pengiriman HTTP/S.

Fanout ke Alur Fork Peristiwa AWS

Untuk pengarsipan dan analitik acara, Amazon SNS sekarang merekomendasikan penggunaa n integrasi aslinya dengan Amazon Data Firehose. Anda dapat berlangganan aliran pengirima n Firehose ke topik SNS, 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 OpenSearch 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 yang digerakkan peristiwa yang menggunakan layanan pelanggan untuk melakukan pekerjaan secara otomatis sebagai respons terhadap peristiwa dipicu oleh layanan penerbit. 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 yang digerakkan peristiwa, Anda dapat berlangganan alur penanganan peristiwa—yang didukung oleh Alur Fork Peristiwa AWS—untuk topik Amazon SNS. AWS Alur Fork Peristiwa adalah rangkaian dari aplikasi bersarang sumber terbuka, berdasarkan Serverless Application ModelAWS (AWSSAM), di mana Anda dapat men-deploy secara langsung dari rangkaian Alur Fork Peristiwa AWS (pilih Tampilkan aplikasi yang membuat IAM role kustom atau kebijakan sumber daya) ke dalam akun AWS Anda.

Untuk kasus penggunaan Alur Fork Peristiwa AWS, lihat Men-deploy dan menguji aplikasi sampel Alur Fork Peristiwa AWS.

Topik

- Bagaimana Alur Fork Peristiwa AWS bekerja
- Men-deploy Alur Fork Peristiwa AWS
- Men-deploy dan menguji aplikasi sampel Alur Fork Peristiwa AWS
- Berlangganan Alur Fork Peristiwa AWS untuk topik Amazon SNS

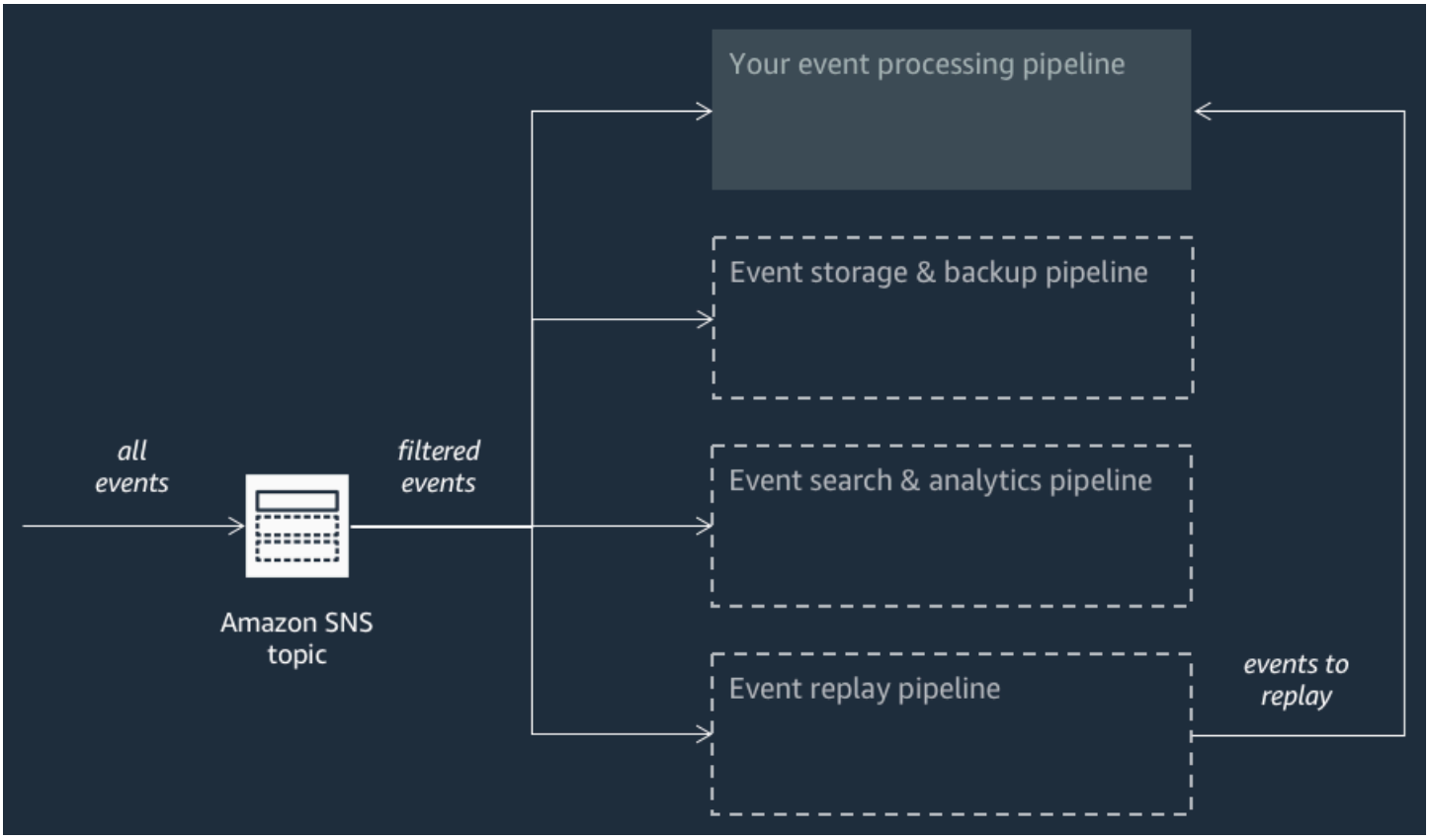
Bagaimana Alur Fork Peristiwa AWS bekerja

Alur Fork Peristiwa AWS adalah pola desain nirserver. Namun demikian, alur ini juga merupakan rangkaian aplikasi nirserver bersarang berdasarkan SAM AWS (di mana Anda dapat men-deploy secara langsung dari AWS Serverless Application Repository (SAR AWS) ke Akun AWS Anda untuk memperkaya platform yang digerakkan peristiwa milik Anda). 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 Alur Fork Peristiwa AWS yang dilengkapi dengan tiga aplikasi bersarang. Anda dapat men-deploy salah satu alur dari rangkaian Alur Fork Peristiwa AWS pada SAR AWS secara independen, sesuai kebutuhan arsitektur Anda.



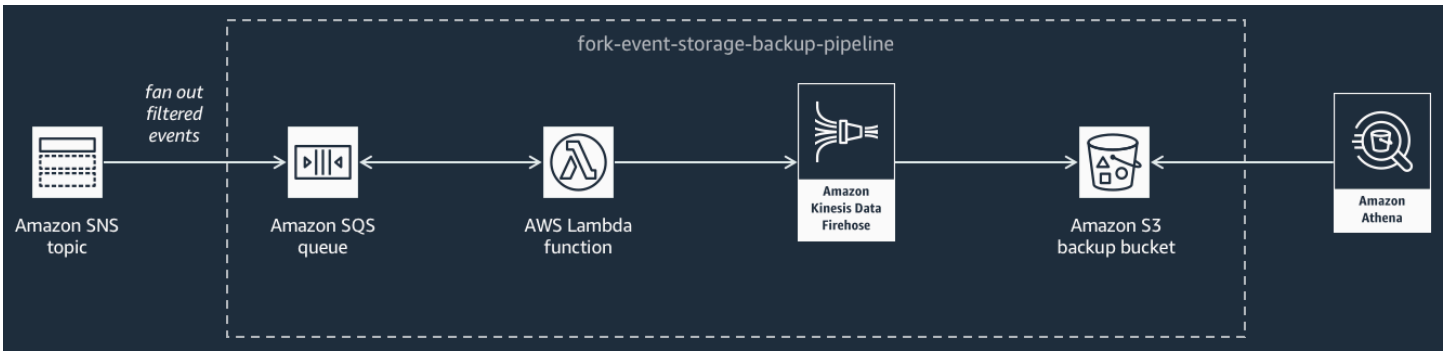
Setiap alur berlangganan pada topik Amazon SNS yang sama, yang mengizinkan dirinya sendiri untuk memproses peristiwa secara paralel saat peristiwa tersebut diterbitkan untuk topik. Setiap alur bersifat independen dan dapat mengatur Kebijakan Filter Berlangganan miliknya sendiri. Hal ini mengizinkan alur untuk memproses hanya subset dari peristiwa yang menjadi perhatian (bukan semua peristiwa yang diterbitkan untuk topik).

> **Note**
>
> Karena Anda menempatkan tiga Alur Fork Peristiwa AWS bersama alur pemrosesan peristiwa reguler Anda (mungkin sudah berlangganan ke topik Amazon SNS Anda), Anda tidak perlu mengubah bagian mana pun dari penerbit pesan Anda saat ini untuk mengambil keuntungan dari Alur Fork Peristiwa AWS di beban kerja Anda yang ada.

Penyimpanan peristiwa dan alur cadangan

Diagram berikut menunjukkan Penyimpanan Peristiwa dan Alur Cadangan. Anda dapat berlangganan alur ini untuk topik Amazon SNS Anda untuk secara otomatis membuat cadangan peristiwa yang mengalir melalui sistem Anda.

Pipeline ini terdiri dari antrian Amazon SQS yang menyangga peristiwa yang dikirimkan oleh topik Amazon SNS, fungsi yang secara otomatis AWS Lambda melakukan polling untuk peristiwa ini dalam antrian dan mendorongnya ke aliran Firehose Data Amazon, dan bucket Amazon S3 yang secara tahan lama mencadangkan peristiwa yang dimuat oleh aliran.

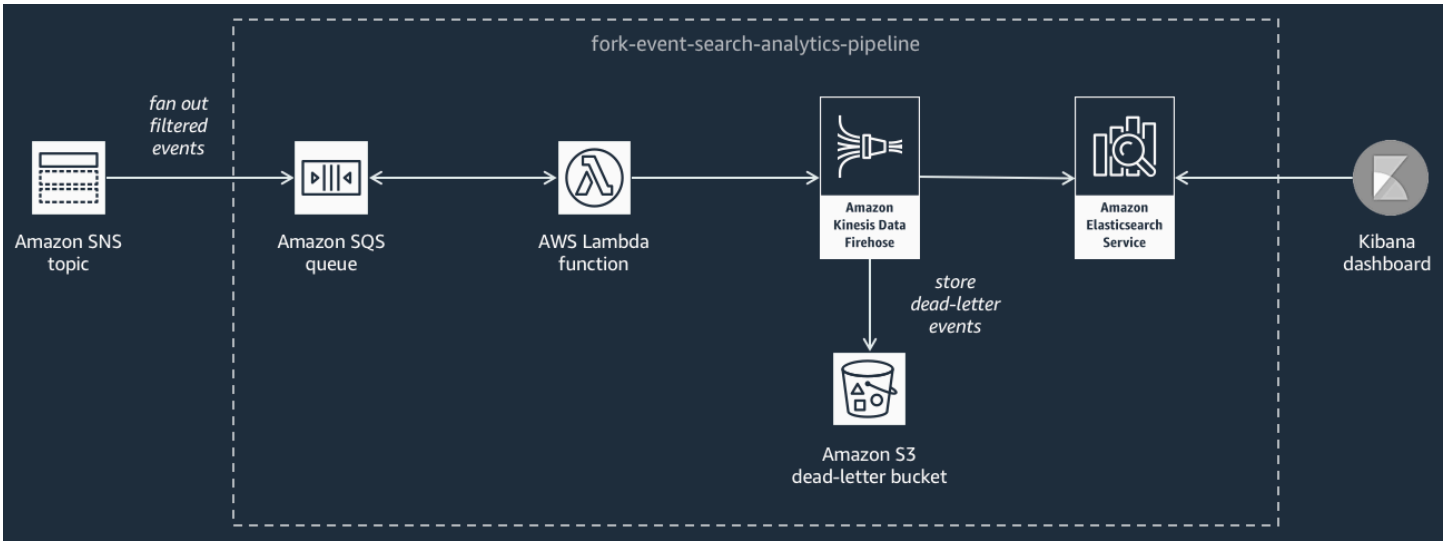


Untuk penyetelan halus perilaku pengaliran Firehose Anda, Anda dapat mengonfigurasinya untuk membuffer, mengubah, dan mengompres peristiwa Anda sebelum memuatnya ke dalam bucket. Saat peristiwa dimuat, Anda dapat menggunakan Amazon Athena untuk kueri bucket menggunakan kueri SQL standar. Anda juga dapat 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 alur ini untuk topik Amazon SNS Anda untuk indeks peristiwa yang mengalir melalui sistem Anda dalam domain pencarian dan kemudian menjalankan analitik padanya.

Pipeline ini terdiri dari antrian Amazon SQS yang menyangga peristiwa yang dikirimkan oleh topik Amazon SNS, fungsi yang AWS Lambda 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 dapat tidak diindeks di domain pencarian.



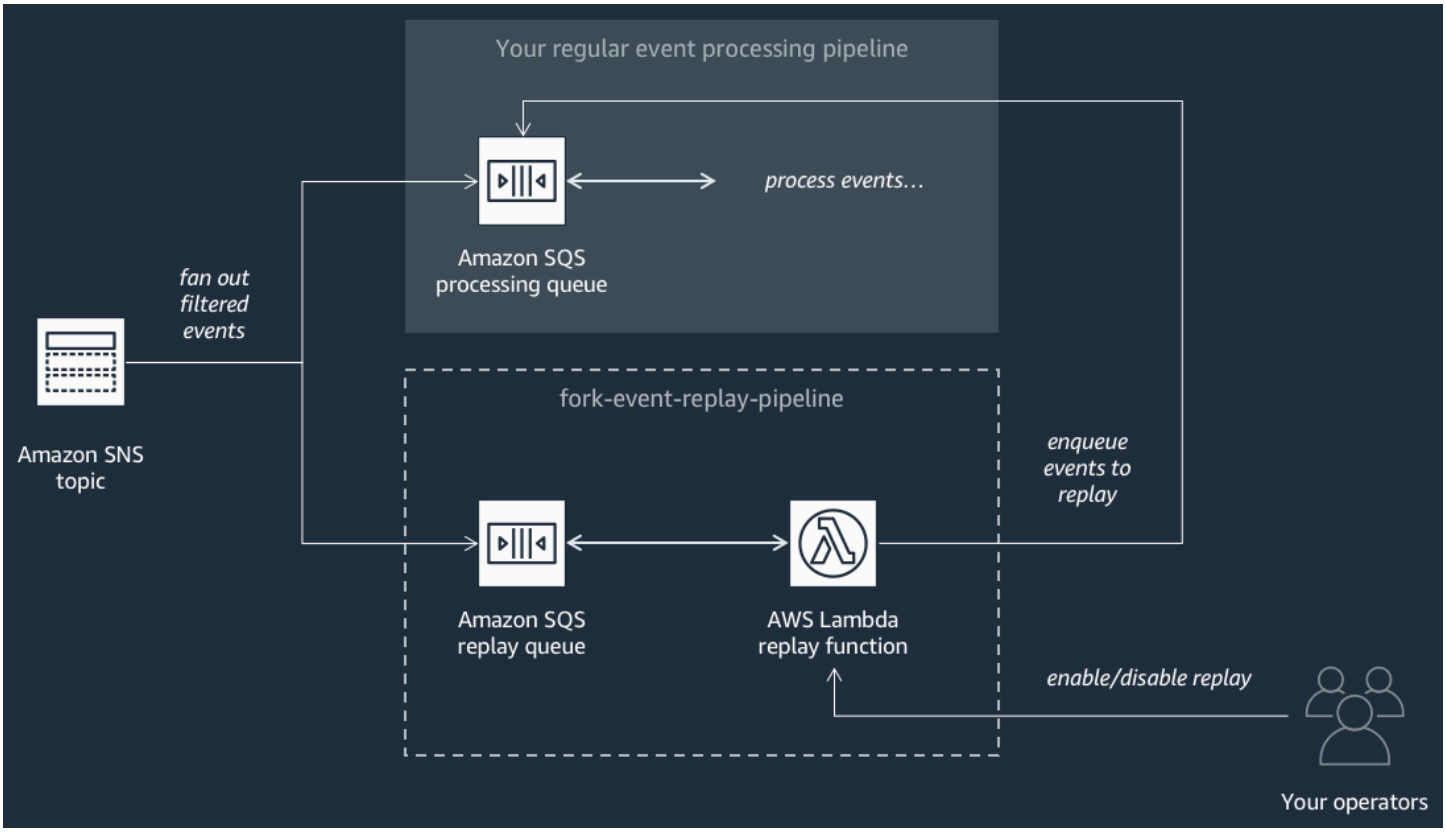
Untuk penyetelan halus pengaliran Firehose Anda dalam syarat buffering peristiwa, transformasi, dan kompresi, Anda dapat mengonfigurasi alur ini.

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 (sebagai contoh ketika platform Anda harus pulih dari kegagalan), Anda dapat berlangganan alur ini untuk topik Amazon SNS Anda dan kemudian memproses ulang peristiwa.

Alur ini terdiri dari antrean Amazon SQS yang membuffer peristiwa yang disampaikan oleh topik Amazon SNS, dan fungsi AWS Lambda yang mengumpulkan peristiwa dari antrean dan memindahkannya kembali ke alur pemrosesan peristiwa reguler Anda, yang juga berlangganan ke topik Anda.



> **Note**
>
> Secara default, fungsi ulangan dinonaktifkan, tidak memindahkan kembali peristiwa Anda. Jika Anda perlu untuk memproses ulang peristiwa, Anda harus mengaktifkan antrean ulangan Amazon SQS sebagai sumber peristiwa untuk fungsi ulangan AWS Lambda.

Men-deploy Alur Fork Peristiwa AWS

Rangkaian Alur Fork Peristiwa AWS (pilih Tampilkan aplikasi yang membuat IAM role kustom atau kebijakan sumber daya) tersedia sebagai grup aplikasi publik di AWS Serverless Application Repository, dari mana Anda dapat men-deploy dan mengujinya secara manual menggunakan konsol AWS Lambda. Untuk informasi tentang men-deploy alur menggunakan konsol AWS Lambda, lihat Berlangganan Alur Fork Peristiwa AWS untuk topik Amazon SNS.

Dalam skenario produksi, kami merekomendasikan penyematan Alur Fork Peristiwa AWS di dalam keseluruhan templat SAM AWS aplikasi Anda. Fitur aplikasi bersarang memungkinkan Anda melakukannya dengan menambahkan sumber daya `AWS::Serverless::Application` ke templat

SAM AWS Anda, dengan referensi pada AWS SAR `ApplicationId` dan `SemanticVersion` dari aplikasi bersarang.

Sebagai contoh, Anda dapat menggunakan Penyimpanan Peristiwa dan Alur Cadangan sebagai aplikasi bersarang dengan menambahkan potongan YAML berikut ini untuk bagian `Resources` dari templat SAM AWS Anda.

```
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
```

Bila Anda menentukan nilai parameter, Anda dapat menggunakan fungsi intrinsik AWS CloudFormation untuk referensi sumber daya lain dalam templat Anda. Sebagai contoh, dalam potongan YAML di atas, parameter `TopicArn` mereferensikan `AWS::SNS::Topic` sumber daya `MySNSTopic`, yang ditetapkan di tempat lain dalam templat AWS SAM. Untuk informasi selengkapnya, lihat Referensi Fungsi Intrinsik di Panduan Pengguna AWS CloudFormation.

> **Note**
>
> Halaman konsol AWS Lambda untuk aplikasi SAR AWS Anda mencakup tombol Salin sebagai Sumber Daya SAM, di mana salinan YAML diperlukan untuk menyarangkan aplikasi SAR AWS ke clipboard.

Men-deploy dan menguji aplikasi sampel Alur Fork Peristiwa AWS

Untuk mempercepat pengembangan aplikasi yang digerakkan peristiwa, Anda dapat berlangganan alur penanganan peristiwa—yang didukung oleh Alur Fork Peristiwa AWS—untuk topik Amazon SNS. AWS Alur Fork Peristiwa adalah rangkaian dari aplikasi bersarang sumber terbuka, berdasarkan Serverless Application ModelAWS (SAM AWS), di mana Anda dapat men-deploy secara langsung dari rangkaian Alur Fork Peristiwa AWS (pilih Tampilkan aplikasi yang membuat IAM role kustom

atau kebijakan sumber daya) ke dalam akun AWS Anda. Untuk informasi selengkapnya, lihat Bagaimana Alur Fork Peristiwa AWS bekerja.

Halaman ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Consoleuntuk men-deploy dan menguji aplikasi sampel Alur Fork Peristiwa AWS.

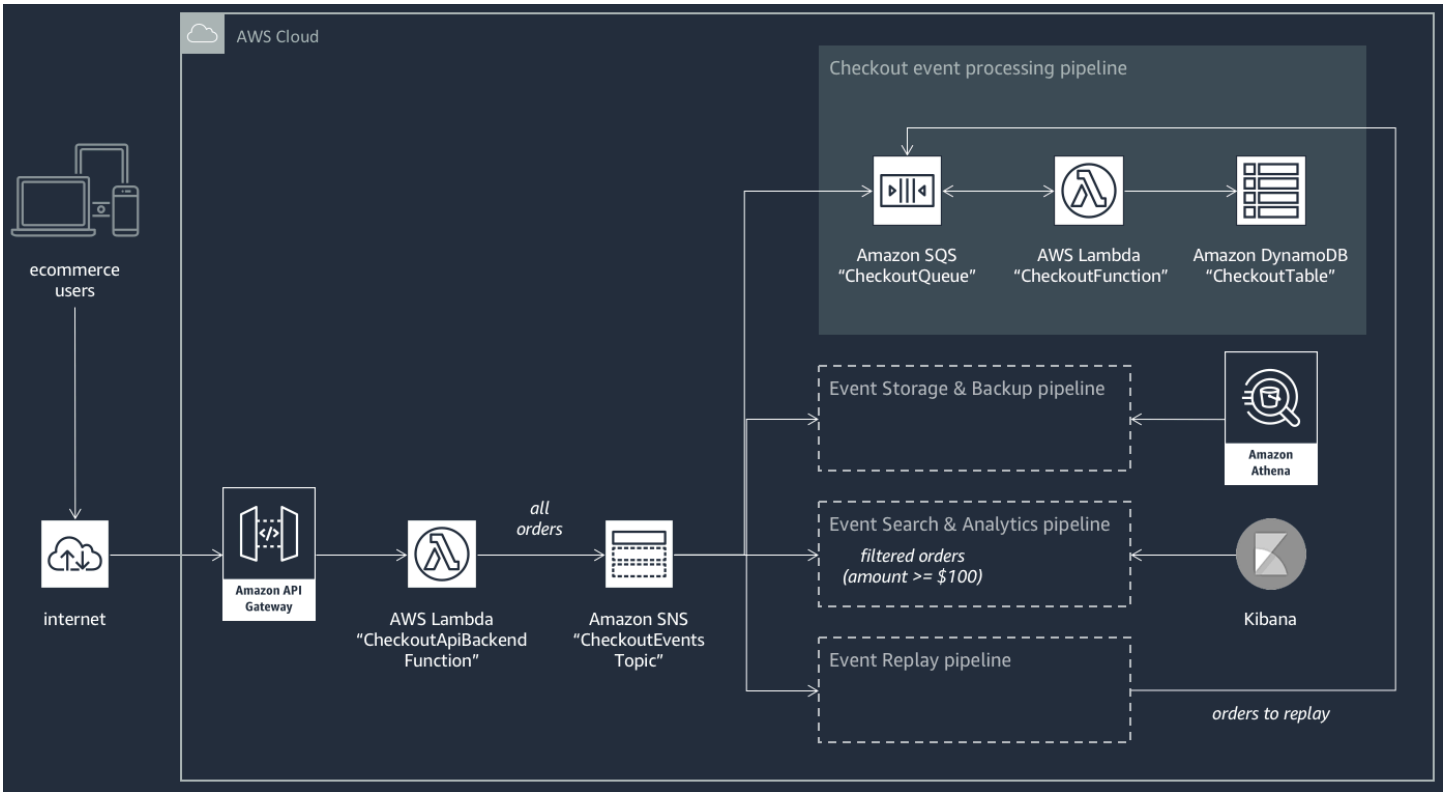
> **Important**
>
> Untuk menghindari timbulnya biaya yang tidak diinginkan setelah Anda selesai men-deploy aplikasi sampel Alur Fork Peristiwa AWS, hapus tumpukan AWS CloudFormation. Untuk informasi selengkapnya, lihat Menghapus Tumpukan pada Konsol AWS CloudFormation di Panduan Pengguna AWS CloudFormation.

Topik

- Contoh kasus penggunaan Alur Fork Peristiwa AWS
- Langkah 1: Untuk men-deploy aplikasi sampel
- Langkah 2: Untuk menjalankan aplikasi sampel
- Langkah 3: Untuk memverifikasi eksekusi aplikasi sampel dan alurnya
- Langkah 4: Untuk simulasi masalah dan mengulang peristiwa untuk pemulihan

Contoh kasus penggunaan Alur Fork Peristiwa AWS

Skenario berikut ini menjelaskan aplikasi perdagangan elektronik nirserver yang digerakkan peristiwa yang menggunakan Alur Fork Peristiwa AWS. Anda dapat menggunakan contoh aplikasi e-commerce ini di AWS Serverless Application Repository dan kemudian menyebarkannya di AWS Lambda konsol AndaAkun AWS, di mana Anda dapat mengujinya dan memeriksa kode sumbernya. GitHub



Aplikasi perdagangan elektronik ini mengambil pesanan dari pembeli melalui RESTful API yang dihosting oleh API Gateway dan didukung oleh AWS Lambda fungsi `CheckoutApiBackendFunction`. Fungsi ini menerbitkan semua pesanan yang diterima untuk topik Amazon SNS yang bernama `CheckoutEventsTopic` yang, pada gilirannya, dikeluarkan dalam pesanan menjadi empat alur yang berbeda.

Alur pertama adalah alur pemrosesan checkout reguler yang dirancang dan diimplementasikan oleh pemilik aplikasi perdagangan elektronik. Alur ini memiliki antrean Amazon SQS `CheckoutQueue` yang membuffer semua pesanan yang diterima, fungsi AWS Lambda yang bernama `CheckoutFunction` yang mengumpulkan antrean untuk memproses pesanan tersebut, dan tabel DynamoDB `CheckoutTable` yang secara aman menyimpan semua pesanan yang dibuat.

Menerapkan Alur Fork Peristiwa AWS

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 mengimplementasikan logika pemrosesan peristiwa ini, pemilik aplikasi dapat berlangganan Alur Fork Peristiwa AWS untuk topik Amazon SNS `CheckoutEventsTopic`

- Penyimpanan peristiwa dan alur cadangandikonfigurasi untuk mengubah data untuk menghapus detail kartu kredit, menyangga data selama 60 detik, mengompresnya menggunakan GZIP, dan mengenkripsi menggunakan kunci yang dikelola 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 peristiwa dikonfigurasi dengan bagian antrean Amazon SQS dari alur pemrosesan pesanan reguler yang dirancang dan diimplementasikan oleh pemilik aplikasi perdagangan elektronik.

  Untuk informasi lebih lanjut, lihat Antrean Nama dan URL di Panduan Developer Layanan ANtrean Sederhana Amazon.

Kebijakan filter JSON berikut ini diatur dalam konfigurasi untuk Pencarian Peristiwa dan Alur Analitik. Ini hanya akan mencocokkan pesanan yang masuk di mana jumlah total adalah $100 atau lebih tinggi. Untuk informasi selengkapnya, lihat Pemfilteran pesan Amazon SNS.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

Dengan menggunakan pola Alur Fork Peristiwa AWS, pemilik aplikasi perdagangan elektronik dapat menghindari overhead pengembangan yang sering mengikuti koding logika yang tidak terdiferensiasi

untuk penanganan peristiwa. Alih-alih, mereka dapat men-deploy Alur Fork Peristiwa AWS secara langsung dari AWS Serverless Application Repository ke Akun AWS miliknya.

Langkah 1: Untuk men-deploy aplikasi sampel

1. Masuk ke konsol AWS Lambda.
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 nirserver, Aplikasi publik, Tampilkan aplikasi yang membuat IAM role 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, deployment hanya akan memperbarui tumpukan AWS CloudFormation yang di-deploy sebelumnya (alih-alih 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 menciptakan IAM role kustom, kebijakan sumber daya dan men-deploy aplikasi bersarang. dan kemudian, di bagian bawah halaman, pilih Deploy.

Pada status Deployment for fork-example-ecommerce - ***my-app page***, Lambda menampilkan status Aplikasi Anda sedang di-deploy.

Di bagian Sumber Daya, AWS CloudFormation mulai membuat tumpukan dan menampilkan status CREATE_IN_PROGRESS untuk setiap sumber daya. Saat proses selesai, AWS CloudFormation menampilkan status CREATE_COMPLETE.

> **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: Untuk menjalankan aplikasi sampel

1. Di konsol AWS Lambda, 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 jenisnya 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 Titik akhir Prod API dan tambahkan `/checkout` padanya, sebagai contoh:

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. Salin JSON berikut ini ke file bernama `test_event.json`.

   ```
   {
      "id": 15311,
      "date": "2019-03-25T23:41:11-08:00",
      "status": "confirmed",
      "customer": {
         "id": 65144,
         "name": "John Doe",
         "email": "john.doe@example.com"
   ```

```
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
      "currency": "usd",
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. Untuk mengirim permintaan HTTPS ke titik akhir API Anda, lewatkan muatan peristiwa sebagai masukan dengan menjalankan perintah `curl`, sebagai contoh:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API mengembalikan respons kosong berikut ini, menunjukkan berhasilnya eksekusi:

```
{ }
```

Langkah 3: Untuk memverifikasi eksekusi aplikasi sampel dan alurnya

Langkah 1: Untuk memverifikasi eksekusi alur 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: Untuk memverifikasi eksekusi penyimpanan peristiwa dan alur 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 muatan JSON yang disimpan tidak berisi informasi kartu kredit apa pun, dekompresi file.

Langkah 3: Untuk memverifikasi eksekusi pencarian peristiwa dan alur analitik

1. Masuk ke konsol OpenSearch Layanan.
2. Pada panel navigasi, di bawah Domain saya, pilih domain dengan prefiks `serverl-analyt`.
3. Alur dikonfigurasi dengan kebijakan filter langganan Amazon SNS yang mengatur syarat pencocokan numerik.

Untuk memverifikasi bahwa peristiwa diindeks karena mengacu pada pesanan yang nilainya lebih tinggi dari USD $100, pada halaman serverl-analyt-***abcdefgh1ijk***, pilih Indeks, checkout_events.

Langkah 4: Untuk memverifikasi eksekusi alur ulangan peristiwa

1. Masuk ke konsol Amazon SQS.
2. Dalam daftar antrean, cari untuk `serverlessrepo-fork-example` dan pilih `ReplayQueue`.
3. Pilih Kirim dan terima pesan.
4. Di kotak dialog Kirim dan terima pesan di fork-example-ecommerce - ***aplikasi*** saya... ReplayP- ReplayQueue - ***123ABCD4E5F6***, pilih Poll untuk pesan.
5. Untuk memverifikasi bahwa peristiwa diantrekan, pilih Detail Selengkapnya di samping pesan yang muncul di antrean.

Langkah 4: Untuk simulasi masalah dan mengulang peristiwa untuk pemulihan

Langkah 1: Untuk mengaktifkan masalah yang disimulasikan dan mengirim permintaan API kedua

1. Masuk ke konsol AWS Lambda.
2. Pada panel navigasi, pilih Fungsi.
3. Cari untuk `serverlessrepo-fork-example` dan pilih `CheckoutFunction`.
4. *Di* ***- aplikasi fork-example-ecommerce saya - - CheckoutFunction ABCDEF...*** page, di bagian Environment variables, atur variabel BUG_ENABLED ke true dan kemudian pilih Save.
5. Salin JSON berikut ini ke file bernama `test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
    "payment": {
```

```
        "id": 3311,
        "amount": 75.00,
        "currency": "usd",
        "method": "credit",
        "card-network": "mastercard",
        "card-number": "1234 5678 9012 3456",
        "card-expiry": "12/2025",
        "card-owner": "Marcia Oliveira",
        "card-cvv": "321"
    },
    "shipping": {
        "id": 9900,
        "time": 20,
        "unit": "days",
        "method": "plane"
    },
    "items": [{
        "id": 9993,
        "product": 3120,
        "name": "Hockey Stick",
        "quantity": 1,
        "price": 75.00,
        "subtotal": 75.00
    }]
}
```

6. Untuk mengirim permintaan HTTPS ke titik akhir API Anda, lewatkan muatan peristiwa sebagai masukan dengan menjalankan perintah `curl`, sebagai contoh:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API mengembalikan respons kosong berikut ini, menunjukkan berhasilnya eksekusi:

```
{ }
```

Langkah 2: Untuk memverifikasi rusaknya data yang disimulasikan

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 ditampilkan, beberapa ditandai sebagai RUSAK!

Langkah 3: Untuk menonaktifkan masalah yang disimulasikan

1. Masuk ke konsol AWS Lambda.
2. Pada panel navigasi, pilih Fungsi.
3. Cari untuk `serverlessrepo-fork-example` dan pilih `CheckoutFunction`.
4. *Di - **aplikasi fork-example-ecommerce saya - - CheckoutFunction** **ABCDEF...*** page, di bagian Environment variables, atur variabel BUG_ENABLED ke false lalu pilih Save.

Langkah 4: Untuk mengaktifkan ulangan untuk memulihkan dari masalah

1. Di konsol AWS Lambda, pada panel navigasi, pilih Fungsi.
2. Cari untuk `serverlessrepo-fork-example` dan pilih `ReplayFunction`.
3. Perluas bagian Desainer, pilih tile SQS dan kemudian, dalam bagian SQS, pilih Diaktifkan.

   > **Note**
   >
   > Dibutuhkan sekitar 1 menit agar sumber peristiwa Amazon SQS memicu untuk menjadi diaktifkan.

4. Pilih Simpan.
5. Untuk melihat atribut yang dipulihkan, kembali ke konsol Amazon DynamoDB.
6. Untuk menonaktifkan ulangan, kembali ke konsol AWS Lambda dan nonaktifkan pemicu sumber peristiwa Amazon SQS untuk `ReplayFunction`.

Berlangganan Alur Fork Peristiwa AWS untuk topik Amazon SNS

Untuk mempercepat pengembangan aplikasi yang digerakkan peristiwa, Anda dapat berlangganan alur penanganan peristiwa—yang didukung oleh Alur Fork Peristiwa AWS—untuk topik Amazon SNS. AWS Alur Fork Peristiwa adalah rangkaian dari aplikasi bersarang sumber terbuka, berdasarkan

Serverless Application ModelAWS (SAM AWS), di mana Anda dapat men-deploy secara langsung dari rangkaian Alur Fork Peristiwa AWS (pilih Tampilkan aplikasi yang membuat IAM role kustom atau kebijakan sumber daya) ke dalam akun AWS Anda. Untuk informasi selengkapnya, lihat Bagaimana Alur Fork Peristiwa AWS bekerja.

Bagian ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console untuk men-deploy alur dan kemudian berlangganan Alur Fork Peristiwa AWS untuk topik Amazon SNS. Sebelum Anda memulai, buat topik Amazon SNS.

Untuk menghapus sumber daya yang terdiri dari pipeline, cari pipeline di halaman Applications di AWS Lambda konsol, perluas bagian template SAM, pilih CloudFormationstack, lalu pilih Other Actions, Delete Stack.

Topik

- Untuk men-deploy dan berlangganan penyimpanan peristiwa dan alur cadangan
- Untuk men-deploy dan berlangganan pada pencarian peristiwa dan alur analitik
- Untuk men-deploy dan berlangganan pada alur ulangan peristiwa

Untuk men-deploy dan berlangganan penyimpanan peristiwa dan alur cadangan

> Untuk pengarsipan dan analitik acara, Amazon SNS sekarang merekomendasikan penggunaa n integrasi aslinya dengan Amazon Data Firehose. Anda dapat berlangganan aliran pengirima n Firehose ke topik SNS, 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 OpenSearch 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 untuk men-deploy Penyimpanan Peristiwa dan Alur Cadangan dan berlangganlah ke topik Amazon SNS. Proses ini secara otomatis mengubah templat AWS SAM yang berkaitan dengan alur ke dalam tumpukan AWS CloudFormation, dan kemudian men-deploy tumpukan ke Akun AWS Anda. Proses ini juga menciptakan dan mengonfigurasi rangkaian sumber daya yang terdiri atas Penyimpanan Peristiwa dan Alur Cadangan, termasuk yang berikut ini:

- Antrean Amazon SQS

- Fungsi Lambda
- Aliran pengiriman Firehose
- Bucket cadangan Amazon S3

Untuk informasi selengkapnya tentang mengonfigurasi stream dengan bucket S3 sebagai tujuan, lihat `S3DestinationConfiguration` di Referensi API Amazon Data Firehose.

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 langganan Amazon SNS dalam panduan ini.

1. Masuk ke konsol AWS Lambda.
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 nirserver, Aplikasi publik, Tampilkan aplikasi yang membuat IAM role 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, deployment hanya akan memperbarui tumpukan AWS CloudFormation yang di-deploy sebelumnya (alih-alih membuat yang baru).

   b. (Opsional) Untuk BucketArn, masukkan ARN bucket S3 tempat acara masuk dimuat. Jika Anda tidak memasukkan nilai, bucket S3 baru dibuat di akun AWS Anda.
   c. (Opsional) Untuk DataTransformationFunctionArn, masukkan ARN dari fungsi Lambda di mana peristiwa yang masuk diubah. 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 dari topik Amazon SNS 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 langganan Amazon SNS, dalam format JSON, 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 menciptakan IAM role kustom, kebijakan sumber daya dan men-deploy aplikasi bersarang. dan kemudian pilih Deploy.

Pada halaman Status deployment untuk ***`aplikasi saya`***, Lambda menampilkan status Aplikasi Anda sedang di-deploy.

Di bagian Sumber Daya, AWS CloudFormation mulai membuat tumpukan dan menampilkan status CREATE_IN_PROGRESS untuk setiap sumber daya. Saat proses selesai, AWS CloudFormation menampilkan status CREATE_COMPLETE.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Pesan yang diterbitkan ke topik Amazon SNS Anda disimpan dalam bucket cadangan S3 yang ditetapkan oleh Penyimpanan Peristiwa dan Alur Cadangan secara otomatis.

Untuk men-deploy dan berlangganan pada pencarian peristiwa dan alur analitik

Untuk pengarsipan dan analitik acara, Amazon SNS sekarang merekomendasikan penggunaa n integrasi aslinya dengan Amazon Data Firehose. Anda dapat berlangganan aliran pengirima n Firehose ke topik SNS, 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 OpenSearch 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 untuk men-deploy Pencarian Peristiwa dan Alur Analitik dan berlangganlah ke topik Amazon SNS. Proses ini secara otomatis mengubah templat AWS SAM yang berkaitan dengan alur ke dalam tumpukan AWS CloudFormation, dan kemudian men-deploy tumpukan ke Akun AWS Anda. Proses ini juga membuat dan mengonfigurasi rangkaian sumber daya yang terdiri atas Pencarian Peristiwa dan Alur Analitik, termasuk yang berikut ini:

- Antrean Amazon SQS
- 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 Amazon Data Firehose 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 langganan Amazon SNS dalam panduan ini.

1. Masuk ke konsol AWS Lambda.
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 nirserver, Aplikasi publik, Tampilkan aplikasi yang membuat IAM role 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, deployment hanya akan memperbarui tumpukan AWS CloudFormation yang di-deploy sebelumnya (alih-alih membuat yang baru).

   b. (Opsional) Untuk DataTransformationFunctionArn, masukkan ARN dari fungsi Lambda yang digunakan untuk mengubah peristiwa yang masuk. 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 ARN domain OpenSearch Layanan, klaster yang mengonfigurasi fungsionalitas komputasi dan penyimpanan yang diperlukan. Jika Anda tidak memasukkan nilai, domain baru dibuat dengan konfigurasi default.

e. Untuk TopicArn, masukkan ARN dari topik Amazon SNS 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 langganan Amazon SNS, dalam format JSON, 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 menciptakan IAM role kustom, kebijakan sumber daya dan men-deploy aplikasi bersarang. dan kemudian pilih Deploy.

Pada status Deployment for ***`my-app-search`***page, Lambda menampilkan status Aplikasi Anda sedang di-deploy.

Di bagian Sumber Daya, AWS CloudFormation mulai membuat tumpukan dan menampilkan status CREATE_IN_PROGRESS untuk setiap sumber daya. Saat proses selesai, AWS CloudFormation menampilkan status CREATE_COMPLETE.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Pesan yang dipublikasikan ke topik Amazon SNS 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.

Untuk men-deploy dan berlangganan pada alur ulangan peristiwa

Halaman ini menunjukkan cara untuk men-deploy Alur Ulangan Peirstiwa dan berlangganan ke topik Amazon SNS. Proses ini secara otomatis mengubah templat AWS SAM yang berkaitan dengan alur ke dalam tumpukan AWS CloudFormation, dan kemudian men-deploy tumpukan ke Akun AWS Anda. Proses ini juga menciptakan dan mengonfigurasi rangkaian sumber daya yang terdiri atas Alur Ulangan Peristiwa, termasuk antrean Amazon SQS dan fungsi Lambda.

Untuk informasi selengkapnya tentang filter peristiwa, lihat Kebijakan filter langganan Amazon SNS dalam panduan ini.

1. Masuk ke konsol AWS Lambda.
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 nirserver, Aplikasi publik, Tampilkan aplikasi yang membuat IAM role 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, deployment hanya akan memperbarui tumpukan AWS CloudFormation yang di-deploy sebelumnya (alih-alih 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 antrean pemutaran ulang Amazon SQS menyimpan pesan. Jika Anda tidak memasukkan nilai, 1.209.600 detik (14 hari) akan digunakan.

d. Untuk TopicArn, masukkan ARN dari topik Amazon SNS tempat instance pipa garpu ini akan berlangganan.

e. Untuk DestinationQueueName, masukkan nama antrean Amazon SQS tempat fungsi replay Lambda meneruskan pesan.

f. (Opsional) Untuk SubscriptionFilterPolicy, masukkan kebijakan filter langganan Amazon SNS, dalam format JSON, 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 menciptakan IAM role kustom, kebijakan sumber daya dan men-deploy aplikasi bersarang. dan kemudian pilih Deploy.

Pada status Deployment for ***`my-app-replay`***page, Lambda menampilkan status Aplikasi Anda sedang di-deploy.

Di bagian Sumber Daya, AWS CloudFormation mulai membuat tumpukan dan menampilkan status CREATE_IN_PROGRESS untuk setiap sumber daya. Saat proses selesai, AWS CloudFormation menampilkan status CREATE_COMPLETE.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Pesan yang diterbitkan ke topik Amazon SNS Anda dibuffer untuk ulangan di antrean Amazon SQS yang ditetapkan oleh Alur Ulangan Peristiwa secara otomatis.

> **Note**
>
> Secara default, ulangan dinonaktifkan. Untuk mengaktifkan ulangan, navigasikan ke halaman fungsi pada konsol Lambda, perluas bagian Desainer, pilih tile SQS dan kemudian, dalam bagian SQS, 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 pemanggilan API yang gagal.

Halaman ini menjelaskan cara menggunakan EventBridge Scheduler untuk mempublikasikan pesan dari topik Amazon SNS sesuai jadwal.

Topik

- Mengatur peran eksekusi
- Buat jadwal
- Sumber daya terkait

Mengatur peran eksekusi

Saat Anda membuat jadwal baru, EventBridge Scheduler harus memiliki izin untuk menjalankan operasi API 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 API target yang ingin 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 SDK EventBridge Penjadwal, atau AWS CLIAWS CloudFormation, Anda harus memiliki peran eksekusi yang ada yang memberikan izin yang diperlukan EventBridge Penjadwal untuk memanggil target. Untuk informasi selengkapnya tentang mengatur peran eksekusi secara manual untuk jadwal Anda, lihat Menyiapkan 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/home.
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. Sebagai contoh, **`MyTestSchedule`**.

   b. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk jadwal Anda. Sebagai contoh, **`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`.	

Kejadian	Lakukan ini...	
	• Untuk Timezone, pilih zona waktu.	

Kejadian	Lakukan ini...	
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. 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 sebelumnya. Misalnya, jika Anda memilih 15 menit dan	

Kejadian	Lakukan ini...	
	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 Selanjutnya.

7. Pada halaman Select target, pilih operasi AWS API yang dipanggil EventBridge Scheduler:

   a. Pilih Amazon SNS Publish.

   b. Di bagian Publikasikan, pilih SNS topik atau pilih Buat SNS topik baru.

   c. (Opsional) Masukkan JSON muatan. Jika Anda tidak memasukkan payload, EventBridge Scheduler menggunakan peristiwa kosong untuk menjalankan fungsi.

8. Pilih Selanjutnya.

9. Pada halaman Pengaturan, lakukan hal berikut:

   a. Untuk mengaktifkan jadwal, di bawah Status jadwal, alihkan Aktifkan jadwal.

   b. Untuk mengonfigurasi kebijakan coba lagi untuk jadwal Anda, di bawah Kebijakan Coba ulang dan antrian surat 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 surat mati (DLQ)	Lakukan ini...
Jangan simpan	Pilih Tidak Ada.
Simpan acara di tempat yang sama Akun AWS di mana Anda membuat jadwal	a. Pilih Pilih antrian Amazon SQS di saya Akun AWS sebagai DLQ. b. Pilih Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.
Simpan acara di tempat yang berbeda Akun AWS dari tempat Anda membuat jadwal	a. Pilih Tentukan antrean Amazon SQS di lain Akun AWS sebagai DLQ. b. Masukkan Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.

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 ARN kunci KMS yang ada atau pilih AWS KMS key Buat 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 Selanjutnya.
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 Penjadwal
- EventBridge Referensi API Scheduler
- EventBridge Penetapan Harga Scheduler

Menggunakan Amazon SNS untuk olahpesan aplikasi-ke-orang (application-to-person/A2P)

Bagian ini menyediakan informasi tentang penggunaan Amazon SNS untuk notifikasi pengguna dengan pelanggan seperti aplikasi seluler, nomor ponsel, dan alamat email.

Topik

- Olahpesan teks seluler (SMS)
- Notifikasi push seluler
- Pemberitahuan email

Olahpesan teks seluler (SMS)

Anda dapat menggunakan Amazon SNS untuk mengirim pesan teks, atau pesan SMS, ke perangkat yang mendukung SMS. Anda dapat mengirim pesan langsung ke sebuah nomor telepon, atau Anda dapat mengirim pesan ke beberapa nomor telepon sekaligus dengan berlangganan topik untuk nomor telepon tersebut dan mengirim pesan Anda ke topik tersebut.

Anda dapat mengatur preferensi SMS untuk akun AWS Anda untuk menyesuaikan pengiriman SMS dengan kasus penggunaan dan anggaran Anda. Misalnya, Anda dapat memilih apakah pesan Anda dioptimalkan biayanya atau dioptimalkan untuk pengiriman yang andal. Anda juga dapat menentukan kuota pengeluaran untuk pengiriman pesan individu dan kuota pengeluaran bulanan untukAkun AWS Anda.

Jika diwajibkan oleh undang-undang dan peraturan setempat (seperti AS dan Kanada), penerima SMS dapat memilih keluar, yang berarti mereka memilih untuk berhenti menerima pesan SMS dari Akun AWS Anda. Setelah penerima memilih keluar, Anda dapat, dengan batasan, memilih nomor telepon itu lagi sehingga Anda dapat melanjutkan pengiriman pesan ke sana.

Amazon SNS mendukung olahpesan SMS di beberapa wilayah, dan Anda dapat mengirim pesan ke lebih dari 200 negara dan wilayah. Untuk informasi selengkapnya, lihat Negara dan wilayah yang didukung.

Topik

- Sandbox SMS

- Identitas asal untuk pesan SMS
- Meminta dukungan untuk olahpesan SMS dengan Amazon SNS
- Mengatur preferensi olahpesan SMS
- Mengirim pesan SMS
- Memantau aktivitas SMS
- Mengelola nomor telepon dan langganan SMS
- Negara dan wilayah yang didukung
- Praktik terbaik SMS

Sandbox SMS

Ketika Anda mulai menggunakan Amazon SNS untuk mengirim pesan SMS, akun AWS Anda berada di sandbox SMS. Sandbox SMS menyediakan lingkungan yang aman bagi Anda untuk mencoba fitur Amazon SNS tanpa mempertaruhkan reputasi Anda sebagai pengirim SMS. Saat akun Anda berada di sandbox SMS, Anda dapat menggunakan semua fitur Amazon SNS, dengan batasan berikut:

- Anda hanya dapat mengirim pesan SMS ke nomor telepon tujuan yang terverifikasi.
- Anda dapat memiliki hingga 10 nomor telepon tujuan terverifikasi.
- Anda hanya dapat menghapus nomor telepon tujuan setelah 24 jam atau lebih sejak verifikasi atau upaya verifikasi terakhir.

Ketika akun Anda dikeluarkan dari sandbox, batasan ini dihapus, dan Anda dapat mengirim pesan SMS ke penerima mana pun.

Topik

- Menambahkan dan memverifikasi nomor telepon di sandbox SMS
- Menghapus nomor telepon dari sandbox SMS
- Keluar dari sandbox SMS

Menambahkan dan memverifikasi nomor telepon di sandbox SMS

Untuk memulai mengirim pesan SMS saat AWS akun Anda berada di kotak pasir SMS, buat identitas originasi, tambahkan nomor telepon tujuan, lalu verifikasi.

> **Note**
>
> Seperti halnya akun yang tidak ada di sandbox SMS, identitas asal diperlukan sebelum Anda dapat mengirim pesan SMS ke penerima di beberapa negara atau wilayah. Untuk informasi selengkapnya, lihat Negara dan wilayah yang didukung.
> ID asal meliputi ID pengirim dan berbagai jenis nomor asal. Untuk melihat nomor asal yang ada, di panel navigasi konsol Amazon SNS, pilih Origination numbers (Nomor asal). Saat ini, ID pengirim tidak ditampilkan dalam daftar ini.

Cara menambahkan dan memverifikasi nomor telepon tujuan

1. Masuk ke konsol Amazon SNS.
2. Buat identitas originasi untuk nomor telepon.
3. Di menu konsol, pilih AWS Wilayah yang mendukung pesan SMS.
4. Di panel navigasi, pilih Text messaging (SMS) (Pesan teks).
5. Di halaman Mobile text messaging (SMS) (Pesan teks seluler), di bawah Sandbox destination phone numbers (Nomor telepon tujuan sandbox), pilih Add phone number (Tambahkan nomor telepon).
6. Di bawah Destination details (Detail tujuan), masukkan kode negara dan nomor telepon, tentukan bahasa yang akan digunakan untuk pesan verifikasi, lalu pilih Add phone number (Tambahkan nomor telepon).

   Amazon SNS mengirimkan kata sandi satu kali (one-time password/OTP) ke nomor telepon tujuan. Jika nomor telepon tujuan tidak menerima OTP dalam waktu 15 menit, pilih Resend verification code (Kirim ulang kode verifikasi). Anda dapat mengirim OTP ke nomor telepon tujuan yang sama hingga lima kali setiap 24 jam.
7. Di kotak Verification code (Kode verifikasi), masukkan OTP yang dikirim ke nomor telepon tujuan, lalu pilih verify phone number (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.

8. 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 teks OTP.

- Batas pengeluaran SMS Amazon SNS: Jika Anda Akun AWS telah melampaui batas pengeluaran untuk mengirim pesan SMS, pesan lebih lanjut, termasuk teks OTP, mungkin tidak dikirimkan sampai batas meningkat atau masalah penagihan diselesaikan.
- Nomor telepon yang tidak dipilih untuk pemberitahuan SMS: Di beberapa negara atau wilayah, penerima harus memilih untuk menerima pesan SMS dari kode pendek, yang biasanya digunakan untuk teks OTP. Jika nomor telepon penerima tidak dipilih, mereka tidak akan menerima teks OTP.
- Pembatasan atau penyaringan operator: Beberapa operator seluler mungkin memiliki batasan atau mekanisme penyaringan yang mencegah pengiriman jenis pesan SMS tertentu, termasuk teks OTP. 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, teks OTP tidak akan dikirimkan.
- Masalah jaringan: Masalah jaringan sementara atau pemadaman dapat mencegah pengiriman pesan SMS, termasuk teks OTP, ke telepon penerima.
- Pengiriman tertunda: Dalam beberapa kasus, pesan SMS mungkin mengalami keterlambatan pengiriman karena kemacetan jaringan atau faktor lainnya. Teks OTP pada akhirnya dapat dikirimkan, 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 pengiriman pesan Amazon SNS, termasuk teks OTP, dapat ditangguhkan atau dihentikan.

Menghapus nomor telepon dari sandbox SMS

Anda dapat menghapus nomor telepon tujuan dalam status menunggu dan terverifikasi dari sandbox SMS.

Cara menghapus nomor telepon tujuan dari sandbox SMS

1. Tunggu 24 jam setelah memverifikasi nomor telepon, atau 24 jam setelah upaya verifikasi terakhir Anda.
2. Masuk ke konsol Amazon SNS.
3. Di menu konsol tersebut, pilih Wilayah AWS yang mendukung olahpesan SMS tempat Anda menambahkan nomor telepon tujuan.
4. Di panel navigasi, pilih Text messaging (SMS) (Pesan teks).
5. Di halaman Mobile text messaging (SMS) (Olahpesan teks seluler), di bawah Sandbox destination phone numbers (Nomor telepon tujuan sandbox), pilih nomor telepon yang akan dihapus, lalu pilih Delete phone number (Hapus nomor telepon).
6. Untuk mengonfirmasi bahwa Anda ingin menghapus nomor telepon, masukkan **`delete me`**, lalu pilih Delete (Hapus).

   Jika 24 jam atau lebih telah berlalu sejak Anda memverifikasi atau mencoba untuk memverifikasi nomor telepon tujuan, nomor itu akan dihapus, dan Amazon SNS memperbarui daftar nomor telepon tujuan Anda.
7. Ulangi langkah-langkah ini di setiap Wilayah tempat Anda menambahkan nomor telepon tujuan ini dan tidak akan menggunakannya lagi.

Keluar dari sandbox SMS

Memindahkan Anda Akun AWS keluar dari kotak pasir SMS mengharuskan Anda menambahkan, memverifikasi, dan menguji nomor telepon tujuan terlebih dahulu. Kemudian, Anda harus membuat kasus dengan AWS Support.

Untuk meminta agar AWS akun Anda dipindahkan dari kotak pasir SMS

1. Verifikasi nomor telepon

   a. Saat Anda Akun AWS berada di kotak pasir SMS, buka konsol Amazon SNS.

   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. Uji penerbitan SMS

- Konfirmasikan bahwa Anda dapat mengirim dan menerima pesan ke setidaknya satu nomor telepon terverifikasi. Untuk petunjuk lebih rinci tentang cara mempublikasikan pesan SMS, lihatMenerbitkan ke ponsel.

3. Memulai suntingan kotak pasir

   - Pada halaman pesan teks Seluler (SMS) konsol Amazon SNS, di bawah Informasi akun, pilih Keluar dari kotak pasir SMS. 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 Teks SNS sebagai layanan.

     ii. Berikan URL situs web atau nama aplikasi tempat Anda ingin mengirim pesan SMS.

     iii. Tentukan jenis pesan yang akan Anda kirim: One Time Password, Promotional, atau Transactional.

     iv. Pilih Wilayah AWSdari mana Anda akan mengirim pesan SMS.

     v. Buat daftar negara atau wilayah tempat Anda berencana mengirim pesan SMS.

     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 limit dalam USD yang 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 asal untuk pesan SMS

Ketika Anda mengirim pesan SMS menggunakan Amazon SNS, Anda dapat menjelaskan identitas diri Anda kepada penerima Anda menggunakan jenis identitas asal:

- ID Pengirim
- Nomor asal

> **Note**
>
> Olahpesan SMS Amazon SNS tersedia di Wilayah di mana Amazon Pinpoint saat ini tidak didukung. Jika Anda beroperasi di Europe (Stockholm), Middle East (Bahrain), Europe (Paris), South America (Sao Paulo), atau US West (N. California), buka konsol Amazon Pinpoint di Wilayah US East (N. Virginia) untuk mendaftarkan perusahaan dan kampanye 10DLC Anda, tetapi jangan meminta nomor 10DLC. Sebagai gantinya, gunakan konsolAWS Service Quotas untuk membuat kasus peningkatan batas layanan saat meminta nomor 10DLC untuk Wilayah tersebut. Untuk mempelajari selengkapnya tentang cara meminta identitas awal, lihat Meminta dukungan untuk olahpesan SMS dengan Amazon SNS.

ID Pengirim

ID pengirim adalah nama abjad yang mengidentifikasi pengirim pesan SMS. Bila Anda mengirim pesan SMS menggunakan ID pengirim, dan penerima berada di area di mana autentikasi ID pengirim

didukung, ID pengirim Anda muncul di perangkat penerima, bukan nomor telepon. ID pengirim memberikan lebih banyak informasi tentang pengirim kepada penerima SMS daripada yang diberikan oleh nomor telepon, kode panjang, atau kode pendek.

ID pengirim didukung di beberapa negara dan wilayah di seluruh dunia. Di beberapa tempat, jika Anda adalah bisnis yang mengirim pesan SMS 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 ID pengirim, lihatNegara dan wilayah yang didukung.

Tidak ada biaya tambahan untuk menggunakan ID pengirim. Namun, dukungan dan persyaratan untuk autentikasi ID pengirim bervariasi. Beberapa pasar utama (termasuk Kanada, Cina, dan Amerika Serikat) tidak mendukung penggunaan ID pengirim. Beberapa area mengharuskan perusahaan yang mengirim pesan SMS ke pelanggan perorangan harus menggunakan ID pengirim yang telah terdaftar sebelumnya dengan badan pengawas atau grup industri.

> **Important**
>
> AWSmelarang spoofing SMS, di mana ID pengirim digunakan untuk menyamar sebagai orang lain, perusahaan, atau produk. Hanya gunakan ID pengirim yang mewakili merek atau merek dagang yang Anda miliki.

Keuntungan

ID pengirim memberi penerima informasi selengkapnya tentang pengirim pesan. Lebih mudah untuk menetapkan identitas merek Anda dengan menggunakan ID pengirim daripada dengan menggunakan kode pendek atau panjang. Tidak ada biaya tambahan untuk menggunakan ID pengirim.

Kekurangan

Support dan persyaratan untuk otentikasi ID pengirim tidak konsisten di semua negara atau wilayah. Beberapa pasar utama (termasuk Kanada, China, dan Amerika Serikat) tidak mendukung ID pengirim. Di beberapa area, Anda harus memiliki ID pengirim yang telah disetujui sebelumnya oleh badan pengatur sebelum Anda dapat menggunakannya.

Pendaftaran ID pengirim menurut negara

Anda perlu membuka kasing dengan AWS dukungan untuk mendaftarkan ID pengirim untuk pesan SMS. Setelah mengangkat kasus dukungan, AWS akan berbagi dokumen tambahan yang diperlukan. Anda juga harus memberikan informasi berikut untuk negara yang sesuai tempat Anda mendaftarkan ID pengirim.

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Australia (AU)	Transaksi dan promosi	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus • ID pengirim harus nama merek perusahaan yang mengirim SMS	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran

			tentang hubungan antara nama perusahaan dan ID pengirim
			• Nomor izin bisnis/perdagangan resmi perusahaan atau PPN #
			• Template pesan yang ingin Anda kirim
			• Izin pendaftaran bisnis. Contohnya termasuk tetapi tidak terbatas pada:
			  • Nomor Bisnis Australia (ABN)
			  • Nomor Perusahaan Australia (ACN)
			  • Nomor Badan Terdaftar Australia (ARBN)
			  • Nomor Perusahaan Adat (ICN)
			• Surat Otorisasi (LOA)

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Belarusia (OLEH)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus • ID pengirim harus nama merek perusahaan yang mengirim SMS	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Nomor izin bisnis/perdagangan resmi perusahaan atau PPN # • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Tiongkok (CN) China tidak memerlukan pendaftaran ID pengirim, tetapi memang memerlukan pendaftaran konten/templat dengan tanda tangan badan yang telah ditambahkan (misalnya, [Amazon]).	Kata kunci berikut tidak diperbolehkan: • Falung Gong • SB • Lapangan Tiananmen Pesan dari kategori berikut: • Kartu kredit • Pembayaran digital (termasuk cryptocurrency) • URL lalu lintas penipuan (phishing atau spam) • Judi • Konten yang tidak pantas (dewasa, kekerasan, narkoba, alkohol) • Pinjaman • Operasi plastik • Politik • Agama • Perdagangan saham • Mata uang virtual	• Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• Nama perusahaan • Alamat perusahaan • Negara Bagian/Propinsi • Negara • Nomor telepon perusahaan • Situs web perusahaan • Perkiraan volume bulanan • Jenis pesan: promosional/transaksional • Penjelasan kasus penggunaan • Template yang ingin Anda daftarkan (SMS yang dikirim tidak boleh menyimpang dari templat yang disediakan untuk memastikan kepatuhan dan pengiriman yang berhasil). Misalnya, [CompanyName] Kata sandi satu kali Anda adalah {OTP}. Kode ini

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
	Tanda tangan dalam tanda kurung siku harus dimasukkan ke badan pesan SMS. Agar pengiriman berhasil ke China, Anda diharuskan mendaftarkan tanda tangan pesan dengan templat konten pesan Anda. Tanda tangan pesan ini harus dilampiskan ke konten pesan pada setiap SMS yang dikirim. Tidak mendahului badan pesan SMS dengan tanda tangan terdaftar dapat mengakibatkan SMS diblokir atau difilter. Tanda tangan harus dimasukkan ke badan SMS dalam tanda kurung siku.		akan kedaluwarsa dalam 10 menit. • Konfirmasi bahwa Anda tidak akan mengirim konten promosi sebagai jenis pesan transaksional • Tanda tangan pesan. Lihat Pembatasan dan persyaratan format tanda tangan.

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Mesir (EG)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Apakah perusahaan Anda memiliki entitas bisnis/kantor di Mesir? Atau apakah ini perusahaan yang berbasis di non-Mesir yang mengirim ke Mesir? • Konfirmasi tertulis bahwa kasus penggunaan mencakup semua pesan yang akan dikirim oleh ID pengirim ini • Jika tidak jelas, berikan hubungan antara nama perusahaan dan ID pengirim • Template pesan yang ingin Anda kirim
Indonesia (IN)	Pesan transaksional saja	• Alfanumerik • Maksimal 6 karakter • Tidak ada spasi • Tidak ada karakter khusus	Lihat Persyaratan pendaftaran ID pengirim untuk India.

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Yordania (JO)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Sertifikat pendaftaran bisnis • Jenis pesan yang Anda rencanakan untuk dikirim (misalnya, OTP, peringatan) • Konfirmasi tertulis bahwa kasus penggunaan menjelaskan semua pesan yang akan dikirim oleh ID pengirim ini • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Kuwait (KW)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jenis pesan yang Anda rencanakan untuk dikirim (misalnya, OTP, peringatan) • Konfirmasi tertulis bahwa kasus penggunaan menjelaskan semua pesan yang akan dikirim oleh ID pengirim ini • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Filipina (PH)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jenis pesan yang Anda rencanakan untuk dikirim (misalnya, OTP, peringatan) • Konfirmasi tertulis bahwa kasus penggunaan menjelaskan semua pesan yang akan dikirim oleh ID pengirim ini • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Qatar (QA)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jenis SMS yang dikirim • (Transaksional/ Promosional/ OTP) Perhatikan bahwa hanya pesan transaksional/OTP yang dapat digunakan dengan ID pengirim yang ditujukan ke Qatar agar sesuai. • Konfirmasi tertulis bahwa kasus penggunaan menjelaskan semua pesan yang akan dikirim oleh ID pengirim ini • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Rusia (RU)	Pesan transaksional saja	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • ID Pajak atau nomor lisensi • Sertifikat pendaftaran bisnis • Alamat email kontak • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim • Konfirmasi bahwa kasus penggunaan ini akan berlaku untuk semua pesan yang dikirim ke Rusia dari akun ini • Pengakuan bahwa pesan non-transaksional harus dikirim menggunakan ID pengirim terpisah • $272/biaya bulanan harap konfirmasikan bahwa Anda menyetujui biaya bulanan berulang ini: Ya/Tidak • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Arab Saudi (SA)	Setiap ID pengirim harus bersifat transaksional atau promosi. Satu ID pengirim tidak dapat digunakan untuk kedua jenis lalu lintas. Jika mengirim lalu lintas OTP atau 2FA, ID pengirim harus digunakan hanya untuk tujuan itu. ID pengirim promosi akan tunduk pada daftar Jangan Ganggu (DND) Arab Saudi.	• Panjang ID pengirim promo: 2 - 8 karakter, ID pengirim ditambah dengan “-AD” • Panjang ID pengirim transaksi onal: 2 - 11 karakter • ID Pengirim harus mewakili identitas merek pengirim • Sertakan setidakny a satu huruf • Jangan gunakan karakter khusus ASCII (misalnya, #, @) • Anda dapat memasukkan huruf besar dan kecil, dan angka 0 - 9	Support untuk pendaftaran ID pengirim di Arab Saudi hanya untuk perusahaan internasi onal. Saat ini kami tidak mendukung perusahaan lokal Arab Saudi untuk mendaftarkan ID pengirim • ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaa n (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda)

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim • Template pesan yang Anda rencanakan untuk dikirim. • Apakah ID pengirim ini akan digunakan untuk konten transaksional atau promosi? • Pengakuan bahwa pesan non-transaksional harus dikirim menggunakan ID pengirim terpisah • Jika mengirimkan lalu lintas 2FA atau OTP, konfirmasi bahwa ID Pengirim ini hanya akan

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			digunakan untuk tujuan ini
Singapura (SG)	Pesan transaksional saja	• Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	Lihat Persyaratan pendaftaran ID pengirim untuk Singapura.

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Sri Lanka (LK)	Tidak ada batasan atau persyaratan khusus	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Jenis perusahaan (lokal/internasional) • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Penjelasan kasus penggunaan, tujuan pesan • Template pesan yang ingin Anda kirim • Apakah ID pengirim ini akan digunakan untuk konten transaksional atau promosi? • Pengakuan bahwa pesan non-transaksional harus dikirim menggunakan ID pengirim terpisah

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jika mengirimkan lalu lintas 2FA atau OTP, konfirmasi bahwa ID Pengirim ini hanya akan digunakan untuk tujuan ini

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Thailand (TH)	Tidak ada batasan atau persyaratan khusus	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jenis SMS yang dikirim (transaksional/promosional/OTP) • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Turki (TR)	Tidak ada batasan atau persyaratan khusus	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Jika perusahaan Anda lokal atau Internasional ke Turki • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jenis SMS yang dikirim (transaksional/promosional/OTP) • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Ukraina (UA)	Tidak ada batasan atau persyaratan khusus	• Alfanumerik • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Nomor PPN • Perusahaan lokal atau internasional • Perkiraan volume bulanan • Penjelasan kasus penggunaan, tujuan pesan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Jenis SMS yang dikirim (transaksional/promosional/OTP) • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Uni Emirat Arab (AE)	Pesan transaksional saja	• Alfanumerik • ID pengirim generik tidak diizinkan, dan harus mengidentifikasi merek • Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Nama perusahaan • Alamat perusahaan (termasuk kota perusahaan, negara bagian/provinsi, kode pos) • Negara perusahaan • Nomor telepon kontak perusahaan • URL Perusahaan (tautan ke aplikasi atau situs web perusahaan Anda) • Perkiraan volume bulanan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim • Penjelasan kasus penggunaan, tujuan pesan

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
			• Nomor izin bisnis/perdagangan resmi perusahaan atau PPN # • Konfirmasi tertulis bahwa semua lalu lintas yang dikirim oleh ID pengirim ini dijelaskan oleh kasus penggunaan yang diberikan • Template pesan yang ingin Anda kirim

Nama negara	Jenis pesan	Pembatasan dan persyaratan format	Persyaratan pendaftaran
Vietnam (VN)	Pesan transaksional saja. Pesan pemasaran dan promosi tidak diperbolehkan. Konten terlarang meliputi: • Konten dewasa • Layanan amal • Mata Uang Kripto • undian • Perjudian seluler/kasino • Taruhan olahraga • Pemungutan suara	• Maksimal 11 karakter • Tidak ada spasi • Tidak ada karakter khusus	• ID pengirim yang ingin Anda daftarkan • Dari AWS wilayah mana pengguna akan memanggil API/layanan • Perkiraan volume bulanan • Jika tidak jelas, berikan penjelasan tentang hubungan antara nama perusahaan dan ID pengirim • Penjelasan kasus penggunaan, tujuan pesan • Konfirmasi tertulis bahwa semua lalu lintas yang dikirim oleh ID pengirim ini dijelaskan oleh kasus penggunaan yang diberikan

Pembatasan dan persyaratan format tanda tangan

Agar pengiriman berhasil ke China, Anda diharuskan mendaftarkan tanda tangan pesan dengan templat konten pesan Anda. Tanda tangan pesan ini harus dilampiaskan ke konten pesan pada setiap SMS yang dikirim. Tidak mendahului badan pesan SMS dengan tanda tangan terdaftar dapat mengakibatkan SMS diblokir atau difilter.

- Tanda tangan harus dimasukkan ke badan SMS dalam tanda kurung siku
- Teks standar harus terkandung dalam tanda kurung siku
- Teks unicode harus menggunakan tanda kurung lentikular untuk memuat tanda tangan - U +3010 LEFT LENTICULAR BRACKET dan U+3011 RIGHT LENTICULAR BRACKET - Contoh: [Pemberitahuan]
- Harus antara 3 — 11 karakter
- Karakter Cina/Inggris didukung

Persyaratan ID Pengirim untuk Prancis

Panduan ini memberikan langkah-langkah dan pedoman yang diperlukan untuk membuat ID Pengirim khusus yang diperlukan oleh operator seluler Prancis untuk mengirim pesan teks SMS ke Prancis.

Topik

- Menyiapkan ID Pengirim khusus untuk Prancis
- Panduan penamaan ID pengirim

Menyiapkan ID Pengirim khusus untuk Prancis

Anda dapat menggunakan salah satu metode berikut untuk menyiapkan ID Pengirim khusus. Amazon SNS akan menggunakan ID Pengirim atas nama Anda untuk pesan SMS yang dipublikasikan menggunakan`Publish` API.

- Anda dapat menggunakan konsol Amazon SNS untuk mengonfigurasi ID Pengirim default yang akan digunakan untuk semua pesan SMS yang diterbitkan. Untuk mempelajari lebih lanjut kunjungan,Mengatur preferensi pesan SMS menggunakan AWS Management Console.
- Anda dapat menggunakan`Publish` API untuk menyetel ID Pengirim menggunakan atribut`AWS.SNS.SMS.SenderID` pesan saat meminta Amazon SNS untuk mempublikasikan pesan SMS. Untuk mempelajari lebih lanjut kunjungan,Mengirim pesan (konsol).

Panduan penamaan ID pengirim

- Nama ID Pengirim harus berupa alfanumerik dengan maksimal 11 karakter.
- Nama ID Pengirim tidak boleh berisi karakter khusus atau spasi.

- Kami menyarankan agar Anda menggunakan nama yang sama untuk ID Pengirim dan nama merek perusahaan yang mengirim pesan teks SMS.

Persyaratan pendaftaran ID pengirim untuk India

Secara default, ketika Anda mengirim pesan ke penerima di India, Amazon SNS menggunakan koneksi International Long Distance Operator (ILDO) untuk mengirimkan pesan tersebut. Saat penerima melihat pesan yang dikirim melalui koneksi ILDO, pesan tersebut tampaknya dikirim dari ID numerik acak (kecuali jika Anda membeli kode pendek khusus).

> **Note**
>
> Harga untuk mengirim pesan menggunakan rute lokal ditampilkan di halaman Harga SMS Amazon SNS di Seluruh Dunia. Harga untuk mengirim pesan menggunakan koneksi ILDO lebih tinggi dari harga untuk mengirim pesan melalui rute lokal.

Jika Anda lebih suka menggunakan ID pengirim abjad untuk pesan SMS Anda, Anda harus mengirim pesan tersebut melalui rute lokal, bukan rute ILDO. Untuk mengirim pesan menggunakan rute lokal, Anda harus terlebih dahulu mendaftarkan kasus penggunaan dan templat pesan Anda pada Telecom Regulatory Authority of India (TRAI) melalui portal Distributed Ledger Technology (DLT). Persyaratan pendaftaran ini dirancang untuk mengurangi jumlah pesan yang tidak diminta yang diterima konsumen India dan untuk melindungi konsumen dari pesan yang berpotensi berbahaya. Proses pendaftaran ini dikelola oleh Vodafone India melalui layanan Vilpower-nya.

Topik

- Langkah 1: Mendaftar pada TRAI
- Langkah 2: Meminta ID pengirim
- Langkah 3: Mengirim pesan SMS
- Pemecahan masalah pesan SMS yang dikirim ke penerima di India

Langkah 1: Mendaftar pada TRAI

Sebelum Anda dapat mengirim pesan SMS ke penerima di India, Anda harus mendaftarkan organisasi Anda ke Telecom Regulatory Authority of India (TRAI). Bersiaplah untuk memberikan informasi berikut selama proses pendaftaran:

- Nomor Akun Permanen (Permanent Account Number/PAN) organisasi Anda.
- Nomor Rekening Pengurangan Pajak (Tax Deduction Account Number/TAN) organisasi Anda.
- Nomor Identifikasi Pajak Barang dan Jasa (Goods and Services Tax Identification Number/GSTIN) organisasi Anda.
- Nomor Identitas Perusahaan (Corporate Identity Number/CIN) organisasi Anda.
- Surat otorisasi yang memberi Anda wewenang untuk mendaftarkan organisasi Anda.

Berikut ini adalah daftar contoh dari beberapa situs pendaftaran Distributed Ledger Technology (DLT) yang dapat Anda gunakan untuk mendaftarkan organisasi Anda dengan TRAI (biaya mungkin berlaku). Proses pendaftaran bervariasi menurut situs. Hubungi tim dukungan masing-masing untuk bantuan.

- BSNL DLT - Pendaftaran gratis.
- Jio Trueconnect - Membebankan biaya untuk menyelesaikan proses pendaftaran.
- Solusi Perusahaan Cerdas - Membebankan biaya untuk menyelesaikan proses pendaftaran.
- Vilpower - Termasuk template yang dapat Anda download dan memodifikasi sesuai kebutuhan Anda. Vilpower mengenakan biaya untuk melakukan proses pendaftaran.

Pelajari cara mendaftarkan organisasi Anda pada TRAI.

Berikut ini merinci cara mendaftarkan organisasi Anda ke TRAI menggunakan Vilpower.

1. Di browser web, buka situs web Vilpower dihttps://www.vilpower.in.
2. Pilih Signup (Pendaftaran) untuk membuat akun lain. Selama proses pendaftaran, lakukan hal berikut:
   - Untuk jenis entitas yang akan didaftarkan, pilih As Enterprise (Sebagai Perusahaan/Korporasi).
   - Untuk Telemarketer Name (Nama Telemarketer), gunakan Infobip Private Limited - ALL (Infobip Privat Terbatas - SEMUA). Saat diminta, ketik **`Infobip`** lalu pilih Infobip Private Limited - ALL (Infobip Privat Terbatas - SEMUA) dari daftar dropdown.
   - Untuk Enter Telemarketer ID (Masukkan ID Telemarketer), masukkan **`110200001152`**.
   - Saat diminta untuk memberikan ID Header Anda, masukkan ID pengirim yang ingin Anda daftarkan.

> **Note**
>
> India membutuhkan ID pengirim yang panjangnya tepat enam karakter.

- Saat diminta untuk memberikan Templat Konten, masukkan konten pesan yang akan dikirim ke penerima. Sertakan templat untuk setiap pesan yang akan dikirim.

> **Note**
>
> Situs web penyedia pendaftaran DLT tidak dikelola oleh Amazon Web Services. Langkah-langkah di situs web mereka dapat berubah.

Langkah 2: Meminta ID pengirim

Untuk meminta ID pengirim di India, Anda harus mengajukan permintaan AWS Support. Selesaikan langkah-langkah dalam Meminta ID pengirim. Dalam permintaan Anda, berikan informasi berikut:

- Wilayah AWS asal tempat pengirim berencana untuk mengirim pesan SMS.
- Nama perusahaan yang digunakan selama proses pendaftaran DLT.
- ID Entitas Utama (Principal Entity ID/PEID) yang Anda terima setelah pendaftaran entitas DLT berhasil.
- Perkiraan volume bulanan.
- Penjelasan kasus penggunaan Anda.
- Deskripsi alur keikutsertaan pengguna akhir.
- Konfirmasi bahwa keikutsertaan pengguna akhir dikumpulkan dan terdaftar.

Langkah 3: Mengirim pesan SMS

Setelah mendaftarkan organisasi Anda pada TRAI, ikuti prosedur ini untuk mengirim pesan SMS ke penerima di India.

1. Masuk ke konsol Amazon SNS.
2. Di menu konsol tersebut, atur pemilih wilayah ke wilayah yang mendukung olahpesan SMS.
3. Di panel navigasi, pilih Text messaging (SMS) (Olahpesan teks (SMS)).

4. Di halaman Mobile text messaging (SMS) (Pesan teks seluler (SMS)), pilih Publish text message (Publikasikan pesan teks). Jendela Publish text message (Publikasikan pesan teks) terbuka.
5. Untuk Message type (Jenis pesan), pilih salah satu jenis berikut:

   - Promotional (Promosi) – Pesan tidak penting, seperti pesan pemasaran.

     Saat menggunakan ID Pengirim numerik, pilih opsi ini.
   - Transactional (Transaksional) – Pesan penting yang mendukung transaksi pelanggan, seperti kode sandi satu kali (OTP) untuk autentikasi multi-faktor.

     Saat menggunakan ID Pengirim alfabet atau alfanumerik, pilih opsi ini.

   Pengaturan tingkat pesan ini menimpa jenis pesan default Anda, yang Anda tetapkan di halaman Text messaging preferences (Preferensi olahpesan teks).

   Untuk informasi harga untuk pesan promosi dan transaksional, lihat Harga SMS Global.
6. Untuk Nomor, masukkan nomor telepon yang ingin Anda kirimi pesan.
7. Untuk Message (Pesan), masukkan pesan yang akan dikirim.

   Saat menambahkan konten ke pesan SMS, pastikan konten itu sama persis dengan konten dalam templat terdaftar DLT. Operator akan memblokir pesan SMS jika konten pesan berisi karakter tambahan, spasi, tanda baca, atau kalimat yang tidak cocok. Variabel dalam templat dapat memiliki 30 karakter atau lebih sedikit.
8. Di bagian Identitas Originasi, untuk ID Pengirim, masukkan ID kustom yang berisi 3-11 karakter.

   ID Pengirim dapat berupa angka untuk pesan promosi, atau abjad atau alfanumerik untuk pesan transaksional. ID Pengirim ditampilkan sebagai pengirim pesan di perangkat penerima.

   Untuk ID Pengirim promosi numerik yang terdaftar di India, tentukan ID Pengirim sebagai parameter Nomor Originasi dalam permintaan pengiriman SMS.
9. Perluas Country-specific attributes (Atribut khusus negara), dan masukkan atribut yang diperlukan berikut untuk mengirim pesan SMS ke penerima di India:

   - Entity ID (ID entitas) – ID entitas atau ID entitas utama (principal entity/PE) untuk mengirim pesan SMS ke penerima di India.

     Ini adalah string khusus yang disediakan TRAI dengan 1—50 karakter yang secara unik mengidentifikasi entitas yang Anda daftarkan dengan TRAI.

- Template ID (ID templat) – ID templat yang Anda dapatkan dari badan pengawas untuk mengirim pesan SMS ke penerima di India.

  Ini adalah string khusus yang disediakan TRAI dari 1-50 karakter yang secara unik mengidentifikasi template yang Anda daftarkan dengan TRAI. ID templat harus dikaitkan dengan ID Pengirim yang Anda tentukan pada langkah sebelumnya, dan dengan konten pesan.

10. Pilih Publish message (Publikasikan Pesan).

Untuk informasi tentang pengiriman pesan SMS ke penerima di negara-negara lain, lihat Menerbitkan ke ponsel.

Pemecahan masalah pesan SMS yang dikirim ke penerima di India

Berikut adalah beberapa alasan yang mebuat operator memblokir pesan SMS:

- Tidak ada template yang ditemukan yang cocok dengan konten yang dikirim.

  Konten yang dikirim: **<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  Templat yang cocok: Tidak ada

  Masalah: Tidak ada templat DLT yang mencakup <#> atau `{#var#}` di awal templat terdaftar DLT.

- Nilai variabel melebihi 30 karakter.

  Konten yang dikirim: **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  Templat yang cocok: **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

  Masalah: Nilai “ABC Company - India Private Limited” dalam konten yang dikirim melebihi satu `{#var#}` batas 30 karakter.

- Kasus kalimat pesan tidak cocok dengan kasus kalimat dalam templat.

Konten yang dikirim: **`12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.`**

Templat yang cocok: **`{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.`**

Masalah: Nama perusahaan yang ditambahkan ke templat DLT yang cocok ditulis dalam huruf besar sementara konten yang dikirim telah mengubah bagian namanya menjadi huruf kecil — "ABC Pvt. Ltd." vs "ABC PVT. LTD."

Persyaratan pendaftaran ID pengirim untuk Singapura

Pelanggan Amazon SNS dapat mengirimkan lalu lintas SMS di Singapura menggunakan Sender ID yang telah didaftarkan melalui Singapore SMS Sender ID Registry (SSIR). SSIR diluncurkan pada Maret 2022 melalui Singapore Network Information Centre (SGNIC) yang dimiliki oleh Info-Communications Media Development Authority (IMDA) Singapura, dan memungkinkan organisasi untuk mendaftarkan ID Pengirim mereka saat mengirim SMS ke ponsel di Singapura.

Untuk menggunakan ID Pengirim Singapura yang terdaftar, Anda harus mendapatkan Unique Entity Number (UEN), kemudian mengirimkan permintaan ke Amazon untuk mengizinkan daftar akun Anda untuk penggunaan ID Pengirim Anda dan akhirnya menyelesaikan proses pendaftaran melalui SSIR.

Jika Anda tidak mendaftarkan ID Anda pada tahun 2023—01—30, pesan apa pun yang dikirim menggunakan ID Pengirim akan diubah menjadi LIKELY-SCAM per aturan badan pengatur. Setelah tanggal ini, regulator akan terus memfilter atau memblokir lalu lintas yang tidak terdaftar atas kebijaksanaan mereka.

> **Important**
>
> Jika Anda meminta ID Pengirim di wilayah Amazon Pinpoint, gunakan konsol Amazon Pinpoint untuk mendaftarkan ID Pengirim. Untuk menyelesaikan proses pendaftaran secara manual untuk wilayah selain wilayah Amazon Pinpoint, gunakan pendaftaran ID Pengirim Singapura.
>
> Untuk memastikan Anda masih dapat mengirim pesan di Singapura, pendaftaran Anda harus diselesaikan pada tahun 2023—01—30.
>
> Sangat penting bagi Anda untuk menyelesaikan langkah-langkah pendaftaran dengan urutan berikut. Melakukan langkah-langkah ini di luar pesanan dapat mengakibatkan ID Pengirim

Anda diblokir oleh layanan atau akan mencegah ID Pengirim Anda disimpan di perangkat seluler.
Langkah 1. Mendaftar untuk Nomor Entitas Unik Singapura (UEN)
Langkah 2. Jika Anda meminta ID Pengirim di Wilayah Amazon Pinpoint, gunakan petunjuk pendaftaran ID Pengirim Amazon Pinpoint untuk mendaftarkan ID Pengirim.

- Untuk mendaftarkan ID Pengirim yang akunnya tidak berada di Wilayah Amazon Pinpoint, gunakan petunjuk pendaftaran ID Pengirim Singapura untuk mendaftarkan ID Pengirim secara manual.
- Saat mengirim pesan teks SMS atas nama perusahaan lain, diperlukan Letter of Authorization (LOA) dari perusahaan.
- Jangan menunggu persetujuan atau perubahan status setelah mengirimkan pendaftaran IDAWS Pengirim Anda. Segera ke Langkah 3.

Langkah 3. Mendaftarkan ID Pengirim ke Singapore Network Information Centre (SGNIC)

Topik

- Mendaftar untuk Nomor Entitas Unik Singapura (UEN)
- Mendaftarkan ID Pengirim Singapura Anda dengan Amazon Pinpoint
- Proses registrasi manual untuk menyelesaikan pendaftaran ID pengirim Singapura
- Mendaftarkan ID Pengirim ke Singapore Network Information Centre (SGNIC)
- Status pendaftaran ID Pengirim Singapura
- Mengedit pendaftaran ID Pengirim Singapura
- Menghapus pendaftaran ID Pengirim Singapura
- Masalah Pendaftaran Singapura
- Pertanyaan umum pendaftaran ID Pengirim Singapura

Mendaftar untuk Nomor Entitas Unik Singapura (UEN)

Untuk memulai pendaftaran dengan SSIR, Anda harus terlebih dahulu mendapatkan Singapore Unique Entity Number (UEN). UEN adalah nomor entitas unik yang Anda terima saat mendaftarkan bisnis Anda ke Account and Corporate Registry Authority (ACRA), untuk informasi lebih lanjut lihat

Siapa yang Harus Mendaftar dengan ACRA? . Jumlah waktu untuk memproses dapat bervariasi tergantung pada seberapa mudah ACRA dapat memvalidasi permintaan Anda.

Mendaftarkan ID Pengirim Singapura Anda dengan Amazon Pinpoint

Setelah mendaftarkan Singapore Unique Entity Number (UEN), Anda dapat menyelesaikan proses pendaftaran ID Pengirim di konsol Amazon Pinpoint (hanya untuk wilayah Amazon Pinpoint). Saat mendaftarkan ID Pengirim Anda, pastikan informasinya lengkap dan akurat atau pendaftaran Anda dapat ditolak.

> **Important**
>
> Informasi yang Anda kirimkan melalui konsol Amazon Pinpoint akan diteruskan ke mitra operator kami untuk menyelesaikan pendaftaran.

Untuk mendaftarkan ID Pengirim Singapura

Gunakan langkah-langkah ini untuk mendaftarkan ID Pengirim saat akun berada di Wilayah Amazon Pinpoint. Jika akun Anda tidak berada di Wilayah Amazon Pinpoint, lihatProses registrasi manual untuk menyelesaikan pendaftaran ID pengirim Singapura.

1. Masuk keAWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS and voice (SMS and voice), pilih Nomor telepon (SMS dan suara), pilih Nomor telepon.
3. Pada tab Pendaftaran ID Pengirim, pilih Buat pendaftaran.
4. Pilih Singapura sebagai negara tujuan Anda.
5. Di bagian Informasi Perusahaan, masukkan yang berikut:
   - Untuk Nama Perusahaan, masukkan nama perusahaan Anda persis seperti yang muncul pada pendaftaran UEN Anda.
   - Untuk ID Pajak, masukkan nomor UEN yang Anda terima dari ACRA.
   - Untuk Situs Web Perusahaan, masukkan URL untuk situs web perusahaan Anda.
   - Untuk Alamat 1, masukkan alamat jalan kantor pusat perusahaan Anda.
   - Untuk Alamat 2 - opsional, jika diperlukan masukkan nomor suite kantor pusat perusahaan Anda.

- Untuk City, masuki kota kantor pusat perusahaan Anda.
- Untuk Negara, masukkan negara bagian kantor pusat perusahaan Anda.
- Untuk Kode Pos, masukkan kode pos kantor pusat perusahaan Anda.
- Untuk Negara, masukkan dua digit kode negara ISO.

6. Di bagian Informasi Kontak, masukkan informasi berikut:
   - Untuk Nama Depan, masukkan nama depan orang yang akan menjadi titik kontak bisnis Anda.
   - Untuk Nama Belakang, masukkan nama belakang orang yang akan menjadi titik kontak bisnis Anda.
   - Untuk Email Support, masukkan alamat email orang yang akan menjadi titik kontak bisnis Anda.
   - Untuk Nomor Telepon Support, masukkan nomor telepon orang yang akan menjadi titik kontak bisnis Anda.
7. Di Informasi ID Pengirim, masukkan yang berikut ini:
   - Untuk ID Pengirim, masukkan ID Pengirim yang ingin ditampilkan untuk pesan Anda.
   - Untuk Mendaftar atas nama merek/entitas lain? jika ya maka pilih Benar. Jika Anda bukan pengguna akhir yang mengirim pesan, Anda dianggap sebagai “Perwakilan” dari merek/entitas lain.
   - Untuk Letter of authorization image — opsional, jika Anda mencentang kotak sebagai Mendaftar atas nama merek/entitas lain? , unggah gambar Letter of Authorization (LOA) lengkap. Jenis file yang didukung adalah PNG dan ukuran file maksimum adalah 400KB. Template untuk LOA dapat diunduh untuk kenyamanan Anda.
   - Untuk koneksi ID Pengirim — opsional Anda dapat menambahkan rincian lebih lanjut tentang koneksi antara senderID yang diminta dan nama perusahaan.
8. Di Messaging Use Case (Kasus Penggunaan Pesan), lakukan hal berikut:
   - Untuk Volume SMS Bulanan pilih jumlah pesan SMS yang akan setiap bulan.
   - Untuk Use Case Category pilih salah satu dari jenis kasus penggunaan berikut untuk nomornya:
     - Otentikasi dua faktor - Gunakan ini untuk mengirim dua kode otentikasi faktor.
     - Kata sandi satu kali — Gunakan ini untuk mengirim kata sandi satu kali kepada pengguna.
     - Notifikasi - Gunakan ini jika Anda hanya bermaksud mengirim pemberitahuan penting kepada pengguna Anda.
     - Polling dan survei - Gunakan ini untuk polling pengguna pada preferensi mereka.
     - Info on demand - Ini untuk mengirim pesan kepada pengguna setelah mereka mengirim permintaan.

    - Promosi dan Pemasaran - Gunakan ini jika Anda hanya bermaksud mengirim pesan pemasaran ke pengguna Anda.
    - Lainnya - Gunakan ini jika kasus penggunaan Anda tidak termasuk dalam kategori lain. Pastikan Anda mengisi Use Case Details untuk opsi ini.
  - Rincian Kasus Penggunaan Lengkap - opsional untuk memberikan konteks tambahan ke Kategori Kasus Penggunaan yang dipilih.
9. Di bagian Messaging Sampel, lakukan hal berikut:
   - Untuk Contoh Pesan 1, masukkan contoh pesan dari badan pesan SMS yang akan dikirim ke pengguna akhir Anda.
   - Untuk Contoh Pesan 2 - opsional dan Sampel Pesan 3 - opsional, masukkan contoh pesan tambahan, jika diperlukan, dari badan pesan SMS yang akan dikirim.
   - Setiap kotak teks Message Sample memiliki batas karakter maksimum 306 karakter.
10. Setelah Anda selesai, pilih Kirim pendaftaran.

> **Important**
>
> Anda dapat memeriksa status pendaftaran Anda dengan mengikuti petunjuk diStatus pendaftaran ID Pengirim Singapura.
> Jangan menunggu persetujuan atau perubahan status setelah mengirimkan pendaftaran ID Pengirim Anda. Pergi segera keMendaftarkan ID Pengirim ke Singapore Network Information Centre (SGNIC).

Proses registrasi manual untuk menyelesaikan pendaftaran ID pengirim Singapura

Gunakan langkah-langkah ini untuk mendaftarkan ID Pengirim saat akun tidak berada di Wilayah Amazon Pinpoint. Jika akun Anda berada di Wilayah Amazon Pinpoint, lihatMendaftarkan ID Pengirim Singapura Anda dengan Amazon Pinpoint.

1. Unduh Singapore_Sender_ID_Registration_LOA_Template.zip dan lengkapi informasi yang diperlukan.
2. Buat kasus dengan AWSSupport.
3. Pada tab Buka kasus dukungan, pilih Buat kasus.
4. Pilih Mencari peningkatan batas layanan, dan untuk jenis batas pilih Pesan Teks SNS.
5. Untuk Jenis Sumber Daya, pilih Registrasi ID Pengirim.

6. Lampirkan dokumen LOA dan kirimkan permintaan.

Mendaftarkan ID Pengirim ke Singapore Network Information Centre (SGNIC)

> **Warning**
>
> Melakukan langkah-langkah ini di luar pesanan dapat mengakibatkan ID Pengirim Anda diblokir oleh layanan atau akan mencegah ID Pengirim Anda disimpan di perangkat seluler.

1. Anda harus terlebih dahulu mendaftarkan ID Pengirim Singapura (SG) untuk akun Anda DenganAWS (konsol Amazon Pinpoint, atau pendaftaran manual untuk wilayah non-Amazon Pinpoint). Setelah langkah ini selesai Anda dapat melanjutkan ke langkah berikutnya.
2. Bekerja dengan SGNIC untuk mendaftarkan ID Pengirim Anda menggunakan proses di SGNIC SMS Sender ID Registry.
   - Saat menyelesaikan proses, pastikan Anda mencantumkan semua hal berikut sebagai Agregator yang Berpartisipasi:
     - AMCS SG Private Limited (Layanan Komunikasi Media Amazon)
     - Nexmo PTE LTD
     - Sinch Singapore PTE LTD
     - Telesign Singapore PTE LTD
     - Twilio Singapura PTD LTD

> **Note**
>
> Anda diharuskan untuk mengirimkan registrasi ID Pengirim dari masing-masingAWS akun yang Anda perlukan untuk menggunakan ID Pengirim.

Status pendaftaran ID Pengirim Singapura

Ketika Anda mendaftarkan ID Pengirim Singapura Anda dengan Amazon SNS, pendaftaran Anda akan berada dalam salah satu dari lima status berbeda:

- Dibuat - Pendaftaran Anda dibuat tetapi tidak dikirimkan.
- Dikirimkan - Pendaftaran Anda telah dikirimkan dan sedang divalidasi.

- Meninjau - Pendaftaran Anda telah diterima dan sedang ditinjau. Mungkin diperlukan waktu antara 1-3 minggu dan dalam beberapa kasus mungkin butuh waktu lebih lama untuk peninjauan selesai.
- Selesai — Pendaftaran Anda telah disetujui dan Anda dapat mulai menggunakan ID Pengirim.
- Membutuhkan Pembaruan - Anda harus memperbaiki pendaftaran Anda dan mengirimkannya kembali. Lihat Mengedit pendaftaran ID Pengirim Singapura untuk informasi selengkapnya. Bidang yang memerlukan pembaruan menampilkan ikon peringatan dan deskripsi singkat tentang masalah tersebut.

Untuk semua wilayah selain wilayah Amazon Pinpoint, AWSSupport akan mengirimkan konfirmasi email pada pendaftaran atau membuat kasus dengan AWSSupport.

- Pada tab Buka kasus dukungan, pilih Buat kasus.
- PilihPeningkatan batas layanan.
- Untuk Jenis Sumber Daya, pilih Pendaftaran ID Pengirim dan untuk batas pilih Pertanyaan Umum.

Periksa status pendaftaran Anda

1. Masuk keAWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS and voice (SMS and voice), pilih Nomor telepon (SMS dan suara), pilih Nomor telepon.
3. Pada tab Pendaftaran ID Pengirim, pilih senderID.
4. Anda kemudian dapat melihat status pendaftaran setiap senderID.

Mengedit pendaftaran ID Pengirim Singapura

Setelah Anda mengirimkan pendaftaran Anda dengan Amazon Pinpoint, status pendaftaran akan ditampilkan sebagai Memerlukan Pembaruan jika ada masalah dengan pendaftaran. Dalam keadaan ini, formulir pendaftaran dapat diedit. Bidang yang memerlukan pembaruan memiliki ikon peringatan dan deskripsi singkat tentang masalah ini.

Mengedit ID Pengirim

1. Buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS and voice (SMS and voice), pilih Nomor telepon (SMS dan suara), pilih Nomor telepon.

3. Pada tab Registrasi SenderID, pilih nomor yang ingin Anda edit dan pilih ID Registrasi.
4. Pilih Perbarui pendaftaran untuk mengedit formulir dan memperbaiki bidang yang memiliki ikon peringatan.
5. Jika Anda mendaftar atas nama merek/entitas lain, Anda harus mengunggah ulang file yang dikirimkan sebelumnya untuk Letter of authorization image - opsional.
6.
> **Important**
>
> Periksa kembali semua bidang untuk memastikan bahwa mereka benar.

7. Pilih Kirim pendaftaran untuk mengirim ulang setelah Anda selesai.

Menghapus pendaftaran ID Pengirim Singapura

Jika Anda tidak lagi ingin melanjutkan pendaftaran ID Pengirim Singapura, Anda dapat menghapus pendaftaran. Pendaftaran hanya dapat dihapus jika statusnya Dibuat atau Membutuhkan Pembaruan.

Untuk menghapus pendaftaran

1. Buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS and voice (SMS and voice), pilih Nomor telepon (SMS dan suara), pilih Nomor telepon.
3. Pada tab ID Pengirim, pilih ID registrasi yang ingin Anda hapus, lalu pilih Hapus Pendaftaran.

Masalah Pendaftaran Singapura

Jika ID Pengirim Indonesia Anda tidak diterima oleh Amazon Pinpoint, Anda akan melihat pesan yang menjelaskan mengapa ID tersebut ditolak. Jika Anda memiliki pertanyaan tentang penolakan ini yang tidak terjawab dalam Praktik Terbaik kami, Anda dapat mengirimkan permintaan kepada tim dukungan kami.

Untuk mengirimkan permintaan informasi tentang ID Pengirim Singapura yang ditolak

1. Buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Pilih Support (Dukungan), lalu Support Center (Pusat Dukungan).
3. Di halaman Support (Dukungan), pilih Create case (Buat kasus).
4. Untuk Jenis kasus, pilih Peningkatan batas layanan.
5. Untuk jenis Limit, pilih Pinpoint SMS.

6. Di bagian Requests (Permintaan), lakukan hal berikut:
   - Untuk Jenis Sumber Daya, pilih Registasi ID Pengirim.
   - Untuk Limit, pilih Permintaan Penolakan Pendaftaran.
7. Untuk deskripsi kasus Penggunaan, masukkan ID Pengirim Singapura yang ditolak dan berikan alasan penolakan.
8. Di bawah Contact option (Opsi kontak), untuk Preferred contact language (Bahasa kontak pilihan), pilih bahasa yang ingin digunakan saat berkomunikasi dengan Tim Support AWS.
9. Untuk Contact method (Metode kontak), pilih metode yang Anda inginkan untuk berkomunikasi dengan Tim Support AWS.
10. Pilih Submit (Kirim).

TimAWS Support akan memberikan informasi tentang alasan pendaftaran ID Pengirim Anda ditolak dalam kasusAWS Support Anda.

Pertanyaan umum pendaftaran ID Pengirim Singapura

Pertanyaan yang sering diajukan tentang proses pendaftaran nomor ID Pengirim Singapura dengan Amazon Pinpoint.

Apakah saat ini saya memiliki ID Pengirim Singapura?

Untuk memeriksa apakah Anda memiliki ID Pengirim Singapura

1. Buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS and voice (SMS and voice), pilih Nomor telepon (SMS dan suara), pilih Nomor telepon.
3. Pada tab Registrasi SenderID, pilih ID Pengirim yang ingin Anda lihat dan pilih ID Pendaftaran.

Berapa lama waktu pendaftaran?

Sementara tinjauan umum membutuhkan waktu 1 - 3 minggu, mungkin diperlukan waktu hingga 5 minggu atau lebih lama dalam beberapa kasus untuk memverifikasi informasi Anda dengan lembaga pemerintah.

Apa yang dimaksud dengan Unique Entity Number (UEN) dan bagaimana cara mendapatkannya?

UEN adalah ID bisnis Singapura yang dikeluarkan oleh Accounting and Corporate Regulatory Agency (ACRA). Perusahaan dan bisnis lokal di Singapura bisa mendapatkan UEN dengan mendaftar

melalui ACRA. Setelah Anda melewati prosedur pendaftaran dan penggabungan standar, itu akan dikeluarkan. Anda dapat mengajukan permohonan untuk UEN dengan ACRA melalui Bizfile.

Apakah saya harus mendaftar untuk ID Pengirim Singapura?

Ya. Jika saat ini Anda belum mendaftarkan ID Pengirim Indonesia Anda pada tahun 2023—01—30, pesan apa pun yang dikirim menggunakan ID Pengirim akan diubah menjadi LIKELY-SCAM

Bagaimana cara mendaftarkan ID Pengirim Singapura saya dengan Amazon Pinpoint?

Ikuti petunjuk di Mendaftarkan ID Pengirim Singapura Anda dengan Amazon Pinpoint untuk mendaftarkan ID Pengirim.

Apa status pendaftaran ID Pengirim Singapura saya dan apa artinya?

Ikuti petunjuk di status pendaftaran ID Pengirim Singapura untuk memeriksa pendaftaran dan status Anda.

Informasi apa yang perlu saya berikan?

Anda perlu memberikan alamat perusahaan Anda, kontak bisnis, dan kasus penggunaan. Anda dapat menemukan informasi yang diperlukan di Mendaftarkan ID Pengirim Singapura Anda dengan Amazon Pinpoint.

Bagaimana jika pendaftaran ID Pengirim Singapura saya ditolak?

Jika pendaftaran Anda ditolak, statusnya akan diubah menjadi Memerlukan Pembaruan dan Anda dapat melakukan pembaruan dengan mengikuti petunjuk dalam Mengedit pendaftaran ID Pengirim Indonesia.

Izin apa yang saya butuhkan?

Pengguna/peran IAM yang Anda gunakan untuk mengunjungi konsol Amazon Pinpoint harus diaktifkan dengan izin *“sms-voice: *”*.

Nomor asal

Sebuah nomor asal adalah string numerik yang mengidentifikasi nomor telepon pengirim pesan SMS. Bila Anda mengirim pesan SMS menggunakan nomor asal, perangkat penerima akan menunjukkan nomor asal sebagai nomor telepon pengirim. Anda dapat menentukan nomor asal yang berbeda berdadsarkan kasus penggunaan.

> **Tip**
>
> Untuk melihat daftar semua nomor asal yang ada di akun AWS Anda, di panel navigasi konsol Amazon SNS, pilih Origination numbers (Nomor asal).

Dukungan untuk nomor asal tidak tersedia di negara-negara di mana hukum setempat mewajibkan penggunaan ID pengirim alih-alih nomor asal.

Topik

- 10DLC
- Nomor bebas pulsa
- Kode pendek
- Person-to-person(P2P) kode panjang
- Perbandingan nomor produk U.S.

10DLC

Operator AS tidak lagi mendukung penggunaan pesan SMS application-to-person (A2P) melalui kode panjang lokal yang tidak terdaftar. Untuk olahpesan SMS A2P volume tinggi, operator US sebaliknya menawarkan jenis baru kode panjang yang disebut kode panjang 10 digit (10-digit long codes/10DLC).

> **Important**
>
> Mulai 26 Januari 2023, vendor SMS Amazon SNS memperkenalkan proses peninjauan manual baru pada kampanye 10DLC untuk mengatasi masalah spam SMS yang diajukan oleh operator AS. Anda dapat menggunakan kode pendek dan nomor bebas pulsa sebagai alternatif 10DLC untuk mengirim SMS di Amerika Serikat.
> Saat ini vendor SMS kami belum memberikan target tingkat layanan tentang berapa lama ulasan kampanye 10DLC akan berlangsung. Ulasan dipicu setelah nomor dikaitkan dengan kampanye 10DLC. Ulasan memakan waktu lebih lama dari perkiraan waktu 14 hari Amazon SNS sebelumnya dikomunikasikan.
> Amazon SNS bekerja sama dengan vendor SMS setiap hari untuk memastikan bahwa:
>
> - Vendor menyelesaikan ulasan kampanye 10DLC yang tertunda sesegera mungkin

- Vendor memprioritaskan permintaan AWS di backlog mereka

Anda dapat memeriksa status kampanye 10DLC dengan mengikuti petunjuk di kampanye 10DLC. Jika informasi tambahan diperlukan untuk menyetujui kampanye 10DLC, tim AWS dukungan akan memberi tahu Anda.
Anda mungkin dapat mendaftarkan nomor bebas pulsa AS lebih cepat daripada mendapatkan nomor 10DLC. Untuk informasi lebih lanjut tentang nomor bebas pulsa AS dan proses pendaftaran, lihat. Persyaratan dan proses pendaftaran nomor bebas pulsa

Apa itu 10DLC?

10DLC adalah jenis kode panjang yang terdaftar pada operator untuk mendukung olahpesan SMS A2P volume tinggi menggunakan format nomor telepon 10 digit. Amazon SNS tidak lagi menawarkan kode panjang lokal sebagai produk SMS dan sebaliknya menawarkan 10DLC. 10DLC tidak mempengaruhi Anda jika Anda hanya menggunakan kode pendek dan nomor bebas pulsa.

10DLC adalah nomor telepon 10 digit yang hanya digunakan di Amerika Serikat. Pesan yang dikirim dari 10DLC ke penerima menampilkan nomor 10 digit sebagai pengirim. Tidak seperti nomor bebas pulsa, 10DLC mendukung olahpesan transaksional dan promosi, dan dapat menyertakan kode area US.

Jika Anda memiliki kode panjang lokal, Anda dapat meminta kode panjang lokal mereka diaktifkan untuk 10DLC. Untuk melakukannya, selesaikan proses pendaftaran 10DLC kemudian kirimkan tiket dukungan. Jika ada masalah dengan mengaktifkan kode panjang Anda untuk 10DLC, Anda diberi tahu dan diinstruksikan untuk meminta 10DLC baru melalui konsol Amazon Pinpoint (bukan Amazon SNS). Untuk informasi tentang cara mengajukan tiket dukungan untuk mengonversi kode panjang, lihat Mengaitkan kode panjang dengan kampanye 10DLC.

Untuk menggunakan nomor 10DLC, pertama-tama daftarkan perusahaan Anda dan buat kampanye 10DLC menggunakan konsol Amazon Pinpoint (bukan Amazon SNS). AWSmembagikan informasi ini dengan The Campaign Registry, pihak ketiga yang menyetujui atau menolak pendaftaran Anda berdasarkan informasi. Dalam beberapa kasus, pendaftaran segera terjadi. Misalnya, jika Anda sebelumnya telah terdaftar pada The Campaign Registry, mereka mungkin telah memiliki informasi Anda. Namun, beberapa kampanye mungkin memerlukan waktu satu minggu atau lebih untuk mendapat persetujuan. Setelah perusahaan dan kampanye 10DLC Anda disetujui, Anda dapat membeli nomor 10DLC dan mengaitkannya dengan kampanye Anda. Meminta 10DLC mungkin juga membutuhkan waktu hingga satu minggu untuk mendapat persetujuan. Meskipun Anda dapat

mengaitkan beberapa 10DLC dengan satu kampanye, Anda tidak dapat menggunakan 10DLC yang sama di beberapa kampanye. Untuk setiap kampanye yang Anda buat, Anda harus memiliki satu 10DLC yang unik.

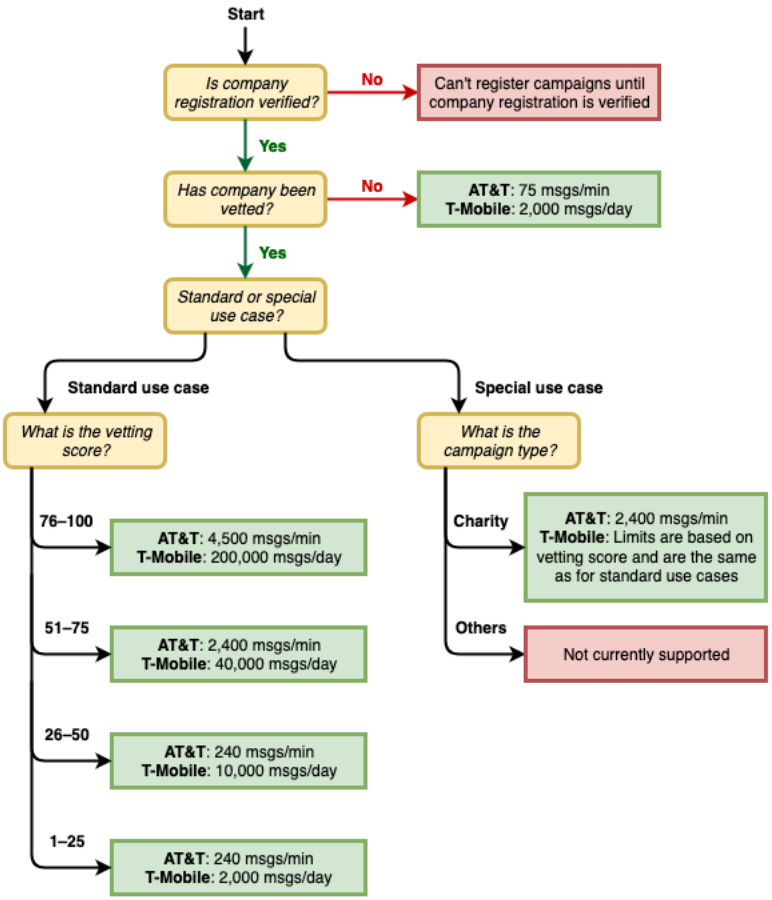
Kemampuan 10DLC

Kemampuan nomor telepon 10DLC bergantung pada operator seluler mana yang digunakan penerima Anda. AT&T memberikan batasan jumlah bagian pesan yang dapat dikirim setiap menit untuk setiap kampanye. T-Mobile memberikan batas harian pesan yang dapat dikirim untuk setiap perusahaan, tanpa batasan jumlah bagian pesan yang dapat dikirim per menit. Verizon belum menerbitkan batas throughput, tetapi menggunakan sistem penyaringan untuk 10DLC yang dirancang untuk menghapus spam, pesan yang tidak diminta, dan konten yang kasar, dengan sedikit penekanan pada throughput pesan yang sebenarnya.

Kampanye 10DLC baru yang terkait dengan perusahaan yang tidak diperiksa dapat mengirim 75 bagian pesan per menit kepada penerima yang menggunakan AT&T, dan 2.000 pesan per hari kepada penerima yang menggunakan T-Mobile. Batas perusahaan dibagikan di semua kampanye 10DLC Anda. Misalnya, jika Anda telah mendaftarkan satu perusahaan dan dua kampanye, penjatahan harian 2.000 pesan kepada pelanggan T-Mobile dibagikan di seluruh kampanye tersebut. Demikian pula, jika Anda mendaftarkan perusahaan yang sama di lebih dari satu AWS akun, penjatahan harian dibagikan di seluruh akun tersebut.

Jika kebutuhan throughput Anda melebihi batas ini, Anda dapat meminta agar pendaftaran perusahaan Anda diperiksa. Ketika Anda memeriksa pendaftaran perusahaan Anda, penyedia verifikasi pihak ketiga menganalisis detail perusahaan Anda. Penyedia verifikasi kemudian memberikan skor pemeriksaan, yang menentukan kemampuan kampanye 10DLC Anda. Ada biaya satu kali untuk layanan pemeriksaan. Untuk informasi selengkapnya, lihat Memeriksa pendaftaran Amazon SNS 10DLC Anda.

Tingkat throughput aktual Anda akan bervariasi tergantung pada berbagai faktor, seperti apakah perusahaan Anda telah diperiksa atau tidak, jenis kampanye Anda, dan skor pemeriksaan Anda. Diagram alir berikut menunjukkan tingkat throughput untuk berbagai situasi.



Tingkat throughput untuk 10DLC ditentukan oleh operator seluler AS bekerja sama dengan Registry Kampanye. Baik Amazon SNS maupun layanan pengiriman SMS lainnya tidak dapat meningkatkan throughput 10DLC melebihi tarif ini. Jika Anda membutuhkan tingkat throughput yang tinggi dan tingkat pengiriman yang tinggi di semua operator AS, kami sarankan Anda menggunakan kode pendek. Untuk informasi lebih lanjut tentang mendapatkan kode pendek, lihatMeminta kode pendek khusus untuk olahpesan SMS dengan Amazon SNS.

Memulai dengan 10DLC

Gunakan konsol Amazon Pinpoint (bukan Amazon SNS) untuk meminta 10DLC Anda. Ikuti langkah-langkah ini untuk menyiapkan 10DLC untuk digunakan dengan kampanye 10DLC Anda.

1. Daftarkan perusahaan Anda.

   Sebelum Anda dapat meminta 10DLC, perusahaan Anda harus terdaftar pada The Campaign Registry; untuk informasinya, lihat Mendaftarkan perusahaan. Pendaftaran biasanya instan kecuali The Campaign Registry memerlukan informasi lebih lanjut. Ada biaya pendaftaran satu kali untuk

mendaftarkan perusahaan Anda, yang ditampilkan di halaman pendaftaran. Biaya satu kali ini dibayarkan secara terpisah dari biaya bulanan Anda untuk kampanye dan 10DLC.

> **Note**
>
> Olahpesan SMS Amazon SNS tersedia di Wilayah di mana Amazon Pinpoint saat ini tidak didukung. Ada dua kasus yang berbeda:
>
> a. Jika Anda menggunakan akun cloud komersial, Anda perlu membuka konsol Amazon Pinpoint di Wilayah AS Timur (Virginia Utara) untuk mendaftarkan perusahaan dan kampanye 10DLC Anda. Jangan meminta nomor 10DLC.
>
> b. Gunakan konsol Kuota AWS Layanan untuk membuat kasus peningkatan batas layanan saat meminta nomor 10DLC untuk Wilayah tersebut. Untuk informasi tentang Wilayah tempat Amazon Pinpoint tersedia, lihat titik akhir dan kuota Amazon Pinpoint di bagian. Referensi Umum AWS
>
> c. Jika Anda menggunakan AWS GovCloud (US)akun, buka konsol Amazon Pinpoint di Wilayah Barat AS untuk mendaftarkan perusahaan dan kampanye 10DLC Anda. Jangan meminta nomor 10DLC. Sebagai gantinya, gunakan konsol Kuota AWS Layanan untuk membuat kasus peningkatan batas layanan saat meminta nomor 10DLC untuk Wilayah tersebut. Untuk informasi tentang Wilayah tempat Amazon Pinpoint tersedia, lihat titik akhir dan kuota Amazon Pinpoint di bagian. Referensi Umum AWS

2. (Opsional, tetapi disarankan) Terapkan untuk pemeriksaan

   Jika pendaftaran perusahaan Anda berhasil, Anda dapat mulai membuat kampanye 10DLC dengan volume rendah dan campuran. Kampanye ini dapat mengirim 75 pesan per menit kepada penerima yang menggunakan AT&T, dan perusahaan terdaftar Anda dapat mengirim 2.000 pesan per hari kepada penerima yang menggunakan T-Mobile. Jika kasus penggunaan Anda memerlukan tingkat throughput yang melebihi nilai ini, Anda dapat mengajukan permohonan pemeriksaan pendaftaran perusahaan Anda. Memeriksa pendaftaran perusahaan Anda dapat meningkatkan tingkat throughput untuk perusahaan dan kampanye Anda, tetapi tidak dijamin untuk melakukannya. Untuk informasi lebih lanjut tentang pemeriksaan, lihat. Memeriksa pendaftaran Amazon SNS 10DLC Anda

3. Daftarkan kampanye Anda.

   Setelah perusahaan Anda terdaftar, buat kampanye 10DLC dan kaitkan dengan salah satu perusahaan terdaftar Anda. Kampanye ini dikirimkan ke The Campaign Registry untuk persetujuan. Dalam kebanyakan kasus, persetujuan kampanye 10DLC segera didapatkan kecuali

The Campaign Registry memerlukan informasi lebih lanjut. Untuk informasi selengkapnya, lihat Mendaftarkan kampanye 10DLC.

4. Minta nomor 10DLC Anda.

   Setelah kampanye 10DLC disetujui, Anda dapat meminta 10DLC dan mengaitkan nomor tersebut dengan kampanye yang disetujui. Kampanye 10DLC Anda hanya dapat menggunakan nomor yang disetujui untuknya. Lihat Meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P untuk olahpesan SMS dengan Amazon SNS.

Biaya pendaftaran 10DLC dan biaya bulanan

Ada biaya pendaftaran dan biaya bulanan yang terkait dengan penggunaan 10DLC, seperti mendaftarkan perusahaan dan kampanye 10DLC Anda. Biaya-biaya ini terpisah dari biaya bulanan lainnya yang dibebankan oleh AWS. Untuk informasi lebih lanjut, lihat halaman Harga SMS Amazon SNS Seluruh Dunia.

Mendaftarkan perusahaan

Sebelum Anda dapat meminta 10DLC, perusahaan Anda harus terdaftar pada The Campaign Registry.

> **Note**
>
> Olahpesan SMS Amazon SNS tersedia di Wilayah di mana Amazon Pinpoint saat ini tidak didukung. Dalam kasus ini, buka konsol Amazon Pinpoint di Wilayah US East (N. Virginia) untuk mendaftarkan perusahaan dan kampanye 10DLC Anda, tetapi jangan meminta nomor 10DLC. Sebagai gantinya, gunakan konsolAWS Service Quotas untuk membuat kasus peningkatan batas layanan saat meminta nomor 10DLC untuk Wilayah tersebut. Untuk informasi tentang Wilayah di mana Amazon Pinpoint tersedia, lihat titik akhir dan kuota Amazon Pinpoint dalam Referensi Umum AWS.

Status pendaftaran perusahaan 10DLC

Saat Anda mendaftarkan perusahaan atau merek Anda, salah satu dari dua status dikembalikan: baik Belum Diverifikasi atau Terverifikasi. Jika status pendaftaran perusahaan Anda tidak diverifikasi, itu berarti ada masalah dengan pendaftaran Anda. Misalnya, Nama Perusahaan Terdaftar yang Anda berikan mungkin tidak sama persis dengan nama terdaftar perusahaan yang terkait dengan ID

Pajak yang Anda berikan. Jika Anda menemukan masalah dengan detail pendaftaran perusahaan Anda, Anda dapat memperbaikinya. Untuk informasi selengkapnya tentang cara memodifikasi detail registrasi perusahaan, lihatMengedit atau menghapus perusahaan terdaftar.

Jika status pendaftaran perusahaan Anda Terverifikasi, maka detail pendaftaran yang Anda berikan akurat, dan Anda dapat mulai membuat kampanye 10DLC.

Mendaftarkan perusahaan atau merek Anda

Anda hanya perlu mendaftarkan perusahaan Anda satu kali. Setelah terdaftar, Anda dapat mengedit informasi perusahaan dan kontak Anda. Untuk menghapus perusahaan terdaftar, buat kasus dengan AWSSupport. Untuk informasi selengkapnya tentang mengedit atau menghapus detail perusahaan, lihatMengedit atau menghapus perusahaan terdaftar.

Cara mendaftarkan perusahaan

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS dan suara, pilih Nomor telepon.
3. Pada tab kampanye 10DLC, pilih Daftar perusahaan.

   > **Note**
   >
   > Halaman Daftarkan perusahaan Anda menampilkan biaya Pendaftaran. Ini adalah biaya satu kali yang terkait dengan pendaftaran perusahaan Anda. Biaya ini terpisah dari biaya atau biaya bulanan lainnya. Ini dibebankan kepada Anda ketika Anda mendaftarkan perusahaan Anda, atau ketika Anda mengubah rincian pendaftaran perusahaan yang ada.

4. Di bagian Info Perusahaan, lakukan hal berikut:

   - Untuk nama perusahaan Legal, masukkan nama perusahaan yang terdaftar di bawah. Nama yang Anda masukkan harus sama persis dengan nama perusahaan yang terkait dengan ID pajak yang Anda berikan.

> **Important**
>
> Pastikan untuk menggunakan nama resmi perusahaan Anda. Setelah dikirimkan, Anda tidak dapat mengubah informasi ini. Informasi yang salah atau tidak lengkap dapat mengakibatkan pendaftaran Anda tertunda atau ditolak.

- Untuk Jenis formulir hukum apa organisasi ini, pilih opsi yang paling menggambarkan perusahaan Anda.

> **Note**
>
> Pemerintah AS danot-for-profit opsi N hanya dapat digunakan untuk mendaftarkan organisasi yang berbasis di Amerika Serikat. Jika organisasi Anda berbasis di negara selain AS, Anda harus mendaftar sebagai Private for-profit, terlepas dari bentuk hukum sebenarnya dari organisasi Anda.

- Jika Anda memilih Public untuk mendapatkan keuntungan pada langkah sebelumnya, masukkan simbol saham perusahaan dan bursa saham tempat ia terdaftar.
- Untuk Negara pendaftaran, pilih negara tempat perusahaan terdaftar.
- Untuk Doing Business As (DBA) atau nama merek, masukkan nama lain yang perusahaan Anda lakukan sebagai bisnis.
- Untuk ID Pajak, masukkan ID pajak perusahaan Anda. ID yang Anda masukkan tergantung pada negara tempat perusahaan Anda terdaftar.
  - Jika Anda mendaftarkan entitas AS atau non-AS yang memiliki IRS Employer Identification Number (EIN), masukkan sembilan digit EIN Anda. Nama perusahaan legal, EIN, dan alamat fisik yang Anda masukkan harus semuanya sesuai dengan informasi perusahaan yang terdaftar di IRS.
  - Jika Anda mendaftarkan entitas Kanada, masukkan nomor Perusahaan federal atau provinsi Anda. Jangan masukkan Nomor Bisnis (BN) yang disediakan oleh CRA. Nama perusahaan legal, nomor Korporasi, dan alamat fisik yang Anda masukkan semuanya harus sesuai dengan informasi perusahaan yang terdaftar di Corporations Canada.

- Jika Anda mendaftarkan entitas yang berbasis di negara lain, masukkan ID pajak utama untuk negara Anda. Di banyak negara, ini adalah bagian numerik dari nomor ID PPN Anda.

- Untuk Vertical, pilih kategori yang paling menggambarkan perusahaan yang Anda daftarkan.

5. Di bagian Info kontak, lakukan hal berikut:

   - Untuk Alamat/Jalan, masukkan alamat jalan fisik yang terkait dengan perusahaan Anda.
   - Untuk City, masukkan kota tempat alamat fisik berada.
   - Untuk Negara atau wilayah, masukkan negara bagian atau wilayah tempat alamat berada.
   - Untuk Kode Pos/Kode Pos, masukkan kode pos atau kode pos untuk alamat tersebut.
   - Untuk situs web Perusahaan, masukkan URL lengkap situs web perusahaan Anda. Sertakan “http://” atau “https://” di awal alamat.
   - Untuk Email Support, masukkan alamat email.
   - Untuk Nomor telepon Support, masukkan nomor telepon dengan kode negara.

   > **Note**
   >
   > Registri Kampanye memerlukan alamat email kontak dan nomor telepon jika mereka perlu memverifikasi informasi pendaftaran dengan perwakilan perusahaan Anda.

6. Setelah selesai, pilih Buat. Pendaftaran perusahaan Anda dikirimkan ke Registri Kampanye. Dalam kebanyakan kasus, pendaftaran Anda diterima segera, dan status disediakan.

Jika status pendaftaran perusahaan Anda Terverifikasi, maka Anda dapat mulai membuat kampanye 10DLC dengan volume rendah dan campuran. Anda dapat menggunakan jenis kampanye ini untuk mengirim hingga 75 pesan per menit kepada penerima yang menggunakan AT&T, dan perusahaan terdaftar Anda dapat mengirim 2.000 pesan per hari kepada penerima yang menggunakan T-Mobile. Anda juga dapat mengirim pesan ke penerima yang menggunakan operator AS lainnya, seperti Verizon dan US Cellular. Operator ini tidak secara ketat menegakkan batas throughput, tetapi mereka sangat memantau pesan 10DLC untuk tanda-tanda spam dan penyalahgunaan.

Jika kasus penggunaan Anda memerlukan tingkat throughput yang melebihi nilai ini, Anda dapat mengajukan pemeriksaan tambahan untuk pendaftaran perusahaan Anda. Untuk informasi lebih lanjut tentang memeriksa pendaftaran merek Anda, lihat Memeriksa pendaftaran Amazon SNS 10DLC Anda.

Jika status pendaftaran perusahaan Anda Tidak Terverifikasi, ada masalah dengan informasi yang Anda berikan. Periksa informasi yang Anda berikan dan konfirmasikan bahwa semua bidang berisi informasi yang benar. Anda dapat membuat perubahan pada beberapa bagian pendaftaran perusahaan Anda di konsol Amazon Pinpoint. Untuk informasi selengkapnya tentang cara memodifikasi detail registrasi perusahaan, lihatMengedit pendaftaran perusahaan 10DLC.

Memeriksa pendaftaran Amazon SNS 10DLC Anda

Jika pendaftaran perusahaan Anda berhasil dan Anda ingin mendaftarkan kampanye dengan kemampuan throughput yang lebih tinggi, maka Anda harus memeriksa pendaftaran perusahaan Anda.

Saat Anda memeriksa pendaftaran Anda, organisasi pihak ketiga menganalisis detail perusahaan yang Anda berikan dan mengembalikan skor pemeriksaan. Skor pemeriksaan yang tinggi dapat menghasilkan tingkat throughput yang lebih tinggi untuk perusahaan 10DLC Anda dan kampanye yang terkait dengannya. Namun, pemeriksaan tidak dijamin untuk meningkatkan throughput Anda.

Skor pemeriksaan tidak diterapkan secara surut. Dengan kata lain, jika Anda telah membuat kampanye 10DLC, dan Anda kemudian memeriksa pendaftaran perusahaan Anda, skor pemeriksaan Anda tidak secara otomatis diterapkan ke kampanye yang ada. Untuk alasan ini, Anda harus memeriksa perusahaan atau merek Anda sebelum membuat kampanye 10DLC Anda.

> **Note**
>
> Ada biaya $40 yang tidak dapat dikembalikan untuk memeriksa perusahaan atau merek Anda.

Untuk memeriksa pendaftaran perusahaan Anda

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawah SMS dan suara, pilih Nomor telepon.
3. Pada tab kampanye 10DLC, pilih perusahaan 10DLC yang ingin Anda dokter hewan.
4. Di halaman detail perusahaan, di bagian bawah halaman, pilih Terapkan untuk pemeriksaan.
5. Pada jendela Apply for additional vetting, pilih Submit.

   Untuk perusahaan yang berbasis di AS, proses pemeriksaan biasanya membutuhkan waktu sekitar satu menit untuk diselesaikan. Untuk perusahaan yang tidak berbasis di AS, proses

pemeriksaan mungkin membutuhkan lebih banyak waktu, tergantung pada seberapa mudah data yang tersedia untuk negara tersebut.

Setelah Anda mengirimkan permintaan pemeriksaan, Anda kembali ke halaman detail perusahaan. Bagian hasil pemeriksaan Perusahaan menampilkan status dan hasil permintaan pemeriksaan Anda. Saat proses pemeriksaan selesai, tabel ini menunjukkan skor pemeriksaan di kolom Skor. Skor pemeriksaan Anda menentukan kemampuan throughput 10DLC Anda. Throughput Anda bervariasi berdasarkan jenis kampanye yang Anda buat. Jika Anda membuat kampanye 10DLC terkait penggunaan campuran atau pemasaran, Anda harus memiliki skor pemeriksaan yang lebih tinggi daripada yang Anda perlukan untuk jenis kampanye lain untuk mencapai tingkat throughput yang tinggi. Untuk informasi selengkapnya tentang kemampuan nomor telepon 10DLC, lihatKemampuan 10DLC.

Jika Anda mengubah detail pendaftaran perusahaan Anda setelah menyelesaikan proses pemeriksaan, Anda dapat meminta agar pendaftaran Anda diperiksa kembali. Jika Anda hanya mengubah Vertikal untuk pendaftaran perusahaan Anda, skor pemeriksaan Anda tidak akan berubah. Jika Anda mengubah detail apa pun selain Vertikal, hasil pemeriksaan Anda bisa berubah. Dalam kedua kasus tersebut, Anda akan dikenakan biaya pemeriksaan satu kali lagi.

Mengedit atau menghapus perusahaan terdaftar

Anda dapat mengedit beberapa informasi pendaftaran 10DLC untuk perusahaan Anda langsung di konsol Amazon Pinpoint. Anda juga dapat menghapus pendaftaran perusahaan 10DLC dengan membuat kasing di Support AWS Center.

Mengedit pendaftaran perusahaan 10DLC

Setelah Anda menyelesaikan proses pendaftaran 10DLC untuk sebuah perusahaan, Anda dapat mengedit detail pendaftaran Anda.

Jika Anda melihat pesan kesalahan setelah mengedit detail pendaftaran perusahaan Anda, mungkin ada masalah lain dengan pendaftaran Anda. Anda dapat membuka tiket dengan AWS Support untuk meminta informasi lebih lanjut.

Untuk mengedit pendaftaran perusahaan

1. Buka AWS SMS konsol di https://console.aws.amazon.com/sms-voice/.
2. Ikuti petunjuk untuk Edit pendaftaran Anda di Panduan Pengguna SMS Amazon Pinpoint.

Menghapus pendaftaran perusahaan 10DLC

Untuk menghapus pendaftaran perusahaan

1. Buka AWS SMS konsol di https://console.aws.amazon.com/sms-voice/.
2. Ikuti petunjuk untuk Hapus pendaftaran Anda di Panduan Pengguna SMS Amazon Pinpoint.

Mendaftarkan kampanye 10DLC

Saat Anda mendaftarkan kampanye 10DLC, Anda memberikan deskripsi kasus penggunaan Anda, serta templat pesan yang akan Anda gunakan. Sebelum Anda dapat membuat dan mendaftarkan kampanye 10DLC, Anda harus mendaftarkan perusahaan Anda terlebih dahulu. Untuk informasi tentang mendaftarkan perusahaan, lihatMendaftarkan perusahaan.

> **Note**
>
> Setelah Anda mendaftarkan perusahaan Anda, Amazon Pinpoint menunjukkan salah satu dari dua status untuk pendaftaran: Verified or Unverified. Anda hanya dapat menyelesaikan proses pendaftaran kampanye 10DLC jika status pendaftaran perusahaan Anda Diverifikasi. Anda akan dapat membuat kampanye penggunaan campuran volume rendah.
> Jika statusnya Belum Diverifikasi, biasanya berarti beberapa data yang Anda berikan saat mendaftarkan perusahaan Anda salah. Anda tidak akan dapat membuat kampanye 10DLC apa pun saat perusahaan Anda memiliki status ini. Anda dapat memodifikasi pendaftaran perusahaan Anda untuk mencoba memperbaiki masalah dengan pendaftaran perusahaan Anda. Untuk informasi lebih lanjut tentang memodifikasi pendaftaran perusahaan 10DLC, lihatMengedit atau menghapus perusahaan terdaftar.

Di halaman ini, Anda terlebih dahulu memberikan detail tentang perusahaan tempat Anda membuat kampanye 10DLC dan kemudian memberikan detail kasus penggunaan kampanye itu sendiri. Informasi di halaman ini kemudian diberikan kepada The Campaign Registry untuk persetujuan.

Di bagian ini, Anda akan memilih perusahaan tempat Anda membuat kampanye 10DLC dan memberikan detail tambahan.

Cara mendaftarkan kampanye 10DLC

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.

2. Di bawah SMS dan suara, pilih Nomor telepon.
3. Di tab Kampanye 10DLC, pilih Buat kampanye 10DLC.
4. Pada halaman Buat kampanye 10DLC, di bagian Info Kampanye, lakukan hal berikut:

   a. Untuk Company name (Nama perusahaan), pilih perusahaan tempat Anda membuat kampanye ini. Jika Anda belum mendaftarkan perusahaan, Anda harus melakukannya terlebih dahulu. Untuk informasi lebih lanjut tentang mendaftarkan perusahaan, lihatMendaftarkan perusahaan.

   b. Untuk nama kampanye 10DLC, masukkan nama untuk kampanye.

   c. Untuk Vertikal, pilih opsi yang paling mewakili perusahaan Anda.

   d. Untuk pesan Bantuan, masukkan pesan yang diterima pelanggan Anda jika mereka mengirim kata kunci “BANTUAN” ke nomor telepon 10DLC Anda.

   e. Untuk pesan Stop, masukkan pesan yang diterima pelanggan Anda jika mereka mengirim kata kunci “STOP” ke nomor telepon 10DLC Anda.

   > **Tip**
   >
   > Pelanggan Anda dapat membalas pesan Anda dengan kata “BANTUAN” untuk mempelajari lebih lanjut tentang pesan yang mereka terima dari Anda. Mereka juga dapat membalas “STOP” untuk memilih untuk tidak menerima pesan dari Anda. Operator seluler AS mengharuskan Anda memberikan tanggapan terhadap kedua kata kunci ini.
   > Berikut ini adalah contoh respons BANTUAN yang sesuai dengan persyaratan operator seluler AS:
   > **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
   > Berikut ini adalah contoh respons STOP yang sesuai:
   > **`You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.`**
   > Tanggapan Anda terhadap kata kunci ini harus berisi 160 karakter atau lebih sedikit.

5. Di bagian kasus penggunaan kampanye, lakukan hal berikut:

   a. Untuk Gunakan jenis kasus, jika Anda memiliki kasus penggunaan terkait amal, pilih Khusus. Jika tidak, pilih Standar.

b. Untuk Use case (Kasus penggunaan), pilih kasus penggunaan yang paling mirip dengan kampanye Anda dari daftar kasus penggunaan yang telah ditetapkan sebelumnya. Biaya bulanan untuk setiap kasus penggunaan muncul di samping nama kasus penggunaan.

> **Note**
>
> Biaya bulanan untuk mendaftarkan kampanye 10DLC ditampilkan di sebelah setiap jenis kasus penggunaan. Sebagian besar jenis kampanye 10DLC memiliki biaya bulanan yang sama. Biaya untuk mendaftarkan kasus penggunaan Campuran Volume Rendah lebih rendah daripada jenis kasus penggunaan lainnya. Namun, kampanye Campuran Volume Rendah mendukung tingkat throughput yang lebih rendah daripada jenis kampanye lainnya.

c. Masukkan setidaknya satu Contoh pesan SMS. Ini adalah contoh pesan yang akan Anda kirim ke pelanggan Anda. Jika Anda berencana untuk menggunakan beberapa template pesan untuk kampanye 10DLC ini, sertakan juga.

> **Important**
>
> Jangan gunakan teks placeholder untuk pesan sampel Anda. Contoh pesan yang Anda berikan harus mencerminkan pesan aktual yang ingin Anda kirim seakurat mungkin.

6. Bagian Kampanye dan atribut konten berisi serangkaian pertanyaan Ya atau Tidak terkait dengan fitur tertentu dari kampanye. Beberapa atribut bersifat wajib, sehingga Anda tidak dapat mengubah nilai defaultnya.

   Pastikan atribut yang Anda pilih akurat untuk kampanye Anda.

   Tunjukkan apakah masing-masing hal berikut berlaku untuk kampanye yang Anda daftarkan:

   - Subscriber opt-in (Keikutsertaan pelanggan) – Pelanggan dapat memilih untuk menerima pesan tentang kampanye ini.
   - Subscriber opt-out (Ketidak ikutsertaan pelanggan) – Pelanggan dapat memilih untuk tidak menerima pesan tentang kampanye ini.
   - Subscriber help (Bantuan pelanggan) – Pelanggan dapat menghubungi pengirim pesan setelah mengirim kata kunci HELP.

- Number pooling (Kumpulan nomor) – Kampanye 10DLC ini menggunakan lebih dari 50 nomor telepon.
- Direct lending or loan arrangement (Pinjaman langsung atau pengaturan pinjaman) – Kampanye ini mencakup informasi tentang pinjaman langsung atau pengaturan pinjaman lainnya.
- Embedded link (Tautan tersemat) – Kampanye 10DLC menyertakan tautan tersemat. Tautan dari Penyingkat URL umum, seperti TinyUrl atau Bit.ly, tidak diperbolehkan. Namun, Anda dapat menggunakan Penyingkat URL yang menawarkan domain kustom.
- Nomor telepon yang disematkan — Kampanye ini menyertakan nomor telepon yang disematkan yang bukan nomor dukungan pelanggan.
- Afiliate marketing (Pemasaran afiliasi) – Kampanye 10DLC mencakup informasi dari pemasaran afiliasi.
- Age-gated content (Konten dengan batasan usia) – Kampanye 10DLC mencakup konten dengan batasan usia seperti yang ditentukan oleh pedoman operator dan Cellular Telecommunications and Internet (CTIA).

7. Pilih Create (Buat).

   Setelah Anda mengirimkan detail pendaftaran untuk kampanye Anda, halaman SMS dan suara akan terbuka. Sebuah pesan akan muncul yang menunjukkan bahwa kampanye Anda telah dikirimkan dan sedang ditinjau. Anda dapat melihat status permintaan Anda di tab 10DLC campaigns (Kampanye 10DLC). Anda dapat memeriksa status pendaftaran Anda di tab 10DLC, yang berupa salah satu dari status berikut:

   - Active (Aktif) – Kampanye 10DLC Anda telah disetujui. Anda dapat meminta nomor telepon 10DLC dan mengaitkan nomor tersebut dengan kampanye. Untuk informasi selengkapnya, lihat Meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P untuk olahpesan SMS dengan Amazon SNS.
   - Tertunda - Kampanye 10DLC Anda belum disetujui. Dalam beberapa kasus, persetujuan mungkin memakan waktu satu minggu atau lebih. Jika statusnya berubah, konsol Amazon Pinpoint akan memperlihatkan perubahan tersebut. Kami tidak memberi tahu Anda tentang perubahan status.
   - Ditolak - Kampanye 10DLC Anda ditolak. Untuk mendapatkan informasi selengkapnya, kirimkan permintaan dukungan yang menyertakan ID kampanye dari kampanye yang ditolak.
   - Suspended (ditangguhkan) – Satu atau beberapa operator menangguhkan kampanye 10DLC Anda. Untuk mendapatkan informasi lebih lanjut, kirimkan permintaan dukungan yang

menyertakan ID kampanye yang ditangguhkan. Amazon Pinpoint tidak menyertakan alasan penangguhan di konsol tersebut, dan kami tidak memberi tahu Anda jika kampanye Anda ditangguhkan.

8. Jika 10DLC Anda disetujui, Anda dapat meminta nomor 10DLC untuk dikaitkan dengan kampanye tersebut. Untuk informasi tentang meminta nomor 10DLC, lihat Meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P untuk olahpesan SMS dengan Amazon SNS.

Menggunakan kampanye 10DLC di beberapaAWS Wilayah

Saat Anda mendaftarkan perusahaan, perusahaan tersebut tersedia diAWS akun Anda di semuaAWS Wilayah. Namun, hal yang sama tidak berlaku untuk kampanye 10DLC. Kampanye 10DLC hanya dapat digunakan diAWS Wilayah tempat ia terdaftar.

Jika Anda berencana untuk menggunakan 10DLC di lebih dari satuAWS Wilayah, Anda harus mendaftarkan kampanye 10DLC terpisah di masing-masing Wilayah tersebut. Langkah ini diperlukan untuk memenuhi persyaratan operator. Anda dikenakan biaya untuk setiap kampanye yang Anda daftarkan, meskipun kasus penggunaannya persis sama.

Mendaftarkan beberapa kampanye memiliki manfaat tambahan yaitu meningkatkan tingkat throughput Anda untuk pesan yang Anda kirim ke penerima yang menggunakan AT&T sebagai operator seluler mereka, karena AT&T memberikan tingkat throughput 10DLC untuk setiap kampanye. Bandingkan ini dengan cara T-Mobile menangani throughput 10DLC, yang didasarkan pada alokasi pesan harian untuk setiap perusahaan (terlepas dari jumlah kampanye).

Mengedit atau menghapus kampanye 10DLC

Anda dapat mengedit respons HELP, respons STOP, dan pesan sampel untuk kampanye 10DLC menggunakan konsol Amazon Pinpoint. Anda juga dapat menghapus kampanye 10DLC menggunakan konsol.

Mengedit kampanye 10DLC

Setelah kampanye disetujui, Anda dapat memodifikasi pesan HELP, STOP, dan sampel. Anda juga dapat menambahkan pesan sampel tambahan. Perubahan pada bidang ini tidak memerlukan persetujuan ulang dari Registry Kampanye atau dari operator. Anda tidak dapat mengubah bidang lain setelah kampanye 10DLC disetujui.

Anda dapat memiliki maksimal lima pesan sampel. Anda tidak dapat mengurangi jumlah pesan sampel yang awalnya Anda daftarkan. Misalnya, jika Anda mendaftarkan kampanye dengan tiga

contoh pesan SMS, Anda tidak dapat mengurangi jumlah pesan SMS sampel menjadi kurang dari tiga.

> **Note**
>
> Jika Anda ingin mengubah bidang apa pun selain pesan HELP, STOP, dan sampel, Anda harus terlebih dahulu menghapus kampanye 10DLC lalu membuat ulang kampanye untuk menyertakan informasi yang diperbarui.

Cara mengedit kampanye 10DLC

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawahSMS dan suara, pilihnomor telepon.
3. PadaKampanye 10DLCTab, pilih kampanye 10DLC yang ingin Anda edit.
4. DiPesan kampanyebagian dari halaman detail kampanye, pilihMengedit.
5. Perbarui bidang berikut:

   - Pesan bantuan
   - Pesan stop
   - Pesan SMS

   Anda tidak dapat menghapus pesan sampel yang ditambahkan sebelumnya, atau menghapus isi pesan sampel sehingga bidang kosong. Jika Anda menghapus isi pesan tanpa mengganti konten tersebut, pesan asli akan digunakan saat memperbarui.
6. Pilih Update (Perbarui). Spanduk konfirmasi muncul yang memberi tahu Anda bahwa pesan kampanye telah diperbarui.

Menghapus kampanye 10DLC

Anda dapat menghapus kampanye 10DLC menggunakan konsol Amazon Pinpoint. Sebelum Anda menghapus kampanye 10DLC, Anda harus menghapus semua nomor telepon yang terkait dengan kampanye tersebut terlebih dahulu.

> **Important**
>
> Saat menghapus nomor 10DLC dari kampanye, Anda tidak lagi memiliki akses ke nomor tersebut. Selain itu, kampanye 10DLC yang dihapus tidak dapat dipulihkan.

Cara menghapus kampanye 10DLC

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di panel navigasi, di bawahSMS dan suara, pilihnomor telepon.
3. PadaKampanye 10DLCTab, pilih kampanye 10DLC yang ingin Anda edit.
4. Dinomor teleponbagian, perhatikan nomor telepon yang terkait dengan kampanye.
5. Padanomor teleponTab, pilih nomor 10DLC yang ingin Anda hapus, lalu pilihMenghapus nomor telepon.

   > **Note**
   >
   > Langkah ini hanya diperlukan jika Anda memiliki beberapa nomor telepon 10DLC yang terkait dengan kampanye. Jika Anda hanya memiliki satu nomor telepon yang terkait dengan kampanye 10DLC yang nomor tersebut akan muncul diKampanye 10DLCTab. Perhatikan nomor yang ditampilkan pada tab.

6. ENTER**`delete`**ke dalam kotak konfirmasi, lalu pilihKonfirmasikan. Sebuah pesan sukses muncul di bagian atas halaman SMS dan suara.
7. Ulangi dua langkah sebelumnya untuk setiap nomor 10DLC yang terkait dengan kampanye.
8. Setelah menghapus nomor yang terkait dengan kampanye 10DLC, pilihKampanye 10DLCTab.
9. Pilih kampanye 10DLC yang ingin Anda hapus.
10. Di pojok kanan atasRincian kampanye 10DLChalaman, pilihHapus.
11. ENTER**`delete`**ke dalam kotak konfirmasi, lalu pilihKonfirmasikan. Sebuah pesan sukses muncul di bagian atas halaman SMS dan suara.

Mengaitkan kode panjang dengan kampanye 10DLC

Jika Anda memiliki kode panjang, Anda dapat mengaitkan kode panjang tersebut dengan salah satu kampanye 10DLC Anda saat ini dengan mengajukan permintaan dukungan. Kode panjang yang

Anda kaitkan dengan kampanye 10DLC hanya dapat digunakan dengan kampanye tersebut dan tidak dapat digunakan untuk kampanye 10DLC lainnya. Sementara kode panjang Anda sedang dimigrasikan ke 10DLC, Anda masih dapat menggunakannya. Namun, Anda tidak akan dapat menggunakannya untuk kampanye 10DLC apa pun hingga kode tersebut disetujui.

Saat mengajukan permintaan, Anda memerlukan:

- Kode panjang untuk dikaitkan dengan kampanye 10DLC
- ID kampanye 10DLC untuk dikaitkan dengan kode panjang

> **Note**
>
> Sebelum Anda dapat mengaitkan kode panjang dengan kampanye, Anda harus memiliki kampanye 10DLC yang terdaftar. Jika Anda belum membuat dan mendaftarkan kampanye 10DLC, lihat Mendaftarkan kampanye 10DLC.

Untuk menetapkan kode panjang ke 10DLC

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.
2. Di bawah Settings (Pengaturan), dan kemudian di bawah SMS and voice (SMS dan suara), pilih tab Phone numbers (Nomor telepon).
3. Pilih kode panjang yang ingin Anda konversi ke 10DLC.
4. Untuk membuka Pusat Dukungan, pilih Assign to 10DLC campaign (Tetapkan ke kampanye 10DLC).
5. Untuk jenis kasus, pilih Service limit increase (Peningkatan batas layanan).
6. Untuk Limit type (Jenis batas), pilih Pinpoint.
7. DiPermintaanbagian, pilihWilayah, dan kemudian untukBatasan, pilih10 DLC - Associate kode panjang AS yang ada ke kampanye 10DLC.
8. Di bawah Case description (Deskripsi kasus), untuk Use case description (Deskripsi kasus penggunaan), pastikan untuk menyertakan ID kampanye 10DLC dan nomor kode panjang yang ingin Anda kaitkan dengan kampanye tersebut. Anda dapat menyertakan beberapa kode panjang dalam permintaan, tetapi Anda harus hanya menyertakan satu ID kampanye.
9. Di bawah Contact option (Opsi kontak), untuk Preferred contact language (Bahasa kontak pilihan), pilih bahasa yang ingin digunakan saat berkomunikasi dengan Tim Support AWS.

10. Untuk Contact method (Metode kontak), pilih metode yang Anda inginkan untuk berkomunikasi dengan Tim Support AWS.
11. Pilih Submit (Kirim).

Akses 10DLC lintas akun

Setiap nomor telepon 10DLC dikaitkan dengan satu akun dalam satuAWSWilayah. Jika Anda ingin menggunakan nomor telepon 10DLC yang sama untuk mengirim pesan di lebih dari satu akun atau Wilayah, Anda memiliki dua opsi:

1. Anda dapat mendaftarkan perusahaan dan kampanye yang sama di masing-masingAWSakun. Pendaftaran ini dikelola dan dibebankan secara terpisah. Jika Anda mendaftarkan perusahaan yang sama di beberapaAWSakun, jumlah pesan yang dapat Anda kirim ke pelanggan T-Mobile per hari dibagikan di masing-masing akun tersebut.
2. Anda dapat menyelesaikan proses pendaftaran 10DLC dalam satuAWSakun, dan penggunaanAWS Identity and Access Management(IAM) untuk memberikan izin akun lain untuk mengirim melalui nomor 10DLC Anda.

> **Note**
>
> Opsi ini memungkinkan akses lintas-akun yang benar ke nomor telepon 10DLC Anda. Namun, perhatikan bahwa pesan yang dikirim dari akun sekunder Anda diperlakukan seolah-olah dikirim dari akun utama Anda. Kuota dan penagihan dihitung terhadap akun utama dan tidak terhadap akun sekunder.

Mengatur akses lintas akun menggunakan kebijakan IAM

Anda dapat menggunakan peran IAM untuk mengaitkan akun lain dengan akun utama Anda. Kemudian, Anda dapat mendelegasikan izin akses dari akun utama Anda ke akun sekunder Anda dengan memberi mereka akses ke nomor 10DLC di akun utama.

Untuk memberikan akses ke nomor 10DLC di akun utama

1. Jika Anda belum melakukannya, selesaikan proses pendaftaran 10DLC di akun utama. Proses ini melibatkan tiga langkah:

- Daftarkan perusahaan Anda. Untuk informasi selengkapnya, lihatMendaftarkan perusahaan atau merek Andauntuk digunakan dengan 10DLC.
- Daftarkan kampanye 10DLC Anda (kasus penggunaan). Untuk informasi selengkapnya, lihat Mendaftarkan kampanye 10DLC.
- Mengaitkan nomor telepon dengan kampanye 10DLC Anda. Untuk informasi selengkapnya, lihat Mengaitkan kode panjang dengan kampanye 10DLC.

2. Buat peran IAM di akun utama Anda yang memungkinkan akun lain untuk memanggilPublishAPI operasi untuk nomor telepon 10DLC Anda. Untuk informasi selengkapnya tentang membuat peran, lihatMembuat Peran IAMdiPanduan Pengguna IAM.
3. Delegasikan dan uji izin akses dari akun utama Anda menggunakan IAM role dengan akun lain Anda yang perlu menggunakan nomor 10DLC. Misalnya, Anda dapat mendelegasikan izin akses dari akun Production ke akun Development. Untuk informasi selengkapnya tentang mendelegasikan izin dan pengujian, lihatDelegasikan akses di seluruhAWSAkun menggunakan IAM rolediPanduan Pengguna IAM.
4. Dengan menggunakan peran yang baru, kirim pesan menggunakan nomor 10DLC dari akun utama. Untuk informasi selengkapnya tentang penggunaan peran, lihatMenggunakan peran IAMdi Panduan Pengguna IAM.

Mendapatkan informasi tentang masalah pendaftaran 10DLC

Dalam beberapa situasi, Anda dapat menerima pesan kesalahan ketika Anda mencoba mendaftarkan perusahaan Anda atau kampanye 10DLC Anda.

Masalah pendaftaran perusahaan

Ketika Anda mendaftarkan perusahaan Anda, Anda melihat salah satu dari dua status pendaftaran: Terverifikasi atau Tidak Terverifikasi. Jika status pendaftaran perusahaan Terverifikasi, maka pendaftaran perusahaan Anda berhasil. Anda dapat mulai membuat kampanye 10DLC.

Jika status pendaftaran perusahaan Anda belum diverifikasi, ada masalah dengan informasi yang Anda berikan. Konsol Amazon Pinpoint memberikan informasi tentang alasan pendaftaran perusahaan Anda menerima status ini.

Untuk melihat masalah pendaftaran untuk pendaftaran perusahaan 10DLC Anda

1. Masuk ke AWS Management Console dan buka konsol Amazon Pinpoint di https://console.aws.amazon.com/pinpoint/.

2. Di panel navigasi, di bawah SMS, pilih Nomor telepon.
3. Pada tab kampanye 10DLC, dalam daftar kampanye, pilih nama perusahaan yang ingin Anda temukan informasi selengkapnya.
4. Halaman detail perusahaan berisi informasi tentang masalah yang diidentifikasi dalam pendaftaran Anda. Jika bidang di bagian Info Perusahaan berisi simbol peringatan, masalah pendaftaran terkait dengan informasi di bidang tersebut.

   Periksa informasi yang Anda berikan dan konfirmasikan bahwa semua bidang berisi informasi yang benar. Anda dapat mengedit pendaftaran perusahaan Anda di konsol Amazon Pinpoint. Untuk informasi selengkapnya tentang memodifikasi detail pendaftaran perusahaan Anda, lihatMengedit atau menghapus perusahaan terdaftar.

Masalah pendaftaran kampanye

Saat Anda mendaftarkan kampanye 10DLC, Anda mungkin melihat pesan kesalahan dalam situasi tertentu.

Jika Anda tidak dapat mengidentifikasi masalah dengan pendaftaran Anda, Anda dapat membuat kasus dengan AWS SupportPusat untuk meminta informasi tambahan. Gunakan prosedur berikut untuk membuat kasus AWS Support. Tim AWS Support akan memberikan informasi tentang alasan mengapa pendaftaran kampanye 10DLC Anda ditolak.

Untuk mengirimkan permintaan informasi tentang kampanye 10DLC yang ditolak

1. Masuk ke AWS Management Console di 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 PinpointSMS.
5. Di bawah Permintaan, lengkapi bagian berikut:
   - Untuk Wilayah, pilih tempat Wilayah AWS Anda mencoba mendaftarkan kampanye.

> **Note**
>
> Wilayah diperlukan di bagian Permintaan. Bahkan jika Anda memberikan informasi ini di bagian Rincian kasus, Anda juga harus memasukkannya di sini.

- Untuk Jenis Sumber Daya, pilih Registrasi 10DLC.
- Untuk Limit, pilih Penolakan Pendaftaran Kampanye Perusahaan atau 10DLC.

6. Untuk nilai batas baru, pilih kenaikan batas untuk tipe batas. Biasanya, nilainya **1**.
7. Di bawah Deskripsi kasus, masukkan ID kampanye 10DLC yang ditolak.
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 olahpesan SMS dengan Amazon SNS.
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.

Nomor bebas pulsa

Nomor bebas pulsa (TFN) adalah nomor 10 digit yang dimulai dengan salah satu kode area berikut: 800, 888, 877, 866, 855, 844, atau 833. Anda dapat menggunakan TFN untuk mengirim pesan transaksional saja.

> **Important**
>
> Operator seluler AS baru-baru ini mengubah peraturan mereka dan mewajibkan semua nomor bebas pulsa (TFN) untuk menyelesaikan proses pendaftaran dengan badan pengawas sebelum 30 September 2022. Periksa status TFN Anda dengan pergi ke. the section called “Status pendaftaran nomor bebas pulsa” Untuk informasi lebih lanjut tentang mendaftarkan perusahaan Anda, lihatthe section called “Mendaftarkan nomor bebas pulsa Anda”. Diperlukan waktu hingga 15 hari kerja untuk pendaftaran Anda diproses setelah diserahkan. Pembaruan 3 Maret 2023: Efektif 1 April 2023, operator seluler akan menerapkan ambang batas industri berikut untuk pengiriman pesan yang dikirim melalui nomor bebas pulsa yang tidak terdaftar:

- Batas harian: 500 pesan, reset pukul 12:00 PT
- Batas mingguan: 1.000 pesan, reset hari Minggu pukul 12:00 PT
- Batas bulanan: 2.000 pesan, reset pada akhir bulan kalender pukul 12:00 PT

Pembaruan 19 September 2022: Efektif 1 Oktober 2022, operator seluler akan menerapkan ambang batas seluruh industri berikut untuk pesan yang dikirim melalui nomor bebas pulsa yang tidak terdaftar:

- Batas harian: 2.000 pesan
- Batas mingguan: 12.000 pesan
- Batas bulanan: 25.000 pesan

Kami sangat menyarankan Anda untuk menyelesaikan pendaftaran Anda sesegera mungkin. Pesan yang dikirim melalui TFN yang tidak terdaftar akan menjadi upaya terbaik. Pesan akan dikenakan peningkatan penyaringan dan pemblokiran dari waktu ke waktu karena operator terus membatasi lalu lintas yang tidak terdaftar.

Topik

- Pedoman untuk menggunakan nomor bebas pulsa
- Beli nomor bebas pulsa
- Persyaratan dan proses pendaftaran nomor bebas pulsa
- Status pendaftaran nomor bebas pulsa
- Mengedit, membuang, dan menghapus pendaftaran Anda
- Masalah pendaftaran
- Nomor bebas pulsa pertanyaan yang sering diajukan
- Keuntungan dan kerugian dari nomor bebas pulsa

Pedoman untuk menggunakan nomor bebas pulsa

TFN biasanya hanya digunakan di AS untuk pengiriman pesan transaksional, seperti konfirmasi pendaftaran atau untuk mengirim kata sandi satu kali. Mereka dapat digunakan untuk pesan suara dan SMS. Throughput rata-rata adalah tiga bagian pesan per detik (MPS). Namun, throughput ini

dipengaruhi oleh pengkodean karakter. Untuk informasi selengkapnya tentang cara pengkodean karakter memengaruhi bagian pesan, lihatBatas karakter SMS di Amazon SNS. Untuk informasi lebih lanjut tentang mendaftarkan TFN, lihat. Persyaratan dan proses pendaftaran nomor bebas pulsa

Setiap akun pelanggan dapat memiliki hingga lima TFN. Jika Anda mengirim lebih dari 15 pesan teks per detik tetapi kurang dari 100, kami sarankan Anda mendaftarkan satu atau lebih ID originasi 10DLC. Jika kasus penggunaan Anda memerlukan pengiriman lebih dari 100 pesan teks per detik, kami sarankan Anda membeli dan mendaftarkan satu atau lebih kode pendek.

Saat menggunakan TFN sebagai nomor originasi, ikuti panduan ini:

- Jangan gunakan URL singkat yang dibuat dari pemendek URL pihak ketiga, karena pesan ini lebih mungkin difilter sebagai spam.

  Jika Anda perlu menggunakan URL singkat, pertimbangkan untuk menggunakan nomor 10DLC atau kode pendek. Menggunakan kode pendek dan 10DLC mengharuskan Anda mendaftarkan template pesan Anda, di mana Anda dapat menentukan URL singkat.
- Ketahuilah bahwa respons kata kunci opt-out (STOP) dan opt-in (UNSTOP) ditetapkan pada tingkat operator. Anda tidak dapat memodifikasi kata kunci ini atau kata kunci lainnya. Anda juga tidak dapat mengubah pesan yang dikirim saat pengguna membalas dengan STOP dan UNSTOP.
- Jangan mengirim konten pesan yang sama atau serupa menggunakan beberapa TFN. Operator menyebut praktik ini snowshoeing atau pengumpulan nomor dan menargetkan pesan-pesan ini untuk penyaringan.
- Setiap pesan yang terkait dengan industri berikut dapat dianggap dibatasi dan tunduk pada penyaringan berat atau blok langsung. Ini dapat mencakup kata sandi satu kali (OTP) dan otentikasi multi faktor (MFA) untuk layanan yang terkait dengan kategori terbatas.

  Jika pendaftaran Anda ditolak karena merupakan kasus penggunaan yang tidak sesuai dan Anda merasa penunjukan ini salah, Anda dapat mengirimkan permintaan melalui dukungan. Untuk detail tentang cara melakukannya, lihat Masalah pendaftaran.

  Tabel berikut menjelaskan jenis konten terbatas:

Kategori	Contoh-contoh
Judi	• Aplikasi/situs web • Kasino

Kategori	Contoh-contoh
	• Undian
Layanan keuangan berisiko tinggi	• Pinjaman mobil • Mata Uang Kripto • Penagihan hutang • Pinjaman gajian • Pinjaman berbunga tinggi jangka pendek • Pinjaman hipotek • Pinjaman mahasiswa • Peringatan stok
Pengampunan hutang	• Konsolidasi utang • Pengurangan hutang • Program perbaikan kredit
et-rich-quick Skema G	• ork-from-home Program W • Peluang risiko-investasi • Skema pemasaran piramida atau multi-level
Zat yang dilarang/dikendalikan	• Cannabis/CBD
Phishing	• Upaya untuk membuat pengguna mengungkapkan informasi pribadi atau informasi login situs web
S.H.A.F.T.	• Seks • Benci • Alkohol • Senjata api • Tembakau/Vape

Beli nomor bebas pulsa

Untuk membeli TFN, gunakan konsol Amazon Pinpoint di https://console.aws.amazon.com/sms-voice/. Untuk informasi selengkapnya, lihat Persyaratan dan proses pendaftaran nomor bebas pulsa.

Saat ini, Amazon Pinpoint SMS mendukung nomor bebas pulsa untuk pesan suara dan SMS. Amazon SNS hanya mendukung pesan SMS.

Persyaratan dan proses pendaftaran nomor bebas pulsa

> **Important**
>
> TFN dapat dicabut jika digunakan untuk tujuan apa pun selain kasus penggunaan yang ditentukan.

Nomor bebas pulsa kasus penggunaan terlarang

Amazon SNS memiliki kemampuan terbatas untuk mengirim pesan dalam kasus di mana pesan diblokir (misalnya, kasus penggunaan yang terkait dengan zat yang dikendalikan, atau phishing), atau ketika penyaringan tingkat tinggi diharapkan (misalnya, pesan keuangan berisiko tinggi). Anda mungkin tidak dapat mendaftarkan TFN yang terkait dengan kasus penggunaan konten terbatas yang ditentukan dalam. Pedoman untuk menggunakan nomor bebas pulsa

Mendaftarkan nomor bebas pulsa Anda

Setelah membeli TFN, Anda harus mendaftarkan nomornya. Untuk petunjuk tentang cara melakukannya, lihat Proses pendaftaran nomor bebas pulsa di Panduan Pengguna SMS Amazon Pinpoint.

Pendaftaran swalayan untuk nomor bebas pulsa di wilayah Amazon Pinpoint SMS

Jika Anda telah meminta TFN di wilayah SMS Amazon Pinpoint, selesaikan proses pendaftaran perusahaan langsung di konsol SMS Amazon Pinpoint menggunakan petunjuk yang terdapat di formulir pendaftaran nomor bebas pulsa AS di Panduan Pengguna SMS Amazon Pinpoint.

Saat mendaftarkan TFN Anda, pastikan informasinya lengkap dan akurat, atau pendaftaran Anda dapat ditolak. Informasi yang Anda masukkan harus sama persis dengan kantor pusat perusahaan Anda.

Proses registrasi berbasis formulir manual untuk nomor bebas pulsa di wilayah selain wilayah Amazon Pinpoint SMS

1. Unduh US_TFN_Registration.zip ini dan gunakan contoh formulir pendaftaran (AWS Formulir Pendaftaran Bebas Pulsa AS-Bisnis - Final.docx) untuk melengkapi informasi yang diperlukan dalam file CSV pendaftaran TFN (BulkustFN - Final.csv).

   Setiap permintaan pendaftaran atau kasus penggunaan hanya dapat memiliki hingga lima TFN. Jika Anda yakin Anda memenuhi syarat untuk pengecualian aturan ini, berikan penjelasan rinci untuk dipertimbangkan. Buat daftar semua nomor telepon yang terkait dengan pendaftaran atau kasus penggunaan.
2. Buat kasus dengan AWS Support. Lampirkan file CSV Anda yang sudah selesai ke kasing, dan kirimkan permintaan pendaftaran TFN.
3. Pilih Buat kasus, lalu pilih Mencari peningkatan batas layanan?
4. Untuk jenis batas pilih Pesan Teks SNS.
5. Untuk Jenis Sumber Daya, pilih pendaftaran nomor 10DLC atau Bebas Pulsa.
6. Lampirkan dokumen US_TFN_Registration dan kirimkan permintaan.

Poin kunci yang perlu diperhatikan

1. Pendaftaran dapat memakan waktu hingga dua minggu untuk diproses setelah semua informasi yang diperlukan telah dikirimkan. Jika informasi hilang atau tidak lengkap, proses pendaftaran akan tertunda. Jika pendaftaran Anda ditolak, kami akan membantu Anda menemukan alasan mengapa ditolak dan menyarankan metode untuk meningkatkan kampanye Anda sehingga dapat didaftarkan.
2. TFN bekerja dengan baik untuk kasus penggunaan transaksional seperti otentikasi multi-faktor (MFA) di mana throughput terbatas diperlukan. Setiap TFN dapat mengirim hingga tiga bagian pesan teks per detik, dan setiap akun pelanggan dapat memiliki hingga lima TFN. Jika Anda mengirim lebih dari 15 bagian pesan teks per detik tetapi kurang dari 100, kami sarankan Anda mendaftarkan satu atau lebih ID 10DLC originasi. Jika kasus penggunaan Anda memerlukan pengiriman lebih dari 100 pesan teks per detik, kami sarankan Anda membeli dan mendaftarkan satu atau lebih kode pendek. Untuk detail selengkapnya, lihat Pedoman untuk menggunakan nomor bebas pulsa.

Status pendaftaran nomor bebas pulsa

Untuk memeriksa status pendaftaran Anda, lihat Memeriksa status pendaftaran Anda di Panduan Pengguna SMS Amazon Pinpoint.

Mengedit, membuang, dan menghapus pendaftaran Anda

Gunakan Panduan Pengguna SMS Amazon Pinpoint untuk melakukan tugas-tugas berikut:

- Edit pendaftaran Anda
- Buang pendaftaran Anda
- Hapus pendaftaran Anda
- Lihat sumber daya pendaftaran Anda

Masalah pendaftaran

Jika pendaftaran nomor bebas pulsa Anda tidak diterima, Anda akan melihat pesan yang menjelaskan mengapa nomor tersebut ditolak.

Untuk mengirimkan permintaan informasi tentang nomor bebas pulsa yang ditolak

1. Masuk ke AWS Management Console 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 Pinpoint SMS.
5. Di bawah Permintaan, lengkapi bagian berikut:
   - Untuk Wilayah, tempat Anda mencoba mendaftarkan kampanye.

     > **Note**
     >
     > Wilayah diperlukan di bagian Permintaan. Bahkan jika Anda memberikan informasi ini di bagian Rincian kasus, Anda juga harus memasukkannya di sini.

   - Untuk Jenis Sumber Daya, pilih Registrasi 10DLC atau TFN.

- Untuk Limit, pilih Penolakan Pendaftaran Perusahaan atau Kampanye.

6. Untuk nilai batas baru, pilih kenaikan batas untuk tipe batas. Biasanya, nilainya **1**.
7. (Opsional) Jika Anda ingin mengirimkan permintaan lebih lanjut, pilih Tambahkan permintaan lain. Untuk informasi yang diperlukan, lihat bagian lain di dalamnyaMeminta dukungan untuk olahpesan SMS dengan Amazon SNS.
8. Di bawah Deskripsi kasus, masukkan nomor bebas pulsa yang ditolak.
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.

Nomor bebas pulsa pertanyaan yang sering diajukan

Pertanyaan yang sering diajukan tentang proses pendaftaran TFN.

Apakah saat ini saya memiliki nomor bebas pulsa?

Untuk memeriksa apakah Anda memiliki nomor bebas pulsa

- Buka konsol SMS Amazon Pinpoint di https://console.aws.amazon.com/sms-voice/.
- Di panel navigasi, di bawah SMS, pilih Nomor telepon.
- Jenis TFN terdaftar sebagai bebas pulsa.

Apakah saya harus mendaftarkan nomor bebas pulsa saya?

Ya. Untuk terus menggunakan TFN yang Anda miliki saat ini, Anda harus mendaftarkannya sebelum 30 September 2022. Jika Anda membeli TFN baru setelah 30 September 2022, Anda harus mendaftarkannya sebelum dapat mengirim pesan.

Bagaimana cara membeli nomor bebas pulsa?

Ikuti petunjuk di Meminta nomor telepon menggunakan konsol SMS Amazon Pinpoint untuk membeli TFN.

Bagaimana cara mendaftarkan nomor bebas pulsa saya?

Ikuti petunjuk di the section called “Mendaftarkan nomor bebas pulsa Anda” untuk mendaftarkan TFN.

Apa status pendaftaran nomor bebas pulsa saya dan apa artinya?

Ikuti petunjuk di the section called “Status pendaftaran nomor bebas pulsa” untuk memeriksa pendaftaran dan status Anda.

Informasi apa yang perlu saya berikan?

Anda perlu memberikan alamat perusahaan Anda, kontak bisnis, dan kasus penggunaan untuk TFN. Anda dapat menemukan informasi yang diperlukan dithe section called “Mendaftarkan nomor bebas pulsa Anda”.

Bagaimana jika pendaftaran saya ditolak?

Jika pendaftaran Anda ditolak, status diubah menjadi Memerlukan Pembaruan. Untuk melakukan pembaruan, lihatthe section called “Mengedit, membuang, dan menghapus pendaftaran Anda”.

Izin apa yang saya butuhkan?

Pengguna/peran IAM yang Anda gunakan untuk mengunjungi konsol SMS Amazon Pinpoint harus memiliki izin *“sms-voice: *”*, jika tidak, Anda akan mendapatkan kesalahan akses ditolak.

Keuntungan dan kerugian dari nomor bebas pulsa

Keuntungan

Pencetus bebas pulsa memiliki MPS yang lebih tinggi daripada kode panjang serta kemampuan pengiriman yang baik.

Kekurangan

Tidak ada kontrol atas opt-out dan opt-in, karena ini dikelola di tingkat operator.

Jangan sertakan URL singkat dalam pesan Anda, atau gunakan nomor tersebut untuk mengirim pesan promosi. Sebagai gantinya gunakan nomor 10DLC atau kode pendek. Saat Anda menggunakan kode pendek atau nomor 10DLC, Anda perlu mendaftarkan templat pesan Anda, yang dapat berisi URL singkat dan dapat berupa pesan promosi. Untuk informasi lebih lanjut tentang kode pendek, lihatKode pendek. Untuk informasi lebih lanjut tentang 10DLC, lihat. 10DLC

Kode pendek

Kode pendek adalah urutan numerik yang lebih pendek dari nomor telepon biasa. Misalnya, di Amerika Serikat dan Kanada, nomor telepon standar (kode panjang) terdiri dari 11 digit, sementara kode pendek terdiri dari lima atau enam digit. Amazon SNS mendukung kode pendek khusus.

Kode pendek khusus

Jika Anda mengirim banyak pesan SMS ke penerima di Amerika Serikat atau Kanada, Anda dapat membeli kode pendek khusus. Tidak seperti kode pendek di kolam bersama, kode pendek khusus disediakan untuk penggunaan eksklusif Anda.

Keuntungan

Menggunakan kode pendek yang mudah diingat dapat membantu membangun kepercayaan. Jika Anda perlu mengirim informasi sensitif, seperti kata sandi satu kali (OTP), sebaiknya Anda mengirimkannya menggunakan kode pendek sehingga pelanggan Anda dapat dengan cepat menentukan apakah pesan tersebut benar-benar berasal dari Anda.

Jika Anda menjalankan kampanye akuisisi pelanggan baru, Anda dapat mengundang calon pelanggan untuk mengirim kata kunci ke kode pendek Anda (misalnya, “Teks FOOTBALL ke 10987 untuk berita dan informasi sepakbola”). Kode pendek lebih mudah diingat daripada kode panjang, dan lebih mudah bagi pelanggan untuk memasukkan kode pendek ke perangkat mereka. Dengan mengurangi kesulitan yang dihadapi pelanggan saat mereka mendaftar untuk program pemasaran Anda, Anda dapat meningkatkan efektivitas kampanye Anda.

Karena operator seluler harus menyetujui kode pendek baru sebelum mengaktifkannya, mereka cenderung tidak menandai pesan yang dikirim dari kode pendek sebagai tidak diminta.

Bila Anda menggunakan kode pendek untuk mengirim pesan SMS, Anda dapat mengirim volume pesan yang lebih tinggi per periode 24 jam daripada yang Anda bisa ketika Anda menggunakan jenis identitas asal lainnya. Dengan kata lain, Anda memiliki kuota pengiriman yang jauh lebih tinggi. Anda juga dapat mengirim jauh lebih banyak pesan per detik. Artinya, Anda memiliki tingkat pengiriman yang jauh lebih tinggi.

Kekurangan

Ada biaya tambahan untuk mendapatkan kode pendek, dan perlu waktu lama untuk menerapkannya. Misalnya, di Amerika Serikat, ada biaya penyiapan satu kali sebesar $650,00 (USD) untuk setiap kode pendek, ditambah biaya berulang tambahan sebesar $995,00 per bulan untuk setiap kode pendek. Diperlukan waktu 8-12 minggu sampai kode pendek aktif di semua jaringan operator. Untuk menemukan harga dan waktu penyediaan untuk negara atau wilayah yang berbeda, selesaikan prosedur yang dijelaskan di Meminta kode pendek khusus untuk olahpesan SMS dengan Amazon SNS.

Person-to-person(P2P) kode panjang

> **Important**
>
> Efektif 31 Agustus 2023, nomor khusus seperti a10DLCnomor atau anomor bebas pulsadiperlukan untuk mengirim pesan teks SMS ke Amerika Serikat dan wilayahnya (Puerto Riko, Guam, Kepulauan Samoa Amerika danUSKepulauan Virgin). Permintaan kode panjang Anda akan ditolak jika Anda menggunakan Amerika Serikat sebagai lokasi untuk wilayah ini.

> **Important**
>
> Efektif 1 Juni 2021, penyedia telekomunikasi AS tidak lagi mendukung penggunaanperson-to-person(P2P) kode panjang untukapplication-to-person(A2P) komunikasi ke tujuan AS. Sebaliknya, Anda perlu menggunakan jenis ID asal lain untuk pesan-pesan ini. Untuk informasi selengkapnya, lihat 10DLC.

Kode panjang P2P adalah nomor telepon yang menggunakan format nomor negara atau wilayah tempat penerima Anda berada. Kode panjang P2P juga disebut sebagai nomor panjang atau nomor ponsel virtual. Misalnya, di Amerika Serikat dan Kanada, kode panjang P2P terdiri dari 11 digit: nomor 1 (kode negara), kode area tiga digit, dan nomor telepon tujuh digit.

Untuk informasi selengkapnya tentang meminta kode panjang P2P, lihat Meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P untuk olahpesan SMS dengan Amazon SNS.

Keuntungan

Kode panjang P2P khusus disediakan untuk digunakan oleh akun Amazon SNS Anda saja—kode panjang tersebut tidak dibagikan dengan pengguna lain. Bila Anda menggunakan kode panjang P2P khusus, Anda dapat menentukan kode panjang P2P mana yang ingin Anda gunakan saat Anda mengirim setiap pesan. Jika Anda mengirim beberapa pesan ke pelanggan yang sama, Anda dapat memastikan agar setiap pesan tampak dikirim dari nomor telepon yang sama. Karena itu, kode panjang P2P khusus dapat membantu dalam membangun merek atau identitas Anda.

Kekurangan

Kode panjang P2P tidak didukung untuk komunikasi A2PUStujuan.

Jika Anda mengirim beberapa ratus pesan per hari dari kode panjang P2P khusus, operator seluler mungkin mengidentifikasi nomor Anda sebagai nomor yang mengirim pesan yang tidak diminta. Jika kode panjang P2P Anda ditandai, pesan Anda mungkin tidak akan terkirim ke penerima.

Kode panjang P2P juga memiliki throughput yang terbatas. Laju pengiriman maksimum bervariasi menurut negara. Hubungi Support AWS untuk informasi lebih lanjut. Jika Anda berencana untuk mengirim pesan SMS dalam jumlah besar, atau Anda berencana untuk mengirim lebih dari satu pesan per detik, Anda harus membeli kode pendek khusus.

Beberapa operator tidak mengizinkan Anda menggunakan kode panjang P2P untuk mengirim pesan SMS A2P, termasuk di US. SMS A2P adalah pesan yang dikirim ke perangkat seluler pelanggan ketika pelanggan tersebut mengirimkan nomor ponselnya ke aplikasi. Pesan A2P adalah percakapan satu arah, seperti pesan pemasaran, kata sandi satu kali (OTP), dan pengingat janji temu. Jika Anda berencana untuk mengirim pesan A2P, Anda harus membeli kode pendek khusus (jika pelanggan Anda berada di Amerika Serikat atau Kanada), atau menggunakan ID pengirim (jika penerima Anda berada di negara atau wilayah di mana ID pengirim didukung).

Nomor 10DLC hanya digunakan untuk mengirim pesan di AS. Menggunakan nomor 10DLC mengharuskan Anda mendaftarkan merek perusahaan Anda dan kampanye yang ingin Anda kaitkan dengan nomor tersebut. Setelah disetujui, Anda dapat meminta nomor telepon 10DLC diSMS dan suarahalaman konsol Amazon Pinpoint dihttps://console.aws.amazon.com/pinpoint/. Setelah diminta, waktu untuk menerima persetujuan adalah 7-10 hari. Nomor tersebut tidak dapat digunakan dengan kampanye lain.

Perbandingan nomor produk U.S.

Tabel ini menunjukkan perbandingan dukungan untuk jenis nomor telepon US.

Fitur produk	Kode pendek	Nomor bebas pulsa	10DLC	
Format nomor	5-6 digit	Nomor 10 digit	Nomor 10 digit	
Dukungan saluran	SMS	SMS	SMS	
Jenis lalu lintas SMS	Promosi dan transaksional	Transaksional	Promosi dan transaksional	

Fitur produk	Kode pendek	Nomor bebas pulsa	10DLC	
Memerlukan pemeriksaan	Ya	Tidak	Ya	
Waktu penyediaan yang diperkirakan	12 minggu[1]	15 hari hari kerja hari	1 minggu	
Throughput SMS (jumlah pesan SMS per detik)[2]	100 bagian pesan per detik; throughput yang lebih tinggi tersedia dengan biaya tambahan.	3 bagian pesan per detik	Bervariasi berdasarkan pendaftaran 10DLC Anda. Mendukung hingga 100 bagian pesan per detik.	
Kata kunci diperlukan	Opt-in (ikut serta), opt-out (tidak ikut serta), dan HELP (bantuan)	STOP (berhenti), UNSTOP (batal berhanti). Ini dikelola oleh jaringan. Anda tidak dapat mengubah pilihan opt out (tidak ikut serta) dan opt back (kembali ikut serta) dalam pesan.	Opt-in (ikut serta), opt-out (tidak ikut serta), dan HELP (bantuan)	

[1] Prakiraan waktu penyediaan tidak termasuk waktu persetujuan.

[2] Untuk informasi lebih lanjut tentang ukuran maksimum untuk pesan SMS, lihat Menerbitkan ke ponsel.

Meminta dukungan untuk olahpesan SMS dengan Amazon SNS

Pilihan SMS tertentu dengan Amazon SNS tidak tersedia untuk akun AWS Anda sampai Anda menghubungi AWS Support. Buat kasus di Pusat AWS Support untuk meminta hal-hal berikut:

- Peningkatan ambang batas biaya pengeluaran SMS bulanan Anda

  Secara default, ambang batas pengeluaran bulanan adalah sebesar $1,00 (USD). Ambang batas pengeluaran Anda menentukan volume pesan yang dapat Anda kirim dengan Amazon SNS. Anda dapat meminta ambang batas biaya pengeluaran yang memenuhi volume pesan bulanan yang diharapkan untuk kasus penggunaan SMS Anda.
- Sebuah langkah dari sandbox SMS agar Anda dapat mengirim pesan SMS tanpa batasan. Untuk informasi selengkapnya, lihat Keluar dari sandbox SMS.
- 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. Dukungan untuk ID pengirim bervariasi menurut negara atau wilayah. Untuk informasi selengkapnya, lihat Negara dan wilayah yang didukung.

Saat Anda membuat kasus di Pusat AWS Support , 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 permintaan SMS tertentu, lihat topik berikut.

Topik

- Meminta kode pendek khusus untuk olahpesan SMS dengan Amazon SNS
- Meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P untuk olahpesan SMS dengan Amazon SNS
- Meminta ID pengirim untuk olahpesan SMS dengan Amazon SNS
- Meminta kenaikan kuota pengeluaran SMS bulanan Anda untuk Amazon SNS

Meminta kode pendek khusus untuk olahpesan SMS dengan Amazon SNS

Kode pendek adalah nomor yang dapat Anda gunakan untuk pengiriman pesan SMS volume tinggi. Kode pendek sering digunakan untuk pesan application-to-person (A2P), otentikasi dua faktor (2FA), dan pemasaran. Kode pendek biasanya berisi antara tiga hingga tujuh digit, tergantung pada negara atau wilayahnya.

Anda hanya dapat menggunakan kode pendek untuk mengirim pesan ke penerima di negara yang sama dengan kode pendek tersebut. Jika kasus penggunaan mengharuskan Anda menggunakan kode pendek di lebih dari satu negara, Anda harus meminta kode pendek terpisah untuk setiap negara tempat penerima Anda berada.

Untuk informasi selengkapnya tentang harga kode pendek, lihat harga Amazon SNS.

Important

Jika Anda baru menggunakan olahpesan SMS dengan Amazon SNS, mintalah ambang batas pengeluaran SMS bulanan yang memenuhi permintaan yang diharapkan dari kasus penggunaan SMS Anda. Secara default, ambang batas pengeluaran bulanan Anda adalah $1,00 (USD). Anda dapat meminta untuk meningkatkan ambang batas pengeluaran Anda dalam kasus dukungan yang sama yang mencakup permintaan Anda untuk kode pendek. Atau, Anda dapat menggunakan kasus terpisah. Untuk informasi selengkapnya, lihat Meminta kenaikan kuota pengeluaran SMS bulanan Anda untuk Amazon SNS.
Selain itu, jika Anda meminta kode pendek khusus untuk mengirim pesan yang berisi atau mungkin berisi Protected Health Information (PHI), Anda harus menjelaskan tujuan ini di Deskripsi kasus saat Anda membuka kasus dukungan, seperti yang dijelaskan di bawah ini.

Buka kasus dukungan kode pendek Amazon SNS

Buka kasus AWS Support dengan melakukan langkah-langkah berikut.

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.

Untuk meminta kode pendek khusus

1. Buka Pusat Dukungan AWS.
2. Masuk ke AWS Management Console di https://console.aws.amazon.com/.
3. Di menu Dukungan, pilih Pusat Dukungan.
4. Pada panel Kasus dukungan Anda, pilih Buat kasus.
5. Pilih peningkatan batas Mencari layanan? link, lalu lengkapi yang berikut ini:
   - Untuk tipe Limit, pilih SNSText Messaging lalu lengkapi yang berikut ini:
   - (Opsional) Untuk Berikan tautan ke situs atau aplikasi yang akan mengirim pesan SMS, berikan tautan ke situs atau nama aplikasi tempat anggota audiens Anda memilih untuk menerima pesan SMS Anda.
   - (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 AWSWilayah 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 yang Anda rencanakan untuk mengirim pesan SMS.
   - (Opsional) Dalam Bagaimana pelanggan Anda memilih untuk menerima pesan dari Anda, berikan deskripsi tentang bagaimana pelanggan Anda memilih untuk menerima pesan dari 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.
6. Di bawah Permintaan, lengkapi bagian berikut:
   - Untuk Wilayah, pilih permintaan kode singkat Anda Wilayah AWS

> **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 Dedicated SMS Short Codes (Kode Pendek SMS Khusus).
- Untuk Limit (Batasan), pilih opsi yang paling mirip dengan kasus penggunaan Anda.

7. Untuk nilai batas baru, pilih jumlah ID pengirim yang Anda minta. Biasanya, nilainya **1**.
8. (Opsional) Jika Anda ingin mengirimkan permintaan lebih lanjut, pilih Tambahkan permintaan lain. Untuk informasi yang diperlukan, lihat bagian lain di dalamnyaMeminta dukungan untuk olahpesan SMS dengan Amazon SNS.
9. Di bawah Deskripsi kasus, rangkum kasus penggunaan Anda dan sertakan bagaimana penerima Anda akan mendaftar untuk pesan yang dikirim dengan kode singkat Anda dan berikan informasi berikut:

   - Informasi perusahaan:
     - Nama perusahaan
     - Alamat surat perusahaan
     - Nama dan nomor telepon untuk kontak utama untuk permintaan Anda
     - Alamat email dan nomor bebas pulsa untuk dukungan di perusahaan Anda
     - ID pajak perusahaan
     - Nama produk atau layanan Anda

   - Proses pendaftaran pengguna:
     - Situs web perusahaan, atau situs web tempat pelanggan Anda akan mendaftar untuk menerima pesan dari kode pendek Anda.
     - Cara pengguna mendaftar untuk menerima pesan dari kode pendek Anda. Tentukan satu atau beberapa opsi berikut:
       - **`Text messages`**
       - **`Website`**
       - **`Mobile app`**

    - **Other**Jika lain, jelaskan.
  - Teks untuk opsi pendaftaran pesan di situs web, aplikasi, atau di tempat lain.
  - Urutan pesan yang akan Anda gunakan untuk keikutsertaan ganda. Berikan semua hal berikut:
    1. Pesan SMS yang akan Anda kirimkan saat pengguna mendaftar. Pesan ini meminta persetujuan pengguna untuk pesan berulang. Misalnya: ExampleCorp: Balas YA untuk menerima peringatan transaksi akun. Tarif pesan & data mungkin berlaku.
    2. Respons keikutsertaan yang Anda harapkan dari pengguna. Respons ini biasanya berupa kata kunci, seperti YA.
    3. Pesan konfirmasi yang ingin Anda kirim ketika pelanggan mengirim kata kunci ini ke kode pendek Anda. Misalnya: Anda sekarang terdaftar untuk peringatan akun dari ExampleCorp. Tarif pesan & data mungkin berlaku. Ketik STOP untuk membatalkan atau BANTUAN untuk info.

- Tujuan dari pesan Anda:
  - Tujuan dari pesan yang akan Anda kirim dengan kode pendek Anda. Tentukan satu atau beberapa opsi berikut:
    - **Promotions and marketing**
    - **Location-based services**
    - **Notifications**
    - **Information on demand**
    - **Group chat**
    - **Two-factor authentication (2FA)**
    - **Polling and surveys**
    - **Sweepstakes or contests**
    - **Other**Jika lain, jelaskan.
  - Jika Anda berencana menggunakan kode pendek Anda untuk mengirim pesan promosi atau pemasaran untuk bisnis selain milik Anda sendiri.
  - Atau jika Anda berencana menggunakan kode pendek untuk mengirim pesan yang mungkin akan berisi Protected Health Information (PHI), sebagaimana yang ditentukan oleh Health Insurance Portability and Accountability Act (HIPAA) dan undang-undang serta peraturan terkait.

- Isi pesan:
  - Pesan yang akan dikirim ketika pelanggan memilih untuk menerima pesan Anda dengan mengirimkan kata kunci tertentu kepada Anda. Hati-hati saat Anda menentukan kata kunci dan pesan ini—mungkin perlu beberapa minggu untuk mengubah pesan ini. Ketika kami membuat kode pendek Anda, kami mendaftarkan kata kunci dan pesan dengan operator ponsel di negara tempat Anda menggunakan kode pendek. Pesan Anda mungkin menyerupai contoh berikut: Selamat datang di *ProductName*peringatan! Tarif pesan & data berlaku. *2* pesan per bulan. Balas BANTUAN untuk bantuan, STOP untuk membatalkan.
  - Respons yang ingin Anda kirim ketika pelanggan membalas pesan Anda dengan kata kunci BANTUAN. Pesan ini harus menyertakan informasi kontak dukungan pelanggan. Misalnya: *ProductName*Peringatan: Bantuan di *example.com/help* atau *(*800) 555-0199. Tarif pesan & data berlaku. *2* pesan per bulan. Balas STOP untuk membatalkan.
  - Respons yang ingin Anda kirim ketika pelanggan membalas pesan Anda dengan kata kunci STOP. Pesan ini harus mengonfirmasi bahwa pengguna tidak akan lagi menerima pesan dari Anda. Misalnya: Anda berhenti berlangganan dari *ProductName*Alerts. Tidak ada lagi pesan yang akan dikirim. Balas BANTUAN untuk mendapatkan bantuan atau hubungi *(800) 555-0199*.
  - Teks yang akan dikirim sebagai pengingat berkala bahwa pengguna berlangganan pesan Anda. Misalnya: Pengingat: Anda berlangganan peringatan akun dari. ExampleCorp Tarif pesan & data mungkin berlaku. Ketik STOP untuk membatalkan atau BANTUAN untuk info.
  - Contoh setiap jenis pesan yang akan dikirim menggunakan kode pendek Anda. Berikan setidaknya tiga contoh. Jika Anda berencana untuk mengirim lebih dari tiga jenis pesan, berikan contoh untuk semuanya.

> **Important**
>
> Operator seluler mengharuskan kami untuk memberikan semua informasi yang tercantum di atas untuk menyediakan kode pendek. Kami tidak dapat memproses permintaan Anda sampai Anda memberikan semua informasi ini.

10. (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 olahpesan SMS dengan Amazon SNS.

11. Di bawah Opsi kontak, untuk Bahasa kontak pilihan, pilih bahasa yang Anda inginkan untuk menerima komunikasi untuk kasus ini.
12. Setelah selesai, pilih Submit (Kirim).

Setelah kami menerima permintaan Anda, kami akan memberikan respons awal dalam 24 jam. Kami mungkin akan menghubungi Anda untuk meminta informasi tambahan. Jika kami dapat memberikan kode pendek kepada Anda, kami akan mengirimkan informasi tentang biaya yang terkait dengan penggunaan kode pendek di negara atau wilayah yang Anda tentukan dalam permintaan Anda. Kami juga menyediakan perkiraan waktu yang diperlukan untuk menyediakan kode pendek di negara atau wilayah Anda. Biasanya diperlukan beberapa minggu untuk menyediakan kode pendek, walaupun jauh lebih cepat atau lebih lama tergantung pada negara atau wilayah tempat kode pendek digunakan.

> **Note**
>
> Biaya yang terkait dengan penggunaan kode pendek segera dimulai setelah kami memulai permintaan kode pendek Anda dengan operator. Anda bertanggung jawab untuk membayar biaya ini, bahkan jika kode pendek belum sepenuhnya disediakan.

Untuk mencegah sistem kami digunakan untuk mengirim konten yang tidak diinginkan atau berbahaya, kami mempertimbangkan setiap permintaan dengan hati-hati. Jika kasus penggunaan Anda tidak sesuai dengan kebijakan kami, kami mungkin tidak dapat mengabulkan permintaan Anda.

Langkah selanjutnya

Anda telah mendaftarkan kode pendek dengan operator nirkabel dan meninjau pengaturan Anda di konsol Amazon SNS. Sekarang Anda dapat menggunakan Amazon SNS untuk mengirim pesan SMS dengan kode pendek Anda sebagai nomor asal.

Meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P untuk olahpesan SMS dengan Amazon SNS

> **Important**
>
> 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. Sebaliknya, Anda perlu menggunakan jenis ID asal lain untuk pesan-pesan ini. Untuk informasi selengkapnya, lihat 10DLC.

Untuk meminta nomor 10DLC, nomor bebas pulsa, dan kode panjang P2P, gunakan konsol Amazon Pinpoint. Untuk instruksi mendetail, lihat Meminta nomor di Panduan Pengguna Amazon Pinpoint.

Important

Operator seluler AS baru-baru ini mengubah peraturan mereka, dan akan mengharuskan semua nomor bebas pulsa (TFN) menyelesaikan proses pendaftaran dengan badan pengawas sebelum 30 September 2022. Untuk informasi lebih lanjut tentang mendaftarkan nomor bebas pulsa, lihat. Mendaftarkan nomor bebas pulsa Anda
Jika Anda membeli nomor bebas pulsa pada atau sebelum 30 September 2022, statusnya akan berada dalam status Aktif hingga 1 Oktober 2022, kecuali Anda telah menyelesaikan pendaftaran dan telah dikembalikan dengan negara yang disetel ke Selesai. Jika tidak, itu akan ditempatkan dalam status Tertunda dan Anda tidak akan dapat mengirim pesan dengannya sampai Anda telah mendaftarkan nomornya, pendaftaran telah dikembalikan atau pendaftaran diatur ke status Aktif.
Pendaftaran dapat memakan waktu hingga 15 hari kerja.

Untuk mendaftarkan perusahaan dan kampanye 10DLC Anda, buka konsol Amazon Pinpoint di Wilayah AS Timur (Virginia N.). Alih-alih meminta nomor 10DLC, gunakan konsol Service Quotas AWS untuk membuat kasus peningkatan batas layanan sambil meminta nomor 10DLC untuk Wilayah tersebut. Untuk informasi tentang Wilayah di mana Amazon Pinpoint tersedia, lihat titik akhir dan kuota Amazon Pinpoint di. Referensi Umum AWS

Note

Jika Anda baru menggunakan olahpesan SMS dengan Amazon SNS, mintalah ambang batas pengeluaran SMS bulanan yang memenuhi permintaan yang diharapkan dari kasus penggunaan SMS Anda. Secara default, ambang batas pengeluaran bulanan Anda adalah $1,00 (USD). Lihat informasi yang lebih lengkap di Meminta kenaikan kuota pengeluaran SMS bulanan Anda untuk Amazon SNS.

Meminta ID pengirim untuk olahpesan SMS dengan Amazon SNS

> **Important**
>
> Jika Anda baru menggunakan olahpesan SMS dengan Amazon SNS, mintalah ambang batas pengeluaran SMS bulanan yang memenuhi permintaan yang diharapkan dari kasus penggunaan SMS Anda. Secara default, ambang batas pengeluaran bulanan Anda adalah $1,00 (USD). Anda dapat meminta untuk meningkatkan ambang batas pengeluaran Anda dalam kasus dukungan yang sama yang mencakup permintaan Anda untuk ID pengirim. Atau, jika mau, Anda dapat membuka kasus terpisah. Untuk informasi selengkapnya, lihat Meminta kenaikan kuota pengeluaran SMS bulanan Anda untuk Amazon SNS.

Dalam olahpesan SMS, ID pengirim adalah nama yang muncul sebagai pengirim pesan di perangkat penerima. ID Pengirim adalah cara yang baik agar penerima pesan mengenali identitas Anda.

Dukungan untuk ID pengirim bervariasi menurut negara. Misalnya, operator di Amerika Serikat sama sekali tidak mendukung ID pengirim, tetapi operator di India mengharuskan pengirim untuk menggunakan ID pengirim. Untuk daftar lengkap negara yang mendukung ID pengirim, lihat Negara dan wilayah yang didukung.

> **Important**
>
> Beberapa negara mengharuskan Anda untuk mendaftarkan ID pengirim sebelum menggunakannya untuk mengirim pesan. Tergantung pada negara, proses pendaftaran ini mungkin memakan waktu beberapa minggu. Negara yang memerlukan ID pengirim terdaftar ditunjukkan dalam tabel di halaman Negara-negara yang didukung.
> Anda dapat menggunakan dan mendaftarkan ID pengirim yang sama di beberapa AWS Akun untuk pesan SMS. Jika Anda memiliki dukungan perusahaan dan mendaftarkan beberapa templat di beberapa akun, ikuti langkah-langkah di bawah ini, dan bekerjalah dengan Manajer Akun Teknis Anda untuk memastikan bahwa pengalaman orientasi Anda terkoordinasi.

Jika Anda mengirim pesan ke penerima di negara di mana ID pengirim didukung, dan negara tersebut tidak mengharuskan Anda untuk mendaftarkan ID pengirim, Anda tidak perlu melakukan langkah tambahan apa pun. Anda dapat segera mulai mengirim pesan yang menyertakan nilai ID pengirim.

Anda hanya perlu melakukan prosedur di halaman ini jika Anda berencana untuk mengirim pesan ke negara di mana pendaftaran ID pengirim diperlukan.

Langkah 1: Buka kasus SMS Amazon SNS

Jika Anda berencana mengirim pesan ke penerima di negara yang memerlukan ID pengirim, Anda dapat meminta ID pengirim dengan membuat kasus baru di Pusat Dukungan AWS.

> **Note**
>
> Jika Anda berencana mengirim pesan ke penerima di negara yang mengizinkan penggunaan ID pengirim namun tidak memerlukannya, Anda tidak perlu membuka kasus di Pusat Dukungan. Anda dapat segera mulai mengirim pesan yang menyertakan nilai ID pengirim.

Cara meminta ID pengirim

1. Masuk ke AWS Management Console di 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 Pinpoint SMS.
   - (Opsional) Untuk Menyediakan tautan ke situs atau aplikasi yang akan mengirim pesan SMS, mengidentifikasi situs web atau aplikasi tempat anggota audiens Anda memilih untuk menerima pesan SMS Anda.
   - (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 AWSWilayah mana Anda akan mengirim pesan dari, pilih dari mana Anda akan mengirim pesan SMS. Wilayah AWS
- (Opsional) Untuk negara mana Anda berencana untuk mengirim pesan, masukkan negara tempat Anda ingin mendaftarkan ID pengirim. Dukungan untuk ID pengirim dan persyaratan pendaftaran ID pengirim bervariasi berdasarkan negara. Untuk informasi selengkapnya, lihat Negara dan wilayah yang didukung.

  Jika daftar negara melebihi jumlah karakter yang diizinkan oleh kotak teks ini, Anda dapat mencantumkan negara di bagian Deskripsi kasus.
- (Opsional) Dalam Bagaimana pelanggan Anda memilih untuk menerima pesan dari Anda, berikan deskripsi tentang bagaimana pelanggan Anda memilih untuk menerima pesan dari Anda.
- (Opsional) Di kolom Harap berikan templat pesan yang Anda rencanakan untuk digunakan untuk mengirim pesan ke pelanggan Anda, sertakan templat pesan apa pun yang akan Anda gunakan.

5. Di bawah Permintaan, lengkapi bagian berikut:

   - Untuk Wilayah, pilih ID pengirim Anda. Wilayah AWS

     > **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 SMS Production Access.
6. Untuk nilai batas baru, pilih jumlah ID pengirim yang Anda minta. Biasanya, nilainya **1**.
7. (Opsional) Jika Anda ingin mengirimkan permintaan lebih lanjut, pilih Tambahkan permintaan lain. Untuk informasi yang diperlukan, lihat bagian lain di dalamnyaMeminta dukungan untuk olahpesan SMS dengan Amazon SNS.
8. Di bawah deskripsi Kasus, untuk deskripsi kasus Penggunaan, berikan rincian berikut:

   - ID pengirim yang ingin Anda daftarkan.
   - Templat yang akan Anda gunakan untuk pesan SMS Anda.
   - Jumlah pesan yang akan dikirim ke setiap penerima per bulan.

- Informasi tentang cara pelanggan Anda memilih untuk menerima pesan dari Anda.
- Nama perusahaan atau organisasi Anda.
- Alamat yang terkait dengan perusahaan atau organisasi Anda.
- Negara tempat perusahaan atau organisasi Anda berada.
- Nomor telepon untuk perusahaan atau organisasi Anda.
- URL situs web untuk perusahaan atau organisasi Anda.

9. Di bawah Opsi kontak, untuk Bahasa kontak pilihan, pilih bahasa yang Anda inginkan untuk menerima komunikasi untuk kasus ini.
10. Setelah selesai, pilih Submit (Kirim).

Setelah kami menerima permintaan Anda, kami akan memberikan respons awal dalam 24 jam. Kami mungkin akan menghubungi Anda untuk meminta informasi tambahan. Jika kami dapat memberikan ID Pengirim, kami akan mengirimkan perkiraan waktu yang diperlukan untuk menyediakannya.

Untuk mencegah sistem kami digunakan untuk mengirim konten yang tidak diinginkan atau berbahaya, kami mempertimbangkan setiap permintaan dengan hati-hati. Jika kasus penggunaan Anda tidak sesuai dengan kebijakan kami, kami mungkin tidak dapat mengabulkan permintaan Anda.

Langkah 2: Perbarui pengaturan SMS Anda di konsol Amazon SNS

Saat proses mendapatkan ID pengirim Anda telah selesai, kami akan merespons kasus Anda. Ketika Anda menerima notifikasi ini, selesaikan langkah-langkah di bagian ini untuk mengonfigurasi Amazon SNS agar dapat menggunakan ID pengirim Anda sebagai ID pengirim default untuk semua pesan yang dikirim menggunakan akun Anda. Atau, Anda dapat memilih untuk menentukan ID pengirim mana yang akan digunakan saat mengirimkan pesan.

1. Masuk ke konsol Amazon SNS.
2. Pada panel navigasi, pilih Seluler, lalu pilih Pesan teks (SMS).
3. Di bagian Preferensi pesan teks, pilih Edit.
4. Di bagian Detail, di bidang ID pengirim default, masukkan ID pengirim yang disediakan untuk digunakan sebagai default untuk semua pesan dari akun Anda.
5. Setelah selesai, pilih Simpan perubahan.

Langkah selanjutnya

Anda telah mendaftarkan ID pengirim dan memperbarui pengaturan Anda di konsol Amazon SNS. Sekarang Anda dapat menggunakan Amazon SNS untuk mengirim pesan SMS dengan ID pengirim Anda. Penerima SMS di negara yang didukung akan melihat ID pengirim Anda sebagai pengirim pesan di perangkat mereka. Jika ID pengirim yang berbeda digunakan saat mengirim pesan, ID pengirim akan menimpa ID default yang dikonfigurasi di sini.

Meminta kenaikan kuota pengeluaran SMS bulanan Anda untuk Amazon SNS

Amazon SNS menyediakan kuota pengeluaran untuk membantu Anda mengelola biaya maksimum per bulan yang dikeluarkan dengan mengirim SMS menggunakan akun Anda. 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 pesan SMS ketika menentukan bahwa pengiriman pesan SMS 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 kasing SMS Amazon SNS. 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 SMS Anda. 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, ia berhenti mengirim pesan SMS dalam beberapa menit jika kuota pengeluaran terlampaui. Selama periode ini, jika Anda terus mengirim pesan SMS, Anda mungkin akan dikenakan biaya yang melebihi kuota Anda.

Kami menetapkan kuota pengeluaran untuk semua akun baru sebesar $1,00 (USD) per bulan. Kuota ini dimaksudkan agar Anda dapat mencoba kemampuan pengiriman pesan dari Amazon SNS. Untuk meminta peningkatan kuota pengeluaran SMS untuk akun Anda, buka kasus peningkatan kuota di Pusat Dukungan AWS.

Langkah 1: Buka kasus SMS Amazon SNS

Untuk meminta peningkatan kuota pengeluaran bulanan Anda dengan membuka kasus peningkatan kuota di Pusat Dukungan AWS.

> **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 di 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 jenis Limit, pilih Pesan SNS Teks.
   - (Opsional) Untuk Menyediakan tautan ke situs atau aplikasi yang akan mengirim pesan SMS, memberikan informasi tentang situs web, aplikasi, atau layanan yang akan mengirim pesan SMS.
   - (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 AWSWilayah 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 New limit, masukkan jumlah maksimum (dalam USD) yang dapat Anda belanjakan untuk 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 pesan SMS.
   - Layanan yang disediakan oleh situs web atau aplikasi Anda, dan bagaimana pesan SMS Anda berkontribusi pada layanan itu.
   - Bagaimana pengguna mendaftar untuk secara sukarela menerima pesan SMS 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 rincian 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 olahpesan SMS dengan Amazon SNS.
9. Di bawah Opsi kontak, untuk Bahasa kontak pilihan, pilih bahasa yang Anda inginkan untuk menerima komunikasi untuk kasus ini.
10. Setelah selesai, pilih Submit (Kirim).

Tim AWS Support memberikan respons 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 pengaturan SMS Anda di konsol Amazon SNS

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 pengeluaran SMS Anda tidak akan meningkat.

Cara menyesuaikan kuota pengeluaran Anda di konsol tersebut

1. Masuk ke Konsol Amazon SNS.
2. Buka menu navigasi kiri, perluas Seluler, lalu pilih Pesan teks (SMS).
3. Di halaman Mobile text messaging (SMS) (Olahpesan teks seluler (SMS)), di bagian Text messaging preferences(Preferensi pesan teks), pilih Edit.
4. Pada halaman Edit preferensi pesan teks, di bagian Detail, masukkan batas pengeluaran SMS 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 Save changes (Simpan perubahan).

> **Note**
>
> Jika Anda mendapatkan kesalahan "Parameter Invalid" (Parameter Tidak Valid), periksa kontak dari Support AWS dan konfirmasikan bahwa Anda memasukkan batas pengeluaran SMS baru yang benar. Jika Anda masih mengalami masalah, buka kasus di Pusat Dukungan AWS.

Mengatur preferensi olahpesan SMS

Gunakan Amazon SNS untuk menentukan preferensi untuk olahpesan SMS. Misalnya, Anda dapat menentukan jika akan mengoptimalkan pengiriman untuk biaya atau keandalan, batas pengeluaran bulanan Anda, bagaimana pengiriman dicatat, dan jika akan berlangganan laporan penggunaan SMS harian atau tidak.

Preferensi ini berlaku untuk setiap pesan SMS yang Anda kirim dari akun Anda, namun Anda dapat mengganti sebagian pesan tersebut saat mengirim pesan individual. Untuk informasi selengkapnya, lihat Menerbitkan ke ponsel.

Topik

- Mengatur preferensi pesan SMS menggunakan AWS Management Console
- Pengaturan preferensi (AWS SDK)

Mengatur preferensi pesan SMS menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
2. Pilih wilayah yang mendukung olahpesan SMS.
3. Di panel navigasi, pilih Mobile (Seluler), Text messaging (SMS) (Olahpesan teks (SMS)).

4. Di halaman Mobile text messaging (SMS) (Olahpesan teks seluler (SMS)), di bagian Text messaging preferences(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 agar dikenakan biaya terendah.
      - Transaksional (default) — Pesan penting yang mendukung transaksi pelanggan, seperti kode sandi satu kali untuk otentikasi multi-faktor. Amazon SNS mengoptimalkan pengiriman pesan agar mencapai keandalan tertinggi.

      Untuk informasi harga untuk pesan promosi dan transaksional, lihat Harga SMS Global.

   b. (Opsional) Untuk Account spend limit (Batas pengeluaran akun), masukkan jumlah (dalam USD) yang ingin Anda gunakan untuk pengeluaran pesan SMS setiap bulannya.

      > **Important**
      >
      > - Secara default, kuota pengeluaran diatur sebesar 1,00 USD. Jika Anda ingin meningkatkan kuota layanan, kirimkan permintaan.
      > - Jika jumlah yang ditetapkan di konsol tersebut melebihi kuota layanan Anda, Amazon SNS berhenti mengirimkan pesan SMS.
      > - Karena Amazon SNS adalah sistem terdistribusi, ia berhenti mengirimkan pesan SMS dalam beberapa menit sejak kuota pengeluaran terlampaui. Selama interval ini, jika Anda terus mengirim pesan SMS, Anda mungkin akan 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**
   >
   > Dukungan untuk ID pengirim bervariasi berdasarkan negara atau wilayah.

7. (Opsional) Masukkan nama Nama bucket Amazon S3 untuk laporan penggunaan.

> **Note**
>
> Kebijakan bucket S3 harus memberikan akses tulis ke Amazon SNS.

8. Pilih Save changes (Simpan perubahan).

Pengaturan preferensi (AWS SDK)

Untuk mengatur preferensi SMS Anda menggunakan salah satu AWS SDK, gunakan tindakan dalam SDK yang sesuai dengan `SetSMSAttributes` permintaan di Amazon SNS API. Dengan permintaan ini, Anda menetapkan nilai ke atribut SMS yang berbeda, seperti kuota pengeluaran bulanan dan jenis SMS default Anda (promosi atau transaksional). Untuk semua atribut SMS, lihat tindakan SetSMSAttributes di Referensi API Amazon Simple Notification Service.

Contoh kode berikut menunjukkan cara menggunakan`SetSMSAttributes`.

C++

SDK untuk C++

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

Cara menggunakan Amazon SNS untuk mengatur atribut defaultsMStype.

```
//! 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 detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for C++

CLI

AWS CLI

Untuk mengatur atribut pesan SMS

`set-sms-attributes`Contoh berikut menetapkan ID pengirim default untuk pesan SMS ke`MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat SetSmSatTributes di Referensi Perintah.AWS CLI

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for Java 2.x

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for JavaScript

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for PHP

Mengirim pesan SMS

Bagian ini menjelaskan cara mengirim pesan SMS.

Topik

- Menerbitkan ke topik
- Menerbitkan ke ponsel

Menerbitkan ke topik

Anda dapat menerbitkan satu pesan SMS ke banyak nomor telepon sekaligus dengan berlangganan nomor telepon tersebut ke topik Amazon SNS. Topik SNS adalah saluran komunikasi di mana Anda dapat menambahkan pelanggan dan kemudian menerbitkan pesan ke semua pelanggan tersebut. Pelanggan menerima semua pesan yang diterbitkan ke topik sampai Anda membatalkan langganan, atau sampai pelanggan memilih untuk tidak menerima pesan SMS dari akun AWS Anda.

Topik

- Mengirim pesan ke topik (konsol)
- Mengirim pesan ke topik (SDK AWS)

Mengirim pesan ke topik (konsol)

Cara membuat topik

Lakukan langkah-langkah berikut jika Anda belum memiliki topik untuk dikirimi pesan SMS.

1. Masuk ke konsol Amazon SNS.
2. Di menu konsol tersebut, pilih Wilayah AWS yang mendukung olahpesan SMS.
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 Display name (Nama tampilan), masukkan prefiks kustom untuk pesan SMS Anda. Ketika Anda mengirim pesan ke topik, Amazon SNS menambahkan nama tampilan diikuti tanda kurung siku (**>**) 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 Amazon Resource Name (ARN) muncul di halaman Topics (Topik).

Cara membuat langganan SMS

Anda dapat menggunakan langganan untuk mengirim pesan SMS ke beberapa penerima hanya dengan menerbitkan pesan ke topik Anda satu kali.

> **Note**
>
> Ketika Anda mulai menggunakan Amazon SNS untuk mengirim pesan SMS, akun AWS Anda berada di sandbox SMS. Sandbox SMS menyediakan lingkungan yang aman bagi Anda untuk mencoba fitur Amazon SNS tanpa mempertaruhkan reputasi Anda sebagai pengirim SMS. Saat akun Anda berada di sandbox SMS, Anda dapat menggunakan semua fitur Amazon SNS, tetapi Anda hanya dapat mengirim pesan SMS ke nomor telepon tujuan yang terverifikasi. Untuk informasi selengkapnya, lihat Sandbox SMS.

1. Masuk ke konsol Amazon SNS.
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 Topic ARN (ARN topik), masukkan atau pilih Amazon Resource Name (ARN) topik yang ingin Anda kirimi pesan SMS.
   b. Untuk Protocol (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

Ketika Anda menerbitkan pesan ke topik, Amazon SNS mencoba untuk mengirimkan pesan tersebut ke setiap nomor telepon yang berlangganan ke topik.

1. Di konsol Amazon SNS, di halaman Topics (Topik), pilih nama topik yang ingin Anda kirimi pesan SMS.
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 Subject (Subjek), kosongkan bidang kecuali topik Anda berisi langganan email dan Anda ingin menerbitkannya ke langganan email dan langganan SMS. Amazon SNS menggunakan Subject (Subjek) yang Anda masukkan sebagai baris subjek email.

   b. (Opsional) Untuk Time to Live (TTL) (waktu untuk tayang), masukkan jumlah detik yang digunakan Amazon SNS untuk mengirim pesan SMS Anda ke setiap pelanggan titik akhir aplikasi seluler.

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 menambah panjang pesan. Panjang nama tampilan adalah jumlah karakter dalam nama tersebut ditambah dua karakter untuk tanda kurung siku (>) dan ruang yang ditambahkan Amazon SNS.

      Untuk informasi tentang kuota ukuran untuk pesan SMS, lihat Menerbitkan ke ponsel.

5. (Opsional) Untuk Message attributes (Atribut pesan), tambahkan metadata pesan seperti stempel waktu, tanda tangan, dan ID.
6. Pilih Publish message (Publikasikan Pesan). Amazon SNS mengirimkan pesan SMS dan menampilkan pesan keberhasilan.

Mengirim pesan ke topik (SDK AWS)

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

Contoh kode berikut ini menunjukkan cara untuk melakukan

- Buat topik Amazon SNS.
- Berlangganan nomor telepon ke topik.
- Publikasikan pesan SMS ke topik sehingga semua nomor telepon berlangganan menerima pesan sekaligus.

Java

SDK untuk 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 kembalikan ARN-nya.

```
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 ke ponsel

Anda dapat menggunakan Amazon SNS untuk mengirim pesan SMS langsung ke ponsel tanpa membuat nomor telepon berlangganan ke topik Amazon SNS.

> **Note**
>
> Membuat nomor telepon berlangganan ke topik berguna jika Anda ingin mengirim satu pesan ke beberapa nomor telepon sekaligus. Untuk petunjuk penerbitan pesan SMS ke topik, lihat Menerbitkan ke topik.

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 SDK, Anda dapat menentukan harga maksimum untuk pengiriman pesan.

Setiap pesan SMS dapat berisi hingga 140 byte, dan kuota karakter tergantung pada skema pengkodean. Misalnya, sebuah pesan SMS dapat berisi:

- 160 karakter GSM
- 140 karakter ASCII
- 70 karakter UCS-2

Jika Anda menerbitkan pesan yang melebihi kuota ukuran, Amazon SNS mengirimkannya sebagai beberapa pesan, masing-masing sesuai dalam kuota ukuran. Pesan tidak terputus di tengah kata, melainkan di batas seluruh kata. Kuota ukuran total untuk satu tindakan penerbitan SMS adalah 1.600 byte.

Ketika Anda mengirim pesan SMS, 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 nampak sebagai +1XXX5550100.

Topik

- Mengirim pesan (konsol)
- Mengirim pesan (AWS SDK)

Mengirim pesan (konsol)

1. Masuk ke konsol Amazon SNS.
2. Di menu konsol tersebut, pilih Wilayah AWS yang mendukung olahpesan SMS.
3. Di panel navigasi, pilih Text messaging (SMS) (Pesan teks).
4. Di halaman Mobile text messaging (SMS) (Pesan teks seluler (SMS)), pilih Publish text message (Publikasikan pesan teks).
5. Di halaman Publish text message (Publikasikan pesan SMS), untuk Message type (Jenis pesan), pilih salah satu jenis berikut:

   - 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 menetapkan jenis pesan default tingkat akun dari bagian Text messaging preferences (Preferensi olahpesan teks) dari halaman Mobile text messaging (SMS) (Olahpesan teks seluler (SMS)).

Untuk informasi harga untuk pesan promosi dan transaksional, lihat Harga SMS Global.

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.

     Dukungan untuk ID pengirim bervariasi berdasarkan negara dan/atau wilayah. Misalnya, pesan yang dikirim ke nomor telepon US tidak akan menampilkan ID pengirim. Untuk negara dan wilayah yang mendukung ID pengirim, lihat Negara dan wilayah yang didukung.

     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 yang hanya mendukung ID pengirim, PEMBERITAHUAN akan ditampilkan.

     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 asal yang dikonfigurasi di Akun AWS Anda untuk negara tujuan. Nomor originasi dapat berupa nomor 10DLC, nomor bebas pulsa, kode person-to-person panjang, atau kode pendek. Untuk informasi selengkapnya, lihat Identitas asal untuk pesan SMS.

     Jika Anda tidak menentukan nomor originasi, Amazon SNS memilih nomor originasi yang akan digunakan untuk pesan teks SMS, berdasarkan konfigurasi Anda. Akun AWS
9. Jika Anda mengirim pesan SMS ke penerima di India, perluas Country-specific attributes (Atribut khusus negara), dan tentukan atribut berikut:

- Entity ID (ID entitas) – ID entitas atau ID entitas utama (principal entity/PE) untuk mengirim pesan SMS 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 di TRAI.
- Template ID (ID templat) – ID templat untuk mengirim pesan SMS ke penerima di India. ID ini adalah string unik yang disediakan TRAI dari 1-50 karakter yang mengidentifikasi template yang Anda daftarkan dengan TRAI. ID templat harus dikaitkan dengan ID pengirim yang Anda tentukan untuk pesan.

Untuk informasi lebih lanjut tentang pengiriman pesan SMS ke penerima di India, lihat Persyaratan pendaftaran ID pengirim untuk India.

10. Pilih Publish message (Publikasikan Pesan).

> **Tip**
>
> Untuk mengirim pesan SMS dari nomor asal, Anda juga dapat memilih Origination numbers (Nomor asal) di panel navigasi konsol Amazon SNS. Pilih nomor asal yang mencakup SMS di kolom Capabilities (Kemampuan), dan kemudian pilih Publish text message (Publikasikan pesan teks).

Mengirim pesan (AWS SDK)

Untuk mengirim pesan SMS menggunakan salah satu AWS SDK, gunakan operasi API di SDK yang sesuai dengan `Publish` permintaan di Amazon SNS API. Dengan permintaan ini, Anda dapat mengirim pesan SMS 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.

Dukungan untuk ID pengirim 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 ID pengirim, lihat Negara dan wilayah yang didukung.

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 abjad,PEMBERITAHUAN muncul 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 nomor 10DLC, nomor bebas pulsa, kode panjang person-to-person (P2P), atau kode pendek. Untuk informasi selengkapnya, lihat Nomor asal.

Jika Anda tidak menentukan nomor originasi, Amazon SNS memilih nomor originasi berdasarkan konfigurasi akun Anda. AWS

`AWS.SNS.SMS.MaxPrice`

Harga maksimum dalam USD yang akan Anda gunakan untuk mengirim pesan SMS. Jika Amazon SNS menentukan bahwa mengirim pesan akan dikenakan biaya yang melebihi harga maksimum Anda, ia tidak akan mengirim pesan.

Atribut ini tidak berpengaruh jika biaya month-to-date SMS Anda telah melebihi kuota yang ditetapkan untuk atribut tersebut`MonthlySpendLimit`. Anda dapat mengatur atribut `MonthlySpendLimit` menggunakan permintaan `SetSMSAttributes`.

Jika Anda mengirim pesan ke topik Amazon SNS, harga maksimum berlaku untuk setiap pengiriman pesan ke setiap nomor telepon yang berlangganan topik.

`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 pesan SMS ke penerima di India.

Ini merupakan ID entitas atau ID entitas utama (principal entity/PE) Anda untuk mengirim pesan SMS 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 di TRAI.

`AWS.MM.SMS.TemplateId`

Atribut ini hanya diperlukan untuk mengirim pesan SMS ke penerima di India.

Ini adalah templat Anda untuk mengirim pesan SMS ke penerima di India. ID ini adalah string unik yang disediakan TRAI dari 1-50 karakter yang mengidentifikasi template yang Anda daftarkan dengan TRAI. ID templat harus dikaitkan dengan ID pengirim yang Anda tentukan untuk pesan.

Mengirim pesan

Contoh kode berikut menunjukkan cara mempublikasikan pesan SMS menggunakan Amazon SNS.

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for C++ API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Menerbitkan di AWS SDK untuk referensi API Kotlin.

PHP

SDK for PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for PHP API.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Menerbitkan di AWS SDK for Python (Boto3) Referensi API.

Memantau aktivitas SMS

Dengan memantau aktivitas SMS Anda, Anda dapat melacak nomor telepon tujuan, pengiriman yang sukses atau gagal, alasan kegagalan, biaya, dan informasi lainnya. Amazon SNS membantu dengan meringkas statistik di konsol tersebut, mengirim informasi ke Amazon CloudWatch, dan mengirim laporan penggunaan SMS harian ke bucket Amazon S3 yang Anda tentukan.

Topik

- Melihat statistik pengiriman SMS

- Melihat AmazonCloudWatchmetrik dan log untuk pengiriman SMS
- Melihat laporan penggunaan SMS harian

Melihat statistik pengiriman SMS

Anda dapat menggunakan konsol Amazon SNS untuk melihat statistik tentang pengiriman SMS terbaru Anda.

1. Masuk ke konsol Amazon SNS.
2. Di menu konsol tersebut, atur pemilih wilayah ke wilayah yang mendukung olahpesan SMS.
3. Di panel navigasi, pilih Text messaging (SMS) (Olahpesan teks (SMS)).
4. Di halaman Text messaging (SMS) (Olahpesan teks (SMS)), di bagian Account stats (Statistik akun), lihat grafik untuk pengiriman pesan SMS transaksional dan promosi 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 Melihat laporan penggunaan SMS harian.

Melihat AmazonCloudWatchmetrik dan log untuk pengiriman SMS

Anda dapat menggunakan AmazonCloudWatchdan AmazonCloudWatchLogs untuk memantau pengiriman pesan SMS Anda.

Topik

- Melihat AmazonCloudWatchmetrik
- MelihatCloudWatchBeberapa catatan
- Contoh log untuk pengiriman SMS yang berhasil
- Contoh log untuk pengiriman SMS yang gagal
- Alasan kegagalan pengiriman SMS

Melihat AmazonCloudWatchmetrik

Amazon SNS secara otomatis mengumpulkan metrik tentang pengiriman pesan SMS Anda dan mendorongnya ke AmazonCloudWatch. Anda dapat menggunakanCloudWatchuntuk memantau metrik ini dan membuat alarm untuk memberi tahu Anda bila metrik melintasi ambang batas. Misalnya, Anda dapat memantauCloudWatchmetrik untuk mempelajari tingkat pengiriman SMS Anda danmonth-to-dateBiaya SMS.

Untuk informasi tentang pemantauanCloudWatchmetrik, pengaturanCloudWatchalarm, dan jenis metrik yang tersedia, lihatMemantau topik Amazon SNS menggunakan CloudWatch.

MelihatCloudWatchBeberapa catatan

Anda dapat mengumpulkan informasi tentang pengiriman pesan SMS yang berhasil dengan mengaktifkan Amazon SNS ke AmazonCloudWatchLog. Untuk setiap pesan SMS yang Anda kirim, Amazon SNS menulis log yang mencakup harga pesan, status keberhasilan atau kegagalan, alasan kegagalan (jika pesan gagal), waktu tunggu pesan, dan informasi lainnya.

Cara mengaktifkan dan melihatCloudWatchLogs untuk pesan SMS Anda

1. Masuk ke konsol Amazon SNS.
2. Di menu konsol tersebut, atur pemilih wilayah ke wilayah yang mendukung olahpesan SMS.
3. Di panel navigasi, pilih Text messaging (SMS) (Olahpesan teks (SMS)).
4. Di halaman Mobile text messaging (SMS) (Olahpesan teks seluler (SMS)), di bagian Text messaging preferences(Preferensi pesan teks), pilih Edit.
5. Di halaman berikutnya, perluas bagian Delivery status logging (Pencatatan status pengiriman).
6. UntukTingkat sampel keberhasilan, tentukan persentase pengiriman SMS yang berhasil yang akan ditulis Amazon SNSCloudWatchLog. Misalnya:

   - 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 Allow (Izinkan) untuk memberi Amazon SNS akses tulis ke sumber daya akun Anda.
- Untuk menggunakan peran layanan yang ada, pilih Use existing service role (Gunakan peran layanan yang ada) lalu tempelkan nama ARN di kotak IAM role for successful and failed deliveries (Peran IAM untuk pengiriman yang berhasil dan gagal).

  Peran layanan yang Anda tentukan harus mengizinkan akses tulis ke sumber daya akun Anda. Untuk informasi selengkapnya tentang cara membuat IAM role, lihat Membuat peran untuk layanan AWS dalam Panduan Pengguna IAM.

8. Pilih Save changes (Simpan perubahan).
9. Kembali ke halaman Mobile text messaging (SMS) (Pesan teks seluler (SMS)), masuk ke bagian Delivery status logs (Log status pengiriman) untuk melihat log yang tersedia.

> **Note**
>
> Tergantung pada operator nomor telepon tujuan, perlu waktu hingga 72 jam agar log pengiriman muncul di konsol Amazon SNS.

Contoh log untuk pengiriman SMS yang berhasil

Log status pengiriman untuk pengiriman SMS 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 pengiriman SMS yang gagal

Log status pengiriman untuk pengiriman SMS 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"
}
```

Alasan kegagalan pengiriman SMS

Alasan kegagalan diberikan dengan atribut `providerResponse`. Pesan SMS mungkin gagal dikirim karena alasan-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
- Telepon telah memblokir 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

Melihat laporan penggunaan SMS harian

Anda dapat memantau pengiriman SMS Anda dengan berlangganan laporan penggunaan harian dari Amazon SNS. Untuk setiap hari saat Anda mengirim setidaknya satu pesan SMS, Amazon SNS mengirimkan laporan penggunaan dalam file CSV ke bucket Amazon S3 tertentu. Diperlukan waktu 24 jam hingga laporan penggunaan SMS tersedia dalam bucket S3.

Topik

- Informasi laporan penggunaan harian
- Berlangganan laporan penggunaan harian

Informasi laporan penggunaan harian

Laporan penggunaan mencakup informasi berikut untuk setiap pesan SMS 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 penerbitan untuk pesan (dalam UTC)
- ID Pesan
- Nomor telepon tujuan
- Jenis pesan
- Status pengiriman
- Harga pesan (dalam USD)
- Jumlah bagian (pesan dibagi menjadi beberapa bagian jika terlalu panjang untuk satu pesan)
- Jumlah total bagian

> **Note**
>
> Jika Amazon SNS tidak menerima jumlah bagian, kami mengatur 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

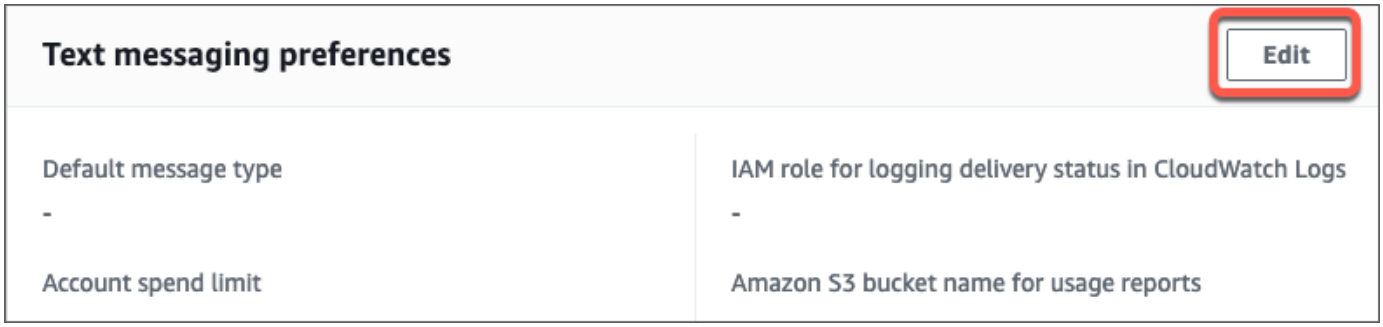
1. Dari Akun AWS yang mengirim pesan SMS, masuk ke konsol Amazon S3.
2. Pilih Create Bucket (Buat Bucket).
3. Untuk Bucket Name (Nama Bucket), sebaiknya masukkan nama yang unik untuk akun dan organisasi Anda. Misalnya, gunakan pola `<my-bucket-prefix>-<account_id>-<org-id>`.

   Untuk informasi tentang konvensi dan pembatasan nama bucket, lihatAturan penamaan bucketdiPanduan Pengguna Amazon Simple Storage Service.
4. Pilih Create (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 (Editor Kebijakan Bucket), berikan kebijakan yang mengizinkan perwakilan layanan Amazon SNS untuk menulis ke bucket Anda. Sebagai contoh, lihat Contoh kebijakan bucket.

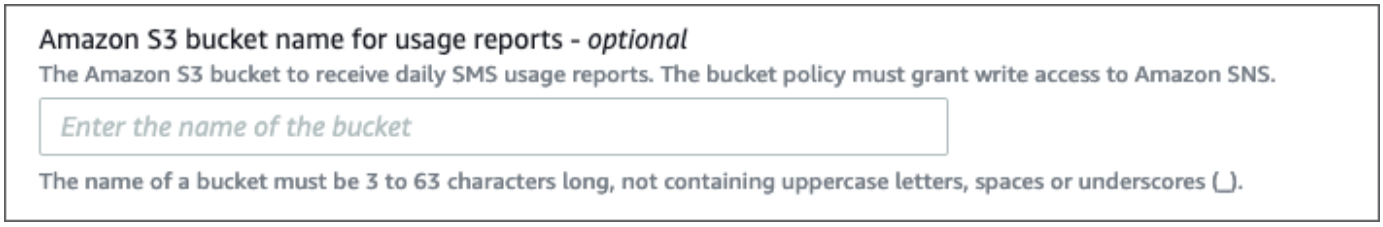
   Jika Anda menggunakan kebijakan contoh, ingat untuk mengganti *`my-s3-bucket`* dengan nama bucket yang Anda pilih di Langkah 3.
8. Pilih Save (Simpan).

Cara berlangganan laporan penggunaan harian

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi, pilih Text messaging (SMS) (Olahpesan teks (SMS)).
3. Di halaman Text messaging (SMS) (Olahpesan teks (SMS)), di bagian Text messaging preferences (Preferensi olahpesan 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 Save changes (Simpan perubahan).

Contoh kebijakan bucket

Kebijakan berikut mengizinkan perwakilan layanan Amazon SNS untuk melakukan tindakan `s3:PutObject`, `s3:GetBucketLocation`, dan `s3:ListBucket`.

AWSmenyediakan alat untuk semua layanan dengan prinsipal layanan yang telah diberikan akses ke sumber daya di akun Anda. Saat prinsipal dalam pernyataan kebijakan bucket Amazon S3 adalahAWSPrinsipal layanan, Anda dapat menggunakan`aws:SourceArn`atau`aws:SourceAccount`kunci kondisi global untuk melindungi terhadapmasalah deputi. Untuk membatasi wilayah dan akun mana bucket dapat menerima laporan penggunaan harian, gunakan`aws:SourceArn`seperti yang ditunjukkan dalam 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 ARN sumber daya, gunakan`aws:SourceAccount`.

Gunakan contoh berikut yang mencakup perlindungan deputy saat Anda membuat bucket Amazon S3 untuk menerima laporan penggunaan SMS 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:::my-s3-bucket/*",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   },
   {
    "Sid": "AllowGetBucketLocation",
    "Effect": "Allow",
    "Principal": {
     "Service": "sns.amazonaws.com"
    },
    "Action": "s3:GetBucketLocation",
    "Resource": "arn:aws:s3:::my-s3-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   },
   {
    "Sid": "AllowListBucket",
    "Effect": "Allow",
    "Principal": {
     "Service": "sns.amazonaws.com"
    },
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::my-s3-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
```

```
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  }
 ]
}
```

> **Note**
>
> 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 dimiliki oleh Akun AWS lainnya, lihat Bagaimana cara menyalin objek S3 dari Akun AWS lain?.

Contoh laporan penggunaan harian

Setelah Anda berlangganan laporan penggunaan harian, setiap hari, Amazon SNS menempatkan file CSV 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 untuk 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
```

Mengelola nomor telepon dan langganan SMS

Amazon SNS menyediakan beberapa pilihan untuk mengelola siapa yang menerima pesan SMS dari akun Anda. Dengan frekuensi terbatas, Anda dapat memilih nomor telepon yang telah memilih untuk tidak menerima pesan SMS dari akun Anda. Untuk berhenti mengirim pesan ke langganan SMS, Anda dapat menghapus langganan atau topik yang menerbitkan ke mereka.

Topik

- Memilih untuk tidak menerima pesan SMS
- Mengelola nomor telepon dan langganan (konsol)
- Mengelola nomor telepon dan langganan (SDK AWS)

Memilih untuk tidak menerima pesan SMS

Jika diwajibkan oleh undang-undang dan peraturan setempat (seperti US dan Kanada), penerima SMS dapat menggunakan perangkat mereka untuk memilih tidak menerima SMS dengan membalas pesan dengan salah satu dari berikut ini:

- ARRET (Perancis)
- CANCEL (BATALKAN)
- END (AKHIRI)
- OPT-OUT (memilih tidak menerima SMS)
- OPTOUT (memilih tidak menerima SMS)
- QUIT (BERHENTI)
- REMOVE (HAPUS)
- STOP (BERHENTI)
- TD
- UNSUBSCRIBE (BERHENTI BERLANGGANAN)

Untuk memilih tidak menerima SMS, penerima harus membalaske nomor asal yang sama dengan yang digunakan Amazon SNS untuk mengirim pesan. Setelah memilih keluar, penerima tidak akan lagi menerima pesan SMS yang dikirim dari Anda Akun AWS kecuali Anda memilih nomor telepon.

Jika nomor telepon berlangganan topik Amazon SNS, memilih tidak menerima SMS tidak akan menghapus langganan, tetapi pesan SMS akan gagal dikirim ke langganan, itu kecuali Anda memilih nomor telepon untuk menerima SMS kembali.

Mengelola nomor telepon dan langganan (konsol)

Anda dapat menggunakan konsol Amazon SNS untuk mengontrol nomor telepon mana yang menerima pesan SMS dari akun Anda.

Memilih nomor telepon yang telah memilih untuk tidak menerima SMS

Anda dapat melihat nomor telepon mana yang telah memilih untuk tidak menerima pesan SMS dari akun Anda, dan Anda dapat memilih nomor telepon ini kembali untuk melanjutkan pengiriman pesan kepada mereka.

Anda hanya dapat memilih nomor telepon kembali sekali setiap 30 hari.

1. Masuk ke konsol Amazon SNS.
2. Di menu konsol tersebut, atur pemilih wilayah ke wilayah yang mendukung olahpesan SMS.
3. Di panel navigasi, pilih Text messaging (SMS) (Olahpesan teks (SMS)).
4. Di halaman Text messaging (SMS) (Olahpesan teks (SMS)), pilih View opted out phone numbers (Lihat nomor telepon yang memilih untuk tidak menerima SMS. Halaman Opted out phone numbers (Nomor telepon yang memilih untuk tidak menerima SMS) menampilkan nomor telepon yang memilih untuk tidak menerima SMS.
5. Pilih kotak centang untuk nomor telepon yang ingin Anda pilih untuk kembali menerima SMS, lalu pilih Opt in (Pilih kembali). Nomor telepon tidak lagi memilih tidak menerima SMS dan akan menerima pesan SMS yang Anda kirim ke sana.

Menghapus langganan SMS

Menghapus langganan SMS untuk berhenti mengirim pesan SMS ke nomor telepon tersebut saat Anda menerbitkan 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

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 (SDK AWS)

Anda dapat menggunakan AWS SDK untuk membuat permintaan terprogram ke Amazon SNS dan mengelola nomor telepon mana yang dapat menerima pesan SMS dari akun Anda.

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

Melihat semua nomor telepon yang memilih untuk tidak menerima SMS

Untuk melihat semua nomor telepon yang memilih untuk tidak menerima SMS, kirimkan permintaan `ListPhoneNumbersOptedOut` dengan API Amazon SNS.

Contoh kode berikut menunjukkan cara menggunakan`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Untuk membuat daftar opt-out pesan SMS

`list-phone-numbers-opted-out`Contoh berikut mencantumkan nomor telepon yang dipilih untuk tidak menerima pesan SMS.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
```

```
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Untuk detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS SDK for Java 2.x API.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS SDK for PHP API.

Memeriksa jika nomor telepon telah memilih untuk tidak menerima SMS

Untuk memeriksa jika nomor telepon memilih untuk tidak menerima SMS, kirimkan permintaan `CheckIfPhoneNumberIsOptedOut` dengan API Amazon SNS.

Contoh kode berikut menunjukkan cara menggunakan`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

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

```
    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 detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk memeriksa pesan SMS opt-out untuk nomor telepon

`check-if-phone-number-is-opted-out`Contoh berikut memeriksa apakah nomor telepon yang ditentukan dipilih untuk tidak menerima pesan SMS dari AWS akun saat ini.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Untuk detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for JavaScript API.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for PHP API.

Memilih nomor telepon yang telah memilih untuk tidak menerima SMS

Untuk memilih kembali nomor telepon agar menerima SMS, kirimkan permintaan `OptInPhoneNumber` dengan API Amazon SNS.

Anda hanya dapat memilih nomor telepon kembali sekali setiap 30 hari.

Menghapus langganan SMS

Untuk menghapus langganan SMS dari topik Amazon SNS, dapatkan ARN langganan dengan mengirimkan permintaan `ListSubscriptions` dengan API Amazon SNS, 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for C++ API.

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 detail API, lihat Berhenti berlangganan di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat Berhenti berlangganan di Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat Berhenti berlangganan di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat Berhenti berlangganan di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat Berhenti berlangganan di AWS SDK untuk referensi SAP ABAP API.

Menghapus topik

Untuk menghapus topik dan semua langganannya, dapatkan ARN topik dengan mengirimkan permintaan `ListTopics` dengan API Amazon SNS, 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

Hapus topik berdasarkan topiknya ARN.

```
    /// <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 detail API, lihat DeleteTopicdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat DeleteTopicdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk menghapus topik SNS

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

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

Perintah ini tidak menghasilkan output.

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

Go

SDK untuk Go V2

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

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



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

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat DeleteTopicdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DeleteTopicdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat DeleteTopicdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat DeleteTopicdi Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat DeleteTopicdi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat DeleteTopicdi AWS SDK untuk referensi SAP ABAP API.

Negara dan wilayah yang didukung

> **Important**
>
> Efektif 31 Agustus 2023, nomor khusus seperti nomor atau 10DLC nomor bebas pulsa diperlukan untuk mengirim pesan SMS ke Amerika Serikat dan wilayahnya (Guam, Puerto Riko, Kepulauan Samoa Amerika, dan Kepulauan Virgin). US

Saat ini, Amazon SNS mendukung pesan SMS di Wilayah berikut: AWS

Nama wilayah	Wilayah	Titik Akhir	Protokol
US East (Ohio)	us-east-2	sns.us-east-2.amazonaws.com	HTTP dan HTTPS
US East (N. Virginia)	us-east-1	sns.us-east-1.amazonaws.com	HTTP dan HTTPS
US West (N. California)	us-west-1	sns.us-west-1.amazonaws.com	HTTP dan HTTPS
US West (Oregon)	us-west-2	sns.us-west-2.amazonaws.com	HTTP dan HTTPS
Africa (Cape Town)	af-south-1	sns.af-south-1.amazonaws.com	HTTP dan HTTPS
Asia Pasifik (Hyderabad)	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP dan HTTPS
Asia Pasifik (Jakarta)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP dan HTTPS

Nama wilayah	Wilayah	Titik Akhir	Protokol
Asia Pasifik (Melbourne)	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP dan HTTPS
Asia Pacific (Mumbai)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP dan HTTPS
Asia Pacific (Osaka)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP dan HTTPS
Asia Pacific (Singapore)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP dan HTTPS
Asia Pacific (Sydney)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP dan HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP dan HTTPS
Canada (Central)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP dan HTTPS
Europe (Frankfurt)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP dan HTTPS
Europe (Ireland)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP dan HTTPS
Europe (London)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP dan HTTPS
Europe (Milan)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP dan HTTPS
Europe (Paris)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP dan HTTPS
Eropa (Spanyol)	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP and HTTPS

Nama wilayah	Wilayah	Titik Akhir	Protokol
Europe (Stockholm)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP dan HTTPS
Eropa (Zürich)	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP dan HTTPS
Israel (Tel Aviv)	il-central-1	sns.il-central-1.amazonaws.com	HTTP dan HTTPS
Middle East (Bahrain)	me-south-1	sns.me-south-1.amazonaws.com	HTTP dan HTTPS
Timur Tengah (UEA)	me-central-1	sns.me-central-1.amazonaws.com	HTTP dan HTTPS
South America (São Paulo)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP dan HTTPS
AWS GovCloud (AS-Timur)	us-gov-east-1	sns. us-gov-east-1.amazonaws.com	HTTP dan HTTPS
AWS GovCloud (AS-Barat)	us-gov-west-1	sns. us-gov-west-1.amazonaws.com	HTTP dan HTTPS

Anda dapat menggunakan Amazon SNS untuk mengirim pesan SMS ke negara dan wilayah berikut:

> **Note**
>
> Menggunakan ID Pengirim di negara-negara yang didukung dapat meningkatkan pengiriman SMS.

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
A						
Afghanistan	AF	93	Tidak	Tidak	Ya	Tidak
Albania	AL	355	Tidak	Tidak	Ya	Tidak
Aljazair	DZ	213	Tidak	Tidak	Ya	Tidak
Andorra	AD	376	Tidak	Tidak	Ya	Tidak
Anguilla	AI	1-264	Tidak	Tidak	Ya	Tidak
Antigua dan Barbuda	AG	1-268	Tidak	Tidak	Ya	Tidak
Argentina	AR	54	Ya	Tidak	Tidak	Tidak
Armenia	AM	374	Tidak	Tidak	Ya	Tidak
Aruba	AW	297	Tidak	Tidak	Ya	Tidak
Australia	AU	61	Tidak	Ya	Pendaftaran diperlukan [1]	Ya
Austria	AT	43	Ya	Ya	Ya	Ya
Azerbaijan	AZ	994	Tidak	Tidak	Ya	Tidak
B						
Bahama	BS	1-242	Tidak	Tidak	Tidak	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Bahrain	BH	973	Tidak	Tidak	Ya	Tidak
Bangladesh	BD	880	Tidak	Tidak	Ya	Tidak
Barbados	BB	1-246	Tidak	Tidak	Ya	Tidak
Belarus	BY	375	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
Belgium	BE	32	Tidak	Ya	Tidak	Ya
Belize	BZ	501	Tidak	Tidak	Ya	Tidak
Bermuda	BM	1-441	Tidak	Tidak	Ya	Tidak
Bhutan	BT	975	Tidak	Tidak	Ya	Tidak
Bolivia	BO	591	Tidak	Tidak	Ya	Tidak
Bosnia dan Herzegovina	BA	387	Tidak	Tidak	Ya	Tidak
Botswana	BW	267	Tidak	Tidak	Ya	Tidak
Brazil	BR	55	Ya	Tidak	Tidak	Ya
Brunei	BN	673	Tidak	Tidak	Ya	Tidak
Bulgaria	BG	359	Ya	Tidak	Ya	Ya
Burkina Faso	BF	226	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Burundi	BL	257	Tidak	Tidak	Ya	Tidak
C						
Kamboja	KH	855	Tidak	Tidak	Ya	Tidak
Kamerun	CM	237	Tidak	Tidak	Ya	Tidak
Kanada	CA	1	Ya	Ya	Tidak	Ya
Tanjung Verde	CV	238	Tidak	Tidak	Ya	Tidak
Kepulauan Cayman	KY	1-345	Tidak	Tidak	Tidak	Tidak
Republik Afrika Tengah	CF	236	Tidak	Tidak	Ya	Tidak
Chad	TD	235	Tidak	Tidak	Ya	Tidak
Chili	CL	56	Ya	Ya	Tidak	Ya
China	CN	86	Ya	Tidak	Tidak [2]	Ya
Kolombia	CO	57	Ya	Ya	Tidak	Ya
Komoro	KM	269	Tidak	Tidak	Ya	Tidak
Kepulauan Cook	CK	682	Tidak	Tidak	Ya	Ya
Kosta Rika	CR	506	Tidak	Tidak	Tidak	Tidak
Croatia	HR	385	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Cyprus	CY	357	Tidak	Tidak	Ya	Tidak
Ceko (Republik Ceko)	CZ	420	Tidak	Ya	Ya	Ya
D						
Republik Demokrasi Kongo	CD	243	Tidak	Tidak	Ya	Tidak
Denmark	DK	45	Ya	Ya	Ya	Ya
Djibouti	DJ	253	Tidak	Tidak	Ya	Tidak
Dominika	DM	1-767	Tidak	Tidak	Ya	Tidak
Republik Dominika	DO	1-809, 1-829, 1-849	Ya	Tidak	Tidak	Ya
E						
Ekuador	EC	593	Ya	Tidak	Tidak	Ya
Mesir	EG	20	Ya	Tidak	Pendaftaran diperlukan [1]	Ya
El Salvador	SV	503	Tidak	Tidak	Tidak	Tidak
Guinea Khatulistiwa	GQ	240	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Eritrea	ER	291	Tidak	Tidak	Ya	Tidak
Estonia	EE	372	Tidak	Ya	Ya	Ya
Etiopia	ET	251	Tidak	Tidak	Ya	Tidak
F						
Kepulauan Faroe	FO	298	Tidak	Tidak	Ya	Tidak
Fiji	FJ	679	Tidak	Tidak	Ya	Tidak
Finland	FI	358	Ya	Ya	Ya	Ya
Perancis	FR	33	Ya	Tidak	Ya	Ya
Guyana Prancis	GF	594	Tidak	Tidak	Ya	Tidak
Polinesia Prancis	PF	689	Tidak	Tidak	Ya	Tidak
G						
Gabon	GA	241	Tidak	Tidak	Ya	Tidak
Gambia	GM	220	Tidak	Tidak	Ya	Tidak
Georgia	GE	995	Tidak	Tidak	Ya	Tidak
Germany	DE	49	Ya	Ya	Ya	Ya
Ghana	GH	233	Tidak	Tidak	Ya	Tidak
Gibraltar	GI	350	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Greece	GR	30	Tidak	Tidak	Ya	Tidak
Greenland	GL	299	Tidak	Tidak	Ya	Tidak
Grenada	GD	1-473	Tidak	Tidak	Ya	Tidak
Guadeloupe	GP	590	Tidak	Tidak	Ya	Tidak
Guam	GU	1-671	Tidak	Tidak	Tidak	Ya
Guatemala	GT	502	Tidak	Tidak	Tidak	Tidak
Guernsey	GG	44-1481	Tidak	Tidak	Ya	Tidak
Guinea	GN	224	Tidak	Tidak	Ya	Tidak
Guinea-Bissau	GW	245	Tidak	Tidak	Ya	N/A
Guyana	GY	592	Tidak	Tidak	Ya	Tidak
H						
Haiti	H	509	Tidak	Tidak	Ya	Tidak
Honduras	HN	504	Tidak	Tidak	Ya	Tidak
Hong Kong	HK	852	Tidak	Ya	Ya	Ya
Hungaria	HU	36	Tidak	Ya	Tidak	Ya
Aku						
Islandia	IS	354	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
India	IN	91	Ya	Ya [4]	Pendaftaran diperlukan [3]	Ya
Indonesia	ID	62	Tidak	Tidak	Ya	Tidak
Irak	IQ	964	Tidak	Tidak	Ya	Tidak
Ireland	IE	353	Tidak	Ya	Ya	Ya
Pulau Man	IM	44-1624	Tidak	Tidak	Ya	Tidak
Israel	IL	972	Tidak	Ya	Ya	Ya
Italia	IT	39	Ya	Ya	Ya	Ya
Pantai Gading	CI	225	Tidak	Tidak	Ya	Tidak
J						
Jamaika	JM	1-876	Tidak	Tidak	Ya	Tidak
Jepang	JP	81	Ya	Ya	Ya	Ya
Jersey	JE	44-1534	Tidak	Ya	Ya	Ya
Yordania	JO	962	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
K						
Kazakstan	KZ	7	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Kenya	KE	254	Tidak	Tidak	Ya	Tidak
Kosovo	XK	383	Tidak	Tidak	Ya	Tidak
Kuwait	KW	965	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
Kirgistan	KG	996	Tidak	Tidak	Ya	Tidak
L						
Laos	LA	856	Tidak	Tidak	Ya	Tidak
Latvia	LV	371	Tidak	Tidak	Ya	Tidak
Libanon	LB	961	Tidak	Tidak	Ya	Tidak
Lesotho	LS	266	Tidak	Tidak	Ya	Tidak
Liberia	LR	231	Tidak	Ya	Tidak	
Libya	LY	218	Tidak	Tidak	Ya	Tidak
Liechtenstein	LI	423	Tidak	Tidak	Ya	Tidak
Lithuania	LT	370	Tidak	Ya	Ya	Ya
Luksemburg	LU	352	Tidak	Ya	Ya	Ya
M						
Macau	MO	853	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Makedonia	MK	389	Tidak	Tidak	Ya	Tidak
Madagaskar	MG	261	Tidak	Tidak	Ya	Tidak
Malawi	MW	265	Tidak	Tidak	Ya	Tidak
Malaysia	MY	60	Ya	Tidak	Tidak	Ya
Maladewa	MV	960	Tidak	Tidak	Ya	Tidak
Mali	ML	223	Tidak	Tidak	Ya	Tidak
Malta	MT	356	Tidak	Tidak	Ya	Tidak
Kepulauan Marshall,	MH	692	Tidak	Tidak	Tidak	Tidak
Martinik	MQ	596	Tidak	Tidak	Ya	Tidak
Mauritania	MR	222	Tidak	Tidak	Ya	Tidak
Mauritius	MU	230	Tidak	Tidak	Ya	Tidak
Mayotte	YT	262	Tidak	Tidak	Ya	Tidak
Meksiko	MX	52	Ya	Tidak	Tidak	Ya
Mikronesia (Negara Federasi)	FM	691	Tidak	Tidak	Tidak	Tidak
Moldova	MD	373	Tidak	Tidak	Ya	Tidak
Monako	MC	377	Tidak	Tidak	Tidak	Tidak
Mongolia	MN	976	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Montenegro	ME	382	Tidak	Tidak	Ya	Tidak
Montserrat	MS	1-664	Tidak	Tidak	Ya	Tidak
Maroko	MA	212	Ya	Tidak	Ya	Ya
Mozambik	MZ	258	Tidak	Tidak	Tidak	Tidak
Myanmar	MM	95	Tidak	Ya	Ya	Ya
N						
Namibia	NA	264	Tidak	Tidak	Ya	Tidak
Nepal	NP	977	Tidak	Tidak	Ya	Tidak
Netherlands	NL	31	Ya	Ya	Ya	Ya
Antillen Belanda	AN	599	Tidak	Tidak	Ya	Tidak
Kaledonia Baru	NC	687	Tidak	Tidak	Ya	Tidak
Selandia Baru [6]	NZ	64	Ya	Tidak	Tidak	Ya
Nikaragua	NI	505	Tidak	Tidak	Tidak	Tidak
Niger	NE	227	Tidak	Tidak	Ya	Tidak
Nigeria	NG	234	Tidak	Tidak	Ya	Tidak
Niue	NU	683	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Norwegia	NO	47	Tidak	Ya	Ya	Ya
O						
Oman	OM	968	Tidak	Tidak	Tidak	N/A
P						
Pakistan	PK	92	Tidak	Tidak	Ya	N/A
Palestina	PS	970	Tidak	Tidak	Ya	Tidak
Panama	PA	507	Tidak	Tidak	Ya	Tidak
Papua Nugini	PG	675	Tidak	Tidak	Ya	Tidak
Paraguay	PY	595	Tidak	Tidak	Tidak	Tidak
Peru	PE	51	Ya	Tidak	Tidak	Ya
Filipina	PH	63	Tidak	Ya	Pendaftaran diperlukan [1]	Ya
Polandia	PL	48	Tidak	Ya	Ya	Ya
Portugal	PT	351	Tidak	Ya	Ya	Ya
Puerto Riko	PR	1-797, 1-939	Tidak	Tidak	Tidak	Ya
Q						

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Qatar	QA	974	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
R						
Republik Kongo	CG	242	Tidak	Tidak	Tidak	Tidak
Réunion (Prancis)	RE	262	Tidak	Tidak	Ya	Tidak
Romania	RO	40	Tidak	Ya	Ya	Ya
Rusia	RU	7	Ya	Tidak	Pendaftaran diperlukan [1]	Ya
Rwanda	RW	250	Tidak	Tidak	Ya	Tidak
S						
Saint Kitts dan Nevis	KN	1-869	Tidak	Tidak	Tidak	Tidak
Saint Lucia	LC	1-758	Tidak	Tidak	Tidak	Tidak
Samoa	WS	685	Tidak	Tidak	Ya	Tidak
San Marino	SM	378	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Sao Tomé dan Principe	ST	239	Tidak	Tidak	Ya	Tidak
Arab Saudi	SA	966	Tidak	Ya [4]	Pendaftaran diperlukan [1]	Tidak
Senegal	SN	221	Tidak	Tidak	Ya	Tidak
Serbia	RS	381	Tidak	Tidak	Ya	Tidak
Seychelles	SC	248	Tidak	Tidak	Ya	Tidak
Sierra Leone	SL	232	Tidak	Tidak	Ya	Tidak
Singapura	SG	65	Ya	Ya	Ya [5]	Ya
Slovakia	SK	421	Tidak	Ya	Ya	Tidak
Slovenia	SI	386	Tidak	Tidak	Ya	Tidak
Kepulauan Solomon	SB	677	Tidak	Tidak	Ya	Tidak
Somalia	SO	252	Tidak	Tidak	Ya	Tidak
Afrika Selatan	ZA	27	Ya	Ya	Tidak	Ya
Korea Selatan	KR	82	Tidak	Tidak	Tidak	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Sudan Selatan	SS	211	Tidak	Tidak	Ya	Tidak
Spain	ES	34	Ya	Ya	Ya	Ya
Sri Lanka	LK	94	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
Suriname	SR	597	Tidak	Tidak	Ya	Tidak
Swaziland	SZ	268	Tidak	Tidak	Ya	Tidak
Sweden	SE	46	Ya	Ya	Ya	Ya
Swiss	CH	41	Tidak	Ya	Ya	Ya
T						
Taiwan	TW	886	Tidak	Ya	Tidak	Ya
Tajikistan	TJ	992	Tidak	Tidak	Ya	Tidak
Tanzania	TX	255	Tidak	Tidak	Ya	Tidak
Thailand	TH	66	Tidak	Ya	Pendaftaran diperlukan [1]	Ya
Timor-Leste	TL	670	Tidak	Tidak	Ya	Tidak
Togo	TG	228	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Tonga	TO	676	Tidak	Tidak	Ya	Tidak
Trinidad dan Tobago	TT	1-868	Tidak	Tidak	Ya	Tidak
Tunisia	TN	216	Tidak	Tidak	Ya	Tidak
Turki	TR	90	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
Turkmenistan	TM	993	Tidak	Tidak	Tidak	Tidak
Kepulauan Turks dan Caicos	TC	1-649	Tidak	Tidak	Ya	Tidak
Tuvalu	TC	688	Tidak	Tidak	Ya	Tidak
U						
Uganda	UG	256	Tidak	Tidak	Ya	Tidak
Ukraina	UA	380	Tidak	Ya	Ya	Ya
Uni Emirat Arab (UEA)	AE	971	Ya	Ya	Pendaftaran diperlukan [1]	Ya
Britania Raya	GB	44	Ya	Ya	Ya	Ya

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Amerika Serikat	US	1	Ya	Ya	Tidak	Ya
Uruguay	UY	598	Ya	Tidak	Tidak	Ya
Uzbekistan	UZ	998	Tidak	Tidak	Ya	Tidak
V						
Vanuatu	VU	678	Tidak	Tidak	Ya	Tidak
Venezuela	VE	58	Tidak	Tidak	Tidak	Tidak
Vietnam	VN	84	Tidak	Tidak	Pendaftaran diperlukan [1]	Tidak
Kepulauan Virgin, Inggris	VG	1-284	Tidak	Tidak	Ya	Tidak
Kepulauan Virgin, US	VI	1-340	Tidak	Tidak	Tidak	Ya
W						
X						
Y						
Yaman	YE	967	Tidak	Tidak	Ya	Tidak
Z						
Zambia	ZM	260	Tidak	Tidak	Ya	Tidak

Negara atau wilayah	Kode ISO	Kode panggilan	Mendukung kode pendek	Mendukung kode panjang	Mendukung ID Pengirim	Mendukung SMS dua arah
Zimbabwe	ZW	263	Tidak	Tidak	Ya	Tidak

Catatan

1. Pengirim diharuskan menggunakan ID Pengirim abjad yang telah terdaftar sebelumnya. Untuk meminta ID Pengirim dari AWS Support, lihatMeminta ID pengirim untuk olahpesan SMS dengan Amazon SNS. Beberapa negara mewajibkan pengirim untuk memenuhi persyaratan tertentu atau mematuhi batasan tertentu untuk mendapatkan persetujuan. Dalam kasus ini, AWS Support dapat menghubungi Anda untuk informasi tambahan setelah Anda mengirimkan permintaan ID Pengirim Anda.
2. Pengirim harus menggunakan templat yang telah terdaftar sebelumnya untuk setiap jenis pesan yang akan dikirim. Jika pengirim tidak memenuhi persyaratan ini, pesan mereka akan diblokir. Untuk mendaftarkan templat, buka kasing SMS Amazon SNS dengan. AWS Support Saat Anda membuat kasus, berikan informasi yang sama yang akan Anda berikan untuk meminta ID pengirim. Untuk informasi selengkapnya, lihat Meminta ID pengirim untuk olahpesan SMS dengan Amazon SNS. Beberapa negara mewajibkan pengirim untuk memenuhi persyaratan tambahan tertentu atau mematuhi batasan tertentu untuk mendapatkan persetujuan. Dalam kasus ini, AWS Support mungkin meminta Anda untuk informasi tambahan.

   > **Note**
   >
   > Untuk mengirim pesan ke China, Anda harus terlebih dahulu mendaftarkan templat Anda AWS Support untuk mendapatkan persetujuan.

3. Pengirim diharuskan menggunakan ID Pengirim abjad yang telah terdaftar sebelumnya. Diperlukan langkah pendaftaran tambahan. Untuk informasi selengkapnya, lihat Persyaratan pendaftaran ID pengirim untuk India.
4. Kode panjang di negara-negara ini hanya mendukung pesan masuk. Dengan kata lain, Anda tidak dapat menggunakan kode panjang ini untuk mengirim pesan ke penerima, tetapi Anda dapat

menggunakannya untuk menerima pesan dari penerima Anda. Kode panjang ini adalah cara yang berguna untuk memungkinkan penerima memilih keluar jika Anda mengirim pesan menggunakan ID Pengirim abjad, karena ID Pengirim hanya mendukung pesan keluar.

5. Amazon SNS dapat mengirim lalu lintas SMS ke Singapura menggunakan ID Pengirim yang telah terdaftar di Singapore SMS Sender ID Registry (SSIR), registri yang dibuat oleh Info-Communications Media Development Authority (IMDA) Singapura. Untuk informasi selengkapnya tentang persyaratan untuk menggunakan ID Pengirim Singapura, lihatPersyaratan pendaftaran ID pengirim untuk Singapura.

   Anda juga dapat mengirim lalu lintas SMS di Singapura menggunakan ID Pengirim yang tidak terdaftar atau jenis identitas originasi alternatif seperti Kode Pendek atau Kode Panjang.

6. Tanpa kode pendek khusus, Amazon SNS masih mencoba mengirim pesan ke penerima Selandia Baru menggunakan kumpulan kode pendek bersama. Karena pembatasan operator lokal seputar nomor bersama, pengiriman atas nomor bersama ini dilakukan atas dasar upaya terbaik. Oleh karena itu, Amazon SNS sangat merekomendasikan pengadaan kode pendek khusus untuk semua lalu lintas yang dikirim ke Selandia Baru. Pesan yang berisi URL harus diizinkan terdaftar melalui proses kode pendek khusus. Untuk informasi lebih lanjut tentang membeli kode pendek, lihatMeminta kode pendek khusus untuk olahpesan SMS dengan Amazon SNS.

Praktik terbaik SMS

Pengguna ponsel cenderung memiliki toleransi yang sangat rendah untuk pesan SMS yang tidak diminta. Tingkat respons untuk kampanye SMS yang tidak diminta hampir selalu rendah, dan oleh karena itu laba atas investasi Anda juga akan rendah.

Selain itu, operator ponsel terus mengaudit pengirim SMS 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. Tim Amazon SNS secara rutin mengaudit kampanye SMS, dan mungkin mencegah atau memblokir kemampuan Anda untuk mengirim pesan jika tampaknya Anda mengirim pesan yang tidak diinginkan.

Akhirnya, di banyak negara, wilayah, dan yurisdiksi, ada hukuman berat untuk mengirim pesan SMS yang tidak diminta. Misalnya, di Amerika Serikat, Telephone Consumer Protection Act (TCPA)

menyatakan bahwa konsumen berhak atas kerugian sebesar $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
- Mendapatkan izin
- Jangan kirim ke daftar lama
- Audit daftar pelanggan Anda
- Simpan catatan
- Buat pesan Anda jelas, jujur, dan ringkas
- Merespons dengan tepat
- Sesuaikan pengiriman Anda berdasarkan keterlibatan
- Kirim pada waktu yang tepat
- Hindari keletihan lintas-saluran
- Gunakan kode pendek khusus
- Verifikasi nomor telepon tujuan Anda
- Desain dengan mempertimbangkan redundansi
- Batas dan batasan SMS
- Mengelola kata kunci keluar
- CreatePool
- PutKeyword
- Mengelola pengaturan nomor
- Batas karakter SMS di Amazon SNS

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, sangat penting untuk memahami hukum yang terkait dengan olahpesan SMS di setiap negara atau wilayah tempat Anda berbisnis.

Daftar berikut mencakup tautan ke undang-undang utama yang berlaku untuk komunikasi SMS di pasar utama di seluruh dunia.

- Amerika Serikat: Undang-Undang Perlindungan Konsumen Telepon tahun 1991, juga dikenal sebagai TCPA, berlaku untuk jenis pesan SMS 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 sebagai PECR, berlaku untuk jenis pesan SMS tertentu. Untuk informasi selengkapnya, lihat Apa itu PECR? di situs UK Information Commissioner's Office.
- Uni Eropa: Privacy and Electronic Communications Directive 2002, kadang-kadang dikenal sebagai EPrivacy Directive, berlaku untuk beberapa jenis pesan SMS. Untuk informasi selengkapnya, lihat dokumen hukum lengkap di situs Europa.eu.
- Kanada: Fighting Internet and Wireless Spam Act, yang lebih dikenal sebagai Hukum Anti-Spam Kanada atau CASL (Canada's Anti-Spam Law), berlaku untuk jenis pesan SMS tertentu. Untuk informasi selengkapnya, lihat dokumen hukum lengkap di situs Parliament of Canada.
- Jepang: Act on Regulation of Transmission of Specific Electronic Mail berlaku untuk beberapa jenis pesan SMS. 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 mengatasi email atau panggilan telepon yang tidak diminta, tetapi telah ditafsirkan atau diperluas untuk diterapkan ke pesan SMS juga. Negara dan wilayah lain mungkin memiliki undang-undang sendiri terkait dengan transmisi pesan SMS. 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 konten SMS 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 keikutsertaan SMS, kirimkan pesan kepada penerima yang meminta mereka untuk 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 keikutsertaan Anda. Maket atau tangkapan layar harus sangat mirip dengan alur kerja opt-in 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 pesan berulang mungkin mengatakan “satu pesan per minggu.” Kata sandi satu kali atau kasus

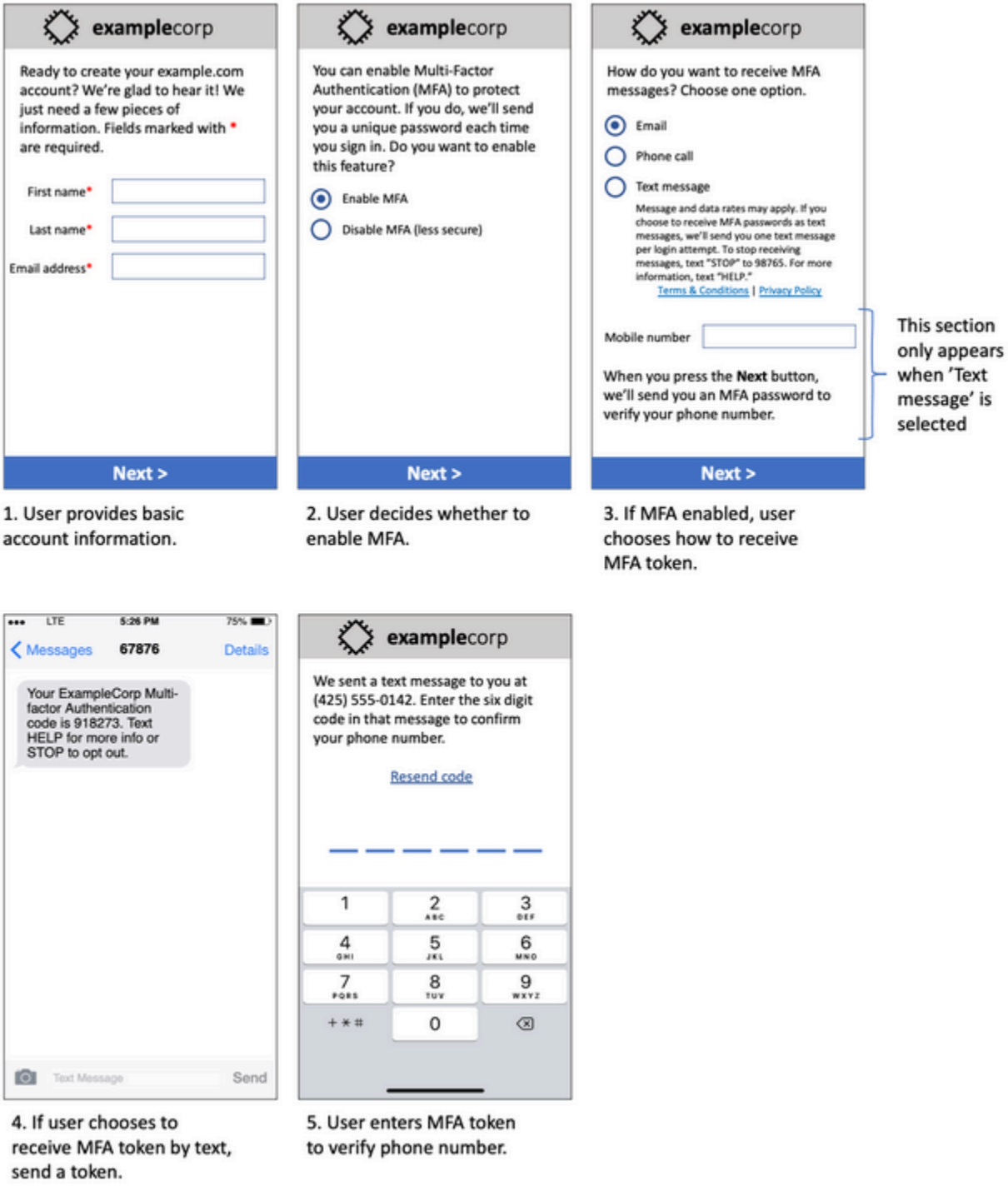
penggunaan otentikasi multi-faktor mungkin mengatakan “frekuensi pesan bervariasi” atau “satu pesan per upaya login.”

- 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 opt-in yang sesuai dan skrip verbal 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 kampanye SMS berulang, audit daftar pelanggan Anda secara berkala. 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 diminta untuk menerima pesan SMS dari Anda, dan pesan mana yang Anda kirim ke setiap pelanggan. Banyak negara dan wilayah di seluruh dunia mewajibkan pengirim SMS 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 selengkapnya tentang persyaratan penyimpanan catatan, tinjau peraturan tentang olahpesan SMS 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 akan menghubungi Anda dengan daftar informasi yang dibutuhkan oleh operator atau badan pengawas. Jika Anda tidak dapat memberikan informasi yang diperlukan, kami dapat menghentikan sementara kemampuan Anda untuk mengirim pesan SMS tambahan.

Buat pesan Anda jelas, jujur, dan ringkas

SMS adalah 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 saluran SMS, 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 pesan SMS 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 pesan SMS 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 alfabet GSM) yang dapat dimasukkan dalam pesan SMS 160 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 karakter SMS, lihatBatas karakter SMS 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 pesan SMS, URL yang sangat panjang dapat dibagi menjadi beberapa pesan. Anda harus menggunakan domain pengalihan untuk menyediakan URL yang dipersingkat. 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 dari saluran SMS 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 merespons salah satu pesan Anda dengan kata kunci "HELP" (BANTUAN), kirimkan informasi tentang program langganan mereka, jumlah pesan yang akan Anda kirim setiap bulan, dan cara mereka dapat menghubungi Anda untuk informasi lebih lanjut. Respons HELP (BANTUAN) 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" (BERHENTI), beri tahu mereka bahwa mereka tidak akan menerima pesan lebih lanjut. Respons STOP (BERHENTI) 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 pesan SMS ketika pelanggan Anda tidak dapat segera merespons pesan tersebut.

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, jika Anda menggunakan beberapa saluran komunikasi (seperti email, SMS, 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 selengkapnya tentang meminta kode pendek, lihat Meminta kode pendek khusus untuk olahpesan SMS dengan Amazon SNS.

Verifikasi nomor telepon tujuan Anda

Saat Anda mengirim pesan SMS melalui Amazon SNS, 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 harga SMS, lihat Harga Amazon SNS.

Saat Amazon SNS menerima permintaan untuk mengirim pesan SMS (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 digunakan, 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 pesan SMS dan mencoba mengirimkannya. Untuk alasan ini, Anda harus memvalidasi bahwa nomor telepon yang Anda kirimi pesan adalah nomor ponsel yang valid. Anda dapat menggunakan layanan validasi nomor telepon Amazon SNS untuk menentukan apakah nomor telepon valid dan jenis nomornya (seperti ponsel, telepon rumah, atau VoIP). Untuk informasi selengkapnya, lihat Memvalidasi nomor telepon di Amazon Pinpoint di Panduan Pengembang Amazon Pinpoint.

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 pesan SMS — termasuk kode pendek, kode panjang, nomor bebas pulsa, dan nomor 10DLC — 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 Amazon SNS. 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, nomor 10DLC, 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.

Batas dan batasan SMS

Untuk batasan dan batasan SMS, lihat batasan dan batasan SMS di Amazon Pinpoint di Panduan Pengguna Amazon Pinpoint.

Mengelola kata kunci keluar

Penerima SMS dapat menggunakan perangkat mereka untuk memilih keluar dari pesan dengan membalas dengan kata kunci. Untuk informasi selengkapnya, lihat Memilih untuk tidak menerima pesan SMS.

CreatePool

Gunakan aksi `CreatePool` API untuk membuat kumpulan baru dan mengaitkan identitas originasi tertentu ke pool. Untuk informasi selengkapnya, lihat CreatePooldi Amazon Pinpoint SMS dan Voice API.

PutKeyword

Gunakan tindakan `PutKeyword` API untuk membuat atau memperbarui konfigurasi kata kunci pada nomor telepon atau kumpulan originasi. Untuk informasi selengkapnya, lihat PutKeyworddi Amazon Pinpoint SMS dan Voice API.

Mengelola pengaturan nomor

Anda dapat menggunakan opsi di bagian Pengaturan angka pada halaman pengaturan SMS dan suara untuk mengelola pengaturan kode pendek khusus dan kode panjang yang Anda minta dari AWS Support dan ditetapkan ke akun Anda. Untuk informasi selengkapnya, lihat Mengelola setelan nomor di Panduan Pengguna Amazon Pinpoint.

Batas karakter SMS di Amazon SNS

Satu pesan SMS dapat berisi hingga 140 byte informasi. Jumlah karakter yang dapat Anda sertakan dalam satu pesan SMS 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, 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 pesan SMS, 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 karakter GSM 03.38) 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 panjang SMS, beberapa di antaranya tersedia secara online. Ukuran maksimum yang didukung dari pesan apa pun adalah 1600 karakter GSM atau 630 karakter non-GSM. Untuk informasi selengkapnya tentang throughput dan ukuran pesan, lihat batas karakter SMS di Amazon Pinpoint di Panduan Pengguna Amazon Pinpoint.

Untuk melihat jumlah bagian pesan untuk setiap pesan yang Anda kirim, Anda harus mengaktifkan pengaturan aliran Acara terlebih dahulu. Ketika Anda melakukannya, Amazon SNS menghasilkan `_SMS.SUCCESS` peristiwa 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**
>
> Saat mengirim pesan yang berisi lebih dari satu bagian pesan, Anda akan dikenakan biaya untuk jumlah bagian pesan yang terdapat dalam pesan.

GSM 03.38 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.

GSM 03.38 karakter standar												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	D	T	U	V	W	X	Y	Z

GSM 03.38 karakter standar												
a	b	c	d	e	f	g	-h	saya	j	k	l	m
n	o	p	q	r	d	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:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Akhirnya, 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 olahpesan

Bagian ini berisi beberapa contoh pesan SMS. 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 di 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 karakter GSM 03.38. 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 karakter non-GSM

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 pesan SMS Anda di Microsoft Word dan menempelkannya ke Amazon SNS, 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.

Notifikasi push seluler

Dengan Amazon SNS, Anda memiliki kemampuan untuk mengirim pesan notifikasi push langsung ke aplikasi di perangkat seluler. Pesan notifikasi push yang dikirim ke endpoint seluler dapat muncul di aplikasi seluler sebagai peringatan pesan, pembaruan lencana, atau bahkan peringatan suara.

Topik

- Cara kerja notifikasi pengguna
- Gambaran umum proses notifikasi pengguna
- Menyiapkan aplikasi seluler
- Mengirim notifikasi push seluler
- Atribut aplikasi seluler
- Peristiwa aplikasi seluler
- Tindakan API push seluler
- Kesalahan API push seluler
- Menggunakan atribut pesan time to live (TTL) Amazon SNS untuk notifikasi push seluler
- Wilayah yang Didukung untuk aplikasi seluler
- Praktik terbaik pemberitahuan push seluler

Cara kerja notifikasi pengguna

Anda mengirim pesan notifikasi push ke kedua perangkat seluler dan desktop menggunakan salah satu layanan notifikasi push yang didukung berikut:

- Olahpesan Perangkat Amazon (ADM)
- Layanan Notifikasi Push Apple (APN) untuk iOS dan Mac OS X

- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Layanan Notifikasi Push Microsoft untuk Ponsel Windows (MPNS)
- Layanan Notifikasi Push Windows (WNS)

Layanan notifikasi push, seperti APN dan FCM, menjaga koneksi dengan setiap aplikasi dan perangkat seluler terkait terdaftar untuk menggunakan layanan mereka. Ketika aplikasi dan perangkat seluler terdaftar, layanan notifikasi push mengembalikan token perangkat. Amazon SNS menggunakan token perangkat untuk membuat endpoint seluler, yang dapat mengirim pesan notifikasi push langsung. Agar Amazon SNS dapat berkomunikasi dengan layanan notifikasi push yang berbeda, Anda mengirimkan kredensial layanan notifikasi push Anda ke Amazon SNS untuk digunakan atas nama Anda. Untuk informasi selengkapnya, lihat Gambaran umum proses notifikasi pengguna.

Selain mengirim pesan notifikasi push langsung, Anda juga dapat menggunakan Amazon SNS untuk mengirim pesan ke endpoint seluler yang berlangganan suatu topik. Konsepnya sama dengan berlangganan jenis endpoint lainnya, seperti Amazon SQS, HTTP/S, email, dan SMS, ke suatu topik, seperti yang dijelaskan dalam Apa itu Amazon SNS?. Perbedaannya adalah bahwa Amazon SNS berkomunikasi menggunakan layanan notifikasi push agar endpoint seluler berlangganan menerima pesan notifikasi push yang dikirim ke topik.

Gambaran umum proses notifikasi pengguna

1. Dapatkan kredensial dan token perangkat untuk platform seluler yang ingin Anda dukung.
2. Gunakan kredensial untuk membuat objek aplikasi platform (`PlatformApplicationArn`) menggunakan Amazon SNS. Untuk informasi selengkapnya, lihat Membuat aplikasi platform.
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 Amazon SNS. Untuk informasi selengkapnya, lihat Membuat endpoint platform.
5. Gunakan `EndpointArn` untuk memublikasikan pesan ke aplikasi di perangkat seluler. Untuk informasi selengkapnya, lihat Memublikasikan ke perangkat seluler dan Referensi API Publikasikan API di Amazon Simple Notification Service.

Menyiapkan aplikasi seluler

Bagian ini menjelaskan cara menggunakan AWS Management Console informasi yang dijelaskan Prasyarat untuk notifikasi pengguna Amazon SNS untuk menyiapkan aplikasi seluler.

Topik

- Prasyarat untuk notifikasi pengguna Amazon SNS
- Membuat aplikasi platform
- Membuat endpoint platform
- Menambahkan token perangkat atau ID pendaftaran
- Metode otentikasi Apple
- Metode autentikasi Firebase Cloud Messaging (FCM)
- Manajemen titik akhir Firebase Cloud Messaging (FCM)

Prasyarat untuk notifikasi pengguna Amazon SNS

Untuk mulai menggunakan notifikasi push seluler Amazon SNS, Anda memerlukan hal berikut:

- Satu set kredensi untuk menghubungkan ke salah satu layanan notifikasi push yang didukung: ADM, APN, Baidu, FCM, MPNS, atau WNS.
- Token perangkat atau ID registrasi untuk aplikasi dan perangkat seluler.
- Amazon SNS dikonfigurasi untuk mengirim pesan notifikasi push ke endpoint 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 notifikasi push untuk mengirim pesan notifikasi push langsung ke endpoint 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 APN sebagai layanan notifikasi push, Anda memerlukan token perangkat. Atau, saat menggunakan FCM, token perangkat yang setara disebut ID pendaftaran. 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 kredensial (ditambah beberapa pengaturan lainnya) sebagai sumber daya aplikasi platform. Token perangkat (sekali lagi dengan beberapa pengaturan tambahan) direpresentasikan sebagai objek yang disebut endpoint 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 mendapatkan informasi prasyarat, Anda dapat mengirim pesan notifikasi push menggunakan AWS Management Console atau API push seluler Amazon SNS. Untuk informasi selengkapnya, lihat Gambaran umum proses notifikasi pengguna.

Membuat aplikasi platform

Agar Amazon SNS mengirim pesan notifikasi ke endpoint seluler, baik secara langsung atau melalui berlangganan suatu topik, Anda harus membuat aplikasi platform terlebih dahulu. Setelah mendaftarkan aplikasi denganAWS, langkah selanjutnya adalah membuat endpoint untuk aplikasi dan perangkat seluler. Amazon SNS kemudian menggunakan endpoint untuk mengirimkan pesan notifikasi ke aplikasi dan perangkat.

Untuk membuat aplikasi platform

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi, pilihSeluler, dan kemudian pilihNotifikasi Push.
3. DiAplikasi platformbagian, pilihBuat aplikasi platform.

   Untuk daftarAWSWilayah tempat Anda dapat membuat aplikasi seluler, lihatWilayah yang Didukung untuk aplikasi seluler.
4. UntukNama aplikasi, masukkan nama untuk mewakili aplikasi Anda.

Nama aplikasi hanya boleh terdiri dari huruf besar dan kecil ASCII, angka, garis bawah, tanda hubung, dan titik. Nama juga harus panjangnya 1-256 karakter.

5. UntukPlatform notifikasi push, pilih platform tempat aplikasi terdaftar, lalu masukkan kredential yang sesuai.

> **Note**
>
> Jika Anda menggunakan salah satu platform Apple Push Notification Service (APN), Anda dapat memilih antaratoken atau otentikasi berbasis sertifikat, lalu pilihPilih fileuntuk mengunggah file .p8 atau .p12 (diekspor dari Akses Rantai Kunci) ke Amazon SNS.

6. Pilih Buat aplikasi platform.

   Ini mendaftarkan aplikasi dengan Amazon SNS, yang membuat objek aplikasi platform untuk platform yang dipilih lalu mengembalikan aplikasi yang sesuai PlatformApplicationArn.

Membuat endpoint platform

Saat aplikasi dan perangkat seluler terdaftar dengan layanan notifikasi push, layanan notifikasi push mengembalikan token perangkat. Amazon SNS menggunakan token perangkat untuk membuat endpoint seluler, yang dapat mengirim pesan notifikasi push langsung. Untuk informasi selengkapnya, lihat Prasyarat untuk notifikasi pengguna Amazon SNS dan Gambaran umum proses notifikasi pengguna.

Bagian ini menjelaskan pendekatan yang disarankan untuk membuat endpoint platform.

Topik

- Membuat endpoint platform
- Kode semu
- AWS Contoh SDK
- Pemecahan Masalah

Membuat endpoint platform

Untuk mendorong notifikasi ke aplikasi dengan Amazon SNS, token perangkat aplikasi tersebut harus terlebih dahulu terdaftar ke Amazon SNS dengan memanggil tindakan buat endpoint platform.

Tindakan ini mengambil Amazon Resource Name (ARN) dari aplikasi platform dan token perangkat sebagai parameter dan mengembalikan ARN dari endpoint platform yang dibuat.

`CreatePlatformEndpoint`Tindakan tersebut melakukan hal berikut:

- Jika endpoint platform sudah ada, maka jangan buat lagi. Kembali ke pemanggil ARN dari endpoint platform yang ada.
- Jika endpoint platform dengan token perangkat yang sama tetapi pengaturan yang berbeda sudah ada, maka jangan buat lagi. Lempar pengecualian ke pemanggil.
- Jika endpoint platform tidak ada, buatlah. Kembali ke pemanggil ARN dari endpoint 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
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

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 ARN endpoint platformnya sendiri, seperti yang terjadi selama pendaftaran pertama kali. Ini juga dipanggil jika panggilan tindakan awal mendapatkan atribut endpoint gagal dengan pengecualian yang tidak ditemukan, seperti yang akan terjadi jika aplikasi mengetahui ARN endpoint-nya tetapi telah dihapus.
- Tindakan mendapatkan atribut endpoint dipanggil untuk memverifikasi status endpoint platform meskipun endpoint 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 Contoh SDK

Kode berikut menunjukkan cara menerapkan kode semu sebelumnya menggunakan klien Amazon SNS yang disediakan oleh AWS SDK.

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

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
```

```
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Untuk informasi selengkapnya, lihat Tindakan API push seluler.

Pemecahan Masalah

Panggilan berulang kali membuat endpoint platform dengan token perangkat yang sudah usang

Khusus untuk titik akhir FCM, 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 start-up. 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. FCM menyimpan informasi tentang token perangkat lama untuk beberapa waktu, tetapi tidak selamanya. Setelah FCM melupakan koneksi antara token perangkat lama dan token perangkat baru, Amazon SNS tidak akan lagi dapat memperbarui token perangkat yang disimpan di endpoint platform ke nilai yang benar; itu hanya akan menonaktifkan endpoint platform saja.
- Aplikasi platform akan berisi beberapa endpoint platform yang sesuai dengan token perangkat yang sama.

- Amazon SNS memberlakukan kuota pada jumlah endpoint platform yang dapat dibuat mulai 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 titik akhir FCM, lihat. Manajemen titik akhir Firebase Cloud Messaging (FCM)

Mengaktifkan kembali endpoint platform yang terkait dengan token perangkat yang tidak valid

Saat platform seluler (seperti APN atau FCM) memberi tahu Amazon SNS bahwa token perangkat yang digunakan dalam permintaan publikasi tidak valid, Amazon SNS menonaktifkan endpoint platform yang terkait dengan token perangkat tersebut. Amazon SNS kemudian akan menolak publikasi berikutnya untuk token perangkat tersebut. 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. Saat berikutnya dipublikasikan, platform seluler akan kembali menginformasikan Amazon SNS bahwa token perangkat tidak valid, dan Amazon SNS akan menonaktifkan lagi endpoint 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.

Menambahkan token perangkat atau ID pendaftaran

Saat Anda pertama kali mendaftarkan aplikasi dan perangkat seluler dengan layanan notifikasi, seperti Layanan Apple Push Notification (APN) dan Firebase Cloud Messaging (FCM), token

perangkat atau ID pendaftaran dikembalikan dari layanan notifikasi. Saat Anda menambahkan token perangkat atau ID pendaftaran ke Amazon SNS, token tersebut digunakan dengan API `PlatformApplicationArn` untuk membuat endpoint untuk aplikasi dan perangkat. Saat Amazon SNS membuat endpoint, `EndpointArn` dikembalikan. `EndpointArn` adalah cara Amazon SNS mengetahui aplikasi dan perangkat seluler mana yang akan dikirimi pesan notifikasi.

Anda dapat menambahkan token perangkat dan ID pendaftaran ke Amazon SNS menggunakan metode berikut:

- Tambahkan satu token secara manual ke AWS menggunakan AWS Management Console
- Unggah beberapa token menggunakan API `CreatePlatformEndpoint`
- Daftarkan token dari perangkat yang akan menginstal aplikasi Anda di masa mendatang

Untuk menambahkan token perangkat atau ID pendaftaran secara manual

1. Masuk ke konsol Amazon SNS.
2. Pilih Seluler, lalu pilih Pemberitahuan Push.
3. Di bagian Aplikasi Platform, pilih aplikasi Anda dan kemudian pilih Edit. Jika Anda belum membuat aplikasi platform, buat sekarang. Untuk petunjuk tentang cara melakukannya, lihat Membuat aplikasi platform.
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. Terakhir, pilih Tambahkan Endpoint.

   Sekarang dengan endpoint yang dibuat, Anda dapat mengirim pesan langsung ke perangkat seluler atau mengirim pesan ke perangkat seluler yang berlangganan suatu topik.

Untuk mengunggah beberapa token menggunakan API **`CreatePlatformEndpoint`**

Langkah-langkah berikut menunjukkan cara menggunakan contoh aplikasi Java (paket `bulkupload`) yang disediakan oleh AWS untuk mengunggah beberapa token (perangkat token atau ID

pendaftaran) ke Amazon SNS. Anda dapat menggunakan aplikasi contoh ini untuk membantu Anda memulai mengunggah token yang ada.

> **Note**
>
> Langkah-langkah berikut menggunakan Eclipse Java IDE. Langkah-langkahnya mengasumsikan Anda telah menginstal AWS SDK for Java dan Anda memiliki kredensial keamanan AWS untuk Akun AWS Anda. 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 Pustaka OpenCSV dan tambahkan ke Build Path dari paket `bulkupload`.
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 file CSV Anda yang berisi token.
   - Nama untuk file CSV (seperti `goodTokens.csv` dan `badTokens.csv`) yang akan dibuat untuk mencatat token yang diurai Amazon SNS dengan benar dan yang gagal.
   - (Opsional) Karakter untuk menentukan pembatas dan kutipan dalam file CSV 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 Amazon SNS.

   Dalam contoh ini, endpoint yang dibuat untuk token yang berhasil diunggah ke Amazon SNS akan dicatat ke `goodTokens.csv`, sedangkan token yang cacat akan dicatat ke `badTokens.csv`. Selain itu, Anda akan melihat log STD OUT 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 kredensial untuk membuat endpoint yang terkait dengan aplikasi platform Amazon SNS Anda. 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 Amazon SNS, Anda dapat mendaftar untuk menerima peristiwa Amazon SNS yang akan menyediakan ARN endpoint baru. Anda juga dapat menggunakan API `ListEndpointByPlatformApplication` untuk mendapatkan daftar lengkap endpoint yang terdaftar di Amazon SNS.
- 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 proksi dan meneruskan token perangkat ke notifikasi push seluler Amazon SNS, bersama dengan data pengguna apa pun yang ingin Anda simpan. Untuk tujuan ini, server proksi akan terhubung ke Amazon SNS menggunakan kredensial AWS Anda dan menggunakan panggilan API `CreatePlatformEndpoint` untuk mengunggah informasi token. Endpoint Amazon Resource Name (ARN) yang baru dibuat akan dikembalikan,

yang dapat disimpan oleh server Anda untuk melakukan panggilan publikasi berikutnya ke Amazon SNS.

Metode otentikasi Apple

Anda dapat mengotorisasi Amazon SNS untuk mengirim notifikasi 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 digunakan Amazon SNS untuk menandatangani token autentikasi Apple Push Notification Service (APN).

Jika Anda memberikan kunci penandatanganan, Amazon SNS menggunakan token untuk mengautentikasi dengan APN untuk setiap notifikasi push yang Anda kirim. Dengan kunci penandatanganan, Anda dapat mengirim notifikasi push ke lingkungan produksi dan sandbox APN.

Kunci penandatanganan Anda tidak kedaluwarsa, dan Anda dapat menggunakan kunci penandatanganan yang sama untuk beberapa aplikasi. Untuk informasi selengkapnya, lihat Berkomunikasi dengan APN menggunakan token autentikasi di bagian Bantuan Akun Developer di situs web Apple.

Sertifikat

Sertifikat TLS yang digunakan Amazon SNS untuk mengautentikasi dengan APN saat Anda mengirim notifikasi push. Anda memperoleh sertifikat dari akun Pengembang Apple Anda.

Sertifikat kedaluwarsa setelah satu tahun. Ketika ini terjadi, Anda harus membuat sertifikat baru dan memberikannya ke Amazon SNS. Untuk informasi selengkapnya, lihat Membuat Koneksi Berbasis Sertifikat ke APN di situs web Pengembang Apple.

Mengelola setelan APN menggunakan AWS Management Console

1. Masuk ke Konsol Amazon SNS.
2. Di bagian Seluler, pilih Notifikasi push.
3. Pilih Aplikasi yang ingin Anda edit pengaturan APN, lalu pilih Edit.

4. Pada halaman Edit, untuk jenis Otentikasi, pilih Token atau Sertifikat.
5. Muat kredensyal yang sesuai untuk kunci penandatanganan sertifikat atau 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 autentikasi.
     - Kunci penandatanganan — Kunci penandatanganan token autentikasi 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 autentikasi. Untuk menemukan nilai ini di akun Pengembang Apple, pilih Sertifikat, ID, & 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.
     - Pengenal bundel — ID yang ditetapkan ke aplikasi Anda. Untuk menemukan nilai ini, pilih Sertifikat, ID, & Profil, pilih ID Aplikasi di bagian Pengidentifikasi, lalu pilih aplikasi Anda.
   - Jika Anda memilih Sertifikat, berikan informasi berikut:
     - Sertifikat SSL - File.p12 untuk sertifikat TLS Anda. Anda dapat mengekspor file ini dari Keychain Access setelah mengunduh dan menginstal sertifikat dari akun Pengembang Apple.
     - Kata sandi sertifikat — Jika Anda menetapkan kata sandi ke sertifikat Anda, tentukan di sini.
7. Setelah selesai, pilih Simpan perubahan.

Metode autentikasi Firebase Cloud Messaging (FCM)

Topik ini menjelaskan cara mendapatkan kredenal FCM API (HTTP v1) yang diperlukan dari Google untuk digunakan dengan AWS API, dan file. AWS CLI AWS Management Console

Topik

- Prasyarat
- Mengelola pengaturan FCM (API)

- Mengelola pengaturan FCM (CLI)
- Mengelola pengaturan FCM (konsol)

> **Important**
>
> 20 Juni 2023 — Google tidak menggunakan API HTTP lama Firebase Cloud Messaging (FCM) mereka. Amazon SNS sekarang mendukung pengiriman ke semua jenis perangkat menggunakan FCM HTTP v1 API. Kami menyarankan Anda memigrasikan aplikasi push seluler yang ada ke FCM HTTP v1 API terbaru 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 API terbaru pada atau sebelum 1 Juni 2024 untuk menghindari gangguan aplikasi.

Anda dapat mengotorisasi Amazon SNS untuk mengirim pemberitahuan push ke aplikasi Anda dengan memberikan informasi yang mengidentifikasi Anda sebagai pengembang aplikasi. Untuk mengautentikasi, berikan kunci API atau token saat membuat aplikasi platform. Anda bisa mendapatkan informasi berikut dari konsol aplikasi Firebase:

Kunci API

Kunci API adalah kredensi yang digunakan saat memanggil API Legacy Firebase. FCM Legacy API akan dihapus oleh Google 20 Juni 2024. Jika saat ini Anda menggunakan kunci API sebagai kredensi platform, Anda dapat memperbarui kredensi platform dengan memilih Token sebagai opsi, dan mengunggah file JSON terkait untuk aplikasi Firebase Anda.

Token

Token akses berumur pendek digunakan saat memanggil HTTP v1 API. Ini adalah API yang disarankan Firebase 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 service.json FCM Anda sebelum Anda dapat mulai mengelola pengaturan FCM di Amazon SNS. Untuk mendapatkan kredensial service.json, lihat Memigrasi dari API FCM lama ke HTTP v1 di dokumentasi Google Firebase.

Mengelola pengaturan FCM (API)

Anda dapat membuat notifikasi push FCM menggunakan AWS API. Jumlah dan ukuran sumber daya Amazon SNS dalam suatu AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service di Panduan.Referensi Umum AWS

Untuk membuat notifikasi push FCM bersama dengan topik AWS Amazon SNS (API)

Saat menggunakan kredensi kunci, adalah. `PlatformCredential` `API key` Saat menggunakan kredensi token, file kunci `PlatformCredential` pribadi berformat JSON:

- `CreatePlatformApplication`

Untuk mengambil jenis kredensi FCM untuk topik Amazon SNS (API) yang ada AWS

Mengambil jenis kredensi`"AuthenticationMethod": "Token"`, atau: `"AuthenticationMethod": "Key"`

- GetPlatformApplicationAttributes

Untuk menyetel atribut FCM untuk topik AWS Amazon SNS (API) yang ada

Menetapkan atribut FCM:

- SetPlatformApplicationAttributes

Mengelola pengaturan FCM (CLI)

Anda dapat membuat notifikasi push FCM menggunakan AWS Command Line Interface (CLI). Jumlah dan ukuran sumber daya Amazon SNS dalam suatu AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Untuk membuat notifikasi push FCM bersama dengan topik Amazon SNS ()AWS CLI

Saat menggunakan kredensi kunci, adalah. `PlatformCredential` `API key` Saat menggunakan kredensi token, file kunci `PlatformCredential` pribadi berformat JSON. Saat menggunakan AWS CLI, file harus dalam format string dan karakter khusus harus diabaikan. Untuk memformat file dengan benar, Amazon SNS merekomendasikan penggunaan perintah berikut: `SERVICE_JSON=`jq @json <<< cat service.json``

- create-platform-application

Untuk mengambil jenis kredensi FCM untuk topik Amazon SNS yang ada ()AWS CLI

Mengambil jenis kredensi`"AuthenticationMethod": "Token"`, atau: `"AuthenticationMethod": "Key"`

- get-platform-application-attributes

Untuk menetapkan atribut FCM untuk topik Amazon SNS yang ada ()AWS CLI

Menetapkan atribut FCM:

- set-platform-application-attributes

Mengelola pengaturan FCM (konsol)

Gunakan langkah-langkah berikut untuk memasukkan kredenal yang digunakan aplikasi Anda untuk terhubung ke FCM.

1. Masuk ke Konsol Amazon SNS.
2. Di Seluler, pilih Pemberitahuan push.
3. Pilih aplikasi FCM yang ada dan pilih Edit. Jika Anda belum membuat aplikasi platform, lihatMembuat aplikasi platform.
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 Key, masukkan kunci Google API.
5. Setelah selesai, pilih Simpan perubahan.

Topik terkait

- Menggunakan payload Google Firebase Cloud Messaging (FCM) v1 di Amazon SNS

Manajemen titik akhir Firebase Cloud Messaging (FCM)

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, ARN endpoint Amazon SNS yang sesuai, dan stempel waktu di server aplikasi Anda.
2. Hapus semua token basi dan hapus ARN endpoint Amazon SNS 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 aplikasi FCM Anda. Setelah Anda menerima token perangkat ini, Anda dapat mendaftarkannya dengan Amazon SNS sebagai titik akhir platform. Kami menyarankan Anda menyimpan token perangkat, ARN endpoint platform Amazon SNS, dan stempel waktu dengan menyimpannya ke server aplikasi Anda, atau toko persisten lainnya. Untuk menyiapkan aplikasi FCM 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 titik akhir Amazon SNS 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 semu FCM 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, Amazon SNS akan menerima salah satu pengecualian berikut:

- `UNREGISTERED`(HTTP 404) — Saat Amazon SNS menerima pengecualian ini, Anda akan menerima peristiwa kegagalan pengiriman dengan `FailureType` `InvalidPlatformToken` of,

dan token Platform *`FailureMessage`* yang terkait dengan titik akhir tidak valid. Amazon SNS akan menonaktifkan titik akhir platform Anda saat pengiriman gagal dengan pengecualian ini.

- `INVALID_ARGUMENT`(HTTP 400) — Ketika 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 a `FailureTypeInvalidNotification`, dan badan Pemberitahuan tidak valid. `FailureMessage` Ketika Anda menerima kesalahan ini, verifikasi bahwa payload Anda sudah benar. Jika benar, verifikasi bahwa token perangkat tersebut up-to-date. Amazon SNS tidak 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 teramati yang diterima dari FCM v1 API tersedia untuk Anda CloudWatch saat Anda menyiapkan 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 peristiwa `EventDeliveryFailure` dengan `FailureType` `EndpointDisabled` dari, dan `FailureMessage` Endpoint dinonaktifkan. Untuk menerima acara pengiriman untuk aplikasi Anda, lihatPeristiwa aplikasi yang tersedia.

Saat Anda menerima kesalahan ini dari Amazon SNS, Anda perlu menghapus atau memperbarui token basi di aplikasi platform Anda.

Mengirim notifikasi push seluler

Bagian ini menjelaskan cara mengirim notifikasi push seluler.

Topik

- Menerbitkan ke topik
- Memublikasikan ke perangkat seluler
- Penerbitan dengan muatan khusus platform

Menerbitkan ke topik

Anda juga dapat menggunakan Amazon SNS untuk mengirim pesan ke endpoint seluler yang berlangganan suatu topik. Konsepnya sama dengan berlangganan jenis endpoint lainnya, seperti Amazon SQS, HTTP/S, email, dan SMS, ke suatu topik, seperti yang dijelaskan dalam Apa itu 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.

Memublikasikan ke perangkat seluler

Anda dapat mengirim pesan notifikasi push Amazon SNS langsung ke endpoint yang mewakili aplikasi pada perangkat seluler.

Untuk mengirim pesan langsung

1. Masuk ke konsol Amazon SNS.
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 lalu 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, mengirimkan pesan ke aplikasi.

Penerbitan dengan muatan khusus platform

Anda dapat menggunakan API Amazon SNS AWS Management Console atau Amazon untuk mengirim pesan khusus dengan muatan khusus platform ke perangkat seluler. Untuk informasi tentang menggunakan Amazon SNS API, lihat Tindakan API push seluler dan file `SNSMobilePush.java` di `snsmobilepush.zip`.

Topik

- Mengirim pesan berformat JSON
- Mengirim pesan khusus platform
- Mengirim pesan ke aplikasi di berbagai platform
- Mengirim pesan ke APN sebagai peringatan atau notifikasi latar belakang
- Menggunakan payload Google Firebase Cloud Messaging (FCM) v1 di Amazon SNS

Mengirim pesan berformat JSON

Saat Anda mengirim muatan khusus platform, data harus diformat sebagai string pasangan nilai kunci JSON, dengan tanda kutip dililoskan.

Contoh berikut menunjukkan pesan khusus untuk platform FCM.

```
{
"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 parameter FCM `time_to_live` dan `collapse_key` setelah pasangan nilai kunci data kustom dalam parameter `data` FCM.

```
{
"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 kunci-nilai yang didukung oleh masing-masing layanan notifikasi push yang didukung di Amazon SNS, lihat berikut ini:

> **Important**
>
> Amazon SNS sekarang mendukung Firebase Cloud Messaging (FCM) HTTP v1 API untuk mengirimkan 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 API terbaru pada atau sebelum 1 Juni 2024 untuk menghindari gangguan aplikasi.

- Referensi Kunci Muatan dalam dokumentasi APN
- Protokol HTTP Firebase Cloud Messaging dalam dokumentasi FCM
- Kirim Pesan di dokumentasi ADM

Mengirim pesan ke aplikasi di berbagai platform

Untuk mengirim pesan ke aplikasi yang diinstal pada perangkat untuk beberapa platform, seperti FCM dan APN, Anda harus terlebih dahulu berlangganan endpoint seluler ke suatu topik di Amazon SNS dan kemudian memublikasikan pesan ke topik tersebut.

Contoh berikut menunjukkan pesan untuk dikirim ke endpoint seluler berlangganan di APN, 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 APN sebagai peringatan atau notifikasi latar belakang

Amazon SNS dapat mengirim pesan ke APN sebagai notifikasi `alert` atau `background` (untuk informasi selengkapnya, lihat Mendorong Pembaruan Latar Belakang ke Aplikasi Anda di dokumentasi APN).

- Notifikasi APN `alert` memberi tahu pengguna dengan menampilkan pesan peringatan, memutar suara, atau menambahkan lencana ke ikon aplikasi Anda.
- Notifikasi APN `background` membangunkan atau menginstruksikan aplikasi Anda untuk bertindak berdasarkan konten notifikasi, tanpa memberi tahu pengguna.

Menentukan nilai header APN kustom

Sebaiknya tentukan nilai kustom untuk atribut pesan yang `AWS.SNS.MOBILE.APNS.PUSH_TYPE` dicadangkan menggunakan tindakan Amazon `Publish` SNS API AWS , SDK, atau file. AWS CLI Contoh CLI berikut menyetel `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
```

Menyimpulkan header jenis push APN dari muatan

Jika Anda tidak menyetel header APN `apns-push-type`, Amazon SNS menyetel header ke `alert` atau `background` bergantung pada kunci `content-available` dalam kamus `aps` konfigurasi muatan APN berformat JSON.

> **Note**
>
> Amazon SNS hanya dapat menyimpulkan header `alert` atau `background`, meskipun header `apns-push-type` 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 untuk APN sebagai notifikasi latar belakang saja, Anda harus menyertakan `content-available` yang diatur 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 APN header `apns-push-type` untuk pesan ke `background`. Saat Amazon SNS mendeteksi bahwa kamus `apn` berisi kunci `content-available` yang diatur ke 1—dan tidak berisi kunci lain yang dapat memicu interaksi pengguna—itu menyetel header ke `background`.

Menggunakan payload Google Firebase Cloud Messaging (FCM) 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 menggunakan CLI, atau Amazon SNS API.

Anda dapat menyertakan jenis pesan berikut di payload saat mengirim notifikasi FCM:

- 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 objek JSON 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
  - Kunci pemberitahuan APNS
  - Tombol notifikasi Webpush

Untuk informasi selengkapnya tentang jenis pesan FCM, lihat Jenis pesan di dokumentasi Firebase Google.

Daftar Isi

- Menggunakan struktur payload FCM v1 untuk mengirim pesan
- Menggunakan struktur payload lama untuk mengirim pesan ke FCM v1 API
- Peristiwa kegagalan pengiriman FCM

Menggunakan struktur payload FCM v1 untuk mengirim pesan

Jika Anda membuat aplikasi FCM 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 API FCM lama ke HTTP v1 dan Menyesuaikan pesan di seluruh platform dalam dokumentasi Firebase Google.

Contoh payload FCM v1 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 payload JSON, pastikan untuk menyertakan `message-structure` atribut dalam permintaan Anda, dan atur ke. `json`

Contoh CLI:

```
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 pengiriman payload berformat FCM v1, lihat hal berikut di dokumentasi Firebase Google:

- Migrasi dari API FCM lama ke HTTP v1
- Tentang pesan FCM
- Sumber Daya REST: projects.messages

Menggunakan struktur payload lama untuk mengirim pesan ke FCM v1 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 yang 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 APN Anda. Jika saat ini Anda mengirim notifikasi senyap ke perangkat APN Anda, mereka tidak akan berfungsi dengan pendekatan lama. Sebagai gantinya, sebaiknya gunakan payload FCM v1 untuk menghindari masalah yang tidak terduga. Untuk menemukan daftar header APN dan tujuan penggunaannya, lihat Berkomunikasi dengan APN di Panduan Pengembang Apple.
- Jika Anda menggunakan atribut TTL Amazon SNS saat mengirim notifikasi, itu hanya akan diperbarui di `android` lapangan. Jika Anda ingin menyetel atribut TTL APNS, 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.

Peristiwa kegagalan pengiriman FCM

Tabel berikut menyediakan jenis kegagalan Amazon SNS 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.

Kode kesalahan/ status FCM	Jenis kegagalan Amazon SNS	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 titik akhir Amazon SNS 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 titik akhir Amazon SNS 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

Kode kesalahan/ status FCM	Jenis kegagalan Amazon SNS	Pesan kegagalan	Penyebab dan mitigasi
			untuk mengirim ke token perangkat . Verifikasi bahwa Anda menggunak an kredensi FCM yang benar di aplikasi platform Amazon SNS Anda.
`UNAVAILABLE`	`Dependenc yUnavailable`	Ketergantungan tidak tersedia.	FCM tidak dapat memproses permintaa n tepat waktu. Semua percobaan ulang yang dijalankan oleh Amazon SNS telah gagal. Anda dapat menyimpan pesan-pesan ini dalam antrian huruf mati (DLQ) dan mengaktif kannya kembali nanti.

Kode kesalahan/ status FCM	Jenis kegagalan Amazon SNS	Pesan kegagalan	Penyebab dan mitigasi
`INTERNAL`	`UnexpectedFailure`	Kegagalan tak terduga; silakan hubungi Amazon. Frasa kegagalan [Kesalahan Internal].	Server FCM mengalami kesalahan saat mencoba memproses permintaan Anda. Semua percobaan ulang yang dijalankan oleh Amazon SNS telah gagal. Anda dapat menyimpan pesan-pesan ini dalam antrian huruf mati (DLQ) dan mengaktifkannya kembali nanti.
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	Kredensi aplikasi platform tidak valid.	Pesan yang ditargetkan ke perangkat iOS atau perangkat Webpush tidak dapat dikirim. Verifikasi 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.

Kode kesalahan/ status FCM	Jenis kegagalan Amazon SNS	Pesan kegagalan	Penyebab dan mitigasi
`PERMISSIO N_DENIED`	`InvalidNo tification`	Badan notifikasi tidak valid.	Dalam kasus `PERMISSIO N_DENIED` pengecualian, pemanggil (aplikasi FCM Anda) tidak memiliki izin untuk menjalankan operasi yang ditentuka n dalam muatan. Arahkan ke konsol FCM Anda, dan verifikasi bahwa tindakan API yang diperlukan telah diaktifkan.

Atribut aplikasi seluler

Amazon Simple Notification Service (Amazon SNS) menyediakan dukungan untuk mencatat status pengiriman pesan notifikasi push. Setelah Anda mengonfigurasi atribut aplikasi, entri log akan dikirim ke CloudWatch Logs untuk pesan yang dikirim dari Amazon SNS ke endpoint seluler. Mencatat status pengiriman pesan membantu memberikan wawasan operasional yang lebih baik, seperti berikut ini:

- Mengetahui apakah pesan notifikasi push dikirim dari Amazon SNS ke layanan notifikasi push.
- Mengidentifikasi respons yang dikirim dari layanan notifikasi push ke Amazon SNS.
- 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, perangkat pengembangan perangkat lunak (SDK) AWS, atau API kueri.

Topik

- Mengonfigurasi atribut status pengiriman pesan menggunakan AWS Management Console
- Status pengiriman pesan Amazon SNS contoh log CloudWatch
- Mengonfigurasi atribut status pengiriman pesan dengan AWS SDK
- Kode respons platform

Mengonfigurasi atribut status pengiriman pesan menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
2. Pada panel navigasi, arahkan ke Seluler, lalu pilih Notifikasi push.
3. Dari bagian Aplikasi platform, pilih aplikasi yang berisi endpoint yang Anda inginkan untuk menerima CloudWatch Logs.
4. Pilih Tindakan Aplikasi lalu pilih Status Pengiriman.
5. Pada kotak dialog Status Pengiriman, pilih Buat Peran IAM.

   Anda kemudian akan diarahkan ke konsol IAM.
6. Pilih Izinkan untuk memberikan akses tulis Amazon SNS untuk menggunakan CloudWatch Logs atas nama Anda.
7. Sekarang, kembali ke kotak dialog Status Pengiriman, masukkan angka di bidang Persentase Sukses untuk Sampel (0-100) untuk persentase pesan yang berhasil dikirim yang ingin Anda menerima CloudWatch Logs.

   > **Note**
   >
   > Setelah Anda mengonfigurasi atribut aplikasi untuk status pengiriman pesan, semua pengiriman pesan yang gagal menghasilkan CloudWatch Logs.
8. Terakhir, pilih Simpan Konfigurasi. Anda sekarang dapat melihat dan menguraikan CloudWatch Logs yang berisi status pengiriman pesan. Untuk informasi selengkapnya tentang menggunakan CloudWatch, lihat Dokumentasi CloudWatch.

Status pengiriman pesan Amazon SNS contoh log CloudWatch

Setelah Anda mengonfigurasi atribut status pengiriman pesan untuk endpoint aplikasi, CloudWatch Logs akan dibuat. Contoh log, dalam format JSON, ditampilkan sebagai berikut:

KEBERHASILAN


```
{
  "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"
  }
}
```


KEGAGALAN


```
{
  "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.

Mengonfigurasi atribut status pengiriman pesan dengan AWS SDK

AWS SDK menyediakan API dalam beberapa bahasa untuk menggunakan atribut status pengiriman pesan dengan Amazon SNS.

Contoh Java berikut menunjukkan cara menggunakan API `SetPlatformApplicationAttributes` untuk mengonfigurasi atribut aplikasi untuk status pengiriman pesan dari pesan notifikasi push. Anda dapat menggunakan atribut berikut untuk status pengiriman pesan: `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn`, dan `SuccessFeedbackSampleRate`. Atribut `SuccessFeedbackRoleArn` dan `FailureFeedbackRoleArn` digunakan untuk memberikan akses tulis Amazon SNS untuk menggunakan CloudWatch Logs atas nama Anda. Atribut `SuccessFeedbackSampleRate` adalah untuk menentukan persentase tingkat sampel (0-100) dari pesan yang berhasil terkirim. Setelah Anda mengonfigurasi atribut `FailureFeedbackRoleArn`, maka semua pengiriman pesan yang gagal menghasilkan CloudWatch Logs.

```
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 SDK for Java, lihat Memulai dengan AWS SDK for Java.

Kode respons platform

Berikut ini adalah daftar tautan untuk kode respons layanan notifikasi push:

Layanan notifikasi push	Kode respons
Olahpesan Perangkat Amazon (ADM)	Lihat Format Respons dalam dokumentasi ADM.
Layanan Notifikasi Push Apple (APN)	Lihat Respons HTTP/2 dari APN dalam Berkomunikasi dengan APN di Panduan Pemrograman Notifikasi Lokal dan Jarak Jauh.
Firebase Cloud Messaging (FCM)	Lihat Kode Respons Kesalahan Pesan Hilir di dokumentasi Firebase Cloud Messaging.
Layanan Notifikasi Push Microsoft untuk Ponsel Windows (MPNS)	Lihat Kode Respons Layanan Notifikasi Push untuk Ponsel Windows 8 dalam dokumentasi Pengembangan Windows 8.
Layanan Notifikasi Push Windows (WNS)	Lihat "Kode respons" di Permintaan Layanan Notifikasi Push dan Header Respons (Aplikasi Windows Runtime) di dokumentasi Pengembangan Windows 8.

Peristiwa aplikasi seluler

Amazon SNS menyediakan dukungan untuk memicu notifikasi saat peristiwa aplikasi tertentu terjadi. Anda kemudian dapat mengambil beberapa tindakan terprogram pada peristiwa itu. Aplikasi Anda harus menyertakan dukungan untuk layanan notifikasi push seperti Apple Push Notification Service (APN), Firebase Cloud Messaging (FCM), dan Windows Push Notification Services (WNS). Anda mengatur notifikasi peristiwa aplikasi menggunakan konsol Amazon SNS AWS CLI, atau SDK AWS .

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 Aplikasi Amazon SNS 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 mengarahkannya ke endpoint 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 titik akhir HTTP atau HTTPS, antrian Amazon SQS, 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 notifikasi acara aplikasi menggunakan konsol Amazon SNS, AWS Command Line Interface (AWS CLI), atau SDK AWS .

AWS Management Console

1. Masuk ke konsol Amazon SNS.
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 ARN untuk topik yang akan digunakan untuk peristiwa berikut:
   - Endpoint Dibuat
   - Endpoint Dihapus
   - Endpoint Diperbarui
   - Kegagalan Pengiriman
7. Pilih Save changes (Simpan perubahan).

AWS CLI

Jalankan perintah set-platform-application-attributes.

Contoh berikut menetapkan topik Amazon SNS yang sama untuk keempat peristiwa 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 SDK

Tetapkan notifikasi peristiwa aplikasi dengan mengirimkan `SetPlatformApplicationAttributes` permintaan dengan Amazon SNS API menggunakan SDK. AWS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, termasuk bantuan memulai dan informasi tentang versi sebelumnya, lihatMenggunakan Amazon SNS dengan SDK AWS.

Tindakan API push seluler

Untuk menggunakan API push seluler Amazon SNS, Anda harus terlebih dahulu memenuhi prasyarat untuk layanan notifikasi push, seperti Apple Push Notification Service (APN) dan Firebase Cloud Messaging (FCM). Untuk informasi selengkapnya tentang prasyarat, lihat Prasyarat untuk notifikasi pengguna Amazon SNS.

Untuk mengirim pesan notifikasi push ke aplikasi dan perangkat seluler menggunakan API, Anda harus terlebih dahulu menggunakan tindakan `CreatePlatformApplication`, yang mengembalikan atribut `PlatformApplicationArn`. 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 Gambaran umum proses notifikasi pengguna.

API push seluler Amazon SNS adalah sebagai berikut:

`CreatePlatformApplication`

> Membuat objek aplikasi platform untuk salah satu layanan notifikasi push yang didukung, seperti APN dan FCM, 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 tindakan ConfirmSubscription dengan token dari pesan konfirmasi.

`Unsubscribe`

Menghapus langganan.

Kesalahan API push seluler

Kesalahan yang dikembalikan oleh Amazon SNS API untuk push seluler tercantum dalam tabel berikut. Untuk informasi selengkapnya tentang Amazon SNS API untuk push seluler, lihat Tindakan API push seluler.

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
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	Nilai yang tidak valid atau di luar rentang diberikan untuk nama platform.	400	`CreatePlatformApplication`
APN — Prinsipal bukan sertifikat yang valid	Sertifikat yang tidak valid diberikan untuk prinsipal APN, yang merupakan sertifikat SSL. Untuk informasi	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
	selengkapnya, lihat CreatePlatformApplication di Referensi API Amazon Simple Notification Service.		
APN — Prinsipal adalah sertifikat yang valid tetapi tidak dalam format .pem	Sertifikat valid yang tidak dalam format .pem diberikan untuk prinsipal APN, yang merupakan sertifikat SSL.	400	`CreatePlatformApplication`
APN — Prinsipal adalah sertifikat yang kedaluwarsa	Sertifikat yang kedaluwarsa diberikan untuk prinsipal APN, yang merupakan sertifikat SSL.	400	`CreatePlatformApplication`
APN — Prinsipal bukan sertifikat yang dikeluarkan Apple	Sertifikat yang diterbitkan non-Apple diberikan untuk prinsipal APN, yang merupakan sertifikat SSL.	400	`CreatePlatformApplication`
APN — Prinsipal tidak disediakan	Prinsipal APN, yang merupakan sertifikat SSL, tidak disediakan.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
APN — Kredensial tidak disediakan	Kredensial APN, yang merupakan kunci privat, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplication di Referensi API Amazon Simple Notification Service.	400	`CreatePlatformApplication`
APN — Kredensial tidak dalam format .pem yang valid	Kredensial APN, yang merupakan kunci privat, tidak dalam format .pem yang valid.	400	`CreatePlatformApplication`
FCM — serverAPIKey tidak disediakan	Kredensial FCM, yang merupakan kunci API, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplication di Referensi API Amazon Simple Notification Service.	400	`CreatePlatformApplication`
FCM — serverAPIKey kosong	Kredensial FCM, yang merupakan kunci API, kosong.	400	`CreatePlatformApplication`
FCM — serverAPIKey adalah string null	Kredensial FCM, yang merupakan kunci API, adalah null.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
FCM — serverAPIKey tidak valid	Kredensial FCM, yang merupakan kunci API, 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`
ADM — clientid adalah string null	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`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
EventEndpointCreated memiliki format ARN yang tidak valid	EventEndpointCreated memiliki format ARN yang tidak valid.	400	`CreatePlatformApplication`
EventEndpointDeleted memiliki format ARN yang tidak valid	EventEndpointDeleted memiliki format ARN yang tidak valid.	400	`CreatePlatformApplication`
EventEndpointUpdated memiliki format ARN yang tidak valid	EventEndpointUpdated memiliki format ARN yang tidak valid.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure memiliki format ARN yang tidak valid	EventDeliveryAttemptFailure memiliki format ARN yang tidak valid.	400	`CreatePlatformApplication`
EventDeliveryFailure memiliki format ARN yang tidak valid	EventDeliveryFailure memiliki format ARN yang tidak valid.	400	`CreatePlatformApplication`
EventEndpointCreated bukan Topik yang sudah ada	EventEndpointCreated bukan topik yang sudah ada.	400	`CreatePlatformApplication`
EventEndpointDeleted bukan Topik yang sudah ada	EventEndpointDeleted bukan topik yang sudah ada.	400	`CreatePlatformApplication`
EventEndpointUpdated bukan Topik yang sudah ada	EventEndpointUpdated bukan topik yang sudah ada.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure bukan Topik yang sudah ada	EventDeliveryAttemptFailure bukan topik yang sudah ada.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
EventDeliveryFailure bukan Topik yang sudah ada	EventDeliveryFailure bukan topik yang sudah ada.	400	`CreatePlatformApplication`
ARN platform tidak valid	ARN platform tidak valid.	400	`SetPlatformAttributes`
ARN platform valid tetapi bukan milik pengguna	ARN platform valid tetapi bukan milik pengguna.	400	`SetPlatformAttributes`
APN — Prinsipal bukan sertifikat yang valid	Sertifikat yang tidak valid diberikan untuk prinsipal APN, yang merupakan sertifikat SSL. Untuk informasi selengkapnya, lihat CreatePlatformApplication di Referensi API Amazon Simple Notification Service.	400	`SetPlatformAttributes`
APN — Prinsipal adalah sertifikat yang valid tetapi tidak dalam format .pem	Sertifikat valid yang tidak dalam format .pem diberikan untuk prinsipal APN, yang merupakan sertifikat SSL.	400	`SetPlatformAttributes`
APN — Prinsipal adalah sertifikat yang kedaluwarsa	Sertifikat yang kedaluwarsa diberikan untuk prinsipal APN, yang merupakan sertifikat SSL.	400	`SetPlatformAttributes`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
APN — Prinsipal bukan sertifikat yang dikeluarkan Apple	Sertifikat yang diterbitkan non-Apple diberikan untuk prinsipal APN, yang merupakan sertifikat SSL.	400	`SetPlatformAttributes`
APN — Prinsipal tidak disediakan	Prinsipal APN, yang merupakan sertifikat SSL, tidak disediakan.	400	`SetPlatformAttributes`
APN — Kredensial tidak disediakan	Kredensial APN, yang merupakan kunci privat, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplication di Referensi API Amazon Simple Notification Service.	400	`SetPlatformAttributes`
APN — Kredensial tidak dalam format .pem yang valid	Kredensial APN, yang merupakan kunci privat, tidak dalam format .pem yang valid.	400	`SetPlatformAttributes`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
FCM — serverAPIKey tidak disediakan	Kredensial FCM, yang merupakan kunci API, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplication di Referensi API Amazon Simple Notification Service.	400	`SetPlatformAttributes`
FCM — serverAPIKey adalah string null	Kredensial FCM, yang merupakan kunci API, adalah null.	400	`SetPlatformAttributes`
ADM — clientId tidak disediakan	String yang diperlukan untuk ID klien tidak disediakan.	400	`SetPlatformAttributes`
ADM — clientid adalah string null	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`
EventEndpointUpdated memiliki format ARN yang tidak valid	EventEndpointUpdated memiliki format ARN yang tidak valid.	400	`SetPlatformAttributes`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
EventEndpointDeleted memiliki format ARN yang tidak valid	EventEndpointDeleted memiliki format ARN yang tidak valid.	400	`SetPlatformAttributes`
EventEndpointUpdated memiliki format ARN yang tidak valid	EventEndpointUpdated memiliki format ARN yang tidak valid.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure memiliki format ARN yang tidak valid	EventDeliveryAttemptFailure memiliki format ARN yang tidak valid.	400	`SetPlatformAttributes`
EventDeliveryFailure memiliki format ARN yang tidak valid	EventDeliveryFailure memiliki format ARN yang tidak valid.	400	`SetPlatformAttributes`
EventEndpointCreated bukan Topik yang sudah ada	EventEndpointCreated bukan topik yang sudah ada.	400	`SetPlatformAttributes`
EventEndpointDeleted bukan Topik yang sudah ada	EventEndpointDeleted bukan topik yang sudah ada.	400	`SetPlatformAttributes`
EventEndpointUpdated bukan Topik yang sudah ada	EventEndpointUpdated bukan topik yang sudah ada.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure bukan Topik yang sudah ada	EventDeliveryAttemptFailure bukan topik yang sudah ada.	400	`SetPlatformAttributes`
EventDeliveryFailure bukan Topik yang sudah ada	EventDeliveryFailure bukan topik yang sudah ada.	400	`SetPlatformAttributes`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
ARN platform tidak valid	ARN platform tidak valid.	400	`GetPlatformApplicationAttributes`
ARN platform valid tetapi bukan milik pengguna	ARN platform valid tetapi bukan milik pengguna.	403	`GetPlatformApplicationAttributes`
Token yang ditentukan tidak valid	Token yang ditentukan tidak valid.	400	`ListPlatformApplications`
ARN platform tidak valid	ARN platform tidak valid.	400	`ListEndpointsByPlatformApplication`
ARN platform valid tetapi bukan milik pengguna	ARN platform valid tetapi bukan milik pengguna.	404	`ListEndpointsByPlatformApplication`
Token yang ditentukan tidak valid	Token yang ditentukan tidak valid.	400	`ListEndpointsByPlatformApplication`
ARN platform tidak valid	ARN platform tidak valid.	400	`DeletePlatformApplication`
ARN platform valid tetapi bukan milik pengguna	ARN platform valid tetapi bukan milik pengguna.	403	`DeletePlatformApplication`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
ARN platform tidak valid	ARN platform tidak valid.	400	`CreatePlatformEndpoint`
ARN platform valid tetapi bukan milik pengguna	ARN platform 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	Panjang data pengguna pelanggan tidak boleh lebih dari 2048 byte dalam pengkodean UTF-8.	400	`CreatePlatformEndpoint`
ARN endpoint tidak valid	ARN endpoint tidak valid.	400	`DeleteEndpoint`
ARN endpoint valid tetapi bukan milik pengguna	ARN endpoint valid tetapi bukan milik pengguna.	403	`DeleteEndpoint`
ARN endpoint tidak valid	ARN endpoint tidak valid.	400	`SetEndpointAttributes`
ARN endpoint valid tetapi bukan milik pengguna	ARN endpoint valid tetapi bukan milik pengguna.	403	`SetEndpointAttributes`
Token tidak ditentukan	Token tidak ditentukan.	400	`SetEndpointAttributes`

Kesalahan	Deskripsi	Kode status HTTPS	Tindakan API
Token tidak memiliki panjang yang benar	Token tidak memiliki panjang yang benar.	400	`SetEndpointAttributes`
Data Pengguna Pelanggan terlalu besar	Panjang data pengguna pelanggan tidak boleh lebih dari 2048 byte dalam pengkodean UTF-8.	400	`SetEndpointAttributes`
ARN endpoint tidak valid	ARN endpoint tidak valid.	400	`GetEndpointAttributes`
ARN endpoint valid tetapi bukan milik pengguna	ARN endpoint valid tetapi bukan milik pengguna.	403	`GetEndpointAttributes`
ARN target tidak valid	ARN target tidak valid.	400	`Publish`
ARN target valid tetapi bukan milik pengguna	ARN target 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 time to live (TTL) Amazon SNS untuk notifikasi push seluler

Amazon Simple Notification Service (Amazon SNS) menyediakan dukungan untuk menyetel atribut pesan Time To Live (TTL) untuk pesan notifikasi push seluler. Ini merupakan tambahan dari

kemampuan yang ada untuk menyetel TTL dalam badan pesan Amazon SNS untuk layanan notifikasi push seluler yang mendukung hal ini, seperti Amazon Device Messaging (ADM) dan Firebase Cloud Messaging (FCM) saat mengirim ke Android.

Atribut pesan TTL digunakan untuk menentukan metadata kedaluwarsa tentang pesan. Ini memungkinkan Anda untuk menentukan berapa lama layanan notifikasi push, seperti Apple Push Notification Service (APN) atau FCM, harus mengirimkan pesan ke endpoint. Jika karena alasan tertentu (seperti perangkat seluler telah dimatikan) pesan tidak terkirim dalam TTL yang ditentukan, maka pesan akan dihapus dan tidak ada upaya pengiriman lebih lanjut yang akan dilakukan. Untuk menentukan TTL dalam atribut pesan, Anda dapat menggunakan AWS Management Console, kit pengembangan AWS perangkat lunak (SDK), atau API kueri.

Topik

- Atribut pesan TTL untuk layanan notifikasi push
- Urutan prioritas untuk menentukan TTL
- Menentukan TTL menggunakan AWS Management Console

Atribut pesan TTL untuk layanan notifikasi push

Berikut ini adalah daftar atribut pesan TTL untuk layanan pemberitahuan push yang dapat Anda gunakan untuk mengatur saat menggunakan AWS SDK atau API kueri:

Layanan notifikasi push	Atribut pesan TTL
Olahpesan Perangkat Amazon (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Layanan Notifikasi Push Apple (APN)	`AWS.SNS.MOBILE.APNS.TTL`
Sandbox Layanan Notifikasi Push Apple (APNS_Sandbox)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCM saat mengirim ke Android)	`AWS.SNS.MOBILE.FCM.TTL`
Layanan Notifikasi Push Windows (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Setiap layanan notifikasi push menangani TTL secara berbeda. Amazon SNS memberikan tampilan abstrak TTL di semua layanan notifikasi push, yang memudahkan untuk menentukan TTL. Saat Anda menggunakan AWS Management Console untuk menentukan TTL (dalam detik), Anda hanya perlu memasukkan nilai TTL sekali dan Amazon SNS kemudian akan menghitung TTL untuk setiap layanan pemberitahuan push yang dipilih saat menerbitkan pesan.

TTL relatif terhadap waktu publikasi. Sebelum menyerahkan pesan notifikasi push ke layanan notifikasi push tertentu, Amazon SNS menghitung waktu diam (waktu antara stempel waktu publikasi dan sesaat sebelum menyerahkan ke layanan notifikasi push) untuk notifikasi push dan meneruskan TTL yang tersisa ke spesifik layanan notifikasi push. Jika TTL lebih pendek dari waktu diam, Amazon SNS tidak akan mencoba untuk memublikasikan.

Jika Anda menentukan TTL untuk pesan notifikasi push, maka nilai TTL harus berupa bilangan bulat positif, kecuali nilai `0` memiliki arti khusus untuk layanan notifikasi push—seperti dengan APN dan FCM (saat mengirim ke Android). Jika nilai TTL diatur ke `0` dan layanan notifikasi push tidak memiliki arti khusus untuk `0`, maka Amazon SNS akan menghapus pesan tersebut. Untuk informasi selengkapnya tentang parameter TTL yang disetel ke `0` saat menggunakan APN, lihat Tabel A-3 Pengidentifikasi item untuk notifikasi jarak jauh dalam dokumentasi API Penyedia Biner.

Urutan prioritas untuk menentukan TTL

Prioritas yang digunakan Amazon SNS untuk menentukan TTL untuk pesan notifikasi push didasarkan pada urutan berikut, di mana angka terendah memiliki prioritas tertinggi:

1. TTL atribut pesan
2. TTL isi pesan
3. TTL default layanan notifikasi push (bervariasi per layanan)
4. TTL default Amazon SNS (4 minggu)

Jika Anda menetapkan nilai TTL yang berbeda (satu di atribut pesan dan lainnya di isi pesan) untuk pesan yang sama, maka Amazon SNS akan memodifikasi TTL di isi pesan agar sesuai dengan TTL yang ditentukan dalam atribut pesan.

Menentukan TTL menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
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 Detail pesan, masukkan TTL (jumlah detik yang dimiliki layanan notifikasi push untuk mengirimkan pesan ke endpoint).
6. Pilih Publish message (Publikasikan pesan).

Wilayah yang Didukung untuk aplikasi seluler

Saat ini, Anda dapat membuat aplikasi seluler di Wilayah berikut:

- AS Timur (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Jakarta)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Milan)
- Europe (Paris)
- Europe (Stockholm)
- Middle East (Bahrain)

- Middle East (UAE)
- South America (São Paulo)
- AWS GovCloud(AS-Barat)

Praktik terbaik pemberitahuan push seluler

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 dengan APN setiap kali aplikasi Anda diluncurkan.

Karena token perangkat tidak akan berubah setiap kali aplikasi dibuka oleh pengguna, `CreatePlatformEndpoint`API idempoten dapat 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, lihat. Manajemen titik akhir Firebase Cloud Messaging (FCM)

Pencatatan status pengiriman

Untuk memantau status pengiriman pemberitahuan push, kami sarankan Anda mengaktifkan pencatatan status pengiriman untuk aplikasi platform Amazon SNS 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 topik Amazon SNS untuk pemberitahuan push? .

Pemberitahuan peristiwa

Untuk mengelola titik akhir dengan cara yang didorong oleh acara, Anda dapat menggunakan fungsionalitas pemberitahuan acara. Hal ini memungkinkan topik Amazon SNS yang dikonfigurasi

untuk mengobarkan peristiwa ke pelanggan seperti fungsi Lambda, untuk peristiwa aplikasi platform pembuatan titik akhir, penghapusan, pembaruan, dan kegagalan pengiriman.

Pemberitahuan email

Halaman ini menjelaskan cara berlangganan alamat email ke topik Amazon SNS menggunakan AWS Management Console, AWS SDK for Java, atau. AWS SDK for .NET

> **Catatan**
>
> - Anda tidak dapat mengkustomisasi isi pesan e-mail. 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 di-throttling berdasarkan Kuota Amazon SNS.

> **Important**
>
> Untuk mencegah penerima milis berhenti berlangganan semua penerima dari email topik Amazon SNS, lihat Mengatur langganan email yang memerlukan autentikasi untuk berhenti berlangganan dari Support. AWS

Untuk berlangganan alamat email ke topik Amazon SNS menggunakan AWS Management Console

1. Masuk ke konsol Amazon SNS.
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 sebuah 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 langganan Amazon SNS.

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

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 Antrean surat mati Amazon SNS (DLQs).

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 Confirm subscription (Konfirmasi berlangganan) di email dari Amazon SNS.
2. Amazon SNS membuka browser web Anda dan menampilkan konfirmasi berlangganan beserta ID langganan Anda.

Untuk berlangganan alamat email ke topik Amazon SNS menggunakan SDK AWS

Untuk menggunakan AWS SDK, Anda harus mengonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi AWS SDK 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 mengatur 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 detail API, lihat Berlangganan di Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 antrian SQS 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 detail API, lihat Berlangganan di Referensi AWS SDK for C++ API.

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 detail API, lihat Berlangganan di Referensi AWS CLI Perintah.

Go

SDK for Go V2

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

Berlangganan antrian ke topik dengan filter opsional.

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
```

```
}
```

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

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 titik akhir HTTP 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 detail API, lihat Berlangganan di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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 antrian SQS 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, silakan lihat Panduan Developer AWS SDK for JavaScript.

- Untuk detail API, lihat Berlangganan di Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Berlangganan di AWS SDK untuk referensi API Kotlin.

PHP

SDK for PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 titik akhir HTTP 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 detail API, lihat Berlangganan di Referensi AWS SDK for PHP API.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Berlangganan di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk detail API, lihat Berlangganan di Referensi AWS SDK for Ruby API.

Rust

SDK for Rust

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Berlangganan di AWS SDK untuk referensi Rust API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Berlangganan di AWS SDK untuk referensi API SAP ABAP.

Contoh kode untuk Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menggunakan Amazon SNS dengan kit pengembangan AWS perangkat lunak (SDK).

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks pada skenario terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Contoh lintas layanan adalah contoh aplikasi yang bekerja di beberapa Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai

Halo Amazon SNS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SNS.

.NET

AWS SDK for .NET

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

```
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 detail API, lihat ListTopicsdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

Kode untuk file CMake MakeLists C.txt.

```
# 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 detail API, lihat ListTopicsdi Referensi AWS SDK for C++ API.

Go

SDK untuk Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
```

```
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat ListTopicsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

Inisialisasi klien SNS 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 detail API, lihat ListTopicsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat ListTopicsdi AWS SDK untuk referensi API Kotlin.

Contoh kode

- Tindakan untuk Amazon SNS menggunakan SDK AWS
  - 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 atau CLI

  - 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
- Skenario untuk Amazon SNS menggunakan SDK AWS
  - Membuat titik akhir platform untuk notifikasi push Amazon SNS menggunakan SDK AWS
  - Membuat dan memublikasikan ke topik FIFO Amazon SNS menggunakan SDK AWS
  - Mempublikasikan pesan SMS ke topik Amazon SNS menggunakan SDK AWS
  - Publikasikan pesan besar ke Amazon SNS dengan Amazon S3 menggunakan SDK AWS
  - Mempublikasikan pesan teks SMS Amazon SNS menggunakan SDK AWS
  - Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS
- Contoh tanpa server untuk Amazon SNS menggunakan SDK AWS
  - Memanggil fungsi Lambda dari pemicu Amazon SNS
- Contoh lintas layanan untuk Amazon AWS SNS menggunakan SDK
  - Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB
  - Membangun aplikasi terbitkan dan berlangganan yang menerjemahkan pesan
  - Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label
  - Membuat aplikasi penjelajah Amazon Textract
  - Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS
  - Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS
  - Menggunakan API Gateway untuk menginvokasi fungsi Lambda
  - Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda

Tindakan untuk Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara melakukan tindakan Amazon SNS individual dengan AWS SDK. Kutipan ini memanggil Amazon SNS API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi API Amazon Simple Notification Service (Amazon SNS).

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 atau CLI
- 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

```
   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 detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk memeriksa pesan SMS opt-out untuk nomor telepon

`check-if-phone-number-is-opted-out`Contoh berikut memeriksa apakah nomor telepon yang ditentukan dipilih untuk tidak menerima pesan SMS dari AWS akun saat ini.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Untuk detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for JavaScript API.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for PHP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 topik SNS 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
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

Output:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Untuk detail API, lihat ConfirmSubscriptiondi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat ConfirmSubscriptiondi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat ConfirmSubscriptiondi Referensi AWS SDK for JavaScript API.

PHP

SDK untuk PHP

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

```
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 detail API, lihat ConfirmSubscriptiondi Referensi AWS SDK for PHP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 topik FIFO
- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

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 FIFO dan de-duplikasi tertentu.

```
    /// <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 detail API, lihat CreateTopicdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat CreateTopicdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk membuat topik SNS

`create-topic`Contoh berikut membuat topik SNS 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 detail API, lihat CreateTopicdi Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

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

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
```

```
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat CreateTopicdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat CreateTopicdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat CreateTopicdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat CreateTopicdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat CreateTopicdi Referensi AWS SDK for Ruby API.

Rust

SDK untuk Rust

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

```
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 detail API, lihat CreateTopicreferensi AWS SDK for Rust API.

SAP ABAP

SDK untuk SAP ABAP

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

```
   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 detail API, lihat CreateTopicdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

Hapus topik berdasarkan topiknya ARN.

```
/// <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 detail API, lihat DeleteTopicdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat DeleteTopicdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk menghapus topik SNS

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

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

Perintah ini tidak menghasilkan output.

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

Go

SDK untuk Go V2

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

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



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

```
 }
 return err
}
```

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat DeleteTopicdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DeleteTopicdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat DeleteTopicdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat DeleteTopicdi Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat DeleteTopicdi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat DeleteTopicdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`GetSMSAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`GetSMSAttributes`.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat GetSmSatTributes di Referensi API.AWS SDK for C++

CLI

AWS CLI

Untuk mencantumkan atribut pesan SMS default

`get-sms-attributes`Contoh berikut mencantumkan atribut default untuk mengirim pesan SMS.

```
aws sns get-sms-attributes
```

Output:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Untuk detail API, lihat GetSmSatTributes di Referensi Perintah.AWS CLI

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat GetSmSatTributes di Referensi API.AWS SDK for Java 2.x

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat GetSmSatTributes di Referensi API.AWS SDK for JavaScript

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat GetSmSatTributes di Referensi API.AWS SDK for PHP

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

```
    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 detail API, lihat GetTopicAttributesdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat GetTopicAttributesdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk mengambil atribut dari suatu 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 detail API, lihat GetTopicAttributesdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat GetTopicAttributesdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat GetTopicAttributesdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

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

Mengimpor modul SDK dan klien dan memanggil API.

```
// 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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat GetTopicAttributesdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat GetTopicAttributesdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
$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 detail API, lihat GetTopicAttributesdi Referensi AWS SDK for PHP API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat GetTopicAttributesdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan `ListPhoneNumbersOptedOut` dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Untuk membuat daftar opt-out pesan SMS

`list-phone-numbers-opted-out`Contoh berikut mencantumkan nomor telepon yang dipilih untuk tidak menerima pesan SMS.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Untuk detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS SDK for Java 2.x API.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS SDK for PHP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

```
    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 detail API, lihat ListSubscriptionsdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat ListSubscriptionsdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk membuat daftar langganan SNS Anda

`list-subscriptions`Contoh berikut menampilkan daftar langganan SNS di akun Anda AWS .

```
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 detail API, lihat ListSubscriptionsdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat ListSubscriptionsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat ListSubscriptionsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat ListSubscriptionsdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat ListSubscriptionsdi Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat ListSubscriptionsdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat ListSubscriptionsdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

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

```
   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 detail API, lihat ListSubscriptionsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

```
   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 detail API, lihat ListTopicsdi Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat ListTopicsdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk membuat daftar topik SNS Anda

`list-topics`Contoh berikut mencantumkan semua topik SNS di AWS akun Anda.

```
aws sns list-topics
```

Output:

```
{
```

```
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Untuk detail API, lihat ListTopicsdi Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
```

```
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat ListTopicsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat ListTopicsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat ListTopicsdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat ListTopicsdi Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat ListTopicsdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk detail API, lihat ListTopicsdi Referensi AWS SDK for Ruby API.

Rust

SDK untuk Rust

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

```
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 detail API, lihat ListTopicsreferensi AWS SDK for Rust API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat ListTopicsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 topik FIFO
- Publikasikan pesan teks SMS
- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

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 detail API, lihat Publikasikan di Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat Publikasikan di Referensi AWS SDK for C++ API.

CLI

AWS CLI

Contoh 1: Untuk mempublikasikan pesan ke topik

`publish`Contoh berikut menerbitkan pesan yang ditentukan ke topik SNS 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 pesan SMS 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 detail API, lihat Menerbitkan di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

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

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



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

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

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat Publikasikan di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, deduplicationId, choices;

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

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

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

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

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

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

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

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat Menerbitkan di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for PHP API.

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 detail API, lihat Menerbitkan di Referensi AWS Tools for PowerShell Cmdlet.

Python

SDK untuk Python (Boto3)

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

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 detail API, lihat Menerbitkan di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for Ruby API.

Rust

SDK untuk Rust

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

```
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 detail API, lihat Menerbitkan di AWS SDK untuk referensi API Rust.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat Publikasikan di AWS SDK untuk referensi API SAP ABAP.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`SetSMSAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`SetSMSAttributes`.

C++

SDK untuk C++

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

Cara menggunakan Amazon SNS untuk mengatur atribut defaultsMStype.

```
//! 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 detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for C++

CLI

AWS CLI

Untuk mengatur atribut pesan SMS

`set-sms-attributes`Contoh berikut menetapkan ID pengirim default untuk pesan SMS ke`MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat SetSmSatTributes di Referensi Perintah.AWS CLI

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for Java 2.x

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for JavaScript

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat SetSmSatTributes di Referensi API.AWS SDK for PHP

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 langganan SQS.

```
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 langganan SQS.

```
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 langganan SQS.

```
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 detail API, lihat SetSubscriptionAttributesdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat SetSubscriptionAttributesdi Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat SetSubscriptionAttributesdi AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`SetSubscriptionAttributesRedrivePolicy`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK for Java 1.x

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

```
// 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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 detail API, lihat SetTopicAttributesdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat SetTopicAttributesdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat SetTopicAttributesdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat SetTopicAttributesdi AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

```
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 detail API, lihat SetTopicAttributesdi Referensi AWS SDK for PHP API.

Ruby

SDK untuk Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat SetTopicAttributesdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat SetTopicAttributesdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 topik FIFO
- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

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 detail API, lihat Berlangganan di Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

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 antrian SQS 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 detail API, lihat Berlangganan di Referensi AWS SDK for C++ API.

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 detail API, lihat Berlangganan di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

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

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(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
```

```
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

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

Java

SDK untuk Java 2.x

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

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 titik akhir HTTP 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 detail API, lihat Berlangganan di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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 antrian SQS 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 detail API, lihat Berlangganan di Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

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 detail API, lihat Berlangganan di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

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

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 titik akhir HTTP 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 detail API, lihat Berlangganan di Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

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 detail API, lihat Berlangganan di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

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

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 detail API, lihat Berlangganan di Referensi AWS SDK for Ruby API.

Rust

SDK untuk Rust

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

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 detail API, lihat Berlangganan di AWS SDK untuk referensi Rust API.

SAP ABAP

SDK untuk SAP ABAP

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

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 detail API, lihat Berlangganan di AWS SDK untuk referensi API SAP ABAP.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 detail API, lihat TagResourcedi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat TagResourcedi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat TagResourcedi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 mengatur dan menjalankannya di AWS Repositori Contoh Kode.

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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
//! 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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for C++ API.

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 detail API, lihat Berhenti berlangganan di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

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

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({});
```

Mengimpor modul SDK dan klien dan memanggil API.

```
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, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat Berhenti berlangganan di Referensi AWS SDK for JavaScript API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat Berhenti berlangganan di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Berhenti berlangganan di Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat Berhenti berlangganan di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

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

```
    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 detail API, lihat Berhenti berlangganan di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Skenario untuk Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Amazon SNS dengan AWS SDK. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Amazon SNS. Setiap skenario menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan kode.

Contoh

- Membuat titik akhir platform untuk notifikasi push Amazon SNS menggunakan SDK AWS
- Membuat dan memublikasikan ke topik FIFO Amazon SNS menggunakan SDK AWS
- Mempublikasikan pesan SMS ke topik Amazon SNS menggunakan SDK AWS
- Publikasikan pesan besar ke Amazon SNS dengan Amazon S3 menggunakan SDK AWS
- Mempublikasikan pesan teks SMS Amazon SNS menggunakan SDK AWS
- Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS

Membuat titik akhir platform untuk notifikasi push Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara membuat titik akhir platform untuk notifikasi push Amazon SNS.

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

SDK untuk Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
```

```
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Membuat dan memublikasikan ke topik FIFO Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara membuat dan mempublikasikan ke topik FIFO Amazon SNS.

Java

SDK untuk Java 2.x

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

Contoh ini

- membuat topik Amazon SNS FIFO, dua antrian FIFO Amazon SQS, 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 detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - CreateTopic
  - Publikasikan
  - Berlangganan

Python

SDK untuk Python (Boto3)

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

Buat topik Amazon SNS FIFO, berlangganan Amazon SQS FIFO dan antrian standar ke topik tersebut, 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 detail API, lihat topik berikut ini adalah Referensi API SDK untuk Python (Boto3)AWS .
  - CreateTopic
  - Publikasikan
  - Berlangganan

SAP ABAP

SDK untuk SAP ABAP

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

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

```
    " 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 detail API, lihat topik berikut di referensi API SDK untuk SAP ABAP AWS .
  - CreateTopic
  - Publikasikan
  - Berlangganan

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mempublikasikan pesan SMS ke topik Amazon SNS menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Buat topik Amazon SNS.
- Berlangganan nomor telepon ke topik.
- Publikasikan pesan SMS ke topik sehingga semua nomor telepon berlangganan menerima pesan sekaligus.

Java

SDK untuk Java 2.x

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

Buat topik dan kembalikan ARN-nya.

```
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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Publikasikan pesan besar ke Amazon SNS dengan Amazon S3 menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mempublikasikan pesan besar ke Amazon SNS menggunakan Amazon S3 untuk menyimpan muatan pesan.

Java

SDK for Java 1.x

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

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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mempublikasikan pesan teks SMS Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mempublikasikan pesan SMS menggunakan Amazon SNS.

.NET

AWS SDK for .NET

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

```
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 detail API, lihat Publikasikan di Referensi AWS SDK for .NET API.

C++

SDK untuk C++

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

```
/**
 * 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 detail API, lihat Publikasikan di Referensi AWS SDK for C++ API.

Java

SDK untuk Java 2.x

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

```
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 detail API, lihat Publikasikan di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

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

```
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 detail API, lihat Menerbitkan di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat Publikasikan di Referensi AWS SDK for PHP API.

Python

SDK untuk Python (Boto3)

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

```
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 detail API, lihat Menerbitkan di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Buat topik (FIFO atau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.
- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

.NET

AWS SDK for .NET

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

Jalankan skenario interaktif di penggugah/prompt perintah.

```
/// <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 operasi Amazon SQS.

```
/// <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 operasi Amazon SNS.

```
/// <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 detail API, lihat topik berikut di Referensi API AWS SDK for .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- Publikasikan
- ReceiveMessage
- SetQueueAttributes
- Berlangganan
- Berhenti berlangganan

C++

SDK untuk C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur 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 detail API, lihat topik berikut di Referensi API AWS SDK for C++ .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publikasikan
  - ReceiveMessage
  - SetQueueAttributes

- Berlangganan
- Berhenti berlangganan

Go

SDK untuk Go V2

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

Jalankan skenario interaktif di penggugah/prompt perintah.

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic() (string, string, bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
```

```
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}
```

```
func (runner ScenarioRunner) CreateQueue(ordinal string, isFifoTopic bool)
 (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
```

```
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
   for askAboutTones {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelections = append(toneSelections, ToneChoices[toneIndex])
    askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
 the filter? (y/n) ", "y")
   }
   log.Printf("Your subscription will be filtered to only pass the following
 tones: %v\n", toneSelections)
   filterPolicy = map[string][]string{TONE_KEY: toneSelections}
  }
 }

 subscriptionArn, err := runner.snsActor.SubscribeQueue(topicArn, queueArn,
 filterPolicy)
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(topicArn string, isFifoTopic bool,
 contentBasedDeduplication bool, usingFilters bool) {
```

```
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
```

```
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
```

```
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor
```

```
 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic()
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup()
 }

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

Tentukan struct yang membungkus tindakan Amazon SNS 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(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
```

```
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Tentukan struct yang membungkus tindakan Amazon SQS 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(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
```

```
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(queueUrl string, queueArn string,
 topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(context.TODO(),
 &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}
```

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}
```

```
// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(queueUrl string, messages []types.Message)
 error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(context.TODO(),
 &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Publikasikan
- ReceiveMessage
- SetQueueAttributes
- Berlangganan
- Berhenti berlangganan

JavaScript

SDK untuk JavaScript (v3)

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

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";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

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.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
```

```
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
```

```
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
```

```
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

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

    let groupId, deduplicationId, choices;

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

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

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

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

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

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

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

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

```
    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
```

```
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publikasikan
  - ReceiveMessage
  - SetQueueAttributes
  - Berlangganan
  - Berhenti berlangganan

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh tanpa server untuk Amazon SNS menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menggunakan Amazon SNS dengan AWS SDK.

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 topik SNS. 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 acara SNS dengan Lambda menggunakan.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

SDK untuk Go V2

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

Mengkonsumsi acara SNS dengan Lambda menggunakan Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
```

```
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK untuk Java 2.x

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

Mengkonsumsi acara SNS dengan Lambda menggunakan 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

SDK untuk JavaScript (v3)

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

Mengkonsumsi acara SNS dengan JavaScript Lambda menggunakan.

```
// 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 acara SNS dengan TypeScript Lambda menggunakan.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK untuk PHP

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

Mengkonsumsi acara SNS dengan Lambda menggunakan 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

SDK untuk Python (Boto3)

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

Mengkonsumsi acara SNS dengan Lambda menggunakan 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

SDK untuk Ruby

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

Mengkonsumsi acara SNS dengan Lambda menggunakan 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

SDK untuk Rust

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

Mengkonsumsi acara SNS dengan Lambda menggunakan 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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh lintas layanan untuk Amazon AWS SNS menggunakan SDK

Contoh aplikasi berikut menggunakan AWS SDK untuk menggabungkan Amazon SNS dengan yang lain. Layanan AWS Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan aplikasi.

Contoh

- Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB
- Membangun aplikasi terbitkan dan berlangganan yang menerjemahkan pesan
- Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label
- Membuat aplikasi penjelajah Amazon Textract
- Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS
- Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS
- Menggunakan API Gateway untuk menginvokasi 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

SDK untuk Java 2.x

Menunjukkan cara membuat aplikasi web dinamis yang mengirimkan data menggunakan API Java Amazon DynamoDB dan mengirim pesan teks menggunakan API Java Amazon Simple Notification Service.

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

SDK untuk JavaScript (v3)

Contoh ini menunjukkan cara membangun 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

SDK untuk Kotlin

Menunjukkan cara membuat aplikasi Android native yang mengirimkan data menggunakan API Kotlin Amazon DynamoDB dan mengirim pesan teks menggunakan API Kotlin Amazon SNS.

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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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 .NET 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.

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

Java

SDK untuk 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 Async API, lihat contoh lengkapnya di. GitHub

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

Kotlin

SDK untuk 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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK untuk 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

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK untuk 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

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK untuk 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

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK untuk 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

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK untuk 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

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK untuk 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

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Membuat aplikasi penjelajah Amazon Textract

Contoh kode berikut ini menunjukkan cara menjelajahi output Amazon Textract melalui aplikasi interaktif.

JavaScript

SDK untuk 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. Contoh ini menggunakan Amazon Simple Storage Service (Amazon S3) untuk penyimpanan, dan untuk notifikasi, contoh ini mengambil polling antrean Amazon Simple Queue Service (Amazon SQS) 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

SDK untuk 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 API asinkron untuk memulai tugas yang menerbitkan notifikasi ke topik Amazon Simple Notification Service (Amazon SNS) saat tugas selesai.
- Lakukan polling pada antrean Amazon Simple Queue Service (Amazon SQS) untuk mendapatkan pesan penyelesaian tugas 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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mendeteksi orang dan objek dalam video dengan Amazon Rekognition.

Python

SDK untuk 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 (Amazon SNS) saat pekerjaan selesai dan berlangganan antrian Amazon Simple Queue Service (Amazon SQS) 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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Buat topik (FIFO atau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.
- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

Java

SDK untuk Java 2.x

Mendemonstrasikan pesan dengan topik dan antrian menggunakan Amazon Simple Notification Service (Amazon SNS) dan Amazon Simple Queue Service (Amazon SQS).

Untuk kode sumber lengkap dan instruksi yang mendemonstrasikan pesan dengan topik dan antrian di Amazon SNS dan Amazon SQS, lihat contoh lengkapnya di. GitHub

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon SQS

Kotlin

SDK untuk Kotlin

Mendemonstrasikan pesan dengan topik dan antrian menggunakan Amazon Simple Notification Service (Amazon SNS) dan Amazon Simple Queue Service (Amazon SQS).

Untuk kode sumber lengkap dan instruksi yang mendemonstrasikan pesan dengan topik dan antrian di Amazon SNS dan Amazon SQS, lihat contoh lengkapnya di. GitHub

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon SQS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Menggunakan API Gateway untuk menginvokasi fungsi Lambda

Contoh kode berikut menunjukkan cara membuat AWS Lambda fungsi yang dipanggil oleh Amazon API Gateway.

Java

SDK untuk Java 2.x

Menunjukkan cara membuat AWS Lambda fungsi dengan menggunakan Lambda Java runtime API. Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat fungsi Lambda yang diinvokasi oleh Amazon API Gateway yang memindai peringatan hari jadi kerja di tabel Amazon DynamoDB dan menggunakan Amazon Simple Notification Service (Amazon SNS) untuk mengirim pesan teks berisi ucapan selamat kepada karyawan Anda pada tanggal hari jadi kerja 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

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara membuat AWS Lambda fungsi dengan menggunakan API JavaScript runtime Lambda. Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat fungsi Lambda yang diinvokasi oleh Amazon API Gateway yang memindai peringatan hari jadi kerja di tabel Amazon DynamoDB dan menggunakan Amazon Simple Notification Service (Amazon SNS) untuk mengirim pesan teks berisi ucapan selamat kepada karyawan Anda pada tanggal hari jadi kerja 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

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK 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

SDK untuk 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 menggunakan API runtime Java Lambda. 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

SDK untuk 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 menggunakan API runtime JavaScript Lambda. 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 pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon SNS dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Keamanan Amazon SNS

Bagian ini menyediakan informasi tentang keamanan, autentikasi dan kontrol akses Amazon SNS, serta Bahasa Kebijakan Akses Amazon SNS.

Topik

- Perlindungan data
- Identity and access management di Amazon SNS
- Pencatatan dan Pemantauan di Amazon SNS
- Validasi Kepatuhan untuk Amazon SNS
- Ketahanan di Amazon SNS
- Keamanan infrastruktur di Amazon SNS
- Praktik terbaik keamanan Amazon SNS

Perlindungan data

Model tanggung jawab bersama AWS berlaku untuk perlindungan data di Layanan Notifikasi Sederhana Amazon. Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk memberikan perlindungan terhadap infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Konten ini mencakup konfigurasi keamanan dan tugas manajemen untuk layanan AWS yang Anda gunakan. Untuk informasi lebih lanjut tentang privasi data, lihat FAQ tentang Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya Anda melindungi kredensial Akun AWS dan menyiapkan akun pengguna individu dengan AWS Identity and Access Management (IAM). Dengan cara seperti itu, 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 autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk melakukan komunikasi dengan sumber daya AWS. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru.

- Siapkan API dan log aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama dengan semua kontrol keamanan default dalam layanan AWS.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon Simple Storage Service (Amazon S3).
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi lebih lanjut tentang titik akhir FIPS yang tersedia, lihat Standar Pemrosesan Informasi Federal (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
  - Menyediakan 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. Hal ini termasuk ketika Anda bekerja dengan Amazon SNS atau Layanan Amazon Web Services lainnya dengan menggunakan konsol tersebut, API,AWS CLI, atauAWSSDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, sebaiknya Anda tidak menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Bagian berikut ini memberikan informasi tambahan tentang perlindungan data di Amazon SNS.

Topik

- Enkripsi data
- Privasi lalu lintas jaringan Internet
- Keamanan Perlindungan Data Pesan

Enkripsi data

Perlindungan data mengacu pada perlindungan data saat dalam transit (saat melakukan perjalanan ke dan dari Amazon SNS) dan saat diam (sementara data disimpan di dalam disk di pusat data Amazon SNS). Anda dapat melindungi data saat transit menggunakan Secure Socket Layer (SSL) atau enkripsi di 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

- Enkripsi diam
- Manajemen kunci
- Mengaktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS
- Mengaktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS dengan antrean Amazon SQS terenkripsi berlangganan

Enkripsi diam

Enkripsi sisi server (SSE) memungkinkan Anda menyimpan data sensitif dalam topik terenkripsi dengan melindungi konten pesan dalam topik Amazon SNS menggunakan kunci yang dikelola di (). AWS Key Management Service AWS KMS

SSE mengenkripsi olahpesan 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 menetapkan atribut `KmsMasterKeyId` menggunakan tindakan API `CreateTopic` dan `SetTopicAttributes`), lihat Mengaktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS.
- Untuk informasi tentang membuat topik terenkripsi menggunakan AWS CloudFormation(dengan menetapkan properti `KmsMasterKeyId` menggunakan properti sumber daya `AWS::SNS::Topic`), lihat Panduan Pengguna AWS CloudFormation.

> **Important**
>
> Semua permintaan untuk topik dengan SSE yang 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 kunci KMS enkripsi simetris. Anda tidak dapat menggunakan jenis kunci KMS lainnya untuk mengenkripsi sumber daya layanan Anda. Untuk bantuan menentukan apakah kunci KMS adalah kunci enkripsi simetris, lihat Mengidentifikasi kunci KMS asimetris.

AWS KMS menggabungkan perangkat keras dan perangkat lunak yang aman dan selalu tersedia untuk menyediakan sistem manajemen kunci yang diskalakan untuk cloud. Saat Anda menggunakan Amazon SNS dengan AWS KMS, kunci datayang 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 kunci KMS terkelola AWS untuk Amazon SNS, yang unik untuk setiap akun dan wilayah.
- Standar keamanan AWS KMS dapat membantu Anda memenuhi persyaratan kepatuhan terkait enkripsi.

Untuk informasi lebih lanjut, lihat Apa yang dimaksud AWS Key Management Service? dalam Panduan Developer AWS Key Management Service.

Topik

- Lingkup enkripsi
- Istilah kunci

Lingkup enkripsi

SSE mengenkripsi isi pesan dalam topik Amazon SNS.

SSE tidak mengenkripsi berikut ini:

- 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 backlogged.
> - Setiap pesan terenkripsi tetap dienkripsi bahkan jika enkripsi topiknya dinonaktifkan.

Istilah kunci

Istilah kunci berikut ini dapat membantu Anda lebih memahami fungsionalitas SSE. Untuk deskripsi detail, lihat Referensi API Layanan Notifikasi Sederhana Amazon.

Kunci data

Kunci enkripsi data (DEK) bertanggung jawab untuk mengenkripsi isi pesan Amazon SNS.

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 keyID

Alias, alias ARN, ID kunci, atau ARN kunci dari, atau kustom AWS KMS —di AWS KMS key akun Anda atau di akun lain. Sementara alias yang AWS dikelola AWS KMS untuk Amazon SNS `alias/aws/sns` selalu, alias AWS KMS kustom dapat, misalnya, menjadi. `alias/MyAlias` Anda dapat menggunakan AWS KMS kunci ini untuk melindungi pesan dalam topik Amazon SNS.

> **Note**
>
> Ingatlah hal-hal berikut ini:
>
> - Pertama kali Anda menggunakan AWS Management Console untuk menentukan KMS AWS terkelola untuk Amazon SNS untuk suatu topikAWS KMS, membuat AWS KMS terkelola untuk Amazon SNS.

- Atau, saat pertama kali Anda menggunakan `Publish` tindakan pada topik dengan SSE diaktifkan, AWS KMS membuat KMS AWS terkelola untuk Amazon SNS.

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 Referensi AWS Key Management Service API. Untuk informasi tentang menemukan AWS KMS pengenal, lihat Menemukan ID Kunci dan ARN di Panduan AWS Key Management ServicePengembang.

Important

Tidak ada biaya tambahan untuk penggunaan AWS KMS. Untuk informasi selengkapnya, lihat Memperkirakan biaya AWS KMS dan Harga AWS Key Management Service.

Manajemen kunci

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 kunci KMS enkripsi simetris. Anda tidak dapat menggunakan jenis kunci KMS lainnya untuk mengenkripsi sumber daya layanan Anda. Untuk bantuan menentukan apakah kunci KMS adalah kunci enkripsi simetris, lihat Mengidentifikasi kunci KMS asimetris.

Topik

- Memperkirakan biaya AWS KMS
- Mengonfigurasi izin AWS KMS
- Kesalahan AWS KMS

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 memberikan ide yang sangat baik dari biaya yang diharapkan, biaya yang sebenarnya mungkin lebih tinggi karena sifat terdistribusi Amazon SNS.

Untuk menghitung jumlah permintaan API (R) per topikgunakan rumus berikut ini:

```
R = B / D * (2 * P)
```

B adalah periode penagihan (dalam detik).

D adalah periode penggunaan kembali kunci data (dalam hitungan detik—Amazon SNS menggunakan kembali kunci data hingga 5 menit).

P adalah jumlah prinsipal penerbitan yang dikirim ke topik Amazon SNS.

Berikut ini adalah contoh perhitungan. Untuk informasi harga sebenranya, lihat Harga AWS Key Management Service.

Contoh 1: Menghitung jumlah panggilan API AWS KMS 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 panggilan API AWS KMS 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
```

Mengonfigurasi izin AWS KMS

Sebelum Anda dapat menggunakan SSE, Anda harus mengonfigurasi AWS KMS key kebijakan untuk mengizinkan enkripsi topik dan enkripsi dan dekripsi pesan. Untuk contoh dan informasi selengkapnya tentang izin AWS KMS, lihatIzin API AWS KMS: Tindakan dan Referensi Sumber Daya dalam Panduan Developer AWS Key Management Service. Untuk detail tentang cara mengatur topik Amazon SNS dengan enkripsi sisi server, lihat. Menyiapkan topik Amazon SNS dengan enkripsi sisi server

> **Note**
>
> Anda juga dapat mengelola izin untuk kunci KMS enkripsi simetris menggunakan kebijakan IAM. Untuk informasi selengkapnya, lihat Menggunakan kebijakan IAM dengan AWS KMS. Meskipun Anda dapat mengonfigurasi izin global untuk mengirim dan menerima dari Amazon SNSAWS KMS, memerlukan secara eksplisit penamaan ARN lengkap KMS di wilayah tertentu di bagian kebijakan IAM. `Resource`

Anda juga harus memastikan bahwa kebijakan utama AWS KMS key mengizinkan izin yang diperlukan. Untuk melakukannya, beri nama utama yang menghasilkan dan menggunakan pesan terenkripsi di Amazon SNS sebagai pengguna dalam kebijakan kunci KMS.

Atau, Anda dapat menentukan AWS KMS tindakan yang diperlukan dan KMS ARN dalam kebijakan IAM yang ditetapkan ke prinsipal yang menerbitkan 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 topik Amazon SNS Anda dan Anda menggunakan alias untuk mengontrol akses ke kunci KMS menggunakan kebijakan IAM atau kebijakan kunci KMS

dengan 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 kunci KMS, lihat Menggunakan alias untuk mengontrol akses ke kunci KMS di Panduan Pengembang. AWS Key Management Service

Mengizinkan 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 layanan AWS dan topik terenkripsi

Beberapa layanan AWS menerbitkan peristiwa untuk topik Amazon SNS. 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 tentang Membuat Kunci dalam AWS Key Management ServicePanduan Developer.
2. Untuk mengizinkan AWS layanan memiliki `kms:GenerateDataKey*` dan `kms:Decrypt` izin, tambahkan pernyataan berikut ke kebijakan KMS.

   ```
   {
     "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
Pergeseran Merah Amazon	redshift.amazonaws.com
Amazon RDS	events.rds.amazonaws.com
Gletser Amazon S3	glacier.amazonaws.com
Layanan Email Sederhana Amazon	ses.amazonaws.com
Layanan Penyimpanan Sederhana Amazon	s3.amazonaws.com
AWS Snowball	importexport.amazonaws.com

Sumber peristiwa	Prinsipal layanan
AWSManajer Insiden Systems Manager	AWSSystems Manager Incident Manager terdiri dari dua prinsip layanan: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Beberapa sumber acara Amazon SNS mengharuskan Anda untuk memberikan peran IAM (bukan prinsip layanan) dalam kebijakan: AWS KMS key
>
> - Auto Scaling Amazon EC2
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. Tambahkan kunci `aws:SourceAccount` dan `aws:SourceArn` kondisi ke kebijakan sumber daya KMS untuk lebih melindungi kunci KMS 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 topik yang EventBridge dienkripsi.

```
{
  "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. Aktifkan SSE untuk topik Anda menggunakan KMS Anda.
5. Berikan ARN topik terenkripsi ke sumber peristiwa.

Kesalahan AWS KMS

Ketika Anda bekerja dengan Amazon SNS dan AWS KMS, Anda mungkin mengalami kesalahan. Daftar berikut ini menjelaskan kesalahan dan solusi pemecahan masalah yang dimungkinkan.

KMS AccessDeniedException

Ciphertext merujuk pada kunci yang tidak ada atau bahwa Anda tidak memiliki akses ke padanya.

Kode Status HTTP: 400

KMS DisabledException

Permintaan ditolak karena KMS yang ditentukan tidak diaktifkan.

Kode Status HTTP: 400

KMS InvalidStateException

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.

Kode Status HTTP: 400

KMS NotFoundException

Permintaan ditolak karena entitas atau sumber daya yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

KMS OptInRequired

Access key ID AWS membutuhkan berlangganan untuk layanan.

Kode Status HTTP: 403

KMS ThrottlingException

Permintaan ditolak karena throttling permintaan. Untuk informasi selengkapnya tentang pembatasan, lihat Kuota di Panduan Pengembang. AWS Key Management Service

Kode Status HTTP: 400

Mengaktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS

Dengan enkripsi sisi server (SSE), Anda dapat menyimpan data sensitif dalam topik terenkripsi. SSE melindungi konten pesan dalam topik Amazon SNS menggunakan kunci yang dikelolaAWS Key Management Service(AWS KMS). Untuk informasi selengkapnya tentang enkripsi sisi server dengan Amazon SNS, lihatEnkripsi diam. Untuk selengkapnya tentang membuatAWS KMSkunci-kunci, lihatMembuat kuncidi dalamAWS Key Management ServicePanduan Pengembang.

> **Important**
>
> Semua permintaan untuk topik dengan SSE yang diaktifkan harus menggunakan HTTPS dan Versi Tanda Tangan 4.

Aktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS menggunakanAWS Management Console

1. Masuk ke Konsol Amazon SNS.
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. TentukanAWS KMSkunci. Untuk informasi selengkapnya, lihat Istilah kunci.

Untuk setiap jenis KMS,Deskripsi,Rekening, danKMSditampilkan.

> **Important**
>
> Jika Anda bukan pemilik KMS, atau jika Anda masuk dengan akun yang tidak memilikikms:ListAliasesdankms:DescribeKeyizin, Anda tidak akan dapat melihat informasi tentang KMS di konsol Amazon SNS.
> Minta pemilik KMS untuk memberi Anda izin ini. Untuk informasi selengkapnya, lihat Izin API AWS KMS: Referensi Tindakan dan Sumber Daya dalam Panduan Developer AWS Key Management Service.

- YangAWSKMS terkelola untuk Amazon SNS(Default) alias/aws/snsdipilih secara default.

> **Note**
>
> Ingatlah hal-hal berikut ini:
>
> - Pertama kali Anda menggunakanAWS Management Consoleuntuk menentukanAWSKMS terkelola untuk Amazon SNS untuk suatu topik,AWS KMSmenciptakanAWSKMS terkelola untuk Amazon SNS.
> - Atau, pertama kali Anda menggunakanPublishtindakan pada topik dengan SSE diaktifkan,AWS KMSmenciptakanAWSKMS terkelola untuk Amazon SNS.

- Untuk menggunakan KMS khusus dariAWSakun, pilihKunci KMSbidang dan kemudian pilih KMS kustom dari daftar.

> **Note**
>
> Untuk petunjuk tentang membuat KMSS kustom, lihatMembuat Keysdi dalamAWS Key Management ServicePanduan Pengembang

- Untuk menggunakan ARN KMS khusus dariAWSakun atau dari yang lainAWSakun, masukkan keKunci KMSbidang.

5. Pilih Save changes (Simpan perubahan).

SSE diaktifkan untuk topik Anda dan*MyTopic*halaman ditampilkan.

Status Enkripsi topik, Akun AWS, Kunci utama pelanggan (CMK), ARN CMK, dan Deskripsi ditampilkan pada tab Enkripsi.

Menyiapkan topik Amazon SNS dengan enkripsi sisi server

Saat membuat kunci KMS Anda, gunakan kebijakan kunci KMS 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"
        }
    }
}
```

Mengaktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS dengan antrean Amazon SQS terenkripsi berlangganan

Anda dapat mengaktifkan enkripsi sisi server (SSE) untuk topik untuk melindungi datanya. Untuk mengizinkan Amazon SNS untuk mengirim pesan ke antrean Amazon SQS terenkripsi, kunci terkelola pelanggan yang berkaitan dengan antrean Amazon SQS harus memiliki pernyataan kebijakan yang memberikan akses prinsipal layanan Amazon SNS ke TindakanAWS KMS API`GenerateDataKey` dan`Decrypt`. Untuk informasi selengkapnya tentang menggunakan SSE, lihat Enkripsi diam.

Halaman ini menunjukkan bagaimana Anda dapat mengaktifkan SSE untuk topik Amazon SNS ke mana antrean Amazon SQS terenkripsi dibuat langganan, menggunakan AWS Management Console.

Langkah 1: Membuat kunci KMS kustom

1. Masuk ke konsol AWS KMS dengan pengguna yang memiliki setidaknya kebijakan `AWSKeyManagementServicePowerUser`.
2. Pilih Buat kunci.
3. Untuk membuat kunci KMS enkripsi simetris, untuk Key type pilih Symmetric.

   Untuk informasi tentang cara membuat kunci KMS asimetris diAWS KMS konsol, lihat Membuat tombol KMS asimetris (konsol).
4. Dalam Penggunaan kunci, opsi Enkripsi dan dekripsi dipilih untuk Anda.

   Untuk informasi tentang cara membuat kunci KMS yang menghasilkan dan memverifikasi kode MAC, lihat Membuat kunci HMAC KMS.

   Untuk informasi tentang Opsi lanjutan, lihat Kunci tujuan khusus.
5. Pilih Selanjutnya.
6. Ketikkan alias untuk kunci KMS. Nama alias tidak dapat dimulai dengan **`aws/`**. **`aws/`**Prefiks dicadangkan oleh Amazon Web Services untuk mewakiliKunci yang dikelola AWS di akun Anda.

   > **Note**
   >
   > Menambahkan, menghapus, memperbarui alias dapat mengizinkan atau menolak izin ke kunci KMS. Untuk detailnya, lihat ABAC untukAWS KMS dan Menggunakan alias untuk mengontrol akses ke kunci KMS.

   Alias adalah nama tampilan yang dapat Anda gunakan untuk mengidentifikasi kunci KMS. Sebaiknya Anda memilih alias yang menunjukkan jenis data yang akan Anda lindungi atau aplikasi yang akan Anda gunakan dengan kunci KMS.

   Alias diperlukan saat Anda membuat kunci KMS diAWS Management Console. Mereka adalah opsional ketika Anda menggunakan CreateKeyoperasi.
7. (Opsional) Ketikkan deskripsi untuk kunci KMS.

Anda dapat menambahkan deskripsi sekarang atau memperbaruinya kapan saja kecuali status kunci adalah`Pending Deletion` atau`Pending Replica Deletion`. Untuk menambah, mengubah, edit, edit deskripsi dari kunci terkelola pelanggan, edit deskripsi diAWS Management Console atau gunakan UpdateKeyDescriptionoperasi.

8. (Opsional) Ketik kunci tanda dan nilai tanda opsional. Untuk menambahkan lebih dari satu tanda ke kunci KMS, pilih Tambahkan tanda.

> **Note**
>
> Penandaan atau penghapusan tanda kunci dapat mengizinkan atau menolak izin untuk kunci KMS. Untuk detailnya, lihat ABAC untukAWS KMS dan Menggunakan tag untuk mengontrol akses ke kunci KMS.

Saat Anda menambahkan tanda ke sumber daya AWS Anda, AWS membuat laporan alokasi biaya dengan penggunaan dan biaya yang dikumpulkan oleh tanda. Tanda juga dapat digunakan untuk mengontrol akses ke kunci KMS. Untuk informasi tentang penandaan kunci KMS, lihat Menandai kunci dan ABAC untukAWS KMS.

9. Pilih Selanjutnya.
10. Pilih pengguna dan peran IAM yang dapat mengelola kunci KMS.

> **Note**
>
> Kebijakan kunci ini memberikan kontrolAkun AWS penuh atas kunci KMS ini. Hal ini memungkinkan administrator akun untuk menggunakan kebijakan IAM untuk memberikan izin prinsipal lain untuk mengelola kunci KMS. Untuk detailnya, lihat Kebijakan kunci default.
>
> Praktik terbaik IAM mencegah penggunaan pengguna IAM dengan kredensi jangka panjang. Bila memungkinkan, gunakan peran IAM, yang memberikan kredensi sementara. Untuk detailnya, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

11. (Opsional) Untuk mencegah pengguna dan peran IAM terpilih menghapus kunci ini, di bagian Penghapusan kunci di bawah halaman, kosongkan kotak centang Izinkan administrator kunci untuk menghapus kunci ini.

12. Pilih Selanjutnya.
13. Pilih pengguna dan peran IAM yang dapat menggunakan kunci dalam operasi kriptografi. Pilih Selanjutnya.
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 dikelola pelanggan baru Anda muncul dalam daftar kunci.

Langkah 2: Membuat topik Amazon SNS terenkripsi

1. Masuk ke Konsol Amazon SNS.
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 dikelola pelanggan. Untuk informasi selengkapnya, lihat Istilah kunci.

    Untuk setiap jenis kunci yang dikelola pelanggan, Deskripsi, Akun, dan kunci yang dikelola pelanggan ARN ditampilkan.

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

c. Untuk kunci yang dikelola pelanggan, pilih MyCustomKeyyang Anda buat sebelumnya, lalu pilih Aktifkan enkripsi sisi server.

6. Pilih Save changes (Simpan perubahan).

   SSE diaktifkan untuk topik Anda dan MyTopichalaman ditampilkan.

   Status Enkripsi topik,AWS Akun, kunci yang dikelola pelanggan, ARN kunci yang dikelola pelanggan, dan Deskripsi ditampilkan di tab Enkripsi.

Topik terenkripsi baru Anda muncul dalam daftar topik.

Langkah 3: Membuat dan berlangganan antrean Amazon SQS terenkripsi

1. Masuk ke konsol Amazon SQS.
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. Pada konsol Amazon SQS, pilih `MyEncryptedQueue1` dan `MyEncryptedQueue2` dan kemudian pilih Tindakan Antrean, Berlangganan Antrean ke Topik SNS.
6. Di kotak dialog Berlangganan Topik, 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 OK.

Langkah 4: Terbitkan pesan ke topik terenkripsi Anda

1. Masuk ke Konsol Amazon SNS.
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 konsol Amazon SQS.
2. Dari daftar antrian, pilih MyEncryptedQueue1 lalu pilih Kirim dan terima pesan.
3. Pada halaman Kirim dan terima pesan diMyEncryptedQueue 1 halaman, 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 untuk MyEncryptedQueue2.

Privasi lalu lintas jaringan Internet

Titik akhir Amazon Virtual Private Cloud (Amazon VPC) untuk Amazon SNS adalah entitas logis di dalam VPC yang mengizinkan konektivitas hanya ke Amazon SNS. Rute VPC meminta Amazon SNS dan rute merespons kembali ke VPC. Bagian berikut ini memberikan informasi tentang bekerja dengan VPC endpoint dan membuat kebijakan VPC endpoint.

Jika Anda menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk menghosting sumber daya AWS Anda, Anda dapat membuat koneksi privat antara VPC Anda dan Amazon SNS. Dengan koneksi ini, Anda dapat menerbitkan pesan ke topik Amazon SNS Anda tanpa mengirim mereka melalui internet publik.

Amazon VPC adalah layanan AWS yang dapat Anda gunakan untuk meluncurkan sumber daya AWS dalam jaringan virtual yang Anda tentukan. Dengan VPC, Anda memiliki kendali terhadap pengaturan jaringan Anda, seperti rentang alamat IP, subnet, tabel rute, dan gateway jaringan. Untuk menghubungkan VPC Anda ke Amazon SNS, Anda menentukan VPC endpoint antarmuka. Jenis titik akhir ini memungkinkan Anda untuk menghubungkan VPC Anda ke layanan AWS. Titik akhir memberikan konektivitas yang dapat diandalkan, dapat diskalakan ke Amazon SNS tanpa memerlukan gateway internet, instans terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi selengkapnya, lihat VPC endpoint antarmuka dalam Panduan Pengguna Amazon VPC.

Informasi di bagian ini ditujukan untuk pengguna Amazon VPC. Untuk informasi selengkapnya, dan untuk memulai dengan membuat VPC, lihat Memulai Amazon VPC dalam Panduan Pengguna Amazon VPC.

> **Note**
>
> VPC endpoint tidak mengizinkan Anda untuk berlangganan topik Amazon SNS ke alamat IP privat.

Topik

- Membuat VPC endpoint Amazon untuk Amazon SNS
- Membuat kebijakan VPC endpoint Amazon untuk Amazon SNS
- Menerbitkan pesan Amazon SNS dari Amazon VPC

Membuat VPC endpoint Amazon untuk Amazon SNS

Untuk menerbitkan pesan ke topik Amazon SNS Anda dari Amazon VPC, buat VPC endpoint antarmuka. Kemudian, Anda dapat menerbitkan pesan ke topik Anda sambil menjaga lalu lintas di dalam jaringan yang Anda kelola dengan VPC.

Gunakan informasi berikut ini untuk membuat titik akhir dan menguji koneksi antara VPC Anda dan Amazon SNS. Atau, untuk panduan yang membantu Anda memulai dari scratch, lihat Menerbitkan pesan Amazon SNS dari Amazon VPC.

Membuat titik akhir

Anda dapat membuat endpoint Amazon SNS di VPC menggunakan, SDK AWS Management Console, Amazon SNS AWS API, atau. AWS CLI AWS CloudFormation

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan konsol Amazon VPC atau AWS CLI, lihat Membuat Titik Akhir Antarmuka dalam Panduan Pengguna Amazon VPC.

> **Important**
>
> Anda dapat menggunakan Amazon Virtual Private Cloud hanya dengan titik akhir HTTPS Amazon SNS.
> Ketika Anda membuat titik akhir, tetapkan Amazon SNS sebagai layanan yang ingin Anda hubungkan ke VPC Anda. Di konsol Amazon VPC, 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 Amazon VPC, Anda harus mengaktifkan DNS pribadi dan menentukan titik akhir dalam format. `sns.us-east-2.amazonaws.com`
> DNS pribadi tidak mendukung titik akhir lama seperti atau. `queue.amazonaws.com` `us-east-2.queue.amazonaws.com`

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan AWS CloudFormation, lihat `AWS::EC2::VPCEndpoint`sumber daya di AWS CloudFormation Panduan Pengguna.

Menguji koneksi antara VPC Anda dan Amazon SNS

Setelah Anda membuat titik akhir untuk Amazon SNS, Anda dapat menerbitkan pesan dari VPC Anda ke topik Amazon SNS Anda. Untuk menguji koneksi ini, lakukan hal berikut ini:

1. Connect ke instans Amazon EC2 yang berada di VPC Anda. Untuk informasi tentang menghubungkan, lihat Hubungkan ke Instans Linux Anda atau Connect ke Instans Windows Andadalam dokumentasi Amazon EC2.

   Sebagai contoh, untuk menghubungkan ke instans Linux menggunakan klien SSH, jalankan perintah berikut ini dari terminal:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Di mana:

   - ec2-key-pair.pem adalah file yang berisi pasangan kunci yang disediakan Amazon EC2 ketika Anda membuat instans.
   - instance-hostname adalah nama host publik dari instans. Untuk mendapatkan nama host di konsol Amazon EC2: Pilih Instans, pilih instans Anda, dan temukan nilai untuk DNS Publik (IPv4).
2. Dari instans Anda, gunakan perintah `publish` Amazon SNS dengan 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 konsol Amazon SNS: PilihTopik, temukan topik Anda, dan temukan nilai di kolom ARN.

   Jika pesan berhasil diterima oleh Amazon SNS, 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 endpoint Amazon untuk Amazon SNS di mana Anda menentukan hal berikut ini:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang dapat digunakan untuk mengambil tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan VPC Endpoint dalam Panduan Pengguna Amazon VPC.

Kebijakan VPC endpoint contoh berikut ini menentukan bahwa pengguna `MyUser` diizinkan untuk menerbitkan topik Amazon SNS `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:

- Tindakan Amazon SNS API lainnya, seperti `sns:Subscribe` dan `sns:Unsubscribe`.
- Pengguna IAM dan aturan lainnya yang mencoba untuk menggunakan VPC endpoint ini.
- `MyUser` menerbitkan ke topik Amazon SNS yang berbeda.

> **Note**
>
> Pengguna IAM masih dapat menggunakan tindakan Amazon SNS API lainnya dari luar VPC.

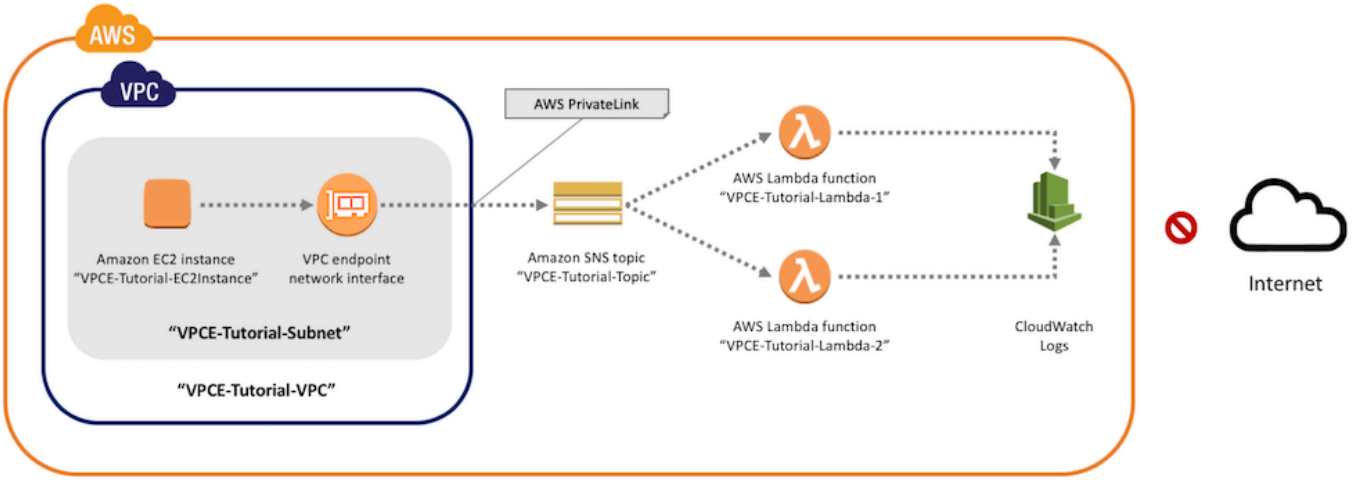
Menerbitkan pesan Amazon SNS dari Amazon VPC

Bagian ini menjelaskan cara untuk menerbitkan ke topik Amazon SNS sambil menjaga pesan aman dalam jaringan privat. Anda menerbitkan pesan dari instans Amazon EC2 yang dihosting di Amazon Virtual Private Cloud (Amazon VPC). Pesan tetap berada di dalam jaringan AWS tanpa menjelajahi internet publik. Dengan menerbitkan pesan secara privat dari VPC, Anda dapat meningkatkan keamanan lalu lintas antara aplikasi Anda dan Amazon SNS. Keamanan ini penting ketika Anda menerbitkan informasi yang dapat diidentifikasi secara pribadi (PII) tentang pelanggan Anda, atau ketika aplikasi Anda tunduk pada peraturan pasar. Sebagai contoh, menerbitkan secara privat sangat membantu jika Anda memiliki sistem pemeliharaan kesehatan yang harus mematuhi Health Insurance Portability and Accountability Act (HIPAA), atau sistem keuangan yang harus mematuhi Payment Card Industry Data Security Standard (PCI DSS).

Langkah-langkah umumnya adalah sebagai berikut:

- Gunakan templat AWS CloudFormation untuk secara otomatis membuat jaringan privat sementara di Akun AWS Anda.
- Buat VPC endpoint yang menghubungkan VPC dengan Amazon SNS.
- Login ke instans Amazon EC2 dan terbitkan pesan secara privat ke topik Amazon SNS.
- Verifikasi bahwa pesan telah berhasil dikirimkan.
- Hapus sumber daya yang Anda buat selama proses ini sehingga tidak tetap berada di Akun AWS Anda.

Diagram berikut menggambarkan jaringan privat yang Anda buat di akun AWS Anda saat Anda menyelesaikan langkah-langkah berikut ini:



Jaringan ini terdiri dari VPC yang berisi instans Amazon EC2. Instans menghubungkan ke Amazon SNS melalui VPC endpoint antarmuka. Titik akhir jenis ini terhubung ke layanan yang didukung oleh PrivateLink AWS. Dengan dibuatnya koneksi ini, Anda dapat login ke instans Amazon EC2 dan menerbitkan pesan ke topik Amazon SNS, meskipun jaringan terputus dari internet publik. Topik melakukan fan out pesan yang diterima ke dua fungsi AWS Lambda berlangganan. Fungsi tersebut mencatat pesan yang mereka terima di Amazon CloudWatch Logs.

Dibutuhkan sekitar 20 menit untuk menyelesaikan langkah-langkah tersebut.

Topik

- Sebelum Anda memulai
- Langkah 1: Membuat key pair Amazon EC2
- Langkah 2: BuatAWSsumber daya
- Langkah 3: Konfirmasi bahwa instans Amazon EC2 Anda tidak memiliki akses internet
- Langkah 4: Membuat VPC endpoint Amazon untuk Amazon SNS
- Langkah 5: Memublikasikan pesan ke topik Amazon SNS Anda
- Langkah 6: Memverifikasi pengiriman pesan Anda
- Langkah 7: Membersihkan
- Sumber daya terkait

Sebelum Anda memulai

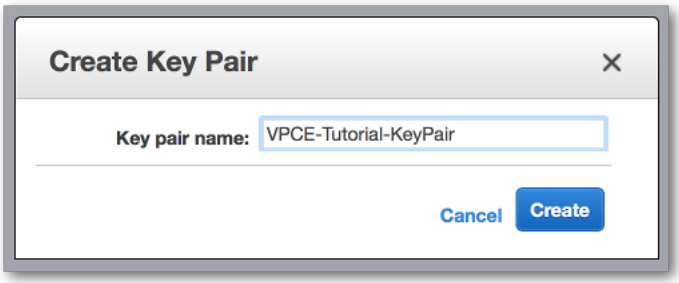
Sebelum Anda memulai, Anda memerlukan akun Amazon Web Services (AWS). Saat Anda daftar, akun Anda secara otomatis terdaftar untuk semua layanan di AWS, termasuk Amazon SNS dan Amazon VPC. Jika Anda belum membuat akun, buka https://aws.amazon.com/, dan kemudian pilih Buat Akun Gratis.

Langkah 1: Membuat key pair Amazon EC2

Pasangan kunci digunakan untuk login ke instans Amazon EC2. 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 pasangan kunci untuk login ke instans Amazon EC2. 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 konsol Amazon EC2 pada 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 secara otomatis akan diunduh oleh peramban Anda. Simpan di tempat yang aman. Amazon EC2 memberikan file ekstensi `.pem`.
6. (Opsional) Jika Anda akan menggunakan klien SSH pada komputer Mac atau Linux untuk menghubungkan ke instans Anda, gunakan perintah `chmod` untuk mengatur izin file kunci privat 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: BuatAWSsumber daya

Untuk menyiapkan infrastruktur, Anda harus menggunakan templat AWS CloudFormation. Templat adalah file yang bertindak sebagai cetak biru untuk membangun sumber daya AWS, seperti instans Amazon EC2 dan topik Amazon SNS. Templat untuk proses ini disediakan pada GitHub untuk Anda unduh.

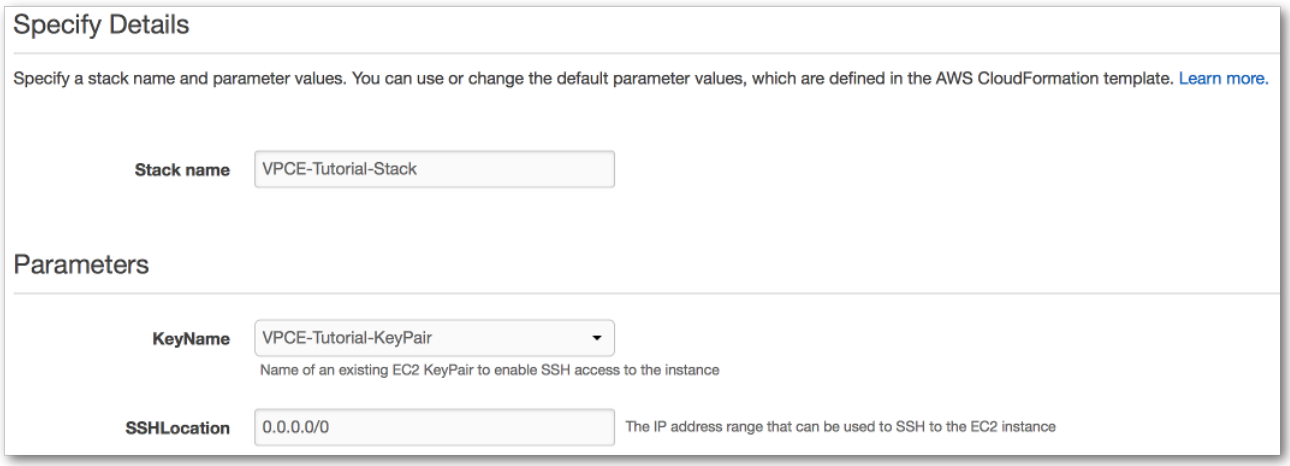
Anda menyediakan templat untuk AWS CloudFormation, dan ketentuan AWS CloudFormation sumber daya yang Anda butuhkan sebagai tumpukan di Akun AWS Anda. Tumpukan adalah kumpulan sumber daya yang Anda kelola sebagai unit tunggal. Saat Anda menyelesaikan langkah-langkah tersebut, Anda dapat menggunakan AWS CloudFormation untuk menghapus semua sumber daya dalam tumpukan sekaligus. Sumber daya tersebut tidak tetap berada di Akun AWS Anda, kecuali jika Anda menginginkannya.

Tumpukan untuk proses ini mencakup sumber daya berikut ini:

- VPC dan sumber daya jaringan yang berkaitan, termasuk subnet, grup keamanan, gateway internet, dan tabel rute.
- Instans Amazon EC2 yang diluncurkan ke subnet di VPC.
- Topik Amazon SNS.
- Dua fungsi AWS Lambda. Fungsi tersebut menerima pesan yang diterbitkan ke topik Amazon SNS, dan mereka mencatat peristiwa di CloudWatch Logs.
- Metrik dan log Amazon CloudWatch.
- IAM role yang mengizinkan instans Amazon EC2 untuk menggunakan Amazon SNS, dan IAM role yang mengizinkan fungsi Lambda untuk menulis ke CloudWatch logs.

Untuk membuat sumber daya AWS

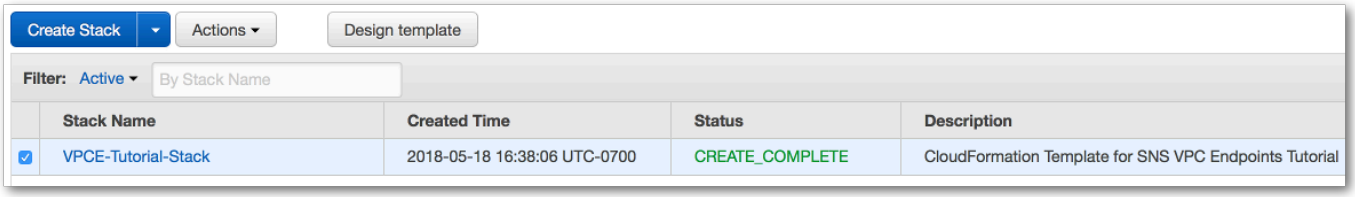
1. Unduh file templat dari situs web GitHub.
2. Masuk ke konsol AWS CloudFormation.
3. Pilih Buat Tumpukan.
4. Pada halaman Pilih Templat, pilih Unggah templat ke Amazon S3, pilih file, dan pilih Berikutnya.
5. Pada halaman Tentukan Detail, tentukan nama tumpukan dan kunci:
   a. Untuk Nama tumpukan, ketik **`VPCE-Tutorial-Stack`**.
   b. Untuk Nama Kunci, pilih VPCE-Tutorial-KeyPair.
   c. Untuk Lokasi SSH, simpan 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 Kapabilitas, akui bahwa AWS CloudFormation mungkin membuat sumber daya IAM dengan nama kustom.
9. Pilih Buat.

   Konsol AWS CloudFormation membuka halaman Tumpukan. VPCE-Tutorial-Stack memiliki 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: Konfirmasi bahwa instans Amazon EC2 Anda tidak memiliki akses internet

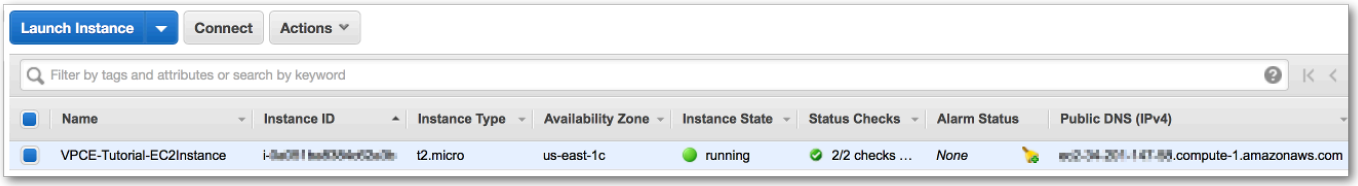
Instans Amazon EC2 yang diluncurkan di VPC Anda dalam langkah sebelumnya tidak memiliki akses internet. Ini melarang lalu lintas keluar, dan tidak dapat menerbitkan pesan ke Amazon SNS.

Verifikasikan ini dengan masuk ke instans. Kemudian, upayakan untuk menghubungkan ke titik akhir publik, dan upayakan untuk mengirim pesan Amazon SNS.

Pada titik ini, upaya menerbitkan gagal. Dalam langkah berikutnya, setelah Anda membuat VPC endpoint untuk Amazon SNS, upaya Anda untuk menerbitkan berhasil.

Untuk menghubungkan ke instans Amazon EC2 Anda

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Dalam menu navigasi di sebelah kiri, cari bagian Instans. Kemudian, pilih Instans.
3. Dalam daftar instans, pilih VPCE-Tutorial-EC2Instance.
4. Salin nama host yang disediakan di kolom DNS Publik (IPv4).



5. Buka terminal. Dari direktori yang berisi pasangan kunci, hubungkan ke instans menggunakan perintah berikut ini, di mana *nama host instans* adalah nama host yang Anda salin dari konsol Amazon EC2:

```
$ 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 instans tidak memiliki konektivitas ke Amazon SNS

1. Masuk ke konsol Amazon SNS.
2. Di panel navigasi di sebelah kiri, pilih Topik.

3. Pada halaman Topik, salin Amazon Resource Name (ARN) ke 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: Membuat VPC endpoint Amazon untuk Amazon SNS

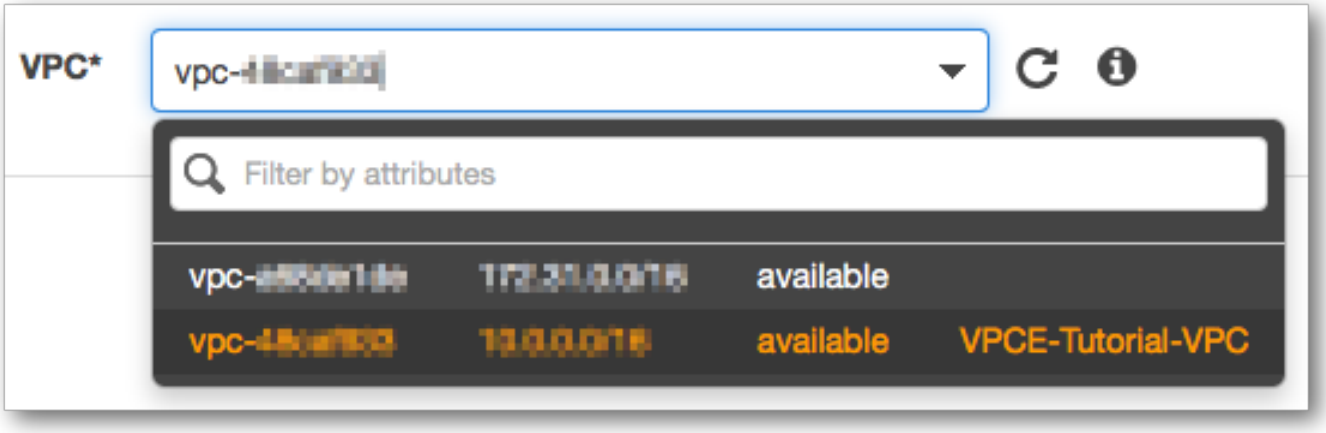
Untuk menghubungkan VPC ke Amazon SNS, Anda harus menentukan VPC endpoint antarmuka. Setelah Anda menambahkan titik akhir, Anda dapat login ke instans Amazon EC2 di VPC Anda, dan dari sana Anda dapat menggunakan API Amazon SNS. Anda dapat menerbitkan pesan ke topik, dan pesan diterbitkan secara privat. Pesan tetap di dalam jaringan AWS, dan pesan tidak menjelajahi internet publik.

> **Note**
>
> Instans masih tidak memiliki akses ke layanan AWS dan titik akhir pada internet.

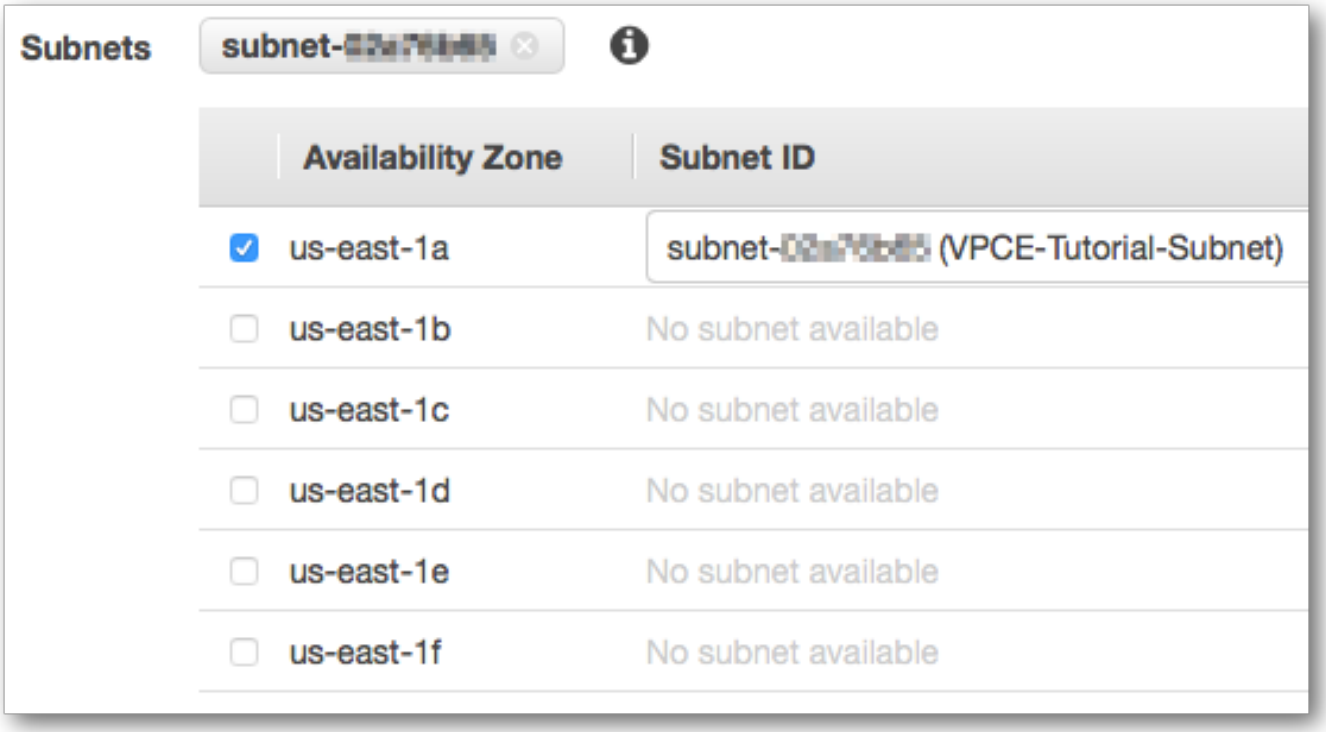
Untuk membuat titik akhir

1. Buka konsol Amazon VPC pada 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 Amazon SNS.

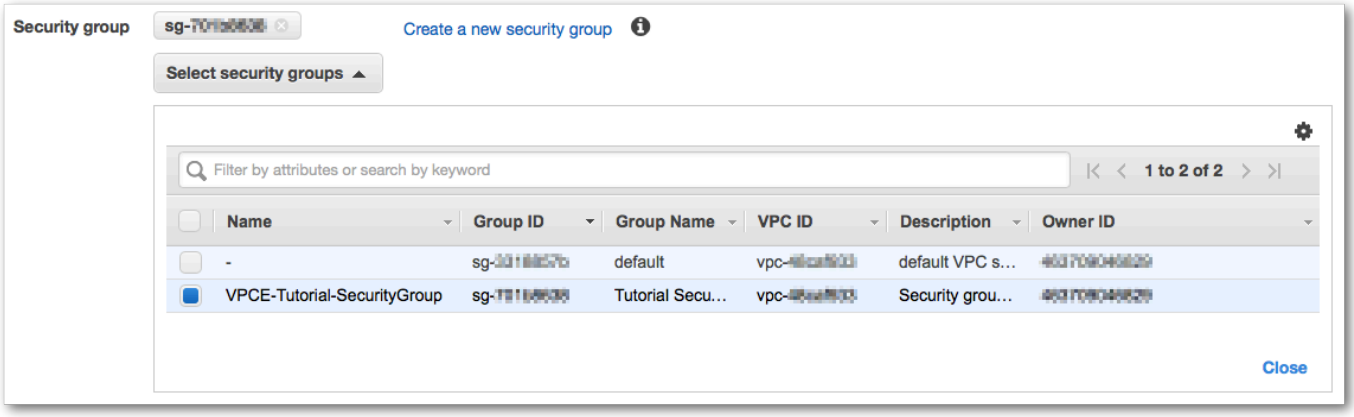
   Nama layanan bervariasi berdasarkan wilayah yang dipilih. Sebagai contoh, jika Anda memilih US East (N. Virginia), nama layanan adalah com.amazonaws.*us-east-1*.sns.
6. Untuk VPC, pilih VPC yang memiliki nama VPCE-Tutorial-VPC.



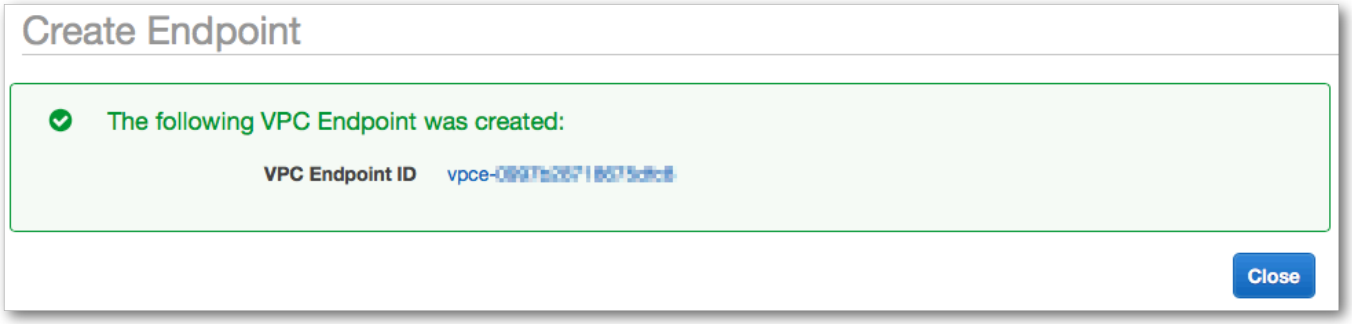
7. Untuk Subnet, pilih subnet yang memiliki VPCE-Tutorial-Subnet di ID subnet.



8. Untuk Aktifkan Nama DNS Privat, pilih Aktifkan untuk titik akhir ini.
9. Untuk Grup keamanan, pilih Pilih grup keamanan, dan pilih VPCE-Tutorial-SecurityGroup.

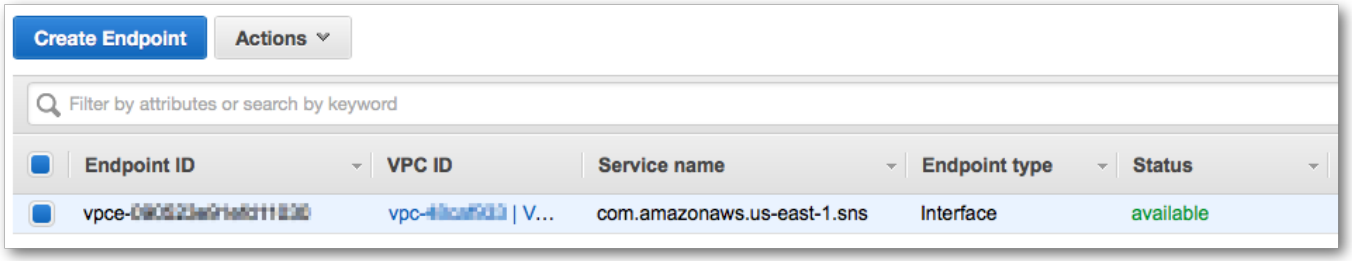


10. Pilih Buat titik akhir. Konsol Amazon VPC mengonfirmasi bahwa VPC endpoint dibuat.



11. Pilih Tutup.

    Konsol Amazon VPC membuka halaman Titik akhir. Titik akhir baru memiliki status tertunda. Dalam beberapa menit, setelah proses pembuatan selesai, status berubah menjadi tersedia.



Langkah 5: Memublikasikan pesan ke topik Amazon SNS Anda

Sekarang VPC Anda mencakup titik akhir untuk Amazon SNS, Anda dapat login ke instans Amazon EC2 dan menerbitkan pesan ke topik.

Untuk menerbitkan pesan

1. Jika terminal Anda tidak lagi terhubung ke instans Amazon EC2 Anda, hubungkan lagi:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Jalankan perintah yang sama yang Anda lakukan sebelumnya untuk menerbitkan pesan ke topik Amazon SNS Anda. Kali ini, upaya untuk menerbitkan 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"
}
```

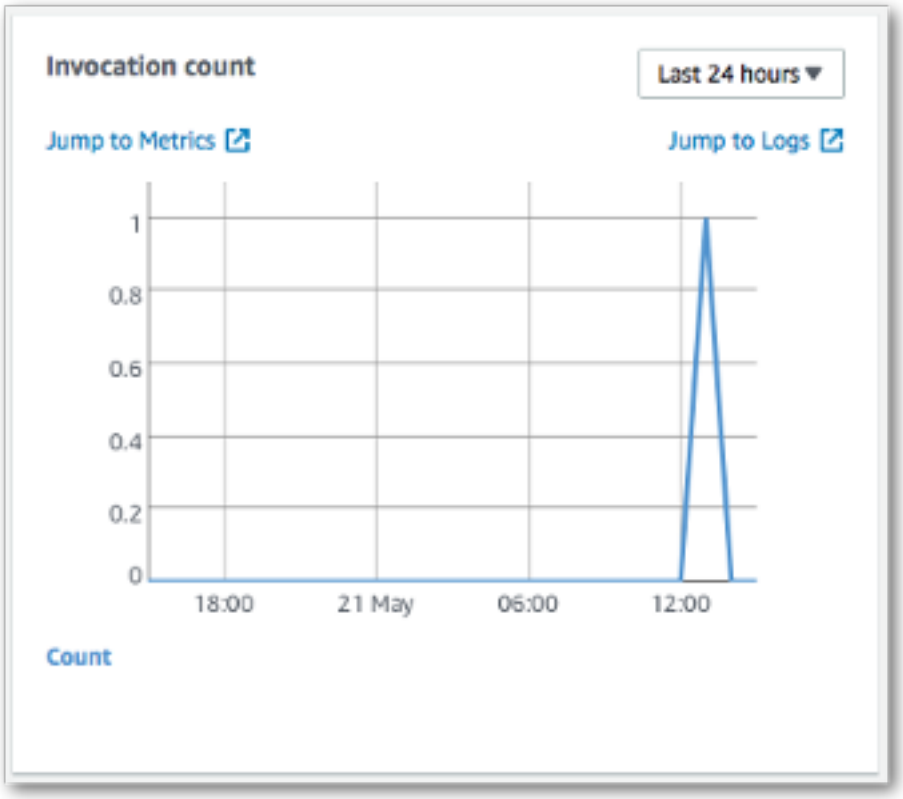
Langkah 6: Memverifikasi pengiriman pesan Anda

Ketika topik Amazon SNS menerima pesan, akan melakukan fan out pesan dengan mengirimkannya ke dua fungsi Lambda yang berlangganan. Saat fungsi tersebut menerima pesan, mereka mencatat peristiwa ke CloudWatch logs. Untuk memverifikasi bahwa pengiriman pesan Anda berhasil, periksa apakah fungsi telah dipanggil, dan periksa apakah CloudWatch logs telah diperbarui.

Untuk memverifikasi bahwa fungsi Lambda dipanggil

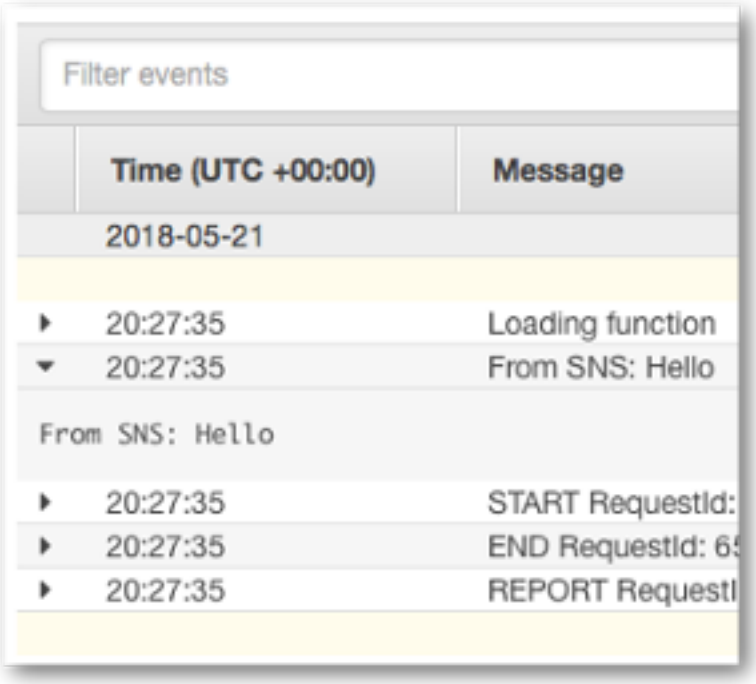
1. Buka konsol AWS Lambda pada https://console.aws.amazon.com/lambda/.
2. Pada halaman Fungsi, pilih VPCe-Tutorial-Lambda-1.
3. Pilih Pemantauan.
4. Periksa grafik Hitungan invokasi. Grafik ini menunjukkan jumlah waktu saat fungsi Lambda telah dijalankan.

   Jumlah invokasi cocok dengan jumlah waktu saat Anda menerbitkan pesan ke topik.



Untuk memverifikasi bahwa CloudWatch logs telah diperbarui

1. Buka konsol CloudWatch pada 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 grup log /aws/lambda/VPCE-Tutorial-Lambda-2/.

Selamat! Dengan menambahkan titik akhir untuk Amazon SNS ke VPC, Anda dapat menerbitkan pesan ke topik dari dalam jaringan yang terkelola 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 sumber daya AWS yang tidak lagi Anda gunakan, Anda mencegah biaya yang tidak perlu untuk Akun AWS Anda.

Pertama, hapus VPC endpoint Anda menggunakan konsol Amazon VPC. Kemudian, hapus sumber daya lain yang Anda buat dengan menghapus tumpukan di konsol AWS CloudFormation. Saat Anda menghapus tumpukan, AWS CloudFormation menghapus sumber daya tumpukan dari Akun AWS.

Untuk menghapus VPC endpoint Anda

1. Buka konsol Amazon VPC pada 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 tumpukan AWS CloudFormation Anda

1. Buka konsol AWS CloudFormation pada 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.

- AWSBlog Keamanan: Mengamankan pesan yang diterbitkan ke Amazon SNS denganAWSPrivateLink
- Apa yang Dimaksud Amazon VPC?
- VPC Endpoint
- Apa yang Dimaksud Amazon EC2?
- AWS CloudFormationKonsep

Keamanan Perlindungan Data Pesan

- Perlindungan Data Pesanadalah fitur di Amazon SNS yang digunakan untuk menentukan aturan dan kebijakan Anda sendiri untuk mengaudit dan mengontrol konten untuk data yang bergerak, dibandingkan dengan data yang tidak aktif.
- Perlindungan Data Pesan menyediakan layanan tata kelola, kepatuhan, dan audit untuk aplikasi perusahaan yang berpusat pada pesan, sehingga masuknya data dan jalan keluar dapat dikontrol oleh pemilik topik Amazon SNS, 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 masing-masing pelanggan, dan mengaudit seluruh proses alur konten.

Identity and access management di Amazon SNS

Akses ke Amazon SNS memerlukan kredensial yang dapat digunakan AWS untuk mengautentikasi permintaan Anda. Kredensial ini harus memiliki izin untuk mengakses sumber daya AWS, seperti topik dan pesan Amazon SNS. 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 mengendalikan siapa yang dapat mengaksesnya.

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke sumber daya AWS secara aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon SNS. IAM adalah layanan Layanan AWS yang dapat Anda gunakan tanpa dikenakan biaya tambahan.

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon SNS.

Pengguna layanan - Jika Anda menggunakan layanan Amazon SNS untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensyal dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Amazon SNS 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 Amazon SNS, lihat. Memecahkan masalah identitas dan akses Amazon Simple Notification Service

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Amazon SNS di perusahaan Anda, Anda mungkin memiliki akses penuh ke Amazon SNS. Tugas Anda adalah menentukan fitur dan sumber daya Amazon SNS mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Amazon SNS, lihat. Bagaimana Amazon Simple Notification Service bekerja dengan IAM

Administrator IAM - Jika Anda administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Amazon SNS. Untuk melihat contoh kebijakan berbasis identitas Amazon SNS yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Mengautentikasi dengan identitas

Autentikasi adalah cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Anda harus diautentikasi (masuk keAWS) sebagai pengguna Akun AWS root, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk ke AWS sebagai identitas terfederasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. Pengguna AWS IAM Identity Center Pengguna (Pusat Identitas IAM), autentikasi Single Sign-On perusahaan Anda, dan kredensial Google atau Facebook Anda merupakan contoh identitas terfederasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil suatu peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal akses AWS. Untuk informasi selengkapnya tentang cara masuk ke AWS, lihat Cara masuk ke Akun AWS dalam Panduan Pengguna AWS Sign-In.

Jika Anda mengakses AWS secara terprogram, AWS memberikan Kit Pengembangan Perangkat Lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan peralatan AWS, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang cara menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan API AWS dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Sebagai contoh, AWS menyarankan Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Pengguna root Akun AWS

Ketika membuat Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses penuh ke semua Layanan AWS dan sumber daya di akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk menggunakan alamat email dan kata sandi yang Anda

gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Praktik terbaiknya adalah mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensial temporer.

Identitas terfederasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, AWS Directory Service, direktori Pusat Identitas, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas terfederasi mengakses Akun AWS, identitas tersebut mengambil peran, dan peran ini memberikan kredensial sementara.

Untuk pengelolaan akses terpusat, sebaiknya Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apa yang dimaksud Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM merupakan identitas dalam Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara dalam AWS Management Console dengan berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center.
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, pada beberapa Layanan AWS, Anda dapat menyertakan kebijakan secara langsung ke sumber daya (bukan menggunakan peran sebagai proksi). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan – Sebagian Layanan AWS menggunakan fitur di Layanan AWS lainnya. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut

menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, menggunakan peran layanan, atau peran terkait layanan.

- Sesi akses maju (FAS) – Ketika Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai seorang pengguna utama. Saat menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.
- Peran IAM – Peran layanan adalah peran IAM yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
- Peran terkait layanan – Peran terkait layanan adalah tipe peran layanan yang terkait dengan Layanan AWS. Layanan tersebut dapat mengambil peran untuk melakukan sebuah tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan di instans EC2 dan mengajukan permintaan API AWS CLI atau AWS. Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan peran AWS ke instans EC2 dan menyediakannya bagi semua aplikasinya, Anda dapat membuat profil instans yang dilampirkan ke instans tersebut. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan adalah objek di AWS yang, ketika terkait dengan identitas atau sumber daya, akan menentukan izinnya. AWS mengevaluasi kebijakan-kebijakan tersebut ketika seorang pengguna utama (pengguna, pengguna root, atau sesi peran) mengajukan permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, silakan lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses terhadap apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau API AWS.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM 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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran 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 secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.

- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS yang dimiliki bisnis Anda secara terpusat. Jika Anda mengaktifkan semua fitur di sebuah organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau ke semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap pengguna root Akun AWS. Untuk informasi selengkapnya tentang Organizations dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. 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 eksplisit di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan ketika beberapa tipe kebijakan digunakan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Pengendalian akses

Amazon SNS memiliki sistem perizinan berbasis sumber daya sendiri yang menggunakan kebijakan yang ditulis dalam bahasa yang sama yang digunakan untuk kebijakan AWS Identity and Access Management (IAM). Hal ini berarti Anda dapat mencapai hal yang sama dengan kebijakan Amazon SNS dan kebijakan IAM.

> **Note**
>
> Penting untuk memahami bahwa semua Akun AWS dapat mendelegasikan izinnya kepada pengguna di bawah akunnya. 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 dalam Panduan Pengguna IAM.

Gambaran umum tentang 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

- Kapan pengendalian akses harus digunakan
- Konsep utama
- Gambaran umum arsitektur
- Menggunakan Bahasa Kebijakan Akses
- Logika evaluasi
- Contoh kasus untuk pengendalian akses Amazon SNS

Kapan pengendalian akses harus digunakan

Anda memiliki banyak fleksibilitas dalam cara Anda memberikan atau menolak akses ke sumber daya. Namun, kasus penggunaan yang umum cukup sederhana:

- Anda ingin memberikan Akun AWS yang lain jenis tindakan topik tertentu (misalnya, Publikasikan). Untuk informasi selengkapnya, lihat Berikan Akun AWS akses ke topik.
- Anda ingin membatasi langganan topik Anda ke protokol HTTPS saja. Untuk informasi selengkapnya, lihat Batasi langganan ke HTTPS.
- Anda ingin memungkinkan Amazon SNS memublikasikan pesan ke antrean Amazon SQS Anda. Untuk informasi selengkapnya, lihat Memublikasikan pesan ke antrean Amazon SQS.

Konsep 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

- Izin
- Pernyataan

- 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 memublikasikan (B) ke TopicA (C) selama dia menggunakan protokol HTTP (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 karakter ASCII yang diizinkan dalam dokumen kebijakan. Anda dapat memanfaatkan `aws:SourceAccount` dan `aws:SourceOwner` mengatasi skenario di mana Anda perlu plug-in ARN AWS layanan lain yang berisi karakter non-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 pemilik sumber daya. AWS tidak mengizinkan pengguna layanan AWS membuat kebijakan sumber daya yang tidak dimilikinya. Jika John adalah pemilik sumber daya, AWS mengautentikasi identitas John ketika dia menyerahkan kebijakan yang dia tulis untuk memberikan izin untuk sumber daya itu.

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

ini hanya operasi dalam permintaan untuk AWS. Misalnya, Jane mengirimkan 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 peminta harus bagian dari kisaran CIDR 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. Layanan AWS itu sendiri (misalnya, Amazon SQS atau Amazon SNS) mungkin juga mendefinisikan kunci khusus layanan. Untuk informasi selengkapnya, lihat Izin API Amazon SNS: Tindakan dan referensi 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 layanan AWS untuk menentukan apakah permintaan masuk harus ditolak atau diizinkan berdasarkan kebijakan yang diterapkan. 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

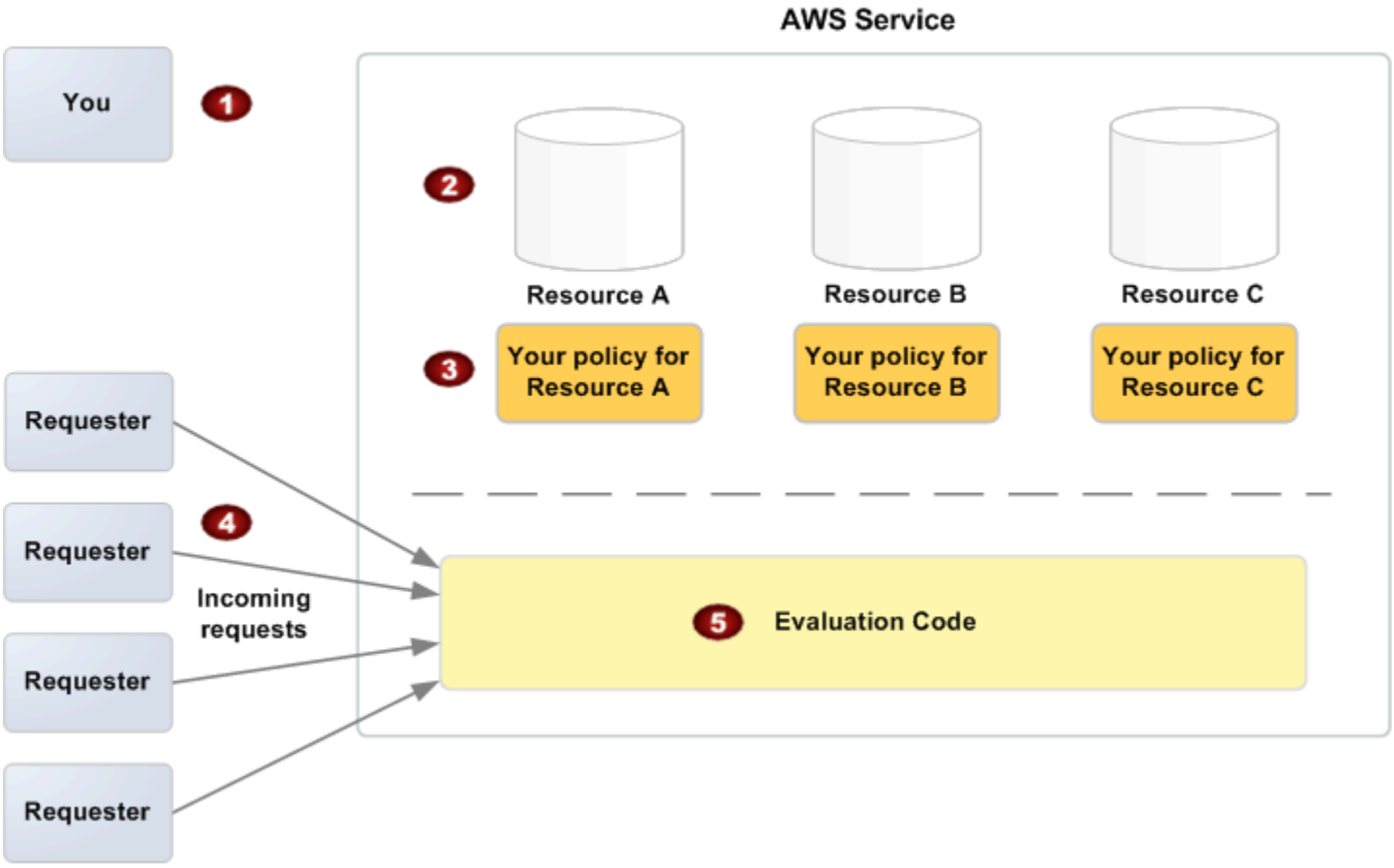
Perizinan dihasilkan dari pernyataan yang memiliki efek=mengizinkan, dengan asumsi syarat apa pun yang dinyatakan terpenuhi. Contoh: Mengizinkan permintaan jika diterima sebelum pukul 1:00 siang pada tanggal 30 April 2010. Perizinan mengabaikan semua blokir secara default, tetapi tidak mengabaikan penolakan 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.

Gambaran umum arsitektur

Gambar dan tabel berikut menggambarkan komponen utama yang berinteraksi untuk memberikan pengendalian akses untuk sumber daya Anda.

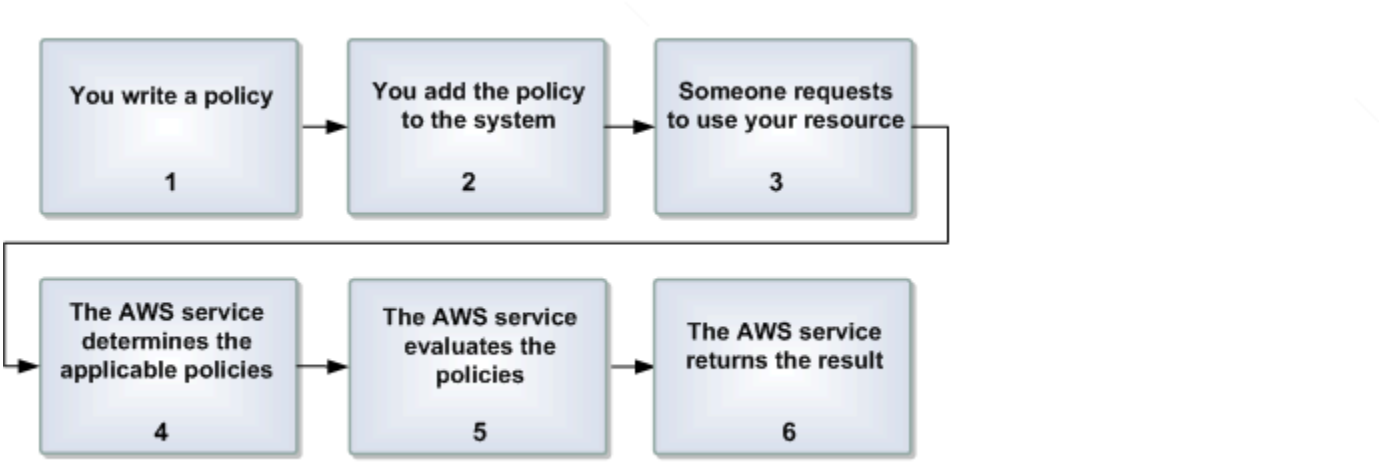


1	Anda, pemilik sumber daya.
2	Sumber daya Anda (terdapat dalam layanan AWS; misalnya, antrean Amazon SQS).
3	Kebijakan Anda. Biasanya Anda memiliki satu kebijakan per sumber daya, meskipun Anda bisa memiliki beberapa. Layanan AWS sendiri menyediakan API yang Anda gunakan untuk mengunggah dan mengelola kebijakan Anda.
4	Peminta dan permintaan mereka yang masuk ke AWS layanan.
5	Kode evaluasi bahasa kebijakan akses.

	Hal ini adalah rangkaian kode dalam layanan AWS yang mengevaluasi permintaan masuk terhadap kebijakan yang dapat diterapkan dan menentukan apakah peminta diizinkan mendapat akses ke sumber daya. Untuk informasi tentang cara membuat keputusan, lihat Logika evaluasi.

Menggunakan Bahasa Kebijakan Akses

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 topik Amazon SNS Anda.
2	Anda mengunggah kebijakan Anda ke AWS. Layanan AWS sendiri menyediakan API yang Anda gunakan untuk mengunggah kebijakan Anda. Misalnya, Anda menggunakan tindakan `SetTopicAttributes` Amazon SNS untuk mengunggah kebijakan untuk topik Amazon SNS tertentu.
3	Seseorang mengirimkan permintaan untuk menggunakan sumber daya Anda. Misalnya, pengguna mengirimkan permintaan ke Amazon SNS untuk menggunakan salah satu topik Anda.
4	Layanan AWS menentukan kebijakan mana yang diterapkan pada permintaan.

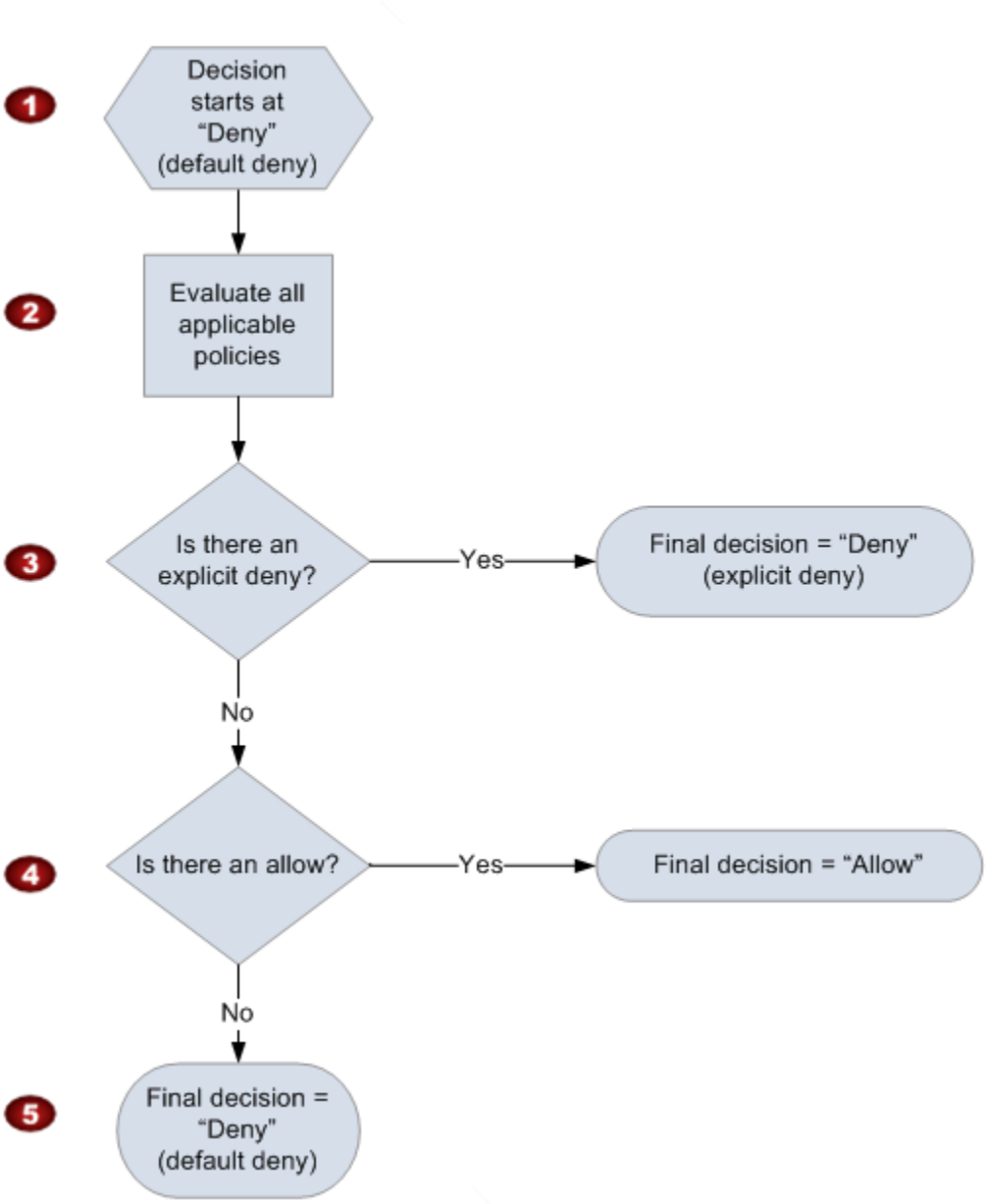
	Misalnya, Amazon SNS melihat semua kebijakan Amazon SNS yang tersedia dan menentukan yang mana yang diterapkan (berdasarkan sumber daya apa, siapa peminta, dll.).
5	Layanan AWS mengevaluasi kebijakan. Misalnya, Amazon SNS mengevaluasi kebijakan dan menentukan apakah peminta diizinkan untuk menggunakan topik Anda atau tidak. Untuk informasi tentang logika keputusan, lihat Logika evaluasi.
6	Layanan AWS menolak permintaan atau melanjutkan untuk 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.

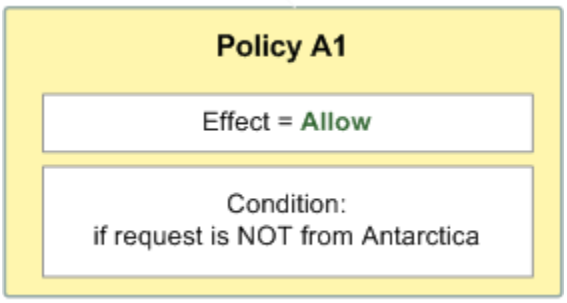
	Jika menemukan satu penolakan pun, kode penerapan akan mengembalikan keputusan "menolak" dan proses selesai (ini adalah penolakan eksplisit; untuk informasi lebih lanjut, lihat Penolakan eksplisit).
4	Jika penolakan eksplisit tidak ditemukan, kode penerapan mencari instruksi "mengizinkan" apa pun yang 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. Sebagai contoh, jika pengguna meminta menggunakan Amazon SNS, tetapi kebijakan tentang topik tidak merujuk ke Akun AWS sama sekali, maka keputusan itu menghasilkan blokir secara default.

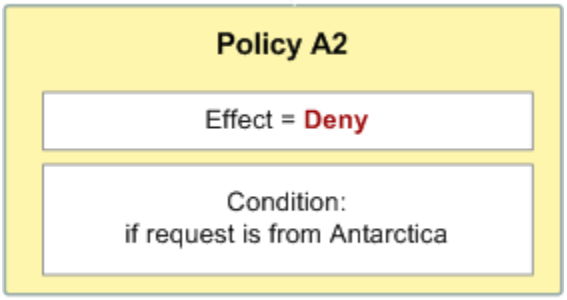
Kebijakan juga mengakibatkan blokir secara default jika suatu syarat dalam pernyataan tidak terpenuhi. Jika semua syarat dalam pernyataan terpenuhi, maka kebijakan menghasilkan perizinan atau penolakan eksplisit, berdasarkan nilai elemen Efek dalam kebijakan. Kebijakan tidak menentukan apa yang harus 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.



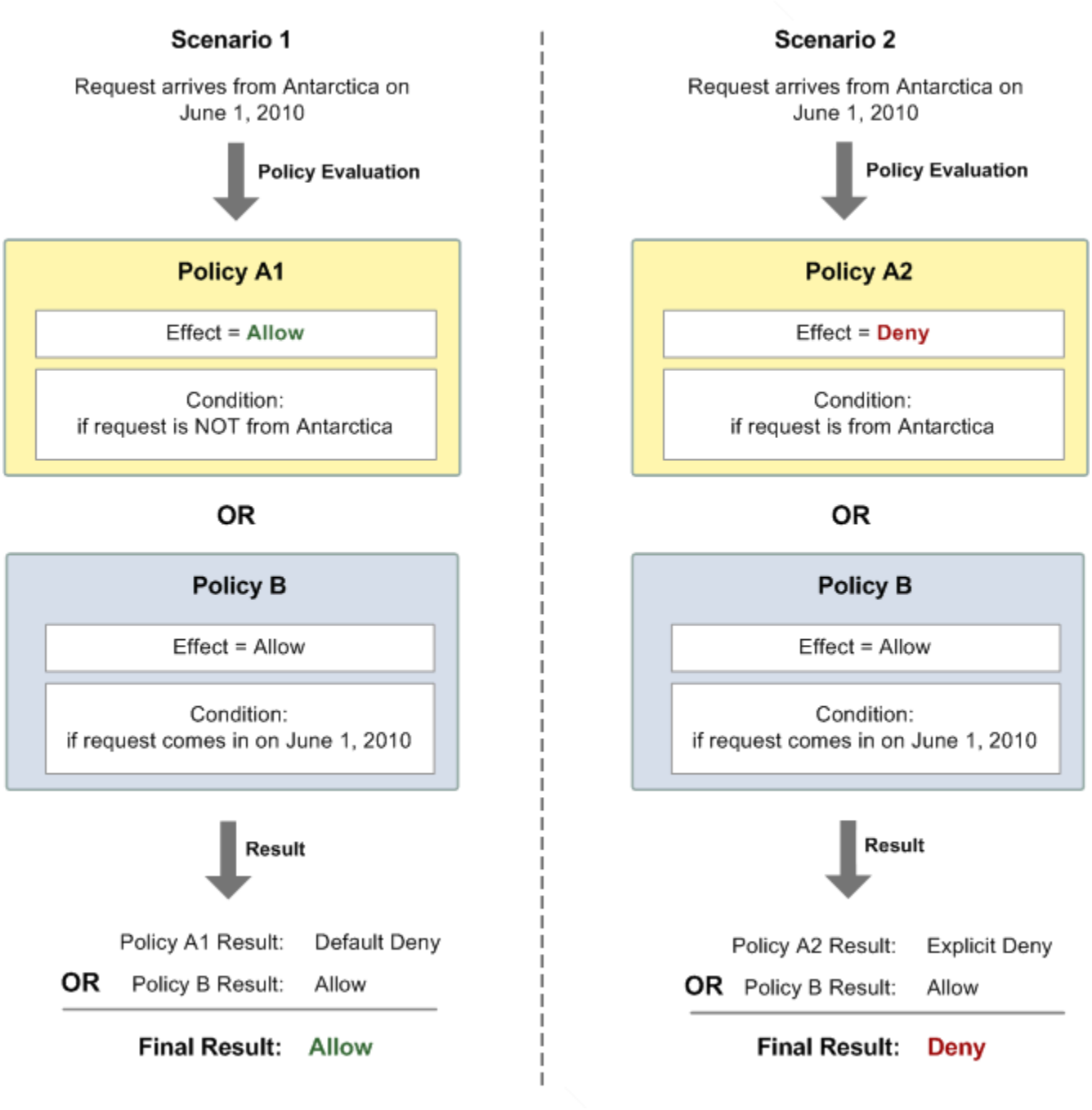
Jika seseorang mengirimkan permintaan dari AS, syaratnya terpenuhi (permintaannya bukan dari Antartika). Oleh karena itu, permintaan diizinkan. Namun, jika seseorang mengirimkan permintaan dari Antartika, syarat tidak terpenuhi, dan oleh karena itu hasil kebijakan adalah blokir secara default.

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 pengendalian akses Amazon SNS

Bagian ini menjelaskan beberapa contoh kasus penggunaan umum untuk kontrol akses.

Topik

- Berikan Akun AWS akses ke topik
- Batasi langganan ke HTTPS
- Memublikasikan pesan ke antrean Amazon SQS
- Izinkan pemberitahuan acara Amazon S3 untuk mempublikasikan ke suatu topik
- Izinkan Amazon SES mempublikasikan ke topik yang dimiliki oleh akun lain
- aws:SourceAccount versus aws:SourceOwner
- Izinkan akun dalam organisasi di AWS Organizations untuk memublikasikan topik di akun yang berbeda
- Izinkan CloudWatch alarm apa pun untuk mempublikasikan ke topik di akun yang berbeda
- Membatasi publikasi ke topik Amazon SNS hanya dari VPC endpoint tertentu

Berikan Akun AWS akses ke topik

Katakan Anda mempunyai topik dalam sistem Amazon SNS. Dalam kasus yang paling sederhana, Anda ingin mengizinkan satu atau lebih Akun AWS untuk mengakses tindakan topik tertentu (misalnya, Publikasikan).

Anda dapat melakukannya menggunakan tindakan API `AddPermission` Amazon SNS. Dibutuhkan topik, daftar ID Akun AWS, daftar tindakan, dan label, dan secara otomatis membuat pernyataan baru dalam kebijakan pengendalian akses topik. Dalam kasus ini, Anda tidak menulis kebijakan sendiri, karena Amazon SNS secara otomatis menghasilkan pernyataan kebijakan baru untuk Anda. Anda dapat menghapus pernyataan kebijakan nanti dengan memanggil `RemovePermission` dengan labelnya.

Misalnya, jika Anda memanggil `AddPermission` topik arn:aws:sns:us-east- 2:444455556666:, dengan MyTopic ID 1111-2222-3333, tindakan, dan label, Amazon SNS akan menghasilkan dan menyisipkan pernyataan kebijakan Akun AWS kontrol akses berikut: `Publish` `grant-1234-publish`

```
{
```

```
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Setelah pernyataan ini ditambahkan, pengguna dengan Akun AWS 1111-2222-3333 dapat memublikasikan pesan ke topik tersebut.

Batasi langganan ke HTTPS

Pada contoh berikut, Anda membatasi protokol pengiriman notifikasi ke HTTPS.

Anda perlu tahu cara menulis kebijakan Anda sendiri untuk topik karena tindakan `AddPermission` Amazon SNS tidak memungkinkan Anda menentukan pembatasan protokol ketika memberikan seseorang akses ke topik Anda. Dalam kasus ini, Anda akan menulis kebijakan Anda sendiri, dan kemudian menggunakan tindakan `SetTopicAttributes` untuk mengatur atribut `Policy` topik untuk kebijakan baru Anda.

Contoh kebijakan penuh berikut memberikan ID Akun AWS 1111-2222-3333 kemampuan untuk berlangganan notifikasi dari topik.

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

```
}
```

Memublikasikan pesan ke antrean Amazon SQS

Dalam kasus penggunaan ini, Anda ingin memublikasikan pesan dari topik Anda ke antrean Amazon SQS Anda. Seperti Amazon SNS, Amazon SQS menggunakan bahasa kebijakan pengendalian akses Amazon. Untuk mengizinkan Amazon SNS mengirimkan pesan, Anda harus menggunakan tindakan `SetQueueAttributes` Amazon SQS untuk mengatur kebijakan pada antrean.

Sekali lagi, Anda akan perlu tahu bagaimana menulis kebijakan Anda sendiri karena tindakan `AddPermission` Amazon SQS tidak membuat pernyataan kebijakan dengan syarat.

> **Note**
>
> Contoh yang disajikan di bawah ini adalah kebijakan Amazon SQS (mengendalikan akses ke antrean Anda), bukan kebijakan Amazon SNS (mengendalikan akses ke topik Anda). Tindakan adalah tindakan Amazon SQS, dan sumber daya adalah Amazon Resource Name (ARN) antrean. Anda dapat menentukan ARN antrean dengan mengambil atribut `QueueArn` antrean dengan 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 syarat `aws:SourceArn` untuk membatasi akses ke antrean berdasarkan sumber pesan yang dikirim ke antrean. Anda dapat menggunakan jenis kebijakan ini untuk

mengizinkan Amazon SNS mengirimkan pesan ke antrean Anda hanya jika pesan berasal dari salah satu topik Anda sendiri. Dalam hal ini, Anda menentukan topik tertentu, yang ARN adalah arn:aws:sns:us-east- 2:444455556666:. MyTopic

Kebijakan sebelumnya adalah contoh kebijakan Amazon SQS Anda dapat menulis dan menambahkan antrean tertentu. Ini akan memberikan akses ke Amazon SNS dan layanan AWS lainnya. Amazon SNS memberikan kebijakan default untuk semua topik yang baru dibuat. Kebijakan default memberikan akses ke topik Anda ke semua layanan AWS lain. Kebijakan default ini menggunakan syarat `aws:SourceArn` untuk memastikan bahwa layanan AWS mengakses topik Anda hanya atas nama sumber daya AWS yang Anda miliki.

Izinkan pemberitahuan acara Amazon S3 untuk mempublikasikan ke suatu topik

Dalam kasus ini, Anda ingin mengkonfigurasi kebijakan topik sehingga bucket Amazon S3 Akun AWS yang lain dapat memublikasikan ke topik Anda. Untuk informasi selengkapnya tentang memublikasikan notifikasi dari Amazon S3, kunjungi Menyiapkan notifikasi Peristiwa Bucket.

Contoh ini mengasumsikan bahwa Anda menulis kebijakan Anda sendiri dan kemudian menggunakan tindakan `SetTopicAttributes` untuk mengatur atribut `Policy` topik untuk kebijakan baru Anda.

Contoh pernyataan berikut menggunakan syarat `SourceAccount` untuk memastikan bahwa hanya akun pemilik Amazon S3 yang dapat mengakses topik. Dalam contoh ini, pemilik topik adalah 111122223333 dan pemilik Amazon S3 adalah 444455556666. Contoh tersebut menyatakan bahwa setiap bucket Amazon S3 yang dimiliki oleh 444455556666 diizinkan untuk dipublikasikan. MyTopic

```
{
  "Statement": [{
    "Effect": "Allow",
     "Principal": {
      "Service": "s3.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      }
    }
  }]
}
```

Saat memublikasikan acara ke Amazon SNS, layanan berikut mendukung: `aws:SourceAccount`

- Amazon API Gateway
- Amazon CloudWatch
- DevOpsGuru Amazon
- Amazon ElastiCache
- Amazon GameLift
- API SMS dan Suara Amazon Pinpoint
- 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

Izinkan Amazon SES mempublikasikan ke topik yang dimiliki oleh akun lain

Anda dapat mengizinkan layanan AWS yang lain untuk memublikasikan ke topik yang dimiliki oleh Akun AWS lain. Misalkan Anda masuk ke akun 111122223333, membuka Amazon SES, dan membuat email. Untuk memublikasikan notifikasi tentang email ini ke topik Amazon SNS yang dimiliki akun 444455556666, Anda akan membuat kebijakan seperti berikut. Untuk melakukannya, Anda perlu memberikan informasi tentang penanggung jawab (layanan lainnya) dan kepemilikan setiap sumber daya. Pernyataan `Resource` menyediakan ARN topik, 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 Amazon SNS, layanan berikut mendukung: `aws:SourceOwner`

- Amazon API Gateway
- Amazon CloudWatch
- DevOpsGuru Amazon
- Amazon ElastiCache
- Amazon GameLift
- API SMS dan Suara Amazon Pinpoint
- 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 topik Amazon SNS. 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 Izinkan Amazon SES mempublikasikan ke topik yang dimiliki oleh akun lain bagaimana Amazon SES menggunakan `aws:SourceOwner` dan daftar AWS layanan yang mendukung kondisi tersebut.

Izinkan akun dalam organisasi di AWS Organizations untuk memublikasikan topik di akun yang berbeda

Layanan AWS Organizations membantu Anda mengelola penagihan secara terpusat, mengendalikan akses dan keamanan, serta membagikan sumber daya di seluruh Akun AWS.

Anda dapat menemukan ID organisasi Anda di konsol Organizations. Untuk informasi selengkapnya, lihat Melihat detail organisasi dari akun manajemen.

Dalam contoh ini, Akun AWS apa pun dalam `myOrgId` organisasi dapat memublikasikan ke topik Amazon SNS `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 topik Amazon SNS di akun. 444455556666

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

Membatasi publikasi ke topik Amazon SNS hanya dari VPC endpoint tertentu

Dalam kasus ini, topik di akun 444455556666 diizinkan untuk memublikasikan hanya dari VPC endpoint dengan ID 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 Simple Notification Service bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon SNS, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon SNS.

Fitur IAM yang dapat Anda gunakan dengan Amazon Simple Notification Service

Fitur IAM	Dukungan Amazon SNS
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Ya
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Parsial
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Ya
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon SNS dan layanan AWS lainnya dengan sebagian besar fitur IAM, AWSlihat layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Tindakan kebijakan untuk Amazon SNS

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama seperti operasi API AWS terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar tindakan Amazon SNS, 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-tindakan tersebut dengan koma.

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Amazon SNS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Sumber daya kebijakan untuk Amazon SNS

Mendukung sumber daya kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, 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 mengindikasikan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Amazon SNS dan ARNnya, 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 berbasis identitas Amazon SNS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Kunci kondisi kebijakan untuk Amazon SNS

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional 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 satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi spesifik layanan. Untuk melihat semua kunci kondisi global AWS, lihat kunci konteks kondisi global AWS dalam Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Amazon SNS, 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 berbasis identitas Amazon SNS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

ACL di Amazon SNS

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Amazon SNS

Mendukung ABAC (tanda dalam kebijakan)	Parsial

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Di AWS, atribut ini disebut tag. Anda dapat melampirkan tanda ke entitas IAM (pengguna atau peran) dan ke banyak sumber daya AWS. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi-operasi ketika tanda milik pengguna utama cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna dalam situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `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 hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial terkait langkah-langkah penyiapan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Menggunakan kredensyal sementara dengan Amazon SNS

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensial sementara. Sebagai informasi tambahan, termasuk tentang Layanan AWS mana saja yang berfungsi dengan kredensial sementara, lihat Layanan AWS yang berfungsi dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensial sementara jika Anda masuk ke AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda

mengakses AWS dengan menggunakan tautan masuk tunggal (SSO) milik perusahaan Anda, proses itu secara otomatis akan membuat kredensial temporer. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensial sementara secara manual menggunakan AWS CLI atau AWS API. Anda kemudian dapat menggunakan kredensial sementara untuk mengakses AWS. AWS menyarankan Anda membuat kredensial sementara secara dinamis, alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Izin utama lintas layanan untuk Amazon SNS

Mendukung sesi akses maju (FAS)	Ya

Jika menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai pengguna utama. Jika menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

Peran layanan untuk Amazon SNS

Mendukung peran layanan	Ya

Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Amazon SNS. Edit peran layanan hanya jika Amazon SNS memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Amazon SNS

Mendukung peran terkait layanan	Tidak

Peran tertaut layanan adalah jenis peran layanan yang terkait dengan Layanan AWS. Layanan ini dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau pengelolaan peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Temukan sebuah 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 sumber daya Amazon SNS. Pengguan dan peran tersebut juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau API AWS. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon SNS, termasuk format ARN 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 konsol Amazon SNS
- Jenis kebijakan lainnya
- Berbagai jenis kebijakan
- Izinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon SNS di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulai menggunakan kebijakan yang dikelola AWS dan beralih ke izin dengan hak akses paling rendah – Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan yang dikelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Kebijakan ini ada di Akun AWS Anda. Sebaiknya Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan yang dikelola AWS atau kebijakan yang dikelola AWS untuk fungsi pekerjaan di Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan harus dikirim menggunakan SSL. Anda juga dapat menggunakan kondisi untuk memberi akses ke tindakan layanan jika digunakan melalui Layanan AWS yang spesifk, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.

- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access 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 validasi kebijakan Analizer Akses IAM di Panduan Pengguna IAM.
- Wajibkan autentikasi multi-faktor (MFA) – Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Akun AWS Anda, aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Menggunakan konsol Amazon SNS

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 sumber daya Amazon SNS 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 memberikan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau API AWS. Sebaliknya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang coba dilakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Amazon SNS, lampirkan juga Amazon `ConsoleAccess` SNS `ReadOnly` AWS atau kebijakan terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna di Panduan Pengguna IAM.

Jenis kebijakan lainnya

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran 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 secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.

- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS yang dimiliki bisnis Anda secara terpusat. Jika Anda mengaktifkan semua fitur di sebuah organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau ke semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap pengguna root Akun AWS. Untuk informasi selengkapnya tentang Organizations dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. 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 eksplisit di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan ketika ada beberapa jenis kebijakan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau menggunakan AWS CLI atau AWS API secara terprogram.

```
{
    "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 izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak

dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat digunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Amazon SNS

Untuk melihat contoh kebijakan berbasis identitas Amazon SNS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Kebijakan berbasis sumber daya dalam Amazon SNS

Mendukung kebijakan berbasis sumber daya	Ya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM 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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya bagian dari membangun hubungan kepercayaan. Ketika pengguna utama dan sumber daya berada di Akun AWS yang berbeda, administrator IAM di akun tepercaya juga harus memberikan izin kepada entitas pengguna utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas tersebut. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, kebijakan berbasis identitas lainnya tidak diperlukan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dengan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Menggunakan kebijakan berbasis identitas dengan Amazon SNS

Topik

- Kebijakan IAM dan Amazon SNS bersama-sama
- Format ARN sumber daya Amazon SNS
- Tindakan API Amazon SNS
- Kunci kebijakan Amazon SNS
- Contoh kebijakan untuk Amazon SNS

Layanan Pemberitahuan Sederhana Amazon berintegrasi dengan AWS Identity and Access Management (IAM) sehingga Anda dapat menentukan tindakan Amazon SNS mana yang dapat dilakukan pengguna di Akun AWS dengan sumber daya Amazon SNS. Anda dapat menentukan topik tertentu dalam kebijakan. Misalnya, Anda dapat menggunakan variabel saat membuat kebijakan IAM yang memberikan pengguna tertentu dalam organisasi Anda izin untuk menggunakan tindakan `Publish` dengan topik tertentu di Akun AWS Anda. Untuk informasi selengkapnya, lihat Variabel Kebijakan dalam panduan Menggunakan IAM.

> **Important**
>
> Menggunakan Amazon SNS dengan IAM tidak mengubah cara Anda menggunakan Amazon SNS. Tidak ada perubahan pada tindakan Amazon SNS, dan tidak ada tindakan Amazon SNS baru yang terkait dengan pengguna dan pengendalian akses.

Untuk contoh kebijakan yang mencakup tindakan dan sumber daya Amazon SNS, lihat Contoh kebijakan untuk Amazon SNS.

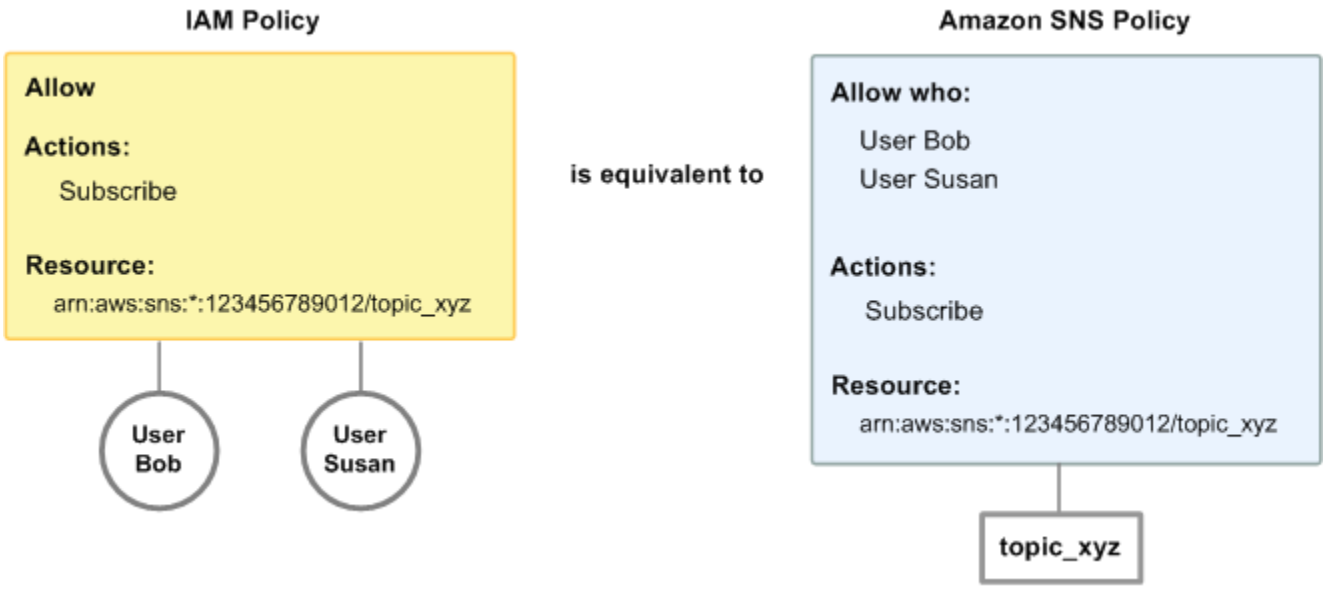
Kebijakan IAM dan Amazon SNS bersama-sama

Anda menggunakan kebijakan IAM untuk membatasi akses pengguna Anda ke tindakan dan topik Amazon SNS. Kebijakan IAM hanya dapat membatasi akses ke pengguna dalam akun AWS Anda, tidak ke Akun AWS lain.

Anda menggunakan kebijakan Amazon SNS dengan topik tertentu untuk membatasi siapa yang dapat bekerja dengan topik tersebut (misalnya, siapa yang dapat memublikasikan pesan ke topik, yang dapat berlangganan ke topik, dll.). Kebijakan Amazon SNS dapat memberikan akses ke Akun AWS, atau untuk pengguna dalam Akun AWS Anda.

Untuk memberikan izin untuk topik Amazon SNS Anda ke pengguna Anda, Anda dapat menggunakan kebijakan IAM, kebijakan Amazon SNS, atau keduanya. Umumnya, Anda dapat

mencapai hasil yang sama dengan baik. Sebagai contoh, diagram berikut menunjukkan kebijakan IAM dan kebijakan Amazon SNS yang setara. Kebijakan IAM memungkinkan tindakan `Subscribe` Amazon SNS untuk topik yang disebut topic_xyz di akun AWS Anda. Kebijakan IAM dilampirkan pada pengguna Bob dan Susan (yang berarti Bob dan Susan memiliki izin yang tercantum dalam kebijakan). Kebijakan Amazon SNS juga memberikan Bob dan Susan izin untuk mengakses `Subscribe` untuk 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.

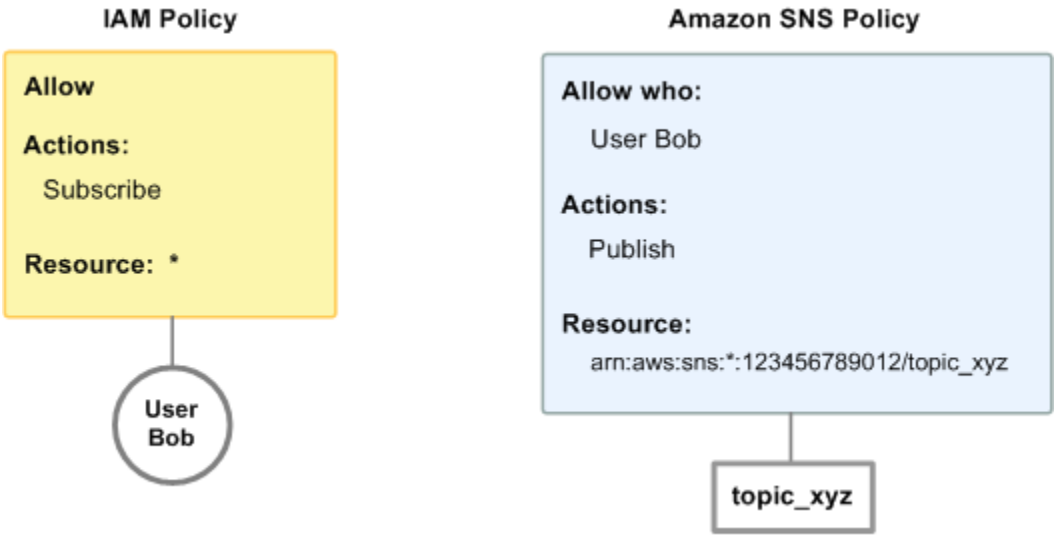
Ada satu perbedaan antara Kebijakan IAM AWS dan Amazon SNS: Sistem kebijakan Amazon SNS memungkinkan Anda memberikan izin kepada Akun AWS, sedangkan kebijakan IAM tidak.

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, baik kebijakan IAM maupun kebijakan Amazon SNS diterapkan pada Bob. Kebijakan IAM memberikan dia izin untuk`Subscribe` pada yang mana pun dari topik Akun AWS,

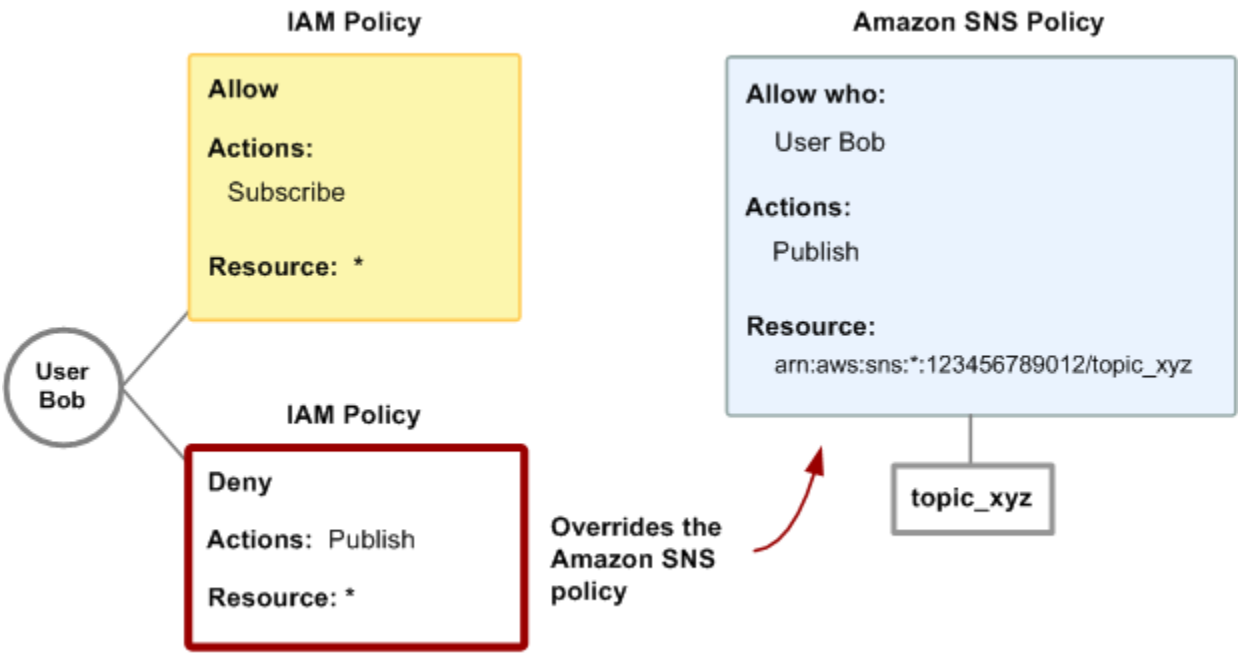
sedangkan kebijakan Amazon SNS memberikan dia izin untuk menggunakan `Publish` pada topik tertentu (topic_xyz). Diagram berikut menggambarkan konsep.



Jika Bob mengirimkan permintaan untuk berlangganan ke topik apa pun di akun AWS, kebijakan IAM akan mengizinkan tindakan. Jika Bob mengirimkan permintaan untuk memublikasikan pesan ke topic_xyz, kebijakan Amazon SNS 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 yang dapat dilakukan adalah menambahkan kebijakan IAM yang menolak Bob mengakses tindakan `Publish` pada semua topik. Kebijakan ketiga ini mengabaikan kebijakan Amazon SNS yang awalnya memberikan dia izin untuk memublikasikan ke topic_xyz, karena penolakan eksplisit selalu mengabaikan perizinan (untuk informasi lebih lanjut tentang logika evaluasi kebijakan, lihat Logika evaluasi). Diagram berikut menggambarkan konsep.



Untuk contoh kebijakan yang mencakup tindakan dan sumber daya Amazon SNS, lihat Contoh kebijakan untuk Amazon SNS. Untuk informasi selengkapnya tentang menulis kebijakan Amazon SNS, kunjungi dokumentasi teknis untuk Amazon SNS.

Format ARN sumber daya Amazon SNS

Untuk Amazon SNS, topik adalah satu-satunya jenis sumber daya yang dapat Anda tentukan dalam kebijakan. Berikut adalah format Amazon Resource Name (ARN) untuk topik:

```
arn:aws:sns:region:account_ID:topic_name
```

Untuk informasi lebih lanjut tentang ARN, kunjungi ARN dalam Panduan Pengguna IAM.

Example

Berikut ini adalah ARN untuk topik yang dinamai MyTopic di wilayah us-timur-2, milik 123456789012. Akun AWS

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

Example

Jika Anda memiliki topik yang disebutkan MyTopic di masing-masing Wilayah berbeda yang didukung Amazon SNS, Anda dapat menentukan topik dengan ARN berikut.

```
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_*
```

Untuk kenyamanan Anda, ketika Anda membuat topik, Amazon SNS mengembalikan ARN topik dalam respon.

Tindakan API Amazon SNS

Dalam kebijakan IAM, Anda dapat menentukan tindakan apa pun yang ditawarkan Amazon SNS. Namun, tindakan `ConfirmSubscription` dan `Unsubscribe` tidak memerlukan autentikasi, yang berarti meskipun Anda menetapkan tindakan tersebut dalam kebijakan, IAM tidak akan membatasi akses pengguna ke tindakan tersebut.

Setiap tindakan yang Anda tentukan dalam kebijakan harus diawali dengan string huruf kecil `sns:`. Untuk menentukan semua tindakan Amazon SNS, misalnya, Anda akan menggunakan `sns:*`. Untuk daftar tindakan, kunjungi Referensi API Layanan Notifikasi Sederhana Amazon.

Kunci kebijakan Amazon SNS

Amazon SNS menerapkan kunci kebijakan di seluruh AWS berikut, ditambah beberapa kunci khusus layanan.

Untuk daftar kunci kondisi yang didukung oleh masing-masingLayanan AWS, lihat Tindakan, sumber daya, dan kunci kondisi untuk Layanan AWS dalam Panduan Pengguna IAM. Untuk daftar kunci kondisi yang dapat digunakan dalam beberapaLayanan AWS, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Amazon SNS menggunakan kunci khusus layanan berikut. Gunakan kunci ini dalam kebijakan yang membatasi akses ke permintaan `Subscribe`.

- sns:endpoint—URL, alamat email, atau ARN dari permintaan `Subscribe` atau langganan yang telah 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 Amazon SNS.

> **Note**
>
> Di masa depan, Amazon SNS mungkin menambahkan tindakan baru yang harus dimasukkan secara logis ke dalam salah satu kebijakan berikut, berdasarkan tujuan yang dinyatakan kebijakan.

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 di 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 kebijakan Amazon SNS atau kebijakan IAM untuk memungkinkan mitra memublikasikan ke topik tertentu. Jika mitra Anda memiliki Akun AWS, mungkin akan lebih mudah untuk menggunakan kebijakan Amazon SNS. Namun, siapa pun di perusahaan mitra yang memiliki kredensial keamanan AWS dapat memublikasikan pesan ke topik. Contoh ini mengasumsikan bahwa Anda ingin membatasi akses ke orang (atau aplikasi) tertentu. Untuk melakukan ini, Anda perlu memperlakukan mitra seperti pengguna dalam perusahaan Anda sendiri, dan menggunakan kebijakan IAM bukan kebijakan Amazon SNS.

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 tindakan Amazon SNS selain topik apa pun `Publish` selain topik apa pun selain. WidgetPartnerTopic Hal ini hanya diperlukan jika terdapat kebijakan luas di tempat lain dalam sistem yang memberikan pengguna akses luas ke Amazon SNS.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Menggunakan kredensial sementara dengan Amazon SNS

Selain membuat pengguna IAM dengan kredensial keamanan mereka sendiri, IAM juga memungkinkan Anda memberikan kredensial keamanan sementara kepada pengguna apa pun yang mengizinkan pengguna ini mengakses layanan dan sumber daya AWS. Anda dapat mengelola pengguna yang memiliki Akun AWS; pengguna ini adalah pengguna IAM. Anda juga dapat mengelola pengguna untuk sistem Anda yang tidak memiliki Akun AWS; pengguna ini disebut pengguna gabungan. Selain itu, "pengguna" dapat berupa aplikasi yang Anda buat untuk mengakses sumber daya AWS Anda.

Anda dapat menggunakan kredensial keamanan sementara ini dalam mengajukan permintaan kepada Amazon SNS. Pustaka API menghitung nilai tanda tangan yang diperlukan menggunakan kredensial tersebut untuk mengautentikasi permintaan Anda. Jika Anda mengirimkan permintaan menggunakan kredensial kedaluwarsa, Amazon SNS menolak permintaan tersebut.

Untuk informasi selengkapnya tentang dukungan IAM untuk kredensial keamanan sementara, kunjungi Memberikan Akses Sementara untuk Sumber Daya AWS Anda dalam Menggunakan IAM.

Example Menggunakan kredensial keamanan sementara untuk mengautentikasi permintaan Amazon SNS

Contoh berikut menunjukkan cara mendapatkan kredensial keamanan sementara untuk mengautentikasi permintaan Amazon SNS.

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Izin API Amazon SNS: Tindakan dan referensi sumber daya

Daftar berikut memberikan informasi khusus untuk implementasi pengendalian akses oleh Amazon SNS:

- 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.)

Nama	Kuota maksimum
Sumber daya	1 (0 tidak valid. Nilai harus sesuai dengan ARN topik kebijakan.)

Tindakan kebijakan Amazon SNS yang valid

Amazon SNS mendukung tindakan yang ditunjukkan dalam 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: ListSubscriptionsByTopic	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 menerbitkan dan menerbitkan batch ke topik atau titik akhir. Untuk informasi selengkapnya, lihat Menerbitkan dan PublishBatchdi Referensi API 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.

Tindakan	Deskripsi
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—URL, alamat email, atau ARN dari permintaan `Subscribe` atau langganan yang telah 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**
>
> Ketika Anda menggunakan kebijakan untuk mengontrol akses oleh sns:Endpoint, perhatikan bahwa masalah DNS dapat mempengaruhi resolusi nama titik akhir di masa depan.

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 dan IAM.

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 sumber daya Amazon SNS 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 membutuhkan bantuan, hubungi administrator AWS 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 Amazon SNS.

Sebagian Layanan AWS mengizinkan Anda untuk memberikan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait-layanan. Untuk melakukan tindakan tersebut, Anda harus memiliki izin untuk memberikan peran pada layanan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon SNS. 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 membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon SNS saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna 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 (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Amazon SNS mendukung fitur-fitur ini, lihat. Bagaimana Amazon Simple Notification Service bekerja dengan IAM
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke pihak ketiga Akun AWS, lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Pencatatan dan Pemantauan di Amazon SNS

Bagian ini menyediakan informasi tentang pencatatan dan pemantauan Amazon SNS topik.

Topik

- Mencatat panggilan API Amazon SNS menggunakan CloudTrail
- Memantau topik Amazon SNS menggunakan CloudWatch

Mencatat panggilan API Amazon SNS menggunakan CloudTrail

Amazon SNS terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon SNS. CloudTrail menangkap panggilan API untuk Amazon SNS sebagai acara. Panggilan yang direkam mencakup panggilan dari konsol Amazon SNS dan panggilan kode ke operasi API Amazon SNS. 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 Amazon SNS, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, termasuk cara mengonfigurasi dan mengaktifkannya, lihat Panduan AWS CloudTrail Pengguna.

Informasi Amazon SNS di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas peristiwa yang didukung terjadi di Amazon SNS, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di Akun AWS Anda. Untuk informasi lain, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan kejadian yang sedang berlangsung di Akun AWS Anda, termasuk kejadian untuk Amazon SNS, buatlah jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di dalam konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS 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 yang berikut:

- Ikhtisar untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Kontrol 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 hal berikut:

- Bahwa permintaan dibuat dengan kredensial pengguna root atau pengguna AWS Identity and Access Management (IAM).
- Bahwa permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Bahwa permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail.

Peristiwa pesawat data di CloudTrail

Untuk mengaktifkan pencatatan tindakan API berikut dalam CloudTrail file, Anda harus mengaktifkan pencatatan aktivitas API bidang data CloudTrail. Untuk informasi selengkapnya, lihat Mencatat peristiwa data di Panduan AWS CloudTrail Pengguna.

Peristiwa bidang data juga dapat difilter berdasarkan jenis sumber daya, untuk kontrol terperinci atas panggilan Amazon SNS API yang ingin Anda log dan bayar secara selektif. CloudTrail Misalnya, dengan menentukan `AWS::SNS::Topic` sebagai tipe sumber daya, Anda dapat mencatat panggilan ke `Publish` dan tindakan `PublishBatch` API untuk topik. Demikian juga, dengan menentukan `AWS::SNS::PlatformEndpoint` sebagai tipe sumber daya, Anda dapat mencatat panggilan ke tindakan Publikasikan API untuk titik akhir platform. Untuk informasi selengkapnya, lihat `AdvancedEventSelector`di Referensi AWS CloudTrail API.

> **Note**
>
> Jenis sumber daya Amazon SNS tidak `AWS::SNS::PhoneNumber` dicatat oleh. CloudTrail

API pesawat data Amazon SNS

- `Publish`
- `PublishBatch`

Contoh: entri file log Amazon SNS

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. Peristuwa mewakili satu permintaan dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut 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 topik Amazon SNS menggunakan CloudWatch

Amazon SNS dan Amazon CloudWatch terintegrasi sehingga Anda dapat mengumpulkan, melihat, dan menganalisis metrik untuk setiap notifikasi Amazon SNS yang aktif. Setelah Anda mengonfigurasi CloudWatch untuk Amazon SNS, Anda dapat memperoleh wawasan yang lebih baik tentang kinerja topik Amazon SNS, pemberitahuan push, dan pengiriman SMS Anda. Misalnya, Anda dapat mengatur alarm untuk mengirimkan notifikasi email jika ambang batas tertentu dipenuhi untuk metrik Amazon SNS, seperti `NumberOfNotificationsFailed`. Untuk daftar semua metrik yang CloudWatch dikirimkan Amazon SNS, lihat. Metrik Amazon SNS Untuk informasi selengkapnya tentang notifikasi push Amazon SNS, lihat Notifikasi push seluler.

> **Note**
>
> Metrik yang Anda konfigurasikan CloudWatch untuk topik Amazon SNS Anda dikumpulkan dan didorong secara otomatis pada interval 1 CloudWatch menit. Metrik ini dikumpulkan pada semua topik yang memenuhi CloudWatch pedoman untuk aktif. Topik dianggap aktif hingga CloudWatch enam jam dari aktivitas terakhir (yaitu, panggilan API apa pun) pada topik tersebut.
>
> Tidak ada biaya untuk metrik Amazon SNS yang dilaporkan CloudWatch; mereka disediakan sebagai bagian dari layanan Amazon SNS.

Lihat CloudWatch metrik untuk Amazon SNS

Anda dapat memantau metrik untuk Amazon SNS menggunakan konsol CloudWatch, CloudWatch antarmuka baris perintah (CLI) sendiri, atau menggunakan API secara terprogram. CloudWatch 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 All metrics (Semua metrik), pilih SNS, lalu pilih salah satu dimensi berikut:
   - Negara, Jenis SMS
   - 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 pesan Amazon SNS yang diterbitkan untuk periode 1 menit selama rentang waktu 6 jam akan ditampilkan.
5. Untuk melihat metrik penggunaan Amazon SNS, pada tab Semua metrik, pilih Penggunaan, dan pilih metrik penggunaan Amazon SNS 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 menggunakan metrik Amazon SNS untuk menemukan metrik yang ingin Anda aktifkan alarmnya. 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 topik Amazon SNS 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 Create New Email Topic (Buat Topik Email Baru) untuk membuat topik Amazon SNS baru, alamat email harus diverifikasi sebelum menerima

notifikasi. 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 topik Amazon SNS Anda. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik Valid: Sum
AWS/SNS	`NumberOfNotificationsDelivered`	Jumlah pesan berhasil dikirim dari topik Amazon SNS Anda ke endpoint berlangganan. Agar upaya pengiriman berhasil, langganan dari endpoint harus menerima pesan tersebut. Langganan menerima pesan jika.) Tidak memiliki kebijakan filter atau b.) kebijakan filter mencakup atribut yang cocok dengan yang ditetapkan ke pesan. Jika langganan menolak

Namespace	Metrik	Deskripsi
		pesan, upaya pengiriman tidak dihitung untuk metrik ini. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik Valid: Sum

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFailed`	Jumlah pesan yang gagal dikirim oleh Amazon SNS. Untuk Amazon SQS, email, SMS, atau endpoint push seluler, metrik bertambah 1 saat Amazon SNS berhenti mencoba pengiriman pesan. Untuk endpoint HTTP atau HTTPS, 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 pengulangan untuk endpoint HTTP. Untuk informasi selengkapnya, lihat Pengiriman ulang pesan Amazon SNS. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, 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: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Jumlah pesan yang ditolak oleh kebijakan filter langganan untuk pemfilteran berbasis atribut. Unit: CountValid Dimensi: Aplikasi, PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Jumlah pesan yang ditolak oleh kebijakan filter langganan untuk pemfilteran berbasis muatan. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Jumlah pesan yang ditolak oleh kebijakan filter langganan karena atribut pesan tidak valid — misalnya, karena atribut JSON salah diformat. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	Jumlah pesan yang ditolak oleh kebijakan filter langganan karena pesan tidak memiliki atribut. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Jumlah pesan yang ditolak oleh kebijakan filter langganan karena badan pesan tidak valid untuk pemfilteran — misalnya, badan pesan JSON tidak valid. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Jumlah pesan yang telah dipindahkan ke antrean surat mati. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Jumlah pesan yang tidak dapat dipindahkan ke antrean surat mati. Unit: Count (Jumlah) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Sum, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`PublishSize`	Ukuran pesan yang diterbitkan. Unit: Bytes (Byte) Dimensi yang Valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang Valid: Minimum, Maksimum, Rata-rata dan Jumlah

Namespace	Metrik	Deskripsi
AWS/SNS	`SMSMonthToDateSpentUSD`	Biaya yang telah Anda dapatkan sejak awal bulan kalender saat ini untuk mengirim pesan SMS. Anda dapat mengatur alarm untuk metrik ini untuk mengetahui kapan month-to-date tagihan Anda mendekati kuota belanja SMS bulanan untuk akun Anda. Ketika Amazon SNS menentukan bahwa mengirim pesan SMS akan dikenakan biaya yang melebihi kuota ini, Amazon SNS akan berhenti menerbitkan pesan SMS dalam beberapa menit. Selengkapnya tentang pengaturan kuota belanja SMS bulanan Anda, atau untuk informasi tentang permintaan kenaikan kuota belanja dengan AWS, lihat Mengatur preferensi olahpesan SMS. Unit: USD Dimensi yang Valid: PhoneNumber Statistik yang Valid: Maksimum

Namespace	Metrik	Deskripsi
AWS/SNS	`SMSSuccessRate`	Tingkat keberhasilan pengiriman pesan SMS. Unit: Count (Jumlah) Dimensi yang Valid: PhoneNumber Statistik yang Valid: Jumlah, Rata-rata, Sampel Data

Dimensi untuk metrik Amazon SNS

Amazon Simple Notification Service mengirimkan dimensi berikut ke CloudWatch.

Dimensi	Deskripsi
`Application`	Filter pada objek aplikasi, yang mewakili aplikasi dan perangkat yang terdaftar dengan salah satu layanan notifikasi push yang didukung, seperti APN dan FCM.
`Application,Platform`	Filter pada objek aplikasi dan platform, di mana objek platform untuk layanan notifikasi push yang didukung, seperti APN dan FCM.
`Country`	Filter di negara tujuan atau wilayah pesan SMS. Negara atau wilayah diwakili oleh kode ISO 3166-1 alpha-2.
`PhoneNumber`	Filter pada nomor telepon saat Anda mempublikasikan SMS langsung ke nomor telepon (tanpa topik).
`Platform`	Filter pada objek platform untuk layanan notifikasi push, seperti APN dan FCM.
`TopicName`	Filter pada nama topik Amazon SNS.
`SMSType`	Filter pada jenis pesan SMS. Bisa promotional (promosi) atau transactional (transaksional).

Metrik penggunaan Amazon SNS

Amazon Simple Notification Service mengirimkan metrik penggunaan berikut ke CloudWatch.

Namespace	Layanan	Metrik	Resource	Jenis	Deskripsi
AWS/Penggunaan	SNS	`ResourceCount`	`NumberOfMessagesPublishedPerAccount`	Resource	• Jumlah pesan yang dipublikasikan ke topik Amazon SNS Anda di seluruh akun AndaAWS. • Unit: Tidak ada • Statistik Valid: Sum
AWS/Penggunaan	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Resource	• Perkiraan jumlah topik di seluruh AWS akun Anda. • Unit: Tidak ada • Statistik yang Valid: Rata-rata, Minimum, Maksimum, Jumlah

Namespace	Layanan	Metrik	Resource	Jenis	Deskripsi
AWS/Penggunaan	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Resource	• Perkiraan jumlah kebijakan filter di seluruh AWS akun Anda. • Unit: Tidak ada • Statistik yang Valid: Rata-rata, Minimum, Maksimum, Jumlah
AWS/Penggunaan	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Resource	• Perkiraan jumlah langganan yang tertunda di seluruh AWS akun Anda. • Unit: Tidak ada • Statistik yang Valid: Rata-rata, Minimum, Maksimum, Jumlah

Namespace	Layanan	Metrik	Resource	Jenis	Deskripsi
AWS/Penggunaan	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber` • `DeleteTopic`	API	• Jumlah panggilan API untuk Amazon SNS API yang dipilih di seluruh akun AndaAWS. • Unit: Tidak ada • Statistik Valid: Sum

Namespace	Layanan	Metrik	Resource	Jenis	Deskripsi
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

Namespace	Layanan	Metrik	Resource	Jenis	Deskripsi
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

Namespace	Layanan	Metrik	Resource	Jenis	Deskripsi
			• Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Validasi Kepatuhan untuk Amazon SNS

Auditor pihak ke tiga menilai keamanan dan kepatuhan Amazon SNS sebagai bagian dari beberapa program kepatuhan AWS, termasuk Health Insurance Portability and Accountability Act (HIPAA).

Untuk daftar layanan AWS dalam cakupan program kepatuhan tertentu, lihat AWS Layanan dalam Cakupan melalui Program Kepatuhan. Untuk informasi umum, lihat Program Kepatuhan AWS.

Anda bisa mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Pengunduhan Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda ketika menggunakan Amazon SNS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan perundangan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu dengan kepatuhan:

- Panduan Keamanan dan Kepatuhan Quick Start – Deployment ini membahas pertimbangan arsitektur dan menyediakan langkah untuk men-deploy lingkungan dasar yang berfokus pada keamanan dan kepatuhan pada AWS.
- Merancang Laporan Resmi Keamanan dan Kepatuhan HIPAA – Laporan resmi ini menjelaskan cara perusahaan dapat menggunakan AWS untuk membuat aplikasi yang patuh-HIPAA.
- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.

- Mengevaluasi Sumber Daya dengan Aturan di Panduan Developer AWS Config – Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda dalam mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub – Layanan AWS ini memberikan pandangan komprehensif tentang status keamanan Anda dalam AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di Amazon SNS

Ketahanan di Amazon SNS dipastikan dengan memanfaatkan infrastruktur global, yang berputar AWS 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, pelanggan Amazon SNS 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 topik Amazon SNS 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 tindakan AWS API untuk mengakses Amazon SNS melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Selain itu, klien harus mendukung suite sandi dengan Perfect Forward Secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Diffie-Hellman Ephemeral (ECDHE).

Anda harus menandatangani permintaan menggunakan access key ID dan secret access key yang terhubung dengan IAM utama. Atau, Anda dapat menggunakanAWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk permintaan penandatanganan.

Anda dapat menghubungi tindakan API 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 kebijakan Amazon SNS untuk mengontrol akses dari Amazon VPC endpoint tertentu atau VPC tertentu. Ini secara efektif mengisolasi akses jaringan ke topik Amazon SNS tertentu dari hanya VPC tertentu dalam jaringan. AWS Untuk informasi selengkapnya, lihat Membatasi publikasi ke topik Amazon SNS hanya dari VPC endpoint tertentu.

Praktik terbaik keamanan Amazon SNS

AWS menyediakan banyak fitur keamanan untuk Amazon SNS. 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 adalah praktik terbaik keamanan pencegahan untuk Amazon SNS.

Topik

- Memastikan topik tidak dapat diakses secara publik
- Menerapkan akses hak istimewa yang paling rendah
- Menggunakan IAM role untuk aplikasi dan layanan AWS yang memerlukan akses Amazon SNS
- Menerapkan enkripsi sisi server
- Menegakkan enkripsi data saat transit
- Pertimbangkan titik akhir VPC untuk mengakses Amazon SNS
- Pastikan langganan tidak dikonfigurasi untuk dikirimkan ke endpoint http mentah

Memastikan topik tidak dapat diakses secara publik

Kecuali jika Anda meminta siapa pun secara eksplisit di internet untuk dapat membaca atau menulis ke topik Amazon SNS, pastikan bahwa topik Anda tidak dapat diakses publik (dapat diakses oleh semua orang di dunia atau oleh pengguna AWS yang diautentikasi).

- 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, untuk topik apa izin tersebut, dan tindakan API tertentu yang ingin Anda izinkan untuk topik. 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 langganan, 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:

- Identity and access management di Amazon SNS
- Izin API Amazon SNS: Tindakan dan referensi sumber daya

Menggunakan IAM role untuk aplikasi dan layanan AWS yang memerlukan akses Amazon SNS

Aplikasi atau layanan AWS, seperti Amazon EC2, untuk mengakses topik Amazon SNS, mereka harus menggunakan kredensial AWS yang valid di permintaan AWS API. Karena kredensial ini tidak diputar secara otomatis, sebaiknya Anda tidak menyimpan kredensial AWS secara langsung di aplikasi atau instans EC2.

Sebaliknya, Anda harus menggunakan IAM role dalam mengelola kredensial sementara untuk aplikasi atau layanan yang perlu mengakses Amazon SNS. Saat Anda menggunakan peran, Anda tidak perlu mendistribusikan kredensia jangka panjang (seperti nama pengguna dan kata sandi atau access key) ke instans EC2 atauAWS layanan, sepertiAWS Lambda. Sebaliknya, peran menyediakan izin sementara yang dapat digunakan aplikasi saat mereka melakukan panggilan ke sumber daya AWS lainnya.

Untuk informasi selengkapnya, lihat IAM Role dan Skenario Umum untuk Peran: Pengguna, Aplikasi, dan Layanan di Panduan Pengguna IAM.

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 di tingkat pesan saat menyimpannya, dan mendekripsi pesan untuk Anda saat mengaksesnya. SSE menggunakan 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 Enkripsi diam dan Manajemen kunci.

Menegakkan enkripsi data saat transit

Mungkin, namun tidak disarankan, untuk mempublikasikan pesan yang tidak dienkripsi selama transit menggunakan HTTP. Namun, Anda tidak dapat menggunakan HTTP saat menerbitkan topik SNS terenkripsi.

AWS merekomendasikan Anda untuk menggunakan HTTPS bukan HTTP. Saat Anda menggunakan HTTPS, pesan akan dienkripsi secara otomatis selama transit, meskipun topik SNS itu sendiri tidak dienkripsi. Tanpa HTTPS, penyerang berbasis jaringan dapat menguping lalu lintas jaringan atau memanipulasinya menggunakan serangan seperti man-in-the-middle.

Untuk menegakkan hanya koneksi terenkripsi melalui HTTPS, tambahkan kondisi `aws:SecureTransport` di kebijakan IAM yang dilampirkan ke topik SNS yang tidak dienkripsi. Hal ini memaksa penerbit pesan untuk menggunakan HTTPS bukan 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 titik akhir VPC untuk mengakses Amazon SNS

Jika Anda memiliki topik yang harus berinteraksi dengan Anda, tetapi topik ini harus benar-benar tidak terekspos internet, gunakan titik akhir VPC untuk membatasi akses topik hanya ke host dalam VPC tertentu. Anda dapat menggunakan kebijakan topik untuk mengontrol akses ke topik dari titik akhir Amazon VPC tertentu atau dari VPC tertentu.

Titik akhir VPC Amazon SNS menyediakan dua cara untuk mengontrol akses ke pesan Anda:

- Anda dapat mengontrol permintaan, pengguna, atau grup yang diizinkan melalui VPC endpoint tertentu.
- Anda dapat mengendalikan VPC atau titik akhir VPC mana yang memiliki akses ke topik Anda menggunakan kebijakan topik.

Untuk informasi selengkapnya, lihat Membuat titik akhir dan Membuat kebijakan VPC endpoint Amazon untuk Amazon SNS.

Pastikan langganan tidak dikonfigurasi untuk dikirimkan ke endpoint http mentah

Hindari mengonfigurasi langganan untuk dikirim ke endpoint 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`.

Pemecahan masalah topik Amazon SNS

Bagian ini menyediakan informasi tentang pemecahan masalah topik Amazon SNS.

Memecahkan masalah topik Amazon SNS menggunakan AWS X-Ray

AWS X-Ray mengumpulkan data tentang permintaan yang dilayani aplikasi Anda, dan memungkinkan Anda melihat dan memfilter data untuk mengidentifikasi potensi masalah dan peluang pengoptimalan. Untuk setiap permintaan yang dilacak ke aplikasi Anda, Anda dapat melihat informasi yang mendetail tentang permintaan, respons, dan panggilan yang dibuat aplikasi Anda untuk membawa sumber daya AWS, microservice, database dan API web HTTP ke hilir.

Anda dapat menggunakan X-Ray dengan Amazon SNS untuk melacak dan menganalisis pesan yang berjalan 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 Developer AWS X-Ray.

Penelusuran aktif di Amazon SNS

Anda dapat menggunakan AWS X-Ray untuk melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui topik Amazon SNS Anda ke Amazon Data Firehose, Amazon AWS LambdaSQS, dan langganan titik akhir HTTP/S Anda. Karena X-Ray memberi Anda end-to-end tampilan seluruh permintaan, Anda dapat melihat apa yang memanggil topik Amazon SNS Anda, dan apa yang menjadi 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**
>
> Topik Amazon SNS 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 meneruskan jejak, bahkan jika penelusuran X-Ray tidak diaktifkan pada API.

Amazon SNS mendukung penelusuran X-Ray untuk topik standar dan FIFO. Anda dapat mengaktifkan X-Ray untuk topik Amazon SNS dengan menggunakan konsol Amazon SNS, Amazon SNS `SetTopicAttributes` API, Referensi CLI Layanan Pemberitahuan Sederhana Amazon, atau. AWS CloudFormation

Untuk mempelajari selengkapnya tentang menggunakan Amazon SNS dengan X-Ray, lihat Amazon SNS AWS X-Ray dan di AWS X-Ray Panduan Pengembang.

Topik

- Izin penelusuran aktif
- Mengaktifkan penelusuran aktif pada topik Amazon SNS (konsol)
- Mengaktifkan penelusuran aktif pada topik Amazon SNS (SDK) AWS
- Mengaktifkan penelusuran aktif pada topik Amazon SNS (CLI) AWS
- Mengaktifkan penelusuran aktif pada topik Amazon SNS () AWS CloudFormation
- Memverifikasi penelusuran aktif diaktifkan untuk topik Anda
- Menguji penelusuran aktif

Izin penelusuran aktif

Saat menggunakan konsol Amazon SNS, Amazon SNS mencoba membuat izin yang diperlukan untuk topik Amazon SNS untuk memanggil X-Ray. Upaya dapat ditolak jika Anda tidak memiliki izin yang cukup untuk menggunakan konsol Amazon SNS. Lihat informasi yang lebih lengkap di Identity and access management di Amazon SNS dan Contoh kasus untuk pengendalian akses Amazon SNS.

Saat menggunakan CLI, Anda harus mengonfigurasi izin secara manual. Izin tersebut dikonfigurasi menggunakan kebijakan sumber daya. Untuk informasi lebih lanjut tentang penggunaan izin yang diperlukan dalam X-Ray, lihat Amazon SNS AWS X-Ray dan.

Mengaktifkan penelusuran aktif pada topik Amazon SNS (konsol)

Saat penelusuran aktif diaktifkan pada topik Amazon SNS, ia membaca ID jejak, mengirimkan data ke pelanggan berdasarkan ID jejak, dan menyebarkan ID jejak ke layanan hilir.

1. Masuk ke Konsol Amazon SNS.
2. Pilih topik atau buat yang baru. Untuk detail selengkapnya tentang membuat topik, lihat Membuat topik Amazon SNS.
3. Pada halaman Buat topik, di bagian Detail, pilih jenis topik: FIFO atau 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 topik Amazon SNS Anda, Anda dapat menggunakan peta layanan X-Ray untuk melihat end-to-end jejak dan peta layanan untuk topik tersebut.

Mengaktifkan penelusuran aktif pada topik Amazon SNS (SDK) AWS

Contoh kode berikut menunjukkan cara mengaktifkan penelusuran aktif pada topik Amazon SNS dengan menggunakan AWS SDK for 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 topik Amazon SNS (CLI) AWS

Contoh kode berikut menunjukkan cara mengaktifkan penelusuran aktif pada topik Amazon SNS dengan menggunakan CLI. AWS

```
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 SNS () AWS CloudFormation

AWS CloudFormationTumpukan berikut menunjukkan cara mengaktifkan penelusuran aktif pada topik Amazon SNS.

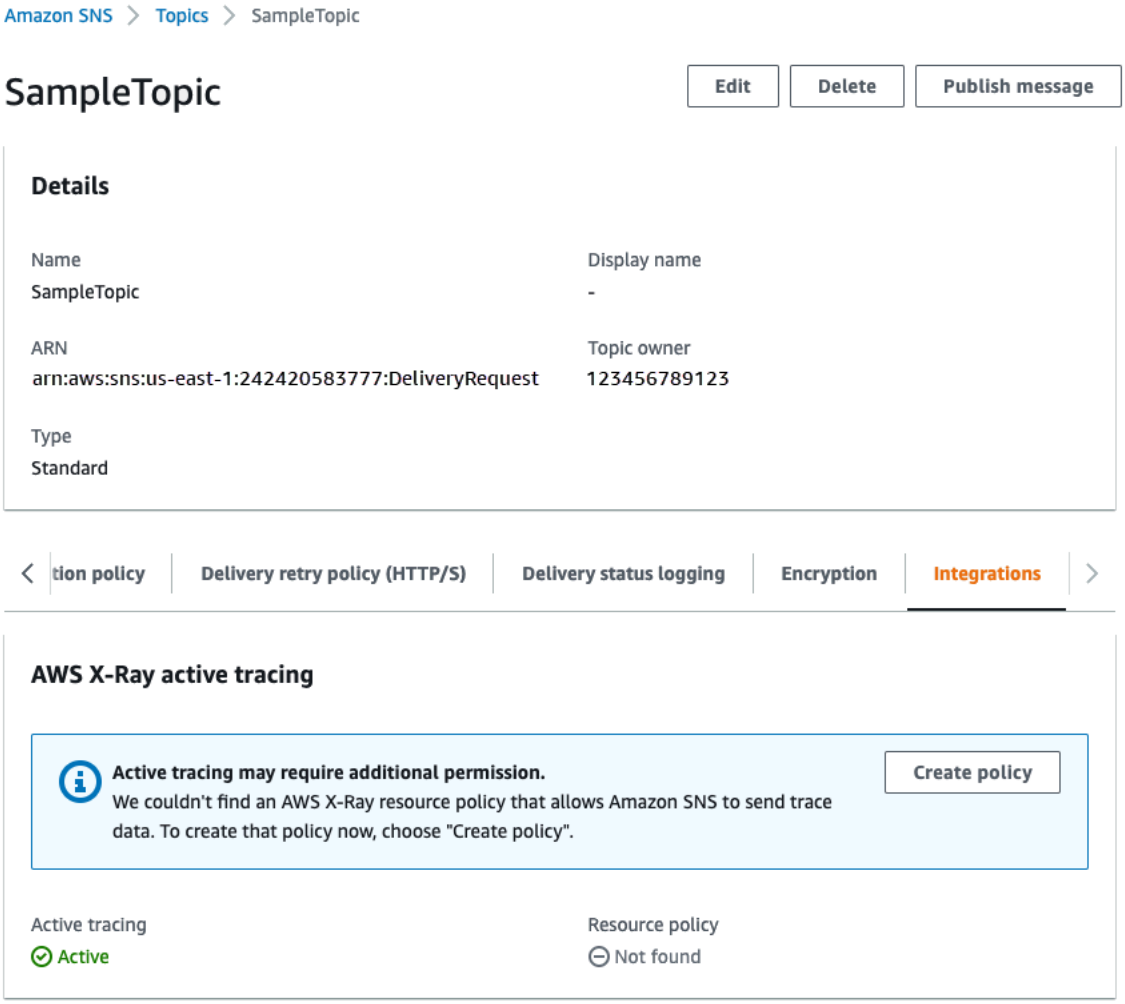
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Memverifikasi penelusuran aktif diaktifkan untuk topik Anda

Anda dapat menggunakan konsol Amazon SNS untuk memverifikasi apakah penelusuran aktif diaktifkan untuk topik Anda, atau bila kebijakan sumber daya gagal ditambahkan.

1. Masuk ke konsol Amazon SNS.
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 Konsol Amazon SNS.
2. Buat topik Amazon SNS. 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 topik Amazon SNS. Untuk detail tentang cara melakukannya, lihat Cara menerbitkan pesan ke topik Amazon SNS menggunakan AWS Management Console.
5. Gunakan peta layanan X-Ray untuk melihat end-to-end jejak dan peta layanan untuk topik tersebut.


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

Riwayat dokumentasi

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
Dukungan Kanada Barat (Calgary) untuk topik FIFO	Amazon SNS mendukung topik FIFO di Kanada Barat (Calgary).	Maret 28, 2024
Dukungan SMS Amazon SNS di lima wilayah baru	Amazon SNS menambahkan dukungan SMS ke wilayah berikut: Asia Pasifik (Hyderabad), Asia Pasifik (Melbourne), Timur Tengah (UEA), Eropa (Zurich). dan Eropa (Spanyol).	Februari 8, 2024
Dukungan HTTP v1 Firebase Cloud Messaging (FCM)	Amazon SNS mendukung kredensi FCM v1.	Januari 18, 2024
Amazon SNS SMS didukung di Asia Pasifik (Jakarta)	Amazon SNS mendukung pesan SMS di Asia Pasifik (Jakarta).	14 Desember 2023
AWS CloudFormation dukungan untuk mengonfigurasi topik `DeliveryStatusLogging` Amazon SNS	AWS CloudFormation dukungan tersedia untuk mengonfigurasi `DeliveryStatusLogging` saat membuat atau memperbarui topik Amazon SNS.	Desember 7, 2023

Operator penyaringan pesan baru ditambahkan	Sekarang Anda dapat menggunakan pencocoka n akhiran, kasus sama-abai kan, dan operator OR saat memfilter pesan Amazon SNS.	16 November 2023
Support ditambahkan untuk pengarsipan dan pemutaran ulang pesan	Pemilik topik dapat mengarsip kan 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 antrian Amazon SQS FIFO atau antrian standar ke topik Amazon SNS FIFO. Hanya antrian Amazon SQS FIFO yang menjamin pesan diterima secara berurutan dan tanpa duplikat.	14 September 2023
Dukungan SMS ditambahkan untuk Israel (Tel Aviv)	Amazon SNS SMS sekarang didukung di wilayah Israel (Tel Aviv).	28 Agustus 2023

Support untuk penelusuran aktif X-Ray ditambahkan ke topik FIFO	Sebelumnya hanya didukung dengan topik standar Amazon SNS, AWS X-Ray sekarang melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui topik FIFO Anda ke langganan Amazon Data Firehose, AWS Lambda Amazon SQS, dan HTTP/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 standar Amazon SNS Anda ke langganan Amazon Data Firehose, AWS Lambda Amazon SQS, dan HTTP/S endpoint Anda.	Februari 8, 2023
Pendaftaran ID Pengirim Singapura	Petunjuk ditambahkan untuk mendaftarkan ID Pengirim 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
Algoritma hash SHA256 ditambahkan untuk penandatanganan pesan Amazon SNS	Support ditambahkan untuk algoritma hash SHA256 saat menggunakan penandatanganan pesan Amazon SNS.	15 September 2022
Wilayah tambahan ditambahkan ke pesan SMS	Amazon SNS mendukung pesan SMS 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 topik Amazon SNS Anda dengan menggunakan kebijakan perlindungan data untuk mengaudit dan memblokir informasi sensitif yang berpindah antar aplikasi atau layanan. AWS	September 8, 2022
Proses pendaftaran baru untuk nomor bebas pulsa	Support ditambahkan untuk mengirim pesan Amazon SNS 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 tindakan API termasuk dan. `Publish` `PublishBatch` ABAC adalah strategi otorisasi yang mendefinisikan izin akses berdasarkan tag yang dapat dilampirkan ke sumber daya IAM, seperti pengguna dan peran IAM, dan ke sumber daya, AWS seperti topik Amazon SNS, untuk menyederhanakan manajemen izin.	10 Januari 2022
Support untuk otentikasi berbasis token Apple untuk pemberitahuan push	Anda dapat mengotorisasi 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 pesan SMS baru ditempatkan di kotak pasir SMS	Sandbox SMS ada untuk membantu mencegah penipuan dan penyalahg unaan, dan untuk membantu melindungi reputasi Anda sebagai pengirim. Saat AWS akun Anda berada di kotak pasir SMS, Anda dapat mengirim pesan SMS hanya ke nomor telepon tujuan yang diverifikasi.	1 Juni 2021

Pengirim pesan SMS baru ditempatkan di kotak pasir SMS	Sandbox SMS ada untuk membantu mencegah penipuan dan penyalahg unaan, dan untuk membantu melindungi reputasi Anda sebagai pengirim. Saat AWS akun Anda berada di kotak pasir SMS, Anda dapat mengirim pesan SMS hanya ke nomor telepon tujuan yang diverifikasi.	1 Juni 2021
Atribut baru untuk mengirim pesan SMS ke penerima di India	Dua atribut baru, ID Entitas dan ID templat, sekarang diperlukan untuk mengirim pesan SMS 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. Sebaliknya, Anda dapat menggunakan kode pendek, nomor bebas pulsa, atau nomor originasi tipe baru yang disebut 10DLC.	16 Februari 2021
Dukungan untuk metrik Amazon CloudWatch 1 menit	CloudWatch Metrik 1 menit untuk Amazon SNS sekarang tersedia di AWS semua 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

Dukungan untuk topik Amazon SNS FIFO	Untuk mengintegrasikan aplikasi terdistribusi yang memerlukan konsistensi data hampir secara langsung, Anda dapat menggunakan topik Amazon SNS masuk pertama, keluar pertama (FIFO) dengan antrean Amazon SQS FIFO.	22 Oktober 2020
Perpustakaan Klien Diperpanjang Amazon SNS untuk Java tersedia	Anda dapat menggunakan perpustakaan ini untuk mempublikasikan pesan Amazon SNS besar.	25 Agustus 2020
SSE tersedia di Wilayah China	Enkripsi sisi server (SSE) untuk Amazon SNS tersedia di Wilayah China.	20 Januari 2020
Support untuk menggunakan DLQ untuk menangkap pesan yang tidak terkirim	Untuk menangkap pesan yang tidak terkirim, Anda dapat menggunakan antrean surat mati (DLQ) Amazon SQS dengan langganan Amazon SNS.	14 November 2019
Support untuk menentukan nilai header APN kustom	Anda dapat menentukan nilai header APN khusus.	18 Oktober 2019
Support untuk bidang header `apns-push-type "` untuk APN	Anda dapat menggunakan kolom header `apns-push-type` untuk pemberitahuan mobile dikirim melalui APN.	10 September 2019
Support untuk pemecahan masalah topik menggunakan AWS X-Ray	Anda dapat menggunakan X-Ray untuk memecahkan masalah pesan yang melewati topik SNS.	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 rilis Amazon SNS tersedia sebagai umpan RSS	Mengikuti judul di halaman ini (Riwayat dokumentasi), pilih RSS.	22 Juni 2019

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.