Entwicklerhandbuch

Amazon Simple Notification Service

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Notification Service: Entwicklerhandbuch

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon SNS? ... 1
- Funktionsweise ... 1
- Auf Amazon SNS zugreifen ... 2
- Amazon SNS-Szenarien ... 3
  - Anwendungsintegration ... 3
  - Warnungen für die Anwendung ... 4
  - Benutzerbenachrichtigungen ... 5
  - Mobile Push-Benachrichtigungen ... 5
- Preismodell für Amazon SNS ... 5
- Funktionen und Fähigkeiten von Amazon SNS ... 5
- Häufig gemeinsam genutzte Dienste ... 7
- Arbeitet mit AWS SDKs ... 11

Erstellen Sie ein Thema und veröffentlichen Sie Nachrichten ... 13
- Einrichtung ... 13
  - Ein Konto und einen IAM-Benutzer erstellen ... 13
  - Nächste Schritte ... 15
- Schritt 1: Ein Thema erstellen ... 16
  - AWS Management Console ... 16
  - AWS SDKs ... 20
- Schritt 2: Erstellen eines Abonnements für ein Thema ... 35
  - So richten Sie ein Endpunkt-Abonnement für ein Amazon-SNS-Thema ein ... 35
- Schritt 3: Eine Nachricht veröffentlichen ... 37
  - AWS Management Console ... 37
  - AWS SDKs ... 39
  - Payloads großer Nachrichten ... 63
  - Nachrichtenattribute ... 75
  - Batch-Messaging ... 80
- Schritt 4: Abonnement und Thema löschen ... 83
  - AWS Management Console ... 84
  - AWS SDKs ... 85
- Nächste Schritte ... 95

Reihenfolge und Deduplizierung von Nachrichten mithilfe von FIFO-Themen ... 96
- FIFO-Themen mit hohem Durchsatz ... 96
  - Anwendungsfälle ... 97

Partitionen und Datenverteilung ... 97
Partitionen und Datenverteilung ... 98
Aktivieren Sie einen hohen Durchsatz ... 98
Aktivieren Sie den Modus mit hohem Durchsatz für Amazon SQS SQS-Warteschlangen ... 99
Anwendungsfall zum FIFO-Thema ... 99
Nachrichtenreihenfolge ... 101
Gruppierung von Nachrichten ... 105
Verteilung der Daten nach Nachrichtengruppen IDs zur Verbesserung der Leistung ... 106
Nachrichtenzustellung ... 107
Nachrichtenfilterung ... 108
Nachrichtendeduplizierung ... 110
Nachrichten-Sicherheit ... 112
Speicherdauer der Nachrichten ... 113
Nachrichtenarchivierung und -wiederholung ... 116
Was ist die Nachrichtenarchivierung und -wiederholung? ... 116
Für Themenbesitzer ... 117
Für Subscriber des Themas ... 123
Codebeispiele ... 127
FIFO-Beispiel (AWS SDKs) ... 127
FIFO-Beispiel (AWS CloudFormation) ... 141
Nachrichtenfilterung ... 145
Filterrichtlinien für Abonnements – Geltungsbereich ... 145
Abonnement-Filterrichtlinien ... 146
Beispielfilterrichtlinien für Amazon SNS ... 147
Einschränkungen für Filterrichtlinien ... 150
UND/ODER-Logik ... 154
Schlüsselabgleich ... 158
Abgleichen numerischer Werten ... 161
Abgleich von Zeichenfolgewerten ... 163
Anwenden von Abonnementfilterrichtlinien ... 171
AWS Management Console ... 172
AWS CLI ... 173
AWS SDKs ... 174
Amazon SNS-API ... 179
AWS CloudFormation ... 179
Entfernen von Abonnementfilterrichtlinien ... 180

Mit dem AWS Management Console ... 180
Verwenden Sie den AWS CLI ... 180
Verwenden der Amazon SNS SNS-API ... 180
Nachrichtendatenschutz ... 181
Was ist Nachrichtendatenschutz ... 181
Weshalb sollte ich den Nachrichtendatenschutz verwenden ... 182
Datenschutzrichtlinien ... 182
Was sind Datenschutzrichtlinien? ... 182
Übersicht über den Aufbau der Datenschutzrichtlinie ... 183
Wie ermittle ich die IAM-Prinzipale? ... 186
Operationen der Datenschutzrichtlinien ... 186
Beispiele für Datenschutzrichtlinien ... 195
Erstellen von Datenschutzrichtlinien ... 202
Löschen von Datenschutzrichtlinien ... 212
Datenkennungen ... 213
Verwaltete Datenkennungen ... 213
Benutzerdefinierte Datenbezeichner ... 253
Nachrichtenzustellung ... 256
Übermittlung unformatierter Nachrichten ... 256
Aktivieren der Übermittlung unformatierter Nachrichten mit der AWS Management Console ... 257
Nachrichtenformat – Beispiele ... 257
Nachrichtenattribute und Zustellung von Rohnachrichten für Amazon SQS SQS-Abonnements ... 258
Kontoübergreifende Bereitstellung ... 258
Abonnement erstellt durch Warteschlangeneigentümer ... 259
Ein Benutzer, der nicht der Warteschlangeneigentümer ist, erstellt ein Abonnement ... 261
Wie erzwinge ich, dass ein Abonnement eine Authentifizierung bei Abmeldeanfragen erfordert? ... 264
Regionsübergreifende Lieferung ... 264
Opt-In-Regionen ... 265
Status von Nachrichtenübermittlungen ... 268
Voraussetzungen für die Protokollierung des Lieferstatus ... 269
Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS Management Console ... 270
Konfiguration der Protokollierung des Lieferstatus mithilfe der AWS SDKs ... 271

AWS SDK-Beispiele zur Konfiguration von Themenattributen ... 273
Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS CloudFormation . 282
Wiederholungsversuche bei der Nachrichtenzustellung ... 283
Zustellungsprotokolle und -richtlinien ... 284
Stufen von Zustellungsrichtlinien ... 285
Erstellung HTTP/S einer Lieferrichtlinie ... 286
Warteschlangen für unzustellbare Nachrichten ... 293
Warum schlagen Nachrichtenzustellungen fehl? ... 294
Funktionsweise von Queues für unzustellbare Nachrichten ... 295
Wie werden Nachrichten in eine Queue für unzustellbare Nachrichten verschoben? ... 295
Wie kann ich Nachrichten aus einer Queue für unzustellbare Nachrichten verschieben? ... 296
Wie kann ich Queues für unzustellbare Nachrichten überwachen und protokollieren? ... 296
Konfigurieren einer Queue für unzustellbare Nachrichten ... 297
Nachrichtenarchivierung und -analytik ... 302
Ressourcenmanagement und Optimierung ... 303
Markierung ... 303
Markierungen für die Kostenzuordnungen ... 303
Markierungen für die Zugriffssteuerung ... 304
Markierungen für die Suche und Filterung von Ressourcen ... 305
Konfigurieren von Tags ... 306
Amazon SNS Ereignisquellen und -ziele ... 313
Ereignisquellen ... 313
Analysen ... 313
Anwendungsintegration ... 314
Fakturierung und Kostenmanagement ... 315
Unternehmensanwendungen ... 316
Datenverarbeitung ... 316
Container ... 318
Kundenbindung ... 318
Datenbank ... 319
Entwickler-Tools ... 321
Front-End Web & Mobil ... 322
Spieleentwicklung ... 323
Internet of Things ... 323
Machine Learning ... 324
Management & Governance ... 326

Medien ... 328
Migration und Übertragung ... 329
Netzwerk und Bereitstellung von Inhalten ... 330
Sicherheit, Identität und Compliance ... 331
Serverless ... 332
Speicher ... 333
Zusätzliche Ereignisquellen ... 335
Eventziel ... 336
A2A-Ziele ... 336
A2P-Ziele ... 338
Application-to-application Nachrichtenübermittlung ... 341
Lieferstreams von Fanout nach Firehose ... 341
Voraussetzungen ... 342
Abonnieren eines Bereitstellungsdatenstroms für ein Thema ... 344
Verwaltung von Nachrichten an mehreren Lieferstream-Zielen ... 345
Beispiel für einen Anwendungsfall für Nachrichtenarchivierung und -analyse ... 359
Aufrufen von Lambda-Funktionen ... 372
Voraussetzungen ... 372
Abonnieren eines Themas mit einer Lambda-Funktion ... 373
Verteilung an Amazon-SQS-Warteschlangen ... 374
Abonnieren einer Warteschlange für ein Thema ... 375
Automatisieren Sie das Versenden von Amazon SNS zu Amazon SQS mit AWS CloudFormation ... 383
Fanout-Benachrichtigungen an HTTPS-Endpunkte ... 391
Abonnieren eines Themas durch einen Endpunkt ... 393
Überprüfen der Signaturen von Nachrichten ... 402
Analysieren von Nachrichtenformaten ... 408
Fanout-Ereignisse an Event Fork-Pipelines weiterleiten AWS ... 419
So funktionieren AWS Event Fork-Pipelines ... 420
Bereitstellung von AWS Event Fork-Pipelines ... 423
Bereitstellung und Testen der Event-Fork-Pipeline-Beispielanwendung ... 425
Abonnieren einer Ereignispipeline für ein Thema ... 434
Scheduler verwenden EventBridge ... 444
Einrichtung der Ausführungsrolle ... 444
Erstellen eines Zeitplans ... 444
Zugehörige Ressourcen ... 449

Application-to-person Nachrichtenübermittlung ... 450
Mobile Textnachrichten ... 450
Wie stellt Amazon SNS meine SMS-Nachrichten zu? ... 452
Erste Schritte ... 453
Ursprungsidentitäten ... 459
Konfigurationen ... 461
Bewährte Methoden für SMS-Nachrichten ... 541
Senden mobiler Push-Benachrichtigungen ... 557
So funktionieren Amazon SNS SNS-Benutzerbenachrichtigungen ... 558
Push-Benachrichtigungen mit Amazon SNS einrichten ... 559
Einrichten einer mobilen App ... 560
Amazon SNS für mobile Push-Benachrichtigungen verwenden ... 580
Mobile App-Attribute ... 595
Mobile App-Ereignisse ... 599
Mobile Push-API-Aktionen ... 602
Häufige Fehler bei der mobilen Push-API ... 604
Mobil-Push-TTL ... 617
Unterstützte Regionen ... 620
Bewährte Methoden für mobile Push-Benachrichtigungen ... 621
Einrichtung und Verwaltung von E-Mail-Abonnements ... 622
AWS Management Console ... 622
AWS SDKs ... 624
Codebeispiele ... 656
Grundlagen ... 669
Hello Amazon SNS ... 670
Aktionen ... 682
Szenarien ... 856
Erstellen Sie eine App zum Senden von Daten an eine DynamoDB-Tabelle ... 857
Erstellen einer Amazon-SNS-Anwendung ... 858
Erstellen eines Plattformendpunkts für Push-Benachrichtigungen ... 860
Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos ... 863
Erstellen Sie eine Amazon-Textract-Explorer-Anwendung ... 867
Erstellen und veröffentlichen zu einem FIFO-Thema ... 869
Erkennen Sie Personen und Objekte in einem Video ... 881
Veröffentlichen einer SMS-Nachricht zu einem Thema ... 882
Veröffentlichen einer großen Nachricht ... 889

Veröffentlichen einer SMS-Nachricht ... 892
Veröffentlichen Sie Nachrichten in Warteschlangen ... 901
Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion ... 1016
Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion ... 1018
Serverless-Beispiele ... 1020
Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen ... 1020
Sicherheit ... 1031
Datenverschlüsselung ... 1032
Sicherung von Daten mit serverseitiger Verschlüsselung ... 1032
Schlüsselverwaltung ... 1035
Themenverschlüsselung mit serverseitiger Verschlüsselung einrichten ... 1043
Themenverschlüsselung mit verschlüsseltem Amazon SQS SQS-Warteschlangenabonnement einrichten ... 1045
Sicherung des Datenverkehrs mit VPC-Endpunkten ... 1051
Erstellung eines VPC-Endpunkts ... 1052
Erstellen einer VPC Richtlinie ... 1054
Veröffentlichen einer Nachricht von einer VPC ... 1055
Verwenden Sie Dual-Stack-Endpunkte für die Konnektivität ... 1069
Sicherheit des Nachrichtendatenschutzes ... 1069
Identity and Access Management ... 1069
Zielgruppe ... 1070
Authentifizierung mit Identitäten ... 1071
Verwalten des Zugriffs mit Richtlinien ... 1075
Zugriffskontrolle ... 1078
Anwendungsfälle für die Zugriffskontrolle ... 1078
Die wichtigsten Konzepte der Zugriffsrichtlinie ... 1078
Übersicht über die Architektur ... 1082
Verwenden der Zugriffskontrollsprache ... 1084
Auswertungslogik ... 1085
Beispiele für die Zugriffskontrolle in Amazon SNS ... 1090
So funktioniert Amazon SNS mit IAM ... 1102
AWS verwaltete Richtlinien ... 1103
Richtlinienaktionen ... 1110
Richtlinienressourcen ... 1111
Bedingungsschlüssel für die Richtlinie ... 1111
ACLs ... 1112

ABAC 1113
Temporäre Anmeldeinformationen 1113
Prinzipalberechtigungen 1114
Servicerollen 1114
Service-verknüpfte Rollen 1115
Beispiele für identitätsbasierte Richtlinien 1115
Identitätsbasierte Richtlinien 1119
Ressourcenbasierte Richtlinien 1120
Verwenden identitätsbasierter Richtlinien 1121
Verwaltung benutzerdefinierter IAM-Richtlinien 1128
Verwenden temporärer Anmeldeinformationen 1129
Referenz für API-Berechtigungen 1131
Protokollierung und Überwachung 1135
CloudTrail protokolliert 1136
Themen überwachen mit CloudWatch 1146
Compliance-Validierung 1165
Ausfallsicherheit 1166
Sicherheit der Infrastruktur 1167
Bewährte Methoden für die Gewährleistung der Sicherheit 1167
Vorbeugende bewährte Methoden 1168
Themen zur Fehlerbehebung mithilfe von AWS X-Ray 1173
Aktives Tracing 1173
Berechtigungen 1174
Aktivieren des aktiven Tracings 1174
Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe des SDK AWS 1175
Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe der CLI AWS 1176
Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mit AWS CloudFormation 1176
Überprüfen, ob das aktive Tracing aktiviert ist 1176
Testen 1177
Verlauf der Amazon SNS SNS-Dokumentation 1179
..... mcxc

Was ist Amazon SNS?

Amazon Simple Notification Service (Amazon SNS) ist ein vollständig verwalteter Service, der die Nachrichtenzustellung von Verlagen (Produzenten) an Abonnenten (Verbraucher) ermöglicht. Herausgeber kommunizieren asynchron mit Abonnenten, indem sie eine Nachricht erstellen und an ein Thema senden, bei dem es sich um einen logischen Zugriffspunkt und Kommunikationskanal handelt.

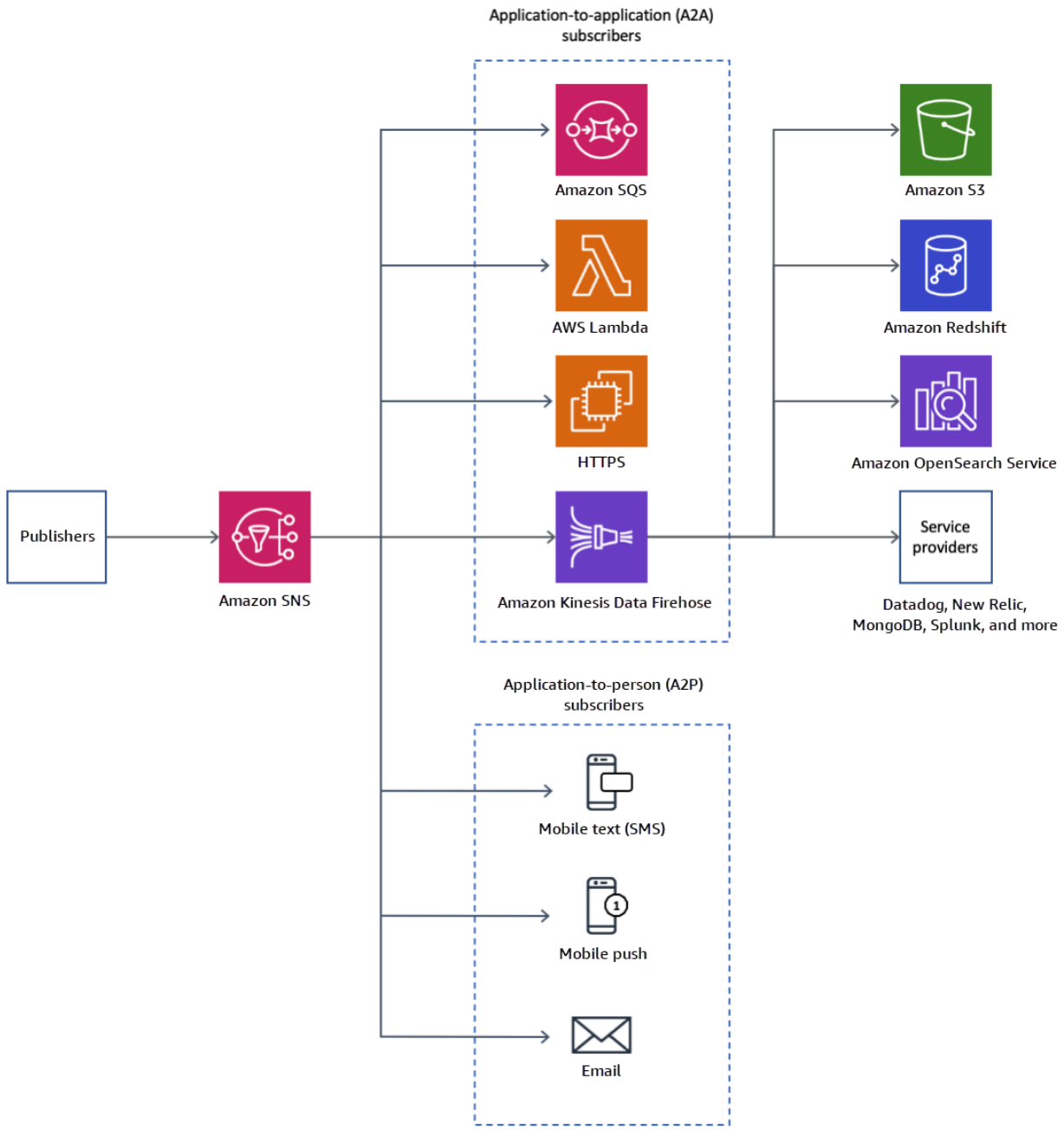
Funktionsweise

In SNS senden Verlage Nachrichten zu einem Thema, das als Kommunikationskanal dient. Das Thema fungiert als logischer Zugangspunkt und stellt sicher, dass Nachrichten an mehrere Abonnenten auf verschiedenen Plattformen zugestellt werden.

Abonnenten eines SNS-Themas können je nach Anwendungsfall Nachrichten über verschiedene Endpunkte empfangen, z. B.:

- Amazon SQS
- Lambda
- HTTP (S) -Endpunkte
- Email
- Mobile Push-Benachrichtigungen
- Mobile Textnachrichten (SMS)
- Amazon Data Firehose
- Dienstanbieter (zum Beispiel Datadog, MongoDB, Splunk)

SNS unterstützt sowohl Application-to-Application (A2A) als auch Application-to-Person (A2P) Messaging und bietet so die Flexibilität, Nachrichten zwischen verschiedenen Anwendungen oder direkt an Mobiltelefone, E-Mail-Adressen und mehr zu senden.



Auf Amazon SNS zugreifen

Sie können über die Konsole oder AWS SDKs, je nach Ihrer bevorzugten Interaktionsmethode AWS CLI, auf Amazon SNS zugreifen und es verwalten. Die Konsole bietet eine grafische

Oberfläche für grundlegende Aufgaben, während sie erweiterte Konfigurations AWS CLI - und Automatisierungsfunktionen für komplexere Anwendungsfälle SDKs bietet.

- DieAmazon-SNS-Konsolebietet eine komfortable Benutzeroberfläche zum Erstellen von Themen und Abonnements, zum Senden und Empfangen von Nachrichten sowie zur Überwachung von Ereignissen und Protokollen.
- Mit AWS Command Line Interface (AWS CLI) erhalten Sie direkten Zugriff auf die Amazon SNS SNS-API für erweiterte Konfiguration und Automatisierung. Weitere Informationen finden Sie unter Verwenden von Lambda mit Amazon SNS AWS CLI.
- AWS bietet SDKs in verschiedenen Sprachen. Weitere Informationen finden Sie unter SDKs und Toolkits.

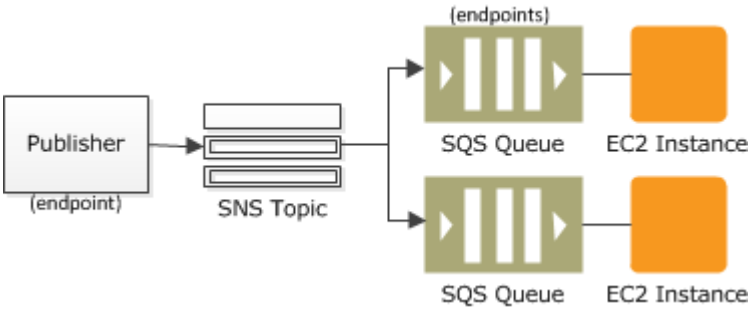
Amazon SNS-Szenarien

Verwenden Sie diese gängigen Amazon SNS SNS-Szenarien, um skalierbare, ereignisgesteuerte Architekturen zu implementieren und eine zuverlässige Echtzeitkommunikation zwischen Anwendungen und Benutzern sicherzustellen.

Anwendungsintegration

Beim Fanout-Szenario wird eine zu einem SNS-Thema veröffentlichte Nachricht repliziert und an mehrere Endpunkte wie Firehose-Zustellungsstreams, Amazon SQS SQS-Warteschlangen, HTTP (S) -Endpunkte und Lambda-Funktionen weitergeleitet. Auf diese Weise kann eine parallele, asynchrone Verarbeitung erfolgen.

Sie können z. B. eine Anwendung entwickeln, die immer dann eine Nachricht an ein SNS-Thema sendet, wenn eine Bestellung für ein Produkt eingeht. Auf diese Weise erhalten alle SQS-Queues, die dieses SNS-Thema abonniert haben, identische Benachrichtigungen für die neue Bestellung. Eine Amazon Elastic Compute Cloud (Amazon EC2) -Serverinstanz, die an eine der SQS-Warteschlangen angehängt ist, kann die Bearbeitung oder Erfüllung der Bestellung übernehmen. Und Sie können eine weitere EC2 Amazon-Serverinstanz an ein Data Warehouse anhängen, um alle eingegangenen Bestellungen zu analysieren.



Eine weitere Möglichkeit zur Verwendung von des Verteilens liegt in der Replikation von Daten, die an Ihre Produktionsumgebung gesendet wurden, um sie mit Ihrer Testumgebung zu teilen. Um beim vorherigen Beispiel zu bleiben könnten Sie demselben SNS-Thema noch eine weitere SQS-Queue für neue eingehende Bestellungen hinzufügen. Wenn Sie diese neue SQS-Queue nun Ihrer Testumgebung anfügen, können Sie Ihre Anwendung mithilfe von Daten aus der Produktionsumgebung weiter verbessern und testen.

> **Important**
>
> Stellen Sie sicher, dass Sie Datenschutz und Sicherheit berücksichtigen, bevor Sie Produktionsdaten an Ihre Testumgebung senden.

Weitere Informationen finden Sie in den folgenden Ressourcen:

- Lieferstreams von Fanout nach Firehose
- Fanout Amazon SNS SNS-Benachrichtigungen an Lambda-Funktionen zur automatisierten Verarbeitung
- Fanout von Amazon SNS SNS-Benachrichtigungen an Amazon SQS SQS-Warteschlangen zur asynchronen Verarbeitung
- Fanout Amazon SNS SNS-Benachrichtigungen an HTTPS-Endpunkte
- Event-Driven Computing mit Amazon SNS und AWS Rechen-, Speicher-, Datenbank- und Netzwerkservices

Warnungen für die Anwendung

Bei Anwendungs- und Systemwarnungen handelt es sich um Benachrichtigungen, die anhand vordefinierter Schwellenwerte ausgelöst und per SMS und/oder E-Mail an angegebene Benutzer gesendet werden. Amazon SNS kann diese Benachrichtigungen per SMS und E-Mail an bestimmte

Benutzer senden. Sie können beispielsweise sofort benachrichtigt werden, wenn ein Ereignis eintritt, z. B. eine bestimmte Änderung an Ihrer Amazon EC2 Auto Scaling Scaling-Gruppe, eine neue Datei, die in einen Amazon S3 S3-Bucket hochgeladen wurde, oder wenn ein metrischer Schwellenwert in Amazon CloudWatch überschritten wurde. Weitere Informationen finden Sie unter Amazon SNS SNS-Benachrichtigungen einrichten im CloudWatch Amazon-Benutzerhandbuch.

Benutzerbenachrichtigungen

Amazon SNS kann Push-E-Mails und Textnachrichten (SMS-Nachrichten) an Einzelpersonen oder Gruppen senden. Beispielsweise könnten Sie E-Commerce-Auftragsbestätigungen als Benutzerbenachrichtigungen senden. Weitere Informationen finden Sie unter Senden von Amazon-SNS-Nachrichten in Mobiles Textnachrichten mit Amazon SNS.

Mobile Push-Benachrichtigungen

Mobile Push-Benachrichtigungen ermöglichen es Ihnen, Nachrichten direkt an mobile Apps zu senden. Sie können Amazon SNS beispielsweise verwenden, um Aktualisierungsbenachrichtigungen an eine App zu senden. Die Benachrichtigung kann einen Link zum Herunterladen und Installieren der Aktualisierung enthalten. Weitere Informationen zur Verwendung von Amazon SNS zum Senden von Push-Benachrichtigungen finden Sie unter.Senden von mobilen Push-Benachrichtigungen mit Amazon SNSaus.

Preismodell für Amazon SNS

Amazon SNS hat keine Vorabkosten. Sie zahlen basierend auf der Anzahl der Nachrichten, die Sie veröffentlichen, der Anzahl der Benachrichtigungen, die Sie bereitstellen, und allen zusätzlichen API-Aufrufen für die Verwaltung von Themen und Abonnements. Die Preise für die Lieferung variieren je nach Endpunkttyp. Mit dem kostenlosen Amazon-SNS-Kontingent können Sie kostenlos loslegen. Weitere Informationen finden Sie unter Weltweite SMS-Preise.

Funktionen und Fähigkeiten von Amazon SNS

Amazon SNS bietet eine umfassende Reihe von Funktionen zur Verbesserung der Nachrichtenübermittlung zwischen Anwendungen und Benutzern. Diese Funktionen ermöglichen eine reibungslose Kommunikation, eine sichere Nachrichtenzustellung und ein robustes Nachrichtenmanagement und gewährleisten so eine hohe Verfügbarkeit, Beständigkeit und Flexibilität für eine Vielzahl von Messaging-Anwendungsfällen.

Application-to-application Nachrichtenübermittlung

Ein pplication-to-application Messaging unterstützt Abonnenten wie Lieferdatenströme, Lambda-Funktionen, Amazon SQS SQS-Warteschlangen, HTTP/S Endpunkte und AWS Event Fork-Pipelines. Dies ermöglicht eine effiziente Nachrichtenzustellung in ereignisgesteuerten Architekturen.

Application-to-person notifications

pplication-to-personA-Benachrichtigungen stellen Benutzerbenachrichtigungen für Abonnenten bereit, z. B. über mobile Anwendungen, Handynummern und E-Mail-Adressen.

Standard- und FIFO-Themen

FIFO-Themen sorgen für eine strikte Nachrichtenreihenfolge, Gruppierung und Deduplizierung, sodass FIFO- und Standardwarteschlangen die Nachrichtenverarbeitung abonnieren können. Standardthemen werden verwendet, wenn die Reihenfolge der Nachrichten und mögliche Duplizierung nicht entscheidend sind. Sie unterstützen alle Übermittlungsprotokolle für umfassendere Anwendungsfälle.

Speicherdauer der Nachrichten

Amazon SNS verwendet eine Reihe von Strategien, die zusammenarbeiten, um die Haltbarkeit von Nachrichten zu gewährleisten:

- Veröffentlichte Nachrichten werden auf mehreren geografisch getrennten Servern und Rechenzentren gespeichert.
- Wenn ein abonnierter Endpunkt nicht verfügbar ist, führt Amazon SNS eineRichtlinie zur Wiederholung von Zustellungenaus.
- Um Nachrichten beizubehalten, die nicht zugestellt werden, bevor die Richtlinie für die Wiederholung der Zustellung endet, können Sie eineQueue unzustellbare Nachrichteneinrichten.

Archivierung, Wiederholung und Analyse von Nachrichten

Sie können Nachrichten auf verschiedene Arten mit Amazon SNS archivieren, z. B. indem Sie Firehose-Lieferstreams für SNS-Themen abonnieren, wodurch Sie Benachrichtigungen an Analyseendpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen und mehr senden können. Darüber hinaus unterstützen Amazon SNS FIFO-Themen die Archivierung und Wiederholung von Nachrichten als integriertes Nachrichtenarchiv ohne Code, mit dem Themenbesitzer Nachrichten innerhalb ihres Themas speichern (oder archivieren) können. Themen-Suscriber können archivierten Nachrichten zurück an einen

abonnierten Endpunkt abrufen (oder wiederholen). Weitere Informationen finden Sie unter Archivierung und Wiedergabe von Amazon SNS SNS-Nachrichten für FIFO-Themen.

Nachrichtenattribute

Amazon-SNS-Nachrichtenattributeermöglicht es Ihnen, beliebige Metadaten zur Nachricht bereitzustellen.

Nachrichtenfilterung

Standardmäßig erhalten Abonnementen von Themen jede Nachricht, die zum Thema veröffentlicht wird. Um nur eine Teilmenge der Nachrichten zu erhalten, müssen Abonnenten dem Themen-Abonnement eine Filterrichtlinie zuweisen. Ein Abonnent kann auch den Geltungsbereich der Filterrichtlinie definieren, um eine nutzlastbasierte oder attributbasierte Filterung zu aktivieren. Der Standardwert für den Geltungsbereich der Filterrichtlinie ist `MessageAttributes`. Wenn die Attribute für eingehende Nachrichten mit den Filterrichtlinienattributen übereinstimmen, wird die Nachricht an den abonnierten Endpunkt übermittelt. Andernfalls wird die Nachricht herausgefiltert. Wenn der Geltungsbereich der Filterrichtlinie `MessageBody` ist, werden die Attribute der Filterrichtlinie mit der Nutzlast abgeglichen. Weitere Informationen finden Sie unter Nachrichtenfilterung.

Nachrichten-Sicherheit

Serverseitige Verschlüsselung schützt den Inhalt von Nachrichten, die in Amazon SNS SNS-Themen gespeichert sind, mithilfe von Verschlüsselungsschlüsseln, die von bereitgestellt werden. AWS KMS Weitere Informationen finden the section called “Sicherung von Daten mit serverseitiger Verschlüsselung” Sie unter Sie können auch eine private Verbindung zwischen Amazon SNS und Ihrer Virtual Private Cloud (VPC) herstellen. Weitere Informationen finden Sie unter. the section called “Sicherung des Datenverkehrs mit VPC-Endpunkten”

AWS Dienste, die häufig mit Amazon SNS verwendet werden

Integrieren Sie Amazon SNS mit mehreren AWS Cloud Services, um die Nachrichtenverarbeitung zu verbessern, die Zugriffskontrolle zu verbessern, eine ereignisgesteuerte Verarbeitung zu ermöglichen und Ressourcen zu automatisieren. Diese Integration optimiert die Leistung, erhöht die Sicherheit und rationalisiert den Betrieb.

Amazon CloudWatch

Amazon CloudWatch bietet Überwachung und Beobachtbarkeit für Amazon SNS und hilft Ihnen dabei, die Nachrichtenzustellung zu verfolgen, Anomalien zu erkennen und Probleme zu beheben. Mit CloudWatch haben Sie folgende Möglichkeiten:

- Überwachen Sie Amazon SNS SNS-Metriken wie die Anzahl der veröffentlichten, zugestellten oder fehlgeschlagenen Nachrichten für verschiedene Themen und Abonnements.
- Richten Sie CloudWatch Alarme ein, um automatisierte Aktionen auszulösen, wenn Amazon SNS SNS-Metriken vordefinierte Schwellenwerte überschreiten, wie z. B. hohe Lieferausfälle oder Drosselung.
- Verwenden Sie CloudWatch Logs, um den Amazon SNS SNS-Lieferstatus für Nachrichten zu erfassen, die an HTTP/S-, Lambda- und Amazon SQS SQS-Endpunkte zum Debuggen und Prüfen gesendet werden.

Weitere Informationen finden Sie unter Überwachung von Amazon SNS SNS-Themen mit CloudWatch.

Amazon SQS

Amazon SQS ist ein vollständig verwalteter Message Queuing-Service, der eine sichere, dauerhafte und skalierbare Kommunikation zwischen verteilten Softwarekomponenten ermöglicht. Er hilft dabei, die Anwendungsarchitektur zu entkoppeln, indem er Nachrichten zwischenspeichert, eine zuverlässige Zustellung sicherstellt und Systemausfälle aufgrund von Nachrichtenverlusten verhindert. Amazon SQS lässt sich auf folgende Weise in Amazon SNS integrieren:

- Warteschlangen für unzustellbare Briefe — Amazon SNS kann unzustellbare Nachrichten zur Fehlerbehebung und erneuten Verarbeitung an eine Amazon SQS SQS-Warteschlange für unzustellbare Briefe weiterleiten.
- Themenabonnements — Sie können eine Amazon SQS-Warteschlange für ein Amazon SNS-Thema abonnieren, sodass Amazon SNS mithilfe von Amazon SQS Nachrichten an mehrere Verbraucher verteilen kann.
- Unterstützung für FIFO-Warteschlangen — Amazon SQS FIFO-Warteschlangen können für Amazon SNS FIFO-Themen abonniert werden, wodurch eine strikte Nachrichtenreihenfolge und eine exakte Einmalverarbeitung gewährleistet werden. Standardmäßige Amazon SQS SQS-Warteschlangen können auch Amazon SNS SNS-Themen abonnieren, garantieren jedoch nicht die Zustellung bestellter Nachrichten oder die Deduplizierung.

AWS CloudFormation

AWS CloudFormation automatisiert die Bereitstellung und Verwaltung von AWS Ressourcen, einschließlich Amazon SNS SNS-Themen und -Abonnements, mithilfe von Infrastructure as Code (IaC). Mit AWS CloudFormation können Sie:

- Definieren Sie Amazon SNS SNS-Themen, Abonnements und Berechtigungen in einer wiederverwendbaren, versionskontrollierten Vorlage.
- Stellen Sie die konsistente Bereitstellung von Amazon SNS SNS-Ressourcen in mehreren AWS-Konten Regionen sicher.
- Aktualisieren oder ändern Sie Amazon SNS SNS-Konfigurationen mithilfe von Änderungssätzen ohne manuelles Eingreifen.

Weitere Informationen finden Sie im AWS CloudFormation -Benutzerhandbuch.

AWS CloudTrail

CloudTrail bietet Einblick in die API-Aktivitäten für Amazon SNS und hilft Ihnen dabei, den Zugriff auf Amazon SNS SNS-Themen, -Abonnements und -Nachrichten zu überwachen und zu prüfen. Mit können CloudTrail Sie:

- Verfolgen Sie API-Aufrufe an Amazon SNS, einschließlich der Personen, die auf Themen, Abonnements und Berechtigungen zugegriffen oder diese geändert haben.
- Erkennen Sie unbefugte oder unerwartete Aktivitäten, indem Sie Protokolle aus Sicherheits- und Compliance-Gründen analysieren.
- Integrieren Sie es mit Amazon CloudWatch oder AWS Security Hub erstellen Sie Benachrichtigungen auf der Grundlage ungewöhnlicher Amazon SNS SNS-Aktionen.

Weitere Informationen hierzu finden Sie unter Protokollieren AWS von SNS-API-Aufrufen mit AWS CloudTrail.

AWS Lambda

AWS Lambda ist ein serverloser Rechenservice, der Ihren Code als Reaktion auf Ereignisse automatisch ausführt, sodass keine Server bereitgestellt oder verwaltet werden müssen. Damit können Sie ereignisgesteuerte Anwendungen erstellen, die automatisch skalieren und in einer hochverfügbaren Rechenumgebung ausgeführt werden.

Amazon SNS lässt sich in Lambda integrieren, indem es Ihnen ermöglicht, eine Lambda-Funktion für ein Amazon SNS SNS-Thema zu abonnieren. Wenn ein Amazon SNS SNS-Thema eine Nachricht empfängt, kann es die Lambda-Funktion auslösen, die Verarbeitung, Automatisierung

und Ausführung der Anwendungslogik in Echtzeit ermöglicht. Diese Integration wird häufig verwendet für:

- Ereignisgesteuerte Verarbeitung — Automatisches Auslösen von Funktionen als Reaktion auf Amazon SNS SNS-Nachrichten.
- Datentransformation — Ändern oder filtern Sie Amazon SNS SNS-Nachrichten, bevor Sie sie an andere Dienste weiterleiten.
- Automatisierte Workflows — Verarbeiten Sie Benachrichtigungen für Anwendungswarnungen, Systemüberwachung oder Event-Orchestrierung.

AWS Identity and Access Management (IAM)

IAM bietet eine sichere Zugriffskontrolle für AWS Ressourcen, sodass Sie verwalten können, wer auf Ihre Amazon SNS SNS-Themen zugreifen kann, welche Aktionen sie ausführen können und unter welchen Bedingungen. Mit IAM können Sie:

- Authentifizieren Sie Benutzer und Dienste, bevor sie mit Amazon SNS SNS-Themen interagieren können.
- Definieren Sie detaillierte Berechtigungen, um festzulegen, für welche Amazon SNS SNS-Themen Benutzer oder Rollen Beiträge veröffentlichen, abonnieren oder verwalten können.
- Verwenden Sie identitätsbasierte Richtlinien, um bewährte Sicherheitsmethoden durchzusetzen, z. B. die Beschränkung des Zugriffs auf bestimmte AWS-Konten IP-Adressen oder Bedingungen.

Weitere Informationen finden Sie unter Verwenden von identitätsbasierten Richtlinien mit Amazon SNS.

AWS Key Management Service (AWS KMS)

AWS KMS verbessert die Sicherheit von Amazon SNS durch die Aktivierung der serverseitigen Verschlüsselung (SSE) für die Vertraulichkeit von Nachrichten. Mit können AWS KMS Sie:

- Verschlüsseln Sie Amazon SNS SNS-Nachrichten im Ruhezustand mit AWS-verwalteten oder vom Kunden verwalteten Verschlüsselungsschlüsseln (). CMKs
- Steuern Sie den Zugriff auf Amazon SNS SNS-Themen, indem Sie detaillierte Richtlinien definieren, die einschränken, wer veröffentlichen oder abonnieren kann.
- Stellen Sie sicher, dass die Sicherheits- und behördlichen Anforderungen eingehalten werden, indem Sie die Verwendung von Schlüsseln bis zum Ende überprüfen. AWS CloudTrail

Weitere Informationen finden Sie unter Verwaltung von Amazon SNS SNS-Verschlüsselungsschlüsseln und Kosten.

AWS X-Ray

X-Ray bietet Tracing für Amazon SNS und hilft Ihnen dabei, den Nachrichtenfluss in Ihrer ereignisgesteuerten Architektur zu analysieren und zu debuggen. Mit X-Ray können Sie:

- Verfolgen Sie die Amazon SNS SNS-Nachrichtenzustellung über mehrere AWS-Services Endpunkte, z. B. Lambda-, Amazon SQS- und HTTP/S-Endpunkte.
- Identifizieren Sie Latenzengpässe, indem Sie visualisieren, wie lange es dauert, Nachrichten zu veröffentlichen, zuzustellen und zu verarbeiten.
- Erkennen Sie Fehler und Wiederholungsversuche im Amazon SNS SNS-Nachrichtenfluss, um fehlgeschlagene Lieferungen oder langsame Verarbeitungszeiten zu beheben.

Weitere Informationen finden Sie unter Aktives Tracing in Amazon SNS.

Amazon SNS mit einem AWS SDK verwenden

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK für Go	AWS SDK für Go Codebeispiele
AWS SDK für Java	AWS SDK für Java Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Codebeispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Codebeispiele
AWS SDK für .NET	AWS SDK für .NET Codebeispiele
AWS SDK für PHP	AWS SDK für PHP Codebeispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Codebeispiele

SDK-Dokumentation	Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Codebeispiele
AWS SDK für Ruby	AWS SDK für Ruby Codebeispiele
AWS SDK für Rust	AWS SDK für Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK für Swift	AWS SDK für Swift Codebeispiele

Weitere Beispiele für Amazon SNS finden Sie unter Codebeispiele für Amazon SNS mit AWS SDKs.

> **Beispiel Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Provide feedback (Feedback geben) auswählen.

Erstellen Sie ein Amazon SNS SNS-Thema und veröffentlichen Sie Nachrichten

Dieses Thema enthält die grundlegenden Schritte zur Verwaltung von Amazon SNS SNS-Ressourcen, wobei der Schwerpunkt insbesondere auf Themen, Abonnements und Nachrichtenveröffentlichung liegt. Zunächst richten Sie die erforderlichen Zugriffsberechtigungen für Amazon SNS ein und stellen sicher, dass Sie über die richtigen Berechtigungen zum Erstellen und Verwalten von Amazon SNS SNS-Ressourcen verfügen. Als Nächstes erstellen Sie ein neues Amazon SNS SNS-Thema, das als zentrale Anlaufstelle für die Verwaltung und Zustellung von Nachrichten an Abonnenten dient. Nachdem Sie das Thema erstellt haben, erstellen Sie ein Abonnement für dieses Thema, sodass bestimmte Endgeräte die darin veröffentlichten Nachrichten empfangen können.

Sobald das Thema und das Abonnement eingerichtet sind, veröffentlichen Sie eine Nachricht zum Thema und beobachten dabei, wie Amazon SNS die Nachricht effizient an alle abonnierten Endpunkte übermittelt. Schließlich erfahren Sie, wie Sie sowohl das Abonnement als auch das Thema löschen und so den Lebenszyklus der von Ihnen verwalteten Amazon SNS SNS-Ressourcen abschließen. Dieser Ansatz vermittelt Ihnen ein klares Verständnis der grundlegenden Abläufe in Amazon SNS und vermittelt Ihnen die praktischen Fähigkeiten, die Sie benötigen, um Messaging-Workflows mithilfe der Amazon SNS SNS-Konsole zu verwalten.

Einrichten des Zugriffs für Amazon SNS

Bevor Sie Amazon SNS zum ersten Mal verwenden können, müssen Sie die folgenden Schritte abschließen.

Erstellen Sie einen AWS-Konto und einen IAM-Benutzer

Um auf einen AWS Dienst zugreifen zu können, müssen Sie zunächst einen AWS-Kontoerstellen. Sie können Ihren verwenden AWS-Konto , um Ihre Aktivitäts- und Nutzungsberichte einzusehen und Authentifizierung und Zugriff zu verwalten.

Melden Sie sich an für eine AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Ein Teil des Anmeldevorgangs umfasst den Empfang eines Telefonanrufs oder einer Textnachricht und die Eingabe eines Bestätigungscodes auf der Telefontastatur.

   Wenn Sie sich für einen anmelden AWS-Konto, wird ein AWS-Konto Root-Benutzer erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können Ihre aktuellen Kontoaktivitäten jederzeit einsehen und Ihr Konto verwalten, indem Sie zu https://aws.amazon.com/gehen und Mein Konto auswählen.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren AWS-Konto Root-Benutzer AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Schützen Sie Ihren AWS-Konto Root-Benutzer

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center Benutzerhandbuch.

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center Benutzerhandbuch.

Nächste Schritte

Nun, da Sie bereit sind, mit Amazon SNS zu arbeiten, beginnen Sie mit:

1. Erstellen Sie ein Amazon SNS-Thema
2. Ein Abonnement für ein Amazon SNS SNS-Thema erstellen
3. Veröffentlichen einer Amazon SNS SNS-Nachricht
4. Löschen eines Amazon SNS SNS-Themas und -Abonnements

Erstellen Sie ein Amazon SNS-Thema

Ein Amazon SNS-Thema ist ein logischer Zugriffspunkt, der als ein Kommunikationskanal fungiert. Mit einem Thema können Sie mehrere Endpunkte gruppieren (z. B. AWS Lambda Amazon SQS, HTTP/S oder eine E-Mail-Adresse).

Zum Senden der Nachrichten eines Nachrichtenproduktionssystems (etwa einer e-Commerce-Website), die mit mehreren anderen Services zusammenarbeiten, die dessen Nachrichten benötigen (beispielsweise Checkout- und Erfüllungssysteme) können Sie ein Thema für Ihr Produktionssystem erstellen.

Die erste und häufigste Amazon SNS-Aufgabe ist das Erstellen eines Themas. Auf dieser Seite wird gezeigt AWS Management Console, wie Sie die, und die verwenden können AWS SDK für Java, um ein Thema AWS SDK für .NET zu erstellen.

Während der Erstellung wählen Sie einen Thementyp (Standard oder FIFO) und benennen das Thema. Der Thementyp oder -name kann nach dem Erstellen eines Themas nicht geändert werden. Alle anderen Konfigurationsoptionen sind während der Themenerstellung optional und Sie können sie später bearbeiten.

> **Important**
>
> Fügen Sie keine persönlich identifizierbare Informationen (PII) oder andere vertrauliche oder sensible Informationen in Themennamen hinzu. Themennamen sind für andere Amazon Web Services zugänglich, einschließlich CloudWatch Logs. Themennamen sind nicht für private oder sensible Daten gedacht.

Um ein Thema mit dem zu erstellen AWS Management Console

Durch die Erstellung eines Themas in Amazon SNS wird die Grundlage für die Nachrichtenverteilung geschaffen, sodass Sie Nachrichten veröffentlichen können, die an mehrere Abonnenten verteilt werden können. Dieser Schritt ist wichtig, um den Typ, die Verschlüsselungseinstellungen und

die Zugriffsrichtlinien des Themas zu konfigurieren und sicherzustellen, dass das Thema den Sicherheits-, Compliance- und Betriebsanforderungen des Unternehmens entspricht.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Führen Sie eine der folgenden Aktionen aus:
   - Wenn noch nie zuvor Themen unter Ihrem AWS-Konto Konto erstellt wurden, lesen Sie die Beschreibung von Amazon SNS auf der Startseite.
   - Wenn AWS-Konto zuvor Themen unter Ihrem Konto erstellt wurden, wählen Sie im Navigationsbereich die Option Themen aus.
3. Klicken Sie auf der Seite Themen auf Create New Topic.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:
   a. FürTypWählen Sie einen Thementyp aus (Standard-oderFIFO).
   b. Geben Sie den Namen des neuen Themas ein. Für eineFIFO-ThemaFügen SieFIFOam Ende des Namens hinzu.
   c. (Optional) Geben Sie einen Display name (Anzeigenamen) für Ihr Thema ein.

      > **Important**
      > Wenn Sie einen E-Mail-Endpunkt abonnieren, darf die kombinierte Zeichenanzahl für den Anzeigenamen des Amazon-SNS-Themas und die sendende E-Mail-Adresse (z. B. no-reply@sns.amazonaws.com) 320 UTF-8-Zeichen nicht überschreiten. Sie können ein Verschlüsselungstool eines Drittanbieters verwenden, um die Länge der Absenderadresse zu überprüfen, bevor Sie einen Anzeigenamen für Ihr Amazon-SNS-Thema konfigurieren.

   d. (Optional) Für ein FIFO-Thema können SieDeduplizierung für inhaltsbasierte Nachrichten, um die Standardnachrichtendeduplizierung zu aktivieren. Weitere Informationen finden Sie unter Amazon SNS SNS-Nachrichtendeduplizierung für FIFO-Themen.
5. (Optional) Erweitern Sie den Abschnitt Encryption (Verschlüsselung) und gehen Sie wie folgt vor. Weitere Informationen finden Sie unter Sicherung von Amazon SNS SNS-Daten mit serverseitiger Verschlüsselung.
   a. Wählen Sie Enable encryption (Verschlüsselung aktivieren) aus.

b. Geben Sie den AWS KMS Schlüssel an. Weitere Informationen finden Sie unter Wichtige Begriffe.

Für jeden KMS-Typ werden Description (Beschreibung), Account (Konto) und KMS ARN (KMS-ARN) angezeigt.

> **Important**
>
> Wenn Sie nicht der Besitzer des KMS-Schlüssels sind oder wenn Sie sich mit einem Konto anmelden, das über keine `kms:ListAliases`- und `kms:DescribeKey`-Berechtigungen verfügt, können Sie auf der Amazon-SNS-Konsole keine Informationen über den KMS aufrufen.
> Bitten Sie den Inhaber des KMS, Ihnen diese Berechtigungen zu erteilen. Beispiele und weitere Informationen zu AWS KMS -Berechtigungen finden Sie unter API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen im AWS Key Management Service Benutzerhandbuch.

- Das AWS verwaltete KMS für Amazon SNS (Standard) alias/aws/sns ist standardmäßig ausgewählt.

> **Note**
>
> Beachten Sie Folgendes:
>
> - Wenn Sie das erste Mal verwenden AWS Management Console , um das AWS verwaltete KMS für Amazon SNS für ein Thema anzugeben, AWS KMS wird das AWS verwaltete KMS für Amazon SNS erstellt.
> - Alternativ können Sie, wenn Sie die `Publish` Aktion zum ersten Mal für ein Thema mit aktivierter SSE verwenden, das AWS KMS AWS verwaltete KMS für Amazon SNS erstellen.

- Um ein benutzerdefiniertes KMS von Ihrem AWS Konto aus zu verwenden, wählen Sie das KMS-Schlüsselfeld und dann das benutzerdefinierte KMS aus der Liste aus.

> **Note**
>
> Anweisungen zur Erstellung benutzerdefinierter KMSs Schlüssel finden Sie unter Creating Keys im AWS Key Management Service Developer Guide

- Um einen benutzerdefinierten KMS-ARN von Ihrem AWS Konto oder einem anderen AWS Konto zu verwenden, geben Sie ihn in das Feld KMS-Schlüssel ein.

6. (Optional) Standardmäßig kann nur der Eigentümer des Themas das Thema abonnieren oder Veröffentlichungen dazu vornehmen. Um zusätzliche Zugriffsberechtigungen zu konfigurieren, erweitern Sie den Abschnitt Access policy (Zugriffsrichtlinie). Weitere Informationen erhalten Sie unter Identity and Access Management in Amazon SNS und Beispiele für die Zugriffskontrolle in Amazon SNS.

> **Note**
>
> Wenn Sie ein Thema mit der Konsole erstellen, verwendet die Standardrichtlinie den Bedingungsschlüssel `aws:SourceOwner`. Dieser Schlüssel ist ähnlich wie `aws:SourceAccount`.

7. (Optional) Um zu konfigurieren, wie Amazon SNS fehlgeschlagene Nachrichtenzustellversuche wiederholt, erweitern Sie den Abschnitt Delivery retry policy (HTTP/S) (Richtlinie für die Zustellungswiederholung (HTTP/S)). Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.
8. (Optional) Um zu konfigurieren, wie Amazon SNS die Zustellung von Nachrichten protokolliert CloudWatch, erweitern Sie den Abschnitt Protokollierung des Lieferstatus. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.
9. (Optional) Wenn Sie Metadaten-Tags zum Thema hinzufügen, erweitern Sie den Tag-Abschnitt, geben Sie einen Schlüssel und einen Wert ein (optional), und wählen Sie Tag hinzufügen aus. Weitere Informationen finden Sie unter Markierung von Amazon-SNS-Themen.
10. Wählen Sie Thema erstellen aus.

    Das Thema wird erstellt und die ***MyTopic***Seite wird angezeigt.

    Der Name, der ARN, (optional) der Anzeigename und die AWS Konto-ID des Themenbesitzers werden im Abschnitt Details angezeigt.
11. Kopieren Sie das Thema ARN in die Zwischenablage, zum Beispiel:

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

Um ein Thema mit einem AWS SDK zu erstellen

Um ein AWS SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Die folgenden Code-Beispiele zeigen, wie `CreateTopic` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema mit einem bestimmten Namen.

```
    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;
        }
    }
```

Erstellen Sie ein neues Thema mit einem Namen und spezifischen FIFO- und Deduplizierungsattributen.

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

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein SNS-Thema

Das folgende `create-topic`-Beispiel erstellt ein SNS-Thema namens `my-topic`.

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

Ausgabe:

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

Weitere Informationen finden Sie unter Verwenden der AWS Befehlszeilenschnittstelle mit Amazon SQS und Amazon SNS im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter CreateTopic AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"
```

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

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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 "";
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

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

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie CreateTopicin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(name: name)
        )

        guard let arn = output.topicArn else {
            print("No topic ARN returned by Amazon SNS.")
            return
        }
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Swift.

Ein Abonnement für ein Amazon SNS SNS-Thema erstellen

Zum Abrufen der für ein Thema veröffentlichten Nachrichten müssen Sie einen Endpunkt für das Thema abonnieren. Wenn ein Endpunkt ein Thema abonniert, beginnt der Endpunkt, für das zugehörige Thema veröffentlichte Nachrichten zu empfangen.

> **Note**
>
> HTTP (S) -Endpunkte, E-Mail-Adressen und andere AWS Ressourcen AWS-Konten erfordern eine Bestätigung des Abonnements, bevor sie Nachrichten empfangen können.

So richten Sie ein Endpunkt-Abonnement für ein Amazon-SNS-Thema ein

Das Abonnieren eines Amazon SNS SNS-Themas über einen Endpunkt ermöglicht die Nachrichtenzustellung an den angegebenen Endpunkt. Dadurch wird sichergestellt, dass die richtigen Systeme oder Benutzer Benachrichtigungen erhalten, wenn eine Nachricht zu dem Thema veröffentlicht wird. Dieser Schritt ist wichtig, um das Thema mit den Verbrauchern zu verknüpfen — unabhängig davon, ob es sich um Anwendungen, E-Mail-Empfänger oder andere Dienste handelt — und ermöglicht so eine reibungslose Kommunikation zwischen den Systemen.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Wählen Sie für Topic ARN den Amazon Resource Name (ARN) eines Themas. Dieser Wert ist beispielsweise `arn:aws:sns:us-east-2:123456789012:your_topic` der AWS ARN, der generiert wurde, als Sie das Amazon SNS SNS-Thema erstellt haben.

   b. Für Protocol wählen Sie einen Endpunkttypen. Folgende Endpunkttypen stehen zur Verfügung:

      - HTTP/HTTPS
      - E-Mail/E-Mail-JSON
      -

- Amazon SQS

> **Note**
>
> Um ein SNS FIFO-Thema zu abonnieren, wählen Sie diese Option.

- AWS Lambda
- Plattformanwendungs-Endpunkt
- SMS

c. Für Endpoint, geben Sie den Endpunktwert ein, wie z. B. eine E-Mail-Adresse oder die ARN einer Amazon-SQS-Warteschlange.

d. Nur Firehose-Endpunkte: Geben Sie für den ARN der Abonnementrolle den ARN der IAM-Rolle an, die Sie für das Schreiben in Firehose-Lieferdatenströme erstellt haben. Weitere Informationen finden Sie unter Voraussetzungen für das Abonnieren von Firehose-Lieferstreams für Amazon SNS SNS-Themen.

e. (Optional) Für Firehose, Amazon SQS und HTTP/S Endpoints können Sie auch die Zustellung von Rohnachrichten aktivieren. Weitere Informationen finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

f. (Optional) Um eine Filterrichtlinie zu konfigurieren, erweitern Sie den Abschnitt Abonnement-Filterrichtlinie. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.

g. (Optional) Zur Aktivierung der nutzlastbasierten Filterung konfigurieren Sie `Filter Policy Scope` auf `MessageBody`. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich.

h. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für das Abonnement zu konfigurieren, erweitern Sie den Abschnitt Redrive-Richtlinie (Warteschlange für unzustellbare Nachrichten). Weitere Informationen finden Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe.

i. Wählen Sie Create subscription (Abonnement erstellen) aus.

   Die Konsole erstellt das Abonnement und öffnet die Seite Details des Abonnements.

Veröffentlichen einer Amazon SNS SNS-Nachricht

Nachdem Sie ein Amazon-SNS-Thema erstellt haben und ein Endpunkt dieses abonniert hat, können Sie Nachrichten zu einem Thema veröffentlichen. Wenn eine Nachricht veröffentlicht wird, versucht Amazon SNS, die Nachricht an die abonnierte Endpunkte auszuliefern.

Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Klicken Sie auf der Seite Topics ein Thema aus und dann Publish to topic.

   Die Konsole öffnet die Nachricht im Thema veröffentlichenSeite
4. Gehen Sie im Abschnitt Basic details (Grundlegende Details) wie folgt vor:
   a. (Optional) Geben Sie den Betreff einer Nachricht ein.
   b. Für ein FIFO-Thema geben Sie eine Nachrichtengruppen-IDein. Für FIFO-Themen ist eine Nachrichtengruppen-ID erforderlich. Nachrichten in derselben Nachrichtengruppe werden in der Reihenfolge zugestellt, in der sie veröffentlicht werden.
   c. (Optional) Geben Sie für ein Standardthema eine Nachrichtengruppen-ID ein. Diese ID wird an alle Amazon SQS SQS-Standardabonnements weitergeleitet und nicht für andere Endpunkttypen verwendet oder an diese gesendet.
   d. Geben Sie für ein FIFO-Thema eine Nachrichtendeduplizierungs-IDein. Diese ID ist optional, wenn Sie die Inhaltsbasierte Nachrichtendeduplizierung-Einstellung für das Thema aktiviert haben.
   e. (Optional) Für Mobile Push-Benachrichtigungen geben Sie einen Lebenszeitwert (TTL) in Sekunden ein. Dies ist die Zeit, die ein Push-Benachrichtigungsdienst — wie Apple Push Notification Service (APNs) oder Firebase Cloud Messaging (FCM) — benötigt, um die Nachricht an den Endpunkt zu übermitteln.
5. Führen Sie im Abschnitt Message body (Nachrichtentext) eine der folgenden Aktionen aus:
   a. Wählen Sie Identical payload for all delivery protocols (Identische Nutzlast für alle Bereitstellungsprotokolle) und geben Sie anschließend die Nachricht ein.

b. Wählen Sie Custom payload for each delivery protocol (Benutzerdefinierte Nutzlast für die einzelnen Bereitstellungsprotokolle) und verwenden Sie anschließend ein JSON-Objekt, um die Nachricht zu definieren, die an die einzelnen Protokolle gesendet werden soll.

   Weitere Informationen finden Sie unter Veröffentlichen von Amazon SNS SNS-Benachrichtigungen mit plattformspezifischen Payloads.

6. Fügen Sie im Abschnitt Message attributes (Nachrichtenattribute) die gewünschten Attribute hinzu, die Amazon SNS mit dem Abonnementattribut `FilterPolicy`abgleichen soll, um zu entscheiden, ob der abonnierte Endpunkt an der veröffentlichten Nachricht interessiert ist.

   a. Als Typ wählen Sie einen Attributtyp aus, z. B. String.Array.

   > **Note**
   >
   > Wenn der Attributtyp String.Array (String.Array) ist, schließen Sie das Array in eckige Klammern (`[]`) ein. Innerhalb des Arrays schließen Sie Zeichenfolgenwerte in doppelte Anführungszeichen ein. Für Zahlen oder die Schlüsselwörter `true`, `false`und `null` benötigen Sie keine Anführungszeichen.

   b. Geben Sie einen Attributnamen, wie beispielsweise `customer_interests`an.

   c. Geben Sie einen Attributvalue, wie beispielsweise `["soccer", "rugby", "hockey"]`an.

   Wenn der Attributtyp String (Zeichenfolge), String.Array oder Number (Zahl) ist, wertet Amazon SNS das Nachrichtenattribut anhand der Filterrichtlinie (sofern vorhanden) des betreffenden Abonnements aus, bevor die Nachricht an dieses Abonnement gesendet wird, vorausgesetzt, der Geltungsbereich der Filterrichtlinie ist nicht explizit auf `MessageBody` festgelegt.

   Weitere Informationen finden Sie unter Amazon-SNS-Nachrichtenattribute.

7. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

   Die Nachricht wird in dem Thema veröffentlicht. Die Konsole öffnet die Details-Seite.

So veröffentlichen Sie eine Nachricht für ein Thema mit AWS -SDK

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Die folgenden Code-Beispiele zeigen, wie `Publish` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen einer Nachricht für ein Thema.

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

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

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

Wenden Sie die Benutzerauswahl auf die Veröffentlichungsaktion an.

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

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        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;
        }
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So veröffentlichen Sie eine Nachricht für ein Thema

Das folgende `publish`-Beispiel veröffentlicht die angegebene Nachricht im angegebenen SNS-Thema. Die Nachricht stammt aus einer Textdatei, in der Sie Zeilenumbrüche einfügen können.

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

Inhalt von `message.txt`:

```
Hello World
Second Line
```

Ausgabe:

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

Beispiel 2: So veröffentlichen Sie eine SMS-Nachricht an eine Telefonnummer

Im folgenden `publish`-Beispiel wird Nachricht `Hello world!` an Telefonnummer `+1-555-555-0100` veröffentlicht.

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

Ausgabe:

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

- API-Details finden Sie unter Publish in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

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

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



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
```

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

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

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

    let groupId;
    let deduplicationId;
    let choices;
```

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

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

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

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
```

```
    );

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

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

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

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

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für PHP -API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

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

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$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
```

- Einzelheiten zur API finden Sie unter In AWS -Tools für PowerShell Cmdlet-Referenz veröffentlichen (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

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

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$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
```

- Einzelheiten zur API finden Sie unter In AWS -Tools für PowerShell Cmdlet-Referenz veröffentlichen (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen Sie eine Nachricht mit Attributen, damit ein Abonnement basierend auf Attributen filtern kann.

```
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
```

Veröffentlichen Sie eine Nachricht, die basierend auf dem Protokoll des Abonnenten unterschiedliche Formen annimmt.

```
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
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Details zu API finden Sie unter Veröffentlichen in der Referenz für das AWS -SDK für Rust-API.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.publish(
            input: PublishInput(
                message: message,
                topicArn: arn
            )
        )

        guard let messageId = output.messageId else {
            print("No message ID received from Amazon SNS.")
            return
        }

        print("Published message with ID \(messageId)")
```

- Einzelheiten zur API finden Sie unter Im AWS SDK veröffentlichen für die Swift-API-Referenz.

Veröffentlichen großer Nachrichten mit Amazon SNS und Amazon S3

Wenn Sie große Amazon-SNS-Nachrichten veröffentlichen möchten, können Sie die Amazon SNS Extended Client Library für Java oder die Amazon SNS Extended Client Library für Python verwenden. Diese Bibliothek ist nützlich für Nachrichten, die größer als das aktuelle Maximum von 256 KB sind. Es gilt ein Maximum von 2 GB. Beide Bibliotheken speichern die tatsächliche Nutzlast in einem Amazon-S3-Bucket und veröffentlicht den Verweis des gespeicherten Amazon-S3-Objekts im

Amazon-SNS-Thema. Abonnierte Amazon SQS Queues können die Amazon SQS Extended Client Library for Jav für Java Verwenden Sie, um die Nutzlasten aus Amazon S3 aufzuheben und diese abzurufen. Andere Endpunkte, wie Lambda, können die Payload-Offloading Java Common Library für AWS, um die Nutzlast zu dereferenzieren und abzurufen.

> **Note**
>
> Die Amazon SNS Extended Client Libraries sind sowohl mit Standard- als auch mit FIFO-Themen kompatibel.

Erweiterte Amazon SNS SNS-Clientbibliothek für Java

Voraussetzungen

Im Folgenden sind die Voraussetzungen für die Verwendung der Amazon SNS Extended Client Library für Java aufgeführt:

- Ein AWS SDK. Das Beispiel auf dieser Seite verwendet das AWS Java-SDK. Informationen zur Installation und Einrichtung des SDK finden Sie unter Setup the AWS SDK for Java im AWS SDK für Java Developer Guide.
- Und AWS-Konto mit den richtigen Anmeldeinformationen. Um ein Konto zu erstellen AWS-Konto, navigieren Sie zur AWS Startseite und wählen Sie dann AWS Konto erstellen. Folgen Sie den Anweisungen.

  Informationen zu Anmeldeinformationen finden Sie im AWS SDK für Java Entwicklerhandbuch unter „AWS Anmeldeinformationen und Region für die Entwicklung einrichten“.
- Java 8 oder höher.
- Die erweiterte Amazon SNS Extended Client Library für Java (ist auch verfügbar von Maven) enthalten.

Konfigurieren der Nachrichtenspeicherung

Die Amazon SNS Extended Client-Bibliothek verwendet die Payload Offloading Java Common Library AWS zum Speichern und Abrufen von Nachrichten. Sie können die folgenden Amazon S3 Optionen für die Nachrichtenspeicherung:

- Schwellenwert für benutzerdefinierte Nachrichtengrößen — Nachrichten mit Nutzlasten und Attributen, die diese Größe überschreiten, werden automatisch in Amazon S3 gespeichert.
- **`alwaysThroughS3`**flag — Setzen Sie diesen Wert auf, um `true` zu erzwingen, dass alle Nachrichten-Payloads in Amazon S3 gespeichert werden. Beispiel:

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

- Benutzerdefinierter KMS-Schlüssel — Der Schlüssel, der für die serverseitige Verschlüsselung in Ihrem Amazon S3 S3-Bucket verwendet werden soll.
- Bucket-Name — Der Name des Amazon S3 S3-Buckets zum Speichern von Nachrichtennutzlasten.

Beispiel: Veröffentlichen von Nachrichten in Amazon SNS mit Payload, die in Amazon S3 gespeichert ist

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Beispielthema und eine Queue.
- Die abonnierte Queue kann nun Nachrichten vom Thema empfangen.
- Veröffentlichen Sie eine Testnachricht.

Die Nachrichtennutzlast wird in Amazon S3 gespeichert und der Verweis darauf wird veröffentlicht. Der Amazon SQS Extended Client wird zum Empfangen der Nachricht verwendet.

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Java. Die Nachricht, die Sie senden, verweist auf ein Amazon S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

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

Andere Endpunktprotokolle

Sowohl die Amazon SNS - als auch die Amazon SQS Bibliotheken verwenden die Payload-Offloading Java Common Library für AWS, um Nachrichten-Payloads mit Amazon S3 zu speichern und abzurufen. Jeder Java-aktivierte Endpunkt (z. B. ein HTTPS-Endpunkt, der in Java implementiert ist) kann dieselbe Bibliothek verwenden, um den Nachrichteninhalt aufzuheben.

Endpunkte, für die die Payload Offloading Java Common Library nicht verwendet werden kann, AWS können trotzdem Nachrichten mit in Amazon S3 gespeicherten Payloads veröffentlichen. Im Folgenden wird ein Beispiel für eine Amazon S3-Referenz gezeigt, die im obigen Codebeispiel veröffentlicht wird:

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

Erweiterte Amazon SNS SNS-Clientbibliothek für Python

Voraussetzungen

Im Folgenden sind die Voraussetzungen für die Verwendung der Amazon SNS Extended Client Library für Python aufgeführt:

- Ein AWS SDK. Das Beispiel auf dieser Seite verwendet das AWS Python-SDK Boto3. Informationen zur Installation und Einrichtung des SDK finden Sie in der Dokumentation zum AWS -SDK für Python.
- Und AWS-Konto mit den richtigen Anmeldeinformationen. Um ein Konto zu erstellen AWS-Konto, navigieren Sie zur AWS Startseite und wählen Sie dann AWS Konto erstellen. Folgen Sie den Anweisungen.

  Weitere Informationen zu Anmeldeinformationen finden Sie unter Anmeldeinformationen im Entwicklerhandbuch zu AWS -SDK für Python.
- Python 3.x (oder höher) und Pip.
- Die erweiterte Amazon SNS Extended Client Library für Python (ist auch verfügbar von Maven) enthalten.

Konfigurieren der Nachrichtenspeicherung

Die folgenden Attribute sind auf Boto3 Amazon SNS Client, Topic und PlatformEndpointObjekten zur Konfiguration der Amazon S3 S3-Nachrichtenspeicheroptionen verfügbar.

- **`large_payload_support`**— Der Name des Amazon S3 S3-Buckets, in dem große Nachrichten gespeichert werden.
- **`use_legacy_attribute`**— Falls`True`, dann verwenden alle veröffentlichten Nachrichten das reservierte Nachrichtenattribut Legacy (`SQSLargePayloadSize`) anstelle des aktuellen reservierten Nachrichtenattributs (`ExtendedPayloadSize`).
- **`message_size_threshold`** – der Schwellenwert für das Speichern der Nachricht im großen Nachrichten-Bucket. Kann nicht kleiner oder größer als sein`262144`. `0` Der Standardwert ist `262144`.
- **`always_through_s3`** – bei `True` werden alle Nachrichten in Amazon S3 gespeichert. Der Standardwert ist `False`.
- **`s3_client`**— Das Boto3 Amazon `client` S3-Objekt, das zum Speichern von Objekten in Amazon S3 verwendet werden soll. Verwenden Sie dies, wenn Sie den Amazon S3

S3-Client steuern möchten (z. B. benutzerdefinierte Amazon S3 S3-Konfiguration oder Anmeldeinformationen). Standardmäßig `boto3.client("s3")` bei der ersten Verwendung, sofern dies nicht zuvor festgelegt wurde.

Beispiel: Veröffentlichen von Nachrichten in Amazon SNS mit einer in Amazon S3 gespeicherten Nutzlast

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Beispiele für ein Amazon-SNS-Thema und eine Amazon-SQS-Warteschlange.
- Hängen Sie die Richtlinie an die Amazon SQS SQS-Warteschlange an, um die Nachricht vom Amazon SNS SNS-Thema zu erhalten.
- Die abonnierte Queue kann nun Nachrichten vom Thema empfangen.
- Veröffentlichen Sie eine Testnachricht mit dem erweiterten Amazon SNS SNS-Client, der Themenressource und der PlatformEndpoint Ressource.
- Die Nachrichtennutzlast wird in Amazon S3 gespeichert und der Verweis darauf wird veröffentlicht.
- Drucken Sie die veröffentlichte Nachricht aus der Warteschlange zusammen mit der aus Amazon S3 abgerufenen Originalnachricht aus.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Python. Die Nachricht, die Sie senden, verweist auf ein Amazon-S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

```
import boto3
from sns_extended_client import SNSExtendedClientSession
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store"  # S3 bucket with the
 given bucket name is a resource which is created and accessible with the given AWS
 credentials
TOPIC_NAME = "---TOPIC-NAME---"
QUEUE_NAME = "---QUEUE-NAME---"

def allow_sns_to_write_to_sqs(topicarn, queuearn):
    policy_document = """{{
        "Version":"2012-10-17",
        "Statement":[
            {{
```

```
            "Sid":"MyPolicy",
            "Effect":"Allow",
            "Principal" : {{"AWS" : "*"}},
            "Action":"SQS:SendMessage",
            "Resource": "{}",
            "Condition":{{
                "ArnEquals":{{
                "aws:SourceArn": "{}"
                }}
            }}
            }}
        ]
        }}""".format(queuearn, topicarn)

    return policy_document

def get_msg_from_s3(body,sns_extended_client):
    """Handy Helper to fetch message from S3"""
    json_msg = loads(body)
    s3_object = sns_extended_client.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


def fetch_and_print_from_sqs(sqs, queue_url,sns_extended_client):
    sqs_msg = sqs.receive_message(
        QueueUrl=queue_url,
        AttributeNames=['All'],
        MessageAttributeNames=['All'],
        VisibilityTimeout=0,
        WaitTimeSeconds=0,
        MaxNumberOfMessages=1
    ).get("Messages")[0]

    message_body = sqs_msg.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is:
 {}\n".format(get_msg_from_s3(message_body,sns_extended_client)))

    # Delete the Processed Message
    sqs.delete_message(
        QueueUrl=queue_url,
```

```
        ReceiptHandle=sqs_msg['ReceiptHandle']
    )


sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
sns_topic_arn = create_topic_response.get("TopicArn")

# create and subscribe an sqs queue to the sns client
sqs = boto3.client("sqs",region_name="us-east-1")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
sqs_queue_arn = sqs.get_queue_attributes(
    QueueUrl=demo_queue_url, AttributeNames=["QueueArn"]
)["Attributes"].get("QueueArn")

# Adding policy to SQS queue such that SNS topic can send msg to SQS queue
policy_json = allow_sns_to_write_to_sqs(sns_topic_arn, sqs_queue_arn)
response = sqs.set_queue_attributes(
    QueueUrl = demo_queue_url,
    Attributes = {
        'Policy' : policy_json
    }
)

# Set the RawMessageDelivery subscription attribute to TRUE if you want to use
# SQSExtendedClient to help with retrieving msg from S3
sns_extended_client.subscribe(TopicArn=sns_topic_arn, Protocol="sqs",
Endpoint=sqs_queue_arn
, Attributes={"RawMessageDelivery":"true"}
)

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# Change default s3_client attribute of sns_extended_client to use 'us-east-1' region
sns_extended_client.s3_client = boto3.client("s3", region_name="us-east-1")


# Below is the example that all the messages will be sent to the S3 bucket
sns_extended_client.always_through_s3 = True
sns_extended_client.publish(
    TopicArn=sns_topic_arn, Message="This message should be published to S3"
)
print("\n\nPublished using SNS extended client:")
```

```
fetch_and_print_from_sqs(sqs, demo_queue_url,sns_extended_client)  # Prints message
 stored in s3

# Below is the example that all the messages larger than 32 bytes will be sent to the
 S3 bucket
print("\nUsing decreased message size threshold:")

sns_extended_client.always_through_s3 = False
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=sns_topic_arn,
    Message="This message should be published to S3 as it exceeds the limit of the 32
 bytes",
)

fetch_and_print_from_sqs(sqs, demo_queue_url,sns_extended_client)  # Prints message
 stored in s3


# Below is the example to publish message using the SNS.Topic resource
sns_extended_client_resource = SNSExtendedClientSession().resource(
    "sns", region_name="us-east-1"
)

topic = sns_extended_client_resource.Topic(sns_topic_arn)
topic.large_payload_support = s3_extended_payload_bucket

# Change default s3_client attribute of topic to use 'us-east-1' region
topic.s3_client = boto3.client("s3", region_name="us-east-1")

topic.always_through_s3 = True
# Can Set custom S3 Keys to be used to store objects in S3
topic.publish(
    Message="This message should be published to S3 using the topic resource",
    MessageAttributes={
        "S3Key": {
            "DataType": "String",
            "StringValue": "347c11c4-a22c-42e4-a6a2-9b5af5b76587",
        }
    },
)
print("\nPublished using Topic Resource:")
fetch_and_print_from_sqs(sqs, demo_queue_url,topic)
```

```
# Below is the example to publish message using the SNS.PlatformEndpoint resource
sns_extended_client_resource = SNSExtendedClientSession().resource(
    "sns", region_name="us-east-1"
)

platform_endpoint = sns_extended_client_resource.PlatformEndpoint(sns_topic_arn)
platform_endpoint.large_payload_support = s3_extended_payload_bucket

# Change default s3_client attribute of platform_endpoint to use 'us-east-1' region
platform_endpoint.s3_client = boto3.client("s3", region_name="us-east-1")

platform_endpoint.always_through_s3 = True
# Can Set custom S3 Keys to be used to store objects in S3
platform_endpoint.publish(
    Message="This message should be published to S3 using the PlatformEndpoint
 resource",
    MessageAttributes={
        "S3Key": {
            "DataType": "String",
            "StringValue": "247c11c4-a22c-42e4-a6a2-9b5af5b76587",
        }
    },
)
print("\nPublished using PlatformEndpoint Resource:")
fetch_and_print_from_sqs(sqs, demo_queue_url,platform_endpoint)
```

Ausgabe

```
Published using SNS extended client:
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
 {"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"}]
Message Stored in S3 Bucket is: This message should be published to S3

Using decreased message size threshold:
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
 {"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"}]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the limit of the 32 bytes

Published using Topic Resource:
```

```
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
 {"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"}]
Message Stored in S3 Bucket is: This message should be published to S3 using the topic
 resource

Published using PlatformEndpoint Resource:
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
 {"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"}]
Message Stored in S3 Bucket is: This message should be published to S3 using the
 PlatformEndpoint resource
```

Amazon-SNS-Nachrichtenattribute

Amazon SNS unterstützt die Übermittlung von Nachrichtenattributen, die Ihnen die Bereitstellung strukturierter Metadatenelemente (Zeitstempel, Geodaten, Signaturen, IDs usw.) zur Nachricht ermöglichen. Für SQS-Abonnements können maximal 10 Nachrichtenattribute gesendet werden, wenn die Übermittlung unformatierter Nachrichten aktiviert ist. Wenn mehr als 10 Nachrichtenattribute gesendet werden sollen, muss die Übermittlung unformatierter Nachrichten deaktiviert sein. Nachrichten werden mit mehr als 10 Nachrichtenattributen, die an Amazon-SQS-Abonnements mit aktiviertem Raw Message Delivery gerichtet sind, als clientseitige Fehler verworfen.

Nachrichtenattribute sind optional und separat vom Nachrichtentext, werden aber mit diesem versendet. Der Receiver kann diese Informationen verwenden, um zu entscheiden, wie die Nachricht behandelt wird, ohne vorher den Nachrichtentext verarbeiten zu müssen.

Informationen zum Senden von Nachrichten mit Attributen mithilfe des AWS Management Console oder des AWS SDK für Java finden Sie im Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen Tutorial.

> **Note**
>
> Nachrichtenattribute werden nur bei der Nachrichtenstruktur "Zeichenfolge" und nicht bei "JSON" gesendet werden.

Sie können die Nachrichtenattribute auch zur Strukturierung der Push-Benachrichtigung für mobile Endpunkte verwenden. In diesem Fall werden die Nachrichtenattribute nur verwendet, um Ihnen bei

der Gliederung der Push-Benachrichtigung zu helfen. Die Attribute werden nicht an den Endpunkt übermittelt, wie beim Senden von Nachrichten mit Nachrichtenattributen an Amazon SQS-Endpunkte.

Sie können Nachrichtenattribute auch verwenden, um Ihre Nachrichten filterbar für Abonnement-Filterrichtlinien zu machen. Sie können Richtlinien auf Themen-Abonnements anwenden. Wenn eine Filterrichtlinie angewendet wird, bei der der Geltungsbereich der Filterrichtlinie auf `MessageAttributes` (Standard) festgelegt ist, erhält ein Abonnement nur diejenigen Nachrichten mit Attributen, die die Richtlinie akzeptiert. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

> **Note**
>
> Wenn Nachrichtenattribute für die Filterung verwendet werden, muss der Wert eine gültige JSON-Zeichenfolge sein. Dadurch wird sichergestellt, dass die Nachricht an ein Abonnement mit aktivierter Filterung von Nachrichtenattributen zugestellt wird.

Nachrichtenattributelemente und Validierung

Jedes Nachrichtenattribut besteht aus den folgenden Elementen:

- Name – Der Name des Nachrichtenattributs kann folgende Zeichen enthalten: A-Z, a-z, 0-9, Unterstrich (_), Bindestrich (-) und Punkt (.). Der Name darf nicht mit einem Punkt beginnen oder enden und darf nicht mehrere aufeinanderfolgende Punkte enthalten. Es wird zwischen Groß- und Kleinschreibung unterschieden. Der Name muss innerhalb der Attributnamen für die Nachricht eindeutig sein. Er kann eine Länge von bis zu 256 Zeichen umfassen. Er darf nicht mit `AWS.` oder `Amazon.` (oder eine Variante bei der Groß-/Kleinschreibung) beginnen, da diese Präfixe für die Verwendung durch Amazon Web Services reserviert sind.
- Typ – Die unterstützten Attributdatentypen für Nachrichten sind `String`, `String.Array`, `Number` und `Binary`. Der Datentyp unterliegt denselben Beschränkungen bezüglich des Inhalts wie der Nachrichtentext. Weitere Informationen finden Sie im Abschnitt Datentypen für Nachrichtenattribute und Validierung.
- Wert – Der benutzerdefinierte Wert des Nachrichtenattributs. Bei Zeichenfolgendatentypen muss das Wertattribut denselben Inhaltsbeschränkungen unterliegen wie der Nachrichtentext. Wenn das Nachrichtenattribut jedoch zum Filtern verwendet wird, muss es sich bei dem Wert um eine gültige JSON-Zeichenfolge handeln, um die Kompatibilität mit den Amazon SNS SNS-Abonnementfilterrichtlinien sicherzustellen. Weitere Informationen finden Sie unter der Aktion Veröffentlichen in der Amazon Simple Notification Service API-Referenz.

Name, Typ und Wert dürfen nicht leer oder null sein. Dasselbe gilt für den Nachrichtentext. Alle Teile des Nachrichtenattributs einschließlich Name, Typ und Wert zählen zur Größenbeschränkung der Nachricht, die bei 256 KB liegt.

Datentypen für Nachrichtenattribute und Validierung

Über den Datentyp des Nachrichtenattributs wird festgelegt, wie die Nachrichtenattributwerte von Amazon SNS verarbeitet werden. Wenn Sie beispielsweise den Datentyp Zahl wählen, wertet Amazon SNS den Wert als Zahl aus.

Amazon SNS unterstützt die folgenden logischen Datentypen für alle Endpunkte, außer wie erwähnt:

- Zeichenfolge – Die Zeichenfolgen sind Unicode mit binärer UTF-8-Kodierung. Eine Liste der Codewerte finden Sie unter http://en.wikipedia. org/wiki/ASCII#ASCII_printable_characters.

  > **Note**
  >
  > Ersatzwerte werden in den Nachrichtenattributen nicht unterstützt. Wenn Sie beispielsweise einen Ersatzwert zur Darstellung eines Emojis verwenden, erhalten Sie den folgenden Fehler: `Invalid attribute value was passed in for message attribute`.

- String.Array – Ein Array, formatiert als Zeichenfolge, das mehrere Werte enthalten kann. Die Werte können Zeichenfolgen, Zahlen oder die Schlüsselwörter `true`, `false` und `null` sein. Für ein String.Array vom Typ „number“ oder „boolean“ sind keine Preise erforderlich. Mehrere String.Array-Werte werden durch Komma getrennt.

  Dieser Datentyp wird für AWS Lambda Abonnements nicht unterstützt. Wenn Sie diesen Datentypen für Lambda-Endpunkte festlegen, wird er als der `String`-Datentyp in der JSON-Nutzlast angesehen, den Amazon SNS an Lambda liefert.
- Zahl – Zahlen sind positive oder negative Ganzzahlen oder Gleitkommazahlen. Der Umfang und die Genauigkeit von Zahlen reichen aus, um die meisten möglichen Werte zu umfassen, die von den Zahlentypen Integer, Float und Double normalerweise unterstützt werden. Eine Zahl kann einen Wert von -10^9 bis 10^9 haben, mit 5 Stellen Genauigkeit nach dem Komma. Nullen am Anfang und am Ende werden abgeschnitten.

  Dieser Datentyp wird für AWS Lambda Abonnements nicht unterstützt. Wenn Sie diesen Datentypen für Lambda-Endpunkte festlegen, wird er als der `String`-Datentyp in der JSON-Nutzlast angesehen, den Amazon SNS an Lambda liefert.

- Binär – Mit den Binärtypattributen können Sie alle binären Daten speichern, z. B. komprimierte oder verschlüsselte Daten und Bilder.

Reservierte Nachrichtenattribute für mobile Push-Benachrichtigungen

Die folgende Tabelle enthält die reservierten Nachrichtenattribute für mobile Push-Benachrichtigungsservices, mit denen Sie Ihre Push-Benachrichtigung strukturieren können:

Push-Benachrichtigungsservice	Reserviertes Nachrichtenattribut
ADM	`AWS.SNS.MOBILE.ADM.TTL`
APNs[1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`
	`AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS.COLLAPSE_ID`
	`AWS.SNS.MOBILE.APNS.PRIORITY`
	`AWS.SNS.MOBILE.APNS.PUSH_TYPE`
	`AWS.SNS.MOBILE.APNS.TOPIC`
	`AWS.SNS.MOBILE.APNS.TTL`
Baidu	`AWS.SNS.MOBILE.BAIDU.DeployStatus`
	`AWS.SNS.MOBILE.BAIDU.MessageKey`

Push-Benachrichtigungsservice	Reserviertes Nachrichtenattribut
	`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 lehnt Amazon-SNS-Benachrichtigungen ab, wenn die Nachrichtenattribute nicht ihren Anforderungen entsprechen. Weitere Informationen finden Sie unter Senden von Benachrichtigungsanfragen an APNs auf der Apple-Entwickler-Website.

Batch-Messaging in Amazon SNS

Was ist Batch-Messaging?

Eine Alternative zur Veröffentlichung von Nachrichten in Standard- oder FIFO-Themen in einzelnen `Publish` API-Anfragen ist die Verwendung der Amazon SNS `PublishBatch`-API, um bis zu 10 Nachrichten in einer einzigen API-Anfrage zu veröffentlichen. Durch das Versenden von Nachrichten in Batches können Sie die Kosten, die mit der Verbindung von verteilten Anwendungen (A2A-Messaging) oder dem Senden von Benachrichtigungen an Personen (A2P-Messaging) mit Amazon SNS verbunden sind, bis zu einem Faktor von 10 reduzieren. Amazon SNS verfügt über Kontingente, wie viele Nachrichten Sie je nach Region, in der Sie tätig sind, für ein Thema pro Sekunde veröffentlichen können. Sehen Sie sich die Seite Amazon-SNS-Endpunkte und -Kontingente im Allgemeine AWS-Referenz-Leitfaden an, um weitere Informationen zu API-Kontingenten zu erhalten.

> **Note**
>
> Die Gesamtgröße aller Nachrichten, die Sie in einer einzigen `PublishBatch` API-Anfrage senden, darf 262.144 Byte (256 KiB) nicht überschreiten.
> Die `PublishBatch`-API verwendet dieselbe `Publish`-API-Aktion für IAM-Richtlinien.

Wie funktioniert Batch-Messaging?

Das Veröffentlichen von Nachrichten mit der `PublishBatch`-API ähnelt dem Veröffentlichen von Nachrichten mit der `Publish`-API. Der Hauptunterschied besteht darin, dass jeder Nachricht innerhalb einer `PublishBatch`-API-Anfrage eine eindeutige Batch-ID (bis zu 80 Zeichen) zugewiesen werden muss. Auf diese Weise kann Amazon SNS individuelle API-Antworten für jede Nachricht innerhalb eines Batches zurückgeben, um zu bestätigen, dass jede Nachricht entweder veröffentlicht wurde oder dass ein Fehler aufgetreten ist. Für Nachrichten, die in FIFO-Themen veröffentlicht werden, müssen Sie zusätzlich zur Zuweisung einer eindeutigen Batch-ID noch eine `MessageDeduplicationID` und `MessageGroupId` für jede einzelne Nachricht zuweisen.

Beispiele

Veröffentlichen eines Batches von 10 Nachrichten in einem Standardthema

```
// Imports
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.PublishBatchRequest;
import software.amazon.awssdk.services.sns.model.PublishBatchRequestEntry;
import software.amazon.awssdk.services.sns.model.PublishBatchResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToTopic(SnsClient snsClient, String topicArn, int
 batchSize) {
    try {
        // Validate the batch size
        if (batchSize > MAX_BATCH_SIZE) {
            throw new IllegalArgumentException("Batch size cannot exceed " +
 MAX_BATCH_SIZE);
        }

        // Create the batch entries
        List<PublishBatchRequestEntry> entries = IntStream.range(0, batchSize)
                .mapToObj(i -> PublishBatchRequestEntry.builder()
                        .id("id" + i)
                        .message("message" + i)
                        .build())
                .collect(Collectors.toList());

        // Build the batch request
        PublishBatchRequest request = PublishBatchRequest.builder()
                .topicArn(topicArn)
                .publishBatchRequestEntries(entries)
                .build();

        // Publish the batch request
        PublishBatchResponse response = snsClient.publishBatch(request);

        // Handle successful messages
        response.successful().forEach(success -> {
            System.out.println("Successful Batch Id: " + success.id());
            System.out.println("Message Id: " + success.messageId());
        });
```

```
        // Handle failed messages
        response.failed().forEach(failure -> {
            System.err.println("Failed Batch Id: " + failure.id());
            System.err.println("Error Code: " + failure.code());
            System.err.println("Sender Fault: " + failure.senderFault());
            System.err.println("Error Message: " + failure.message());
        });

    } catch (SnsException e) {
        // Log and handle exceptions
        System.err.println("SNS Exception: " + e.awsErrorDetails().errorMessage());
    } catch (IllegalArgumentException e) {
        System.err.println("Validation Error: " + e.getMessage());
    }
}
```

Veröffentlichen eines Batches von 10 Nachrichten in einem FIFO-Thema

```
// Imports
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishBatchRequest;
import software.amazon.awssdk.services.sns.model.PublishBatchRequestEntry;
import software.amazon.awssdk.services.sns.model.PublishBatchResponse;
import software.amazon.awssdk.services.sns.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sns.model.SnsException;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToFifoTopic(SnsClient snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> PublishBatchRequestEntry.builder()
                        .id("id" + i)
                        .message("message" + i)
                        .messageGroupId("groupId")
                        .messageDeduplicationId("deduplicationId" + i)
                        .build())
```

```
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = PublishBatchRequest.builder()
                .topicArn(topicArn)
                .publishBatchRequestEntries(entries)
                .build();

        // Publish the batch request
        PublishBatchResponse response = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        response.successful().forEach(success -> {
            System.out.println("Batch Id for successful message: " + success.id());
            System.out.println("Message Id for successful message: " +
 success.messageId());
            System.out.println("Sequence Number for successful message: " +
 success.sequenceNumber());
        });

        // Handle the failed messages
        response.failed().forEach(failure -> {
            System.err.println("Batch Id for failed message: " + failure.id());
            System.err.println("Error Code for failed message: " + failure.code());
            System.err.println("Sender Fault for failed message: " +
 failure.senderFault());
            System.err.println("Failure Message for failed message: " +
 failure.message());
        });

    } catch (SnsException e) {
        // Handle any exceptions from the request
        System.err.println("SNS Exception: " + e.awsErrorDetails().errorMessage());
    }
}
```

Löschen eines Amazon SNS SNS-Themas und -Abonnements

Wenn ein Thema gelöscht wird, werden die zugehörigen Abonnements asynchron gelöscht. Kunden können zwar weiterhin auf diese Abonnements zugreifen, die Abonnements sind jedoch nicht mehr mit dem Thema verknüpft — auch wenn Sie das Thema mit demselben Namen neu erstellen. Wenn ein Herausgeber versucht, eine Nachricht zu dem gelöschten Thema zu veröffentlichen, erhält er eine

Fehlermeldung, dass das Thema nicht existiert. Ebenso führt jeder Versuch, das gelöschte Thema zu abonnieren, zu einer Fehlermeldung. Ein Abonnement, das noch nicht bestätigt ist, kann nicht gelöscht werden. Amazon SNS löscht nicht bestätigte Abonnement nach 48 Stunden automatisch.

Um ein Amazon SNS SNS-Thema oder -Abonnement zu löschen, verwenden Sie den AWS Management Console

Das Löschen eines Amazon SNS SNS-Themas oder -Abonnements gewährleistet eine effiziente Ressourcenverwaltung, verhindert unnötigen Gebrauch und sorgt für Ordnung in der Amazon SNS SNS-Konsole. Dieser Schritt trägt dazu bei, potenzielle Kosten aufgrund ungenutzter Ressourcen zu vermeiden, und optimiert die Verwaltung, indem Themen oder Abonnements entfernt werden, die nicht mehr benötigt werden.

Um ein Thema mit dem zu löschen AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Wählen Sie auf der Seite Topics (Themen) ein Thema und anschließend Edit (Bearbeiten) aus.
4. Geben Sie im Dialogfenster Delete (Löschen) den Text `delete me` ein und wählen Sie dann Delete (Löschen) aus.

   In der Konsole wird das Thema gelöscht.

Um ein Abonnement mit dem zu löschen AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).
3. Wählen Sie auf der Seite Abonnements ein Abonnement mit dem Status Bestätigt aus und klicken Sie dann auf Löschen.
4. Wählen Sie im Dialogfeld Delete subscription die Option Delete aus.

   Die Konsole löscht das Abonnement.

So löschen Sie ein Amazon SNS-Abonnement und Amazon SNS-Thema mithilfe der AWS SDK

Um ein AWS SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Die folgenden Code-Beispiele zeigen, wie `DeleteTopic` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-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;
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

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

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

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

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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
  //   }
```

```
  // }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

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

- API-Details finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Swift.

Nächste Schritte

Nachdem Sie nun ein Thema mit einem Abonnement erstellt und Nachrichten an das Thema gesendet haben, können Sie Folgendes ausprobieren:

- Erkunden Sie das AWS -Entwickler-Center.
- Informationen zum Schutz Ihrer Daten und zum Zugriff auf diese finden Sie im Abschnitt Sicherheit.
- Aktivieren der serverseitigen Verschlüsselung für ein Thema.
- Aktivieren Sie die serverseitige Verschlüsselung für ein Thema mit einem Abbonement für Amazon Simple Queue Service (Amazon SQS).
- Abonnieren Sie AWS Event Fork Pipelines für ein Thema.

Strategien zur Reihenfolge und Deduplizierung von Nachrichten mithilfe von Amazon SNS FIFO-Themen

Dieses Thema enthält Informationen zu den Merkmalen und Funktionen der Amazon SNS FIFO (First-In-First-Out) -Themen und darüber, wie sie in Amazon SQS FIFO-Warteschlangen integriert werden. Sie erfahren, wie Sie diese Services zusammen verwenden können, um eine strikte Nachrichtenreihenfolge und Deduplizierung sicherzustellen, was für Anwendungen, die Datenkonsistenz erfordern, unerlässlich ist. Dieser Inhalt behandelt die spezifischen Anwendungsfälle, in denen Amazon SNS FIFO-Themen von Vorteil sind, und bietet Einblicke in Szenarien, in denen Nachrichtenreihenfolge und Eindeutigkeit entscheidend sind.

Außerdem erfahren Sie mehr über die technischen Details der Reihenfolge und Gruppierung von Nachrichten und wie sich diese auf die Nachrichtenzustellung auswirken. Im Thema Nachrichtendeduplizierung werden die Mechanismen erläutert, mit denen doppelte Nachrichten verhindert werden, sodass sichergestellt wird, dass jede Nachricht nur einmal verarbeitet wird. Darüber hinaus erfahren Sie mehr über Nachrichtenfilterung, Sicherheit und Beständigkeit, die für die Aufrechterhaltung der Integrität und Zuverlässigkeit Ihres Messaging-Systems wichtig sind. Dieser Inhalt enthält auch Informationen zur Archivierung und Wiedergabe von Nachrichten und bietet Strategien für die Verwaltung von Nachrichtenverläufen. Praktische Codebeispiele helfen Ihnen auch bei der Implementierung dieser Funktionen in Ihren eigenen Anwendungen und vermitteln Ihnen praktische Erfahrungen mit Amazon SNS FIFO-Themen und deren Integration in Amazon SQS FIFO-Warteschlangen.

FIFO-Themen mit hohem Durchsatz in Amazon SNS

FIFO-Themen mit hohem Durchsatz in Amazon SNS verwalten effizient einen hohen Nachrichtendurchsatz bei gleichzeitiger Einhaltung einer strikten Nachrichtenreihenfolge und gewährleisten so Zuverlässigkeit und Skalierbarkeit für Anwendungen, die zahlreiche Nachrichten verarbeiten. Diese Lösung ist ideal für Szenarien, in denen sowohl ein hoher Durchsatz als auch eine geordnete Nachrichtenzustellung erforderlich sind. Um den Nachrichtendurchsatz mithilfe von FIFO-Themen mit hohem Durchsatz zu verbessern, wird empfohlen, die Anzahl der Nachrichtengruppen zu erhöhen. Weitere Informationen zu Nachrichtenkontingenten mit hohem Durchsatz finden Sie unter Amazon SNS SNS-Servicekontingenten in der Allgemeine Amazon Web Services-Referenz.

Anwendungsfälle für hohen Durchsatz bei Amazon SNS FIFO-Themen

Die folgenden Anwendungsfälle beleuchten die vielfältigen Anwendungsmöglichkeiten von FIFO-Themen mit hohem Durchsatz und verdeutlichen deren Wirksamkeit in allen Branchen und Szenarien:

- Datenverarbeitung in Echtzeit: Anwendungen, die sich mit Echtzeit-Datenströmen befassen, wie z. B. die Ereignisverarbeitung oder die Erfassung von Telemetriedaten, können von FIFO-Themen mit hohem Durchsatz profitieren, um den kontinuierlichen Zustrom von Nachrichten zu bewältigen und gleichzeitig deren Reihenfolge für eine genaue Analyse beizubehalten.
- E-Commerce-Auftragsabwicklung: Auf E-Commerce-Plattformen, auf denen es entscheidend ist, die Reihenfolge der Kundentransaktionen aufrechtzuerhalten, stellen FIFO-Themen mit hohem Durchsatz sicher, dass Bestellungen auch während der Haupteinkaufssaison sequentiell und ohne Verzögerungen geliefert werden.
- Finanzdienstleistungen: Finanzinstitute, die hochfrequente Handels- oder Transaktionsdaten verarbeiten, verlassen sich auf FIFO-Themen mit hohem Durchsatz, um Marktdaten und Transaktionen mit minimaler Latenz zu verarbeiten und gleichzeitig die strengen regulatorischen Anforderungen für die Bestellung von Nachrichten einzuhalten.
- Medienstreaming: Streaming-Plattformen und Medienvertriebsdienste nutzen FIFO-Themen mit hohem Durchsatz, um die Bereitstellung von Mediendateien und Streaming-Inhalten zu verwalten und so ein reibungsloses Wiedergabeerlebnis für Benutzer zu gewährleisten und gleichzeitig die richtige Reihenfolge der Inhaltsbereitstellung beizubehalten

Partitionen und Datenverteilung für hohen Durchsatz für Amazon SNS FIFO-Themen

Bei Themen mit hohem Durchsatz verteilt Amazon SNS FIFO-Themendaten auf mehrere Partitionen. Eine Partition ist eine Kapazitätszuweisung für ein Thema, die automatisch über mehrere Availability Zones innerhalb einer repliziert wird. AWS-Region Sie verwalten keine Partitionen. Stattdessen verwaltet Amazon SNS automatisch Partitionen in Ihrem Namen, basierend auf der Eingangsrate.

Bei FIFO-Themen ändert Amazon SNS die Anzahl der Partitionen in einem Thema in den folgenden Situationen:

- Wenn sich die aktuelle Veröffentlichungsrate dem Niveau nähert oder übersteigt, das die vorhandenen Partitionen unterstützen können, werden zusätzliche Partitionen zugewiesen, bis das

Thema das regionale Kontingent erreicht. Informationen zu Kontingenten finden Sie unter Amazon SNS SNS-Servicekontingente in der Allgemeine Amazon Web Services-Referenz.

- Wenn die aktuellen Partitionen wenig ausgelastet sind, kann die Anzahl der Partitionen reduziert werden.

Die Partitionsverwaltung wird automatisch im Hintergrund ausgeführt und ist für Ihre Anwendungen transparent. Ihr Thema und Ihre Nachrichten sind jederzeit verfügbar.

> **Note**
>
> Eine vorübergehende Drosselung der API für Veröffentlichungen kann auftreten, wenn Sie plötzlich und deutlich mehr Besucher zu Ihrem Thema aufrufen, während Sie gleichzeitig ein Vielfaches des üblichen Volumens versenden. Diese Drosselung kann bis zur Dauer des Deduplizierungsfensters andauern, während das Thema entsprechend dem erhöhten Traffic skaliert wird.

Verteilen von Daten nach Nachrichtengruppen IDs

Wenn eine Nachricht zu einem FIFO-Thema veröffentlicht wird, verwendet Amazon SNS den Wert der Nachrichtengruppen-ID jeder Nachricht als Eingabe für eine interne Hash-Funktion. Der Ausgabewert der Hash-Funktion bestimmt, welche Partition die Nachricht verarbeitet. Eine oder mehrere Nachrichtengruppen IDs können von einer bestimmten Partition verarbeitet werden.

> **Note**
>
> Amazon SNS ist für die gleichmäßige Verteilung von Elementen auf die Partitionen eines FIFO-Themas optimiert, unabhängig von der Anzahl der Partitionen. AWS empfiehlt die Verwendung einer Nachrichtengruppe IDs , die eine große Anzahl unterschiedlicher Werte haben kann.

Aktivieren Sie einen hohen Durchsatz für Ihr Amazon SNS FIFO-Thema

Standardmäßig sind Amazon SNS FIFO-Themen für die Deduplizierung auf Themenebene konfiguriert. Dies wird durch das Themenattribut gesteuert, das auf

`FifoThroughputScope`eingestellt ist, `Topic` und sie haben eingeschränktere Durchsatzquoten, siehe Amazon SNS SNS-Servicekontingente in der. Allgemeine Amazon Web Services-Referenz

Um einen hohen Durchsatz für Ihr Amazon SNS FIFO-Thema zu aktivieren, aktualisieren Sie das `FifoThroughputScope` Attribut auf. `MessageGroup` Diese Änderung kann über die Konsole oder mithilfe des SDK AWS CLI und vorgenommen werden. Sie kann auch während der Themenerstellung vorgenommen werden. Amazon SNS empfiehlt dies, um das beste Kundenerlebnis zu erzielen und die Wahrscheinlichkeit zu verringern, dass Ihr Thema gedrosselt wird.

> **Important**
>
> Sobald Sie das für ein Thema aktiviert haben`MessageGroup`, kann es nicht mehr `FifoThroughputScope` auf den Durchsatz zurückgesetzt werden. `Topic`

Aktivieren Sie den Hochdurchsatzmodus für jede abonnierte Amazon SQS FIFO-Warteschlange

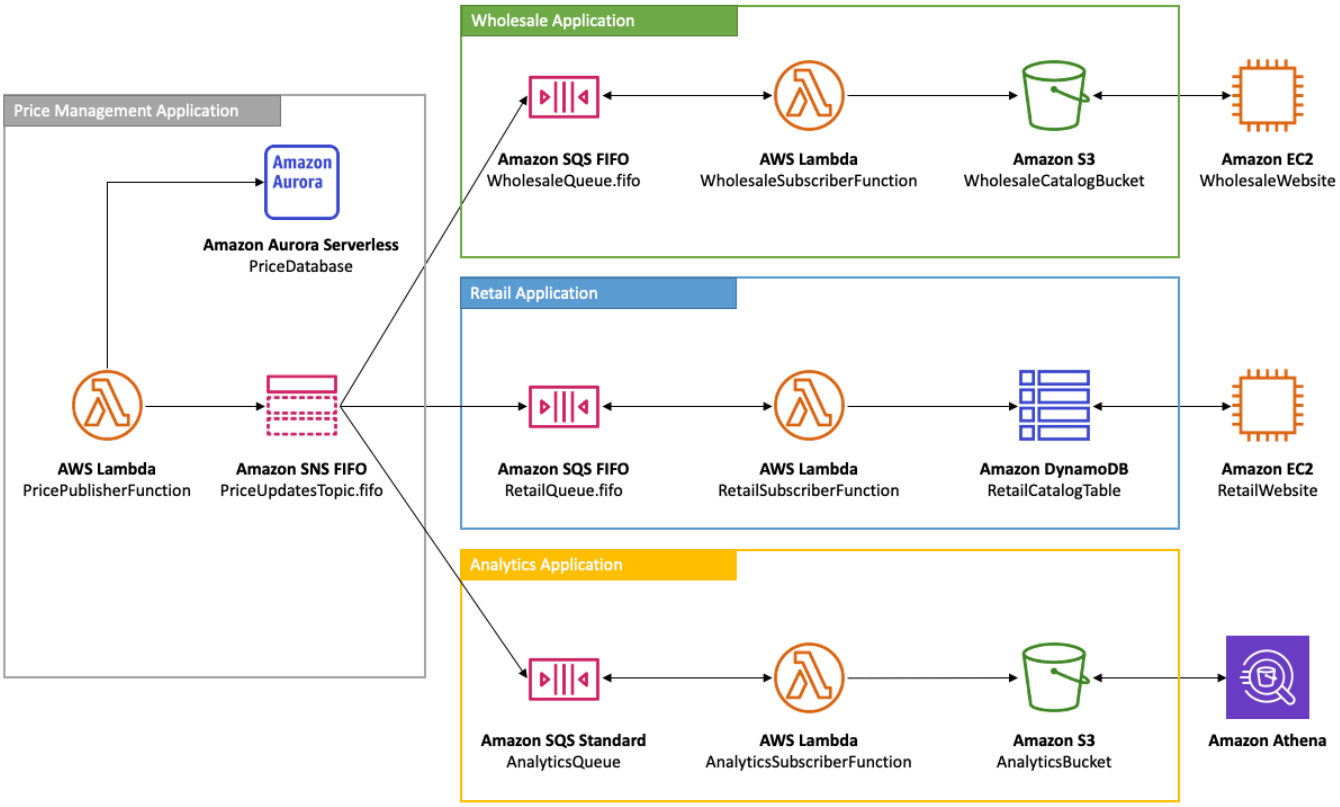
Wenn Sie in Ihrem Amazon SNS FIFO-Thema mit aktiviertem hohem Durchsatz veröffentlichen und eine oder mehrere Amazon SQS FIFO-Warteschlangen abonniert sind, wird empfohlen, den hohen Durchsatz in Ihren Amazon SQS FIFO-Warteschlangen zu aktivieren, damit Ihr Amazon SNS FIFO-Thema mit hohem Durchsatz reibungslos zugestellt werden kann. Weitere Informationen finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen im Entwicklerhandbuch zu Amazon Simple Queue Service.

Anwendungsbeispiel für ein Amazon SNS FIFO-Thema

Im folgenden Beispiel wird eine E-Commerce-Plattform beschrieben, die von einem Hersteller von Autoteilen mithilfe von Amazon-SNS-FIFO-Themen und Amazon-SQS-Warteschlangen erstellt wurde. Die Plattform besteht aus drei Serverless-Anwendungen:

- Lagerverwalter verwenden eine Preisverwaltungsanwendung, um den Preis für jeden Artikel auf Lager festzulegen. Bei diesem Unternehmen können sich die Produktpreise aufgrund von Wechselkursschwankungen, Marktnachfrage und Verschiebungen in der Vertriebsstrategie ändern. Die Preismanagement-Anwendung verwendet eine AWS Lambda -Funktion, die Preisaktualisierungen für ein Amazon SNS-FIFO-Thema veröffentlicht, wenn sich die Preise ändern.

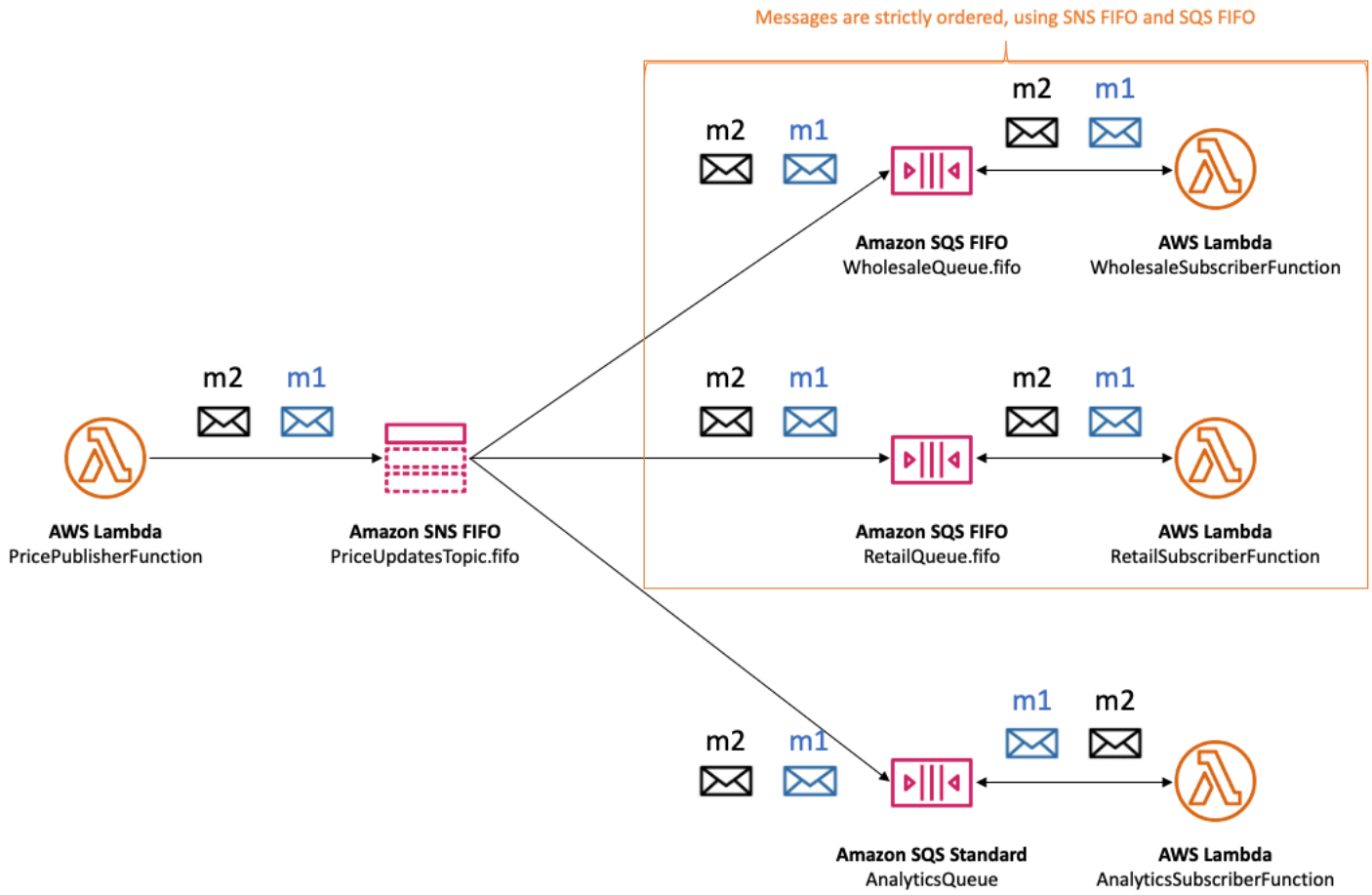
- Eine Wholesale-Anwendung bietet das Backend für eine Website, wo Auto-Karosseriewerkstätten und Autohersteller die Autoteile des Unternehmens in großen Mengen kaufen können. Um Preisänderungsbenachrichtigungen zu erhalten, abonniert die Großhandels-Anwendung ihre Amazon-SQS-FIFO-Warteschlange für das Amazon-SNS-FIFO-Thema der Preisverwaltungsanwendung.
- Eine Einzelhandelsanwendung bietet das Backend für eine andere Website, auf der Autobesitzer und Auto-Tuning-Enthusiasten einzelne Autoteile für ihre Fahrzeuge kaufen können. Um Preisänderungsbenachrichtigungen zu erhalten, abonniert die Einzelhandelsanwendung ebenfalls auch ihre Amazon-SQS-FIFO-Warteschlange für das Amazon-SNS-FIFO-Thema der Preisverwaltungsanwendung.
- Eine Analyseanwendung, die Preisaktualisierungen zusammenfasst und in einem Amazon-S3-Bucket speichert, ermöglicht Amazon Athena, den Bucket für Business Intelligence (BI)-Zwecke abzufragen. Um Preisänderungsbenachrichtigungen zu erhalten, abonniert die Analyse-Anwendung ihre Amazon-SQS-Standard-Warteschlange für das Amazon-SNS-FIFO-Thema der Preisverwaltungsanwendung. Im Gegensatz zu den anderen Anwendungen erfordert die Analyse-Anwendung keine strikte Reihenfolge der Preisaktualisierungen.



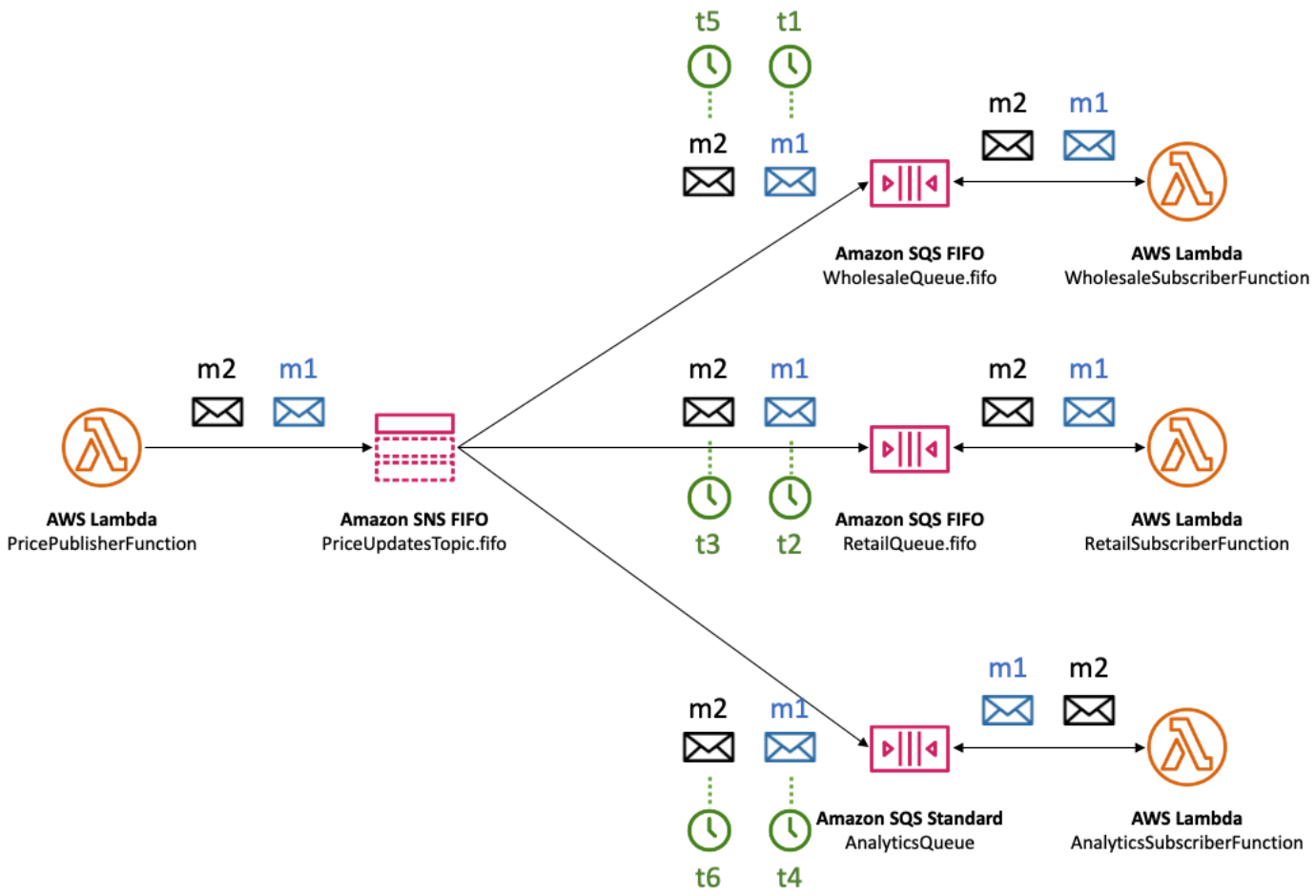
Damit die Groß- und Einzelhandelsanwendungen Preisaktualisierungen in der richtigen Reihenfolge erhalten, muss die Preisverwaltungsanwendung ein streng geordnetes Nachrichtenverteilungssystem verwenden. Die Verwendung von Amazon-SNS-FIFO-Themen und Amazon-SQS-FIFO-Warteschlangen ermöglicht die Verarbeitung von Nachrichten in der Reihenfolge und ohne Duplizierung. Weitere Informationen finden Sie unter Einzelheiten zur Bestellung von Amazon SNS SNS-Nachrichten für FIFO-Themen. Code-Snippets, die diesen Anwendungsfall implementieren, finden Sie unterAmazon SNS SNS-Codebeispiele für FIFO-Themen.

Einzelheiten zur Bestellung von Amazon SNS SNS-Nachrichten für FIFO-Themen

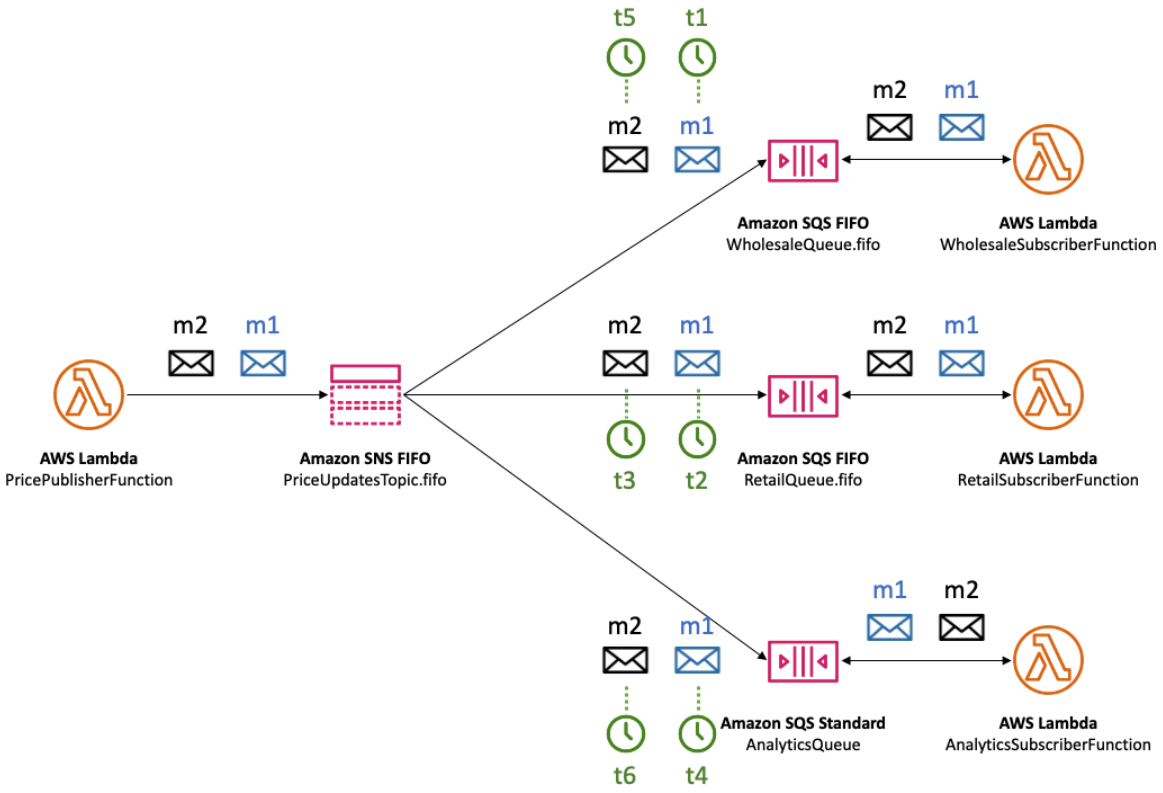
In einem Amazon-SNS-FIFO-Thema werden Nachrichten an abonnierte Amazon-SQS-Warteschlangen in der genauen Reihenfolge übermittelt, in der die Nachrichten für das Thema veröffentlicht werden, und zwar nur einmal. Beim Abonnement einer Amazon-SQS-FIFO-Warteschlange erhält der Benutzer der Warteschlange die Nachrichten in der exakten Reihenfolge, in der die Nachrichten an die Warteschlange zugestellt werden, und ohne Duplikate. Wenn eine -Amazon-SQS-Standard-Warteschlange abonniert ist, kann es jedoch vorkommen, dass der Benutzer der Warteschlange Nachrichten nicht in der richtigen Reihenfolge erhält, und dazu mehr als einmal. Dies ermöglicht eine weitere Entkopplung der Abonnenten von den Herausgebern, was den Abonnenten mehr Flexibilität in Bezug auf die Nachrichtennutzung und die Kostenoptimierung bietet, wie das folgende Diagramm zeigt, das auf der Grundlage von Anwendungsbeispiel für ein Amazon SNS FIFO-Thema erstellt wurde.



Beachten Sie, dass es keine implizite Bestellung der Abonnenten gibt. Das folgende Beispiel zeigt, dass die Nachricht m1 zuerst an den Großhandels-Abonnenten, dann an den Einzelhandel- und schließlich an den Analyse-Abonnenten geht. Nachricht m2 wird zuerst an den Einzelhandels, dann an den Großhandels- und schließlich an den Analyse-Abonnenten zugestellt. Obwohl die beiden Nachrichten an die Abonnenten in einer anderen Reihenfolge zugestellt werden, wird die Nachrichtenreihenfolge für jeden Amazon-SQS-FIFO-Abonnenten beibehalten. Jeder Teilnehmer wird isoliert von anderen Abonnenten wahrgenommen.

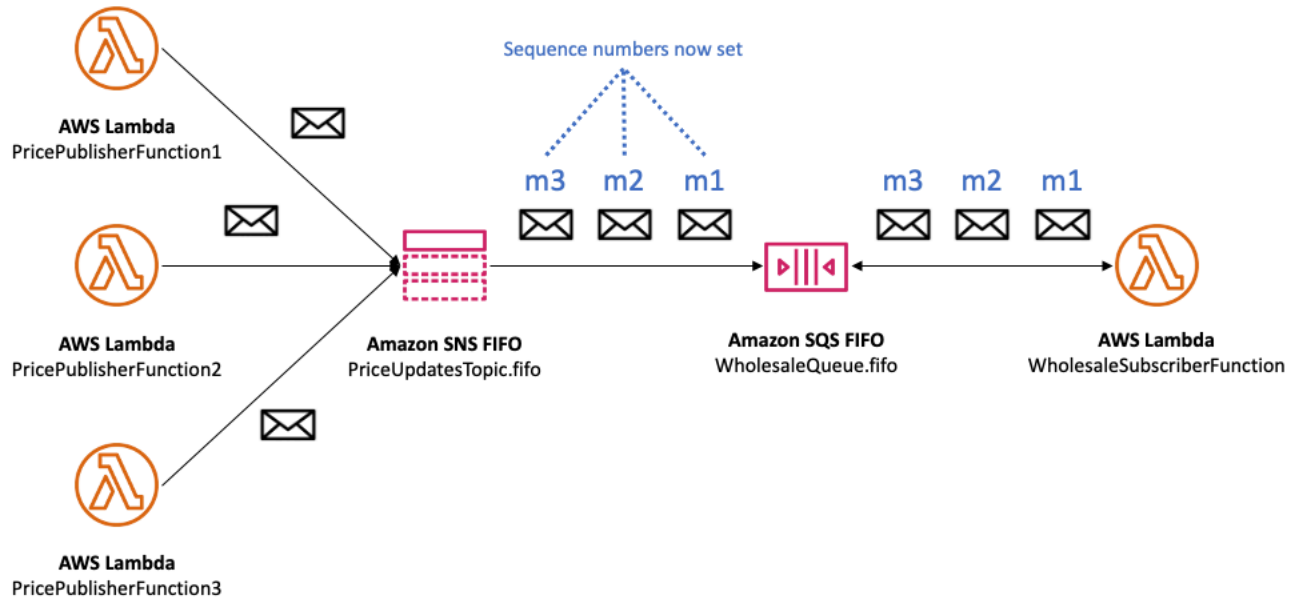


Wenn ein Amazon-SQS-Warteschlangenabonnent nicht erreichbar ist, ist er nicht mehr synchronisiert. Angenommen, der Besitzer der Wholesale-AnwendungsQueue fälschlicherweise ändert dieAmazon-SQS-Warteschlangenrichtlinieauf eine Weise, die verhindert, dass der Amazon-SNS-Serviceprinzipal Nachrichten an die Queue sendet. In diesem Fall schlagen Preisaktualisierungen an die Großhandels-Warteschlange fehl, während die Lieferungen an die Einzelhandels- und die Analyse-Warteschlangen erfolgreich sind, was dazu führt, dass die Abonnenten nicht synchron sind. Wenn der Besitzer der Warteschlange der Großhandelsanwendung die Warteschlangenrichtlinie ändert, setzt Amazon SNS das Zustellen der Nachrichten an die abonnierte Warteschlange fort. Alle Nachrichten, die für das Thema veröffentlicht wurden und auf die falsch konfigurierte Warteschlange abzielen, werden gelöscht, es sei denn, für das Abonnement ist eine Warteschlange für unzustellbare Nachrichten konfiguriert.



Sie können mehrere Anwendungen (oder mehrere Threads innerhalb derselben Anwendung) gleichzeitig Nachrichten in einem SNS-FIFO-Thema veröffentlichen. Wenn Sie dies tun, delegieren Sie die Nachrichtensequenzierung effektiv an den Amazon SNS-Service. Um die etablierte Sequenz von Nachrichten zu bestimmen, können Sie die Sequenznummer überprüfen.

Die Sequenznummer ist eine große, nicht fortlaufende Zahl, die Amazon SNS jeder Nachricht zuweist. Die Länge der Sequenznummer beträgt 128 Bit und nimmt für jede Nachrichtengruppe weiter zu. Die Sequenznummer wird an die abonnierten Amazon-SQS-Warteschlangen als Teil des Nachrichtentextes übergeben. Wenn Sie jedoch Rohnachrichtenzustellung aktivieren, enthält die Nachricht, die an die Amazon-SQS-Warteschlange übermittelt wird, weder die Sequenznummer noch andere Amazon-SNS-Nachrichtenmetadaten.



Amazon SNS FIFO-Themen definieren die Reihenfolge im Kontext einer Nachrichtengruppe. Weitere Informationen finden Sie unter Gruppierung von Amazon SNS SNS-Nachrichten für FIFO-Themen.

Gruppierung von Amazon SNS SNS-Nachrichten für FIFO-Themen

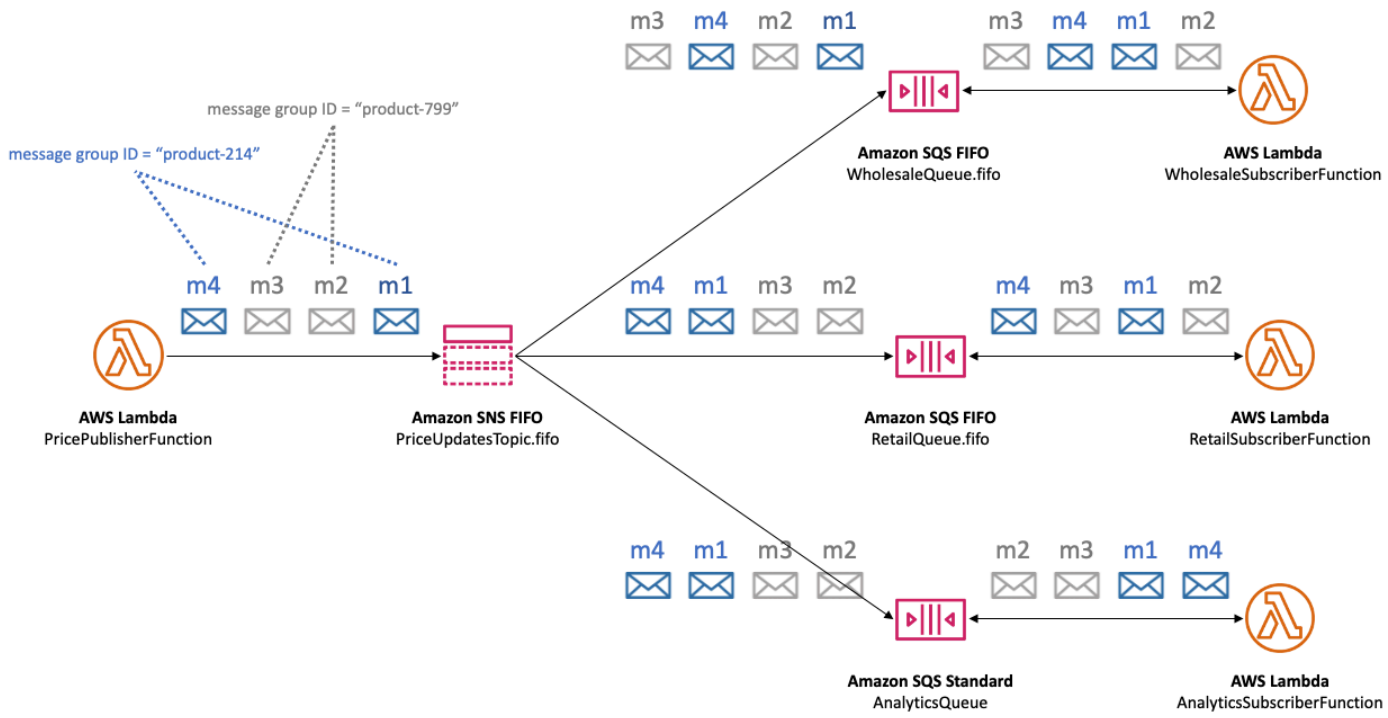
Nachrichten, die derselben Gruppe angehören, werden nacheinander in einer strengen Reihenfolge in Bezug auf die Gruppe verarbeitet.

Wenn Sie Nachrichten in einem Amazon SNS FIFO-Thema veröffentlichen, legen Sie die Nachrichtengruppen-ID fest. Die Gruppen-ID ist ein obligatorisches Token, das angibt, dass eine Nachricht zu einer bestimmten Nachrichtengruppe gehört. Das SNS-FIFO-Thema übergibt die Gruppen-ID an die abonnierten Amazon SQS FIFO-Queues. Die Anzahl der Gruppen IDs in SNS-FIFO-Themen oder SQS-FIFO-Warteschlangen ist unbegrenzt. Die Nachrichtengruppen-ID wird nicht an Amazon-SQS-Standard-Warteschlangen weitergegeben.

Es gibt keine Affinität zwischen einer Nachrichtengruppe und einem Abonnement. Daher werden Nachrichten, die für eine Nachrichtengruppe veröffentlicht werden, an alle abonnierten Queues übermittelt, abhängig von den Filterrichtlinien, die mit Abonnements verknüpft sind. Weitere Informationen erhalten Sie unter Amazon SNS SNS-Nachrichtenzustellung für FIFO-Themen und Amazon SNS SNS-Nachrichtenfilterung für FIFO-Themen.

In Anwendungsbeispiel des Autoteile-Preis-Management gibt es für jedes Produkt, das auf der Plattform verkauft wird, eine dedizierte Nachrichtengruppe. Das gleiche Amazon-SNS-FIFO-Thema wird für die Verarbeitung aller Preisaktualisierungen verwendet. Die Reihenfolge der

Preisaktualisierungen bleibt im Kontext eines einzelnen Autoteile-Produkts erhalten, aber nicht über mehrere Produkte hinweg. In der folgenden Abbildung ist diese Funktionsweise dargestellt. Beachten Sie, dass für das Produkt, dessen Nachrichtengruppen-ID Produkt-214 ist, Nachricht m1 vor m4 verarbeitet wird. Diese Sequenz wird während des gesamten Workflows beibehalten, von Amazon-SNS-FIFO bis zu Amazon-SQS-FIFO. Ebenso wird für das Produkt, dessen Nachrichtengruppen-ID product-799 ist, die Nachricht m2 vor m3 verarbeitet. Bei Verwendung von Amazon SQS SQS-Standardwarteschlangen ist die Nachrichtenreihenfolge jedoch nicht mehr garantiert, und Nachrichtengruppen existieren nicht. Die produkt-214 und produkt-799-Nachrichtengruppen sind unabhängig voneinander, so dass es keine Beziehung zwischen der Sequenzierung ihrer Nachrichten gibt.



Verteilung der Daten nach Nachrichtengruppen IDs zur Verbesserung der Leistung

Um den Zustellungsdurchsatz zu optimieren, stellen Amazon-SNS-FIFO-Themen Nachrichten aus verschiedenen Nachrichtengruppen parallel zu, wobei die Nachrichtenreihenfolge innerhalb der einzelnen Nachrichtengruppen strikt eingehalten wird. Jede einzelne Nachrichtengruppe kann maximal 300 Nachrichten pro Sekunde übermitteln. Um einen hohen Durchsatz für ein einzelnes Thema zu erreichen, sollten Sie daher eine große Anzahl unterschiedlicher Nachrichtengruppen

verwenden IDs. Durch die Verwendung einer Vielzahl von Nachrichtengruppen verteilen Amazon-SNS-FIFO-Themen Nachrichten automatisch auf eine größere Anzahl parallel Partitionen.

> **Note**
>
> Amazon SNS FIFO-Themen sind für die gleichmäßige Verteilung von Nachrichten über Nachrichtengruppen hinweg optimiert IDs, unabhängig von der Anzahl der Gruppen. AWS empfiehlt, eine große Anzahl unterschiedlicher Nachrichtengruppen zu verwenden, um die Leistung IDs zu optimieren.

Wenn Sie in Ihrem Amazon-SNS-FIFO-Thema mit hohem Durchsatz veröffentlichen und mindestens eine Amazon-SQS-FIFO-Warteschlange abonniert ist, wird empfohlen, den hohen Durchsatz für Ihre Warteschlangen zu aktivieren. Weitere Informationen finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen im Entwicklerhandbuch zu Amazon Simple Queue Service.

Amazon SNS SNS-Nachrichtenzustellung für FIFO-Themen

Amazon-SNS-FIFO-Themen (First in, First out) unterstützen die Bereitstellung sowohl an Amazon-SQS-Standard- als auch an FIFO-Warteschlangen, um Kunden Flexibilität und Kontrolle bei der Integration verteilter Anwendungen zu bieten, die Datenkonsistenz nahezu in Echtzeit erfordern.

Für Workloads, die eine strikte Nachrichtenreihenfolge oder Deduplizierung beibehalten müssen, bietet die Kombination von Amazon-SNS-FIFO-Themen mit Amazon-SQS-FIFO-Warteschlangen, die als Zustellungsendpunkt abonniert werden, eine bessere Nachrichtenübermittlung zwischen Anwendungen, wenn die Reihenfolge der Vorgänge und Ereignisse wichtig ist, oder wenn Duplikate nicht toleriert werden können.

Für Workloads, die Bestmögliche Bestellung und at-least-once Lieferung tolerieren, bietet das Abonnieren von Amazon SQS SQS-Standardwarteschlangen für Amazon SNS FIFO-Themen die Möglichkeit, Kosten zu senken und Warteschlangen für Workloads, die kein FIFO verwenden, gemeinsam zu nutzen.

> **Note**
>
> Um Nachrichten von Amazon SNS FIFO-Themen an AWS Lambda Funktionen weiterzuleiten, sind zusätzliche Schritte erforderlich. Abonnieren Sie zunächst Amazon-SQS-FIFO-Warteschlangen für das Thema. Konfigurieren Sie dann die Queues, um die Funktionen

auszulösen. Weitere Informationen finden Sie unter SQS FIFO als Ereignisquelle im AWS Compute Blog.

SNS-FIFO-Themen können keine Nachrichten an vom Kunden verwaltete Endpunkte wie E-Mail-Adressen, mobile Apps, Telefonnummern für SMS oder HTTP(S)-Endpunkte übermitteln. Bei diese Endpunkttypen ist nicht garantiert, dass die strikte Nachrichtenreihenfolge beibehalten wird. Versuche, vom Kunden verwaltete Endpunkte für SNS-FIFO-Themen zu abonnieren, führen zu Fehlern.

SNS-FIFO-Themen unterstützen dieselben Nachrichtenfilterfunktionen wie Standardthemen. Weitere Informationen finden Sie unter Amazon SNS SNS-Nachrichtenfilterung für FIFO-Themenund Vereinfachen Sie Ihr Pub/Sub-Messaging mit Amazon SNS NachrichtenfilterungVeröffentlichen im AWS Compute Blog.

Amazon SNS SNS-Nachrichtenfilterung für FIFO-Themen

Amazon SNS FIFO-Themen unterstützen die Nachrichtenfilterung. Die Verwendung von Nachrichtenfiltern vereinfacht die Architektur, indem die Abonnenten von der Nachrichten-Routing-Logik von den Publisher-Systemen und die Abonnenten von der Nachrichtenfilterlogik befreit werden.

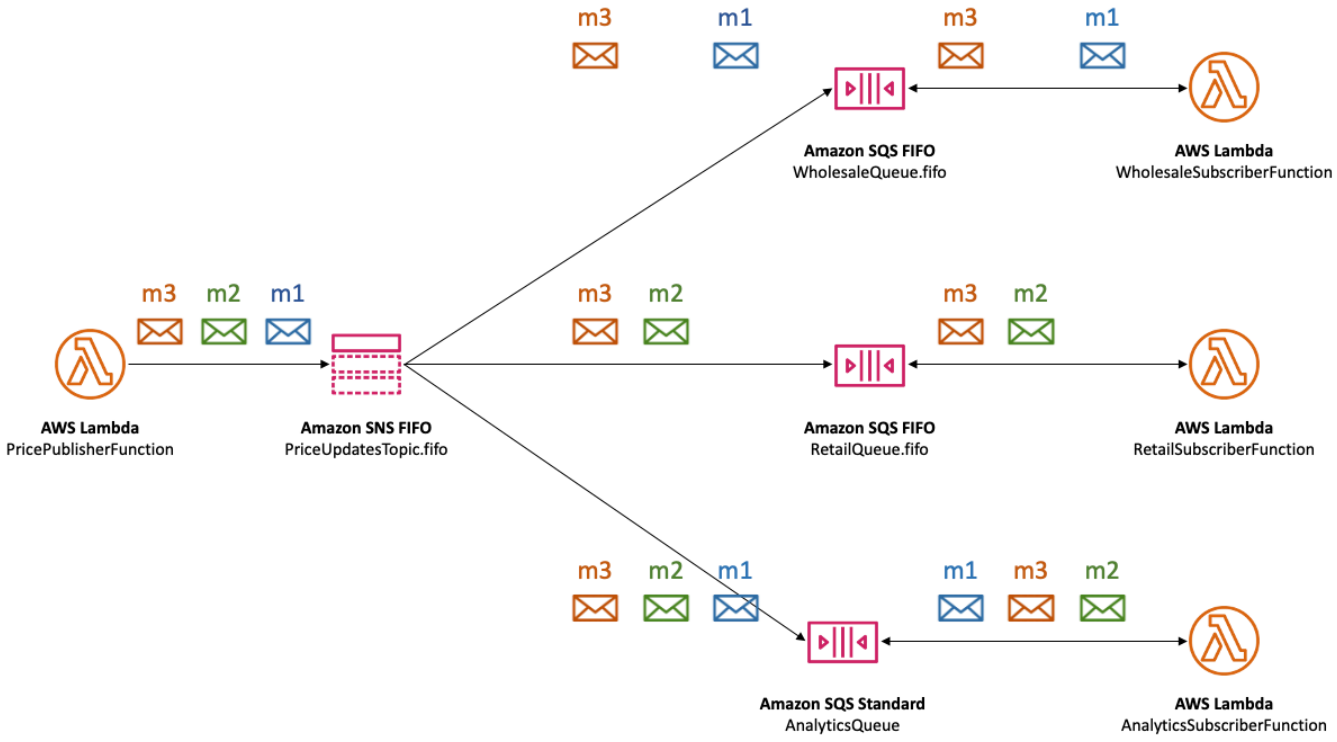
Wenn Sie eine Amazon-SQS-FIFO- oder Standard-Warteschlange für ein SNS-FIFO-Thema abonnieren, können Sie mithilfe der Nachrichtenfilterung angeben, dass der Abonnent anstelle aller Nachrichten eine Teilmenge empfängt. Jeder Abonnent kann seine eigene Filterrichtlinie als Abonnementattributte festlegen. Basierend auf dem Geltungsbereich der Filterrichtlinie wird diese mit den Attributen oder dem Nachrichtentext abgeglichen. Wenn die Filterrichtlinie übereinstimmt, übermittelt das Thema dem Abonnenten eine Kopie der Nachricht. Wenn keine Übereinstimmung vorhanden ist, liefert das Thema keine Kopie der Nachricht.

Im Beispiel für das Preismanagement von Autoteilen wird davon ausgegangen, dass die folgenden Amazon-SNS-Filterrichtlinien festgelegt sind und der Geltungsbereich der Filterrichtlinie `MessageBody` ist:

- Bei der Großhandelswarteschlange stimmt die Filterrichtlinie `{"business":["wholesale"]}` mit jeder Nachricht überein, die einen Schlüssel namens `business` und die Zeichenfolge `wholesale` im Wertesatz enthält. Im folgenden Diagramm ist einer der Schlüssel in der Nachricht m1 `business` mit dem Wert `wholesale`. Einer der Schlüssel in Nachricht m3 ist `business`

mit dem Wert `["wholesale,retail"]`. Daher gilt:m1 und m3 entsprechen den Kriterien der Filterrichtlinie, und beide Nachrichten werden an die Großhandelsqueue übermittelt.

- Für die Einzelhandelswarteschlange stimmt die Filterrichtlinie `{"business":["retail"]}` mit jeder Nachricht überein, die einen Schlüssel namens `business` und die Zeichenfolge `retail` im Wertesatz enthält. Im Diagramm ist einer der Schlüssel in der Nachricht m2 `business` mit dem Wert `retail`. Einer der Schlüssel in Nachricht m3 ist `business` mit dem Wert `["wholesale,retail"]`. Daher gilt:m2 und m3 den Kriterien der Filterrichtlinie entsprechen, und beide Nachrichten werden an die Einzelhandelswarteschlange übermittelt.
- Für die Analyse-Warteschlange möchten wir, dass Amazon Athena alle Datensätze empfängt, so dass keine Filterrichtlinie angewendet wird.



SNS-FIFO-Themen unterstützen eine Vielzahl von übereinstimmenden Operatoren, einschließlich Attributzeichenfolgenwerte, numerische Attributwerte und Attributschlüssel. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

SNS-FIFO-Themen liefern keine doppelten Nachrichten an abonnierte Endpunkte. Weitere Informationen finden Sie unter Amazon SNS SNS-Nachrichtendeduplizierung für FIFO-Themen.

Amazon SNS SNS-Nachrichtendeduplizierung für FIFO-Themen

Amazon SNS FIFO-Themen und Amazon SQS FIFO-Queues unterstützen die Nachrichtendeduplizierung, die eine exakte Nachrichtenübermittlung und -verarbeitung ermöglicht, solange die folgenden Bedingungen erfüllt sind:

- Die abonnierte Amazon-SQS-FIFO-Warteschlange ist vorhanden und verfügt über Berechtigungen, die es dem Amazon-SNS-Service-Prinzipal ermöglichen, Nachrichten an die Warteschlange zu senden.
- Der Nutzer der Amazon-SQS-FIFO-Warteschlange verarbeitet die Nachricht und löscht sie aus der Warteschlange, bevor das Zeitlimit für die Sichtbarkeit abläuft.
- Das Thema Amazon SNS Abonnement“ enthält keine Nachrichtenfilterung. Wenn Sie die Nachrichtenfilterung konfigurieren, unterstützen Amazon SNS FIFO-Themen die at-most-once Zustellung, da Nachrichten auf der Grundlage Ihrer Abonnementfilterrichtlinien herausgefiltert werden können.
- Es gibt keine Netzwerkunterbrechungen, die die Bestätigung der Nachrichtenzustellung verhindern.

> **Note**
>
> Die Nachrichtendeduplizierung gilt für ein gesamtes Amazon SNS FIFO-Thema, wenn das Themenattribut `FifoThroughputScope` auf gesetzt ist. `Topic` Wenn das Themenattribut auf gesetzt `FifoThroughputScope` ist`MessageGroup`, gilt die Nachrichtendeduplizierung für jede einzelne Nachrichtengruppe.

Wenn Sie eine Nachricht in einem Amazon-SNS-FIFO-Thema veröffentlichen, muss die Nachricht eine Deduplizierungskennung enthalten. Diese Kennung ist in der Nachricht enthalten, die das Amazon-SNS-FIFO-Thema an die abonnierten Amazon-SQS-FIFO-Warteschlangen übermittelt.

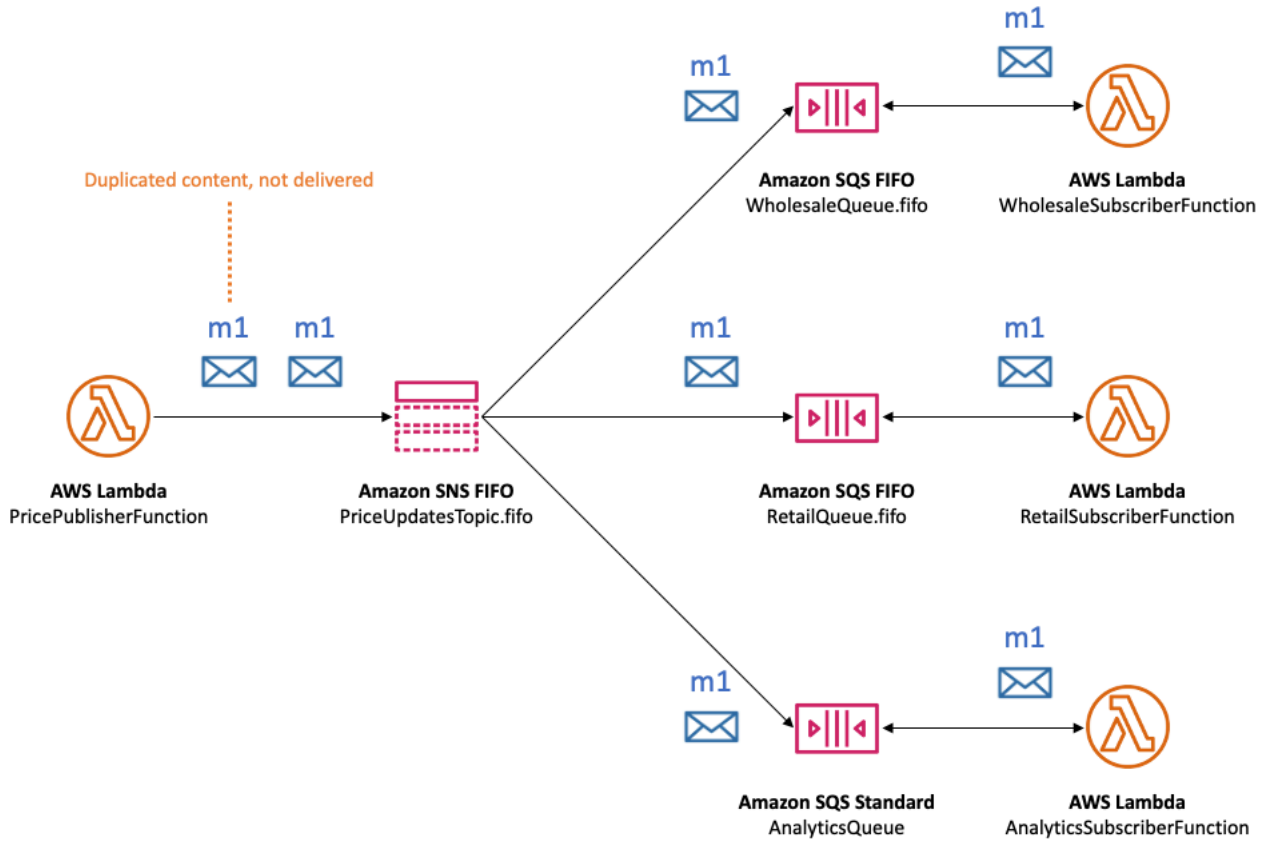
Wenn eine Nachricht mit einer bestimmten Deduplizierungs-ID erfolgreich in einem Amazon SNS FIFO-Thema veröffentlicht wurde, wird jede Nachricht, die innerhalb des Deduplizierungsintervalls von fünf Minuten mit derselben Deduplizierungs-ID veröffentlicht wurde, akzeptiert, aber nicht zugestellt. Das Amazon SNS FIFO-Thema verfolgt weiterhin die Nachrichtendeduplizierungs-ID in dem durch das Topic-Attribut konfigurierten Deduplizierungsbereich`FifoThroughputScope`, auch nachdem die Nachricht an abonnierte Endpunkte zugestellt wurde.

Wenn der Nachrichtentext für jede veröffentlichte Nachricht garantiert eindeutig ist, können Sie die inhaltsbasierte Deduplizierung für ein Amazon SNS FIFO-Thema und die abonnierten Amazon-SQS-FIFO-Warteschlangen aktivieren. Amazon SNS verwendet den Nachrichtentext, um einen eindeutigen Hashwert zu generieren, der als Deduplizierungs-ID für jede Nachricht verwendet wird. Daher müssen Sie beim Senden jeder Nachricht keine Deduplizierungs-ID festlegen.

> **Note**
>
> Nachrichtenattribute sind nicht in der Hashberechnung enthalten.

Wenn die inhaltsbasierte Deduplizierung für ein Amazon SNS FIFO-Thema aktiviert ist und eine Nachricht mit einer Deduplizierungs-ID veröffentlicht wird, überschreibt die veröffentlichte Deduplizierungs-ID die generierte inhaltsbasierte Deduplizierungs-ID.

Im Beispiel des Autoteile-Preismanagement muss das Unternehmen für jede Preisaktualisierung eine universell eindeutige Deduplizierungs-ID festlegen. Dies liegt daran, dass der Nachrichtentext identisch sein kann, selbst wenn sich das Nachrichtenattribut für Groß- und Einzelhandel unterscheidet. Wenn das Unternehmen jedoch den Unternehmenstyp (Groß- oder Einzelhandel) neben der Produkt-ID und dem Produktpreis dem Nachrichtentext hinzugefügt hat, könnte es eine inhaltsbasierte Duplizierung im Amazon-SNS-FIFO-Thema und den abonnierten Amazon-SQS-FIFO-Warteschlangen ermöglichen.



Zusätzlich zur Reihenfolge und Deduplizierung von Nachrichten unterstützen Amazon SNS FIFO-Themen die serverseitige Nachrichtenverschlüsselung (SSE) mit AWS KMS Schlüsseln und den Nachrichtenschutz über VPC-Endpunkte mit. AWS PrivateLink Weitere Informationen finden Sie unter Amazon SNS SNS-Nachrichtensicherheit für FIFO-Themen.

Amazon SNS SNS-Nachrichtensicherheit für FIFO-Themen

Sie können die Verschlüsselung für Amazon SNS SNS-FIFO-Themen und Amazon SQS SQS-FIFO-Warteschlangen mithilfe von AWS Key Management Service (AWS KMS) Kundenhauptschlüsseln () aktivieren. CMKs

- Sie können neue verschlüsselte FIFO-Themen und -Warteschlangen erstellen oder die Verschlüsselung für bestehende aktivieren.
- Nur der Nachrichtentext ist verschlüsselt. Nachrichtenattribute, Ressourcenmetadaten und Ressourcenmetriken bleiben unverschlüsselt.

> **Note**
>
> Durch das Hinzufügen von Verschlüsselung zu einem vorhandenen FIFO-Thema oder einer vorhandenen Queue werden keine nachgelagerten Nachrichten verschlüsselt. Durch das Entfernen der Verschlüsselung aus einem Thema oder einer Queue werden nachgelagerte Nachrichten verschlüsselt.

SNS-FIFO-Themen entschlüsseln die Nachrichten unmittelbar vor der Zustellung an abonnierte Endpunkte. SQS FIFO-Queues entschlüsseln die Nachricht, kurz bevor sie an die Verbraucheranwendung zurückgesendet werden. Weitere Informationen finden Sie unter Amazon SNS SNS-Datenverschlüsselungund Verschlüsseln von auf Amazon SNS veröffentlichten Nachrichten mit AWS KMSVeröffentlichen auf dem AWS Compute Blogsaus.

Darüber hinaus unterstützen SNS FIFO-Themen und SQS FIFO-Queues den Datenschutz bei Schnittstellen-VPC Endpunktepowered by AWS PrivateLink aus. Mithilfe von Schnittstellenendpunkten können Sie Nachrichten von Amazon Virtual Private Cloud (Amazon VPC)-Subnetzen an FIFO-Themen und -Queues senden, ohne das öffentliche Internet zu durchlaufen. Bei diesem Modell bleiben Ihre Nachrichten innerhalb der AWS Infrastruktur und des Netzwerks, wodurch die allgemeine Sicherheit Ihrer Anwendung verbessert wird. Wenn Sie es verwenden AWS PrivateLink, müssen Sie kein Internet-Gateway, keine Network Address Translation (NAT) oder ein virtuelles privates Netzwerk (VPN) einrichten. Weitere Informationen finden Sie unter Sicherung des Amazon SNS SNS-Datenverkehrs mit VPC-Endpunkten und Schutz von auf Amazon SNS veröffentlichten Nachrichten mit AWS PrivateLink im AWS Sicherheitsblog.

SNS-FIFO-Themen unterstützen auch Queues für unzustellbare Nachrichten und Nachrichtenspeicher über Availability Zones hinweg. Weitere Informationen finden Sie unter Dauerhaftigkeit von Amazon SNS SNS-Nachrichten für FIFO-Themen.

Dauerhaftigkeit von Amazon SNS SNS-Nachrichten für FIFO-Themen

Amazon-SNS-FIFO-Themen und Amazon-SQS-Warteschlangen sind dauerhaft. Beide Ressourcentypen speichern Nachrichten redundant über mehrere Availability Zones hinweg und stellen Queues für Dead-Letter bereit, um Ausnahmefälle zu bearbeiten.

In Amazon SNS schlägt die Nachrichtenzustellung fehl, wenn aufgrund eines clientseitigen oder serverseitigen Fehlers nicht auf eine abonnierte Amazon SQS Queue zugreifen kann:

- Client-seitige Fehler treten auf, wenn das Amazon-SNS-FIFO-Thema über veraltete Abonnementmetadaten verfügt. Zwei häufige Fälle von clientseitigen Fehlern treten auf, wenn der Eigentümer der Amazon-SQS-FIFO-Warteschlange eine der folgenden Aktionen ausführt:
  - Es löscht die Queue.
  - Ändert die Queuesrichtlinie so, dass der Amazon SNS Dienstprinzipal daran hindert, Nachrichten an ihn zu senden.

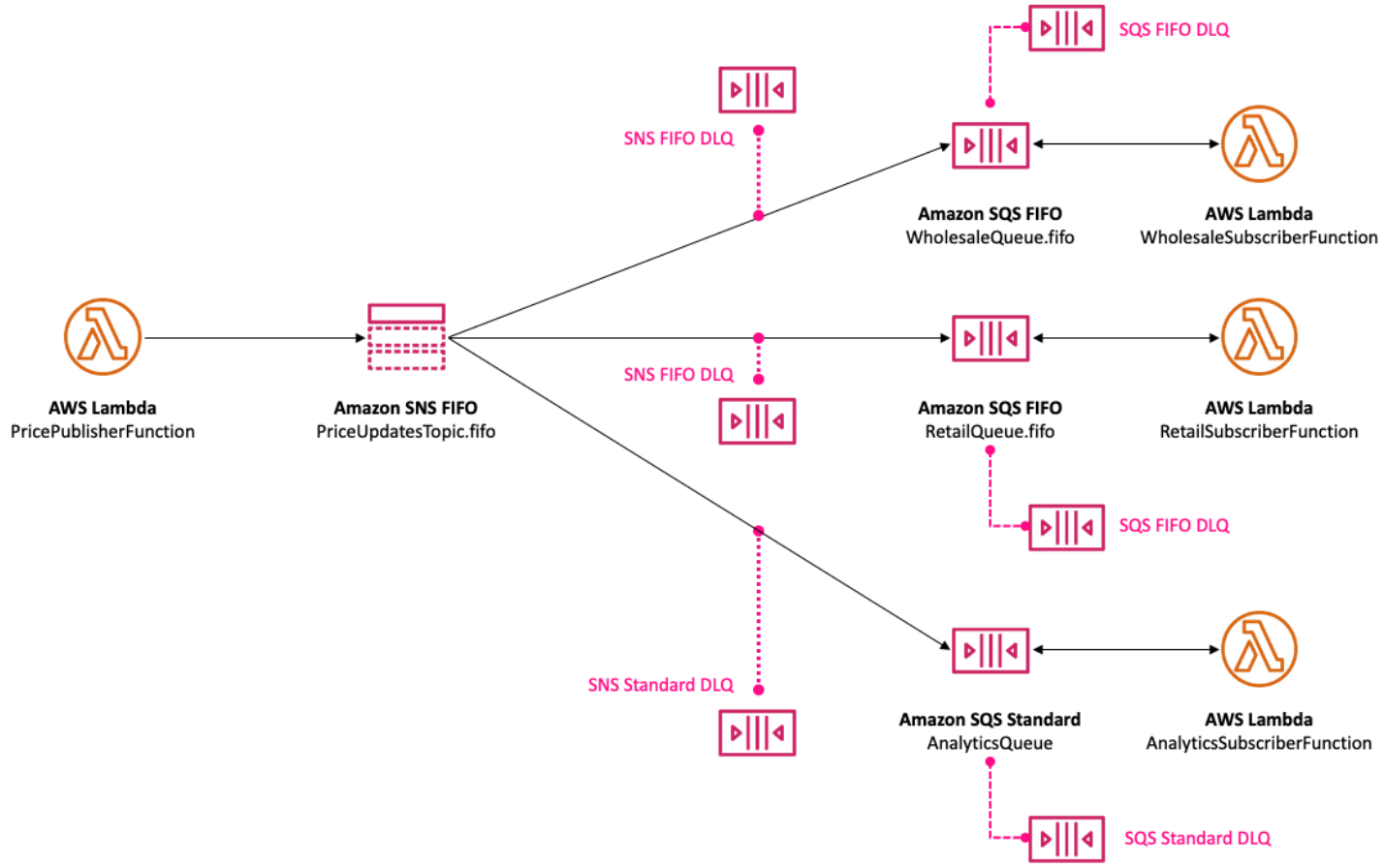
  Amazon SNS versucht nicht erneut, Nachrichten zuzustellen, die aufgrund von clientseitigen Fehlern fehlgeschlagen sind.
- In diesen Situationen können serverseitige Fehler auftreten:
  - Der Amazon SQS Dienst ist nicht verfügbar.
  - Amazon SQS verarbeitet keine gültige Anforderung des Amazon SNS-Service.

  Bei serverseitigen Fehlern versuchen die Amazon-SNS-FIFO-Themen, die fehlgeschlagenen Zustellungen über 23 Tage hinweg bis zu 100.015 mal zu wiederholen. Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

Bei jeder Art von Fehler kann Amazon SNS Nachrichten an Amazon SQS Unzustellbare Nachrichten Queues abstellen, sodass Daten nicht verloren gehen.

In Amazon SQS schlägt die Nachrichtenverarbeitung fehl, wenn die Verbraucheranwendung die Nachricht nicht empfängt, verarbeitet und aus der Queue gelöscht wird. Wenn die maximale Anzahl von Empfangsanforderungen fehlschlägt, kann Amazon SQS Nachrichten an Queues mit Dead-Lettern abstellen, sodass Daten nicht verloren gehen.

Im Beispiel für das Preismanagement von Autoteilen kann das Unternehmen jedem Amazon-SNS-FIFO-Themenabonnement sowie jeder abonnierten Amazon-SQS-Warteschlange eine Amazon-SQS-Warteschlange für unzustellbare Nachrichten (DLQ) zuweisen. Dies schützt das Unternehmen vor jeglichen Preisaktualisierungsverlusten.



Bei der Warteschlange für unzustellbare Nachrichten (DLQ) für ein Amazon-SNS-Abonnement muss es sich um eine Amazon-SQS-Warteschlange handeln, die dem Typ der abonnierenden Warteschlange entspricht. Zum Beispiel muss ein Amazon-SNS-FIFO-Abonnement für eine Amazon-SQS-FIFO-Warteschlange eine Amazon-SQS-FIFO-Warteschlange für unzustellbare Nachrichten haben. Zum Beispiel muss ein Amazon-SNS-FIFO-Abonnement für eine Amazon-SQS-Standard-Warteschlange eine Amazon-SQS-Standard-Warteschlange für unzustellbare Nachrichten haben. Weitere Informationen finden Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe und im AWS Lambda Beitrag Designing durable serverless apps with DLQs for Amazon SNS, Amazon SQS, im Compute-Blog.AWS

Für eine längere Lebensdauer zur Unterstützung der Wiederherstellung nach Ausfällen nachgelagerten Anwendungen können Themenbesitzer auch FIFO-Themen verwenden, um Nachrichten bis zu 365 Tage lang zu archivieren. Themen-Subscriber können diese Nachrichten an einem abonnierten Endpunkt wiederholen, um Nachrichten wiederherzustellen, die auf einen Fehler in einer nachgelagerten Anwendung zurückzuführen sind, oder um den Status einer vorhandenen Anwendung zu replizieren. Weitere Informationen finden Sie unter Archivierung und Wiedergabe von Amazon SNS SNS-Nachrichten für FIFO-Themen.

Archivierung und Wiedergabe von Amazon SNS SNS-Nachrichten für FIFO-Themen

Was ist die Nachrichtenarchivierung und -wiederholung?

Amazon SNS bietet eine Funktion zur Archivierung und Wiedergabe von Nachrichten ohne Code, die speziell für FIFO-Themen (First-In-First-Out) entwickelt wurde. Diese Funktion ermöglicht es den Eigentümern von Themen, Nachrichten bis zu 365 Tage lang direkt im Themenarchiv zu speichern und sie bei Bedarf für Abonnenten erneut abzuspielen. Die Archivierung und Wiedergabe von Nachrichten ist unerlässlich, um verloren gegangene Nachrichten wiederherzustellen und Anwendungen durch die Replikation von Status über Regionen oder Systeme hinweg zu synchronisieren.

Auf diese Funktionalität kann über die AWS API, das SDK und zugegriffen werden. AWS CloudFormation AWS Management Console

Wichtigste Anwendungsfälle

- Nachrichtenwiederherstellung — Stellen Sie Nachrichten wieder her, die aufgrund von Fehlern bei Downstream-Anwendungen verloren gegangen sind, indem Sie sie auf dem Endpunkt des Abonnenten abspielen.
- Statusreplikation — Replizieren Sie den Status eines vorhandenen Systems in einer neuen Umgebung, indem Sie Nachrichten ab einem bestimmten Zeitstempel wiedergeben.
- Fehlerkorrektur — Verpasste Nachrichten bei Ausfällen erneut senden, um sicherzustellen, dass alle Ereignisse korrekt verarbeitet werden.

Komponenten der Nachrichtenarchivierung und -wiedergabe

Verwalten Sie die Archivierung und Wiedergabe von Nachrichten für Amazon SNS FIFO-Themen, einschließlich der Festlegung von Aufbewahrungsfristen, der Überwachung archivierter Nachrichten mithilfe von CloudWatch, der Initiierung von Wiederholungen über Abonnementattribute und der Kenntnis der zum Ändern und Initiieren von Wiederholungen erforderlichen Berechtigungen.

Archivierung von Nachrichten

- Der Eigentümer des Themas aktiviert die Archivierungsfunktion und legt den Aufbewahrungszeitraum für Nachrichten fest, der bis zu 365 Tage betragen kann. Weitere

Informationen finden Sie unter Amazon SNS SNS-Nachrichtenarchivierung für Inhaber von FIFO-Themen

- CloudWatch Metriken helfen bei der Überwachung der archivierten Nachrichten.

Wiedergabe von Nachrichten

- Ein Abonnent initiiert eine Wiederholung und wählt dabei das Zeitfenster aus, in dem die Nachrichten an den abonnierten Endpunkt erneut verarbeitet werden sollen. Weitere Informationen finden Sie unter Wiederholung von Amazon SNS SNS-Nachrichten für Abonnenten zum Thema FIFO.
- Mithilfe der Funktion verwalten Sie die Wiedergabe mithilfe von Abonnementattributen. `ReplayPolicy`

Relevante Berechtigungen

- **`SetSubscriptionAttributes`**— Erforderlich, um die Wiedergabeeinstellungen mithilfe des `ReplayPolicy` Attributs in einem Abonnement zu konfigurieren oder zu ändern.
- **`Subscribe`**— Erforderlich, um ein neues Abonnement anzuhängen und Wiederholungen zu starten.
- **`GetTopicAttributes`**— Ermöglicht das Anzeigen der Eigenschaften des Themas, aber bei der Initiierung von Wiederholungen geht es hauptsächlich um die Abonnementverwaltung.

Amazon SNS SNS-Nachrichtenarchivierung für Inhaber von FIFO-Themen

Mit der Nachrichtenarchivierung können Sie eine einzige Kopie aller zu Ihrem Thema veröffentlichten Nachrichten archivieren. Sie können veröffentlichte Nachrichten in Ihrem Thema speichern, indem Sie die Nachrichtenarchivierungsrichtlinie für das Thema aktivieren, wodurch die Nachrichtenarchivierung für alle Abonnements aktiviert wird, die mit diesem Thema verknüpft sind. Nachrichten können für mindestens einen Tag bis maximal 365 Tage archiviert werden.

Bei der Festlegung einer Archivrichtlinie fallen zusätzliche Gebühren an. Informationen zu den Preisen finden Sie unter Amazon SNS – Preise.

Erstellen Sie eine Nachrichtenarchivierungsrichtlinie mit dem AWS Management Console

Verwenden Sie diese Option zum Erstellen einer neuen Nachrichtenarchivrichtlinie mit AWS Management Console.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Thema aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Themen finden Sie unter Erstellen eines Amazon SNS-Themas.

   > **Note**
   >
   > Die Archivierung und Wiedergabe von Amazon SNS SNS-Nachrichten ist nur für application-to-application (A2A) FIFO-Themen verfügbar.

3. Erweitern Sie auf der Seite Edit topic (Thema bearbeiten) den Abschnitt Archive policy (Archivrichtlinie) .
4. Aktivieren Sie die Funktion Archive policy (Archivrichtlinie) und geben Sie im Thema die Anzahl der Tage ein, für die Sie Nachrichten archivieren möchten.
5. Wählen Sie Änderungen speichern.

Anzeigen, Bearbeiten und Deaktivieren einer Themenrichtlinie für die Nachrichtenarchivierung

- Auf der Seite Topic details (Themendetails) zeigt die Retention policy (Aufbewahrungsrichtlinie) den Status der Archivrichtlinie an, einschließlich der Anzahl der Tage, für die sie festgelegt ist. Wählen Sie die Registerkarte Archive policy (Archivrichtlinie), um die folgenden Details zum Nachrichtenarchiv anzuzeigen:
  - Status — Der Archivierungs- und Wiederholungsstatus wird als active (aktiv) angezeigt, wenn eine Archivrichtlinie angewendet wird. Der Archivierungs- und Wiederholungsstatus wird als inactive (inaktiv) angezeigt, wenn die Archivrichtlinie auf ein leeres JSON-Objekt festgelegt ist.
  - Message retention period (Aufbewahrungszeitraum für Nachrichten) – Die angegebene Anzahl von Tagen für die Aufbewahrung von Nachrichten.
  - Archive start date (Startdatum des Archivs) – Das Datum, ab dem Suscriber Nachrichten wiederholen können.
  - JSON preview (JSON-Vorschau) – Die JSON-Vorschau der Archivrichtlinie.

- (Optional) Um eine Archivrichtlinie zu bearbeiten, wechseln Sie zur Seite mit der Themenzusammenfassung und wählen Sie Edit (Bearbeiten) aus.
- (Optional) Um eine Archivrichtlinie zu deaktivieren, wechseln Sie zur Seite mit der Themenzusammenfassung und wählen Sie Edit (Bearbeiten) aus. Deaktivieren Sie die Archive Policy (Archivrichtlinie) und wählen Sie Save changes (Änderungen speichern).
- (Optional) Um ein Thema mit einer Archivrichtlinie zu löschen, müssen Sie zuerst die Archivrichtlinie wie zuvor beschrieben deaktivieren.

> **Important**
>
> Um das versehentliche Löschen von Nachrichten zu vermeiden, ist es nicht möglich, ein Thema mit einer aktiven Nachrichtenarchivierungsrichtlinie zu löschen. Die Nachrichtenarchivierungsrichtlinie des Themas muss deaktiviert werden, bevor das Thema gelöscht werden kann. Wenn Sie eine Nachrichtenarchivierungsrichtlinie deaktivieren, löscht Amazon SNS alle archivierten Nachrichten. Beim Löschen eines Themas werden Abonnements entfernt und alle Nachrichten, die gerade übertragen werden, können möglicherweise nicht zugestellt werden.

Eine Nachrichtenarchivierungsrichtlinie mit der API erstellen

Um mithilfe der API eine Nachrichtenarchivierungsrichtlinie zu erstellen, müssen Sie das Attribut `ArchivePolicy` zu Ihrem Thema hinzufügen. Sie können `ArchivePolicy` mithilfe der API-Aktionen `CreateTopic` und `SetTopicAttributes` festlegen. `ArchivePolicy` hat einen einzigen Wert, `MessageRetentionPeriod`, der die Anzahl der Tage darstellt, an denen Amazon SNS Nachrichten aufbewahrt. Um die Nachrichtenarchivierung für Ihr Thema zu aktivieren, legen Sie `MessageRetentionPeriod` auf einen ganzzahligen Wert größer als Null fest. Um beispielsweise Nachrichten 30 Tage lang in Ihrem Archiv aufzubewahren, legen Sie für `ArchivePolicy` Folgendes fest:

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

Um die Nachrichtenarchivierung für Ihr Thema zu deaktivieren und das Archiv zu löschen, deaktivieren Sie `ArchivePolicy` wie folgt:

```
{}
```

Eine Nachrichtenarchivierungsrichtlinie mit dem SDK erstellen

Um ein AWS SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Geteilte `credentials` Dateien `config` und Dateien im AWS SDKs Referenzhandbuch zu Tools.

Im folgenden Codebeispiel wird veranschaulicht, wie Sie `ArchivePolicy` für ein Amazon SNS-Thema festlegen, damit alle Nachrichten, die zum Thema veröffentlicht wurden, 30 Tage lang aufbewahrt werden.

```
// 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);
```

Erstellen Sie eine Nachrichtenarchivierungsrichtlinie mit AWS CloudFormation

Informationen zum Erstellen einer Archivrichtlinie AWS CloudFormation finden Sie unter `AWS::SNS::Topic`im AWS CloudFormation Benutzerhandbuch.

Zugriff auf ein verschlüsseltes Archiv gewähren

Bevor ein Suscriber mit der Wiederholung von Nachrichten aus einem verschlüsselten Thema beginnen kann, müssen Sie die folgenden Schritte ausführen. Da frühere Nachrichten wiederholt werden, muss Amazon SNS `Decrypt`-Zugriff auf den KMS-Schlüssel erhalten, der zur Verschlüsselung der Nachrichten im Archiv verwendet wurde.

1. Wenn Sie Nachrichten mit einem KMS-Schlüssel verschlüsseln und innerhalb des Themas speichern, müssen Sie Amazon SNS die Möglichkeit geben, diese Nachrichten über die

Schlüsselrichtlinie zu entschlüsseln. Weitere Informationen finden Sie unter Amazon SNS Entschlüsselungsberechtigungen gewähren.

2. AWS KMS Für Amazon SNS aktivieren. Weitere Informationen finden Sie unter Berechtigungen konfigurieren AWS KMS.

> **Important**
>
> Ändern Sie beim Hinzufügen der neuen Abschnitte zur KMS-Schlüsselrichtlinie keinen der vorhandenen Abschnitte. Wenn die Verschlüsselung für ein Thema aktiviert ist und der KMS-Schlüssel deaktiviert oder gelöscht wird oder die KMS-Schlüsselrichtlinie nicht korrekt für Amazon SNS konfiguriert ist, kann Amazon SNS keine Nachrichten für Ihre Subscriber wiederholen.

Amazon SNS Entschlüsselungsberechtigungen gewähren

Damit Amazon SNS auf verschlüsselte Nachrichten aus dem Archiv Ihres Themas zugreifen und sie auf abonnierten Endpunkten archivieren und wiederholen kann, müssen Sie das Amazon SNS-Serviceprinzip aktivieren, um diese Nachrichten zu entschlüsseln.

Im Folgenden finden Sie eine Beispielrichtlinie, die erforderlich ist, damit der Prinzipal des Amazon SNS gespeicherte Nachrichten während einer Wiederholung von historischen Nachrichten aus Ihrem Thema entschlüsseln kann.

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

Überwachen Sie Messdaten zur Nachrichtenarchivierung mit Amazon CloudWatch

Sie können archivierte Nachrichten mithilfe von Amazon CloudWatch anhand der folgenden Metriken überwachen. Um über Anomalien in Ihren Workloads informiert zu werden und Auswirkungen zu vermeiden, können Sie CloudWatch Amazon-Alarme für diese Metriken konfigurieren. Weitere Details finden Sie unter Protokollierung und Überwachung in Amazon SNS.

Metrik	Beschreibung
ApproximateNumberOfMessagesArchived	Stellt dem Eigentümer des Themas die Gesamtanzahl der im Themenarchiv archivierten Nachrichten mit einer Auflösung von 60 Minuten zur Verfügung.
ApproximateNumberOfBytesArchived	Stellt dem Eigentümer des Themas die Gesamtanzahl der archivierten Bytes für alle Nachrichten im Themenarchiv mit einer Auflösung von 60 Minuten zur Verfügung.
NumberOfMessagesArchiveProcessing	Nennt dem Eigentümer des Themas die Anzahl der Nachrichten, die während des Intervalls im Themenarchiv gespeichert wurden, mit einer Auflösung von 1 Minute.
NumberOfBytesArchiveProcessing	Nennt dem Eigentümer des Themas die Gesamtzahl der Bytes, die während des Intervalls im Themenarchiv gespeichert wurden, mit einer Auflösung von 1 Minute.

Die `GetTopicAttributes`-API verfügt über eine `BeginningArchiveTime`-Eigenschaft, die den ältesten Zeitstempel darstellt, zu dem ein Subscriber eine Wiederholung starten kann. Im Folgenden finden Sie eine Beispielantwort für diese API-Aktion:

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

```
  ...
}
```

Wiederholung von Amazon SNS SNS-Nachrichten für Abonnenten zum Thema FIFO

Amazon SNS Replay ermöglicht es Themenabonnenten, archivierte Nachrichten aus dem Themendatenspeicher abzurufen und erneut an einen abonnierten Endpunkt zu senden.

- Nachrichten können sofort nach der Erstellung des Abonnements wiedergegeben werden.
- Eine wiedergegebene Nachricht behält den gleichen Inhalt,`MessageId`, `Timestamp` wie das Original.
- Die Nachricht enthält ein `Replayed` Attribut, das angibt, dass es sich um eine wiedergegebene Nachricht handelt.
- Um nur bestimmte Nachrichten wiederzugeben, wenden Sie eine Filterrichtlinie auf Ihr Abonnement an.

Weitere Informationen zum Filtern von Nachrichten finden Sie unter Filtert wiederholte Nachrichten.

Erstellen Sie eine Richtlinie für die Nachrichtenwiedergabe mit dem AWS Management Console

Verwenden Sie diese Option zum Erstellen einer neuen Wiederholungsrichtlinie mit AWS Management Console.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Themenabonnement aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Abonnements finden Sie unter Ein Abonnement für ein Amazon SNS SNS-Thema erstellen.
3. Um die Nachrichtenwiederholung zu starten, rufen Sie das Dropdown-Menü Replay (Wiederholung) auf und wählen Sie Start replay (Wiederholung starten).
4. Treffen Sie im Modal Replay timeframe (Zeitrahmen für Wiederholung) die folgenden Auswahlen:

    a. Startdatum und Uhrzeit der Wiedergabe wählen — Wählen Sie das Datum (YYYY/MM/DDFormat) und die Uhrzeit (24-Stunden-Format hh:mm:ss) aus, ab dem Sie mit der Wiedergabe archivierter Nachrichten beginnen möchten. Die Startzeit muss nach Beginn der ungefähren Archivierungszeit liegen.

   b. (Optional) Wählen Sie Datum und Uhrzeit für das Ende der Wiedergabe — Wählen Sie das Datum (YYYY/MM/DDFormat) und die Uhrzeit (24-Stunden-Format hh:mm:ss) aus, zu der Sie die Wiedergabe archivierter Nachrichten beenden möchten.

   c. Wählen Sie Start replay „Wiederholung starten“.

5. (Optional) Um die Wiedergabe einer Nachricht zu beenden, rufen Sie die Seite mit den Subscription details (Abonnementdetails) auf und wählen Sie im Dropdown-Menü Replay (Wiederholung) die Option Stop replay (Wiederholung beenden) aus.

6. (Optional) Informationen zur Überwachung der Nachrichtenwiedergabe-Metriken innerhalb dieses Workflows mithilfe von finden Sie unter. CloudWatch Überwachen Sie die Messdaten zur Nachrichtenwiedergabe mit Amazon CloudWatch

Informationen zum Anzeigen und Bearbeiten einer Nachrichtenwiederholungsrichtlinie

Auf der Seite Subscription details (Abonnementdetails) können Sie die folgenden Aktionen ausführen:

- Um den Wiederholungsstatus der Nachricht anzuzeigen, werden im Feld Replay status (Wiederholungsstatus) die folgenden Werte angezeigt:
  - Completed (Abgeschlossen) – Die Wiederholung hat alle Nachrichten erfolgreich erneut zugestellt und stellt jetzt neu veröffentlichte Nachrichten zu.
  - In progress (In Bearbeitung) – Die Wiederholung wiederholt derzeit die ausgewählten Nachrichten.
  - Failed (Fehlgeschlagen) – Die Wiederholung konnte nicht abgeschlossen werden.
  - Pending (Ausstehend) – Der Standardstatus, während die Wiederholung gestartet wird.
- (Optional) Um eine Nachrichtenwiederholungsrichtlinie zu ändern, rufen Sie die Seite Subscription details (Abonnementdetails) auf und wählen Sie im Dropdown-Menü Replay (Wiederholung) die Option Start replay (Wiederholung starten) aus. Durch das Starten einer Wiederholung wird die bestehende Wiederholung ersetzt.

Dem Abonnement mithilfe der API eine Wiederholungsrichtlinie hinzufügen

Verwenden Sie das Attribut `ReplayPolicy`, um archivierte Nachrichten zu wiederholen. `ReplayPolicy` kann mit den API-Aktionen `Subscribe` und `SetSubscriptionAttributes` verwendet werden. Diese Richtlinie hat folgende Werte:

- **`StartingPoint`** (Obligatorisch) – Zeigt an, ab welchem Punkt die Wiederholung von Nachrichten gestartet werden soll.
- **`EndingPoint`**(Optional) – Zeigt an, wann die Nachrichtenwiederholung beendet werden soll. Wenn `EndingPoint` weggelassen wird, wird die Wiederholung fortgesetzt, bis die aktuelle Uhrzeit erreicht ist.
- **`PointType`** (Obligatorisch) – Legt den Typ der Start- und Endpunkte fest. Derzeit wird `Timestamp` als einziger Wert für `PointType` unterstützt.

Um beispielsweise nach dem Ausfall einer nachgeschalteten Anwendung eine Wiederherstellung durchzuführen und alle Nachrichten für einen Zeitraum von zwei Stunden am 1. Oktober 2023 erneut zu senden, verwenden Sie die API-Aktion `SetSubscriptionAttributes`, um wie folgt ein `ReplayPolicy` festzulegen:

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

Um alle Nachrichten, die am 1. Oktober 2023 an das Thema gesendet wurden, zu wiederholen und weiterhin alle neu veröffentlichten Nachrichten zu Ihrem Thema zu erhalten, verwenden Sie die API-Aktion `SetSubscriptionAttributes`, um wie folgt ein `ReplayPolicy` für Ihr Abonnement festzulegen:

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

Um zu überprüfen, ob eine Nachricht wiederholt wurde, wird jeder wiedergegebenen Nachricht das boolesche Attribut `Replayed` hinzugefügt.

Dem Abonnement mithilfe des SDK eine Wiederholungsrichtlinie hinzufügen

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Geteilte `credentials` Dateien `config` und Dateien im AWS SDKs Referenzhandbuch zu Tools.

Das folgende Codebeispiel zeigt, wie Sie `ReplayPolicy` bei einem Abonnement festlegen, damit Nachrichten aus dem Archiv des Amazon SNS FIFO-Themas für ein 2-stündiges Zeitfenster am 1. Oktober 2023 erneut zugestellt werden.

```
// 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);
```

Verstehen der EndingPoint

Wenn Sie ein `ReplayPolicy` Abonnement für Amazon SNS beantragen, ist der `EndingPoint` Wert optional. Wenn kein `EndingPoint` Wert angegeben wird, beginnt die Wiedergabe an der angegebenen Uhrzeit `StartingPoint` und wird fortgesetzt, bis die aktuelle Uhrzeit erreicht ist, einschließlich der Verarbeitung neu veröffentlichter Nachrichten. Sobald das Abonnement abgeschlossen ist, funktioniert es wie ein reguläres Abonnement, bei dem neue Nachrichten empfangen werden, sobald sie veröffentlicht werden.

Wenn ein angegeben `EndingPoint` ist, spielt der Dienst Nachrichten von `StartingPoint` bis zu erneut ab `EndingPoint` und stoppt dann. Durch diese Aktion wird das Abonnement effektiv angehalten. Solange das Abonnement pausiert ist, werden neu veröffentlichte Nachrichten nicht an den abonnierten Endpunkt zugestellt.

Um die Nachrichtenzustellung wieder aufzunehmen, wenden Sie eine neue an, `ReplayPolicy` ohne eine anzugeben`EndingPoint`, und legen Sie `StartingPoint` den gewünschten Zeitpunkt fest, ab dem weiterhin Nachrichten empfangen werden sollen. Um beispielsweise ein Abonnement an der Stelle fortzusetzen, an der eine vorherige Wiedergabe beendet wurde, setzen Sie das neue Abonnement `StartingPoint` auf den zuvor bereitgestellten `EndingPoint` Wert.

Filtert wiederholte Nachrichten

Mit der Amazon SNS-Nachrichtenfilterung können Sie steuern, welche Nachrichten die Amazon SNS an Ihrem Abonnentenendpunkt wiederholt. Wenn sowohl Nachrichtenfilterung als auch Nachrichtenarchivierung aktiviert sind, ruft Amazon SNS die Nachricht zunächst aus dem Datenspeicher des Themas ab und ordnet sie dann dem `FilterPolicy` des Abonnements zu. Die Nachricht wird an den abonnierten Endpunkt übermittelt, wenn eine Übereinstimmung vorliegt, andernfalls wird die Nachricht herausgefiltert. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.

Überwachen Sie die Messdaten zur Nachrichtenwiedergabe mit Amazon CloudWatch

Sie können Replay-Nachrichten mithilfe von Amazon CloudWatch anhand der folgenden Metriken überwachen. Um über Anomalien in Ihren Workloads informiert zu werden und Auswirkungen zu vermeiden, können Sie CloudWatch Amazon-Alarme für diese Metriken konfigurieren. Weitere Details finden Sie unter Protokollierung und Überwachung in Amazon SNS.

Metrik	Beschreibung
NumberOfReplayedNotificationsDelivered	Nennt dem Suscriber die Gesamtanzahl der aus dem Themenarchiv wiederholten Nachrichten mit einer Auflösung von 1 Minute.
NumberOfReplayedNotificationsFailed	Nennt dem Suscriber die Gesamtanzahl der aus dem wiederholten Nachrichten mit einer Auflösung von 1 Minute, die nicht aus dem Themenarchiv zugestellt werden konnten.

Amazon SNS SNS-Codebeispiele für FIFO-Themen

Verwenden Sie die folgenden Codebeispiele, um den Beispielanwendungsfall für das Preismanagement von Autoteilen in ein Amazon SNS SNS-FIFO-Thema und entweder eine Amazon SQS SQS-FIFO-Warteschlange oder eine Standardwarteschlange zu integrieren.

Verwenden eines SDK AWS

Mithilfe eines AWS SDK erstellen Sie ein Amazon SNS FIFO-Thema, indem Sie sein `FifoTopic` Attribut auf setzen. **`true`** Sie erstellen eine Amazon-SQS-FIFO-Warteschlange, indem Sie ihr

`FifoQueue`-Attribut auf **`true`** setzen. Außerdem müssen Sie das Suffix **`.fifo`**zum Namen jeder FIFO-Ressource hinzufügen. Nachdem Sie ein FIFO-Thema oder eine Queue erstellt haben, können Sie es nicht in ein Standardthema oder eine Standardqueue konvertieren.

In den folgenden Codebeispielen werden diese FIFO- und Standard-Warteschlangenressourcen erstellt:

- Das Amazon-SNS-FIFO-Thema, das die Preisaktualisierungen verteilt
- Die Amazon-SQS-FIFO-Warteschlangen, die diese Updates für die Groß- und Einzelhandel-Anwendung bereitstellen
- Die Amazon-SQS-Standardwarteschlange für die Analyseanwendung, in der Datensätze gespeichert werden, die für Business Intelligence (BI) abgefragt werden können
- Die Amazon-SNS-FIFO-Abonnements, die beide Warteschlangen mit dem Thema verbinden

In diesem Beispiel werden Filterrichtlinien in den Abonnements eingestellt. Wenn Sie das Beispiel überprüfen, indem Sie eine Nachricht im -Thema veröffentlichen, stellen Sie sicher, dass Sie die Nachricht mit dem `business`-Attribut veröffentlichen. Geben Sie entweder `retail`oder `wholesale`als Attributwert an. Andernfalls wird die Nachricht herausgefiltert und nicht an die abonnierten Queues übermittelt. Weitere Informationen finden Sie unter Amazon SNS SNS-Nachrichtenfilterung für FIFO-Themen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
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",
                "FifoThroughputScope", "MessageGroup");

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

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

            return topicArn;

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

```
    }

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

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

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

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

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

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

```
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

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

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

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
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),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            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
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " 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.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Empfangen von Nachrichten von FIFO-Abonnements

Sie können jetzt Preisaktualisierungen in den drei abonnierten Anwendungen erhalten. Wie in der gezeigtthe section called “Anwendungsfall zum FIFO-Thema”, ist der Eingangspunkt für jede Verbraucheranwendung die Amazon SQS SQS-Warteschlange, die von der entsprechenden AWS Lambda Funktion automatisch abgerufen werden kann. Wenn eine Amazon-SQS-Warteschlange eine Ereignisquelle für eine Lambda-Funktion ist, skaliert Lambda seine Pollerflotte nach Bedarf, um Nachrichten effizient zu verbrauchen.

Weitere Informationen finden Sie unter Using AWS Lambda with Amazon SQS im AWS Lambda Developer Guide. Informationen zum Schreiben eigener Warteschlangenabfragen finden Sie unter Empfehlungen für Amazon SQS SQS-Standard- und FIFO-Warteschlangen im Amazon Simple Queue Service Developer Guide und ReceiveMessagein der Amazon Simple Queue Service API-Referenz.

Verwenden AWS CloudFormation

AWS CloudFormation ermöglicht es Ihnen, eine Vorlagendatei zu verwenden, um eine Sammlung von AWS Ressourcen als eine Einheit zu erstellen und zu konfigurieren. Dieser Abschnitt enthält eine Beispieltemplate, das Folgendes erstellt:

- Das Amazon-SNS-FIFO-Thema, das die Preisaktualisierungen verteilt
- Die Amazon-SQS-FIFO-Warteschlangen, die diese Updates für die Groß- und Einzelhandel-Anwendung bereitstellen
- Die Amazon-SQS-Standardwarteschlange für die Analyseanwendung, in der Datensätze gespeichert werden, die für Business Intelligence (BI) abgefragt werden können
- Die Amazon-SNS-FIFO-Abonnements, die beide Warteschlangen mit dem Thema verbinden
- Filterrichtlinien, die angeben, dass Abonnentenanwendungen nur die Preisaktualisierungen erhalten, die sie benötigen

> **Note**
>
> Wenn Sie dieses Codebeispiel testen, indem Sie eine Nachricht im Thema veröffentlichen, stellen Sie sicher, dass Sie die Nachricht mit dem `business`-Attribut veröffentlichen. Geben Sie entweder `retail`oder `wholesale`als Attributwert an. Andernfalls wird die Nachricht herausgefiltert und nicht an die abonnierten Queues übermittelt.

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

Weitere Informationen zum Bereitstellen von AWS Ressourcen mithilfe einer AWS CloudFormation Vorlage finden Sie unter Erste Schritte im AWS CloudFormation Benutzerhandbuch.

Amazon SNS Nachrichtenfilterung

Standardmäßig erhalten Abonnenten von Amazon-SNS-Themen jede Nachricht, die zum Thema veröffentlicht wird. Um eine Teilmenge der Nachrichten zu erhalten, müssen Abonnenten dem Themen-Abonnement eine Filterrichtlinie zuweisen.

Eine Filterrichtlinie ist ein JSON-Objekt mit Eigenschaften, die definieren, welche Nachrichten der Abonnent empfängt. Amazon SNS unterstützt Richtlinien, die sich auf die Nachrichtenattribute oder den Nachrichtentext auswirken, je nach dem Geltungsbereich der Filterrichtlinie, die Sie für das Abonnement festgelegt haben. Bei den Filterrichtlinien für den Nachrichtentext wird davon ausgegangen, dass es sich bei der Nachrichtennutzlast um ein wohlgeformtes JSON-Objekt handelt.

Wenn ein Abonnement keine Filterrichtlinie besitzt, erhält der Abonnement jede Nachricht, die für sein Thema veröffentlicht wird. Wenn Sie eine Nachricht für einem Thema mit aktiver Filterrichtlinie veröffentlichen, vergleicht Amazon SNS die Nachrichtenattribute oder den Nachrichtentext mit den Eigenschaften in den Filterrichtlinien für alle Abonnements des Themas. Wenn alle Nachrichtenattribute oder Eigenschaften des Nachrichtentexts die in der Filterrichtlinie angegebenen Bedingungen erfüllen, sendet Amazon SNS die Nachricht an den Abonnenten. Andernfalls sendet Amazon SNS die Nachricht nicht an diesen Abonnenten.

Weitere Informationen finden Sie unter Filtern von Nachrichten, die in für Themen veröffentlicht sind..

Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich

Mit dem `FilterPolicyScope` Abonnementattribut können Sie den Filterbereich definieren, indem Sie einen der folgenden Werte festlegen:

- `MessageAttributes`— Wendet die Filterrichtlinie auf Nachrichtenattribute an (Standardeinstellung).
- `MessageBody`— Wendet die Filterrichtlinie auf den Nachrichtentext an.

> **Note**
>
> Wenn für eine bestehende Filterrichtlinie kein Geltungsbereich für die Filterrichtlinie definiert ist, wird der Geltungsbereich standardmäßig als `MessageAttributes` definiert.

Filterrichtlinien für Amazon-SNS-Abonnements

Mit einer Abonnementfilterrichtlinie können Sie die Eigenschaftsnamen angeben und die einzelnen Eigenschaftsnamen einer Liste von Werten zuweisen. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

Wenn Amazon-SNS-Nachrichtenattribute oder Eigenschaften des Nachrichtentexts auf der Grundlage der Abonnementfilterrichtlinie auswertet, werden die nicht in der Richtlinie angegebenen ignoriert.

> **Important**
>
> AWS Dienste wie IAM und Amazon SNS verwenden ein verteiltes Rechenmodell, das als Eventual Consistency bezeichnet wird. Hinzufügungen oder Änderungen einer Abonnementfilterrichtlinie dauern bis zu 15 Minuten, bis sie vollständig wirksam werden.

Ein Abonnement akzeptiert eine Nachricht unter den folgenden Bedingungen:

- Wenn der Geltungsbereich der Filterrichtlinie auf `MessageAttributes` festgelegt ist, stimmt nicht jeder Eigenschaftsname in der Filterrichtlinie mit dem Namen eines Nachrichtenattributs überein. Für jeden passenden Eigenschaftsnamen in der Filterrichtlinie stimmt mindestens ein Eigenschaftswert mit dem Wert des Nachrichtenattributs überein.
- Wenn der Geltungsbereich der Filterrichtlinie auf `MessageBody` festgelegt ist, stimmt jeder Eigenschaftsname in der Filterrichtlinie mit dem Namen eines Nachrichtentexts überein. Für jeden passenden Eigenschaftsnamen in der Filterrichtlinie stimmt mindestens ein Eigenschaftswert mit dem Wert des Nachrichtentexts überein.

Amazon SNS unterstützt derzeit die folgenden Werte:

- UND-Logik
- ODER-Logik
- OR-Operator
- Schlüsselabgleich
- Exakter Abgleich mit numerischen Werten
- „Alles außer“-Abgleich für numerische Werte

- Abgleich von numerischen Wertebereichen
- Exakter Abgleich mit Zeichenfolgewerten
- „Alles außer“-Abgleich von Zeichenfolgewerten
- Zeichenfolgeabgleich unter Verwendung eines Präfixes mit dem „Alles außer“-Operator
- Zeichenfolgewert gleich bei ignorierter Groß- und Kleinschreibung
- Abgleich des Zeichenfolgewerts der IP-Adresse
- Präfix-Abgleich von Zeichenfolgewerten
- Suffix-Abgleich von Zeichenfolgewerten

Beispielfilterrichtlinien für Amazon SNS

Das folgende Beispiel zeigt eine Nachrichtennutzlast, zugestellt von einem Amazon-SNS-Thema, das Kundentransaktionen verarbeitet.

Das erste Beispiel enthält das `MessageAttributes`-Feld mit Attributen, die die Transaktion beschreiben:

- Interessen des Kunden
- Name des Geschäfts
- Status des Ereignisses
- Kaufpreis in USD

Da diese Nachricht das `MessageAttributes`-Feld enthält, kann jedes Themenabonnement, das eine `FilterPolicy` festlegt, die Nachricht selektiv akzeptieren oder ablehnen, sofern `FilterPolicyScope` im Abonnement auf `MessageAttributes` festgelegt ist. Weitere Informationen zur Anwendung von Attributen auf Nachrichten finden Sie unter Amazon-SNS-Nachrichtenattribute.

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

Das folgende Beispiel zeigt dieselben Attribute, die im `Message`-Feld enthalten sind, auch als Nachrichtennutzlast oder Nachrichtentext bezeichnet. Jedes Themenabonnement, das eine `FilterPolicy` enthält, kann die Nachricht selektiv akzeptieren oder ablehnen, sofern `FilterPolicyScope` im Abonnement auf `MessageBody` festgelegt ist.

```
{
"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"
}
```

Die folgenden Filterrichtlinien akzeptieren Nachrichten auf der Grundlage ihrer Eigenschaftsnamen und Werte oder weisen sie auf dieser Grundlage ab.

Eine Richtlinie, die die Beispielnachricht akzeptiert

Die Eigenschaften in der folgenden Abonnementfilterrichtlinie entsprechen den der Beispielnachricht zugewiesenen Attributen. Beachten Sie, dass dieselbe Filterrichtlinie für `FilterPolicyScope` gilt, unabhängig davon, ob sie auf `MessageAttributes` oder `MessageBody` festgelegt ist. Jeder Abonnent wählt seinen Filterbereich entsprechend der Zusammensetzung der Nachrichten aus, die er zu dem Thema erhält.

Wenn eine einzelne Eigenschaft in dieser Richtlinie nicht einem der Nachricht zugewiesenen Attribut entsprechen, weist die Richtlinie die Nachricht ab.

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

Eine Richtlinie, die die Beispielnachricht abweist

In der folgenden Abonnementfilterrichtlinie gibt es mehrere Nichtübereinstimmungen zwischen ihren Eigenschaften und den der Beispielnachricht zugewiesenen Attributen. Beispiel: Da der Eigenschaftsname `encrypted` in den Nachrichtenattributen nicht vorhanden ist, veranlasst diese Richtlinieneigenschaft die Zurückweisung der Nachricht, unabhängig von dem ihm zugewiesenen Wert.

Wenn Abweichungen auftreten, weist die Richtlinie die Nachricht zurück.

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

```
      "baseball"
   ]
}
```

Einschränkungen der Filterrichtlinien in Amazon SNS

Wenn Sie Filterrichtlinien in Amazon SNS einrichten, müssen Sie einige wichtige Regeln beachten. Diese Regeln tragen dazu bei, die effektive Anwendung von Filterrichtlinien sicherzustellen und gleichzeitig die Systemleistung und Kompatibilität aufrechtzuerhalten.

Übliche Einschränkungen für Richtlinien

Beachten Sie bei der Konfiguration von Filterrichtlinien in Amazon SNS die folgenden wichtigen Regeln, um sicherzustellen, dass sie effektiv funktionieren und gleichzeitig die Systemleistung und Kompatibilität erhalten bleiben:

- Zeichenkettenabgleich — Beim Zeichenfolgenabgleich in der Filterrichtlinie wird beim Vergleich zwischen Groß- und Kleinschreibung unterschieden.
- Numerischer Abgleich — Beim numerischen Abgleich kann der Wert zwischen -10^9 und 10^9 (-1 Milliarde bis 1 Milliarde) liegen, mit einer Genauigkeit von fünf Stellen nach dem Dezimalkomma.
- Komplexität der Filterrichtlinie — Die Gesamtkombination der Werte in einer Filterrichtlinie darf 150 nicht überschreiten. Um die Gesamtkombination zu berechnen, multiplizieren Sie die Anzahl der Werte in jedem Array in der Filterrichtlinie.
- Anzahl der Schlüssel einschränken — Eine Filterrichtlinie kann maximal fünf Schlüssel haben.

Zusätzliche Überlegungen

- Der JSON-Code der Filterrichtlinie kann Folgendes enthalten:
  - Zeichenfolgen, die von Anführungszeichen umschlossen werden
  - Zahlen
  - Die Schlüsselwörter `true`, `false` und `null`, jeweils ohne Anführungszeichen
- Wenn Sie die Amazon SNS SNS-API verwenden, müssen Sie das JSON der Filterrichtlinie als gültige UTF-8-Zeichenfolge übergeben.
- Die maximale Größe einer Filterrichtlinie beträgt 256 KB.

- Standardmäßig können Sie bis zu 200 Filterrichtlinien pro Thema und 10.000 Filterrichtlinien pro AWS Konto verwenden.

  Dieses Richtlinienlimit verhindert nicht, dass Amazon SQS SQS-Warteschlangenabonnements mit der `Subscribe`API erstellt werden. Es schlägt jedoch fehl, wenn Sie die Filterrichtlinie an den `Subscribe`-API-Aufruf (oder den `SetSubscriptionAttributes`-API-Aufruf) anhängen.

  Zur Erhöhung dieses Kontingents können Sie AWS Service Quotas verwenden.

Richtlinieneinschränkungen bei der attributbasierten Filterung

Attributbasierte Filterung ist die Standardoption. `FilterPolicyScope` ist im Abonnement auf `MessageAttributes` festgelegt.

- Amazon SNS akzeptiert bei der attributbasierten Filterung keine Richtlinie für verschachtelte Filter.
- Amazon SNS vergleicht die Richtlinieneigenschaften nur mit Nachrichtenattributen, die die folgenden Datentypen haben:
  - `String`
  - `String.Array`

> **Important**
>
> Wenn Sie die attributbasierte Filterung in Amazon SNS verwenden, müssen Sie bestimmte Sonderzeichen doppelt maskieren, insbesondere:
>
> - Doppelte Anführungszeichen („)
> - Backslashes ()
>
> Wenn diese Zeichen nicht doppelt maskiert werden, entspricht die Filterrichtlinie nicht den Attributen einer veröffentlichten Nachricht und die Benachrichtigung wird nicht zugestellt.

Zusätzliche Überlegungen

- Es wird nicht empfohlen, Objekte in Arrays zu übergeben, da dies aufgrund der Verschachtelung, die von der attributbasierten Filterung nicht unterstützt wird, zu unerwarteten Ergebnissen führen kann. Nutzlast-basierte Filterung für verschachtelte Richtlinien.
- `Number`wird für numerische Attributwerte unterstützt.
- Amazon SNS ignoriert Nachrichtenattribute mit dem Datentyp Binary.

Beispielrichtlinie für Komplexität:

Im folgenden Richtlinienbeispiel hat der erste Schlüssel drei Match-Operatoren, der zweite einen Match-Operator und der dritte hat zwei Match-Operatoren.

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

Die Gesamtkombination wird als Produkt der Anzahl der Vergleichsoperatoren für jeden Schlüssel in der Filterrichtlinie berechnet:

```
3(match operators of key_a)
x 1(match operators of key_b)
x 2(match operators of key_c)
= 6
```

Richtlinieneinschränkungen für die nutzlastbasierte Filterung

Wenn Sie von der attributbasierten (Standard) zur nutzlastbasierten Filterung wechseln möchten, müssen Sie den `FilterPolicyScope` im Abonnement auf `MessageBody` festlegen.

- Amazon SNS akzeptiert bei der nutzlastbasierten Filterung eine Richtlinie für verschachtelte Filter.
- Bei einer verschachtelten Richtlinie werden nur Blattschlüssel auf die Obergrenze von fünf Schlüsseln angerechnet.

Beispielrichtlinie für das Schlüssellimit:

Im folgenden Richtlinienbeispiel:

- Es gibt zwei Blatttasten: `key_c` und`key_e`.
- `key_c`hat vier Vergleichsoperatoren mit einer verschachtelten Ebene von drei und **`key_e`** verfügt über drei Vergleichsoperatoren mit einer verschachtelten Ebene von zwei.

```
{
"key_a": {
```

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

Die Gesamtkombination wird als Produkt aus der Anzahl der Vergleichsoperatoren und der verschachtelten Ebene für jeden Schlüssel in der Filterrichtlinie berechnet:

```
4(match operators of key_c)
x 3(nested level of key_c)
x 3(match operators of key_e)
x 2(nested level of key_e)
= 72
```

Richtlinien zur Verwendung von Wildcard-Mustern

Amazon SQS implementiert Schutzmaßnahmen für die Registrierung einer Filterrichtlinie mit Platzhaltern, um sicherzustellen, dass keine zu komplexen Filterrichtlinien erstellt werden, da dies die Leistung Ihrer Anwendung beeinträchtigen würde.

Struktur des Musters

Felder enthalten ein oder mehrere Muster. Das folgende Beispiel zeigt ein Feld, das zwei Muster verwendet:

```
{
    "greeting": [
      {"anything-but": {"prefix": "Hello"}},
      {"wildcard": "H*"}
    ] // 2 patterns
  }
```

Regeln für Komplexität

- Die Gesamtkomplexität von Platzhaltern in allen Feldern darf 100 Punkte nicht überschreiten
- Maximal 3 Platzhalter pro Muster

Berechnung der Komplexität

- Feldkomplexität = `(Sum of pattern points)` × `(Number of patterns)`
- Punkte im Muster:

  Einzelner Platzhalter: 1 Punkt

  Mehrere Platzhalter: jeweils 3 Punkte

  Alles außer: 1 Punkt

  Das Folgende ist ein Beispiel für die Komplexitätsberechnung:

```
{
  "filename": [
    {"wildcard": "*.txt"},     // 1 point
    {"wildcard": "log*"}      // 1 point
  ]                           // Total: (1 + 1) × 2 = 4 points
}
```

UND/ODER-Logik

Verwenden Sie AND/OR Logik in Filterrichtlinien, um Nachrichtenattributen oder Nachrichtentexteigenschaften in Amazon SNS zuzuordnen. Dies ermöglicht eine genauere und flexiblere Nachrichtenfilterung.

UND-Logik

Sie können die UND-Logik mit mehreren Attributnamen anwenden.

Betrachten Sie folgende Richtlinie:

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

Sie gleicht alle Nachrichtenattribute oder die Eigenschaft des Nachrichtentexts ab, bei denen der Wert `customer_interests` auf `rugby` und der Wert `price_usd` auf eine Zahl größer als 100 festgelegt ist.

> **Note**
>
> Sie können keine AND-Logik auf Werte desselben Attributs anwenden.

ODER-Logik

Sie können die ODER-Logik durch die Zuweisung mehrerer Werte zu einem Attributnamen anwenden.

Betrachten Sie folgende Richtlinie:

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

Sie gleicht alle Nachrichtenattribute ab, deren `customer_interests`-Wert auf `rugby`, `football`, oder `baseball` festgelegt ist.

OR-Operator

Sie können den `"$or"`-Operator verwenden, um explizit eine Filterrichtlinie zu definieren, um die OR-Beziehung zwischen mehreren Attributen in der Richtlinie auszudrücken.

Amazon SNS erkennt eine `"$or"`-Beziehung nur an, wenn die Richtlinie alle der folgenden Bedingungen erfüllt hat. Wenn all diese Bedingungen nicht erfüllt sind, wird `"$or"` wie ein regulärer Attributname behandelt, genauso wie jede andere Zeichenfolge in der Richtlinie.

- Die Regel enthält beispielsweise ein `"$or"`-Feldattribut, gefolgt von einem Array, z. B. `“$or” : []`.
- Das `"$or"`-Array enthält mindestens 2 Objekte: `"$or": [{}, {}]`.
- Keines der Objekte im `"$or"`-Array hat Feldnamen, bei denen es sich um reservierte Schlüsselwörter handelt.

Andernfalls wird `"$or"` wie ein normaler Attributname behandelt, genau wie andere Zeichenfolgen in der Richtlinie.

Die folgende Richtlinie wird nicht als OR-Beziehung analysiert, da Zahl und Präfix reservierte Schlüsselwörter sind.

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

Beispiele für **OR**-Operatoren

Standard OR:

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

Die Filterlogik für diese Richtlinie lautet:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattributsätze überein:

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

or

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

or

```
{
```

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

Richtlinieneinschränkungen, zu denen auch **OR**-Beziehungen gehören

Betrachten Sie folgende Richtlinie:

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

Die Logik dieser Richtlinie kann auch wie folgt vereinfacht werden:

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

Die Komplexitätsberechnung für Richtlinien mit OR-Beziehungen kann vereinfacht werden, indem die Summe der Kombinationskomplexitäten für jede OR-Anweisung berechnet wird.

Die Gesamtkombination wird wie folgt berechnet:

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

`source` hat einen Wert, `metricName` hat zwei Werte, `metricType` hat einen Wert, `metricId` hat zwei Werte und `spaceId` hat drei Werte.

Beachten Sie die folgende Richtlinie für verschachtelte Filter:

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

Die Logik dieser Richtlinie kann wie folgt vereinfacht werden:

```
("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")
```

Die Berechnung der Gesamtkombinationen ist bei nicht verschachtelten Richtlinien gleich, außer dass wir die Verschachtelungsebene des Schlüssels berücksichtigen müssen.

Die Gesamtkombination wird wie folgt berechnet:

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

`metricName` hat zwei Werte, `namespace` hat zwei Werte, `scope` ist ein zweistufiger verschachtelter Schlüssel mit einem Wert, `source` ist ein zweistufiger verschachtelter Schlüssel mit einem Wert und `type` ist ein zweistufiger verschachtelter Schlüssel mit einem Wert.

Schlüsselabgleich

Verwenden Sie den `exists` Operator in einer Filterrichtlinie, um eingehende Nachrichten danach abzugleichen, ob eine bestimmte Eigenschaft vorhanden ist oder nicht.

- `exists`funktioniert nur bei Blattknoten (letzte Attribute in der Struktur).
- Es gilt nicht für Zwischenknoten innerhalb einer verschachtelten JSON-Struktur.

- Verwenden Sie `"exists": true`, um eingehende Nachrichten abzugleichen, die die angegebene Eigenschaft enthalten. Der Schlüssel muss einen Wert haben, der nicht null ist und einen der nicht leer ist.

  Die folgende Richtlinie verwendet beispielsweise den `exists`-Operator mit dem Wert `true`:

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

  Sie gleicht alle Listen mit Nachrichtenattributen ab, die den `store`-Attributschlüssel besitzen, z. B. die folgenden:

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

  Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

  Sie stimmt jedoch nicht mit Nachrichten überein, die nicht den Attributschlüssel `store` besitzen, z. B. die folgenden:

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

  Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

- Verwenden Sie `"exists": false`, um eingehende Nachrichten abzugleichen, die die angegebene Eigenschaft nicht enthalten.

> **Note**
>
> `"exists": false` stimmt nur überein, wenn mindestens ein Attribut vorhanden ist. Ein leerer Satz von Attributen führt dazu, dass der Filter nicht übereinstimmt.

Die folgende Richtlinie verwendet beispielsweise den `exists`-Operator mit dem Wert `false`:

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

Sie stimmt nicht mit Listen mit Nachrichtenattributen überein, die den Attributschlüssel `store` besitzen, z. B. die folgenden:

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

Sie stimmt jedoch mit allen Listen mit Nachrichtenattributen ohne den Attributschlüssel `store` überein, z. B. die folgenden:

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

Sie stimmt auch mit dem folgenden Nachrichtentext überein:

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

Abgleichen numerischer Werten

Filtern Sie Nachrichten, indem Sie numerische Werte mit Nachrichtenattributwerten oder Eigenschaftswerten des Nachrichtentexts abgleichen. Numerische Werte werden in der JSON-Richtlinie nicht von doppelten Anführungszeichen eingeschlossen. Sie können zum Filtern die folgenden numerischen Vorgänge verwenden.

> **Note**
>
> Präfixe werden nur für den Zeichenfolgeabgleich unterstützt.

Genaue Übereinstimmung

Wenn eine Richtlinieneigenschaft das Schlüsselwort `numeric` und den =-Operator enthält, stimmt sie mit allen Werten des Nachrichtenattributs und des Nachrichtentexts überein, die den gleichen Namen und den gleichen numerischen Wert besitzen.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

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

„Alles außer“-Abgleich

Wenn ein Richtlinieneigenschaftswert das Schlüsselwort `anything-but` enthält, stimmt er mit allen Eigenschaftswerten für Nachrichtenattribute oder Nachrichtentexte überein, die keinen der Richtlinieneigenschaftswerte enthalten.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

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

Sie stimmt zudem mit dem folgenden Nachrichtenattribut überein (da es einen Wert enthält, der nicht `100` oder `500` ist):

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

Außerdem stimmt sie auch mit dem folgenden Nachrichtentext überein (da er einen Wert enthält, der weder `100` noch `500` ist):

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

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

Wertbereichsübereinstimmung

Zusätzlich zum =-Operator kann eine numerische Richtlinieneigenschaft die folgenden Operatoren enthalten: <, <=, > und >=.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit allen Nachrichtenattributen mit negativen numerischen Werten überein.

Betrachten Sie ein weiteres Nachrichtenattribut:

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

Sie stimmt mit allen Nachrichtenattributen mit positiven Zahlen bis einschließlich 150 überein.

Abgleich von Zeichenfolgewerten

Filtern Sie Nachrichten, indem Sie Zeichenfolgenwerte mit Nachrichtenattributwerten oder Eigenschaftswerten des Nachrichtentexts abgleichen. Zeichenfolgenwerte werden in der JSON-Richtlinie von doppelten Anführungszeichen eingeschlossen. Sie können die folgenden Zeichenkettenoperationen verwenden, um Nachrichtenattribute oder Eigenschaften des Nachrichtentexts abzugleichen:

Genaue Übereinstimmung

Eine genaue Übereinstimmung liegt vor, wenn eine Richtlinieneigenschaft einem oder mehreren Nachrichtenattributwerten entspricht. Bei `String.Array` Typattributen wird jedes Element im Array zu Vergleichszwecken als separate Zeichenfolge behandelt.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit den folgenden Nachrichtenattributen überein:

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

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

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

Sie stimmt auch mit den folgenden Nachrichtentexten überein:

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

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

Es entspricht jedoch nicht den folgenden Nachrichtenattributen:

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

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

„Alles außer“-Abgleich

Wenn ein Richtlinieneigenschaftswert das Schlüsselwort `anything-but` enthält, stimmt er mit allen Nachrichtenattributen oder Nachrichtentextwerten überein, die keinen der Richtlinieneigenschaftswerte enthalten. `anything-but` kann mit `"exists": false` kombiniert

werden. Bei `String.Array` Typattributen entspricht es, wenn keines der Array-Elemente in der Richtlinieneigenschaft aufgeführt ist.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Es entspricht einem der folgenden Nachrichtenattribute:

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

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

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

Sie stimmt zudem mit dem folgenden Nachrichtenattribut überein (da es einen Wert enthält, der nicht `rugby` oder `tennis` ist):

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

Außerdem stimmt sie auch mit dem folgenden Nachrichtentext überein (da sie einen Wert enthält, der weder `rugby` noch `tennis` ist):

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

Es entspricht jedoch nicht den folgenden Nachrichtenattributen:

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

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

Verwenden Sie ein Präfix mit **`anything-but`**

Für den Zeichenfolgeabgleich können Sie auch ein Präfix mit dem `anything-but`-Operator verwenden. Die folgende Richtlinieneigenschaft beispielsweise verweigert das `order-`-Präfix:

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

Es entspricht einem der beiden folgenden Attribute:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

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

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

alles außer Wildcard

Die folgende Richtlinieneigenschaft verweigert den Platzhalter: `*ball`

```
"customer_interests" : [{ "anything-but": { "wildcard": "*ball" }}]
```

Sie entspricht den folgenden Attributen:

```
{"customer_interests": ["hockey", "rugby", "soccer] }
```

Es entspricht jedoch nicht dem folgenden Nachrichtenattribut:

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

alles andere als Suffix

Die folgende Richtlinieneigenschaft verweigert die `-ball`

Suffix:

```
"customer_interests": [ { "anything-but": { "suffix": "ball" } } ]
```

Es entspricht den folgenden Attributen:

```
{"customer_interests": ["hockey", "rugby", "soccer] }
```

Es entspricht jedoch nicht dem folgenden Nachrichtenattribut:

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

Equals-ignore-case übereinstimmend

Wenn eine Richtlinieneigenschaft das Schlüsselwort `equals-ignore-case` enthält, stimmt sie mit allen Werten des Nachrichtenattributs oder der Texteigenschaft überein, die mit den angegebenen Zeichen beginnen.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

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

Abgleich von IP-Adressen

Sie können das `cidr`-Operator, um zu überprüfen, ob eine eingehende Nachricht von einer bestimmten IP-Adresse oder einem bestimmten Subnetz stammt.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
```

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

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

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

Übereinstimmung mit einem Präfix

Wenn eine Richtlinieneigenschaft das Schlüsselwort `prefix` enthält, stimmt sie mit allen Werten des Nachrichtenattributs oder der Texteigenschaft überein, die mit den angegebenen Zeichen beginnen.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

```
{
```

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

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

Suffixabgleich

Wenn eine Richtlinieneigenschaft das Schlüsselwort `suffix` enthält, stimmt sie mit allen Werten des Nachrichtenattributs oder der Texteigenschaft überein, die auf das angegebene Zeichen enden.

Betrachten Sie folgende Richtlinieneigenschaft:

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

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

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

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

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

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

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

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

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

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

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

Platzhalter

Sie können das Platzhalterzeichen (*) verwenden, um Zeichenfolgenwerte in Ereignismustern abzugleichen.

Die folgende Richtlinie verwendet das Platzhalterzeichen (*):

```
"customer_interests": [ { "wildcard": "*ball" } ]
```

Es entspricht den folgenden Attributen:

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

Anwenden einer Abonnementfilterrichtlinie in Amazon SNS

Die Nachrichtenfilterung in Amazon SNS ermöglicht es Ihnen, Nachrichten anhand von Filterrichtlinien selektiv an Abonnenten zu senden. Diese Richtlinien definieren Bedingungen, die Nachrichten erfüllen müssen, um im Rahmen eines Abonnements zugestellt zu werden. Die Zustellung von Rohnachrichten ist zwar eine Option, die sich auf die Nachrichtenverarbeitung auswirken kann, sie ist jedoch nicht erforderlich, damit Abonnementfilter funktionieren.

Sie können eine Filterrichtlinie unter Verwendung der Amazon SNS-Konsole auf ein Amazon SNS-Abonnement anwenden. Um Richtlinien programmgesteuert anzuwenden, können Sie auch die Amazon SNS-API, das AWS Command Line Interface (AWS CLI) oder ein beliebiges AWS SDK verwenden, das Amazon SNS unterstützt. Sie können auch verwenden. AWS CloudFormation

Die Zustellung von Rohnachrichten aktivieren

Die Zustellung von Rohnachrichten stellt sicher, dass Nachrichtennutzdaten den Abonnenten unverändert und ohne zusätzliche Kodierung oder Transformation zugestellt werden. Dies kann nützlich sein, wenn Abonnenten das ursprüngliche Nachrichtenformat für die Verarbeitung

benötigen. Die Zustellung von Rohnachrichten steht jedoch nicht in direktem Zusammenhang mit der Funktionalität von Abonnementfiltern.

Abonnementfilter anwenden

Um Nachrichtenfilter auf ein Abonnement anzuwenden, definieren Sie eine Filterrichtlinie mithilfe der JSON-Syntax. Diese Richtlinie legt die Bedingungen fest, die eine Nachricht erfüllen muss, um im Abonnement zugestellt zu werden. Filter können auf Nachrichtenattributen wie Nachrichtenattributen, Nachrichtenstruktur oder sogar Nachrichteninhalten basieren.

Beziehung zwischen Rohnachrichtenzustellung und Abonnementfiltern

Die Aktivierung der Zustellung von Rohnachrichten kann sich zwar darauf auswirken, wie Nachrichten von Abonnenten zugestellt und verarbeitet werden, ist jedoch keine Voraussetzung für die Verwendung von Abonnementfiltern. In Szenarien, in denen Abonnenten das ursprüngliche Nachrichtenformat ohne Änderungen benötigen, kann die Aktivierung der Rohnachrichtenzustellung jedoch zusammen mit Abonnementfiltern von Vorteil sein.

Überlegungen zu einer effektiven Filterung

Berücksichtigen Sie bei der Implementierung der Nachrichtenfilterung die spezifischen Anforderungen Ihrer Anwendung und Ihrer Abonnenten. Definieren Sie Filterrichtlinien, die den Kriterien für die Nachrichtenzustellung genau entsprechen, um eine effiziente und gezielte Nachrichtenverteilung zu gewährleisten.

> **Important**
>
> AWS Dienste wie IAM und Amazon SNS verwenden ein verteiltes Rechenmodell, das als Eventual Consistency bezeichnet wird. Hinzufügungen oder Änderungen einer Abonnementfilterrichtlinie dauern bis zu 15 Minuten, bis sie vollständig wirksam werden.

AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie ein Abonnement und dann Edit (Bearbeiten) aus.
4. Erweitern Sie auf der Seite Edit (Bearbeiten) den Abschnitt Subscription filter policy (Abonnementfilterrichtlinie).

5. Wählen Sie zwischen attributbasierter Filterung oder nutzlastbasierter Filterung aus.
6. Geben Sie im Feld JSON editor (JSON-Editor) den JSON-Text Ihrer Filterrichtlinie an.
7. Wählen Sie Änderungen speichern.

   Amazon SNS wendet Ihre Filterrichtlinie auf das Abonnement an.

AWS CLI

Um eine Filterrichtlinie mit dem AWS Command Line Interface (AWS CLI) anzuwenden, verwenden Sie den `set-subscription-attributes`Befehl, wie im folgenden Beispiel gezeigt. Wählen Sie für die Option `--attribute-name` den Wert `FilterPolicy`. Geben Sie für `--attribute-value` Ihre JSON-Richtlinie an.

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

Um gültiges JSON für Ihre Richtlinie anzugeben, schließen Sie die Attributnamen und Werte in doppelte Anführungszeichen ein. Sie müssen auch das gesamte Richtlinienargument in Anführungszeichen einschließen. Um das Maskieren von Anführungszeichen zu vermeiden, können Sie die Richtlinie in einfache Anführungszeichen und die JSON-Namen und -Werte in doppelte Anführungszeichen setzen, wie im folgenden Beispiel gezeigt.

Wenn Sie von der attributbasierten (Standard) zur nutzlastbasierten Nachrichtenfilterung wechseln möchten, können Sie den Befehl ebenfalls verwenden. set-subscription-attributes Wählen Sie für die Option `--attribute-name` den Wert `FilterPolicyScope`. Legen Sie für `--attribute-value` die Option `MessageBody` fest.

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

Um sicherzustellen, dass Ihre Filterrichtlinie angewendet wurde, verwenden Sie den Befehl `get-subscription-attributes`. Die Attribute in der Terminal-Ausgabe sollten Ihre Filterrichtlinie für den `FilterPolicy`-Schlüssel zeigen, wie im folgenden Beispiel:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
    "Attributes": {
```

```
        "Endpoint": "endpoint . . .",
        "Protocol": "https",
        "RawMessageDelivery": "false",
        "EffectiveDeliveryPolicy": "delivery policy . . .",
        "ConfirmationWasAuthenticated": "true",
        "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed
\"]}",
        "FilterPolicyScope": "MessageAttributes",
        "Owner": "111122223333",
        "SubscriptionArn": "arn:aws:sns: . . .",
        "TopicArn": "arn:aws:sns: . . ."
    }
}
```

AWS SDKs

Die folgenden Code-Beispiele zeigen, wie `SetSubscriptionAttributes` verwendet wird.

> **Important**
>
> Wenn Sie das Beispiel SDK for Java 2.x verwenden, ist die Klasse `SNSMessageFilterPolicy` nicht sofort verfügbar. Anweisungen zur Installation dieser Klasse finden Sie im Beispiel auf der GitHub Website.

CLI

AWS CLI

So legen Sie Abonnementattribute fest

Im folgenden `set-subscription-attributes`-Beispiel wird das `RawMessageDelivery`-Attribut auf ein SQS-Abonnement festgelegt.

```
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
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird ein `FilterPolicy`-Attribut auf ein SQS-Abonnement festgelegt.

```
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\"] }"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird das `FilterPolicy`-Attribut von einem SQS-Abonnement entfernt.

```
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 "{}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin AWS SDK for Python (Boto3) API Reference.

Amazon SNS-API

Um eine Filterrichtlinie mit der Amazon SNS API anzuwenden, stellen Sie eine Anfrage an die `SetSubscriptionAttributes`-Aktion. Setzen Sie den Parameter `AttributeName` auf `FilterPolicy`, und den Parameter `AttributeValue` auf das JSON Ihrer Filterrichtlinie.

Wenn Sie von der attributbasierten (Standard) zur nutzlastbasierten Nachrichtenfilterung wechseln möchten, können Sie auch die Aktion `SetSubscriptionAttributes` verwenden. Legen Sie den `AttributeName`-Parameter auf `FilterPolicyScope` und den `AttributeValue`-Parameter auf `MessageBody` fest.

AWS CloudFormation

Um eine Filterrichtlinie anzuwenden AWS CloudFormation, verwenden Sie eine JSON- oder YAML-Vorlage, um einen AWS CloudFormation Stack zu erstellen. Weitere Informationen finden Sie unter der `FilterPolicy`Eigenschaft der `AWS::SNS::Subscription` Ressource im AWS CloudFormation Benutzerhandbuch und in der AWS CloudFormation Beispielvorlage.

1. Melden Sie sich an der AWS CloudFormation -Konsole an.
2. Wählen Sie Stapel erstellen aus.
3. Wählen Sie auf der Seite Select Template (Vorlage auswählen) die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen) aus. Wählen Sie dann Ihre Datei und Next (Weiter) aus.
4. Führen Sie auf der Seite Specify DB Details (Festlegen von DB-Detail) die folgenden Schritte aus:
   a. Geben Sie für Stack name (Stack-Name) `MyFilterPolicyStack` ein.
   b. Geben Sie zum myHttpEndpointBeispiel den HTTP-Endpunkt ein, um Ihr Thema zu abonnieren.

   > **Tip**
   > Wenn Sie nicht über einen HTTP-Endpunkt verfügen, erstellen Sie einen.

5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Klicken Sie auf der Seite Review auf Create.

Entfernen einer Abonnementfilterrichtlinie in Amazon SNS

Um das Filtern der Nachrichten, die an ein Abonnement gesendet werden, zu beenden, entfernen Sie die Filterrichtlinie für das Abonnement, indem Sie sie mit einer leeren JSON-Nachricht überschreiben. Nachdem Sie die Richtlinie entfernt haben, akzeptiert das Abonnement jede Nachricht, die ihm gesendet wird.

Mit dem AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie ein Abonnement und dann Edit (Bearbeiten) aus.
4. Erweitern Sie auf der *EXAMPLE1-23bc-4567-d890-ef12g3hij456* Seite Bearbeiten den Abschnitt Abonnementfilterrichtlinie.
5. Geben Sie im Feld JSON editor (JSON-Editor) einen leeren JSON-Code für Ihre Filterrichtlinie an: `{}`
6. Wählen Sie Änderungen speichern.

   Amazon SNS wendet Ihre Filterrichtlinie auf das Abonnement an.

Verwenden Sie den AWS CLI

Um eine Filterrichtlinie mit dem zu entfernen AWS CLI, verwenden Sie den `set-subscription-attributes`Befehl und geben Sie einen leeren JSON-Hauptteil für das `--attribute-value` Argument an:

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

Verwenden der Amazon SNS SNS-API

Um eine Filterrichtlinie mit dem Amazon SNS API zu entfernen, stellen Sie eine Anfrage an die `SetSubscriptionAttributes`-Aktion. Setzen Sie den Parameter `AttributeName` auf `FilterPolicy` und geben Sie einen leeren JSON-Rumpf für den Parameter `AttributeValue` an.

Schutz von Nachrichtendaten in Amazon SNS

Was ist Nachrichtendatenschutz?

Der Nachrichtendatenschutz schützt die Daten, die in Ihren Amazon SNS SNS-Themen veröffentlicht werden, indem die vertraulichen Informationen, die zwischen Anwendungen oder Diensten übertragen werden, mithilfe von Datenschutzrichtlinien geprüft, maskiert, geschwärzt oder blockiert werden. AWS

Im Rahmen des Nachrichtendatenschutzes werden Daten mithilfe von Datenkennungen in Übertragung auf persönlich identifizierbare Informationen (PII) und auf geschützte Gesundheitsdaten (PHI) gescannt. Sie können wählen, ob Sie vordefinierte (oder von Amazon SNS verwaltete) Datenkennungen (z. B. Namen, Adressen, Kreditkartennummern und Codes für verschreibungspflichtige Medikamente) verwenden möchten, oder Sie können Ihre eigenen benutzerdefinierten Datenkennungen erstellen, die auf Ihren geschäftlichen Anwendungsfall zugeschnitten sind. Mithilfe der gescannten Informationen stellt der Nachrichtendatenschutz detaillierte Überwachungsprotokolle bereit und ermöglicht es Ihnen, Maßnahmen zum Schutz dieser Daten zu ergreifen.

Der Nachrichtendatenschutz unterstützt die folgenden Aktionen, um sensible Kundeninformationen zu schützen:

- Audit – Überprüft bis zu 99 % der Daten, die in einem Amazon-SNS-Thema veröffentlicht werden. Sie können dann wählen, ob Sie die Ergebnisse an Amazon CloudWatch, Amazon S3 oder senden möchtenhttps://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html.
- Anonymisieren – Maskieren oder Redigieren sensibler Daten, ohne die Nachrichtenveröffentlichung zu unterbrechen.
- Verweigern — Blockiert die Übertragung von Daten zwischen Anwendungen und AWS Ressourcen, wenn die Payload sensible Daten enthält.

> **Note**
>
> Amazon SNS unterstützt den Nachrichtendatenschutz nur für Standardthemen von Amazon SNS.

Weshalb sollte ich den Nachrichtendatenschutz verwenden?

Durch die Einführung des Nachrichtendatenschutzes in Ihre Governance-, Risikomanagement- und Compliance-Programme können Sie Datenschutzrichtlinien implementieren, die Ihnen helfen, Datenlecks zu erkennen und zu verhindern. Dadurch stehen Ihren Teams Tools zur Verfügung, mit denen Sie finanzielle, rechtliche und regulatorische Risiken reduzieren können, indem sie Datenschutzbestimmungen wie HIPAA, DSGVO, PCI und FedRAMP einhalten. Außerdem entlastet es Ihre Entwickler vom betrieblichen Aufwand, der mit der Entwicklung und Verwaltung Ihrer eigenen Tools zum Schutz sensibler Daten verbunden ist.

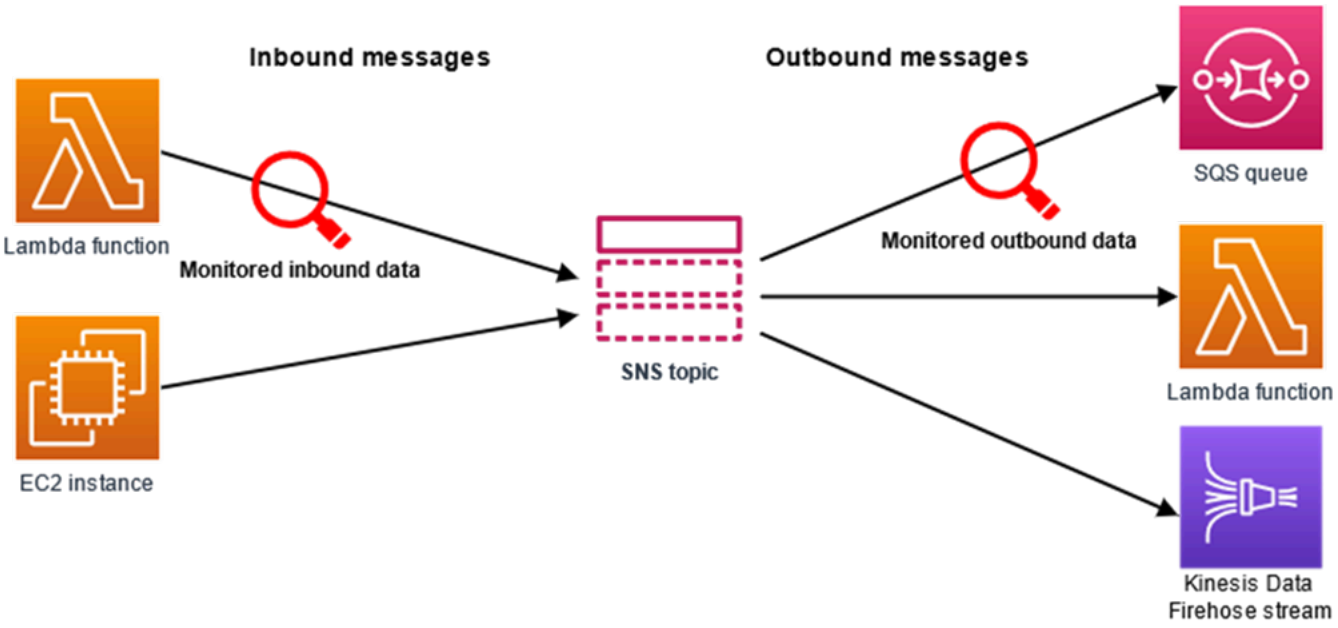
Beispielsweise können Sie mit dem Nachrichtendatenschutz eine Prüfungsrichtlinie erstellen, um festzustellen, ob eines Ihrer Systeme versehentlich sensible Daten sendet oder empfängt. Wenn Ihre Prüfungsergebnisse zeigen, dass Systeme Kreditkarteninformationen an Systeme senden, die diese nicht benötigen, können Sie eine Blockierungsrichtlinie erstellen, um die Zustellung der Daten zu verhindern.

> **Note**
>
> Amazon SNS unterstützt den Nachrichtendatenschutz nur für Standardthemen von Amazon SNS.

Grundlegendes zu den Datenschutzrichtlinien von Amazon SNS

Was sind Datenschutzrichtlinien?

Amazon SNS verwendet Datenschutzrichtlinien, um die sensiblen Daten, nach denen Sie suchen möchten, und die Maßnahmen auszuwählen, die Sie ergreifen sollten, um zu verhindern, dass diese Daten von Ihren Amazon-SNS-Themen ausgetauscht werden. Sie verwenden Datenkennungen, um die entsprechenden sensiblen Daten auszuwählen. Der Amazon-SNS-Nachrichtendatenschutz erkennt dann die sensiblen Daten mithilfe von Machine Learning und Musterabgleich. Um auf gefundene Datenkennungen zu reagieren, können Sie einen Vorgang zum Prüfen Anonymisieren oder Verweigern definieren. Mit diesen Vorgängen können Sie die gefundenen (oder nicht gefundenen) sensiblen Daten protokollieren, maskieren oder redigieren oder die Nachrichtenzustellung verweigern.

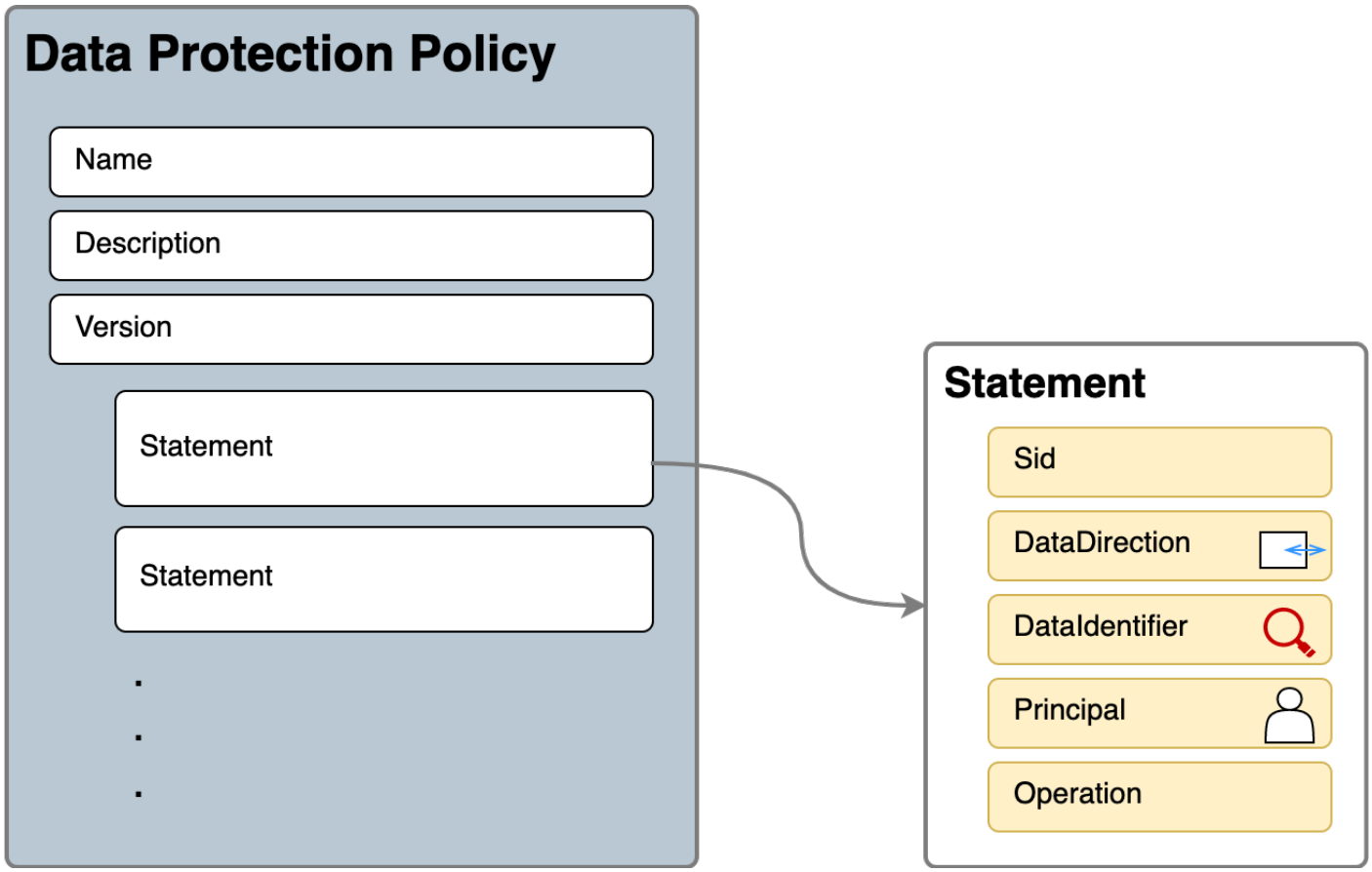


Wie ist die Datenschutzrichtlinie aufgebaut?

Wie in der folgenden Abbildung dargestellt, umfasst ein Datenschutzrichtlinien-Dokument die folgenden Elemente:

- Optionale richtlinienweite Informationen oben im Dokument.
- Eine oder mehrere einzelne Anweisungen

Jede Anweisung enthält Angaben über die jeweilige Berechtigung.



Für jedes Amazon-SNS-Thema kann nur eine Datenschutzrichtlinie definiert werden. Die Datenschutzrichtlinie kann mindestens eine Verweigerungs- oder Anonymisierungsanweisung aber nur eine Prüfungsanweisung enthalten.

JSON-Eigenschaften für die Datenschutzrichtlinie

Eine Datenschutzrichtlinie erfordert die folgenden grundlegenden Richtlinieninformationen zur Identifizierung:

- Name – der Name der Richtlinie
- Beschreibung (Optional) – die Beschreibung der Richtlinie
- Version – die Sprachversion der Richtlinie. Die aktuelle Version ist 2021-06-01.
- Anweisung – Eine Liste mit Anweisungen, die Aktionen der Datenschutzrichtlinie angeben.

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

```
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

JSON-Eigenschaften für eine Richtlinienanweisung

Eine Richtlinienanweisung legt den Erkennungskontext für den Datenschutzvorgang fest.

- Sid (Optional) – der Anweisungsbezeichner
- DataDirection— Eingehend (für Publish-API-Anfragen) oder Ausgehend (für Benachrichtigungszustellungen) in Bezug auf das Amazon SNS SNS-Thema.
- DataIdentifier— Die sensiblen Daten, nach denen das Amazon SNS SNS-Thema suchen soll. Zum Beispiel Name, Adresse oder Telefonnummer.
- Prinzipal – Der IAM-Prinzipal, der in dem Thema veröffentlicht ist, oder der IAM-Prinzipal, der das Thema abonniert hat.
- Vorgang – die Folgeaktion, entweder Audit (Prüfen), De-identify (Anonymisieren) (maskieren oder redigieren) oder Deny (Verweigern) (Blockieren), die das Amazon-SNS-Thema ausführt, sobald es sensible Daten findet.

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

JSON-Eigenschaften für eine Richtlinienanweisungsoperation

Eine Richtlinienanweisung legt eine der folgenden Datenschutzoperationen fest.

- Audit (Prüfen) – Gibt Metriken und Ergebnisprotokolle aus, ohne die Veröffentlichung oder Zustellung von Nachrichten zu unterbrechen.
- Anonymisieren – maskieren oder redigieren Sie sensible Daten, ohne die Nachrichtenveröffentlichung zu unterbrechen.
- Verweigern – blockiert die Amazon-SNS-Veröffentlichungsanforderung oder verhindert die Nachrichtenzustellung.

Wie ermittle ich die IAM-Prinzipale für meine Datenschutzrichtlinie?

Der Nachrichtendatenschutz verwendet zwei IAM-Prinzipale, die mit Amazon SNS interagieren.

1. Publish-API-Prinzipal (eingehend) – Der authentifizierte IAM-Prinzipal, der die `Publish`-API von Amazon SNS aufruft.
2. Abonnementprinzipal (ausgehend) – Der authentifizierte IAM-Prinzipal, der die `Subscribe`-API während der Abonnementerstellung aufgerufen hat.

Der `SubscriptionPrincipal` ist eine öffentlich zugängliche Amazon-SNS-Abonnementeigenschaft, die über die `GetSubscriptionAttributes`-API abgerufen werden kann.

```
{
  "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"
  }
}
```

Datenschutzrichtlinien-Operationen in Amazon SNS

Im Folgenden finden Sie Beispiele für Datenschutzrichtlinien, mit denen Sie sensible Daten überprüfen und verweigern können. Ein vollständiges Tutorial, das eine Beispielanwendung enthält,

finden Sie im Blogbeitrag Introducing message data protection for Amazon SNS (Einführung in den Nachrichtendatenschutz für Amazon SNS).

Audit-Operation

Bei der Prüfung werden eingehende Nachrichten zum Thema geprüft und die Ergebnisse sensibler Daten an einem AWS Zielort protokolliert. Die Samplerate kann eine ganze Zahl zwischen 0 und 99 sein. Für diese Operation ist eine der folgenden Arten von Protokollierungszielen erforderlich:

1. FindingsDestination— Das Logging-Ziel, wenn das Amazon SNS SNS-Thema sensible Daten in der Payload findet.
2. NoFindingsDestination— Das Ziel für die Protokollierung, wenn das Amazon SNS SNS-Thema keine sensiblen Daten in der Nutzlast findet.

Sie können AWS-Services in jedem der folgenden Protokollzieltypen Folgendes verwenden:

- Amazon CloudWatch Logs (optional) — Das `LogGroup` muss sich im Themenbereich befinden und der Name muss mit /aws/vendedlogs/ beginnen.
- (Optional) — Sie `DeliveryStream` müssen sich in der Themenregion befinden und Direct PUT als Quelle für den Lieferstream haben. Weitere Informationen finden Sie unter Quelle, Ziel und Name im Amazon Data Firehose Developer Guide.
- Amazon S3 (Optional) – Ein Amazon-S3-Bucket-Name. Für die Verwendung des Amazon-S3-Buckets mit aktivierter SSE-KMS-Verschlüsselung sind zusätzliche Aktionen erforderlich.

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

Erforderliche Berechtigungen bei der Angabe von Protokollierungszielen

Wenn Sie in der Datenschutzrichtlinie Ziele für die Protokollierung angeben, müssen Sie der IAM-Identitätsrichtlinie des IAM-Prinzipals, der in Amazon SNS die `PutDataProtectionPolicy`-API oder die `CreateTopic`-API mit dem `--data-protection-policy`-Parameter aufruft, die folgenden Berechtigungen hinzufügen.

Audit-Ziel	IAM-Berechtigung
Standard	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole

Audit-Ziel	IAM-Berechtigung
	firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Für die Verwendung des Amazon-S3-Buckets mit aktivierter SSE-KMS-Verschlüsselung sind zusätzliche Aktionen erforderlich.

JSON

```
{
    "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:us-east-1:account-id:SampleLogGroupName:*:*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "firehose:TagDeliveryStream"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

Erforderliche Schlüsselrichtlinie zur Verwendung mit SSE-KMS

Wenn Sie einen Amazon S3 S3-Bucket als Protokollziel verwenden, können Sie die Daten in Ihrem Bucket schützen, indem Sie entweder serverseitige Verschlüsselung mit Amazon S3-Managed Keys (SSE-S3) oder serverseitige Verschlüsselung mit (SSE-KMS) aktivieren. AWS KMS keys Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung im Amazon S3-Entwicklerhandbuch.

Wenn Sie SSE-S3 wählen, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

Wenn Sie sich für SSE-KMS entscheiden, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden. Sie müssen die Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel aktualisieren, damit das Protokollzustellungskonto in Ihren S3-Bucket schreiben kann. Weitere Informationen zur erforderlichen Schlüsselrichtlinie für die Verwendung mit SSE-KMS finden Sie unter serverseitige Verschlüsselung des Amazon S3 S3-Buckets im Amazon CloudWatch Logs-Benutzerhandbuch.

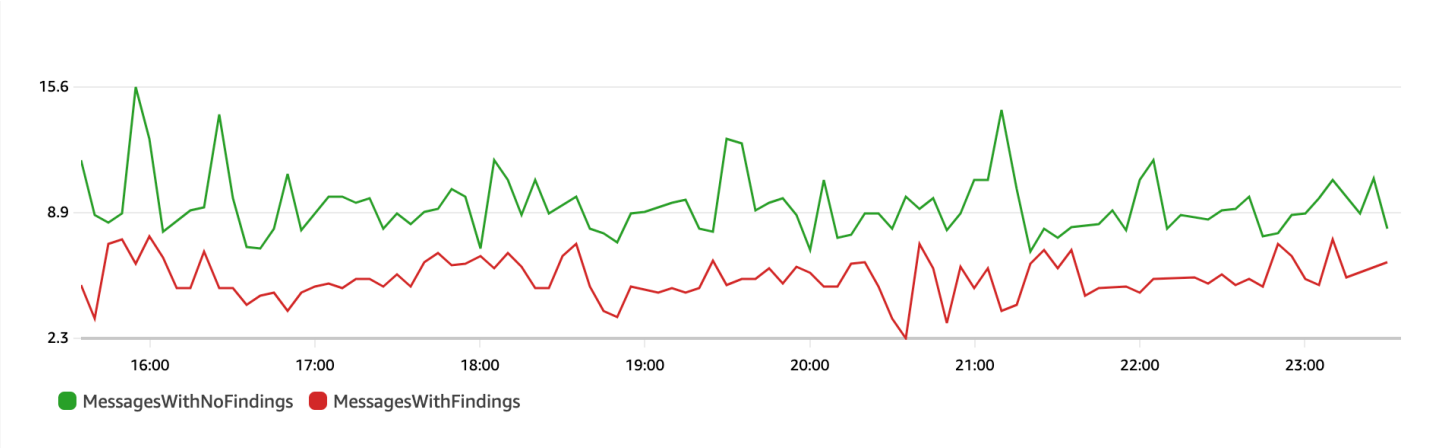
Beispiel für ein Audit-Zielprotokoll

Im folgenden Beispiel wird `callerPrincipal` verwendet, um die Quelle des sensiblen Inhalts zu identifizieren, und `messageID` als Referenz zur Überprüfung der `Publish`-API-Antwort.

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

Metriken für die Audit-Operation

Wenn bei einem Prüfvorgang die `NoFindingsDestination` Eigenschaft `FindingsDestination` oder angegeben wurde, erhalten die Eigentümer des Themas CloudWatch `MessagesWithFindings` auch `MessagesWithNoFindings` Messwerte.



Vorgang anonymisieren

Der Vorgang der Anonymisierung maskiert oder redigiert sensible Daten aus veröffentlichten oder übermittelten Nachrichten. Dieser Vorgang ist nur für eingehende und ausgehende Nachrichten verfügbar und erfordert einen der folgenden Konfigurationstypen:

- MaskConfig— Maske mit einem unterstützten Zeichen aus der folgenden Tabelle. Beispielsweise wird die SSN: `123-45-6789` zur SSN: `###########`.

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

Unterstütztes Maskierungszeichen	Name
*	Sternchen
A–Z, a–z und 0–9	Alphanumerisch
	Leerzeichen
!	Ausrufezeichen

Unterstütztes Maskierungszeichen	Name
$	Dollarzeichen
%	Prozentzeichen
&	Kaufmännisches Und-Zeichen
()	Klammer
+	Pluszeichen
,	Komma
-	Bindestrich
.	Intervall
/\	Schrägstrich, umgekehrter Schrägstrich
#	Doppelkreuz
:	Doppelpunkt
;	Semikolon
=, <>	Gleich. kleiner oder größer als
@	At-Zeichen
[]	Klammern
^	Caret-Symbol
_	Unterstrich
`	Backtick
|	Senkrechter Balken
~	Tilde-Zeichen

- RedactConfig— Redigieren Sie, indem Sie die Daten vollständig entfernen. Beispielsweise wird SSN: `123-45-6789` zur SSN: .

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

Bei einer eingehenden Nachricht werden die sensiblen Daten nach dem Prüfvorgang anonymisiert, und der `SNS:Publish`-API-Aufrufer erhält den folgenden „Ungültiger Parameter“-Fehler, wenn die gesamte Nachricht vertraulich ist.

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

Deny-Operation

Die Deny-Operation unterbricht entweder die `Publish`-API-Anfrage oder die Zustellung der Nachricht, wenn die Nachricht sensible Daten enthält. Das Deny-Operationsobjekt ist leer, da es keine zusätzliche Konfiguration erfordert.

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

Bei einer eingehenden Nachricht erhält der `SNS:Publish`-API-Aufrufer einen Autorisierungsfehler.

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

Bei einer ausgehenden Nachricht stellt das Amazon-SNS-Thema die Nachricht nicht dem Abonnement zu. Wenn Sie nicht autorisierte Zustellungen verfolgen möchten, aktivieren Sie die Protokollierung des Zustellungsstatus für das Thema. Im Folgenden finden Sie ein Beispiel für ein Zustellungsstatusprotokoll:

```
{
    "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"
}
```

Beispiele für Amazon SNS SNS-Datenschutzrichtlinien

Im Folgenden finden Sie Beispiele für Datenschutzrichtlinien, mit denen Sie sensible Daten überprüfen und ablehnen können. Ein vollständiges Tutorial, das eine Beispielanwendung enthält, finden Sie im Blogbeitrag Introducing message data protection for Amazon SNS (Einführung in den Nachrichtendatenschutz für Amazon SNS).

Beispielrichtlinie für das Prüfen

Mit Überwachungsrichtlinien können Sie bis zu 99% der eingehenden Nachrichten prüfen und Ergebnisse an Amazon CloudWatch und Amazon S3 senden. https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

Beispielsweise können Sie eine Prüfungsrichtlinie erstellen, um festzustellen, ob eines Ihrer Systeme versehentlich sensible Daten sendet oder empfängt. Wenn Ihre Prüfungsergebnisse zeigen, dass Systeme Kreditkarteninformationen an Systeme senden, die diese nicht benötigen, können Sie eine Datenschutzrichtlinie erstellen, um die Zustellung der Daten zu blockieren.

Im folgenden Beispiel werden 99% der Nachrichten geprüft, die das Thema behandeln, indem es nach Kreditkartennummern sucht und die Ergebnisse an CloudWatch Logs, Firehose und Amazon S3 sendet.

Datenschutzrichtlinie:

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

Beispiel für das Format von Prüfungsergebnissen:

```
{
    "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"
}
```

Beispielrichtlinie mit eingehender Anonymisierungsmaskierungsanweisung

Das folgende Beispiel verhindert durch die Maskierung der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

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

Beispiel für eingehende Anonymisierungsmaskierungsanweisung:

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

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

Beispielrichtlinie mit eingehender Anonymisierungsredigierungsanweisung

Das folgende Beispiel verhindert durch Redigieren der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

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

Beispiel für eingehende Anonymisierungs- und Redigierungsergebnisse:

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

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

Beispielrichtlinie mit ausgehender Anonymisierungsmaskierungsanweisung

Das folgende Beispiel verhindert durch die Maskierung der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

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

Beispiel für Ergebnisse einer Anonymisierungsmaskierungsanweisung:

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

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

Beispielrichtlinie mit ausgehender Anonymisierungsredaktionsanweisung

Das folgende Beispiel verhindert durch die Redigierung der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

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

Beispiel für Ergebnisse einer Anonymisierungsredaktionsanweisung:

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

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

Beispiel für eine Richtlinie mit einer Verweigerungsanweisung für eingehende Nachrichten

Das folgende Beispiel verhindert, dass ein Benutzer eine Nachricht mit `CreditCardNumber` im Nachrichteninhalt in einem Thema veröffentlicht. Abgelehnte Nutzlasten in der API-Antwort haben den Statuscode „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": {}
      }
    }
  ]
}
```

Beispiel für eine Richtlinie mit einer Verweigerungsanweisung für ausgehende Nachrichten

Im folgenden Beispiel wird verhindert, dass ein AWS Konto Nachrichten empfängt, die Folgendes `CreditCardNumber` enthalten:

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

Beispiel für ausgehende Ablehnungsresultate, bei Amazon CloudWatch angemeldet:

```
{
  "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"
}
```

Erstellen von Datenschutzrichtlinien in Amazon SNS

Datenschutzrichtlinien schützen die Daten, die in Ihren Amazon-SNS-Themen veröffentlicht werden, indem sensible Informationen, die zwischen Anwendungen oder AWS-Servicesübertragen werden geprüft, anonymisiert (maskiert oder redigiert) und verweigert (blockiert) werden. Sie können AWS API, AWS CLI, oder verwenden AWS CloudFormation, AWS Management Console um Datenschutzrichtlinien in Amazon SNS zu erstellen. Für jedes Amazon-SNS-Thema kann nur eine Richtlinie definiert werden. Die Datenschutzrichtlinie kann mindestens eine Anonymisierungs- und Verweigerungsanweisungen aber nur eine Prüfungsanweisung enthalten.

Themen

- Erstellen von Datenschutzrichtlinien in Amazon SNS mithilfe der API
- Erstellen von Datenschutzrichtlinien in Amazon SNS mithilfe der CLI
- Erstellen von Datenschutzrichtlinien in Amazon SNS mit CloudFormation
- Erstellen von Datenschutzrichtlinien in Amazon SNS mithilfe der Konsole
- Erstellung von Amazon SNS SNS-Datenschutzrichtlinien zur Sicherung von Nachrichtendaten mithilfe des SDK

Erstellen von Datenschutzrichtlinien in Amazon SNS mithilfe der API

Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellung einer Datenschutzrichtlinie mithilfe der API

Erstellen Sie mithilfe der AWS API eine Amazon SNS SNS-Datenschutzrichtlinie.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (AWS -API)

Verwenden Sie die `DataProtectionPolicy`-Eigenschaft eines Amazon-SNS-Standardthemas:

- `CreateTopic`

So erstellen oder rufen Sie eine Datenschutzrichtlinie für ein vorhandenes Amazon-SNS-Thema ab (AWS -API)

Rufen Sie eine der folgenden Operationen auf:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Erstellen von Datenschutzrichtlinien in Amazon SNS mithilfe der CLI

Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellung von Datenschutzrichtlinien mit dem AWS CLI

Erstellen Sie eine Amazon SNS SNS-Datenschutzrichtlinie mit dem AWS Command Line Interface.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (AWS CLI)

Verwenden Sie die folgende Option, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen:

- create-topic

So erstellen oder rufen Sie eine Datenschutzrichtlinie für ein vorhandenes Amazon-SNS-Thema ab (AWS CLI)

Rufen Sie eine der folgenden Operationen auf:

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

Erstellen von Datenschutzrichtlinien in Amazon SNS mit CloudFormation

Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellen von Datenschutzrichtlinien (CloudFormation)

Erstellen Sie eine Amazon SNS SNS-Datenschutzrichtlinie mit AWS CloudFormation.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (CloudFormation)

Verwenden Sie die folgende Option, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen:

- AWS::SNS::Topic

Erstellen von Datenschutzrichtlinien in Amazon SNS mithilfe der Konsole

Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (Konsole)

Verwenden Sie die folgende Option, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Thema aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Themen finden Sie unter Erstellen eines Amazon-SNS-Themas.
3. Wählen Sie auf der Seite Create topic (Thema erstellen) im Abschnitt Details die Option Standard aus.
   a. Geben Sie den Namen des neuen Themas ein.
   b. (Optional) Geben Sie einen Anzeigenamen für Ihr Thema ein.
4. Erweitern Sie Data protection policy (Datenschutzrichtlinie).

5. Wählen Sie einen Configuration mode (Konfigurationsmodus) aus:

   - Basic (Basismodus) – Definieren Sie eine Datenschutzrichtlinie über ein einfaches Menü.

   - Advanced (Erweiterter Modus) – Definieren Sie eine benutzerdefinierte Datenschutzrichtlinie mithilfe von JSON.

6. (Optional) Um Ihre eigene benutzerdefinierte Datenkennung zu erstellen, erweitern Sie den Abschnitt Custom data identifier configuration (Konfiguration für benutzerdefinierte Datenkennung) und gehen Sie wie folgt vor:

   a. Geben Sie einen eindeutigen Namen für die benutzerdefinierte Datenkennung ein. Benutzerdefinierte Datenkennungen können alphanumerische Zeichen, Unterstriche (_) und Bindestriche (-) enthalten. Es werden bis zu 128 Zeichen unterstützt. Dieser Name darf nicht denselben Namen wie eine verwaltete Datenkennung haben. Eine vollständige Liste der Einschränkungen für benutzerdefinierte Datenkennungen finden Sie unter Einschränkungen für benutzerdefinierte Datenkennungen.

   b. Geben Sie einen regulären Ausdruck (RegEx) für die benutzerdefinierte Daten-ID ein. RegExunterstützt alphanumerische Zeichen, RegEx reservierte Zeichen und Symbole. RegEx hat eine maximale Länge von 200 Zeichen. Wenn das zu kompliziert RegEx ist, schlägt Amazon SNS den API-Aufruf fehl. Eine vollständige Liste der RegEx Einschränkungen finden Sie unterEinschränkungen für benutzerdefinierte Datenkennungen.

   c. (Optional) Wählen Sie Add custom data identifier (Benutzerdefinierte Datenkennung hinzufügen), um bei Bedarf weitere Datenkennungen hinzuzufügen. Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.

7. Wählen Sie die Anweisung(en) aus, die Sie zu Ihrer Datenschutzrichtlinie hinzufügen möchten. Sie können derselben Datenschutzrichtlinie die Anwendungstypen audit (prüfen), de-identify (anonymisieren) (maskieren oder redigieren) und deny (verweigern) hinzufügen.

   a. Add audit statement (Audit-Anweisung hinzufügen) – Konfigurieren Sie, welche sensiblen Daten überprüft, wie viel Prozent der Nachrichten auf diese Daten überprüft und wohin Überwachungsprotokolle gesendet werden sollen.

   > **Note**
   >
   > Pro Datenschutzrichtlinie oder Thema ist nur eine Audit-Anweisung zulässig.

i. Wählen Sie data identifiers (Datenkennungen) aus, um die sensiblen Daten zu definieren, die Sie prüfen möchten.

ii. Geben Sie unter Audit sample rate (Audit-Samplerate) den Prozentsatz der Nachrichten ein, die auf sensible Informationen überprüft werden sollen (maximal 99 %).

iii. Wählen Sie unter Auditziel aus, AWS-Services wohin die Prüfungsergebnisse gesendet werden sollen, und geben Sie für jedes AWS-Service verwendete Ziel einen Zielnamen ein. Sie können einen der folgenden Amazon Web Services auswählen:

- Amazon CloudWatch — CloudWatch Logs ist die AWS Standard-Logging-Lösung. Mithilfe von CloudWatch Logs können Sie mithilfe von Logs Insights Protokollanalysen durchführen (Beispiele finden Sie hier) und Metriken und Alarme erstellen. CloudWatch In Logs veröffentlichen viele Dienste Protokolle, was es einfacher macht, alle Protokolle mit einer Lösung zu aggregieren. Informationen zu Amazon CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.
- — Firehose erfüllt die Anforderungen an Echtzeit-Streaming zu Splunk und Amazon Redshift für OpenSearch weitere Protokollanalysen. Weitere Informationen dazu finden Sie im Benutzerhandbuch.
- Amazon Simple Storage Service – Amazon S3 ist ein kostengünstiges Protokollziel für Archivierungszwecke. Möglicherweise müssen Sie Protokolle für einen Zeitraum von mehreren Jahren aufbewahren. In diesem Fall können Sie Protokolle in Amazon S3 speichern, um Kosten zu sparen. Weitere Informationen zu Amazon Simple Storage Service finden Sie im Benutzerhandbuch von Amazon Simple Storage Service.

b. Add a de-identify statement (Eine Anonymisierungsanweisung hinzufügen) – konfigurieren Sie die sensiblen Daten, die Sie in der Nachricht anonymisieren möchten, ob Sie diese Daten maskieren oder redigieren möchten, und die Konten, um die Zustellung dieser Daten zu beenden.

i. Wählen Sie bei data identifiers (Datenkennungen) die sensiblen Daten aus, die Sie prüfen möchten.

ii. Wählen Sie unter Diese Erklärung zur Anonymisierung definieren für die AWS Konten oder IAM-Prinzipale aus, für die diese Erklärung zur Anonymisierung gilt. Sie können sie auf alle AWS Konten oder auf bestimmte AWS Konten oder IAM-Entitäten (Kontostammdaten, Rollen oder Benutzer) anwenden, die ein Konto oder eine IAM-

Entität verwenden. IDs ARNs Trennen Sie mehrere IDs oder ARNs verwenden Sie ein Komma (,).

Die folgenden IAM-Prinzipale werden unterstützt:

- IAM account principals (IAM-Kontoprinzipale) – zum Beispiel `arn:aws:iam::AWS-account-ID:root`.
- IAM role principals (IAM-Rollenprinzipale) – zum Beispiel `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM user principals (IAM-Benutzerprinzipale) – zum Beispiel `arn:aws:iam::AWS-account-ID:user/user-name`.

iii. Wählen Sie bei De-identify Option (Anonymisierungsoption) die zu prüfenden sensiblen Daten aus. Die folgenden Optionen werden unterstützt:

- Redact (Redigieren) – löscht Daten vollständig. Beispielsweise wird die E-Mail-Adresse: `classified@amazon.com` zur E-Mail-Adresse: .
- Mask (Maskieren) – ersetzt die Daten durch einzelne Zeichen. Beispielsweise wird die E-Mail-Adresse: `classified@amazon.com` zur E-Mail-Adresse: `**********`.

iv. (Optional) Fügen Sie bei Bedarf weitere de-identify-Anweisungen hinzu.

c. Add deny statement (Verweigerungsanweisung hinzufügen) – legen Sie fest, welche sensiblen Daten sich nicht in Ihrem Thema bewegen dürfen und welche Prinzipale diese Daten nicht senden dürfen.

i. Wählen Sie bei data direction (Datenrichtung) die Richtung der Nachrichten für die Ablehnungsanweisung aus:

- Inbound messages (Eingehende Nachrichten) – Wenden Sie diese Deny-Anweisung auf Nachrichten an, die an das Thema gesendet werden.
- Outbound messages (Ausgehende Nachrichten) – Wenden Sie diese Deny-Anweisung auf Nachrichten an, die das Thema an Abonnementendpunkte sendet.

ii. Wählen Sie die data identifiers (Datenkennungen) aus, um die sensiblen Daten zu definieren, die Sie verweigern möchten.

iii. Wählen Sie die IAM principals (IAM-Prinzipale) aus, für die diese Verweigerungsanweisung gilt. Sie können es auf alle AWS Konten, auf bestimmte AWS-Konten oder auf IAM-Entitäten (z. B. Kontostammverzeichnisse, Rollen oder

Benutzer) anwenden, die Konten IDs oder IAM-Entitäten verwenden. ARNs Trennen Sie mehrere IDs oder ARNs verwenden Sie ein Komma (,). Die folgenden IAM-Prinzipale werden unterstützt:

- IAM-Kontoprinzipale – zum Beispiel `arn:aws:iam::AWS-account-ID:root`.
- IAM-Rollenprinzipale – zum Beispiel `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM-Benutzerprinzipale – zum Beispiel `arn:aws:iam::AWS-account-ID:user/user-name`.

iv. (Optional) Fügen Sie bei Bedarf weitere Deny-Anweisungen hinzu.

Erstellung von Amazon SNS SNS-Datenschutzrichtlinien zur Sicherung von Nachrichtendaten mithilfe des SDK

Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellung von Datenschutzrichtlinien mithilfe des SDK AWS

Erstellen Sie mithilfe des AWS SDK eine Amazon SNS SNS-Datenschutzrichtlinie.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (AWS SDK)

Verwenden Sie die folgenden Optionen, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen:

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();
```

So erstellen oder rufen Sie eine Datenschutzrichtlinie für ein vorhandenes Amazon-SNS-Thema ab (AWS SDK)

Verwenden Sie die folgenden Optionen, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen oder abzurufen:

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();
```

Löschen von Datenschutzrichtlinien in Amazon SNS

Sie können die Amazon SNS SNS-Datenschutzrichtlinien mithilfe der AWS API, AWS CLI AWS CloudFormation, oder AWS Management Console löschen.

Allgemeine Informationen zu Datenschutzrichtlinien von Amazon SNS finden Sie unter Grundlegendes zu den Datenschutzrichtlinien von Amazon SNS.

Anzahl und Größe von Amazon-SNS-Ressourcen für Datenschutzrichtlinien in einem AWS -Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon-SNS-API-Drosselung in Allgemeine AWS-Referenz.

Löschen von Datenschutzrichtlinien mithilfe der Konsole

Um eine verwaltete Datenschutzrichtlinie mithilfe der Konsole zu löschen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie das Thema aus, das die Datenschutzrichtlinie enthält, die Sie löschen möchten.
3. Wählen Sie Edit (Bearbeiten) aus.
4. Erweitern Sie den Abschnitt Data protection policy (Datenschutzrichtlinie).
5. Wählen Sie Remove (Entfernen) neben der Datenschutzrichtlinie aus, die Sie entfernen möchten.
6. Wählen Sie Änderungen speichern.

Löschen einer Datenschutzrichtlinie mit einer leeren JSON-Zeichenfolge

Sie können eine Datenschutzrichtlinie löschen, indem Sie sie auf eine leere JSON-Zeichenfolge aktualisieren.

Löschen einer Datenschutzrichtlinie mit dem AWS CLI

Sie können eine Datenschutzrichtlinie mit der AWS CLI löschen.

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

Amazon SNS SNS-Datenkennungen

Amazon SNS verwendet eine Kombination aus Kriterien und Techniken, einschließlich Machine Learning und Musterabgleich, um sensible Daten zu erkennen. Diese Kriterien und Techniken werden zusammenfassend als Datenkennungen bezeichnet. Sie können eine große und wachsende Liste vertraulicher Datentypen für viele Länder und Regionen erkennen. Amazon SNS verwaltete Datenkennungen bieten vorkonfigurierte Datentypen zum Schutz von Finanzdaten, persönlichen Gesundheitsinformationen (PHI) und persönlich identifizierbaren Informationen (PII). Sie können auch benutzerdefinierte Datenkennungen verwenden, um Ihre eigenen Datenkennungen zu erstellen, die auf Ihren speziellen Anwendungsfall zugeschnitten sind.

Using managed data identifiers in Amazon SNS

Was sind verwaltete Datenkennungen?

Von Amazon SNS verwaltete Datenkennungen dienen dazu, eine bestimmte Art sensibler Daten wie Kreditkartennummern, AWS geheime Zugangsschlüssel oder Passnummern für ein bestimmtes Land oder eine bestimmte Region zu erkennen. Beim Erstellen einer Datenschutzrichtlinie können Sie Amazon SNS so konfigurieren, dass diese Kennungen verwendet werden, um Nachrichten zu analysieren, die das Thema durchlaufen, und bei entsprechender Erkennung Maßnahmen zu ergreifen.

Amazon SNS kann mithilfe verwalteter Datenkennungen die folgenden Kategorien sensibler Daten erkennen:

- Anmeldeinformationen, wie private Schlüssel oder AWS geheime Zugangsschlüssel
- Gerätekennungen, wie IP-Adresse oder MAC-Adresse
- Finanzinformationen, wie Kreditkartennummern
- Gesundheitsinformationen für PHI, wie Krankenversicherungs- oder medizinische Identifikationsnummern
- Personenbezogene Daten für PII, wie Führerscheine oder Sozialversicherungsnummern

Innerhalb jeder Kategorie kann Amazon SNS mehrere Arten sensibler Daten erkennen. In den Themen in diesem Abschnitt werden die einzelnen Arten und alle relevanten Anforderungen für deren Erkennung aufgeführt und beschrieben. Für jede Art geben sie auch die eindeutige Kennung (ID) für die verwaltete Datenkennung an, die für die Erkennung der Daten konzipiert ist. Wenn Sie eine

Datenschutzrichtlinie erstellen, können Sie diese ID verwenden, um die verwaltete Datenkennung für die Erkennung durch den Nachrichtendatenschutz einzuschließen.

Anforderungen an Schlüsselwörter

Um bestimmte Arten sensibler Daten zu erkennen, sucht Amazon SNS in der Nähe der Daten nach Schlüsselwörtern. Wenn dies bei einem bestimmten Datentyp der Fall ist, werden in einem nachfolgenden Thema in diesem Abschnitt spezifische Schlüsselwortanforderungen für diese Daten angegeben.

Bei Schlüsselwörtern muss die Groß- und Kleinschreibung nicht beachtet werden. Wenn ein Schlüsselwort ein Leerzeichen enthält, gleicht Amazon SNS außerdem automatisch Schlüsselwortvarianten ab, die das Leerzeichen nicht enthalten oder anstelle des Leerzeichens einen Unterstrich (_) oder einen Bindestrich (-) enthalten. In bestimmten Fällen erweitert oder kürzt Amazon SNS ein Schlüsselwort auch, um gängige Variationen des Schlüsselworts zu berücksichtigen.

Von Amazon SNS verwaltete Datenkennungen für sensible Datentypen

In der folgenden Liste werden die Arten von Anmeldeinformationen, Geräte-, Finanz- und medizinische Daten sowie persönliche Gesundheitsinformationen (PHI) aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann. Diese gelten zusätzlich zu bestimmten Arten von Daten, die auch als persönlich identifizierbare Informationen (PII) gelten können.

Für regionsabhängige Datenkennungen sind der Kennungsname mit einem Bindestrich und die aus zwei Buchstaben bestehenden Codes (ISO 3166-1 Alpha-2) erforderlich. Zum Beispiel DriversLicense -US.

Kennung	Kategorie	Länder/Sprachen
BankAccountNumber	Finanzanwendungen	DE, ES, FR, GB, IT
CepCode	Persönlich	BR
Cnpj	Persönlich	BR
CpfCode	Persönlich	BR
DriversLicense	Persönlich	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB,

Kennung	Kategorie	Länder/Sprachen
		GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Gesundheit	US
ElectoralRollNumber	Persönlich	GB
HealthInsuranceCardNumber	Gesundheit	EU
HealthInsuranceClaimNumber	Gesundheit	US
HealthInsuranceNumber	Gesundheit	FR
HealthcareProcedureCode	Gesundheit	US
IndividualTaxIdentificationNumber	Persönlich	US
InseeCode	Persönlich	FR
MedicareBeneficiaryNumber	Gesundheit	US
NationalDrugCode	Gesundheit	US
NationalIdentificationNumber	Persönlich	DE, ES, IT
NationalInsuranceNumber	Persönlich	GB
NationalProviderId	Gesundheit	US
NhsNumber	Gesundheit	GB
NieNumber	Persönlich	ES
NifNumber	Persönlich	ES
PassportNumber	Persönlich	CA, DE, ES, FR, GB, IT, US

Kennung	Kategorie	Länder/Sprachen
PermanentResidenceNumber	Persönlich	CA
PersonalHealthNumber	Gesundheit	CA
PhoneNumber	Persönlich	BR, DE, ES, FR, GB, IT, US
PostalCode	Persönlich	CA
RgNumber	Persönlich	BR
SocialInsuranceNumber	Persönlich	CA
Ssn	Persönlich	ES, US
TaxId	Persönlich	DE, ES, FR, GB
ZipCode	Persönlich	US

Unterstützte Kennungen, die sprach-/regionsunabhängig sind

Kennung	Kategorie
Adresse	Persönlich
AwsSecretKey	Anmeldeinformationen
CreditCardExpiration	Finanzanwendungen
CreditCardNumber	Finanzanwendungen
CreditCardSecurityCode	Finanzanwendungen
EmailAddress	Persönlich
IpAddress	Persönlich
LatLong	Persönlich

Kennung	Kategorie
Name	Persönlich
OpenSshPrivateKey	Anmeldeinformationen
PgpPrivateKey	Anmeldeinformationen
PkcsPrivateKey	Anmeldeinformationen
PuttyPrivateKey	Anmeldeinformationen
VehicleIdentificationNumber	Persönlich

Vertrauliche Amazon SNS SNS-Datentypen: Anmeldeinformationen

In der folgenden Liste werden die Arten von Anmeldeinformationen aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
AWS geheimer Zugriffsschlüssel	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Any
Privater OpenSSH-Schlüssel	OpenSshPrivateKey	Nein	Any
Privater PGP-Schlüssel	PgpPrivateKey	Nein	Any
Privater Public Key Cryptography Standard (PKCS)-Schlüssel	PkcsPrivateKey	Nein	Any

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Privater PuTTY-Schlüssel	PuttyPrivateKey	Nein	Any

Datenbezeichner ARNs für Datentypen mit Anmeldeinformationen

Im Folgenden sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

Daten-ID für Anmeldeinformationen ARNs
arn:aws:dataprotection: :aws:data-identifier/ AwsSecretKey
arn:aws:datenschutz: :aws:data-identifier/ OpenSshPrivateKey
arn:aws:datenschutz: :aws:data-identifier/ PgpPrivateKey
arn:aws:datenschutz: :aws:data-identifier/ PkcsPrivateKey
arn:aws:datenschutz: :aws:data-identifier/ PuttyPrivateKey

Vertrauliche Amazon SNS SNS-Datentypen: Geräte

In der folgenden Liste werden die Arten von Gerätekennungen aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
IP-Adresse	IpAddress	Nein	Any

Daten-ID ARNs für Gerätedatentypen

Im Folgenden sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

Gerätedatenkennungs-ARN
arn:aws:dataprotection: :aws:data-identifier/ IpAddress

Vertrauliche Amazon SNS SNS-Datentypen: Finanziell

In der folgenden Liste werden die Arten von Finanzdaten aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Bankkontonummer	BankAccountNumber BankAccountNumber-US	Ja, siehe Schlüsselwörter für Bankkontonummern.	Dazu gehören: Internationale Bankkontonummern (IBANs), die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie der Landesvorwahl.	Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich
Ablaufdatum der Kreditkarte	CreditCardExpiration	exp d, exp m, exp y, expiration, expiry	–	Any
Daten des Kreditkarten-Magnetstreifens	CreditCardMagneticStripe	Ja, einschließlich: Kartendaten, ISO7813, MAG, Magnetstr	Dies beinhaltet die Titel 1 und 2.	Any

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
		eifen, Streifen, streichen.		
Kreditkartennummer	CreditCardNumber	Kontonummer, American Express, Amex, Bankkarte, Karte, Karten-Nr., Kartennummer, cc #, CCN, Scheckkarte, Kredit, Kreditkartennummer, Dankort, Debit, Debitkarte, Diners Club, Discover, Electron, Elo-Verifizierungscode, Japanese Card Bureau, JCB, Mastercard, mc, pan, Zahlungskontonummer, Zahlungskartennummer, PCN, Union Pay, Visa	Für die Erkennung müssen die Daten eine 13—19-stellige Sequenz sein, die der Luhn-Scheckformel entspricht und ein Standard-Kartennummernpräfix für jede der folgenden Arten von Kreditkarten verwendet: American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card Bureau (JCB) UnionPay, Mastercard und Visa (hochgestellter Link unter 1).	Any

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Bestätigungscode für die Kreditkarte	CreditCardSecurityCode	Karten-ID, Kartenidentifikationscode, Kartenidentifikationsnummer, Kartensicherheitscode, Kartenvalidierungscode, Kartenvalidierungsnummer, Kartenüberprüfungsdaten, Kartenüberprüfungswert, cvc, cvc2, cvv, cvv2, elo-Bestätigungscode	–	Any

1. Amazon SNS meldet keine Vorkommen der folgenden Sequenzen, die Kreditkartenaussteller für öffentliche Tests reserviert haben:

   122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,

5204740009900014, 5420923878724339, 5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194, 5555555555554444, 5610591081018250, 6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441, 630495060000000000, 6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.

Schlüsselwörter für Bankkontonummern

Verwenden Sie die folgenden Schlüsselwörter, um internationale Bankkontonummern (IBANs) zu ermitteln, die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie der Landesvorwahl.

Land oder Region	Schlüsselwörter			
Frankreich	account code, account number, accountno #, accountnu mber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
Deutschland	account code, account number, accountno #, accountnu mber#, bankleitz			

Land oder Region	Schlüsselwörter			
	ahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
Italien	account code, account number, accountno #, accountnumber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

Land oder Region	Schlüsselwörter			
Spanien	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			

Land oder Region	Schlüsselwörter			
US	Bankkonto , Bankkto, Girokonto , Girokto, Depotkonto, Depositkto, Sparkonto, Sparkto			

Daten-ID ARNs für Finanzdatentypen

Im Folgenden sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

Kennung für Finanzdaten ARNs
arn:aws:dataprotection: :aws:data-identifier/ -DE BankAccountNumber
arn:aws:datenschutz: BankAccountNumber :aws:data-identifier/ -ES
arn:aws:datenschutz: BankAccountNumber :aws:data-identifier/ -FR
arn:aws:datenschutz: BankAccountNumber :aws:data-identifier/ -GB
arn:aws:datenschutz: BankAccountNumber :aws:data-identifier/ -IT
arn:aws:datenschutz: BankAccountNumber :aws:data-identifier/ -US
arn:aws:datenschutz: :aws:data-identifier/ CreditCardExpiration
arn:aws:datenschutz: :aws:data-identifier/ CreditCardNumber
arn:aws:datenschutz: :aws:data-identifier/ CreditCardSecurityCode

Vertrauliche Amazon SNS SNS-Datentypen: Geschützte Gesundheitsinformationen (PHI)

In der folgenden Liste werden die Arten von geschützten Gesundheitsdaten aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Registrierungsnummer der Drug Enforcement Agency (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	US
Nummer der Krankenversicherungskarte (EHIC)	HealthInsuranceCardNumber	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	EU

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
		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, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	US
Krankenversicherungs- oder medizinische Identifizierungsnummer	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Standardisierte Codes für medizinische Leistungen (HCPCS)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	US
Medicare-Versichertennummer (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	US
National Drug Code (NDC)	NationalDrugCode	national drug code, ndc	US
National Provider Identifier (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	US
Nummer des Nationalen Gesundheitsdienstes (NHS)	NhsNumber	national health service, NHS	GB
Persönliche Gesundheitsnummer (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Schlüsselwörter für Krankenversicherungs- oder medizinische Identifizierungsnummern

Damit Amazon SNS verschiedene Arten von Krankenversicherungs- und medizinischen Identifizierungsnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. Dazu gehören die Nummern der Europäischen Krankenversicherungskarte (EU, Finnland), Krankenversicherungsnummern (Frankreich), Medicare-Versichertennummern (USA), nationale Versicherungsnummern (UK), NHS-Nummern (UK) und persönliche Gesundheitsnummern (Kanada).

In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Kanada	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, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
Finnland	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

Land oder Region	Schlüsselwörter
Frankreich	carte d'assuré social, carte vitale, insurance card
Vereinigtes Königreich	nationaler Gesundheitsdienst, NHS
US	mbi, medicare beneficiary

Daten-ID ARNs für Datentypen mit geschützten Gesundheitsinformationen (PHI)

Im Folgenden ist der Datenbezeichner Amazon Resource Names (ARNs) aufgeführt, der in PHI-Datenschutzrichtlinien verwendet werden kann.

PHI-Datenbezeichner ARNs
arn:aws:dataprotection: :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:datenschutz: HealthcareProcedureCode :aws:data-identifier/ -US
arn:aws:datenschutz: HealthInsuranceCardNumber :aws:data-identifier/ -EU
arn:aws:datenschutz: HealthInsuranceClaimNumber :aws:data-identifier/ -US
arn:aws:datenschutz: HealthInsuranceNumber :aws:data-identifier/ -FR
arn:aws:datenschutz: MedicareBeneficiaryNumber :aws:data-identifier/ -US
arn:aws:datenschutz: NationalDrugCode :aws:data-identifier/ -US
arn:aws:datenschutz: NationalInsuranceNumber :aws:data-identifier/ -GB
arn:aws:datenschutz: NationalProviderId :aws:data-identifier/ -US
arn:aws:datenschutz: NhsNumber :aws:data-identifier/ -GB
arn:aws:datenschutz: PersonalHealthNumber :aws:data-identifier/ -CA

Vertrauliche Amazon SNS SNS-Datentypen: Persönlich identifizierbare Informationen (PII)

In der folgenden Liste werden die Arten von persönlich identifizierbaren Informationen (PII) aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Geburtsdatum	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	Der Support umfasst die meisten Datumsformate, z. B. nur Ziffern und Kombinationen aus Ziffern und Monatsnamen. Datumskomponenten können durch Leerzeichen, Schrägstriche (/) oder Bindestriche (-) getrennt werden.	Any
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brasilien
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro	–	Brasilien

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
		nacional da pessoa juridica, cnpj		
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	Brasilien

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Identifikationsnummer des Führerscheins	DriversLicense	Ja, siehe Schlüsselwörter für Identifikationsnummern des Führerscheins.	–	Australien, Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Italien, Kanada, Kroatien, Lettland, Litauen, Luxemburg, Malta, Niederlande, Österreich, Polen, Portugal, Rumänien, Schweden, Slowakei, Slowenien, Spanien, Tschechische Republik, Ungarn, USA, Vereinigtes Königreich, Zypern

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Nummer der Wählerliste	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
Individuelle Steuerpflichtigen-ID	IndividualTaxIdentificationNumber	Ja, siehe Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern.	–	US
Nationales Institut für Statistik und Wirtschaftsstudien (INSEE)	InseeCode	Ja, siehe Schlüsselwörter für nationale Identifikationsnummern.	–	Frankreich

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Nationale Identifikationsnummern	NationalIdentificationNumber	Ja, siehe Schlüsselwörter für nationale Identifikationsnummern.	Dazu gehören Kennungen des Documento Nacional de Identidad (DNI) (Spanien), Codice fiscale codes (Italien) und Personalausweisnummern (Deutsch).	Deutschland, Italien, Spanien
Landesversicherungsnummer (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	UK
Número de identidad de extranjero (NIE)	NieNumber	Ja, siehe Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern.	–	Spanien

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Número de Identificación Fiscal (NIF)	NifNumber	Ja, siehe Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern.	–	Spanien
Passnummer	PassportNumber	Ja, siehe Schlüsselwörter für Passnummern.	–	Kanada, Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich, USA

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Ständige Wohnsitznummer	Permanent Residence Number	carte résident permanent, numéro carte résident permanent, numéro résident permanent, permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	–	Kanada

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Phone number (Telefonnummer)	PhoneNumber	Brasilien: Zu den Schlüsselwörtern gehören auch: cel, celular, fone, móvel, número residencial, numero residencial, telefone Andere: cell, contact, fax, fax number, mobile, phone, phone number, tel, telephone, telephone number	Dazu gehören gebührenfreie Nummern in den USA und Faxnummern. Wenn sich ein Schlüsselwort in der Nähe der Daten befindet, muss die Nummer keine Landesvorwahl enthalten. Wenn sich ein Schlüsselwort nicht in der Nähe der Daten befindet, muss die Nummer eine Landesvorwahl enthalten.	Brasilien, Kanada, Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich, USA
Postal Code (Postleitzahl)	PostalCode	No	–	Kanada
Registro Geral (RG)	RgNumber	Ja, siehe Schlüsselwörter für nationale Identifikationsnummern.	–	Brasilien

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Sozialversicherungsnummer (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Kanada
Sozialversicherungsnummer (SSN)	Ssn	Spanien – número de la seguridad social, social security no., social security no. número de la seguridad social, social security number, socialsecurityno#, ssn, ssn# USA: social security, ss#, ssn	–	Spanien, USA
Steuerpflichtigen-Identifikationsnummer oder Referenznummer	TaxId	Ja, siehe Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern.	Dazu gehören TIN (Frankreich), Steueridentifikationsnummer (Deutschland), CIF (Spanien) und TRN, UTR (UK).	Frankreich, Deutschland, Spanien, Vereinigtes Königreich

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
US-Postleitzahl	ZipCode	zip code, zip+4	–	US
Postanschrift	Address	No	Obwohl ein Schlüsselwort nicht erforderlich ist, muss die Adresse bei der Erkennung den Namen einer Stadt oder eines Ortes sowie einen Zip-Code oder eine Postleitzahl enthalten.	Australien, Kanada, Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich, USA
E-Mail-Adresse	EmailAddress	Email, Email-Adresse, E-Mail, E-Mail-Adresse	–	Any

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Koordinaten des globalen Positionierungssystems (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS kann GPS-Koordinaten erkennen, wenn die Breiten- und Längengradkoordinaten paarweise gespeichert sind und im DD-Format (Dezimalstelle, Grad) vorliegen, z. B. 41,948614, -87,655311. Nicht unterstützt werden Koordinaten im DDM-Format (Grad, Dezimalstelle, Minuten), z. B. 41°56.9168'N 87°39.3187'W, oder im DMS-Format (Grad, Minuten, Sekunden), z. B. 41°56'55.0104"N 87°39'19.1196"W.	Any

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Vollständiger Name	Name	No	Amazon SNS kann nur vollständige Namen erkennen. Unterstützt werden nur lateinische Zeichensätze.	Any
Fahrgestellnummern (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 kann erkennen VINs , dass diese aus einer Sequenz mit 17 Zeichen bestehen und den Normen ISO 3779 und 3780 entsprechen. Diese Standards wurden für den weltweiten Einsatz konzipiert.	Any

Schlüsselwörter für Identifikationsnummern des Führerscheins

Damit Amazon SNS verschiedene Arten von Führerschein-Identifikationsnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Australien	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
Österreich	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Belgien	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgarien	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Kanada	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,

Land oder Region	Schlüsselwörter
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
Kroatien	vozačka dozvola
Zypern	άδεια οδήγησης
Tschechische Republik	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
Dänemark	kørekort, kørekortnummer
Estland	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finnland	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
Frankreich	permis de conduire
Deutschland	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Griechenland	δεια οδήγησης, adeia odigisis
Ungarn	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Irland	ceadúnas tiomána
Italien	patente di guida, patente di guida numero, patente guida, patente guida numero

Land oder Region	Schlüsselwörter
Lettland	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Litauen	vairuotojo pažymėjimas
Luxemburg	fahrerlaubnis, führerschäin
Malta	liċenzja tas-sewqan
Niederlande	permis de conduire, rijbewijs, rijbewijsnummer
Polen	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
Rumänien	numărul permisului de conducere, permis de conducere
Slowakei	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Slowenien	vozniško dovoljenje

Land oder Region	Schlüsselwörter
Spanien	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
Schweden	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.
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

Schlüsselwörter für nationale Identifikationsnummern

Damit Amazon SNS verschiedene Arten von nationalen Identifikationsnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. Dazu gehören Kennungen des

Documento Nacional de Identidad (DNI) (Spanien), Codes des französischen Nationalen Instituts für Statistik und Wirtschaftsstudien (INSEE), deutsche Personalausweisnummern und Registro Geral (RG)-Nummern (Brasilien).

In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Brasilien	registro geral, rg
Frankreich	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Deutschland	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italien	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Spanien	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#

Schlüsselwörter für Passnummern

Damit Amazon SNS verschiedene Arten von Passnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Kanada	passeport, passeport#, passport, passport#, passportno, passportno#
Frankreich	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
Deutschland	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Italien	italian passport number, numéro passeport , numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Spanien	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

Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern

Damit Amazon SNS verschiedene Arten von Steuerpflichtigenidentifikations- und Referenznummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Brasilien	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
Frankreich	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Deutschland	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
Spanien	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	Individuelle Steuerzahler-Identifikationsnummer (ITIN)

Daten-ID ARNs für persönlich identifizierbare Informationen (PII)

In der folgenden Tabelle sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

PII-Daten-ID ARNs
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection: :aws:data-identifier/ -BR CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:datenschutz: CpfCode :aws:data-identifier/ -BR
arn:aws:datenschutz: :aws:data-identifier/ DateOfBirth
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -AT
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -AU
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -BE
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -BG
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -CA
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -CY
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -CZ
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -DE
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -DK
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -EE
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -ES
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -FI
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -FR

PII-Daten-ID ARNs
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -GB
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -GR
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -HR
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -HU
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -IE
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -IT
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -LT
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -LU
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -LV
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -MT
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -NL
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -PL
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -PT
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -RO
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -SE
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -SI
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -SK
arn:aws:datenschutz: DriversLicense :aws:data-identifier/ -US
arn:aws:datenschutz: ElectoralRollNumber :aws:data-identifier/ -GB
arn:aws:datenschutz: :aws:data-identifier/ EmailAddress

PII-Daten-ID ARNs
arn:aws:datenschutz: IndividualTaxIdentificationNumber :aws:data-identifier/ -US
arn:aws:datenschutz: InseeCode :aws:data-identifier/ -FR
arn:aws:datenschutz: :aws:data-identifier/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:datenschutz: NationalIdentificationNumber :aws:data-identifier/ -DE
arn:aws:datenschutz: NationalIdentificationNumber :aws:data-identifier/ -ES
arn:aws:datenschutz: NationalIdentificationNumber :aws:data-identifier/ -IT
arn:aws:datenschutz: NieNumber :aws:data-identifier/ -ES
arn:aws:datenschutz: NifNumber :aws:data-identifier/ -ES
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -CA
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -DE
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -ES
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -FR
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -GB
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -IT
arn:aws:datenschutz: PassportNumber :aws:data-identifier/ -US
arn:aws:datenschutz: PermanentResidenceNumber :aws:data-identifier/ -CA
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -BR
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -DE
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -ES

PII-Daten-ID ARNs
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -FR
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -GB
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -IT
arn:aws:datenschutz: PhoneNumber :aws:data-identifier/ -US
arn:aws:datenschutz: PostalCode :aws:data-identifier/ -CA
arn:aws:datenschutz: RgNumber :aws:data-identifier/ -BR
arn:aws:datenschutz: SocialInsuranceNumber :aws:data-identifier/ -CA
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:datenschutz: TaxId :aws:data-identifier/ -DE
arn:aws:datenschutz: TaxId :aws:data-identifier/ -ES
arn:aws:datenschutz: TaxId :aws:data-identifier/ -FR
arn:aws:datenschutz: TaxId :aws:data-identifier/ -GB
arn:aws:datenschutz: :aws:data-identifier/ VehicleIdentificationNumber
arn:aws:datenschutz: ZipCode :aws:data-identifier/ -US

Verwenden von benutzerdefinierten Datenkennungen in Amazon SNS

Mit benutzerdefinierten Datenbezeichnern (CDIs) können Sie Ihre eigenen benutzerdefinierten regulären Ausdrücke definieren, die in Ihrer Datenschutzrichtlinie verwendet werden können. Mithilfe von benutzerdefinierten Datenkennungen können Sie gezielt auf geschäftsspezifische Anwendungsfälle mit persönlich identifizierbaren Informationen (PII) abzielen, die verwaltete Datenkennungen nicht bieten können. Sie können beispielsweise eine benutzerdefinierte Daten-ID

verwenden, um nach einem unternehmensspezifischen Mitarbeiter zu suchen. IDs Benutzerdefinierte Datenkennungen können in Verbindung mit verwalteten Datenkennungen verwendet werden.

Was sind benutzerdefinierte SNS-Datenkennungen?

Mit benutzerdefinierten Datenbezeichnern (CDIs) können Sie Ihre eigenen benutzerdefinierten regulären Ausdrücke definieren, die in Ihrer Datenschutzrichtlinie verwendet werden können. Mithilfe von benutzerdefinierten Datenkennungen können Sie gezielt auf geschäftsspezifische Anwendungsfälle mit persönlich identifizierbaren Informationen (PII) abzielen, die verwaltete Datenkennungen nicht bieten können. Sie können beispielsweise eine benutzerdefinierte Daten-ID verwenden, um nach einem unternehmensspezifischen Mitarbeiter zu suchen. IDs Benutzerdefinierte Datenkennungen können in Verbindung mit verwalteten Datenkennungen verwendet werden.

Verwenden benutzerdefinierter Datenkennungen in Ihrer Datenschutzrichtlinie

Die folgende Datenschutzrichtlinie weist das Amazon SNS SNS-Thema an, Nutzlasten zu erkennen, die unternehmensspezifische Mitarbeiter enthalten, und diese dann IDs mit dem IDs Hash-Symbol (#) zu maskieren.

1. Erstellen Sie einen `Configuration`-Block innerhalb Ihrer Datenschutzrichtlinie.
2. Geben Sie eine `Name` als benutzerdefinierte Datenkennung ein. Beispiel, **`EmployeeId`**.
3. Geben Sie eine `Regex` als benutzerdefinierte Datenkennung ein. Beispiel, **`EID-\d{9}-US`**.
4. Beziehen Sie sich in einer Richtlinienerklärung auf die folgende benutzerdefinierte Datenkennung.

```
{
 "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. (Optional) Fügen Sie dem `Configuration`-Block bei Bedarf weitere benutzerdefinierte Datenkennungen hinzu. Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.

Einschränkungen für benutzerdefinierte Datenkennungen

Benutzerdefinierte Datenkennungen von Amazon SNS haben die folgenden Einschränkungen:

- Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.
- Die Namen von benutzerdefinierten Datenkennungen sind auf maximal 128 Zeichen beschränkt. Folgende Zeichen werden unterstützt:
  - Alphanumerisch: (a-zA-Z0-9)
  - Sonderzeichen: („_“ | „-“)
- RegEx ist auf maximal 200 Zeichen beschränkt. Folgende Zeichen werden unterstützt:
  - Alphanumerisch: (a-zA-Z0-9)
  - Sonderzeichen: („_“ | „#“ | „=“ | „@“ | „/“ | „;“ | „,“ | „-“ | „ “)
  - RegEx reservierte Zeichen: („^“ | „$“ | „?“ | „[“ | „]“ | „{“ | „}“ | „|“ | „\\“ | „*“ | „+“ | „.“)
- Benutzerdefinierte Datenkennungen können nicht denselben Namen wie verwaltete Datenkennungen haben.
- Benutzerdefinierte Datenkennungen müssen in jeder Datenschutzrichtlinie für jedes Amazon SNS-Thema angegeben werden.

Nachrichtenzustellung von Amazon SNS

In diesem Thema wird beschrieben, wie Amazon SNS die Nachrichtenzustellung in verschiedenen Szenarien handhabt. Sie erfahren mehr über die Zustellung von Rohnachrichten, bei der Amazon SNS Nachrichten in ihrem ursprünglichen, unveränderten Format an den Endpunkt übermittelt. Außerdem erfahren Sie, wie Sie Nachrichten von einem Amazon SNS-Thema an eine Amazon SQS SQS-Warteschlange in einem anderen Format senden können. So erhalten Sie Einblicke in AWS-Konto kontoübergreifendes Messaging.

Dieses Thema bietet verschiedene Informationen zur Zustellung von Amazon SNS SNS-Nachrichten an eine Amazon SQS SQS-Warteschlange oder eine Lambda-Funktion AWS-Regionen, zur Funktionsweise der regionsübergreifenden Zustellung und zu den damit verbundenen Überlegungen.

Darüber hinaus erfahren Sie, wie Sie den Status der Nachrichtenzustellung überwachen und interpretieren können. Dieser enthält wichtige Informationen darüber, ob Nachrichten erfolgreich zugestellt wurden oder ob Probleme aufgetreten sind. In Fällen, in denen die Nachrichtenzustellung fehlschlägt, kennen Sie den Wiederholungsversuch bei der Nachrichtenzustellung, einschließlich der Tatsache, dass Amazon SNS automatisch versucht, Nachrichten erneut zuzustellen, um sicherzustellen, dass sie ihren Bestimmungsort erreichen. In diesem Thema wird auch die Verwendung von Warteschlangen für unzustellbare Nachrichten zur Erfassung von Nachrichten behandelt, die nach mehreren Versuchen nicht zugestellt werden konnten, sodass Sie diese Fehler effektiv analysieren und beheben können.

Übermittlung unformatierter Nachrichten Amazon SNS

Um zu vermeiden???, dass Amazon SQS - und HTTP/S-Endpunkte die JSON-Formatierung von Nachrichten verarbeiten müssen, ermöglicht Amazon SNS die Zustellung von Rohnachrichten:

- Wenn Sie die Zustellung von Rohnachrichten für Amazon SQS SQS-Endpunkte aktivieren, werden alle Amazon SNS SNS-Metadaten aus der veröffentlichten Nachricht entfernt und die Nachricht wird unverändert gesendet.
- Wenn Sie die Zustellung von Rohnachrichten für HTTP/S Endpunkte aktivieren, `true` wird der Nachricht der HTTP-Header `x-amz-sns-rawdelivery` mit dem Wert auf hinzugefügt, was darauf hinweist, dass die Nachricht ohne JSON-Formatierung veröffentlicht wurde.
- Wenn Sie die Übermittlung von Rohnachrichten für HTTP/S Endpunkte aktivieren, werden der Nachrichtentext, die Client-IP und die erforderlichen Header zugestellt. Wenn Sie Nachrichtenattribute angeben, werden diese nicht gesendet.

- Wenn Sie die Übermittlung von Rohnachrichten für Firehose-Endpunkte aktivieren, wird der Nachrichtentext zugestellt. Wenn Sie Nachrichtenattribute angeben, werden diese nicht gesendet.

Um die Zustellung von Rohnachrichten mithilfe eines AWS SDK zu aktivieren, müssen Sie die `SetSubscriptionAttribute` API-Aktion verwenden und den Wert des `RawMessageDelivery` Attributs auf setzen. `true`

Aktivieren der Übermittlung unformatierter Nachrichten mit der AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Themen ein Thema aus, das einen Firehose-, Amazon SQS- oder Endpunkt abonniert hat. HTTP/S
4. Wählen Sie auf der ***MyTopic*** Seite im Abschnitt Abonnement ein Abonnement aus und klicken Sie auf Bearbeiten.
5. Wählen Sie auf der *EXAMPLE1-23bc-4567-d890-ef12g3hij456* Seite Bearbeiten im Abschnitt Details die Option Zustellung von Rohnachrichten aktivieren aus.
6. Wählen Sie Save Changes (Änderungen speichern).

Nachrichtenformat – Beispiele

In den folgenden Beispielen wird dieselbe Nachricht zweimal an dieselbe Amazon SQS Warteschlange gesendet. Der einzige Unterschied besteht darin, dass die unformatierte Nachrichtenübermittlung für die erste Nachricht deaktiviert und für die zweite aktiviert ist.

- Übermittlung von unformatierten Nachrichten ist deaktiviert

```
{
  "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"
}
```

- Übermittlung von unformatierten Nachrichten ist aktiviert

```
This is a test message.
```

Nachrichtenattribute und Zustellung von Rohnachrichten für Amazon SQS SQS-Abonnements

Amazon SNS unterstützt die Übermittlung von Nachrichtenattributen, mit denen Sie strukturierte Metadatenelemente wie Zeitstempel, Geodaten, Signaturen und Kennungen zur Nachricht bereitstellen können. Bei Amazon SQS SQS-Abonnements mit aktivierter Raw Message Delivery können maximal 10 Nachrichtenattribute gesendet werden. Um mehr als 10 Nachrichtenattribute zu senden, müssen Sie Raw Message Delivery deaktivieren. Amazon SNS verwirft jedoch Nachrichten mit mehr als 10 Nachrichtenattributen, die an Amazon SQS SQS-Abonnements mit aktivierter Raw Message Delivery gerichtet sind, und behandelt sie als clientseitige Fehler.

Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange in einem anderen Konto

In diesem Dokument wird beschrieben, wie Sie eine Benachrichtigung zu einem Amazon SNS-Thema mit einem oder mehreren Abonnements für Amazon SQS-Warteschlangen in einem anderen Konto veröffentlichen. Sie richten das Thema und die Warteschlangen so ein, als wenn sie sich im selben Konto befinden würden (siehe Fanout von Amazon SNS SNS-Benachrichtigungen an Amazon SQS SQS-Warteschlangen zur asynchronen Verarbeitung). Der größte Unterschied besteht in der

Handhabung der Abonnementbestätigung, die davon abhängt, wie Sie die Warteschlange für das Thema abonnieren.

Es empfiehlt sich, möglichst die Schritte zu befolgen, auf die im Abschnitt Abonnement erstellt durch Warteschlangeneigentümer verwiesen wird, da die Bestätigung automatisch erfolgt, wenn der Eigentümer der Warteschlange das Abonnement erstellt.

> **Note**
>
> Wenn die Amazon-SQS-Warteschlange ein hohes Nachrichtenvolumen enthält, empfehlen wir, dass der Warteschlangenbesitzer das Abonnement erstellt.

Abonnement erstellt durch Warteschlangeneigentümer

Das Konto, das die Amazon SQS-Warteschlange erstellt hat, ist der Besitzer der Warteschlange. Wenn der Eigentümer der Warteschlange ein Abonnement erstellt, muss das Abonnement nicht bestätigt werden. Die Warteschlange beginnt mit dem Empfang von Benachrichtigungen aus dem Thema, sobald die Aktion `Subscribe` abgeschlossen ist. Damit der Warteschlangeneigentümer das Thema des Themeneigentümers abonnieren kann, muss der Themeneigentümer die Kontoberechtigung des Warteschlangeneigentümers zum Aufrufen der Aktion `Subscribe` für das Thema erteilen.

Schritt 1: So legen Sie die Themen-Richtlinie mithilfe der AWS Management Console fest

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie ein Thema und dann Edit (Bearbeiten) aus.
4. Erweitern Sie auf der *`MyTopic`* Seite Bearbeiten den Abschnitt Zugriffsrichtlinie.
5. Geben Sie die folgende Richtlinie ein:

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
```

```
            },
            "Action": "sns:Subscribe",
            "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
    ]
}
```

Diese Richtlinie gewährt Konto `111122223333` die Berechtigung zum Anruf von `sns:Subscribe` über `MyTopic` in Konto `123456789012`.

Ein Benutzer mit den Anmeldeinformationen für das Konto `111122223333` kann `MyTopic` abonnieren. Diese Berechtigung ermöglicht es der Konto-ID, die Berechtigung an ihren IAM-Benutzer/ihre IAM-Rolle zu delegieren. Nur das Root-Konto oder die Administratorbenutzer dürfen `sns:Subscribe` aufrufen. Der IAM-Benutzer/die IAM-Rolle muss ebenfalls `sns:subscribe` aufrufen, damit seine/ihre Warteschlange abonniert werden kann.

6. Wählen Sie Änderungen speichern.

   Ein Benutzer mit den Anmeldeinformationen für das Konto `111122223333` kann abonnieren MyTopic.

Schritt 2: Um ein Amazon SQS SQS-Warteschlangenabonnement zu einem Thema in einem anderen hinzuzufügen, AWS-Konto verwenden Sie AWS Management Console

Bevor Sie beginnen, stellen Sie sicher, dass Sie die ARNs Informationen für Ihr Thema und Ihre Warteschlange haben und dass Sie dem Thema die Erlaubnis erteilt haben, Nachrichten an die Warteschlange zu senden.

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie in der Liste der Warteschlangen die Warteschlange aus, für die das Amazon-SNS-Thema abonniert werden soll.
4. Wählen Sie Subscribe to Amazon SNS topic (Amazon-SNS-Thema abonnieren).
5. Wählen Sie im Menü Specify an Amazon SNS topic available for this queue (Ein für dieses Warteschlangenmenü verfügbares Amazon-SNS-Thema angeben) das SNS topic (SNS-Thema) für Ihre Warteschlange aus.
6. Wählen Sie Enter Amazon SNS topic ARN (Den ARN des Amazon-SNS-Themas eingeben) aus und geben Sie dann den Amazon Ressouren Name (ARN, Amazon-Ressourcenname) ein.

7. Wählen Sie Save (Speichern) aus.

> **Note**
>
> - Zur Kommunikation mit dem Service muss die Warteschlange über Berechtigungen für Amazon SNS verfügen.
> - Da Sie der Besitzer der Warteschlange sind, müssen Sie das Abonnement nicht bestätigen.

Ein Benutzer, der nicht der Warteschlangeneigentümer ist, erstellt ein Abonnement

Jeder Benutzer, der ein Abonnement erstellt, aber nicht der Besitzer der Warteschlange ist, muss das Abonnement bestätigen.

Wenn Sie die `Subscribe`-Aktion verwenden, sendet Amazon SNS eine Abonnementbestätigung an die Warteschlange. Das Abonnement wird in der Amazon-SNS-Konsole angezeigt, wobei die Abonnement-ID auf Pending Confirmation (Ausstehende Bestätigung) festgelegt ist.

Zur Bestätigung des Abonnements muss ein Benutzer mit der Berechtigung zum Lesen von Nachrichten aus der Warteschlange die URL zur Bestätigung des Abonnements abrufen und der Eigentümer des Abonnements muss das Abonnement unter Verwendung dieser URL bestätigen. Solange das Abonnement nicht bestätigt ist, werden keine Benachrichtigungen, die zum Thema veröffentlicht wurden, an die Warteschlange gesendet. Sie können das Abonnement über die Amazon-SQS-Konsole oder mit der `ReceiveMessage`-Aktion bestätigen.

> **Note**
>
> Bevor Sie einen Endpunkt für das Thema abonnieren, stellen Sie sicher, dass die Warteschlange Nachrichten aus dem Thema empfangen kann. legen Sie dazu die `sqs:SendMessage`-Berechtigung für die Warteschlange fest. Weitere Informationen finden Sie unter Schritt 2: Erteilen der Berechtigung für das Amazon SNS-Thema zum Senden von Nachrichten an die Amazon-SQS-Warteschlange.

Schritt 1: Um ein Amazon SQS SQS-Warteschlangenabonnement zu einem Thema in einem anderen hinzuzufügen, AWS-Konto verwenden Sie AWS Management Console

Bevor Sie beginnen, stellen Sie sicher, dass Sie über die ARNs für Ihr Thema und Ihre Warteschlange erforderlichen Rechte verfügen und dass Sie dem Thema die Erlaubnis erteilt haben, Nachrichten an die Warteschlange zu senden.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Geben Sie unter Topic ARN (Themen-ARN) den ARN des Themas ein.
   b. Für Protokoll wählen Sie Amazon SQS.
   c. Geben Sie als Endpoint (Endpunkt) den ARN der Warteschlange ein.
   d. Wählen Sie Create subscription (Abonnement erstellen) aus.

   > **Note**
   >
   > - Zur Kommunikation mit dem Service muss die Warteschlange über Berechtigungen für Amazon SNS verfügen.

Nachfolgend finden Sie eine Beispielrichtlinienanweisung, die es dem Amazon-SNS-Thema erlaubt, eine Nachricht an die Amazon-SQS-Warteschlange zu senden.

```
{
   "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"
      }
```

```
    }
}
```

Schritt 2: Um ein Abonnement zu bestätigen, verwenden Sie AWS Management Console

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Wählen Sie die Warteschlange aus, für die ein Abonnement für das Thema aussteht.
3. Wählen Sie Send and receive messages (Nachrichten senden und empfangen) und dann Poll for messages (Nachrichten abfragen) aus.

   Eine Nachricht mit der Abonnementbestätigung wird in der Warteschlange empfangen.
4. Führen Sie in der Spalte Body (Textkörper) die folgenden Schritte aus:

   a. Wählen Sie More Details (Weitere Details) aus.

   b. Suchen Sie im Dialogfeld Message Details (Nachrichtendetails) den Wert für SubscribeURL. Dies ist Ihr Abonnement-Link (Beispiel unten). Weitere Informationen zur Validierung von API-Token finden Sie unter `ConfirmSubscription` in der Amazon-SNS-API-Referenz.

   ```
   https://sns.us-west-2.amazonaws.com/?
   Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
   east-2:123456789012:MyTopic&Token=2336412f37fb...
   ```

   c. Notieren Sie sich den Link zur Bestätigung des Abonnements. Die URL muss vom Eigentümer der Warteschlange an den Eigentümer des Abonnements weitergegeben werden. Der Eigentümer des Abonnements muss die URL in die Amazon-SNS-Konsole eingeben.
5. Melden Sie sich als Eigentümer des Abonnements bei der Amazon-SNS-Konsole an. Die Bestätigung erfolgt durch den Eigentümer des Abonnements.
6. Wählen Sie das relevante Thema aus.
7. Wählen Sie das relevante Abonnement in der Tabelle für Abonnementangebote des Themas aus. Es ist als „Pending confirmation“ (Bestätigung ausstehend) gekennzeichnet.
8. Wählen Sie Confirm subscription (Abonnement bestätigen) aus.
9. Ein Modal wird angezeigt, das den Link zur Bestätigung des Abonnements anfordert. Fügen Sie den Link zur Bestätigung des Abonnements ein.
10. Wählen Sie im Modal Confirm subscription (Abonnement bestätigen) aus.

Eine XML-Antwort wird angezeigt, zum Beispiel:

```
<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>
```

Die abonnierte Warteschlange kann nun Nachrichten vom Thema empfangen.

11. (Optional) Wenn Sie das Themenabonnement in der Amazon-SNS-Konsole anzeigen, können Sie sehen, dass die Meldung Pending Confirmation (Ausstehende Bestätigung) in der Spalte Subscription ID (Abonnement-ID) durch den Abonnement-ARN ersetzt wurde.

Wie erzwinge ich, dass ein Abonnement eine Authentifizierung bei Abmeldeanfragen erfordert?

Der Eigentümer des Abonnements muss das `AuthenticateOnUnsubscribe`-Flag bei der Bestätigung des Abonnements auf „true“ (wahr) festlegen.

- `AuthenticateOnUnsubscribe` wird automatisch auf „true“ (wahr) festgelegt, wenn der Eigentümer der Warteschlange das Abonnement erstellt.
- `AuthenticateOnUnsubscribe` kann nicht auf „true“ (wahr) festgelegt werden, wenn der Link zur Abonnementbestätigung ohne Authentifizierung aufgerufen wird.

Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange oder AWS Lambda -Funktion in einer anderen Region

Amazon SNS unterstützt regionsübergreifende Lieferungen sowohl für Regionen, die standardmäßig aktiviert sind, als auch für Opt-In-Regionen. Für die aktuelle Liste von AWS -Regionen, die Amazon

SNS unterstützt, einschließlich Opt-in-Regionen, finden Sie unter Endpunkte und Kontingente für Amazon-Simple-Notification-Service-Endpunkte in der Allgemeine Amazon Web Services-Referenz.

Amazon SNS unterstützt die regionsübergreifende Übermittlung von Benachrichtigungen an Amazon SQS Queues und an AWS Lambda -Funktionen. Wenn es sich bei einer der Regionen um eine Opt-in-Region handelt, müssen Sie in der Richtlinie der abonnierten Ressource einen anderen Amazon SNS Dienstprinzipal angeben.

Der Amazon SNS-Abonnementbefehl muss in der Region ausgeführt werden, in der Amazon SNS gehostet wird, in der entsprechenden Region. Wenn sich Amazon SNS beispielsweise im Konto „A“ in der Region us-east-1 und die Lambda-Funktion im Konto „B“ in der Region us-east-2 befindet, muss der Abonnement-CLI-Befehl in Konto „A“ in der Region us-east-1 ausgeführt werden.

Opt-In-Regionen

Amazon SNS unterstützt die folgenden Opt-in-Regionen:

Name der Region	Region
Region Afrika (Kapstadt)	af-south-1
Region Asien-Pazifik (Hongkong)	ap-east-1
Region Asien-Pazifik (Hyderabad)	ap-south-2
Region Asien-Pazifik (Jakarta)	ap-southeast-3
Region Asien-Pazifik (Melbourne)	ap-southeast-4
Region Europa (Mailand)	eu-south-1
Region Europa (Spanien)	eu-south-2
Region Europa (Zürich)	eu-central-2
Region Israel (Tel Aviv)	il-central-1
Region Naher Osten (Bahrain)	me-south-1
Region Naher Osten (UAE)	me-central-1

Informationen zur Aktivierung einer Opt-in-Region finden Sie unter AWS Regionen verwalten in der Allgemeine Amazon Web Services-Referenz.

Wenn Sie Amazon SNS zum Übermitteln von Nachrichten aus Opt-in-Regionen an standardmäßig aktivierte Regionen verwenden, müssen Sie die für die Queue erstellte Ressourcenrichtlinie ändern. Ersetzen Sie den Prinzipal `sns.amazonaws.com` durch `sns.<opt-in-region>.amazonaws.com`. Beispiel:

- Wenn Sie beispielsweise eine Amazon-SQS-Warteschlange in USA Ost (Nord-Virginia) für ein Amazon-SNS-Thema in Asien-Pazifik (Hongkong) abonnieren möchten, ändern Sie den Prinzipal in der Warteschlangenrichtlinie in `sns.ap-east-1.amazonaws.com`. Zu den Opt-In-Regionen gehören alle Regionen, die nach dem 20. März 2019 eingeführt wurden, darunter Asien-Pazifik (Hongkong), Asien-Pazifik (Jakarta), Naher Osten (Bahrain), Europa (Mailand) und Afrika (Kapstadt). Regionen, die vor dem 20. März 2019 gestartet wurden, sind standardmäßig aktiviert.

  Regionsübergreifende Zustellungsunterstützung für Amazon SQS

Regionsübergreifende Zustellungsart	Unterstützt/Nicht unterstützt	
Standardmäßig aktivierte Region zu Opt-In-Region	Unterstützt mit `sns.`*`queue-region`*`.amazonaws.com` im Service-Prinzipal für die Warteschlange	
Opt-In-Region zu standardmäßig aktivierter Region	Unterstützt mit `sns.`*`topic-region`*`.amazonaws.com` im Service-Prinzipal für die Warteschlange	
Opt-In Region zu Opt-In-Region	Unterstützt mit `sns.`*`queue-region`*`.amazonaws.com` im Service-Prinzipal für die Warteschlange	

  Im Folgenden finden Sie ein Beispiel für eine Erklärung zur Zugriffsrichtlinie, die es einem Amazon SNS SNS-Thema in einer Opt-in-Region (af-south-1) ermöglicht, an eine Amazon SQS SQS-Warteschlange in einer Region (us-east-1) zu liefern. enabled-by-default Es enthält die notwendige regionalisierte Service-Prinzipal-Konfiguration unter dem Pfad `Statement/Principal/Service`.

JSON

```
{
    "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::111111111111:destination_queue_name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:af-
south-1:111111111111:source_topic_name"
                }
            }
        }
    ]
}
```

- Um eine AWS Lambda Funktion in USA Ost (Nord-Virginia) für ein Amazon SNS SNS-Thema in Asien-Pazifik (Hongkong) zu abonnieren, ändern Sie den Principal in der AWS Lambda Funktionsrichtlinie aufsns.ap-east-1.amazonaws.com. Zu den Opt-In-Regionen gehören alle Regionen, die nach dem 20. März 2019 eingeführt wurden, darunter Asien-Pazifik (Hongkong), Asien-Pazifik (Jakarta), Naher Osten (Bahrain), Europa (Mailand) und Afrika (Kapstadt). Regionen, die vor dem 20. März 2019 gestartet wurden, sind standardmäßig aktiviert.

  Unterstützung bei regionenübergreifender Lieferung an AWS Lambda

Regionsübergreifende Zustellungsart	Unterstützt/Nicht unterstützt	
Standardmäßig aktivierte Region zu Opt-In-Region	Unterstützt mithilfe von sns.*lambda-function-region* .amazonaws.com im Service-Prinzipal für die Lambda-Funktion	

Regionsübergreifende Zustellungsart	Unterstützt/Nicht unterstützt	
Opt-In-Region zu standardmäßig aktivierter Region	Unterstützt mithilfe von `sns.`*`topic-region`*`.amazonaws.com` im Service-Prinzipal für die Lambda-Funktion	
Opt-In Region zu Opt-In-Region	Unterstützt mithilfe von `sns.`*`lambda-function-region`*`.amazonaws.com` im Service-Prinzipal für die Lambda-Funktion	

Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen

Amazon SNS bietet Unterstützung für die Protokollierung des Zustellungsstatus von Benachrichtigungen, die an Themen mit den folgenden Amazon SNS SNS-Endpunkten gesendet wurden:

- Amazon Data Firehose
- Amazon Simple Queue Service
- AWS Lambda
- HTTPS
- Plattformanwendungs-Endpunkt

Versandstatusprotokolle werden an Amazon CloudWatch Logs gesendet und bieten Einblicke in die Nachrichtenzustellungsvorgänge. Diese Protokolle helfen Ihnen dabei:

- Stellen Sie fest, ob eine Nachricht erfolgreich an einen Endpunkt zugestellt wurde.
- Identifizieren Sie die Antwort vom Endpunkt an Amazon SNS.
- Messen Sie die Verweildauer von Nachrichten (Zeit zwischen dem Veröffentlichungszeitstempel und der Übergabe an den Endpunkt).

Sie können die Protokollierung des Zustellungsstatus mithilfe der AWS Management Console, AWS SDKs, Query API oder konfigurieren. AWS CloudFormation

Voraussetzungen für die Protokollierung des Lieferstatus

In diesem Thema werden die erforderlichen IAM-Berechtigungen beschrieben, um Amazon SNS das Schreiben von Versandprotokollen zu ermöglichen, CloudWatch und es wird die standardmäßige Benennungskonvention für Protokollgruppen erläutert. Dadurch wird sichergestellt, dass Sie über die richtige Einrichtung und den richtigen Zugriff zur Überwachung und Analyse von Nachrichtenzustellungsprotokollen in CloudWatch Protokollen verfügen.

Erforderliche IAM-Berechtigungen

Die für die Protokollierung des Lieferstatus zugeordnete IAM-Rolle muss die folgenden Berechtigungen enthalten, damit Amazon SNS in Protokolle schreiben kann CloudWatch . Sie können eine bestehende Rolle mit diesen Berechtigungen verwenden oder während der Einrichtung eine neue Rolle erstellen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

Benennungskonventionen für Gruppen protokollieren

Standardmäßig erstellt Amazon SNS CloudWatch Protokollgruppen für Versandstatusprotokolle unter Verwendung der folgenden Namenskonvention. Protokollstreams innerhalb dieser Gruppe entsprechen den Endpunktprotokollen (z. B. Lambda, Amazon SQS). Stellen Sie sicher, dass Sie berechtigt sind, diese Protokolle in der Logs-Konsole CloudWatch einzusehen.

```
sns/<region>/<account-id>/<topic-name>
```

Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS Management Console

In diesem Thema wird erklärt, wie Sie die Protokollierung des Nachrichtenzustellungsstatus für Amazon SNS SNS-Themen aktivieren, einschließlich der Konfiguration von Protokollierungseinstellungen, der Zuweisung von IAM-Rollen und der Überprüfung, ob Logs CloudWatch Zustellungsprotokolle zur Überwachung und Fehlerbehebung erfassen.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie das gewünschte Thema aus und klicken Sie dann auf Bearbeiten.
4. Erweitern Sie den Abschnitt Protokollierung des Lieferstatus.
5. Wählen Sie das Protokoll aus, für das Sie die Protokollierung aktivieren möchten (z. B. HTTP, Lambda, Amazon SQS).
6. Geben Sie die Erfolgs-Samplerate ein. Dabei handelt es sich um den Prozentsatz erfolgreicher Nachrichten, für die Sie Logs erhalten CloudWatch möchten.
7. Im Bereich IAM-Rollen müssen Sie Rollen sowohl für die Erfolgs - als auch für die Fehlerprotokollierung konfigurieren:

   - Eine bestehende Servicerolle verwenden — Wählen Sie eine bestehende IAM-Rolle aus, die über die erforderlichen Berechtigungen verfügt, in die Amazon SNS Protokolle schreiben kann. CloudWatch

   - Neue Servicerolle erstellen — Wählen Sie Neue Rollen erstellen, um die IAM-Rollen für erfolgreiche und fehlgeschlagene Lieferungen in der IAM-Konsole zu definieren. Einzelheiten zu Berechtigungen finden Sie unter. Voraussetzungen für die Protokollierung des Lieferstatus
8. Wählen Sie Änderungen speichern aus.

   Nachdem Sie die Protokollierung aktiviert haben, können Sie die CloudWatch Protokolle mit dem Status der Nachrichtenzustellung anzeigen und analysieren. Weitere Informationen zur Verwendung CloudWatch finden Sie in der CloudWatchDokumentation.

Das Protokoll-Setup wird überprüft

1. Melden Sie sich bei der CloudWatch Logs-Konsole an.
2. Suchen Sie die Protokollgruppe mit dem Namen`sns/<region>/<account-id>/<topic-name>`.
3. Stellen Sie sicher, dass Protokollstreams für das konfigurierte Endpunktprotokoll vorhanden sind.
4. Senden Sie eine Testnachricht zu Ihrem Thema und vergewissern Sie sich, dass Protokolleinträge angezeigt werden, die auf erfolgreiche oder fehlgeschlagene Lieferungen hinweisen.

Konfiguration der Protokollierung des Lieferstatus mithilfe der AWS SDKs

AWS SDKs Sie bieten APIs in mehreren Sprachen die Möglichkeit, Themenattribute für die Protokollierung des Nachrichtenzustellungsstatus festzulegen. Verwenden Sie beispielsweise die SetTopicAttributesAPI, um Folgendes zu konfigurieren:

- `LambdaSuccessFeedbackRoleArn`— IAM-Rolle für eine erfolgreiche Nachrichtenzustellung an Lambda-Endpunkte.
- `LambdaSuccessFeedbackSampleRate`— Samplingrate für erfolgreiche Nachrichten an Lambda-Endpunkte.
- `LambdaFailureFeedbackRoleArn`— IAM-Rolle für fehlgeschlagene Nachrichtenzustellung an Lambda-Endpunkte.

AWS CLI Beispiel für einen Befehl

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name LambdaSuccessFeedbackRoleArn \
    --attribute-value arn:aws:iam::123456789012:role/MyFeedbackRole
```

Themenattribute

Verwenden Sie die folgenden Namenswerte für Themenattribute für den Status der Nachrichtenzustellung:

HTTP

- `HTTPSuccessFeedbackRoleArn`— Status der erfolgreichen Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen HTTP-Endpunkt abonniert hat.

- `HTTPSuccessFeedbackSampleRate`— Prozentsatz der erfolgreichen Nachrichten, die für ein Amazon SNS SNS-Thema als Stichprobe ausgewählt wurden und über einen HTTP-Endpunkt abonniert wurden.
- `HTTPFailureFeedbackRoleArn`— Status der Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen HTTP-Endpunkt abonniert hat, fehlgeschlagen.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— Status der erfolgreichen Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen Amazon Data Firehose-Endpunkt abonniert hat.
- `FirehoseSuccessFeedbackSampleRate`— Prozentsatz der erfolgreichen Nachrichten, die für ein Amazon SNS SNS-Thema als Stichprobe ausgewählt werden sollen und die einen Amazon Data Firehose-Endpunkt abonniert haben.
- `FirehoseFailureFeedbackRoleArn`— Status der Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen Amazon Data Firehose-Endpunkt abonniert hat, fehlgeschlagen.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— Status der erfolgreichen Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen Lambda-Endpunkt abonniert hat.
- `LambdaSuccessFeedbackSampleRate`— Prozentsatz der erfolgreichen Nachrichten, die für ein Amazon SNS SNS-Thema als Stichprobe ausgewählt werden sollen, das einen Lambda-Endpunkt abonniert hat.
- `LambdaFailureFeedbackRoleArn`— Status der Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen Lambda-Endpunkt abonniert hat, fehlgeschlagen.

Endpunkte von Plattformanwendungen

- `ApplicationSuccessFeedbackRoleArn`— Status der erfolgreichen Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen AWS Anwendungsendpunkt abonniert hat.
- `ApplicationSuccessFeedbackSampleRate`— Prozentsatz der erfolgreichen Nachrichten, die für ein Amazon SNS SNS-Thema als Stichprobe ausgewählt wurden und für einen AWS Anwendungsendpunkt abonniert wurden.
- `ApplicationFailureFeedbackRoleArn`— Status der Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen AWS Anwendungsendpunkt abonniert hat, fehlgeschlagen.

> **Note**
>
> Darüber hinaus können Sie Anwendungsattribute so konfigurieren, dass der Zustellungsstatus direkt bei den Push-Benachrichtigungsdiensten protokolliert wird. Weitere Informationen finden Sie unter Verwendung von Amazon SNS-Anwendungsattributen für den Status von Nachrichtenübermittlungen.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— Status der erfolgreichen Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen Amazon SQS SQS-Endpunkt abonniert hat.
- `SQSSuccessFeedbackSampleRate`— Prozentsatz der erfolgreichen Nachrichten, die für ein Amazon SNS SNS-Thema als Stichprobe ausgewählt wurden und für einen Amazon SQS SQS-Endpunkt abonniert wurden.
- `SQSFailureFeedbackRoleArn`— Status der Nachrichtenzustellung für ein Amazon SNS SNS-Thema, das einen Amazon SQS SQS-Endpunkt abonniert hat, fehlgeschlagen.

Protokolle für Plattformanwendungsendpunkte werden in dieselbe CloudWatch Protokollgruppe geschrieben wie andere Endpunkte.

> **Note**
>
> Die `<ENDPOINT>FailureFeedbackRoleArn` Attribute `<ENDPOINT>SuccessFeedbackRoleArn` und werden verwendet, um Amazon SNS Schreibzugriff auf die Nutzung von CloudWatch Logs in Ihrem Namen zu gewähren. Das Attribut `<ENDPOINT>SuccessFeedbackSampleRate` dient zum Festlegen des Prozentsatzes der Samplerate (0-100) der erfolgreich zugestellten Nachrichten. Nachdem Sie das `<ENDPOINT>FailureFeedbackRoleArn` Attribut konfiguriert haben, werden bei allen fehlgeschlagenen Nachrichtenzustellungen CloudWatch Protokolle generiert.

AWS SDK-Beispiele zur Konfiguration von Themenattributen

Die folgenden Code-Beispiele zeigen, wie `SetTopicAttributes` verwendet wird.

CLI

AWS CLI

So legen Sie ein Attribut für ein Thema fest

Im folgenden `set-topic-attributes`-Beispiel wird das `DisplayName`-Attribute für das angegebene Thema festgelegt.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für PHP API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS CloudFormation

Verwenden Sie zur Konfiguration `DeliveryStatusLogging` der Verwendung eine JSON- oder YAML-Vorlage AWS CloudFormation, um einen AWS CloudFormation Stack zu erstellen. Weitere Informationen finden Sie unter der `DeliveryStatusLogging` Eigenschaft der `AWS::SNS::Topic` Ressource im AWS CloudFormation Benutzerhandbuch. Im Folgenden finden Sie Beispiele für AWS CloudFormation Vorlagen in JSON und YAML, mit denen Sie ein neues Thema erstellen oder ein vorhandenes Thema mit allen `DeliveryStatusLogging` Attributen für das Amazon SQS SQS-Protokoll aktualisieren können.

Vergewissern Sie sich, dass die IAM-Rollen, auf die unter verwiesen wird, `FailureFeedbackRoleArn` über die erforderlichen CloudWatch Logs-Berechtigungen verfügen. `SuccessFeedbackRoleArn`

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
```

Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS

Amazon SNS definiert eine Zustellungsrichtlinie für jedes Zustellungsprotokoll. Die Zustellungsrichtlinie definiert, wie Amazon SNS die Zustellung von Nachrichten bei serverseitigen Fehlern (wenn das den abonnierten Endpunkt hostenden System nicht verfügbar ist) wiederholt. Wenn die Zustellungsrichtlinie erschöpft ist, stoppt Amazon SNS den erneuten Versand und verwirft die Nachricht, es sei denn, dem Abonnement ist eine Warteschlange für unzustellbare Nachrichten angehängt. Weitere Informationen finden Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe.

Zustellungsprotokolle und -richtlinien

> **Note**
>
> - Mit Ausnahme von HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S unterstützt benutzerdefinierte Richtlinien. Siehe Erstellung HTTP/S einer Lieferrichtlinie.
> - Amazon SNS wendet Jittering bei Zustellungswiederholungen an. Weitere Informationen finden Sie im Beitrag Exponential Backoff and Jitter im AWS Architecture Blog.
> - Die Gesamtdauer für den erneuten Versuch einer Richtlinie für einen HTTP/S Endpunkt darf 3.600 Sekunden nicht überschreiten. Dies ist ein hartes Limit, das nicht erhöht werden kann.

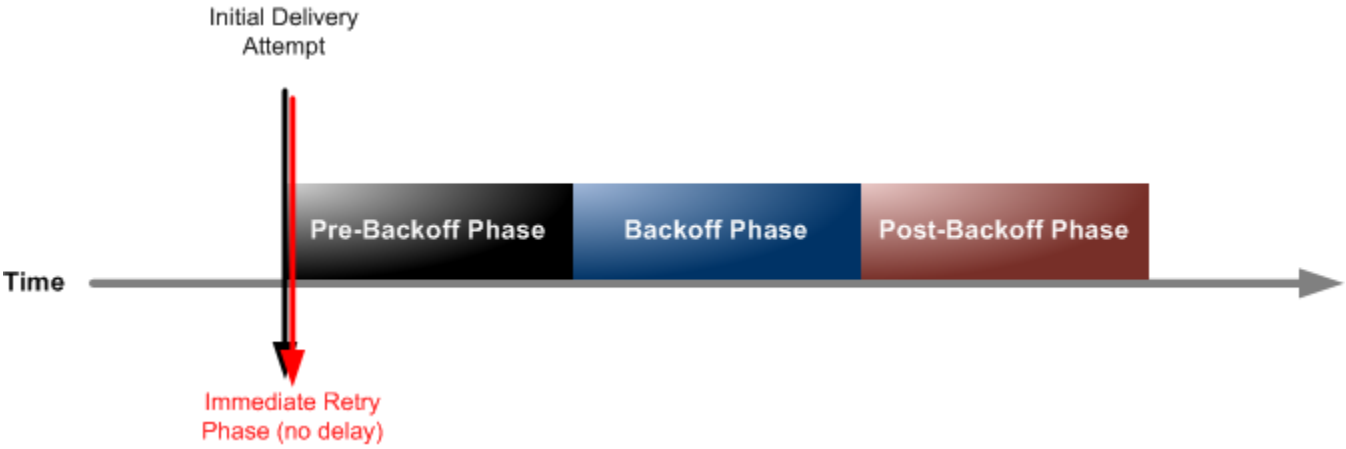
Endpunkttyp	Zustellungsprotokolle	Phase „Sofortiger Wiederholungsversuch (keine Verzögerung)“	Phase „Pre-Backoff“	Phase „Backoff“	Phase „Post-Backoff“	Gesamtzahl der Versuche
AWS verwaltete Endpunkte	[1]	3-mal, ohne Verzögerung	2-mal, mit 1 Sekunde Verzögerung	10-mal, mit exponentiellem Backoff von 1 Sekunde bis 20 Sekunden	100.000-mal, mit 20 Sekunden Verzögerung	100.015-mal, über 23 Tage
	AWS Lambda					
	Amazon SQS					
Vom Kunden verwaltete Endpunkte	SMTP	0-mal, ohne Verzögerung	2-mal, mit 10 Sekunden Verzögerung	10-mal, mit exponentiellem Backoff von 10	38-Mal, mit 600 Sekunden (10 Minuten)	50 Versuche, über 6 Stunden
	SMS					

Endpunkttyp	Zustellungsprotokolle	Phase „Sofortiger Wiederholungsversuch (keine Verzögerung)“	Phase „Pre-Backoff“	Phase „Backoff“	Phase „Post-Backoff“	Gesamtzahl der Versuche
	Mobile Push			Sekunden bis 600 Sekunden (10 Minuten)	Verzögerung	

[1] Für Drosselungsfehler mit dem Firehose-Protokoll verwendet Amazon SNS dieselben Lieferrichtlinien wie für vom Kunden verwaltete Endgeräte.

Stufen von Zustellungsrichtlinien

Das folgende Diagramm zeigt die Phasen einer Zustellungsrichtlinie.



Jede Zustellungsrichtlinie umfasst vier Phasen.

1. Phase der sofortigen Wiederholung (ohne Verzögerung) — Diese Phase findet unmittelbar nach dem ersten Zustellungsversuch statt. Es gibt keine Verzögerung zwischen den Wiederholungen in dieser Phase.

2. Phase vor dem Backoff — Diese Phase folgt auf die Phase der sofortigen Wiederholung. Amazon SNS verwendet diese Phase für eine Reihe von Wiederholungsversuchen, bevor eine Backoff-Funktion angewendet wird. Diese Phase gibt die Anzahl der Wiederholungsversuche und die Verzögerung dazwischen an.
3. Backoff-Phase — In dieser Phase wird die Verzögerung zwischen Wiederholungsversuchen mithilfe der Retry-Backoff-Funktion gesteuert. Diese Phase legt eine Mindestverzögerung, eine Maximalverzögerung und eine retry-backoff-Funktion fest, die definiert, wie schnell die Verzögerung vom Mindest- zum Maximalwert steigt. Die backoff-Funktion kann arithmetisch, exponentiell, geometrisch oder linear sein.
4. Post-Backoff-Phase — Diese Phase folgt auf die Backoff-Phase. Sie gibt die Anzahl der Wiederholungsversuche und die Verzögerung dazwischen an. Dies ist die letzte Phase.

Erstellung HTTP/S einer Lieferrichtlinie

Sie können mithilfe einer Zustellungsrichtlinie mit vier Phasen definieren, wie Amazon SNS die Nachrichtenzustellung an HTTP/S Endpunkte wiederholt: verzögerungsfrei, vor dem Backoff, dem Backoff und dem Post-Backoff. Diese Richtlinie ermöglicht es Ihnen, die Standardeinstellungen für Wiederholungsversuche zu überschreiben und sie an die Kapazität Ihres HTTP-Servers anzupassen.

Sie können Ihre HTTP/S Lieferrichtlinie entweder auf Themen - oder Abonnementebene als JSON-Objekt definieren:

- Richtlinie auf Themenebene — Gilt für alle HTTP/S Abonnements, die mit dem Thema verknüpft sind. Verwenden Sie die `SetTopicAttributes`API-Aktion `CreateTopic`oder, um diese Richtlinie festzulegen.
- Richtlinie auf Abonnementebene — Gilt nur für ein bestimmtes Abonnement. Verwenden Sie die `SetSubscriptionAttributes`API-Aktion `Subscribe`oder, um diese Richtlinie festzulegen.

Alternativ können Sie die AWS::SNS::SubscriptionRessource auch in Ihren AWS CloudFormation Vorlagen verwenden.

Sie sollten Ihre Lieferrichtlinien an die Kapazität Ihres HTTP/S Servers anpassen:

- Ein Server für alle Abonnements — Wenn alle HTTP/S Abonnements in einem Thema denselben Server verwenden, legen Sie die Übermittlungsrichtlinie als Themenattribut fest, um die Konsistenz aller Abonnements sicherzustellen.

- Verschiedene Server für Abonnements — Wenn Abonnements auf unterschiedliche Server abzielen, erstellen Sie für jedes Abonnement eine eigene Bereitstellungsrichtlinie, die auf die Kapazität des jeweiligen Servers zugeschnitten ist.

Sie können auch den `Content-Type` Header in der Anforderungsrichtlinie festlegen, um den Medientyp der Benachrichtigung anzugeben. Standardmäßig sendet Amazon SNS alle Benachrichtigungen an HTTP/S Endpunkte, deren Inhaltstyp auf eingestellt ist. `text/plain; charset=UTF-8` Sie können diese Standardeinstellung jedoch mithilfe des `headerContentType`Felds in der Anforderungsrichtlinie überschreiben.

Das folgende JSON-Objekt definiert eine Zustellungsrichtlinie mit Wiederholungsversuchen, die in vier Phasen unterteilt sind:

1. Phase ohne Verzögerung — Versuchen Sie es sofort dreimal erneut.
2. Pre-Backoff-Phase — Versuchen Sie es zweimal mit einem Intervall von 1 Sekunde.
3. Backoff-Phase — Versuchen Sie es 10 Mal mit exponentiellen Verzögerungen zwischen 1 und 60 Sekunden.
4. Phase nach der Backoff-Phase — Versuchen Sie es 35 Mal mit einem festen Intervall von 60 Sekunden.

Amazon SNS unternimmt insgesamt 50 Versuche, eine Nachricht zuzustellen, bevor sie verworfen wird. Um Nachrichten beizubehalten, die nach allen Wiederholungsversuchen nicht zugestellt werden können, konfigurieren Sie Ihr Abonnement so, dass unzustellbare Nachrichten in eine Warteschlange für unzustellbare Briefe (DLQ) verschoben werden. Weitere Informationen finden Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe.

Amazon SNS betrachtet alle 5XX-Fehler und die Fehler 429 (zu viele gesendete Anfragen) als wiederholbar. Diese Fehler unterliegen den Lieferrichtlinien. Alle anderen Fehler gelten als dauerhafte Fehler und es werden keine erneuten Versuche unternommen.

> **Note**
>
> Diese Zustellungsrichtlinie verwendet die `maxReceivesPerSecond` Eigenschaft, um den Zustellungsverkehr auf durchschnittlich 10 Nachrichten pro Sekunde pro Abonnement zu drosseln. Dieser Mechanismus trägt zwar dazu bei, dass Ihr HTTP/S Endpunkt nicht durch hohen Traffic überlastet wird, ist jedoch darauf ausgelegt, eine durchschnittliche Zustellungsrate aufrechtzuerhalten und keine strikte Obergrenze durchzusetzen.

Gelegentlich kann es zu Spitzenwerten des Lieferverkehrs kommen, die das angegebene Limit überschreiten, insbesondere wenn Ihre Veröffentlichungsrate deutlich über dem Drosselungslimit liegt.
Wenn der (eingehende) Veröffentlichungsverkehr die Zustellungsrate (ausgehender Verkehr) überschreitet, kann dies zu einem Nachrichtenrückstand und einer höheren Zustellungslatenz führen. Um solche Probleme zu vermeiden, stellen Sie sicher, dass der `maxReceivesPerSecond` Wert den Kapazitäts- und HTTP/S Arbeitslastanforderungen Ihres Servers entspricht.

Das folgende Beispiel für eine Zustellungsrichtlinie setzt den Standard-Inhaltstyp für HTTP/S-Benachrichtigungen außer Kraft. `application/json`

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

Die Zustellungsrichtlinie besteht aus einer Wiederholungsrichtlinie, einer Drosselungsrichtlinie und einer Anforderungsrichtlinie. Insgesamt enthält eine Lieferrichtlinie 9 Attribute.

Richtlinie	Beschreibung	Constraint
`minDelayTarget`	Die Mindestverzögerung bei einem Wiederholungsversuch. Einheit: Sekunden	1 bis zur Maximalverzögerung Standard: 20

Richtlinie	Beschreibung	Constraint
`maxDelayTarget`	Die Maximalverzögerung bei einem Wiederholungsversuch. Einheit: Sekunden	Mindestverzögerung auf 3.600 Standard: 20
`numRetries`	Die Gesamtzahl der Wiederholungsversuche, einschließlich sofortiger Wiederholungsversuche sowie, Pre-Backoff-, Backoff- und Post-Backoff-Wiederholungsversuchen.	0 bis 100 Standard: 3
`numNoDelayRetries`	Die Anzahl der Wiederholungen, die sofort durchgeführt werden sollen, ohne Verzögerung zwischen ihnen.	0 oder höher Standard: 0
`numMinDelayRetries`	Die Anzahl der Wiederholungsversuche in der Pre-Backoff-Phase mit der angegebenen Mindestverzögerung dazwischen.	0 oder höher Standard: 0
`numMaxDelayRetries`	Die Anzahl der Wiederholungsversuche in der Post-Backoff-Phase mit der Maximalverzögerung dazwischen.	0 oder höher Standard: 0

Richtlinie	Beschreibung	Constraint
`backoffFunction`	Das Modell für Backoff zwischen Wiederholungsversuchen.	Eine von vier Optionen: • Arithmetisch • Exponentiell • Geometrisch • Linear Standard: linear
`maxReceivesPerSecond`	Die maximale durchschnittliche Anzahl von Nachrichtenzustellungen pro Sekunde pro Abonnement.	1 oder höher Standard: Keine Drosselung (keine Begrenzung der Zustellungsrate)

Richtlinie	Beschreibung	Constraint
`headerContentType`	Der Inhaltstyp der Benachrichtigung, die an HTTP/S-Endpunkte gesendet wird.	Wenn die Anforderungsrichtlinie nicht definiert ist, wird der Inhaltstyp standardmäßig auf `text/plain; charset=UTF-8` festgelegt. Wenn die Bereitstellung von Rohnachrichten für ein Abonnement deaktiviert ist (Standard) oder wenn die Bereitstellungsrichtlinie auf Themenebene definiert ist, werden die Header-Inhaltstypen `application/json` und `text/plain` unterstützt. Wenn die Bereitstellung von Rohnachrichten für ein Abonnement aktiviert ist, werden die folgenden Inhaltstypen unterstützt: • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • Anwendung/octet-stream • application/soap+xml • Anwendung/ x-www-form-urlencoded

Richtlinie	Beschreibung	Constraint
		• application/xhtml+xml • application/xml

Amazon SNS verwendet die folgende Formel, um die Anzahl der Wiederholungsversuche in der Backoff-Phase zu berechnen:

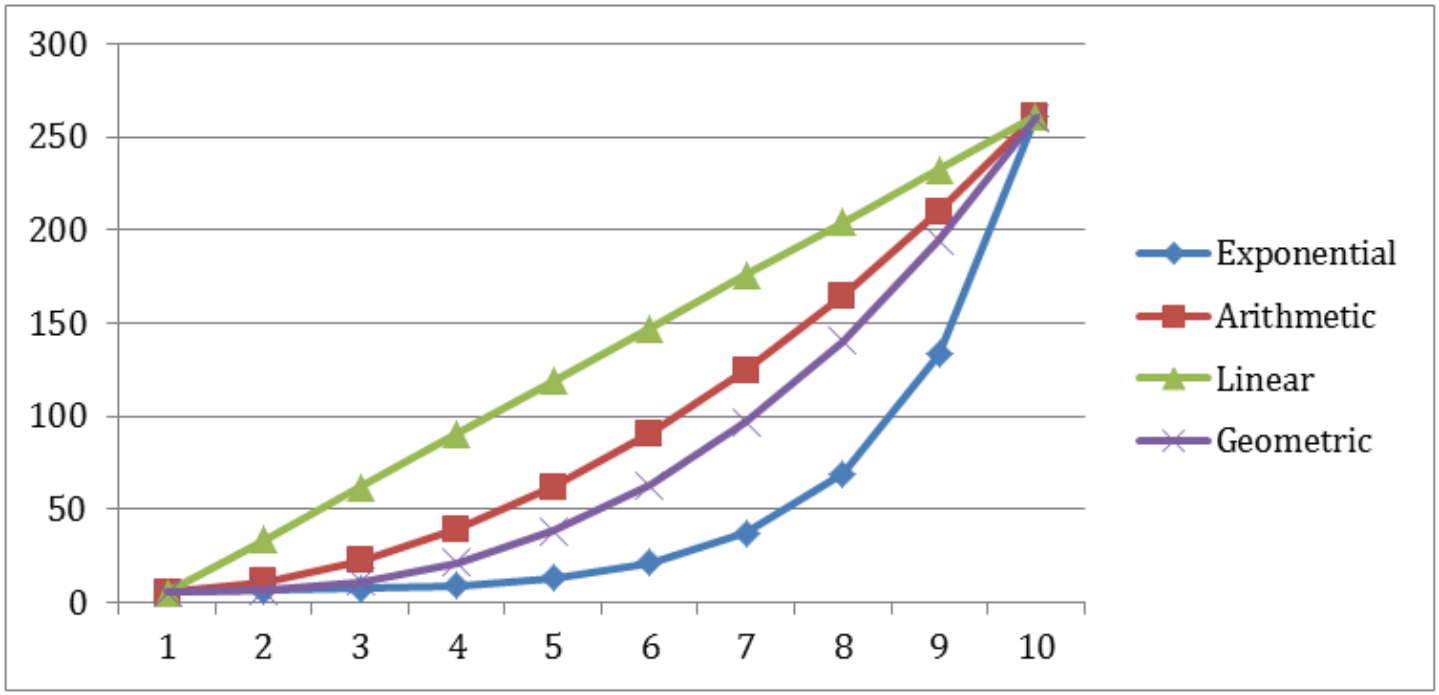
```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

Sie können die Häufigkeit der Wiederholungen während der Backoff-Phase mithilfe von drei Parametern steuern:

- **`minDelayTarget`**— Legt die Verzögerung für den ersten Wiederholungsversuch in der Backoff-Phase fest.
- **`maxDelayTarget`**— Legt die Verzögerung für den letzten Wiederholungsversuch in der Backoff-Phase fest.
- **`backoffFunction`**— Bestimmt den Algorithmus, den Amazon SNS verwendet, um die Verzögerungen für alle Wiederholungsversuche zwischen dem ersten und dem letzten Versuch zu berechnen. Sie können aus vier verfügbaren Wiederholungs-Backoff-Funktionen wählen.

Das folgende Diagramm zeigt, wie sich verschiedene Backoff-Funktionen auf die Verzögerungen zwischen Wiederholungsversuchen während der Backoff-Phase auswirken. Die für dieses Beispiel verwendete Zustellungsrichtlinie umfasst die folgenden Einstellungen: insgesamt 10 Wiederholungen, eine Mindestverzögerung von 5 Sekunden und eine maximale Verzögerung von 260 Sekunden.

- Die vertikale Achse zeigt die Verzögerung (in Sekunden) für jeden Wiederholungsversuch.
- Die horizontale Achse stellt die Wiederholungssequenz dar, die vom ersten bis zum zehnten Versuch reicht.



Amazon SNS SNS-Warteschlangen für unzustellbare Briefe

Eine Queue für unzustellbare Nachrichten ist eine Amazon-SQS-Queue, an die ein Amazon SNS Abonnement Nachrichten senden kann, die nicht erfolgreich an Abonnenten gesendet werden konnten. Nachrichten, die aufgrund von Clientfehlern oder Serverfehlern nicht zugestellt werden können, werden in der Queue für unzustellbare Nachrichten zur weiteren Analyse oder erneuten Verarbeitung gespeichert. Weitere Informationen erhalten Sie unter Konfigurieren einer Amazon-SNS-Warteschlange für unzustellbare Nachrichten für ein Abonnement und Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

> **Note**
>
> - Das Amazon SNS SNS-Abonnement und die Amazon SQS SQS-Warteschlange müssen sich unter demselben AWS Konto und derselben Region befinden.
> - Für ein FIFO-Thema können Sie eine Amazon-SQS-Warteschlange als Warteschlange für unzustellbare Nachrichten für das Amazon-SNS-Abonnement verwenden. FIFO-Themenabonnements verwenden FIFO-Warteschlangen und Standardthemenabonnements verwenden Standardwarteschlangen.
> - Um eine verschlüsselte Amazon SQS SQS-Warteschlange als Warteschlange für unzustellbare Briefe zu verwenden, müssen Sie ein benutzerdefiniertes KMS mit einer

Schlüsselrichtlinie verwenden, die dem Amazon SNS SNS-Service Principal Zugriff auf API-Aktionen gewährt. AWS KMS Weitere Informationen finden Sie unter Sicherung von Amazon SNS SNS-Daten mit serverseitiger Verschlüsselung in dieser Anleitung und Schutz von Amazon SQS Daten mithilfe serverseitiger Verschlüsselung (SSE) und AWS KMS im Amazon Simple Queue Service-Entwicklerhandbuch.

Warum schlagen Nachrichtenzustellungen fehl?

Im Allgemeinen schlägt die Nachrichtenzustellung fehl, wenn Amazon SNS aufgrund eines clientseitigen oder serverseitigen Fehlers nicht auf einen abonnierten Endpunkt zugegriffen werden kann. Wenn Amazon SNS einen clientseitigen Fehler erhält oder weiterhin einen serverseitigen Fehler für eine Nachricht erhält, die über die Anzahl der Wiederholungsversuche hinausgeht, die in der entsprechenden Wiederholungsrichtlinie angegeben ist, verwirft Amazon SNS die Nachricht — es sei denn, es ist eine Queue für unzustellbare Nachrichten an das Abonnement angehängt. Fehlgeschlagene Zustellungen ändern den Status Ihrer Abonnements nicht. Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

Client-seitige Fehler

Client-seitige Fehler können auftreten, wenn Amazon SNS über veraltete Abonnementmetadaten verfügt. Diese Fehler treten im Allgemeinen auf, wenn ein Besitzer den Endpunkt löscht (z. B. eine Lambda-Funktion, mit der ein Thema abonniert wird) oder wenn ein Besitzer die dem abonnierten Endpunkt zugewiesene Richtlinie so ändert, dass Amazon SNS keine Nachrichten an den Endpunkt zugestellt werden können. Amazon SNS versucht nicht, eine durch einen clientseitigen Fehler verursachte fehlerhafte Nachrichtenzustellung zu wiederholen. Amazon SNS versucht keine erneute Nachrichtenzustellung, die aufgrund eines clientseitigen Fehlers fehlschlägt.

Serverseitige Fehler

Serverseitige Fehler können auftreten, wenn das für den abonnierten Endpunkt verantwortliche System nicht verfügbar ist oder eine Ausnahme zurückgibt, die angibt, dass eine gültige Anfrage nicht von Amazon SNS verarbeitet werden kann. Bei serverseitigen Fehlern versucht Amazon SNS die fehlgeschlagenen Zustellungen mithilfe einer linearen oder exponentiellen Backoff-Funktion zu wiederholen. Bei serverseitigen Fehlern, die durch AWS verwaltete Endgeräte verursacht werden, die von Amazon SQS oder Amazon SQS unterstützt werden AWS Lambda, versucht Amazon SNS innerhalb von 23 Tagen bis zu 100.015 Mal, die Zustellung erneut.

Von Kunden verwaltete Endpunkte (wie HTTP, SMTP, SMS oder mobile Pushnachrichten) können ebenfalls serverseitige Fehler verursachen. Amazon SNS versucht auch die Zustellung an diese Endgerätetypen erneut. Obwohl HTTP-Endpunkte von Kunden definierte Wiederholungsrichtlinien unterstützen, legt Amazon SNS eine interne Zustellungsrichtlinie für SMTP-, SMS- und mobile Push-Endpunkte auf 50-mal innerhalb von 6 Stunden fest.

Funktionsweise von Queues für unzustellbare Nachrichten

Eine Queue für unzustellbare Nachrichten wird einem Amazon SNS-Abonnement (nicht einem Thema) zugewiesen, da Nachrichtenzustellungen auf Abonnementebene stattfinden. Auf diese Weise können Sie den ursprünglichen Zielendpunkt für jede Nachricht leichter identifizieren.

Bei einer Queue für unzustellbare Nachrichten, die einem Amazon SNS-Abonnement zugeordnet ist, handelt es sich um eine gewöhnliche Amazon SQS-Queue. Weitere Informationen zum Aufbewahrungszeitraum für Nachrichten finden Sie unter Kontingente im Zusammenhang mit Nachrichten in der Amazon Simple Queue Service-Entwicklerrichtlinie. Sie können den Aufbewahrungszeitraum für Nachrichten mithilfe der `SetQueueAttributes` Amazon SQS-API-Aktion anpassen. Um Ihre Anwendungen widerstandsfähiger zu machen, empfehlen wir Ihnen, die maximale Aufbewahrungsfrist für Queues für unzustellbare Nachrichten auf 14 Tage festzulegen.

Wie werden Nachrichten in eine Queue für unzustellbare Nachrichten verschoben?

Ihre Nachrichten werden mithilfe einer Richtlinie für erneute Ausführung in eine Queue für unzustellbare Nachrichten verschoben. Eine Richtlinie für erneute Ausführung ist ein JSON-Objekt, das auf den ARN der Queue für unzustellbare Nachrichten verweist. Das `deadLetterTargetArn`-Attribut gibt den ARN an. Der ARN muss auf eine Amazon SQS SQS-Warteschlange verweisen, die sich in derselben AWS-Konto Region wie Ihr Amazon SNS SNS-Abonnement befindet. Weitere Informationen finden Sie unter Konfigurieren einer Amazon-SNS-Warteschlange für unzustellbare Nachrichten für ein Abonnement.

Das folgende JSON-Objekt ist eine Beispielrichtlinie für eine Richtlinie für erneute Ausführung, die einem SNS-Abonnement zugewiesen ist.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Wie kann ich Nachrichten aus einer Queue für unzustellbare Nachrichten verschieben?

Sie können Nachrichten auf zwei Arten aus einer Queue für unzustellbare Nachrichten verschieben:

- Vermeiden Sie Schreiben von Amazon SQS-Verbraucherlogik: legen Sie Ihre Queue für unzustellbare Nachrichten als Ereignisquelle für die Lambda-Funktion fest, um Ihre Queue für unzustellbare Nachrichten zu leeren.
- Amazon SQS SQS-Verbraucherlogik schreiben — Verwenden Sie die Amazon SQS SQS-API, das AWS SDK oder schreiben Sie eine benutzerdefinierte Verbraucherlogik für das Abfragen, Verarbeiten und Löschen der Nachrichten in der Warteschlange für AWS CLI unzustellbare Briefe.

Wie kann ich Queues für unzustellbare Nachrichten überwachen und protokollieren?

Sie können CloudWatch Amazon-Metriken verwenden, um Warteschlangen für unzustellbare Briefe im Zusammenhang mit Ihren Amazon SNS-Abonnements zu überwachen. Alle Amazon SQS SQS-Warteschlangen geben in Intervallen von einer CloudWatch Minute Metriken aus. Weitere Informationen finden Sie unter Verfügbare CloudWatch Metriken für Amazon SQS im Amazon Simple Queue Service Developer Guide. Alle Amazon SNS SNS-Abonnements mit Warteschlangen für unzustellbare Briefe senden ebenfalls Metriken aus. CloudWatch Weitere Informationen finden Sie unter Überwachung von Amazon SNS SNS-Themen mit CloudWatch.

Um über Aktivitäten in Ihren Warteschlangen für unzustellbare Briefe informiert zu werden, können Sie Metriken und Alarme verwenden. CloudWatch Das Einrichten eines Alarms für die `NumberOfMessagesSent` Metrik ist nicht geeignet, da diese Metrik keine Nachrichten erfasst, die aufgrund fehlgeschlagener Verarbeitungsversuche an eine DLQ gesendet wurden. Verwenden Sie stattdessen die `ApproximateNumberOfMessagesVisible` Metrik, die alle Nachrichten erfasst, die derzeit in der DLQ verfügbar sind, einschließlich der Nachrichten, die aufgrund von Verarbeitungsfehlern verschoben wurden.

Beispiel für eine CloudWatch Alarmeinrichtung

1. Erstellen Sie einen CloudWatchAlarm für die **`ApproximateNumberOfMessagesVisible`** Metrik.
2. Setzen Sie den Alarmschwellenwert auf 1 (oder einen anderen geeigneten Wert, der Ihren Erwartungen und dem DLQ-Verkehr entspricht).

3. Geben Sie ein Amazon SNS SNS-Thema an, um benachrichtigt zu werden, wenn der Alarm ausgelöst wird. Dieses Amazon-SNS-Thema kann Ihre Alarmbenachrichtigung an jeden Endpunkttyp (z. B. E-Mail-Adresse, Telefonnummer oder mobile Pager-App) zustellen.

Mithilfe von CloudWatch Protokollen können Sie die Ausnahmen untersuchen, die dazu führen, dass Amazon SNS SNS-Lieferungen fehlschlagen und Nachrichten an Warteschlangen für unzustellbare Nachrichten gesendet werden. Amazon SNS kann sowohl erfolgreiche als auch fehlgeschlagene Lieferungen protokollieren. CloudWatch Weitere Informationen finden Sie unter Attribute der mobilen Amazon SNS SNS-App.

Konfigurieren einer Amazon-SNS-Warteschlange für unzustellbare Nachrichten für ein Abonnement

Eine Queue für unzustellbare Nachrichten ist eine Amazon-SQS-Queue, an die ein Amazon SNS Abonnement Nachrichten senden kann, die nicht erfolgreich an Abonnenten gesendet werden konnten. Nachrichten, die aufgrund von Clientfehlern oder Serverfehlern nicht zugestellt werden können, werden in der Queue für unzustellbare Nachrichten zur weiteren Analyse oder erneuten Verarbeitung gespeichert. Weitere Informationen erhalten Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe und Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

Auf dieser Seite wird gezeigt, wie Sie das AWS Management Console, ein AWS SDK, das und verwenden können, AWS CloudFormation um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren. AWS CLI

> **Note**
>
> Für ein FIFO-Thema können Sie eine Amazon-SQS-Warteschlange als Warteschlange für unzustellbare Nachrichten für das Amazon-SNS-Abonnement verwenden. FIFO-Themen-Abonnements verwenden FIFO-Warteschlangen, und Standardthemenabonnements verwenden Standardwarteschlangen.

Voraussetzungen

Führen Sie die folgenden Voraussetzungen aus, bevor Sie eine Queue für unzustellbare Nachrichten konfigurieren:

1. Erstellen Sie ein Amazon-SNS-Thema mit dem Namen `MyTopic`.
2. Erstellen einer Amazon-SQS-Queuebenannt`MyEndpoint`verwenden, um als Endpunkt für das Amazon SNS Abonnement zu verwenden.
3. (Überspringen für AWS CloudFormation) Abonnieren Sie die Warteschlange für das Thema.
4. Erstellen Sie eine weitere Amazon–SQS-Warteschlange mit dem Namen `MyDeadLetterQueue`, die als Queue für das Amazon-SNS-Abonnement dienen soll.
5. Um dem Amazon SNS-Prinzipal Zugriff auf die Amazon SQS-API-Aktion zu gewähren, legen Sie die folgende Queue-Richtlinie für `MyDeadLetterQueue` fest.

   JSON

```
{
  "Version": "2012-10-17",
  "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"
        }
      }
    }
  ]
}
```

Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS Management Console

Bevor Sie mit diesem Tutorial beginnen, müssen Sie die Voraussetzungen erfüllen.

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Erstellen Sie eine Amazon-SQS-Warteschlange oder verwenden Sie eine vorhandene Queue und notieren Sie sich den ARN der Queue auf deren Details-Registerkarte, z. B.:

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

3. Melden Sie sich bei der Amazon SNS-Konsole an.
4. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
5. Wählen Sie auf der Seite Subscriptions (Abonnements) ein vorhandenes Abonnement und dann Edit (Bearbeiten) aus.
6. Erweitern Sie auf der Seite ***1234a567-bc89-012d-3e45-6fg7h890123i*** bearbeiten den Abschnitt Redrive policy (dead-letter queue) Richtlinie für erneute Ausführung (Warteschlange für unzustellbare Nachrichten) und führen Sie dann die folgenden Schritte aus:
   a. Wählen Sie Aktiviert.
   b. Geben Sie den ARN einer Amazon SQS Queue an.
7. Wählen Sie Änderungen speichern aus.

   Ihr Abonnement ist für die Verwendung einer Queue für unzustellbare Nachrichten konfiguriert.

So konfigurieren Sie mithilfe eines SDK eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement AWS

Bevor Sie dieses Beispiel ausführen, müssen Sie sicherstellen, dass Sie dieVoraussetzungenerfüllen.

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Das folgende Codebeispiel zeigt, wie Sie es verwenden`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK für Java 1.x

> **Note**
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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);
```

Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS CLI

Bevor Sie mit diesem Tutorial beginnen, müssen Sie die Voraussetzungen erfüllen.

1. Installieren und Konfigurieren der AWS CLI. Weitere Informationen finden Sie im AWS Command Line Interface -Benutzerhandbuch.
2. Verwenden Sie den folgenden -Befehl.

```
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\"}"
```

Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS CloudFormation

Bevor Sie mit diesem Tutorial beginnen, müssen Sie die Voraussetzungen erfüllen.

1. Kopieren Sie den JSON-Code in eine Datei mit dem Namen `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. Melden Sie sich an der AWS CloudFormation -Konsole an.
3. Klicken Sie auf der Seite Select Template (Vorlage auswählen) auf die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen), wählen Sie dann Ihre `MyDeadLetterQueue.json`-Datei und anschließend Next (Weiter) aus.
4. Geben Sie auf der Seite Specify Details (Details angeben) `MyDeadLetterQueue` als Stack Name (Stack-Name) ein und wählen Sie Next (Weiter) aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Klicken Sie auf der Seite Review auf Create.

   AWS CloudFormation beginnt mit der Erstellung des `MyDeadLetterQueue` Stacks und zeigt den Status CREATE_IN_PROGRESS an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

Amazon SNS Nachrichtenarchivierung, -wiederholung und -analyse

Amazon SNS SNS-Standardthemen unterstützen die Nachrichtenarchivierung durch. Sie können Benachrichtigungen an Firehose-Lieferstreams ausweiten, sodass Sie Benachrichtigungen an Speicher- und Analyseziele senden können, die Firehose unterstützt, darunter Amazon Simple Storage Service (Amazon S3), Amazon Redshift und mehr.

Amazon SNS FIFO-Themen unterstützen ein direktes Nachrichtenarchiv ohne Code, mit dem Themeninhaber Nachrichten, die zu einem Thema veröffentlicht wurden, bis zu 365 Tage lang speichern (oder archivieren) können. Für Themen mit einem aktiven `ArchivePolicy` können Abonnenten dann eine `ReplayPolicy` erstellen, um die archivierten Nachrichten an einen abonnierten Endpunkt abzurufen (oder zu wiederholen). Für weitere Informationen zu dieser Funktion siehe Archivierung und Wiedergabe von Amazon SNS SNS-Nachrichten für FIFO-Themen.

Features	Standardthemen	FIFO-Themen
Nachrichtenarchivierung	Lieferstreams von Fanout nach Firehose	Amazon SNS SNS-Nachrichtenarchivierung für Inhaber von FIFO-Themen
Wiederholung der Nachricht	Die Wiederholung von Standardthemen ist keine integrierte Funktion. Viele Kunden erstellen ihre eigenen Nachrichten anhand ihres Nachrichtenarchivs.	Wiederholung von Amazon SNS SNS-Nachrichten für Abonnenten zum Thema FIFO

Ressourcenmanagement und Optimierung in Amazon SNS

Dieses Thema enthält Anleitungen dazu, wie Sie das volle Potenzial von Amazon SNS ausschöpfen können, indem Sie eine optimale Leistung sicherstellen, unnötige Kosten reduzieren und gut organisierte Ressourcen aufrechterhalten.

Themen

- Markierung von Amazon-SNS-Themen

Markierung von Amazon-SNS-Themen

Amazon SNS unterstützt die Markierung von Amazon-SNS-Themen. Auf diese Weise können Sie die mit Ihren Themen verbundenen Kosten verfolgen und verwalten, die Sicherheit Ihrer AWS Identity and Access Management (IAM) -Richtlinien verbessern und Tausende von Themen einfach durchsuchen oder filtern. Mit Tagging können Sie Ihre Amazon SNS SNS-Themen mithilfe von AWS Resource Groups verwalten. Weitere Informationen zu Resource Groups finden Sie im Benutzerhandbuch zu AWS Resource Groups.

Markierungen für die Kostenzuordnungen

Zur Organisation und Identifizierung Ihrer Amazon-SNS-Themen für die Kostenzuordnung können Sie Tags hinzufügen, die den Zweck eines Themas identifizieren. Das ist vor allem dann nützlich, wenn Sie viele Themen haben. Mithilfe von Tags zur Kostenzuweisung können Sie Ihre AWS Rechnung so organisieren, dass sie Ihrer eigenen Kostenstruktur entspricht. Melden Sie sich dazu an, um Ihre AWS Kontorechnung mit den Tagschlüsseln und -werten zu erhalten. Weitere Informationen finden Sie unter Einrichten eines monatlichen Kostenzuordnungsberichts im Benutzerhandbuch zu AWS Billing and Cost Management.

Beispielsweise können Sie wie folgt Tags hinzufügen, die die Kostenstelle und den Zweck Ihrer Amazon-SNS-Topics repräsentieren.

Ressource	Schlüssel	Wert
Thema 1	Kostenstelle	43289
	Anwendung	Auftragsverarbeitung

Ressource	Schlüssel	Wert
Thema 2	Kostenstelle	43289
	Anwendung	Zahlungsverarbeitung
Thema 3	Kostenstelle	76585
	Anwendung	Archivierung

Dieses Markierungsschema ermöglicht es Ihnen, zwei Themen zu gruppieren, die verwandte Aufgaben für dieselbe Kostenstelle ausführen, während Sie eine unabhängige Aktivität mit einem anderen Tag für die Kostenzuordnung markieren.

Markierungen für die Zugriffssteuerung

AWS Identity and Access Management unterstützt die Steuerung des Zugriffs auf Ressourcen anhand von Tags. Geben Sie nach der Markierung Ihrer Ressourcen Informationen über die Ressourcen-Tags im Bedingungselement einer Identity and Access Management (IAM)-Richtlinie an, um den tagbasierten Zugriff zu verwalten. Weitere Informationen darüber, wie Sie Ihre Ressourcen mit der Amazon-SNS-Konsole oder dem AWS -SDK markieren, finden Sie unter Konfigurieren von Tags.

Sie können den Zugriff auf eine IAM-Identität einschränken. Sie können z. B. den Zugriff von `Publish` und `PublishBatch` auf alle Amazon-SNS-Themen einschränken, die ein Tag mit dem Schlüssel `environment`und dem Wert `production` enthalten, während gleichzeitig Zugriff auf alle anderen Amazon-SNS-Themen gewährt wird. Im folgenden Beispiel schränkt die Richtlinie die Möglichkeit ein, Nachrichten zu Themen zu veröffentlichen, die mit `production` markiert sind, während Nachrichten zu Themen, die mit `development`gekennzeichnet sind, veröffentlicht werden können. Weitere Informationen finden Sie unter Zugriffssteuerung mit Tags im IAM-Benutzerhandbuch.

> **Note**
>
> Das Einrichten der IAM-Berechtigung für `Publish` legt die Berechtigung für sowohl für `Publish` als auch für `PublishBatch` fest.

JSON

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

Markierungen für die Suche und Filterung von Ressourcen

Ein AWS Konto kann Zehntausende von Amazon SNS SNS-Themen enthalten (weitere Informationen finden Sie unter Amazon SNS SNS-Kontingente). Durch das Markieren Ihrer Themen erleichtern Sie sich das Durchsuchen oder Filtern von Themen.

Möglicherweise haben Sie z. B. hunderte Themen, die mit Ihrer Produktionsumgebung verknüpft sind. Anstatt manuell nach diesen Themen suchen zu müssen, können Sie alle Themen mit einem bestimmten Tag abfragen:

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

Konfigurieren von Tags für ein Amazon-SNS-Thema

In diesem Thema wird erklärt, wie Sie Tags für ein Amazon SNS SNS-Thema mithilfe von AWS Management Console, einem AWS SDK oder dem AWS CLI konfigurieren.

> **Important**
>
> Fügen Sie keine personenbezogenen Daten (Personally Identifiable Information, PII) oder andere vertrauliche Informationen in Tags hinzu. Tags sind für andere Amazon Web Services zugänglich, einschließlich der Abrechnung. Tags sind nicht für private oder vertrauliche Daten gedacht.

Auflisten, Hinzufügen und Entfernen von Tags für ein Amazon SNS SNS-Thema mithilfe der AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema und anschließend Edit (Bearbeiten) aus.
4. Erweitern Sie den Abschnitt Tags.

   Die dem Thema hinzugefügten Tags werden aufgelistet.
5. Ändern von Themen-Tags:
   - Um ein Tag hinzuzufügen, wählen Sie Add tag (Tag hinzufügen) aus und geben Sie einen Key (Schlüssel) und einen Value (Wert) (optional) ein.
   - Zum Entfernen eines Tags wählen Sie neben einem Schlüssel-Wert-Paar Remove tag (Tag entfernen) aus.
6. Wählen Sie Save Changes (Änderungen speichern) aus.

Hinzufügen von Tags zu einem Thema über ein AWS -SDK

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Thema ein Tag hinzu

Das folgende `tag-resource`-Beispiel fügt dem angegebenen Amazon-SNS-Thema ein Metadaten-Tag hinzu.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie TagResourceunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- API-Details finden Sie TagResourcein der API-Referenz zum AWS SDK für Kotlin.

Verwalten von Tags mit SNS-API-Aktionen von Amazon

Um Tags über die Amazon-SNS-API zu verwalten, verwenden Sie die folgenden API-Aktionen:

- ListTagsForResource
- TagResource
- UntagResource

API-Aktionen, die ABAC unterstützen

Im Folgenden finden Sie eine Liste von API-Aktionen, die Attribute-based Access Control (ABAC, attributbasierte Zugriffssteuerung) unterstützen. Weitere Informationen zu ABAC finden Sie unter Wozu dient ABAC? AWS im IAM-Benutzerhandbuch.

- AddPermission
- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe

- `UntagResource`

Amazon SNS Ereignisquellen und -ziele

Amazon SNS verbindet AWS-Services externe Systeme, indem es ereignisgesteuerte Benachrichtigungen weiterleitet. Amazon SNS empfängt Ereignisse aus verschiedenen Quellen AWS-Services, wie z. B. Daten-Pipeline-Updates, EC2 Skalierungsaktionen von Amazon oder Sicherheitswarnungen, und veröffentlicht diese Ereignisse zu Amazon SNS SNS-Themen. Diese Themen senden dann Benachrichtigungen an bestimmte Ziele.

Amazon SNS unterstützt zwei Haupttypen von Zielen: Application-to-Application (A2A) und Application-to-Person (A2P). Beim A2A-Messaging kann Amazon SNS Ereignisse an Lambda senden, um eine benutzerdefinierte Geschäftslogik auszulösen, an Amazon SQS, um Nachrichten in die Warteschlange zu stellen, und an Amazon Data Firehose, um Daten an Speicher- und Analysedienste zu streamen. Für A2P-Nachrichten kann Amazon SNS Benachrichtigungen per SMS, E-Mail und Push-Benachrichtigungen an mobile Geräte senden und so sicherstellen, dass Benutzer oder Teams rechtzeitig Benachrichtigungen erhalten.

Amazon SNS fungiert als zentraler Knotenpunkt und leitet Benachrichtigungen an die richtigen Stellen weiter, sodass Sie Ihre AWS Infrastruktur effektiver automatisieren und verwalten können. Dieses Setup ermöglicht eine nahtlose Integration zwischen Diensten und eine zuverlässige Kommunikation mit Benutzern und Systemen.

Amazon-SNS-Ereignisquellen

Amazon SNS lässt sich in eine Vielzahl von AWS-Services verschiedenen Kategorien integrieren, sodass diese Dienste Ereignisse zu Amazon SNS SNS-Themen veröffentlichen können. Diese Integration bietet Benachrichtigungen in Echtzeit über wichtige Ereignisse wie Änderungen der Infrastruktur, der Anwendungsleistung und des Kostenmanagements.

> **Note**
>
> Amazon SNS führte FIFO-Themen im Oktober 2020 ein. Derzeit unterstützen die meisten AWS Dienste das Senden von Ereignissen nur zu Standardthemen.

Analyseservices

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Analysedienste wie Athena und Amazon Redshift integriert wird AWS Data Pipeline, um Echtzeitbenachrichtigungen für wichtige

Ereignisse bereitzustellen, darunter Verstöße gegen Kontrolllimits, Pipeline-Status-Updates und Data Warehouse-Aktivitäten.

Sie können diese Integrationen nutzen, um Antworten zu automatisieren und einen effektiven Überblick über Ihre Datenoperationen zu behalten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Athena – Ermöglicht Ihnen die Analyse von Daten in Amazon S3 mit Standard-SQL.	Erhalten Sie Benachrichtigungen, wenn Kontrolllimits überschritten werden. Weitere Informationen finden Sie unter Festlegen von Limits zur Kontrolle der Datennutzung im Benutzerhandbuch zu Amazon Athena.
AWS Data Pipeline – Hilft bei der Automatisierung der Bewegung und Umwandlung von Daten.	Erhalten Sie Benachrichtigungen über den Status von Pipeline-Komponenten. Weitere Informationen finden Sie unter SnsAlarm im AWS Data Pipeline -Entwicklerhandbuch.
Amazon Redshift – Verwaltet alle Arbeiten zur Einrichtung, zum Betrieb und zum Skalieren eines Data Warehouse.	Erhalten Sie Benachrichtigungen über Amazon-Redshift-Ereignisse. Weitere Informationen finden Sie unter Amazon-Redshift-Ereignisbenachrichtigungen im Amazon-Redshift-Verwaltungshandbuch.

Services für die Anwendungsintegration

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in Anwendungsintegrationsdienste wie EventBridge und integriert wird und AWS Step Functions Datenweiterleitung und Benachrichtigungen für geschäftskritische Anwendungen in Echtzeit ermöglicht.

Sie können diese Integrationen nutzen, um Benachrichtigungen über EventBridge Ereignisse zu erhalten und Workflows mithilfe von Step Functions zu orchestrieren, wodurch die Automatisierung und Reaktionsfähigkeit Ihrer Anwendungen verbessert wird.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon EventBridge — Stellt einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, software-as-a-service (SaaS) -Anwendungen und AWS Diensten bereit und leitet diese Daten an Ziele weiter, einschlie ßlich Amazon SNS. EventBridge hieß früher CloudWatch Events.	Erhalten Sie Benachrichtigungen über EventBridge Ereignisse. Weitere Informationen finden Sie unter EventBridgeAmazon-Ziele im EventBridge Amazon-Benutzerhandbuch.
AWS Step Functions— Ermöglicht die Kombination von AWS Lambda Funktionen und anderen AWS Services zur Erstellung geschäftskritischer Anwendungen.	Erhalten Sie eine Benachrichtigung über Step Functions-Ereignisse. Weitere Informationen finden Sie unter Aufrufen von Amazon SNS mit Step Functions im AWS Step Functions Entwicklerhandbuch.

Fakturierungs- und Kostenmanagementservices

In der folgenden Tabelle wird die AWS Fakturierung und Kostenmanagement Integration mit Amazon SNS beschrieben, um Benachrichtigungen über Budgets, Preisänderungen und Kostenanomalien bereitzustellen.

Sie können diese Integration nutzen, um Amazon SNS SNS-Themen so einzurichten, dass Sie in Echtzeit Benachrichtigungen über Ihre AWS Ausgaben erhalten. So können Sie die Kosten überwachen und effizient auf unerwartete Gebühren reagieren.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Fakturierung und Kostenmanagement – Bietet Funktionen, mit denen Sie Ihre Kosten überwachen und Ihre Rechnung bezahlen können.	Erhalten Sie Budgetbenachrichtigungen, Preisänderungsbenachrichtigungen und Anomaliewarnungen. Weitere Informationen finden Sie auf den folgenden Seiten im AWS Billing -Benutzerhandbuch: • Erstellen eines Amazon-SNS-Themas für Budget-Benachrichtigungen

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	• Einrichten von Benachrichtigungen • Erkennung ungewöhnlicher Ausgaben mit AWS Cost Anomaly Detection

Services für Geschäftsanwendungen

In der folgenden Tabelle wird beschrieben, wie Amazon Chime in Amazon SNS integriert wird, um Benachrichtigungen für wichtige Besprechungsereignisse zu senden, sodass Sie über Ihre Kommunikation und Terminplanung auf dem Laufenden bleiben.

Sie können diese Integration nutzen, um Amazon Chime SDK-Ereignisbenachrichtigungen zu nutzen, um Ihre Tools für die Zusammenarbeit innerhalb und außerhalb Ihres Unternehmens zu verbessern.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Chime – Ermöglicht Ihnen, geschäftl iche Anrufe innerhalb und außerhalb Ihrer Organisation zu tätigen, zu chatten und sich zu treffen.	Erhalten Sie wichtige Benachrichtigungen über Meetingveranstaltungen. Weitere Informati onen finden Sie unter Amazon-Chime-SDK-E reignisbenachrichtigungen im Amazon-Chime-Entwicklerhandbuch.

Datenverarbeitungsservices

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in verschiedene AWS Rechendienste integriert wird, sodass Sie Benachrichtigungen für wichtige Ereignisse wie Auto Scaling Scaling-Aktionen, EC2 Image Builder Builder-Abschlüsse, Elastic Beanstalk Beanstalk-Umgebungsänderungen, Lambda-Funktionsausgaben und Lightsail-Metrik-Schwellenwerte erhalten können.

Sie können diese Integrationen nutzen, um Ihre Anwendungen und Ressourcen effizient zu verwalten, indem Sie über wichtige Updates und Aktionen auf dem Laufenden bleiben. AWS-Services

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon EC2 Auto Scaling — Hilft Ihnen, die richtige Anzahl von Amazon Elastic Compute Cloud (Amazon EC2) -Instances zur Verfügung zu haben, um die Last Ihrer Anwendung zu bewältigen.	Erhalten Sie Benachrichtigungen, wenn Auto Scaling EC2 Amazon-Instances in Ihrer Auto Scaling Scaling-Gruppe startet oder beendet. Weitere Informationen finden Sie unter Amazon SNS-Benachrichtigungen erhalten, wenn Ihre Auto Scaling-Gruppe skaliert wird im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.
EC2 Image Builder — Hilft bei der Automatisierung der Erstellung, Verwaltung und Bereitstellung von benutzerdefinierten, sicheren und up-to-date Server-Images, die mit Software und Einstellungen vorinstalliert und vorkonfiguriert sind, um bestimmte IT-Standards zu erfüllen.	Erhalten Sie Benachrichtigungen, wenn Builds abgeschlossen sind. Weitere Informationen finden Sie im AWS Compute-Blog unter Nachverfolgen der neuesten Serverimages in EC2 Image Builder Builder-Pipelines.
AWS Elastic Beanstalk – Übernimmt automatisch die Kapazitätsbereitstellung, Lastverteilung, Skalierung und Überwachung des Anwendungsstatus.	Erhalten Sie Benachrichtigungen zu wichtigen Ereignissen, die sich auf Ihre Anwendung auswirken. Weitere Informationen finden Sie unter Elastic-Beanstalk-Umgebungsbenachrichtigungen mit Amazon SNS im AWS Elastic Beanstalk Entwicklerhandbuch.
AWS Lambda – Ermöglicht die Ausführung von Code ohne Bereitstellung oder Verwaltung von Servern.	Erhalten Sie Funktionsausgabedaten, indem Sie ein SNS-Thema als Lambda-Warteschlange für unzustellbare Nachrichten oder als Lambda-Ziel festlegen. Weitere Informationen finden Sie unter Asynchroner Aufruf im AWS Lambda Entwicklerhandbuch.
Amazon Lightsail — Hilft Entwicklern beim Einstieg in AWS die Erstellung von Websites oder Webanwendungen.	Erhalten Sie Benachrichtigungen, wenn eine Metrik für eine Ihrer Instances, Datenbanken oder Load Balancer einen angegebenen Schwellenwert überschreitet. Weitere Informationen finden Sie unter Hinzufügen von Benachrichtigungskontakten in Amazon

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	Lightsail im Amazon-Lightsail-Entwickler handbuch.

Containerservices

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Container-Services wie Amazon EKS Distro und Amazon ECS integriert wird, sodass Sie Updates und Sicherheitspatches für Amazon EKS-Cluster verfolgen und Benachrichtigungen über neue ECS-optimierte AMI-Versionen erhalten können.

Sie können diese Integrationen nutzen, um die Sicherheit und Effizienz Ihrer Container-Bereitstellungen aufrechtzuerhalten, indem Sie über wichtige Updates und Änderungen auf dem Laufenden bleiben.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon EKS Distro – Ermöglicht Ihnen die Erstellung von zuverlässigen und sicheren Clustern, wo immer Ihre Anwendungen bereitgestellt werden.	Verfolgen Sie Updates und Sicherheits-Patches für Cluster, die mit Amazon EKS Distro erstellt wurden. Weitere Informationen finden Sie unter Einführung von Amazon EKS Distro – einer Open-Source-Kubernetes-Verteilung, verwendet von Amazon EKS.
Amazon Elastic Container Service (Amazon ECS) – Ermöglicht das Ausführen, Anhalten und Verwalten von Containern in einem Cluster.	Erhalten Sie Benachrichtigungen, wenn ein neues Amazon-ECS-optimiertes AMI verfügbar ist. Weitere Informationen finden Sie unter Abonnieren von Amazon-ECS-optimierten AMI-Update-Benachrichtigungen im Amazon-El astic-Container-Service-Entwicklerhandbuch.

Services zur Kundenbindung

In der folgenden Tabelle wird beschrieben, AWS Endbenutzer-Messaging SMS wie Amazon SNS die Kundenbindungsdienste durch die Integration mit Amazon Connect und Amazon Simple Email Service (SES) verbessert, sodass Sie Benachrichtigungen und Bestätigungen erhalten, bidirektionale

SMS-Nachrichten konfigurieren und E-Mail-Benachrichtigungen für Bounces, Beschwerden und Lieferungen überwachen können.

Diese Integrationen helfen Ihnen bei der Verwaltung der Kundenkommunikation über mehrere Kanäle hinweg.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Connect – Sie können ein Omnichannel-Cloud-Kontaktcenter einrichten, um mit Ihren Kunden zu interagieren.	Erhalten Sie Warnungen und Validierungen. Weitere Informationen finden Sie unter The power of AWS with Amazon Connect im Amazon Connect Connect-Administratorhandbuch.
AWS Endbenutzer-Messaging SMS— Hilft Ihnen, Ihre Kunden anzusprechen, indem Sie ihnen E-Mails, SMS und Sprachnachrichten sowie Push-Benachrichtigungen senden.	Konfigurieren Sie bidirektionale SMS-Nachrichten, sodass Sie Nachrichten von Kunden empfangen können. Weitere Informationen finden Sie im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Bidirektionale SMS-Nachrichten.
Amazon Simple Email Service (Amazon SES) – Bietet eine kostengünstige Möglichkeit, E-Mails mit Ihrer eigenen E-Mail-Adresse und Domänen zu senden und zu empfangen.	Empfangen von Benachrichtigungen zu Unzustellbarkeiten, Beschwerden und Zustellungen. Weitere Informationen finden Sie unter Konfigurieren von Amazon SNS-Benachrichtigungen für Amazon SES im Amazon-Simple-Email-Service-Entwicklerhandbuch.

Datenbankservices

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Datenbankdienste wie AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon, Amazon Neptune ElastiCache, Amazon Redshift und Amazon Relational Database Service (RDS) integriert wird, um Benachrichtigungen über wichtige Ereignisse wie Datenmigrationen, Wartungsaktivitäten, Cache-Updates und Datenbankänderungen zu senden.

Diese Integrationen helfen Ihnen dabei, Ihre Datenbankumgebungen effektiver zu überwachen und zu verwalten, indem sie rechtzeitig über wichtige betriebliche Ereignisse informieren.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Database Migration Service— Migriert Daten aus lokalen Datenbanken in die. AWS Cloud	Erhalten Sie Benachrichtigungen, wenn AWS DMS Ereignisse eintreten, z. B. wenn eine Replikationsinstanz erstellt oder gelöscht wird. Weitere Informationen finden Sie unter Arbeiten mit Ereignissen und Benachrichtigungen in AWS Database Migration Service im AWS Database Migration Service Benutzerhandbuch.
Amazon DynamoDB – Bietet eine schnelle und planbare Leistung mit nahtloser Skalierbarkeit in diesem vollständig verwalteten NoSQL-Datenbank-Service.	Erhalten Sie Benachrichtigungen, wenn Wartungsereignisse auftreten. Weitere Informationen finden Sie unter Anpassen der DAX-Cluster-Einstellungen im Amazon-DynamoDB-Entwicklerhandbuch.
Amazon ElastiCache — Bietet einen leistungsstarken, skalierbaren und kostengünstigen In-Memory-Cache und verringert gleichzeitig die Komplexität, die mit der Bereitstellung und Verwaltung einer verteilten Cache-Umgebung verbunden ist.	Erhalten Sie Benachrichtigungen, wenn wichtige Ereignisse auftreten. Weitere Informationen finden Sie unter Event-Benachrichtigungen und Amazon SNS im Amazon ElastiCache (Memcached) User Guide.
Amazon Neptune – Sie können Anwendungen erstellen und ausführen, die mit hochgradig verbundenen Datensätzen funktionieren.	Erhalten Sie Benachrichtigungen, wenn ein Neptune-Ereignis eintritt. Weitere Informationen finden Sie unter Verwenden von Neptune-Ereignisbenachrichtigungen im Neptune-Benutzerhandbuch.
Amazon Redshift – Verwaltet alle Arbeiten zur Einrichtung, zum Betrieb und zum Skalieren eines Data Warehouse.	Erhalten Sie Benachrichtigungen über Amazon-Redshift-Ereignisse. Weitere Informationen finden Sie unter Amazon-Redshift-Ereignisbenachrichtigungen im Amazon-Redshift-Verwaltungshandbuch.
Amazon Relational Database Service — Erleichtert die Einrichtung, den Betrieb und die	Erhalten Sie Benachrichtigungen über Amazon-RDS-Ereignisse. Weitere Informationen finden

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Skalierung einer relationalen Datenbank in der AWS Cloud.	Sie unter Verwenden von Amazon-RDS-Ereignisbenachrichtigungen im Amazon-RDS-Benutzerhandbuch.

Services für Entwickler-Tools

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Entwicklertools-Dienste wie,, AWS CodeBuild AWS CodeCommit AWS CodeDeploy CodeGuru, Amazon integriert wird und AWS CodePipeline Benachrichtigungen für kritische Ereignisse wie Änderungen des Build-Status, Repository-Updates, Bereitstellungsfortschritt, Leistungsanomalien und Pipeline-Aktionen bereitstellt.

Diese Integrationen helfen Ihnen dabei, Ihre Workflows bei der Softwareentwicklung effizient zu überwachen und zu verwalten, indem Sie rechtzeitig über wichtige Ereignisse informiert werden.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS CodeBuild – Kompiliert den Quellcode, führt Komponententests aus und erzeugt Artefakte, die bereitgestellt werden können.	Erhalten Sie Benachrichtigungen, wenn Builds erfolgreich sind, fehlschlagen oder von einer Build-Phase in eine andere verschoben werden. Weitere Informationen finden Sie unter Beispiel für Build-Benachrichtigungen CodeBuild im AWS CodeBuild Benutzerhandbuch.
AWS CodeCommit – Bietet Versionskontrolle für die private Speicherung und Verwaltung von Assets in der Cloud.	Erhalten Sie Benachrichtigungen über CodeCommit Repository-Ereignisse. Weitere Informationen finden Sie unter Beispiel: Einen AWS CodeCommit Auslöser für ein Amazon SNS SNS-Thema erstellen im AWS CodeCommit Benutzerhandbuch.
AWS CodeDeploy— Automatisiert Anwendungsbereitstellungen auf EC2 Amazon-Instances, lokalen Instances, serverlosen Lambda-Funktionen oder Amazon ECS-Services.	Erhalten Sie Benachrichtigungen für CodeDeploy Bereitstellungen oder Instance-Ereignisse. Weitere Informationen finden Sie im AWS CodeDeploy Benutzerhandbuch unter

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	Einen Auslöser für ein CodeDeploy Ereignis erstellen.
Amazon CodeGuru — Sammelt Laufzeitl eistungsdaten von Ihren Live-Anwendungen und gibt Empfehlungen, die Ihnen bei der Feinabstimmung Ihrer Anwendungsleistung helfen können.	Erhalten Sie Benachrichtigungen, wenn Anomalien auftreten. Weitere Informationen finden Sie unter Arbeiten mit Anomalien und Empfehlungsberichten im CodeGuru Amazon-Benutzerhandbuch.
AWS CodePipeline – Automatisiert die Schritte, die erforderlich sind, um Softwareänderungen kontinuierlich freizugeben.	Erhalten Sie Benachrichtigungen über Genehmigungsaktionen. Weitere Informationen finden Sie CodePipelineim AWS CodePipeline Benutzerhandbuch unter Genehmigungsaktion en verwalten.

Front-End-Web- und Mobildienste

In der folgenden Tabelle wird beschrieben, wie Amazon SNS integriert wird, AWS Endbenutzer-Messaging SMS um die Kundenbindung durch das Senden von E-Mails, SMS, Sprachnachrichten und Push-Benachrichtigungen zu verbessern, einschließlich der Möglichkeit, bidirektionale SMS für den Empfang von Kundennachrichten zu konfigurieren.

Diese Integration ermöglicht Ihnen eine effektivere Interaktion mit Ihren Kunden über verschiedene Kommunikationskanäle.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Endbenutzer-Messaging SMS— Hilft Ihnen, Ihre Kunden anzusprechen, indem Sie ihnen E-Mails, SMS und Sprachnachrichten sowie Push-Benachrichtigungen senden.	Konfigurieren Sie bidirektionale SMS-Nachr ichten, sodass Sie Nachrichten von Kunden empfangen können. Weitere Informationen finden Sie im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Bidirektionale SMS-Nachrichten.

Services für Spieleentwicklung

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in GameLift Amazon-Server integriert wird, um Benachrichtigungen für Matchmaking- und Warteschlangenereignisse auf sitzungsbasierten Multiplayer-Spieleservern bereitzustellen.

Diese Integration hilft Spieleentwicklern dabei, die Bereitstellung, den Betrieb und die Skalierung ihrer Spieleserver zu automatisieren und zu überwachen und so ein nahtloses Spielerlebnis zu gewährleisten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon GameLift Servers — Bietet Lösungen für das Hosten von sitzungsbasierten Multiplayer-Spieleservern in der Cloud, einschließlich eines vollständig verwalteten Dienstes für die Bereitstellung, den Betrieb und die Skalierung von Spieleservern.	Erhalten Sie Benachrichtigungen über Matchmaking und Warteschlangenereignisse. Weitere Informationen finden Sie auf den folgenden Seiten: • Informationen zu Matchmaking-Benachrichtigungen finden Sie unter FlexMatch Event-Benachrichtigung einrichten im Amazon GameLift Servers FlexMatch Developer Guide. • Informationen zu Warteschlangenbenachrichtigungen finden Sie unter Ereignisbenachrichtigung für die Platzierung von Spielsitzungen einrichten im Amazon GameLift Servers Developer Guide.

Internet-of-Things-Services

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS IoT Dienste wie,, und integriert wird AWS IoT Core AWS IoT Device Defender AWS IoT Events AWS IoT Greengrass, um Benachrichtigungen für IoT-Ereignisse und -Warnungen bereitzustellen.

Diese Integrationen ermöglichen es Ihnen, das Geräteverhalten effektiv zu überwachen, Benachrichtigungen über ungewöhnliche Aktivitäten zu erhalten und IoT-Geräte mit Updates und Aktionen in Echtzeit zu verwalten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS IoT Core— Stellt die Cloud-Dienste bereit, die Ihre IoT-Geräte mit anderen Geräten und AWS Cloud Diensten verbinden.	Erhalten Sie Benachrichtigungen über AWS IoT Core Ereignisse. Weitere Informationen finden Sie unter Erstellen einer Amazon SNS-Regel im AWS IoT Entwicklerhandbuch.
AWS IoT Device Defender – Ermöglicht das Überprüfen der Konfiguration Ihrer Geräte und Überwachen der verbundenen Geräte, um anormales Verhalten zu ermitteln und Sicherheitsrisiken zu minimieren.	Erhalten Sie Alarme, wenn ein Gerät gegen ein Verhalten verstößt. Weitere Informationen finden Sie im AWS IoT Entwicklerhandbuch unter So verwenden Sie AWS IoT Device Defender Detect.
AWS IoT Events – Ermöglicht Ihnen die Überwachung Ihrer Geräte und Geräteflotten auf Fehler oder Änderungen im Betrieb sowie für das Auslösen von Aktionen, wenn solche Ereignisse auftreten.	Erhalten Sie Benachrichtigungen über AWS IoT Events Ereignisse. Weitere Informationen finden Sie unter Amazon Simple Notification Service im AWS IoT Events Entwicklerhandbuch.
AWS IoT Greengrass— Erstreckt sich AWS auf physische Geräte, sodass sie lokal auf die von ihnen generierten Daten reagieren und gleichzeitig die Cloud für Verwaltung, Analyse und dauerhaften Speicher nutzen können.	Erhalten Sie Benachrichtigungen über AWS IoT Greengrass Ereignisse. Weitere Informationen finden Sie unter SNS-Konnektor im AWS IoT Greengrass Version 1 Entwicklerhandbuch.

Machine-Learning-Services

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in Dienste für AWS maschinelles Lernen wie Amazon, Amazon DevOps Guru CodeGuru, Amazon Lookout for Metrics, Amazon Rekognition und Amazon SageMaker AI integriert wird, um Benachrichtigungen über Anomalien, betriebliche Einblicke und Datenkennzeichnungsaktivitäten bereitzustellen.

Diese Integrationen ermöglichen es Ihnen, die Anwendungsleistung zu überwachen, Warnmeldungen bei Datenunregelmäßigkeiten zu erhalten und die Bereitstellung von Modellen für maschinelles Lernen mit Aktualisierungen in Echtzeit zu optimieren.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon CodeGuru — Sammelt Laufzeitl eistungsdaten von Ihren Live-Anwendungen und gibt Empfehlungen, die Ihnen bei der Feinabstimmung Ihrer Anwendungsleistung helfen können.	Erhalten Sie Benachrichtigungen, wenn Anomalien auftreten. Weitere Informationen finden Sie unter Arbeiten mit Anomalien und Empfehlungsberichten im CodeGuru Amazon-Benutzerhandbuch.
Amazon DevOps Guru — Generiert mithilfe von maschinellem Lernen betriebliche Erkenntni sse, um Ihnen zu helfen, die Leistung Ihrer betrieblichen Anwendungen zu verbessern.	Weiterleiten von Erkenntnissen und Bestätigu ngen. Weitere Informationen finden Sie im AWS Management & Governance-Blog unter Bereitstellung von ML-gestützten betrieblichen Erkenntnissen für Ihre Bereitschaftsteams PagerDuty mithilfe von Amazon DevOps Guru.
Amazon Lookout for Metrics – Findet Anomalien in Ihren Daten, bestimmt deren Ursachen und ermöglicht Ihnen, schnell Maßnahmen zu ergreifen.	Erhalten Sie Benachrichtigungen über Anomalien. Weitere Informationen finden Sie unter Verwenden von Amazon SNS mit Lookout for Metrics im Amazon Lookout for Metrics Entwicklerhandbuch.
Amazon Rekognition – Sie können Bild- und Videoanalysen zu Ihren Anwendungen hinzufügen	Erhalten Sie Benachrichtigungen über Anforderungsergebnisse. Weitere Informati onen finden Sie unter Referenz: Videoanal yse Ergebnisbenachrichtigung im Amazon-Re kognition-Entwicklerhandbuch.
Amazon SageMaker AI — Ermöglicht es Datenwissenschaftlern und Entwicklern, Modelle für maschinelles Lernen zu erstellen und zu trainieren und sie dann direkt in einer produktionsbereiten gehosteten Umgebung bereitzustellen.	Erhalten Sie Benachrichtigungen, wenn ein Datenobjekt gekennzeichnet ist. Weitere Informationen finden Sie unter Erstellen eines Streaming-Labeling-Jobs im Amazon SageMaker AI Developer Guide.

Management- und Governance-Services

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Management- und Governance-Dienste wie Amazon Q Developer in Chat-Anwendungen AWS CloudFormation, CloudTrail, CloudWatch, AWS Config,, AWS Control Tower, AWS License Manager AWS Service Catalog AWS Systems Manager, und integriert wird und Benachrichtigungen für wichtige Ereignisse wie Infrastrukturänderungen, Compliance-Benachrichtigungen und betriebliche Einblicke bereitstellt.

Diese Integrationen helfen Ihnen dabei, Ihre AWS Umgebungen effizient zu überwachen und zu verwalten, indem sie rechtzeitig Benachrichtigungen und Updates an die entsprechenden Teams und Systeme senden.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Q Developer in Chat-Anwendungen — Ermöglicht es DevOps und Softwaree ntwicklungsteams, Amazon Chime- und Slack-Chatrooms zu verwenden, um betriebliche Ereignisse in den zu überwachen und darauf zu reagieren. AWS Cloud	Senden von Benachrichtigungen an Chatrooms . Weitere Informationen finden Sie unter Amazon Q Developer in Chat-Anwendungen einrichten im Administratorhandbuch für Amazon Q Developer in Chat-Anwendungen.
AWS CloudFormation— Ermöglicht es Ihnen, AWS Infrastrukturbereitstellungen vorhersehbar und wiederholt zu erstellen und bereitzustellen.	Erhalten Sie Benachrichtigungen, wenn Stacks erstellt und aktualisiert werden. Weitere Informationen finden Sie unter Einrichten von AWS CloudFormation Stack-Optionen im AWS CloudFormation -Benutzerhandbuch.
AWS CloudTrail – Bietet den Ereignisverlauf Ihrer AWS-Konto -Aktivität.	Erhalten Sie Benachrichtigungen, wenn neue Protokolldateien in Ihrem Amazon S3 S3-Bucket CloudTrail veröffentlicht werden. Weitere Informationen finden Sie unter Konfiguration von Amazon SNS SNS-Benac hrichtigungen für CloudTrail im AWS CloudTrail Benutzerhandbuch.
Amazon CloudWatch — Überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit.	Erhalten Sie Benachrichtigungen, wenn Aufträge den Status ändern. Weitere Informati onen finden Sie unter Verwenden von

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	CloudWatch Amazon-Alarmen im CloudWatch Amazon-Benutzerhandbuch.
AWS Config— Bietet einen detaillierten Überblick über die Konfiguration der AWS Ressourcen in Ihrem AWS-Konto.	Benachrichtigungen erhalten, wenn Ressourcen aktualisiert werden oder wenn AWS Config benutzerdefinierte oder verwaltete Regeln anhand Ihrer Ressourcen ausgewertet werden. Weitere Informationen finden Sie unter Benachrichtigungen, die AWS Config an ein SNS gesendet werden, und Beispielbenachrichtigungen über Änderungen an Konfigurationselementen im AWS Config Entwicklerhandbuch.
AWS Control Tower— Ermöglicht es Ihnen, eine sichere, konforme Umgebung mit mehreren Konten AWS einzurichten und zu verwalten.	Verwenden Sie Warnungen, um Drift innerhalb Ihrer Landing Zone zu verhindern und Compliance-Benachrichtigungen zu erhalten. Weitere Informationen finden Sie unter Nachverfolgen von Warnungen mit Amazon Simple Notification Service im AWS Control Tower Benutzerhandbuch.
AWS License Manager— Unterstützt Sie bei der zentralen Verwaltung Ihrer Softwarelizenzen von Softwareanbietern in AWS Ihren lokalen Umgebungen.	Erhalten Sie LicenseManager-Benachrichtigungen- und Warnungen. Weitere Informationen finden Sie unter Einstellungen in License Manager im License Manager Manager-Benutzerhandbuch und Erstellen von ServiceNow Vorfällen für AWS License Manager Benachrichtigungen im AWS Management & Governance-Blog.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Service Catalog – Damit können IT-Administratoren Portfolios genehmigter Produkte erstellen, verwalten und für Endbenutzer verteilen, die dann in einem personalisierten Portal Zugriff auf die jeweils benötigten Produkte haben.	Erhalten Sie Benachrichtigungen über Stack-Ereignisse. Weitere Informationen finden Sie unter AWS Service Catalog Benachrichtigungseinschränkungen im Servicekatalog-Administrator-Handbuch.
AWS Systems Manager— Ermöglicht es Ihnen, Ihre Infrastruktur einzusehen und zu kontrollieren AWS.	Erhalten Sie Benachrichtigungen über den Status von Befehlen. Weitere Informationen finden Sie unter Überwachen von Systems-Manager-Statusänderungen mit Amazon-SNS-Benachrichtigungen im AWS Systems Manager Benutzerhandbuch.

Medienservices

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in Amazon Elastic Transcoder integriert wird, um Benachrichtigungen zu senden, wenn sich der Status von Medientranscodierungsaufträgen ändert. Auf diese Weise können Sie die Konvertierung von in Amazon S3 gespeicherten Mediendateien in Formate, die für Wiedergabegeräte für Privatanwender geeignet sind, effizient überwachen und verwalten.

Diese Integration hilft Ihnen dabei, Arbeitsabläufe bei der Medienverarbeitung zu optimieren, indem Sie Benachrichtigungen in Echtzeit über den Auftragsstatus erhalten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Elastic Transcoder – Damit können Sie Mediendateien, die in Amazon S3 gespeichert werden, in ein Mediendateienformat konvertieren, das für Wiedergabegeräte der Kunden geeignet ist.	Erhalten Sie Benachrichtigungen, wenn Aufträge den Status ändern. Weitere Informationen finden Sie unter Benachrichtigungen über den Status eines Auftrags im Amazon-Elastic-Transcoder-Entwicklerhandbuch.

Migrations- und Übertragungsservices

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Migrations- und Übertragungsservices wie AWS Application Discovery Service, AWS Database Migration Service (DMS) integriert wird und AWS Snowball Edge Benachrichtigungen für Ereignisse wie Serverdatenerfassung, Datenbankmigrationsaktivitäten und Datenübertragungsaufträge bereitstellt.

Diese Integrationen helfen Ihnen dabei, Ihre Cloud-Migrationsprozesse effektiv zu verwalten und zu überwachen, indem sie Benachrichtigungen und Updates in Echtzeit zu wichtigen Migrationsaufgaben bieten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Application Discovery Service— Hilft Ihnen bei der Planung Ihrer Migration zum, AWS Cloud indem es Nutzungs- und Konfigura tionsdaten über Ihre lokalen Server sammelt.	Erhalten Sie Benachrichtigungen über Ereignisse über AWS CloudTrail. Weitere Informationen finden Sie unter Protokollieren von Application-Discovery-Service-API-Au frufen mit AWS CloudTrail im Application-Discovery-Service-Benutzerhandbuch.
AWS Database Migration Service— Migriert Daten aus lokalen Datenbanken in die. AWS Cloud	Erhalten Sie Benachrichtigungen, wenn AWS DMS Ereignisse eintreten, z. B. wenn eine Replikationsinstanz erstellt oder gelöscht wird. Weitere Informationen finden Sie unter Arbeiten mit Ereignissen und Benachric htigungen in AWS Database Migration Service im AWS Database Migration Service Benutzerh andbuch.
AWS Snowball Edge— Verwendet physische Speichergeräte, um große Datenmengen schnell zwischen Amazon S3 und Ihrem Datenspeicherort vor Ort zu faster-than-internet übertragen.	Erhalten Sie Benachrichtigungen für Snowball Edge-Jobs. Weitere Informationen finden Sie unter Benachrichtigungen für Snow Family-Geräte im AWS Snowball Edge Benutzerh andbuch.

Services für Netzwerke und zur Bereitstellung von Inhalten

In der folgenden Tabelle wird beschrieben, wie Amazon SNS mit AWS Netzwerk- und Inhaltsbereitstellungsdiensten wie Amazon API Gateway, Amazon, Elastic Load Balancing CloudFront AWS Direct Connect, Amazon Route 53 und Amazon VPC integriert wird, um Benachrichtigungen für Ereignisse wie API-Nachrichten, CloudFront metrische Alarme, Verbindungsstatusänderungen, Load Balancer-Ereignisse, Zustandsprüfungsstatus und VPC-Endpunktaktivitäten zu senden.

Diese Integrationen helfen Ihnen, Ihre Netzwerk- und Inhaltsbereitstellungsvorgänge zu überwachen und zu verwalten, indem sie zeitnahe Benachrichtigungen und Updates bereitstellen.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon API Gateway — Ermöglicht es Ihnen, Ihr eigenes REST in beliebiger Größenordnung WebSocket APIs zu erstellen und bereitzus tellen.	Empfangen von Nachrichten, die an einen API-Gateway-Endpunkt gepostet wurden. Weitere Informationen finden Sie unter Tutorial: Erstellen einer API-Gateway-REST-API mit AWS Integration im API Gateway Developer Guide.
Amazon CloudFront — Beschleunigt die Verteilung Ihrer statischen und dynamischen Webinhalte wie .html-, .css-, .php-, Bild- und Mediendateien.	Erhalten Sie Benachrichtigungen, wenn Alarme auf der Grundlage bestimmter CloudFront Messwerte auftreten. Weitere Informationen finden Sie unter Alarme für den Empfang von Benachrichtigungen einrichten im Amazon CloudFront Developer Guide.
AWS Direct Connect— Verbindet Ihr internes Netzwerk über ein Standard-Ethernet-Glasfaser kabel mit einem AWS Direct Connect Standort.	Erhalten Sie Benachrichtigungen, wenn Alarme, die den Status einer AWS Direct Connect -Verbindung überwachen, den Status ändern. Weitere Informationen finden Sie im AWS Direct Connect Benutzerhandbuch unter Erstellen von CloudWatch Alarmen zur Überwachung von AWS Direct Connect Verbindungen.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Elastic Load Balancing — Verteilt Ihren eingehenden Traffic automatisch auf mehrere Ziele, wie EC2 Amazon-Instances, Container und IP-Adressen, in mehr oder mehr Availability Zones.	Erhalten Sie Benachrichtigungen über Alarme, die Sie für Load-Balancer-Ereignisse erstellt haben. Weitere Informationen finden Sie unter CloudWatch Alarme für Ihren Load Balancer erstellen im Benutzerhandbuch für Classic Load Balancers.
Amazon Route 53 – Bietet Domänenregistrierung, DNS-Routing und Zustandsprüfung.	Erhalten Sie Benachrichtigungen, wenn die Zustandsprüfung fehlerhaft ist. Weitere Informationen finden Sie unter So erhalten Sie eine Amazon-SNS-Benachrichtigung, wenn bei der Zustandsprüfung der Status fehlerhaft ist (Konsole) im Amazon Route 53 Entwickelhandbuch.
Amazon Virtual Private Cloud (Amazon VPC) — Ermöglicht es Ihnen, AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk bereitzustellen.	Sie können eine Benachrichtigung erstellen, um Warnungen bei bestimmten Ereignissen zu erhalten, die auf Ihrem Schnittstellenendpunkt stattfinden. Weitere Informationen finden Sie unter Erstellen und verwalten einer Benachrichtigung für einen Endpunkt-Service im Amazon-VPC-Benutzerhandbuch.

Sicherheits-, Identitäts- und Compliance-Services

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in AWS Sicherheits-, Identitäts- und Compliance-Dienste wie AWS Directory Service Amazon und Amazon GuardDuty Inspector integriert wird AWS Security Hub, um Benachrichtigungen über Änderungen des Verzeichnisstatus, Sicherheitsfeststellungen, Inspector-Ereignisse und Ankündigungen des Security Hubs bereitzustellen.

Diese Integrationen helfen Ihnen dabei, robuste Sicherheitspraktiken aufrechtzuerhalten, indem sie zeitnahe Benachrichtigungen und Updates zu Sicherheits- und Compliance-Ereignissen bieten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Directory Service— Bietet mehrere Möglichkeiten, Microsoft Active Directory (AD) mit anderen zu verwenden AWS-Services.	Erhalten Sie E-Mail- oder Text- (SMS-)Nachrichten, wenn sich der Status Ihres Verzeichnisses ändert. Weitere Informationen finden Sie unter Konfigurieren von Benachrichtigungen über den Verzeichnisstatus im AWS Directory Service Administration-Handbuch.
Amazon GuardDuty — bietet eine kontinuierliche Sicherheitsüberwachung, um unerwartete und potenziell nicht autorisierte oder böswillige Aktivitäten in Ihrer AWS Umgebung zu identifizieren.	Erhalten Sie Benachrichtigungen über neu veröffentlichte Suchtypen, Updates der vorhandenen Suchtypen und andere Änderungen in den Funktionalitäten. Weitere Informationen finden Sie unter dem Thema GuardDuty Ankündigungen abonnieren im GuardDuty Amazon-Benutzerhandbuch.
Amazon Inspector — Testet die Netzwerkzugänglichkeit Ihrer EC2 Amazon-Instances und den Sicherheitsstatus Ihrer Anwendungen, die auf diesen Instances ausgeführt werden.	Erhalten Sie Benachrichtigungen für Amazon Inspector Ereignisse. Weitere Informationen finden Sie unter Einrichten eines SNS-Themas für Amazon-Inspector-Benachrichtigungen im Amazon-Inspector-Benutzerhandbuch.
AWS Security Hub: Automatisiert AWS - Sicherheitsüberprüfungen und zentralisiert Sicherheitswarnungen.	Erhalten Sie Benachrichtigungen über AWS Security Hub Ankündigungen, einschließlich Benachrichtigungen über AWS Security Hub Kontrollen oder Standards, die hinzugefügt, bearbeitet oder eingestellt wurden. Weitere Informationen finden Sie unter Abonnieren von AWS Security Hub Ankündigungen mit Amazon SNS.

Serverless-Services

In der folgenden Tabelle wird beschrieben, wie Amazon SNS in Dienste wie Amazon DynamoDB, Amazon und Lambda integriert wird EventBridge, um Benachrichtigungen für wichtige Ereignisse wie Wartungsupdates, Echtzeit-Datenströme und Lambda-Funktionsausgaben zu senden.

Diese Integrationen helfen Ihnen dabei, Ihre Anwendungen effizient zu überwachen und zu verwalten, indem sie rechtzeitig Benachrichtigungen zu kritischen Vorgängen erhalten und die Reaktionen auf diese Ereignisse automatisieren.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon DynamoDB – Bietet eine schnelle und planbare Leistung mit nahtloser Skalierbarkeit in diesem vollständig verwalteten NoSQL-Datenbank-Service.	Erhalten Sie Benachrichtigungen, wenn Wartungsereignisse auftreten. Weitere Informationen finden Sie unter Anpassen der DAX-Cluster-Einstellungen im Amazon-DynamoDB-Entwicklerhandbuch.
Amazon EventBridge — Stellt einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen software-as-a-service (SaaS) bereit AWS-Services und leitet diese Daten an Ziele weiter, einschließlich Amazon SNS. EventBridge hieß früher CloudWatch Events.	Erhalten Sie Benachrichtigungen über EventBridge Ereignisse. Weitere Informationen finden Sie unter EventBridgeAmazon-Ziele im EventBridge Amazon-Benutzerhandbuch.
AWS Lambda – Ermöglicht die Ausführung von Code ohne Bereitstellung oder Verwaltung von Servern.	Erhalten Sie Funktionsausgabedaten, indem Sie ein SNS-Thema als Lambda-Warteschlange für unzustellbare Nachrichten oder als Lambda-Ziel festlegen. Weitere Informationen finden Sie unter Asynchroner Aufruf im AWS Lambda Entwicklerhandbuch.

Speicherservices

In der folgenden Tabelle wird beschrieben AWS Backup, wie Amazon SNS mit AWS Speicherdiensten wie Amazon Elastic File System (EFS), Amazon S3 Glacier und Amazon S3 integriert wird und AWS Snowball Edge Benachrichtigungen für verschiedene Ereignisse wie Backup-Aktivitäten, Dateisystemalalarme, Datenabrufjobs, Bucket-Änderungen und Datenübertragungsvorgänge bereitstellt.

Diese Integrationen helfen Ihnen dabei, Ihre Speicherlösungen effizient zu überwachen und zu verwalten, indem sie rechtzeitig Benachrichtigungen über kritische Speicherereignisse erhalten.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Backup— Unterstützt Sie bei der Zentralisierung und Automatisierung der Datensicherung AWS-Services in der Cloud und vor Ort	Erhalten Sie Benachrichtigungen über AWS Backup Ereignisse. Weitere Informationen finden Sie unter Verwenden von Amazon SNS zur Nachverfolgung von AWS Backup Ereignissen im AWS Backup Entwicklerhandbuch.
Amazon Elastic File System — Bietet Dateispeicher für Ihre EC2 Amazon-Instances.	Erhalten Sie Benachrichtigungen über Alarme, die Sie für Amazon-EFS-Ereignisse erstellt haben. Weitere Informationen finden Sie unter Automatisierte Überwachungstools im Amazon-Elastic-File-System-Benutzerhandbuch.
Amazon S3 Glacier – Stellt Speicher für selten verwendete Daten bereit.	Sie können eine Benachrichtigungskonfiguration für einen Tresor einrichten, so dass nach Abschluss eines Auftrags eine Nachricht an ein SNS-Thema gesendet wird. Weitere Informationen finden Sie unter Konfigurieren von Tresorbenachrichtigungen in Amazon S3 Glacier im Amazon-S3-Glacier-Entwickelrhandbuch.
Amazon Simple Storage Service (Amazon S3) – Bietet Objektspeicherung.	Erhalten Sie Benachrichtigungen, wenn Änderungen an einem Amazon-S3-Bucket auftreten oder in dem seltenen Fall, dass Objekte nicht in ihre Zielregion repliziert werden. Weitere Informationen finden Sie unter Walkthrough: Konfigurieren eines Buckets für Benachrichtigungen (SNS-Thema und SQS-Warteschlange) und Überwachung des Fortschritts mit Replikationsmetriken und Amazon-S3-Ereignisbenachrichtigungen im Benutzerhandbuch zur Amazon-Simple-Storage-Service-Konsole.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Snowball Edge— Verwendet physische Speichergeräte, um große Datenmengen schnell zwischen Amazon S3 und Ihrem Datenspeicherort vor Ort zu faster-than-internet übertragen.	Erhalten Sie Benachrichtigungen für Snowball Edge-Jobs. Weitere Informationen finden Sie unter Benachrichtigungen für Snow Family-Geräte im AWS Snowball Edge Benutzerh andbuch.

Zusätzliche Ereignisquellen

In der folgenden Tabelle wird beschrieben, wie Amazon SNS verwendet werden kann, um zeitnahe Benachrichtigungen über AWS tägliche Funktionsupdates und Änderungen der AWS IP-Adressbereiche zu erhalten, sodass Sie über die neuesten AWS Service Releases, Instance-Typen, VPC-Endpunkte und Änderungen der öffentlichen IP-Adressen informiert sind.

Diese Integrationen helfen Ihnen dabei, up-to-date mit Änderungen an der AWS Infrastruktur Schritt zu halten und Ihre Ressourcen effektiv zu verwalten.

Quelle	Vorteile der Verwendung mit Amazon SNS
AWS Tägliche Funktionsupdates	Erhalten Sie über ein Amazon-SNS-Thema zeitnah detaillierte Informationen zu Veröffent lichungen und Updates für AWS . Zu diesen Versionen gehören Amazon VPC-Endpu nkte AWS-Regionen AWS-Services, die in AWS Service Quotas AWS-Services integriert sind, Amazon-Instance-Typen, Amazon EC2 SageMaker AI-Instance-Typen, Amazon Nimble Studio-Instance-Typen, Amazon RDS-Daten bank-Engine-Versionen und Amazon MSK Apache Kafka-Versionen. Weitere Informati onen finden Sie unter Abonnieren von AWS täglichen Funktionsupdates über Amazon SNS im AWS News-Blog.
AWS IP-Adressbereiche	Erhalten Sie Benachrichtigungen über Änderungen an AWS IP-Bereichen über ein

Quelle	Vorteile der Verwendung mit Amazon SNS
	Amazon SNS SNS-Thema. Weitere Informationen finden Sie unter Benachrichtigungen zu AWS IP-Adressbereichen im Allgemeine Amazon Web Services-Referenzund Abonnieren von Änderungen AWS öffentlicher IP-Adressen über Amazon SNS im AWS News-Blog.

Weitere Informationen zum ereignisgesteuerten Computing finden Sie in den folgenden Quellen:

- Was ist eine ereignisgesteuerte Architektur?
- Event-Driven Computing mit Amazon SNS und AWS Rechen-, Speicher-, Datenbank- und Netzwerkservices im Compute-Blog AWS
- Bereicherung ereignisgesteuerter Architekturen mit AWS Event Fork-Pipelines im Compute-Blog AWS

Amazon SNS Eventziele

In diesem Thema sind alle Veranstaltungsziele aufgeführt, geordnet nach application-to-application (A2A-) Nachrichten und application-to-person (A2P -) Benachrichtigungen.

> **Note**
>
> Amazon SNS führte FIFO-Themen im Oktober 2020 ein. Derzeit unterstützen die meisten AWS Dienste nur den Empfang von Ereignissen zu SNS-Standardthemen. Amazon SQS unterstützt den Empfang von Ereignissen aus SNS-Standard- und FIFO-Themen.

A2A-Ziele

In der folgenden Tabelle wird beschrieben, wie Amazon SNS Ereignisse an verschiedene application-to-application (A2A) Ziele wie Lambda, Amazon SQS, AWS Event Fork-Pipelines und Endpunkte übermitteln kann. HTTP/S

Diese Integrationen ermöglichen es Ihnen, Daten zu archivieren und zu analysieren, benutzerdefinierte Geschäftslogik auszulösen, die Anwendungsintegration zu erleichtern

und Ereignisse an externe Webhooks weiterzuleiten, wodurch die Effizienz und Flexibilität ereignisgesteuerter Architekturen verbessert wird.

Eventziel	Vorteile der Verwendung mit Amazon SNS
https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html	Bereitstellen von Ereignissen für Delivery Streams zu Archivierungs- und Analysezwecken. Über Delivery Streams können Sie Ereignisse an AWS Ziele wie Amazon Simple Storage Service (Amazon S3), Amazon Redshift und Amazon OpenSearch Service (OpenSearch Service) oder an Ziele von Drittanbietern wie Datadog, New Relic, MongoDB und Splunk versenden. Weitere Informationen finden Sie unter Lieferstreams von Fanout nach Firehose.
AWS Lambda	Ereignisse an Funktionen liefern zum Auslösen der Ausführung benutzerdefinierter Geschäftslogik. Weitere Informationen finden Sie unter Fanout Amazon SNS SNS-Benachrichtigungen an Lambda-Funktionen zur automatisierten Verarbeitung.
Amazon SQS	Bereitstellen von Ereignissen in Queues für Anwendungsintegrationszwecke. Weitere Informationen finden Sie unter Fanout von Amazon SNS SNS-Benachrichtigungen an Amazon SQS SQS-Warteschlangen zur asynchronen Verarbeitung.
AWS Pipelines für Ereignisformulare	Bereitstellen von Ereignissen für Ereignissicherung und -speicherung, Ereignissuche und -analysen oder Ereigniswiedergabepipelines. Weitere Informationen finden Sie unter Fanout von Amazon SNS SNS-Ereignissen in AWS Event Fork-Pipelines.

Eventziel	Vorteile der Verwendung mit Amazon SNS
HTTP/S	Liefern von Ereignissen an externe Webhooks. Weitere Informationen finden Sie unter Fanout Amazon SNS SNS-Benachrichtigungen an HTTPS-Endpunkte.

A2P-Ziele

In der folgenden Tabelle wird beschrieben, wie Amazon SNS application-to-person (A2P) Benachrichtigungen an verschiedene Ziele übermittelt, darunter Mobiltelefone per SMS und native Push-Benachrichtigungen, E-Mail-Posteingänge, Amazon Chime Chime-Chatrooms, Slack-Kanäle und betriebliche Einblicke an Bereitschaftsteams über. PagerDuty

Diese Integrationen verbessern die Kommunikation und die betriebliche Effizienz, indem sie Benachrichtigungen und Updates in Echtzeit auf mehreren Plattformen und Kommunikationskanälen ermöglichen.

Eventziel	Vorteile der Verwendung mit Amazon SNS
SMS	Bereitstellen von Ereignissen an Mobiltelefone als Textnachrichten. Weitere Informationen finden Sie unter Mobiles Textnachrichten mit Amazon SNS.
Email	Übermitteln Sie Ereignisse als E-Mail-Na chrichten an Posteingänge. Weitere Informati onen finden Sie unter Einrichtung und Verwaltung von Amazon SNS SNS-E-Mail-Abonnements.
Plattformendpunkt	Bereitstellen von Ereignissen auf Mobiltele fonen als native Push-Benachrichtigungen. Weitere Informationen finden Sie unter Senden von mobilen Push-Benachrichtigungen mit Amazon SNS.

Eventziel	Vorteile der Verwendung mit Amazon SNS
Amazon Q Developer in Chat-Anwendungen	Liefern Sie Ereignisse an Amazon Chime Chatrooms oder Slack-Kanäle. Weitere Informationen finden Sie auf den folgenden Seiten im Administratorhandbuch für Amazon Q Developer in Chat-Anwendungen: • Amazon Q Developer in Chat-Anwendungen mit Amazon Chime einrichten • Amazon Q Developer in Chat-Anwendungen mit Slack einrichten • Verwendung von Amazon Q Developer in Chat-Anwendungen mit anderen AWS Diensten
PagerDuty	Bereitstellen von operativen Einblicken für Bereitschaftsteams. Weitere Informationen finden Sie im AWS Management & Governance-Blog unter Bereitstellen von ML-gestützten betrieblichen Erkenntnissen für Ihre Bereitschaftsteams über PagerDuty Amazon DevOps Guru.

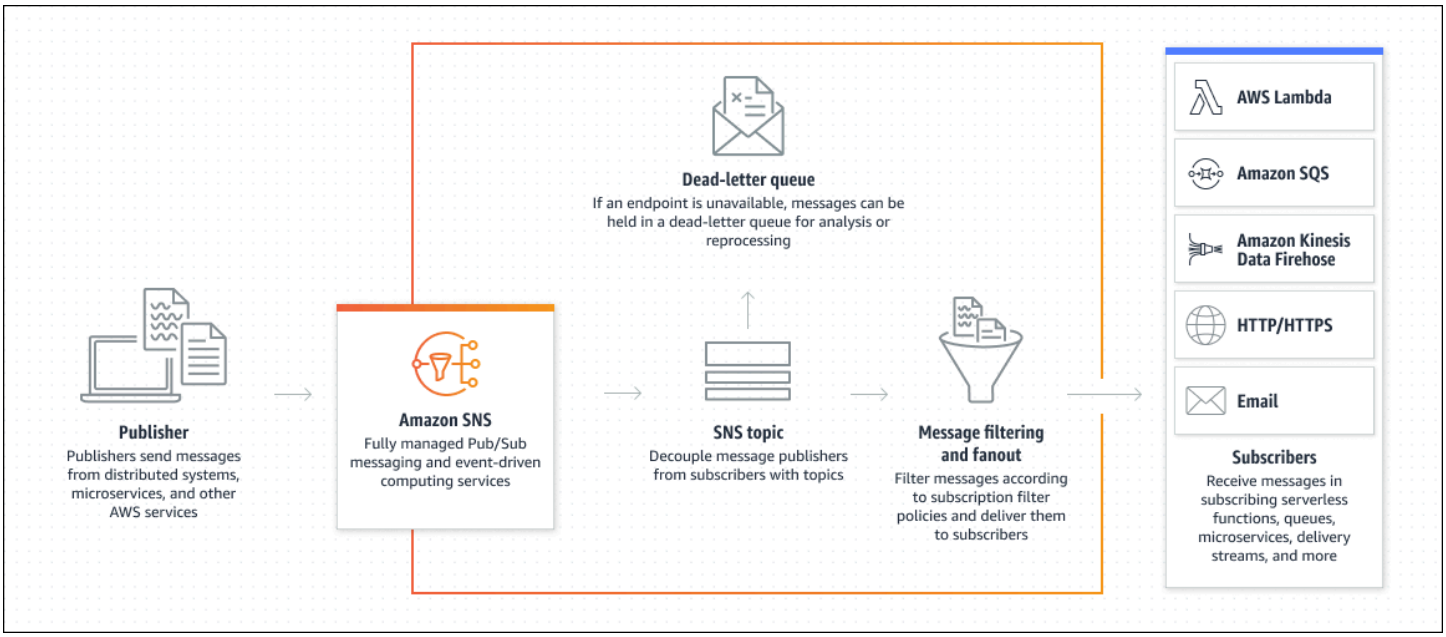
> **Note**
>
> Sie können sowohl systemeigene AWS Ereignisse als auch benutzerdefinierte Ereignisse für Chat-Apps bereitstellen:
>
> - Systemeigene AWS Ereignisse — Sie können Amazon Q Developer in Chat-Anwendungen verwenden, um native AWS Ereignisse über Amazon SNS SNS-Themen an Amazon Chime und Slack zu senden. Zu den unterstützten nativen AWS Veranstaltungen gehören Veranstaltungen von AWS Fakturierung und Kostenmanagement AWS Health, AWS CloudFormation, CloudWatch, Amazon und mehr. Weitere Informationen finden Sie unter

Verwenden von Amazon Q Developer in Chat-Anwendungen mit anderen Diensten im Administratorhandbuch für Amazon Q Developer in Chat-Anwendungen.

- Benutzerdefinierte Ereignisse— Sie können Ihre benutzerdefinierten Ereignisse auch über Amazon SNS-Themen an Amazon Chime, Slack und Microsoft Teams senden. Dazu veröffentlichen Sie benutzerdefinierte Ereignisse in einem SNS-Thema, das die Ereignisse an eine abonnierte Lambda-Funktion übermittelt. Die Lambda-Funktion verwendet dann den Webhook der Chat-App, um die Ereignisse an Empfänger zu übermitteln. Weitere Informationen finden Sie unterWie verwende ich Webhooks, um Amazon SNS Nachrichten in Amazon Chime, Slack oder Microsoft Teams zu veröffentlichen?

Amazon SNS für application-to-application den Nachrichtenversand verwenden

Amazon SNS vereinfacht die Nachrichtenübermittlung application-to-application (A2A), indem Herausgeber von Abonnenten getrennt werden. Dadurch werden Microservices, verteilte Systeme und serverlose Anwendungen unterstützt. Nachrichten werden an Amazon SNS SNS-Themen gesendet, wo sie gefiltert und an Abonnenten wie Lambda-, Amazon SQS- oder HTTP-Endpunkte zugestellt werden können. Schlägt die Zustellung fehl, werden die Nachrichten zur weiteren Analyse oder erneuten Verarbeitung in einer Warteschlange für unzustellbare Briefe gespeichert.



Lieferstreams von Fanout nach Firehose

Sie können Lieferstreams für Amazon SNS SNS-Themen abonnieren, sodass Sie Benachrichtigungen an zusätzliche Speicher- und Analyseendpunkte senden können. Nachrichten, die unter einem Amazon SNS SNS-Thema veröffentlicht wurden, werden an den abonnierten Firehose-Lieferstream gesendet und an das in Firehose konfigurierte Ziel zugestellt. Ein Abonnementinhaber kann bis zu fünf Firehose-Lieferstreams für ein Amazon SNS SNS-Thema abonnieren. Jeder Firehose-Lieferstream hat ein Standardkontingent für Anfragen und Durchsatz pro Sekunde. Dieses Limit könnte dazu führen, dass mehr Nachrichten veröffentlicht werden (eingehender Datenverkehr) als zugestellt (ausgehender Datenverkehr). Wenn mehr eingehender Datenverkehr als ausgehender Datenverkehr vorhanden ist, kann Ihr Abonnement einen großen Nachrichtenrückstand ansammeln, was zu einer hohen Nachrichtenzustellungslatenz führt. Sie

können eine Erhöhung des Kontingents auf der Grundlage der Veröffentlichungsrate anfordern, um negative Auswirkungen auf Ihren Workload zu vermeiden.

Über Firehose-Lieferstreams können Sie Amazon SNS SNS-Benachrichtigungen an Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service (OpenSearch Service) und an Drittanbieter wie Datadog, New Relic, MongoDB und Splunk weiterleiten.

Sie können diese Funktion beispielsweise verwenden, um Nachrichten, die an ein Thema gesendet wurden, dauerhaft in einem Amazon S3 Bucket zu Compliance, Archivierung oder anderen Zwecken zu speichern. Erstellen Sie dazu einen Firehose-Lieferstream mit einem Amazon S3 S3-Bucket-Ziel und abonnieren Sie diesen Lieferstream für das Amazon SNS SNS-Thema. Um beispielsweise Nachrichten zu analysieren, die an ein Amazon SNS SNS-Thema gesendet wurden, erstellen Sie einen Zustellungsstream mit einem OpenSearch Service-Indexziel. Anschließend können Sie den Firehose-Lieferstream für das Amazon SNS SNS-Thema abonnieren.

Amazon SNS unterstützt auch die Protokollierung des Nachrichtenzustellungsstatus für Benachrichtigungen, die an Firehose-Endpunkte gesendet werden. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.

Voraussetzungen für das Abonnieren von Firehose-Lieferstreams für Amazon SNS SNS-Themen

Um einen Delivery-Stream für ein SNS-Thema zu abonnieren, AWS-Konto müssen Sie über Folgendes verfügen:

- Ein Standard–SNS-Thema. Weitere Informationen finden Sie unter Erstellen Sie ein Amazon SNS-Thema.
- Ein Firehose-Lieferstream. Weitere Informationen finden Sie unter Einen Delivery Stream erstellen und Ihrer Anwendung Zugriff auf Ihre Firehose-Ressourcen gewähren im Amazon Data Firehose Developer Guide.
- Eine AWS Identity and Access Management (IAM-) Rolle, die dem Amazon SNS SNS-Service Principal vertraut und über die Berechtigung verfügt, in den Lieferstream zu schreiben. Geben Sie als den Amazon-Ressourcennamen (ARN) dieser Rolle als`SubscriptionRoleARN`Wenn Sie das Abonnement erstellen. Amazon SNS übernimmt diese Rolle, sodass Amazon SNS Datensätze in den Firehose-Lieferstream aufnehmen kann.

  Hier ein Beispiel für eine Berechtigungsrichtlinie.

JSON

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

Um vollständige Berechtigungen für die Verwendung von Firehose zu erteilen, können Sie auch die AWS verwaltete Richtlinie `AmazonKinesisFirehoseFullAccess` verwenden. Um strengere Berechtigungen für die Verwendung von Firehose zu gewähren, können Sie auch Ihre eigene Richtlinie erstellen. Die Richtlinie muss mindestens die Berechtigung zum Ausführen des `PutRecord`-Vorgangs auf einem bestimmten Bereitstellungsdatenstrom haben.

In allen Fällen müssen Sie auch die Vertrauensstellung bearbeiten, um den Amazon SNS-Serviceprinzipal einzubeziehen. Zum Beispiel:

JSON

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

```
        "Action": "sts:AssumeRole"
      }
    ]
  }
```

Weitere Informationen zum Erstellen von Rollen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS Dienst im IAM-Benutzerhandbuch.

Nachdem Sie diese Anforderungen erfüllt haben, können Sie Abonnieren des Bereitstellungsdatenströme für das SNS-Thema auswählen.

Einen Firehose-Lieferstream für ein Amazon SNS SNS-Thema abonnieren

Um Amazon SNS SNS-Benachrichtigungen an Lieferdatenströme zu senden, stellen Sie zunächst sicher, dass Sie alle Voraussetzungen erfüllt haben. Eine Liste der unterstützten Endgeräte finden Sie unter Endpunkte und Kontingente in der. Allgemeine Amazon Web Services-Referenz

Um einen Firehose-Lieferstream für ein Thema zu abonnieren

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions aus.
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:
   a. Für Thema ARN wählen Sie den Amazon-Ressourcennamen (ARN) eines Standard-Themas aus.
   b. Wählen Sie für Protocol Firehose aus.
   c. Wählen Sie für Endpoint den ARN eines Firehose-Lieferdatenstroms aus, der Benachrichtigungen von Amazon SNS empfangen kann.
   d. Geben Sie für die Abonnementrolle ARN den ARN der Rolle AWS Identity and Access Management (IAM) an, die Sie für das Schreiben in Firehose-Lieferdatenströme erstellt haben. Weitere Informationen finden Sie unter Voraussetzungen für das Abonnieren von Firehose-Lieferstreams für Amazon SNS SNS-Themen.

   e. (Optional) Um Amazon SNS-Metadaten aus veröffentlichten Nachrichten zu entfernen, wählen Sie Aktivieren der Übermittlung unformatierter Nachrichten aus. Weitere Informationen finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

5. (Optional) Um eine Filterrichtlinie zu konfigurieren, erweitern Sie den Abschnitt Abonnement-Filterrichtlinie. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.
6. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für das Abonnement zu konfigurieren, erweitern Sie den Abschnitt Redrive-Richtlinie (Warteschlange für unzustellbare Nachrichten). Weitere Informationen finden Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe.
7. Wählen Sie Create subscription (Abonnement erstellen) aus.

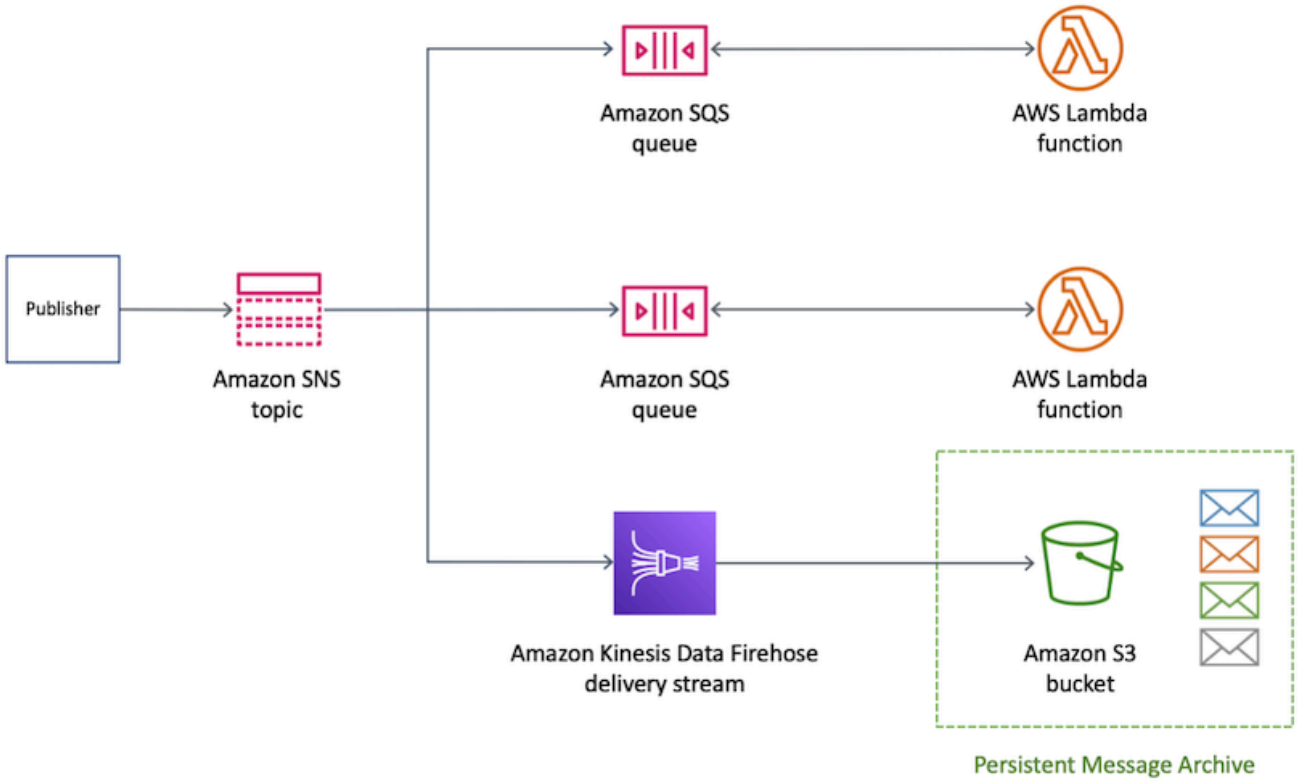
Die Konsole erstellt das Abonnement und öffnet die Seite Details des Abonnements.

Verwaltung von Amazon SNS SNS-Nachrichten an mehreren Lieferstream-Zielen

Mit Lieferdatenströmen können Sie Amazon SNS SNS-Nachrichten an mehreren Zielen verwalten und so die Integration mit Amazon S3, Amazon OpenSearch Service, Amazon Redshift und HTTP-Endpunkten für Speicherung, Indizierung und Analyse ermöglichen. Durch die richtige Konfiguration der Nachrichtenformatierung und -zustellung können Sie Amazon SNS SNS-Benachrichtigungen zur späteren Verarbeitung in Amazon S3 speichern, strukturierte Nachrichtendaten mit Amazon Athena analysieren, Nachrichten OpenSearch für die Suche und Visualisierung in Echtzeit indexieren und Archive in Amazon Redshift für erweiterte Abfragen strukturieren.

Speichern und Analysieren von Amazon SNS SNS-Nachrichten in Amazon S3 S3-Zielen

In diesem Thema wird erklärt, wie Delivery Streams Daten in Amazon Simple Storage Service (Amazon S3) veröffentlichen.



Themen

- Formatieren von Amazon SNS SNS-Benachrichtigungen für die Speicherung in Amazon S3 S3-Zielen
- Analysieren von in Amazon S3 gespeicherten Amazon SNS-Nachrichten mit Athena

Formatieren von Amazon SNS SNS-Benachrichtigungen für die Speicherung in Amazon S3 S3-Zielen

Das folgende Beispiel zeigt eine Amazon SNS SNS-Benachrichtigung, die an einen Amazon Simple Storage Service (Amazon S3) -Bucket gesendet wurde, mit Einrückungen zur besseren Lesbarkeit.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Wenn die Zustellung von unformatierten Nachrichten deaktiviert ist, fügt Amazon SNS der Nachricht JSON-Metadaten hinzu, einschließlich der folgenden Eigenschaften:
>
> - `Type`
> - `MessageId`

- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten 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"
        }
    }
 }
```

Das folgende Beispiel zeigt drei SNS-Nachrichten, die über einen Lieferstream an denselben Amazon S3 S3-Bucket gesendet wurden. Die Pufferung wird angewendet, und jede Nachricht wird durch Zeilenumbrüche getrennt.

```
{"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"}
```

Analysieren von in Amazon S3 gespeicherten Amazon SNS-Nachrichten mit Athena

Auf dieser Seite wird erklärt, wie Amazon SNS SNS-Nachrichten analysiert werden, die über Lieferstreams an Amazon Simple Storage Service (Amazon S3) -Ziele gesendet werden.

Um SNS-Nachrichten zu analysieren, die über Firehose-Lieferstreams an Amazon S3 S3-Ziele gesendet wurden

1. Konfigurieren Sie Ihre Amazon-S3-Ressourcen. Anweisungen finden Sie unter Erstellen eines Buckets im Benutzerhandbuch zu Amazon Simple Storage Service und Arbeiten mit Amazon-S3-Buckets im Benutzerhandbuch zu Amazon Simple Storage Service.
2. Konfigurieren Sie Ihren Bereitstellungsdatenstrom. Anweisungen finden Sie unter Wählen Sie Amazon S3 für Ihr Ziel im Amazon Data Firehose Developer Guide.
3. Verwenden von Amazon Athena zum Abfragen der Amazon-S3-Objekte unter Verwendung von Standard-SQL. Weitere Informationen finden Sie unter Erste Schritte im Benutzerhandbuch zu Amazon Athena.

Beispielabfrage

Nehmen Sie für dieses Beispiel Folgendes an:

- Nachrichten werden in der `notifications`Tabelle im `default` Schema gespeichert.

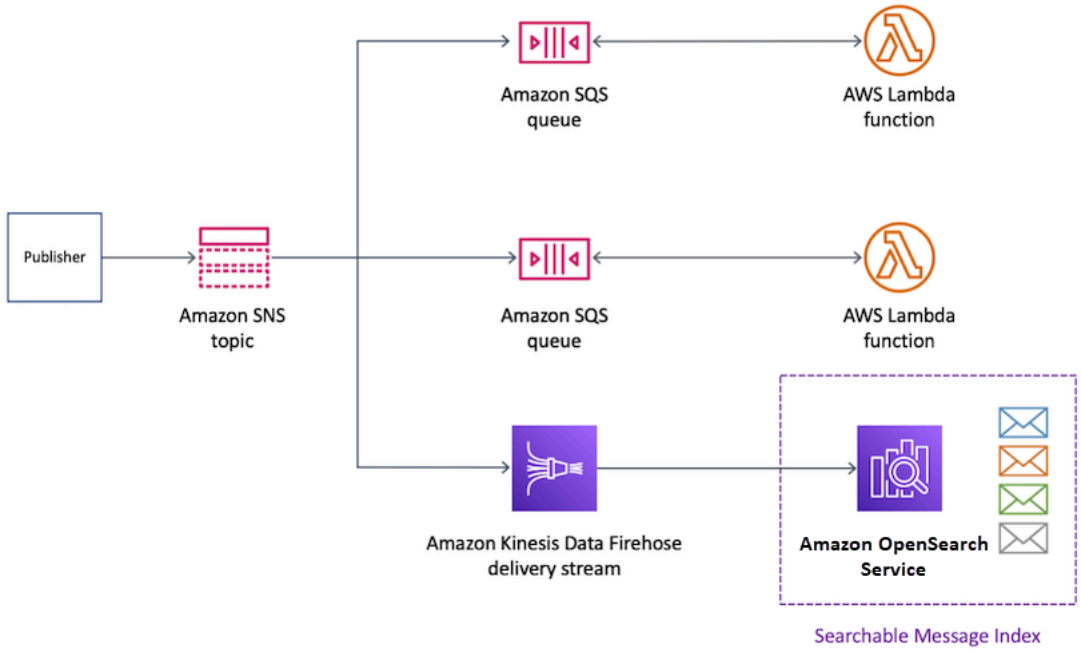
- Die `notifications` Tabelle enthält eine `timestamp` Spalte mit dem Typ `string`.

Die folgende Abfrage gibt alle SNS-Nachrichten zurück, die im angegebenen Datumsbereich empfangen wurden:

```
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';
```

Integration von Amazon SNS SNS-Nachrichten mit Amazon OpenSearch Service-Zielen

In diesem Abschnitt wird erklärt, wie Delivery Streams Daten an Amazon OpenSearch Service (OpenSearch Service) veröffentlichen.



Themen

- Speichern und Formatieren von Amazon SNS SNS-Benachrichtigungen in OpenSearch Service-Indizes
- Analysieren von Amazon SNS SNS-Nachrichten für OpenSearch Serviceziele

Speichern und Formatieren von Amazon SNS SNS-Benachrichtigungen in OpenSearch Service-Indizes

Das folgende Beispiel zeigt eine Amazon SNS SNS-Benachrichtigung, die an einen Amazon OpenSearch Service (OpenSearch Service) -Index mit dem Namen `my-index` gesendet wird. Dieser Index verfügt über ein Zeitfilterfeld auf der Registerkarte `Timestamp`. Die SNS-Benachrichtigung wird in der `_source`-Eigenschaft der Nutzlast angezeigt.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Wenn die Zustellung von unformatierten Nachrichten deaktiviert ist, fügt Amazon SNS der Nachricht JSON-Metadaten hinzu, einschließlich der folgenden Eigenschaften:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "49613100963111323203250405402193283794773886550985932802.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
```

```
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

Analysieren von Amazon SNS SNS-Nachrichten für OpenSearch Serviceziele

In diesem Thema wird erklärt, wie Amazon SNS SNS-Nachrichten analysiert werden, die über Lieferströme an Amazon OpenSearch Service (OpenSearch Service) -Ziele gesendet werden.

Um SNS-Nachrichten zu analysieren, die über Firehose-Zustellungsstreams an Serviceziele gesendet werden OpenSearch

1. Konfigurieren Sie Ihre OpenSearch Serviceressourcen. Eine Anleitung finden Sie unter Erste Schritte mit Amazon OpenSearch Service im Amazon OpenSearch Service Developer Guide.
2. Konfigurieren Sie Ihren Bereitstellungsdatenstrom. Anweisungen finden Sie unter Wählen Sie den OpenSearch Service für Ihr Ziel im Amazon Data Firehose Developer Guide.
3. Führen Sie eine Abfrage mithilfe von OpenSearch Service Queries und Kibana aus. Weitere Informationen finden Sie unter Schritt 3: Dokumente in einer OpenSearch Service-Domain und Kibana durchsuchen im Amazon OpenSearch Service Developer Guide.

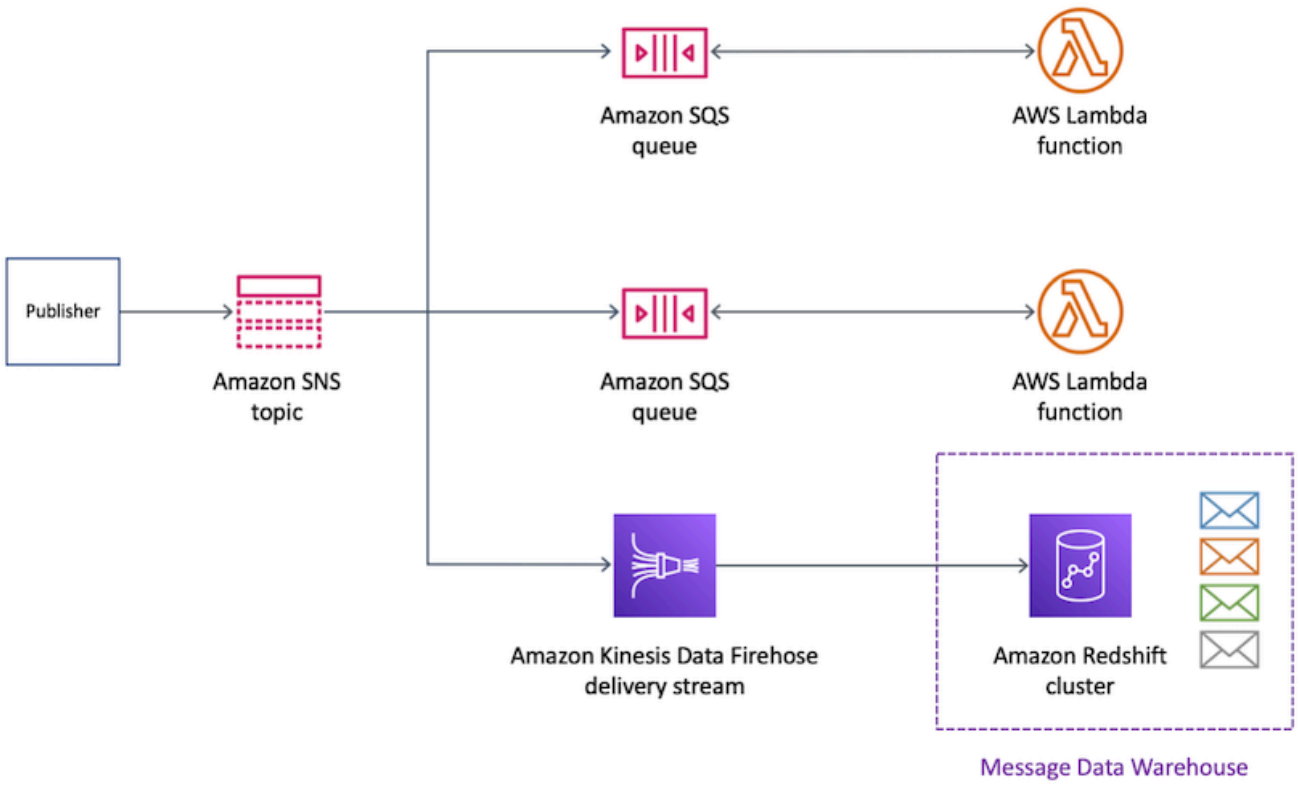
Beispielabfrage

Im folgenden Beispiel werden die `my-index` Indizes für alle SNS-Nachrichten abgeragt, die im angegebenen Datumsbereich empfangen werden:

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

Konfiguration der Zustellung und Analyse von Amazon SNS SNS-Nachrichten in Amazon Redshift Redshift-Zielen

In diesem Thema wird erklärt, wie Sie Amazon SNS SNS-Benachrichtigungen an einen Lieferstream weiterleiten, der dann Daten in Amazon Redshift veröffentlicht. Mit diesem Setup können Sie eine Verbindung zur Amazon Redshift Redshift-Datenbank herstellen und ein SQL-Abfragetool verwenden, um Amazon SNS SNS-Nachrichten abzurufen, die bestimmten Kriterien entsprechen.



Themen

- Strukturierung von Amazon SNS SNS-Nachrichtenarchiven in Amazon Redshift Redshift-Tabellen
- Analysieren von Amazon SNS SNS-Nachrichten, die in Amazon Redshift Redshift-Zielen gespeichert sind

Strukturierung von Amazon SNS SNS-Nachrichtenarchiven in Amazon Redshift Redshift-Tabellen

Für Amazon Redshift Redshift-Endpunkte werden Amazon SNS SNS-Nachrichten als Zeilen in einer Tabelle archiviert. Hier ist ein Beispiel dafür, wie die Daten gespeichert werden:

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Wenn die Zustellung von unformatierten Nachrichten deaktiviert ist, fügt Amazon SNS der Nachricht JSON-Metadaten hinzu, einschließlich der folgenden Eigenschaften:
>
> - `Type`
> - `MessageId`

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

Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.
Obwohl Amazon SNS der Nachricht mithilfe der in dieser Liste angezeigten Groß-/Kleinschreibung Eigenschaften hinzufügt, werden Spaltennamen in Amazon-Redshift-Tabellen in Kleinbuchstaben angezeigt. Um die JSON-Metadaten für den Amazon-Redshift-Endpunkt zu transformieren, können Sie den SQL `COPY`-Befehl nutzen. Weitere Informationen finden Sie unter Beispiele für die COPY-Operation aus JSON und Laden von JSON-Daten unter Verwendung der Option „auto ignorecase“ im Entwicklerhandbuch für Amazon Redshift Datenbanken.

Typ	messageid	topicarn	subject	Nachricht	Zeitstempel	unsubscribeurl	messageattributes
Benachrichtigung	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Beispiel-Betreff	Beispiel-Nachricht	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Abmelden&=arn:aws:sns:us-east-1:111111111111:my-to	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Typ	messageid	topicarn	subject	Nachricht	Zeitstempel	unsubscribeurl	messageattributes
						pic:326deeeb-cbf4-45da-b92b-ca77a247813b SubscriptionArn	
Benachrichtigung	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Beispiel-Betreff 2	Beispielnachricht 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Abmelden&=arn:aws:sns:us-east-1:111111111111:my-to pic:326deeeb-cbf4-45da-b92b-ca77a247813b SubscriptionArn	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Typ	messageid	topicarn	subject	Nachricht	Zeitstempel	unsubscribeurl	messageattributes
Benachrichtigung	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Beispiel-Betreff 3	Beispielnachricht 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Abmelden&=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b SubscriptionArn	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Weitere Informationen zum Verteilen von Benachrichtigungen an Amazon-Redshift-Endpunkte finden Sie unter Konfiguration der Zustellung und Analyse von Amazon SNS SNS-Nachrichten in Amazon Redshift Redshift-Zielen.

Analysieren von Amazon SNS SNS-Nachrichten, die in Amazon Redshift Redshift-Zielen gespeichert sind

In diesem Thema wird beschrieben, wie Amazon SNS SNS-Nachrichten analysiert werden, die über Lieferstreams an Amazon Redshift Redshift-Ziele gesendet werden.

Um SNS-Nachrichten zu analysieren, die über Firehose-Lieferstreams an Amazon Redshift Redshift-Ziele gesendet wurden

1. Konfigurieren Sie Ihre Amazon-Redshift-Ressourcen. Detaillierte Anweisungen finden Sie unter Erste Schritte mit Amazon Redshift im Handbuch für erste Schritte mit Amazon Redshift.
2. Konfigurieren Sie Ihren Bereitstellungsdatenstrom. Anweisungen finden Sie unter Wählen Sie Amazon Redshift für Ihr Ziel im Amazon Data Firehose Developer Guide.
3. Ausführen einer Abfrage Weitere Informationen finden Sie unter Abfragen einer Datenbank mit dem Abfrage-Editor im Amazon-Redshift-Verwaltungshandbuch.

Beispielabfrage

Nehmen Sie für dieses Beispiel Folgendes an:

- Nachrichten werden in der `notifications`-Tabelle im Standardschema `public` gespeichert.
- Die `Timestamp`-Eigenschaft aus der SNS-Nachricht ist in der Spalte der Tabelle `timestamp` mit dem Spaltendatentyp `timestamptz`gespeichert.

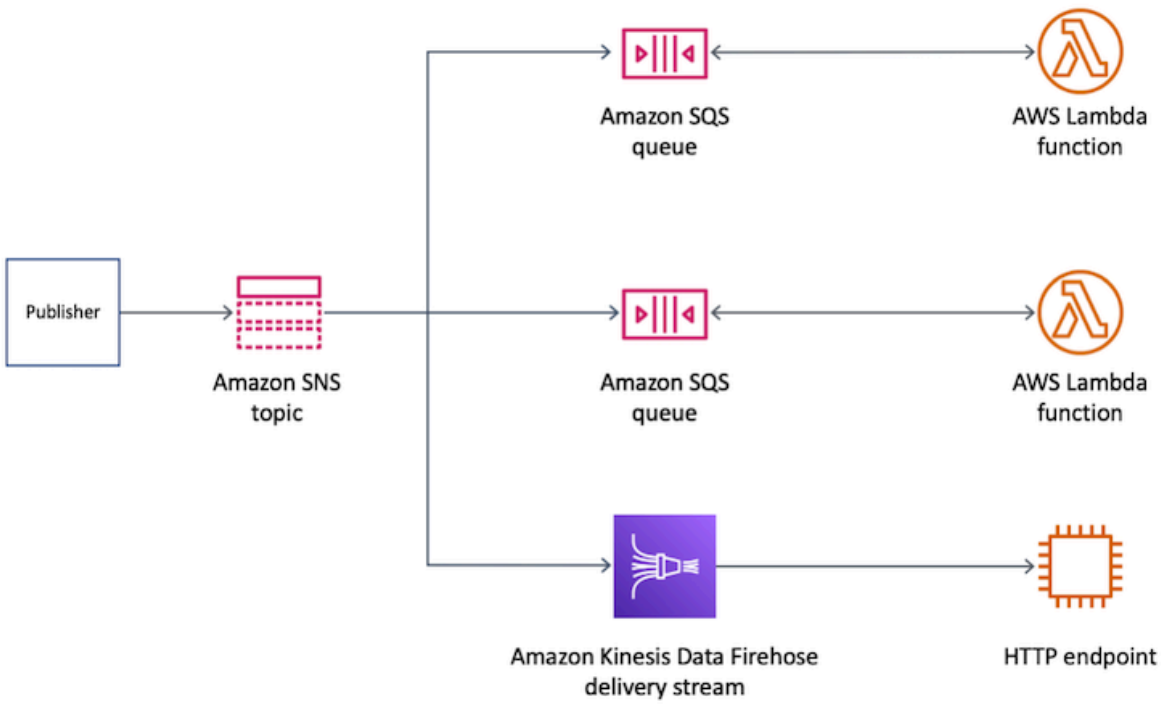
> **Note**
>
> Um die JSON-Metadaten für den Amazon-Redshift-Endpunkt zu transformieren, können Sie den SQL `COPY`-Befehl nutzen. Weitere Informationen finden Sie unter Beispiele für die COPY-Operation aus JSON und Laden von JSON-Daten unter Verwendung der Option „auto ignorecase“ im Entwicklerhandbuch für Amazon Redshift Datenbanken.

Die folgende Abfrage gibt alle SNS-Nachrichten zurück, die im angegebenen Datumsbereich empfangen wurden:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

Konfiguration der Amazon SNS SNS-Nachrichtenzustellung an HTTP-Ziele mithilfe von

In diesem Thema wird erklärt, wie Delivery-Streams Daten auf HTTP-Endpunkten veröffentlichen.



Themen

- Amazon SNS SNS-Benachrichtigungsformat für die Lieferung an HTTP-Ziele

Amazon SNS SNS-Benachrichtigungsformat für die Lieferung an HTTP-Ziele

Hier ist ein Beispiel für einen HTTP-POST-Anforderungstext von Amazon SNS, der über einen Lieferstream an einen HTTP-Endpunkt gesendet wurde. Die Amazon SNS SNS-Benachrichtigung ist als Base64-Nutzlast in der Eigenschaft records codiert.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

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

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

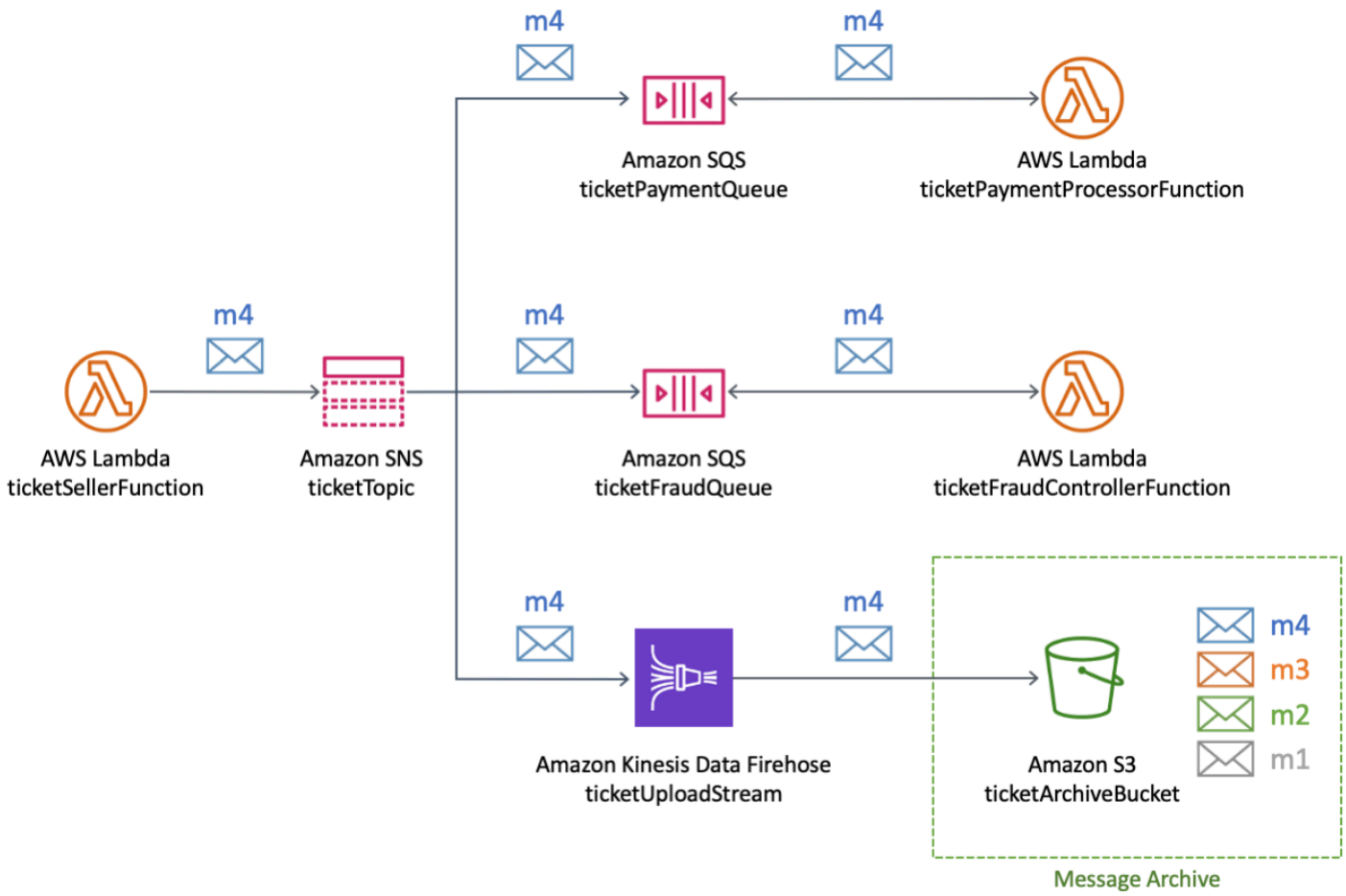

Archivierung und Analyse von Amazon SNS SNS-Nachrichten: Ein Anwendungsbeispiel für Ticketing-Plattformen

Dieses Thema bietet ein Tutorial für einen häufigen Anwendungsfall der Archivierung und Analyse von Amazon SNS SNS-Nachrichten.

Die Einstellung dieses Anwendungsfalles ist eine Flugticket-Plattform, die in einem regulierten Umfeld betrieben wird.

1. Die Plattform unterliegt einem Compliance-Rahmen, der das Unternehmen verpflichtet, alle Ticketverkäufe mindestens fünf Jahre lang zu archivieren.
2. Um das Compliance-Ziel in Bezug auf die Datenspeicherung zu erreichen, abonniert das Unternehmen einen Lieferstream zu einem bestehenden Amazon SNS SNS-Thema.
3. Das Ziel für den Bereitstellungsdatenstrom ist ein Amazon Simple Storage Service (Amazon S3) Bucket. Mit dieser Konfiguration werden alle im SNS-Thema veröffentlichten Ereignisse im Amazon S3 Bucket archiviert.

Das folgende Diagramm illustriert die Architektur dieser Konfiguration.



Um Analysen durchzuführen und Einblicke in den Ticketverkauf zu erhalten, führt das Unternehmen SQL-Abfragen mithilfe von Amazon Athena aus. Zum Beispiel kann das Unternehmen abfragen, um mehr über die beliebtesten Reiseziele und die häufigsten Flyer zu erfahren.

Um die AWS Ressourcen für diesen Anwendungsfall zu erstellen, können Sie die Vorlage AWS Management Console oder eine AWS CloudFormation Vorlage verwenden.

Themen

- Einrichtung erster AWS Ressourcen für die Archivierung und Analyse von Amazon SNS SNS-Nachrichten
- Einrichtung eines Amazon Data Firehose-Lieferdatenstroms für die Amazon SNS SNS-Nachrichtenarchivierung
- Den Firehose-Lieferstream für das Amazon SNS SNS-Thema abonnieren
- Testen und Abfragen einer Amazon SNS SNS-Konfiguration für effektives Datenmanagement

- Automatisieren der Amazon SNS SNS-Nachrichtenarchivierung mit einer Vorlage AWS CloudFormation

Einrichtung erster AWS Ressourcen für die Archivierung und Analyse von Amazon SNS SNS-Nachrichten

In diesem Thema wird beschrieben, wie Sie die Ressourcen erstellen, die für den Beispielanwendungsfall Nachrichtenarchivierung und -analyse benötigt werden:

- Ein Amazon Simple Storage Service (Amazon S3)-Bucket
- Zwei Amazon-Simple-Queue-Service-(Amazon-SQS)-Warteschlangen
- Amazon-SNS-Thema
- Ein Amazon-SNS-Thema mit zwei Abonnements zu Amazon-SQS-Themen

So erstellen Sie die anfänglichen Ressourcen

1. Erstellung eines Amazon S3 Buckets

   a. Öffnen Sie die Amazon S3-Konsole.
   b. Wählen Sie Create Bucket (Bucket erstellen) aus.
   c. Geben Sie für Bucket name einen eindeutigen Namen ein. Behalten Sie die anderen Felder als Standardwerte bei.
   d. Wählen Sie Create Bucket (Bucket erstellen) aus.

   Weitere Informationen über Amazon S3 Buckets finden Sie unter Erstellen eines Buckets im Handbuch für den Amazon Simple Storage Service und Arbeiten mit Amazon-S3-Buckets im Handbuch für den Amazon Simple Storage Service.

2. Erstellen Sie die beiden Amazon-SQS-Warteschlangen:

   a. Öffnen Sie die Amazon-SQS-Konsole.
   b. Wählen Sie Create queue (Warteschlange erstellen) aus.
   c. Wählen Sie unter Type (Typ) die Option Standard aus.
   d. Geben Sie unter Name **`ticketPaymentQueue`** ein.
   e. Unter Zugriffsrichtlinie, für Wählen Sie die Methode, wählen Sie Advanced aus.
   f. Fügen Sie unter Policy Document für JSON Folgendes ein:

JSON

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

Ersetzen Sie in dieser Zugriffsrichtlinie die AWS-Konto Zahl (*123456789012*) durch Ihre eigene und ändern Sie die AWS Region (*us-east-1*) entsprechend.

g. Wählen Sie Create queue (Warteschlange erstellen) aus.

h. Wiederholen Sie diese Schritte, um eine zweite SQS-Warteschlange namens **ticketFraudQueue** zu erstellen.

Weitere Informationen zum Erstellen von SQS-Warteschlangen finden Sie unter Erstellen einer Amazon-SQS-Warteschlange (Konsole) im Entwicklungshandbuch für Amazon-Simple-Warteschlangen.

3. Erstellen eines SNS-Themas

a. Öffnen Sie die Seite Themen der Amazon-SNS-Konsole.

b. Wählen Sie Create topic aus.

c. UnterDetails, für Typ, wählen Sie Standard aus.

d. Geben Sie unter Name **`ticketTopic`** ein.

e. Wählen Sie Thema erstellen aus.

Weitere Informationen zum Erstellen von SNS-Themen finden Sie in Erstellen Sie ein Amazon SNS-Thema.

4. Abonnieren Sie die SQS-Warteschlangen in das SNS-Thema.

   a. In der Amazon-SNS-Konsole, wählen Sie auf der Detailseite zu TicketTopic das ThemaErstellen eines Abonnements aus.

   b. Unter Details, fürProtocol (Protokoll), wählen Sie Amazon SQS aus.

   c. Wählen Sie für Endpoint den Amazon-Ressourcennamen (ARN) der ticketPaymentQueueWarteschlange.

   d. Wählen Sie Create subscription (Abonnement erstellen) aus.

   e. Wiederholen Sie diese Schritte, um ein zweites Abonnement mit dem ARN der ticketFraudQueueWarteschlange zu erstellen.

   Weitere Informationen zum Abonnieren von SNS-Themen finden Sie unter Ein Abonnement für ein Amazon SNS SNS-Thema erstellen. Sie können SQS-Warteschlangen auch von der Amazon-SQS-Konsole aus in SNS-Themen abonnieren. Weitere Informationen finden Sie unter Tutorial: Abonnieren einer Amazon-SQS-Warteschlange zu einem Amazon-SNS-Thema (Konsole) im Amazon-Simple-Queue-Service-Entwicklerhandbuch.

Sie haben die anfänglichen Ressourcen für diesen Anwendungsfall erstellt. Geben Sie Einrichtung eines Amazon Data Firehose-Lieferdatenstroms für die Amazon SNS SNS-Nachrichtenarchivierung ein, um fortzufahren.

Einrichtung eines Amazon Data Firehose-Lieferdatenstroms für die Amazon SNS SNS-Nachrichtenarchivierung

In diesem Thema wird erklärt, wie Sie den Amazon Data Firehose-Lieferstream für den Beispielanwendungsfall Nachrichtenarchivierung und -analyse erstellen.

So erstellen Sie den Amazon Data Firehose-Lieferstream

1. Öffnen Sie die Amazon Data Firehose Services-Konsole.
2. Wählen Sie Firehose und dann Create delivery stream aus.

3. Klicken Sie auf der Seite Neuer Bereitstellungsdatenstrom fürName eines Bereitstellungsdatenstroms den Wert **`ticketUploadStream`** und klicken Sie danach auf Weiter.
4. Klicken Sie auf der Seite Process records (Datensätze verarbeiten) auf Weiter.
5. Führen Sie auf der Seite Choose a Destination (Ziel auswählen) folgende Schritte durch:
   a. Für Ziel, wählen Sie Amazon S3 aus.
   b. Unter S3-Ziel, für S3 bucket wählen Sie den S3-Bucket aus, den Sie anfänglich erstellthaben.
   c. Wählen Sie Weiter aus.
6. Klicken Sie auf der Seite Konfigurieren der Einstellungen für S3-Pufferbedingungen wie folgt:
   - Für Puffer-Größe geben Sie **`1`** ein.
   - Für Puffer-Interval geben Sie **`60`** ein.

   Wenn Sie diese Werte für den Amazon S3-Puffer verwenden, können Sie die Konfiguration schnell testen. Der erste Pufferbedingung, die erfüllt wird, bewirkt, dass dem S3-Bucket die Daten bereitgestellt werden.
7. Wählen Sie auf der Seite „Einstellungen konfigurieren“ für Berechtigungen aus, ob Sie eine AWS Identity and Access Management (IAM-) Rolle erstellen möchten, der die erforderlichen Berechtigungen automatisch zugewiesen werden. Wählen Sie anschließend Weiter.
8. Klicken Sie auf der Seite Prüfen, wählen Sie Erstellen von Bereitstellungsdatenstrom aus.
9. Wählen Sie auf der Seite Amazon Data Firehose Delivery Streams den Delivery Stream aus, den Sie gerade erstellt haben (ticketUploadStream). Notieren Sie auf der Registerkarte Details den Amazon-Ressourcennamen (ARN) des Datenstroms für später.

Weitere Informationen zum Erstellen von Delivery Streams finden Sie unter Creating an Amazon Data Firehose Delivery Stream im Amazon Data Firehose Developer Guide. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen für einen AWS Service im IAM-Benutzerhandbuch.

Sie haben den Firehose-Lieferstream mit den erforderlichen Berechtigungen erstellt. Geben Sie Den Firehose-Lieferstream für das Amazon SNS SNS-Thema abonnieren ein, um fortzufahren.

Den Firehose-Lieferstream für das Amazon SNS SNS-Thema abonnieren

In diesem Thema wird erklärt, wie Sie die folgenden Ressourcen für den Beispielanwendungsfall Nachrichtenarchivierung und -analyse erstellen:

- Die AWS Identity and Access Management (IAM) -Rolle, die es dem Amazon SNS SNS-Abonnement ermöglicht, Datensätze in den Lieferstream aufzunehmen.
- Das Firehose Delivery Stream-Abonnement zum Thema Amazon SNS.

So erstellen Sie die IAM-Rolle für das Amazon-SNS-Abonnement

1. Öffnen Sie die Seite Roles (Rollen) in der IAM-Konsole.
2. Wählen Sie Create role (Rolle erstellen) aus.
3. Wählen Sie unter Select type of trusted entity (Typ der vertrauenswürdigen Entität wählen) die Option AWS service (Service) aus.
4. Für Auswahl eines Anwendungsfalls, wählen Sie SNS aus. Wählen Sie dann Next: Permissions.
5. Wählen Sie Next: Tags (Weiter: Tags) aus.
6. Klicken Sie auf Weiter: Prüfen.
7. Geben Sie auf der Seite Review (Überprüfen) für Role name (Rollenname) **`ticketUploadStreamSubscriptionRole`** ein. Wählen Sie dann Create Role.
8. Wenn die Rolle erstellt ist, wählen Sie ihren Namen () ticketUploadStreamSubscriptionRole.
9. Wählen Sie auf der Seite Summary die Option Add inline policy aus.
10. Wählen Sie auf der Seite Create policy die Registerkarte JSON aus, kopieren Sie dann die folgende JSON-Richtlinie und fügen Sie sie in das Textfeld ein.

    JSON

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

Ersetzen Sie in dieser Richtlinie die AWS-Konto Nummer (*123456789012*) durch Ihre eigene und ändern Sie die AWS Region (*us-east-1*) entsprechend.

11. Wählen Sie Richtlinie prüfen.
12. Geben Sie auf der Seite Create policy (Richtlinie erstellen) für Name **FirehoseSnsPolicy** ein. Wählen Sie dann Richtlinie erstellen aus.
13. Auf der Seite Übersicht der Rolle, notieren Sie die ARN der Rolle für später.

Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen für einen AWS Dienst im IAM-Benutzerhandbuch.

Um den Firehose-Lieferstream für das SNS-Thema zu abonnieren

1. Öffnen Sie die Seite Themen der Amazon-SNS-Konsole.
2. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen) aus.
3. Wählen Sie unter Details für Protokoll die Option.
4. Geben Sie für Endpoint den Amazon-Ressourcennamen (ARN) des ticketUploadStreamLieferdatenstroms ein, den Sie zuvor erstellt haben. Geben Sie z. B. ei **arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream**.
5. Geben Sie für die Abonnementrolle ARN den ARN der ticketUploadStreamSubscriptionRoleIAM-Rolle ein, die Sie zuvor erstellt haben. Geben Sie z. B. ei **arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole**.
6. Wählen Sie Aktivieren der Übermittlung unformatierter Nachrichten aus.

7. Wählen Sie Create subscription (Abonnement erstellen) aus.

Sie haben die IAM-Rolle und das SNS-Themenabonnement erstellt. Geben Sie Testen und Abfragen einer Amazon SNS SNS-Konfiguration für effektives Datenmanagement ein, um fortzufahren.

Testen und Abfragen einer Amazon SNS SNS-Konfiguration für effektives Datenmanagement

In diesem Thema wird erklärt, wie Sie den Beispielanwendungsfall Nachrichtenarchivierung und -analyse testen, indem Sie eine Nachricht im Amazon SNS SNS-Thema veröffentlichen. Die Anweisungen enthalten eine Beispielabfrage, die Sie ausführen und an Ihre eigenen Bedürfnisse anpassen können.

So testen Sie die Konfiguration

1. Öffnen Sie die Seite Themen der Amazon-SNS-Konsole.
2. Wählen Sie das Symbol **`ticketTopic`** Thema.
3. Wählen Sie Publish message (Nachricht veröffentlichen) aus.
4. Geben Sie auf der Seite Nachricht im Thema veröffentlichen für den Nachrichtentext Folgendes ein. Fügen Sie am Ende der Nachricht einen Zeilenumbruch hinzu.

   ```
   {"BookingDate":"2020-12-15","BookingTime":"2020-12-15
    04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
   ```

   Behalten Sie alle anderen Optionen als Standardwerte bei.
5. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

   Weitere Informationen zum Veröffentlichen von Nachrichten finden Sie unter Veröffentlichen einer Amazon SNS SNS-Nachricht.
6. Öffnen Sie nach dem Übermittlungsstromintervall von 60 Sekunden das Dialogfeld Konsole für Amazon Simple Storage Service (Amazon S3) und wählen Sie den Amazon S3 Bucket aus, den SieInital erstellt haben.

   Die veröffentlichte Nachricht wird im Bucket angezeigt.

Abfragen der Daten

1. Öffnen Sie die Amazon-Athena-Konsole.
2. Ausführen einer Abfrage

   Angenommen, dass die `notifications`-Tabelle im `default`-Schema folgende Daten enthält:

```
{"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"}
```

   Führen Sie die folgende Abfrage aus, um das oberste Ziel zu finden:

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

   Führen Sie zum Abfragen nach Tickets aus, die in einem bestimmten Datums- und Zeitbereich verkauft wurden, eine Abfrage wie die folgende aus:

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

   Sie können beide Beispielabfragen an Ihre eigenen Bedürfnisse anpassen. Weitere Informationen über die Verwendung von Athena zum Ausführen von Abfragen finden Sie unter Erste Schritte im Benutzerhandbuch zu Amazon Athena.

Bereinigen

Um zu vermeiden, dass nach dem Testen Nutzungsgebühren entstehen, löschen Sie die folgenden Ressourcen, die Sie während des Lernprogramms erstellt haben:

- Amazon SNS-Abonnement
- Amazon SNS-Thema
- Amazon-Simple-Queue-Service-(Amazon-SQS)-Warteschlangen
- Amazon-S3-Bucket
- Bereitstellungsstream
- AWS Identity and Access Management (IAM) -Rollen und -Richtlinien

Automatisieren der Amazon SNS SNS-Nachrichtenarchivierung mit einer Vorlage AWS CloudFormation

Um die Bereitstellung von Amazon SNS Anwendungsfall für die Nachrichtenarchivierung und -analyse zu automatisieren, können Sie die folgende YAML-Vorlage verwenden:

```
---
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 Amazon SNS SNS-Benachrichtigungen an Lambda-Funktionen zur automatisierten Verarbeitung

Amazon SNS ist integriert und ermöglicht es Ihnen AWS Lambda, Lambda-Funktionen als Reaktion auf Amazon SNS SNS-Benachrichtigungen auszulösen. Wenn eine Nachricht in einem SNS-Thema veröffentlicht wird, das eine Lambda-Funktion abonniert hat, wird die Lambda-Funktion mit der Nutzlast der veröffentlichten Nachricht aufgerufen. Die Lambda-Funktion empfängt die Nachrichtennutzlast als Eingabeparameter und kann die Informationen in der Nachricht bearbeiten, die Nachricht in anderen SNS-Themen veröffentlichen oder die Nachricht an andere Dienste senden. AWS

Amazon SNS unterstützt auch die Attribute für den Nachrichtenübertragungsstatus für Benachrichtigungen, die an Lambda-Endpunkte gesendet wurden. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.

Themen

- Voraussetzungen für die regionsübergreifende Integration von Amazon SNS mit Lambda-Funktionen
- Abonnieren einer Lambda-Funktion für ein Amazon SNS SNS-Thema

Voraussetzungen für die regionsübergreifende Integration von Amazon SNS mit Lambda-Funktionen

Zum Aufrufen von Lambda-Funktionen mithilfe von Amazon SNS-Benachrichtigungen, gehen Sie wie folgt vor:

- Eine Lambda-Funktion
- Amazon-SNS-Thema

Weitere Informationen zum Erstellen einer Lambda-Funktion zur Verwendung mit Amazon SNS finden Sie unter Verwenden von Lambda mit Amazon SNS. Informationen zum Erstellen eines Amazon SNS-Themas finden Sie unter Create a topic (Erstellen eines Themas).

Wenn Sie Amazon SNS zum Übermitteln von Nachrichten aus Opt-in-Regionen an standardmäßig aktivierte Regionen verwenden, müssen Sie die in der AWS Lambda-Funktion erstellte Richtlinie ändern, indem Sie den Prinzipal `sns.amazonaws.com` durch `sns.<opt-in-region>.amazonaws.com` ersetzen.

Wenn Sie beispielsweise eine Lambda-Funktion in USA Ost (Nord-Virginia) für ein SNS-Thema in Asien-Pazifik (Hongkong) abonnieren möchten, ändern Sie den Prinzipal in der AWS Lambda-Funktionsrichtlinie in `sns.ap-east-1.amazonaws.com`. Zu den Opt-in-Regionen gehören alle Regionen, die nach dem 20. März 2019 gestartet wurden, darunter Asien-Pazifik (Hongkong), Naher Osten (Bahrain), EU (Mailand) und Afrika (Kapstadt). Regionen, die vor dem 20. März 2019 gestartet wurden, sind standardmäßig aktiviert.

> **Note**
>
> AWS unterstützt keine regionsübergreifende Lieferung an Lambda von einer Region, die standardmäßig aktiviert ist, in eine Opt-in-Region. Darüber hinaus wird die regionenübergreifende Weiterleitung von SNS-Nachrichten von Opt-in-Regionen an andere Opt-in-Regionen nicht unterstützt.

Abonnieren einer Lambda-Funktion für ein Amazon SNS SNS-Thema

In diesem Thema wird erklärt, wie Sie eine Lambda-Funktion für ein Amazon SNS SNS-Thema abonnieren, sodass die Funktion durch veröffentlichte Nachrichten ausgelöst werden kann.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema aus.
4. Wählen Sie im Abschnitt Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen) aus.
5. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Überprüfen Sie den gewählten Topic ARN (Themen-ARN).

b. Wählen Sie als Protokoll die Option. AWS Lambda
c. Geben Sie für Endpoint (Endpunkt) den ARN einer Funktion ein.
d. Wählen Sie Create subscription (Abonnement erstellen) aus.

Wenn eine Nachricht in einem SNS-Thema veröffentlicht wird, das eine Lambda-Funktion abonniert hat, wird die Lambda-Funktion mit der Nutzlast der veröffentlichten Nachricht aufgerufen. Informationen zur Verwendung AWS Lambda mit Amazon SNS, einschließlich eines Tutorials, finden Sie unter AWS Lambda Mit Amazon SNS verwenden.

Fanout von Amazon SNS SNS-Benachrichtigungen an Amazon SQS SQS-Warteschlangen zur asynchronen Verarbeitung

Amazon SNS arbeitet eng mit Amazon-Simple-Warteschlangen-Service (Amazon SQS) zusammen. Beide Services bieten verschiedene Vorteile für Entwickler. Amazon SNS ermöglicht Anwendungen zeitkritische Nachrichten über einen Push-Mechanismus an mehrere Abonnenten versenden, sodass diese nicht mehr in regelmäßigen Abständen selbst nach Updates sehen müssen. Amazon SQS ist ein Nachrichten-Warteschlangenservice, der von verschiedenen Anwendungen verwendet wird, um Nachrichten über ein Polling-Modell auszutauschen. Mit dem Service kann das Senden und Empfangen von Komponenten entkoppelt werden, ohne dass jede Komponente gleichzeitig verfügbar sein muss. Mithilfe von Amazon SNS und Amazon SQS, können Nachrichten einerseits an Anwendungen übermittelt werden, für die eine direkte Benachrichtigung über Ereignisse erforderlich ist, und gleichzeitig für andere Anwendungen zur späteren Bearbeitung in eine Amazon-SQS-Warteschlange gestellt werden.

Wenn Sie eine Amazon-SQS-Warteschlange für ein Amazon-SNS-Thema abonnieren, können Sie eine Nachricht zum Thema veröffentlichen und Amazon SNS sendet eine Amazon-SQS-Nachricht an die abonnierte Warteschlange. Die Amazon-SQS-Nachricht enthält den Betreff und die Nachricht, die zum Thema veröffentlicht wurde, sowie Metadaten über die Nachricht in einem JSON-Dokument. Die Amazon-SQS-Nachricht sieht ähnlich aus wie das folgende JSON-Dokument.

```
{
   "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"
}
```

Abonnieren einer Amazon-SQS-Warteschlange für ein Amazon-SNS-Thema

Um einem Amazon SNS SNS-Thema das Senden von Nachrichten an eine Amazon SQS SQS-Warteschlange zu ermöglichen, wählen Sie eine der folgenden Optionen:

- Verwenden der Amazon-SQS-Konsole, was den Prozess vereinfacht. Weitere Informationen finden Sie unter Abbonieren einer Amazon-SQS-Warteschlange zu einem Amazon-SNS-Thema im Amazon-Simple-Warteschlangen-Service-Entwicklerhandbuch.
- Gehen Sie dazu wie folgt vor:
  1. Rufen Sie den Amazon-Ressourcennamen (ARN) der Warteschlange ab, an die Sie Nachrichten senden möchten, sowie das Thema, zu dem Sie die Warteschlange abonnieren möchten.
  2. Erteilen Sie die Berechtigung `sqs:SendMessage` für das Amazon-SNS-Thema, damit es Nachrichten an die Warteschlange senden kann.
  3. Abonnieren der Warteschlange für das Amazon-SNS-Thema.
  4. Gewähren Sie IAM-Benutzern oder AWS-Konten die entsprechenden Berechtigungen zur Veröffentlichung im Amazon-SNS-Thema und zum Lesen von Nachrichten in der Amazon-SQS-Warteschlange.
  5. Testen Sie die Ergebnisse, indem Sie eine Nachricht zu dem Thema veröffentlichen und die Nachricht von der Warteschlange lesen.

Weitere Informationen zur Einrichtung eines Themas zum Senden von Nachrichten an eine Warteschlange in einem anderen AWS-Konto finden Sie unter Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange in einem anderen Konto.

Eine AWS CloudFormation Vorlage, die ein Thema erstellt, das Nachrichten an zwei Warteschlangen sendet, finden Sie unter. Automatisieren Sie das Versenden von Amazon SNS zu Amazon SQS mit AWS CloudFormation

Schritt 1: Abrufen des ARN der Warteschlange und des Themas

Beim Abonnieren einer Warteschlange zu Ihrem Thema benötigen Sie eine Kopie des ARN für die Warteschlange. Wenn Sie eine Berechtigung für das Thema erteilen, um Nachrichten an die Warteschlange zu senden, benötigen Sie auch eine Kopie des ARN für das Thema.

Um den Warteschlangen-ARN abzurufen, können Sie die Amazon SQS SQS-Konsole oder die GetQueueAttributesAPI-Aktion verwenden.

So rufen Sie den ARN der Warteschlange von der Amazon-SQS-Konsole ab

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon SQS SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Aktivieren Sie das Kontrollkästchen für die Warteschlange, deren ARN Sie abrufen möchten.
3. Kopieren Sie auf der Registerkarte Details den ARN-Wert, sodass Sie ihn zum Abonnieren des Amazon-SNS-Themas verwenden können.

Um den ARN des Themas abzurufen, können Sie die `sns-get-topic-attributes` Amazon-SNS-Konsole, den Befehl oder die `GetQueueAttributes` API-Aktion verwenden.

So rufen Sie den ARN des Themas von der Amazon-SNS-Konsole ab

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich das Thema aus, dessen ARN Sie abrufen möchten.
3. Kopieren Sie im Bereich Details (Themendetails) den Wert unter Topic ARN (Themen-ARN), um damit die Berechtigung für das Amazon-SNS-Thema zum Senden von Nachrichten an die Warteschlange zu erteilen.

Schritt 2: Erteilen der Berechtigung für das Amazon SNS-Thema zum Senden von Nachrichten an die Amazon-SQS-Warteschlange

Damit ein Amazon-SNS-Thema Nachrichten an eine Warteschlange senden kann, müssen Sie eine Richtlinie für die Warteschlange einrichten, die es dem Amazon-SNS-Thema erlaubt, die Aktion `sqs:SendMessage` auszuführen.

Bevor Sie eine Warteschlange für ein Thema abonnieren, benötigen Sie ein Thema und eine Warteschlange. Wenn Sie noch kein Thema bzw. keine Warteschlange erstellt haben, erstellen

Sie diese jetzt. Weitere Informationen finden Sie unter Erstellen eines Themas und Erstellen einer Warteschlange im Entwicklerhandbuch für den Amazon-Simple-Warteschlangenservice.

Um eine Richtlinie für eine Warteschlange festzulegen, können Sie die Amazon SQS SQS-Konsole oder die SetQueueAttributesAPI-Aktion verwenden. Bevor Sie beginnen, stellen Sie sicher, dass Sie über den ARN für das Thema verfügen, das Nachrichten an die Warteschlange senden soll. Wenn Sie eine Warteschlange für mehrere Themen abonnieren, muss Ihre Richtlinie ein `Statement`-Element für jedes Thema enthalten.

So legen Sie mithilfe der Amazon SQS SQS-Konsole eine SendMessage Richtlinie für eine Warteschlange fest

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon SQS SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Aktivieren Sie das Kontrollkästchen für die Warteschlange, deren Richtlinien Sie einrichten möchten. Wählen Sie die Registerkarte Access policy und danach Edit.
3. In derZugriffsrichtliniedefinieren Sie, wer auf Ihre Warteschlange zugreifen kann.

   - Fügen Sie eine Bedingung hinzu, die die Aktion für das Thema zulässt.
   - Legen Sie `Principal` als den Amazon-SNS-Service fest, wie im folgenden Beispiel gezeigt.
   - Verwenden Sie den globale Konditionsschlüssel von `aws:SourceArn` oder den `aws:SourceAccount` zum Schutz vor dem Szenario Confused Deputy. Legen Sie zur Verwendung dieses Bedingungsschlüssels den Wert auf den ARN Ihres Themas fest. Wenn Ihre Warteschlange mehrere Themen abonniert hat, können Sie stattdessen `aws:SourceAccount` verwenden.

   Die folgende Richtlinie ermöglicht beispielsweise das Senden MyTopic von Nachrichten an 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"
        }
      }
    }
  ]
}
```

Schritt 3: Abonnieren der Warteschlange für das Amazon-SNS-Thema

Um Nachrichten über ein Thema an eine Warteschlange zu senden, müssen Sie die Warteschlange für das Amazon-SNS-Thema abonnieren. Sie geben die Warteschlange durch ihre ARN an. Um ein Thema zu abonnieren, können Sie die Amazon-SNS-Konsole, den `sns-subscribe`-CLI-Befehl oder die `Subscribe`-API-Aktion verwenden. Bevor Sie beginnen, stellen Sie sicher, dass Sie über den ARN für die Warteschlange verfügen, die Sie abonnieren möchten.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema aus.
4. Wählen Sie auf der ***MyTopic*** Seite auf der Seite Abonnements die Option Abonnement erstellen aus.
5. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Überprüfen Sie den Topic ARN (Themen-ARN).
   b. FürProtocol (Protokoll), wählen Sie Amazon SQSaus.
   c. Geben Sie für Endpoint (Endpunkt) den ARN einer Amazon-SQS-Warteschlange ein.
   d. Klicken Sie auf Create subscription (Abonnement erstellen).

   Wenn das Abonnement bestätigt wurde, wird die Subscription ID (Abonnement-ID) für Ihr neues Abonnement angezeigt. Wenn der Eigentümer der Warteschlange das Abonnement erstellt, wird das Abonnement automatisch bestätigt und es sollte nahezu umgehend aktiv sein.

   Normalerweise abonnieren Sie Ihre eigene Warteschlange für ein eigenes Thema in Ihrem Konto. Sie können aber auch eine Warteschlange aus einem anderen Konto für Ihr Thema abonnieren. Wenn der Benutzer, der das Abonnement erstellt, nicht der Eigentümer der

Warteschlange ist (z. B. wenn ein Benutzer von Konto A eine Warteschlange von Konto B für ein Thema in Konto A erstellt), muss das Abonnement bestätigt werden. Weitere Informationen zum Abonnieren einer Warteschlange von einem anderen Konto und zum Bestätigen des Abonnements finden Sie unter Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange in einem anderen Konto.

Schritt 4: Erteilen von Berechtigungen an Benutzer für die entsprechenden Themen- und Warteschlangenaktionen

Sie sollten AWS Identity and Access Management (IAM) verwenden, um nur geeigneten Benutzern das Veröffentlichen im Amazon SNS SNS-Thema und das Lesen/Löschen von Nachrichten aus der Amazon SQS SQS-Warteschlange zu ermöglichen. Weitere Informationen zum Steuern von Aktionen für Themen und Warteschlangen für IAM-Benutzer finden Sie unter Verwenden von identitätsbasierten Richtlinien mit Amazon SNS, und Identity and Access Management in Amazon SQS im Amazon Simple Warteschlange Service-Entwicklerhandbuch.

Es gibt zwei Möglichkeiten, den Zugriff auf ein Thema oder eine Warteschlange zu steuern:

- Hinzufügen einer Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe. Die einfachste Möglichkeit, Benutzern Berechtigungen für Themen oder Warteschlangen zu gewähren, besteht darin, eine Gruppe zu erstellen, dieser die entsprechende Richtlinie hinzuzufügen und anschließend der Gruppe Benutzer hinzuzufügen. Es ist wesentlich einfacher, Benutzer einer Gruppe hinzufügen und zu entfernen, um die Richtlinien für einzelne Benutzer im Auge zu behalten.
- Hinzufügen einer Richtlinie zum Thema oder zur Warteschlange. Wenn Sie einem Thema oder einer Warteschlange einem anderen AWS Konto Berechtigungen erteilen möchten, können Sie dies nur tun, indem Sie eine Richtlinie hinzufügen, deren Hauptbestandteil die Person ist, für die AWS-Konto Sie Berechtigungen erteilen möchten.

Sie sollten die erste Methode für die meisten Fällen verwenden (Anwenden von Richtlinien für Gruppen und Verwalten von Berechtigungen für Benutzer durch Hinzufügen oder Entfernen der entsprechenden Benutzer zu den Gruppen). Wenn Sie Berechtigungen für einen Benutzer in einem anderen Konto erteilen müssen, sollten Sie die zweite Methode verwenden.

Hinzufügen einer Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe

Wenn Sie die folgende Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe hinzufügen würden, würden Sie diesem Benutzer oder Mitgliedern dieser Gruppe die Erlaubnis geben, die `sns:Publish` Aktion für das Thema MyTopic auszuführen.

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

Wenn Sie einem IAM-Benutzer oder einer IAM-Gruppe die folgende Richtlinie hinzufügen würden, würden Sie diesem Benutzer oder Mitgliedern dieser Gruppe die Berechtigung erteilen, die `sqs:DeleteMessage` Aktionen `sqs:ReceiveMessage` und in den Warteschlangen MyQueue 1 und 2 auszuführen. MyQueue

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

Hinzufügen einer Richtlinie zu einem Thema oder einer Warteschlange

Die folgenden Beispielrichtlinien zeigen, wie weitere Kontoberechtigungen für ein Thema und eine Warteschlange erteilt werden.

> **Note**
>
> Wenn Sie einer anderen Person AWS-Konto Zugriff auf eine Ressource in Ihrem Konto gewähren, gewähren Sie damit auch IAM-Benutzern, die über Administratorrechte (Wildcard-Zugriff) für diese Ressource verfügen. Allen anderen IAM-Benutzern des anderen Kontos wird der Zugriff auf Ihre Ressource automatisch verweigert. Wenn Sie möchten, dass bestimmte IAM-Benutzer dieses AWS-Konto Zugriff auf Ihre Ressource haben, muss ein IAM-Benutzer mit Administratorrechten diesen IAM-Benutzern die Berechtigungen für die Ressource übertragen. Weitere Informationen zur kontenübergreifenden Berechtigungsübertragung finden Sie unter Aktivieren des kontenübergreifenden Zugriffs im Leitfaden zur Verwendung von IAM.

Wenn Sie die folgende Richtlinie zu einem Thema MyTopic im Konto 123456789012 hinzufügen würden, würden Sie dem Konto 111122223333 die Berechtigung erteilen, die Aktion für dieses Thema auszuführen. `sns:Publish`

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

Wenn Sie die folgende Richtlinie zu einer Warteschlange MyQueue im Konto 123456789012 hinzufügen würden, würden Sie dem Konto 111122223333 die Berechtigung erteilen, die Aktionen in dieser Warteschlange auszuführen. `sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
```

```
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Schritt 5: Testen der Warteschlangen-Abonnements des Themas

Sie können die Warteschlange-Abonnements für ein Thema testen, indem Sie eine Veröffentlichung für das Thema vornehmen und die Nachricht anzeigen, die das Thema an die Warteschlange sendet.

So veröffentlichen Sie über die Amazon-SNS-Konsole zu einem Thema

1. Melden Sie sich mit den Anmeldeinformationen des AWS-Konto oder des IAM-Benutzers mit der Berechtigung zum Veröffentlichen des Themas an AWS Management Console und öffnen Sie die Amazon SNS SNS-Konsole unter. https://console.aws.amazon.com/sns/
2. Wählen Sie im Navigationsbereich das Thema und Publish to Topic (Zu Thema veröffentlichen) aus.
3. Geben Sie in das Feld Subject (Betreff) einen Betreff ein (z. B. **`Testing publish to queue`**). Geben Sie in das Feld Message (Nachricht) Text ein (z. B. **`Hello world!`**) und wählen Sie Publish Message (Nachricht veröffentlichen) aus. Die folgende Meldung wird angezeigt: Ihre Nachricht wurde erfolgreich veröffentlicht.

So zeigen Sie die Nachricht von dem Thema mithilfe der Amazon-SQS-Konsole an

1. Melden Sie sich mit den Anmeldeinformationen des AWS-Konto oder des IAM-Benutzers mit der Berechtigung zum Anzeigen von Nachrichten in der Warteschlange an AWS Management Console und öffnen Sie die Amazon SQS SQS-Konsole unter. https://console.aws.amazon.com/sqs/
2. Wählen Sie eine Warteschlange, die das Thema abonniert hat.

3. Wählen Sie Send and receive messages (Nachrichten senden und empfangen) und dann Poll for messages (Nachrichten abfragen) aus. Es wird eine Nachricht des Typs Notification (Benachrichtigung) angezeigt.
4. Wählen Sie in der Spalte Bodydie Option More Details. Das Feld Message Details (Nachrichtendetails) enthält ein JSON-Dokument mit dem Betreff und der Nachricht, die Sie zum Thema veröffentlicht haben. Die Nachricht sieht ähnlich aus wie das folgende JSON-Dokument.

```
{
  "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. Klicken Sie auf Close (Schließen). Sie haben erfolgreich eine Nachricht zu einem Thema veröffentlicht, das Benachrichtigungen an eine Warteschlange sendet.

Automatisieren Sie das Versenden von Amazon SNS zu Amazon SQS mit AWS CloudFormation

AWS CloudFormation ermöglicht es Ihnen, eine Vorlagendatei zu verwenden, um eine Sammlung von AWS Ressourcen als eine Einheit zu erstellen und zu konfigurieren. Dieser Abschnitt enthält eine Beispielvorlage, die das Bereitstellen von Themen, die für Warteschlangen veröffentlicht werden, vereinfacht. Die Vorlagen übernehmen für Sie die Einrichtung und führen folgende Schritte aus: Erstellen von zwei Warteschlangen, Erstellen eines Themas mit Abonnements für die Warteschlangen, Hinzufügen einer Richtlinie zu den Warteschlangen, sodass das Thema Nachrichten an die Warteschlangen senden kann, und Erstellen von IAM-Benutzern und -Gruppen, um den Zugriff auf diese Ressourcen zu steuern.

Weitere Informationen zum Bereitstellen von AWS Ressourcen mithilfe einer AWS CloudFormation Vorlage finden Sie unter Erste Schritte im AWS CloudFormation Benutzerhandbuch.

Verwenden einer AWS CloudFormation Vorlage zum Einrichten von Themen und Warteschlangen innerhalb eines AWS-Konto

Die Beispielvorlage erstellt ein Amazon-SNS-Thema, das Nachrichten an zwei Amazon-SQS-Warteschlanges senden kann, mit den entsprechenden Berechtigungen für die Mitglieder einer IAM-Gruppe für die Veröffentlichung zum Thema und einer weiteren zum Lesen von Nachrichten aus den Warteschlangen. Die Vorlage erstellt außerdem IAM-Benutzer, die jeder Gruppe hinzugefügt werden.

Sie kopieren den Inhalt der Vorlage in eine Datei. Sie können die Vorlage auch von der AWS CloudFormation Vorlagenseite herunterladen. Wählen Sie auf der Vorlagenseite Mustervorlagen nach AWS Service durchsuchen und wählen Sie dann Amazon Simple Queue Service aus.

My SNSTopic ist so eingerichtet, dass es auf zwei abonnierten Endpunkten veröffentlicht, bei denen es sich um zwei Amazon SQS SQS-Warteschlangen (MyQueue1 und 2) handelt. MyQueue MyPublishTopicGroup ist eine IAM-Gruppe, deren Mitglieder berechtigt sind, SNSTopic mithilfe der API-Aktion Publish oder des Befehls sns-publish auf My zu veröffentlichen. Die Vorlage erstellt die IAM-Benutzer MyPublishUser und gibt ihnen Anmeldeprofile MyQueueUser und Zugriffsschlüssel. Der Benutzer, der einen Stack mit dieser Vorlage erstellt, gibt die Passwörter für die Anmeldeprofile als Eingabeparameter an. Die Vorlage erstellt Zugriffsschlüssel für die beiden IAM-Benutzer mit MyPublishUserKey und. MyQueueUserKey AddUserToMyPublishTopicGroup fügt dem MyPublishUser hinzu, MyPublishTopicGroup sodass dem Benutzer die der Gruppe zugewiesenen Berechtigungen zugewiesen werden.

Meine RDMessage QueueGroup ist eine IAM-Gruppe, deren Mitglieder über die Berechtigung verfügen, mithilfe der API-Aktionen und Nachrichten aus den beiden Amazon SQS SQS-Warteschlangen zu lesen ReceiveMessageund DeleteMessagezu löschen. AddUserToMyQueueGroup wird MyQueueUser zu My hinzugefügt, RDMessage QueueGroup sodass dem Benutzer die der Gruppe zugewiesenen Berechtigungen zugewiesen werden. MyQueuePolicy weist My die Erlaubnis SNSTopic zu, seine Benachrichtigungen den beiden Warteschlangen zu veröffentlichen.

Die folgende Liste zeigt den Inhalt der AWS CloudFormation Vorlage.

```
{
  "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 Amazon SNS SNS-Benachrichtigungen an HTTPS-Endpunkte

Sie können Amazon SNS verwenden, um Benachrichtigungen an einzelne oder mehrere HTTP- oder HTTPS-Endpunkte zu senden. Wenn Sie ein Thema für einen Endpunkt abonniert haben, können Sie eine Benachrichtigung zu diesem Thema veröffentlichen Amazon SNS sendet eine HTTP POST-Anforderung zur Zustellung des Inhalts der Benachrichtigung an den registrierten Endpunkt. Wenn Sie den Endpunkt abonniert haben, können Sie auswählen, ob Amazon SNS HTTP oder HTTPS zum Senden der POST-Anforderung an den Endpunkt verwenden soll. Wenn Sie HTTPS verwenden, können Sie die Vorteile der Amazon SNS-Unterstützung für folgende Aktionen nutzen:

- Server Name Indication (SNI, Servernamensanzeige) - Auf diese Weise kann Amazon SNS; HTTPS-Endpunkte unterstützen, die SNI benötigen. Das kann z. B. ein Server sein, der mehrere Zertifikate für das Hosten mehrerer Domänen benötigt. Weitere Informationen zur SNI (Servernamensanzeige) finden Sie unter Server Name Indication.

- Standard- und Digest-Zugriffsauthentifizierung— Auf diese Weise können Sie in der HTTPS-URL einen Benutzernamen und ein Kennwort für die HTTP POST-Anforderung angeben, z. B.`https://user:password@domain.com`oder`https://user@domain.com` Der Benutzername und das Kennwort sind über die SSL-Verbindung verschlüsselt, die bei der Verwendung von HTTPS aufgebaut wird. Nur der Domänenname wird als Klartext gesendet. Weitere Informationen zu Basic and Digest Access Authentication finden Sie unter RFC-2617.

> **Important**
>
> Amazon SNS unterstützt derzeit keine privaten HTTP(S)-Endpunkte.

HTTPS können nur über die Amazon SNS `GetSubscriptionAttributes` SNS-API-Aktion abgerufen URLs werden, und zwar für Prinzipale, denen Sie API-Zugriff gewährt haben.

> **Note**
>
> Die Client-Service muss die `HTTP/1.1 401 Unauthorized`-Header-Antwort unterstützen.

Die Anforderung enthält den Betreff und die Nachricht, die zum Thema veröffentlicht wurde, sowie Metadaten zur Benachrichtigung in einem JSON-Dokument. Die Anforderung sieht ähnlich aus wie die folgende HTTP POST-Anforderung: Weitere Informationen über den HTTP-Header und das JSON-Format des Anforderungstextes finden Sie unter HTTP/HTTPS-Header und JSON-Format für eine HTTP/HTTPS-Benachrichtigung.

> **Note**
>
> Amazon SNS betrachtet alle 5XX-Fehler und die Fehler 429 (zu viele gesendete Anfragen) als wiederholbar. Diese Fehler unterliegen den Lieferrichtlinien. Alle anderen Fehler gelten als dauerhafte Fehler und es werden keine erneuten Versuche unternommen.

```
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"
}
```

Abonnieren eines HTTPS-Endpunkts für ein Amazon SNS SNS-Thema

In diesem Thema wird erklärt, wie Sie HTTP/S-Endpunkte für Amazon SNS SNS-Themen abonnieren.

Themen

- Schritt 1: Stellen Sie sicher, dass Ihr Endpunkt Amazon-SNS-Nachrichten verarbeiten kann
- Schritt 2: Abonnieren des HTTP-/HTTPS-Endpunkts zum Amazon-SNS-Thema
- Schritt 3: Bestätigen Sie Ihr Amazon SNS SNS-Abonnement
- Schritt 4: Optional — Legen Sie die Lieferbedingungen für das Amazon SNS SNS-Abonnement fest
- Schritt 5: Optional — Erteilen Sie Benutzern die Erlaubnis, im Amazon SNS SNS-Thema zu veröffentlichen
- Schritt 6: Amazon SNS SNS-Nachrichten an den HTTP/HTTPS-Endpunkt senden

Schritt 1: Stellen Sie sicher, dass Ihr Endpunkt Amazon-SNS-Nachrichten verarbeiten kann

Bevor Sie ein Thema für Ihren HTTP- oder HTTPS-Endpunkt abonnieren, müssen Sie sicherstellen, dass der HTTP- oder HTTPS-Endpunkt über ausreichend Kapazität zur Durchführung von HTTP POST-Anforderungen verfügt, die Amazon SNS zum Senden der Abonnementbestätigungs- und Benachrichtigungsmitteilungen verwendet. In der Regel bedeutet dies das Erstellen und Bereitstellen einer Webanwendung, die HTTP-Anforderungen von Amazon SNS verarbeitet (z. B. ein Java Servlet, wenn Ihr Endpunkt-Host Linux mit Apache und Tomcat ausführt). Wenn Sie einen HTTP-Endpunkt abonnieren, sendet Amazon SNS eine Anforderung für die Abonnementbestätigung an diesen

Endpunkt. Der Endpunkt muss beim Erstellen des Abonnements bereit für den Empfang und die Verarbeitung der Anforderung sein, da Amazon SNS diese zu diesem Zeitpunkt sendet. Amazon SNS sendet solange keine Benachrichtigungen an diesen Endpunkt, bis Sie das Abonnement bestätigen. Sobald Sie das Abonnement bestätigen, sendet Amazon-SNS-Benachrichtigungen an den Endpunkt, wenn eine Nachricht zum abonnierten Thema veröffentlicht wird.

Einrichten des Endpunkts für die Verarbeitung von Abonnementsbestätigungen und Benachrichtigungsmitteilungen

1. Der Code sollte die HTTP-Header der HTTP POST-Anforderungen lesen, die Amazon SNS an Ihrem Endpunkt sendet. Ihr Code sollte im Header-Feld den Nachrichtentyp abrufen `x-amz-sns-message-type`, den Ihnen Amazon SNS gesendet hat. Im Header finden Sie den Nachrichtentyp, ohne dass Sie den Text der HTTP-Anforderung analysieren müssen. Diese beiden Typen müssen Sie verarbeiten: `SubscriptionConfirmation` und `Notification`. Die `UnsubscribeConfirmation` Mitteilung wird nur verwendet, wenn das Abonnement des Themas gelöscht wird.

   Details zum HTTP-Header finden Sie unter HTTP/HTTPS-Header. Die folgende HTTP POST-Anforderung ist ein Beispiel für eine Abonnement-Bestätigungsnachricht.

```
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. Der Code sollte das JSON-Dokument im Text der HTTP POST-Anfrage und im text/plain vom Inhaltstyp analysieren, um die Name-Wert-Paare zu lesen, aus denen die Amazon-SNS-Nachricht besteht. Verwenden Sie einen JSON-Parser, der das Umwandeln der Escape-Sequenzen von Steuerzeichen in die entsprechenden ASCII-Zeichenwerte durchführt (z. B. Konvertieren von \n in einen Zeilenumbruch). Sie können einen vorhandenen JSON-Parser wie den Jackson JSON Processor verwenden oder einen eigenen schreiben. Zum Senden des Textes in den Betreff- und Nachrichtenfeldern im gültigen JSON-Format muss Amazon SNS einige Steuerzeichen in Escape-Sequenzen umwandeln, die in das JSON-Dokument integriert werden können. Wenn Sie das JSON-Dokument im Text der POST-Anforderung am Endpunkt empfangen, müssen Sie die Escape-Zeichen wieder in die ursprünglichen Zeichenwerte zurückkonvertieren, wenn Sie eine exakte Darstellung des ursprünglichen Betreffs und der zum Thema veröffentlichten Nachrichten sehen möchten. Dies ist besonders wichtig, wenn Sie die Signatur einer Benachrichtigung überprüfen möchten, da die Signatur die Benachrichtigung und den Betreff in der ursprünglichen Form als Teil der zu signierenden Zeichenfolge verwendet.
3. Der Code sollte die Authentizität einer Benachrichtigung, die Abonnementbestätigung oder die Bestätigungsnachricht für die Beendigung des Abonnements überprüfen, die Amazon SNS gesendet hat. Unter Verwendung der Informationen in der Amazon SNS-Nachricht kann der Endpunkt die Signatur neu erstellen. Sie können dann den Inhalt der Nachricht überprüfen, wenn Ihre Signatur mit der Signatur übereinstimmt, die Amazon SNS zusammen mit der Nachricht gesendet hat. Weitere Informationen zum Überprüfen der Signatur einer Nachricht finden Sie unter Überprüfen der Signaturen von Amazon-SNS-Nachrichten.
4. Basierend auf dem Typ, der im Header-Feld `x-amz-sns-message-type` angegeben ist, sollte der Code das JSON-Dokument im Text der HTTP-Anforderung lesen und die Mitteilung verarbeiten. Hier finden Sie die Richtlinien für die beiden primären Nachrichtentypen:

   SubscriptionConfirmation

   Lesen Sie den Wert `SubscribeURL` und rufen Sie die URL auf. Um das Abonnement zu bestätigen und Benachrichtigungen am Endpunkt zu empfangen, müssen Sie die `SubscribeURL`URL aufrufen (z. B. indem Sie eine HTTP GET-Anforderung an die URL senden). Sehen Sie sich das Beispiel der HTTP-Anforderung im vorherigen Schritt an,

um zu sehen, wie `SubscribeURL` aussieht. Weitere Informationen zum Format der `SubscriptionConfirmation`-Nachricht finden Sie unter JSON-Format für die HTTP/HTTPS-Abonnement-Bestätigung. Wenn Sie die URL aufrufen, erhalten Sie eine Antwort zurück, die wie das folgende XML-Dokument aussieht. Das Dokument gibt den Abonnement-ARN für den Endpunkt innerhalb des `ConfirmSubscriptionResult`-Elements zurück.

```
<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>
```

Als Alternative zum Besuch von können Sie das Abonnement mit der ConfirmSubscriptionAktion bestätigen`SubscribeURL`, wobei der Wert auf den entsprechenden Wert in der `SubscriptionConfirmation` Nachricht `Token` gesetzt wird. Wenn Sie möchten, dass nur der Eigentümer des Themas und der Abonnent den Endpunkt abmelden können sollen, rufen Sie die `ConfirmSubscription`-Aktion mit einer AWS -Signatur auf.

Benachrichtigung

Lesen Sie die Werte für `Subject` und `Message`, um die Benachrichtigungsinformation zu erhalten, die zum Thema veröffentlicht wurde.

Weitere Informationen über das Format der `Notification`-Nachricht finden Sie unter HTTP/HTTPS-Header. Die folgende HTTP POST-Anforderung ist ein Beispiel für eine Benachrichtigung, die an den Endpunkt example.com gesendet wurde.

```
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. Stellen Sie sicher, dass der Endpunkt auf die HTTP-POST-Nachricht von Amazon SNS mit dem entsprechenden Statuscode antwortet. Die Verbindung wird in etwa 15 Sekunden unterbrochen. Wenn Ihr Endpunkt nicht reagiert, bevor die Verbindung abgelaufen ist, oder wenn Ihr Endpunkt einen Statuscode außerhalb des Bereichs 200—4 xx zurückgibt, betrachtet Amazon SNS die Zustellung der Nachricht als fehlgeschlagenen Versuch.
6. Stellen Sie sicher, dass Ihr Code wiederholte Nachrichtenzustellungen von Amazon SNS verarbeiten kann. Wenn Amazon SNS keine erfolgreiche Antwort von Ihrem Endpunkt erhält, versucht der Dienst, die Nachricht erneut zuzustellen. Dies gilt für alle Nachrichten, einschließlich der Abonnement-Bestätigungsnachricht. Wenn die erste Zustellung der Nachricht fehlschlägt, versucht Amazon SNS bis zu drei Mal eine erneute Zustellung mit einer Verzögerung von 20 Sekunden zwischen den fehlgeschlagenen Versuchen.

> **Note**
>
> Die Nachrichtenanforderung läuft nach etwa 15 Sekunden ab. Das bedeutet, wenn die Nachrichtenzustellung wegen einer Zeitüberschreitung nicht zugestellt werden kann, startet Amazon SNS nach etwa 35 Sekunden einen erneuten Zustellversuch. Sie können eine andere Zustellungsrichtlinie für den Endpunkt festlegen.

Amazon SNS verwendet das Header-Feld `x-amz-sns-message-id`, um jede zu einem Amazon-SNS-Thema veröffentlichte Nachricht eindeutig zu identifizieren. Indem Sie die IDs verarbeiteten Nachrichten mit den eingehenden Nachrichten vergleichen, können Sie feststellen, ob es sich bei der Nachricht um einen Wiederholungsversuch handelt.

7. Wenn Sie einen HTTPS-Endpunkt abonnieren, stellen Sie sicher, dass Ihr Endpunkt ein Serverzertifikat von einer vertrauenswürdigen Zertifizierungsstelle (CA) aufweist. Amazon SNS sendet nur Nachrichten an HTTPS-Endpunkte, die über ein Serverzertifikat mit der Signatur einer für Amazon SNS vertrauenswürdigen CA verfügt.
8. Stellen Sie den Code bereit, den Sie für den Empfang von Amazon-SNS-Nachrichten erstellt haben. Wenn Sie den Endpunkt abonnieren, muss dieser mindestens die Abonnement-Bestätigungsnachricht empfangen können.

Schritt 2: Abonnieren des HTTP-/HTTPS-Endpunkts zum Amazon-SNS-Thema

Um Nachrichten über ein Thema an einen HTTP- oder HTTPS-Endpunkt zu senden, müssen Sie den Endpunkt für das Amazon SNS-Thema abonnieren. Sie geben den Endpunkt über seine URL an. Zum Abonnieren eines Themas können Sie die Amazon-SNS-Konsole, den Befehl sns-subscribe oder die API-Aktion Subscribe verwenden. Bevor Sie beginnen, stellen Sie sicher, dass Sie die URL für den Endpunkt kennen, den Sie abonnieren möchten, und dass der Endpunkt Bestätigungen und Benachrichtigungsmitteilungen wie in Schritt 1 beschrieben empfangen kann.

Abonnieren eines Themas auf einen HTTP- oder HTTPS-Endpunkt mit der Amazon-SNS-Konsole

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie Abonnement erstellen aus.
4. Wählen Sie in der Dropdown-Liste Protocol (Protokoll) entweder HTTP oder HTTPS aus.
5. Kopieren Sie in das Feld Endpoint (Endpunkt) die URL für den Endpunkt, an den das Thema Nachrichten senden soll, und wählen Sie dann Create Subscription (Abonnement erstellen).
6. Die Bestätigungsmeldung wird angezeigt. Klicken Sie auf Schließen.

   Die Abonnement-ID Ihres neuen Abonnements wird angezeigt PendingConfirmation. Wenn Sie das Abonnement bestätigen, zeigt Subscription ID (Abonnement-ID) die Abonnement-ID an.

Schritt 3: Bestätigen Sie Ihr Amazon SNS SNS-Abonnement

Gehen Sie wie folgt vor, um Ihr Amazon SNS SNS-Abonnement zu bestätigen, um sicherzustellen, dass Ihr Endgerät erfolgreich Nachrichten empfangen kann. Dieser Vorgang umfasst die Einrichtung Ihres Endgeräts für die Bearbeitung eingehender Bestätigungsnachrichten, das Abrufen der Bestätigungs-URL und die Bestätigung des Abonnements. Sie können das Abonnement je nach Konfiguration entweder automatisch oder manuell bestätigen.

1. Nachdem Sie ein Amazon SNS-Thema abonniert haben, sendet Amazon SNS eine Bestätigungsnachricht an Ihren Endpunkt. Diese Nachricht enthält eine`SubscribeURL`, mit der Sie das Abonnement bestätigen müssen.
2. Ihr Endpunkt muss so eingerichtet sein, dass er eingehende Nachrichten von Amazon SNS abhört. Wenn die Bestätigungsnachricht eintrifft, extrahieren Sie sie **`SubscribeURL`**aus der Nachricht.
3. Sobald Sie die haben`SubscribeURL`, können Sie das Abonnement auf zwei Arten bestätigen:
   - Automatische Bestätigung — Ihr Endpunkt kann das Abonnement automatisch bestätigen, indem er eine HTTP-GET-Anfrage an den sendet`SubscribeURL`.

     Diese Methode erfordert kein manuelles Eingreifen.
   - Manuelle Bestätigung — Wenn die automatische Bestätigung nicht eingerichtet ist, kopieren Sie die Bestätigung **`SubscribeURL`**aus der Bestätigungsnachricht und fügen Sie sie in die Adressleiste Ihres Browsers ein.

     Dadurch wird das Abonnement manuell bestätigt.
4. Sie können den Abonnementstatus auch über die Amazon SNS SNS-Konsole überprüfen:
   a. Melden Sie sich bei der Amazon SNS-Konsole an.
   b. Wählen Sie im Navigationsbereich Subscriptions aus.
   c. Finden Sie Ihr Abonnement in der Liste.
      - Wenn bestätigt, `SubscriptionArn` wird das angezeigt.
      - Wenn es immer noch nicht bestätigt ist, wird es als `PendingConfirmation` angezeigt.

Schritt 4: Optional — Legen Sie die Lieferbedingungen für das Amazon SNS SNS-Abonnement fest

Wenn die erste Zustellung der Nachricht fehlschlägt, versucht Amazon SNS bis zu drei Mal eine erneute Zustellung mit einer Verzögerung von 20 Sekunden zwischen den fehlgeschlagenen Versuchen. Wie in Schritt 1 erläutert, sollte Ihr Endpunkt über Code verfügen, der wiederholte Nachrichten verarbeiten kann. Indem Sie die Zustellungsrichtlinie für ein Thema oder ein Abonnement festlegen, können Sie die Häufigkeit und das Intervall steuern, die Amazon SNS für wiederholte Zustellungen fehlgeschlagener Nachrichten verwendet. Sie können den Inhaltstyp für Ihre HTTP/S-Benachrichtigungen auch in `DeliveryPolicy` angeben. Weitere Informationen finden Sie unter Erstellung HTTP/S einer Lieferrichtlinie.

Schritt 5: Optional — Erteilen Sie Benutzern die Erlaubnis, im Amazon SNS SNS-Thema zu veröffentlichen

Standardmäßig hat der Eigentümer des Themas die Berechtigung zur Veröffentlichung des Themas. Um anderen Benutzern oder Anwendungen das Veröffentlichen des Themas zu ermöglichen, sollten Sie AWS Identity and Access Management (IAM) verwenden, um die Veröffentlichungsberechtigung für das Thema zu erteilen. Weitere Informationen über die Berechtigungen für Amazon SNS-Aktionen finden Sie unter Verwenden von identitätsbasierten Richtlinien mit Amazon SNS.

Es gibt zwei Möglichkeiten, den Zugriff auf ein Thema zu steuern:

- Hinzufügen einer Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe. Die einfachste Möglichkeit, Benutzern Berechtigungen für Themen zu gewähren, besteht darin, eine Gruppe zu erstellen, dieser die entsprechende Richtlinie hinzuzufügen und anschließend der Gruppe Benutzer hinzuzufügen. Es ist wesentlich einfacher, Benutzer einer Gruppe hinzufügen und zu entfernen, um die Richtlinien für einzelne Benutzer im Auge zu behalten.
- Hinzufügen einer Richtlinie zum Thema: Wenn Sie Berechtigungen für ein Thema einem anderen AWS -Konto gewähren möchten, können Sie dies nur durch Hinzufügen einer Richtlinie erreichen, deren Prinzipal das AWS-Konto ist, für das Sie die Berechtigungen vergeben möchten.

Sie sollten die erste Methode für die meisten Fällen verwenden (Anwenden von Richtlinien für Gruppen und Verwalten von Berechtigungen für Benutzer durch Hinzufügen oder Entfernen der entsprechenden Benutzer zu den Gruppen). Wenn Sie Berechtigungen für einen Benutzer in einem anderen Konto erteilen müssen, verwenden Sie die zweite Methode.

Wenn Sie einem IAM-Benutzer oder einer IAM-Gruppe die folgende Richtlinie hinzufügen würden, würden Sie diesem Benutzer oder Mitgliedern dieser Gruppe die Berechtigung erteilen, die `sns:Publish` Aktion für das Thema auszuführen. MyTopic

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

Die folgende Beispielrichtlinie zeigt, wie weitere Kontoberechtigungen für ein Thema erteilt werden.

> **Note**
>
> Wenn Sie einer anderen Person AWS-Konto Zugriff auf eine Ressource in Ihrem Konto gewähren, gewähren Sie damit auch IAM-Benutzern, die über Administratorberechtigungen (Wildcard-Zugriff) für diese Ressource verfügen. Allen anderen IAM-Benutzern des anderen Kontos wird der Zugriff auf Ihre Ressource automatisch verweigert. Wenn Sie bestimmten IAM-Benutzern in diesem Bereich AWS-Konto Zugriff auf Ihre Ressource gewähren möchten, muss das Konto oder ein IAM-Benutzer mit Administratorzugriff die Berechtigungen für die Ressource an diese IAM-Benutzer delegieren. Weitere Informationen zur kontenübergreifenden Berechtigungsübertragung finden Sie unter Aktivieren des kontenübergreifenden Zugriffs im Leitfaden zur Verwendung von IAM.

Wenn Sie die folgende Richtlinie zu einem Thema MyTopic im Konto 123456789012 hinzufügen würden, würden Sie dem Konto 111122223333 die Berechtigung erteilen, die Aktion zu diesem Thema auszuführen. `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"
  }]
}
```

Schritt 6: Amazon SNS SNS-Nachrichten an den HTTP/HTTPS-Endpunkt senden

Sie können eine Nachricht an die Abonnements eines Themas senden, indem Sie für das Thema veröffentlichen. Um zu einem Thema zu veröffentlichen, können Sie die Amazon-SNS-Konsole, den `sns-publish` CLI-Befehl oder die API-Aktion `Publish` verwenden.

Falls Sie Schritt 1 befolgt haben, sollte der bereitgestellte Code am Endpunkt die Benachrichtigung verarbeiten.

So veröffentlichen Sie über die Amazon-SNS-Konsole zu einem Thema

1. Melden Sie sich mit den Anmeldeinformationen des AWS-Konto oder des IAM-Benutzers mit der Berechtigung zum Veröffentlichen des Themas an AWS Management Console und öffnen Sie die Amazon SNS SNS-Konsole unter. https://console.aws.amazon.com/sns/
2. Wählen Sie im Navigationsbereich Topics (Themen) und dann ein Thema aus.
3. Wählen Sie die Schaltfläche Publish message (Nachricht veröffentlichen).
4. Geben Sie im Feld Subject (Betreff) einen Betreff ein (z. B. **`Testing publish to my endpoint`**).
5. Geben Sie im Feld Message Text (Nachrichtentext) Text ein (z. B. **`Hello world!`**) und wählen Sie Publish message (Nachricht veröffentlichen).

   Die folgende Meldung wird angezeigt: Ihre Nachricht wurde erfolgreich veröffentlicht.

Überprüfen der Signaturen von Amazon-SNS-Nachrichten

Amazon SNS verwendet Nachrichtensignaturen, um die Echtheit von Nachrichten zu bestätigen, die an Ihren HTTP-Endpunkt gesendet werden. Um die Nachrichtenintegrität zu gewährleisten und Spoofing zu verhindern, müssen Sie die Signatur überprüfen, bevor Sie Amazon SNS SNS-Nachrichten verarbeiten.

Wann sollten Sie Amazon SNS SNS-Signaturen überprüfen?

Sie sollten Amazon SNS SNS-Nachrichtensignaturen in den folgenden Szenarien überprüfen:

- Wenn Amazon SNS eine Benachrichtigung an Ihren HTTP (S) -Endpunkt sendet.

- Wenn Amazon SNS nach einem `Unsubscribe`API-Aufruf eine Bestätigungsnachricht an Ihren Endpunkt sendet. `Subscribe`

Amazon SNS unterstützt zwei Signaturversionen:

- SignatureVersion1 — Verwendet ein SHA1 Hash der Nachricht.
- SignatureVersion2 — Verwendet ein SHA256 Hash der Nachricht. Dies bietet eine höhere Sicherheit und ist die empfohlene Option.

Gehen Sie wie folgt vor, um die Signaturen von SNS-Nachrichten korrekt zu überprüfen:

- Rufen Sie das Signaturzertifikat immer über HTTPS ab, um unbefugte Abfangangriffe zu verhindern.
- Vergewissern Sie sich, dass das Zertifikat von Amazon SNS ausgestellt wurde.
- Vergewissern Sie sich, dass die Vertrauenskette des Zertifikats gültig ist.
- Das Zertifikat sollte von einer SNS-signierten URL stammen.
- Vertrauen Sie keinen in der Nachricht bereitgestellten Zertifikaten ohne Überprüfung.
- Lehnen Sie alle Nachrichten mit einem unerwarteten Ergebnis ab`TopicArn`, um Spoofing zu verhindern.
- Die AWS SDKs für Amazon SNS bieten eine integrierte Validierungslogik, wodurch das Risiko einer Fehlimplementierung reduziert wird.

Konfiguration der Nachrichtensignaturversion zu Amazon SNS SNS-Themen

Durch die Konfiguration der Nachrichtensignaturversion in Amazon SNS SNS-Themen können Sie die Sicherheit und Kompatibilität Ihres Nachrichtenüberprüfungsprozesses verbessern.

Wählen Sie zwischen `SignatureVersion` 1 (SHA1) und `SignatureVersion` 2 (SHA256), um den Hash-Algorithmus zu steuern, der zum Signieren von Nachrichten verwendet wird. Amazon SNS SNS-Themen sind standardmäßig auf `SignatureVersion` 1 eingestellt. Sie können diese Einstellung mithilfe der `SetTopicAttributes`API-Aktion konfigurieren.

Verwenden Sie das folgende Beispiel, um das Topic-Attribut `SignatureVersion` mithilfe von festzulegen AWS CLI:

```
aws sns set-topic-attributes \
```

```
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

Überprüfung der Signatur einer Amazon SNS SNS-Nachricht bei Verwendung von abfragebasierten HTTP-Anfragen

Die Überprüfung der Signatur einer Amazon SNS SNS-Nachricht bei Verwendung von HTTP-Abfragen gewährleistet die Authentizität und Integrität der Nachricht. Dieser Vorgang bestätigt, dass die Nachricht von Amazon SNS stammt und während der Übertragung nicht manipuliert wurde. Indem Sie die Nachricht analysieren, die richtige Zeichenfolge zum Signieren erstellen und die Signatur anhand eines vertrauenswürdigen öffentlichen Schlüssels validieren, schützen Sie Ihr System vor Spoofing und unbefugten Nachrichtenänderungen.

1. Extrahieren Sie Schlüssel-Wert-Paare aus dem JSON-Dokument im HTTP POST-Anforderungstext, der von Amazon SNS gesendet wurde. Diese Felder sind erforderlich, um die zu signierende Zeichenfolge zu erstellen.

   - `Message`
   - `Subject`(falls vorhanden)
   - `MessageId`
   - `Timestamp`
   - `TopicArn`
   - `Type`

   Zum Beispiel:

   ```
   MESSAGE_FILE="message.json"
   FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
   ```

   > **Note**
   >
   > Wenn ein Feld Escape-Zeichen enthält (z. B.`\n`), konvertieren Sie diese in ihre ursprüngliche Form, um eine exakte Übereinstimmung sicherzustellen.

2. Suchen Sie das `SigningCertURL` Feld in der Amazon SNS SNS-Nachricht. Dieses Zertifikat enthält den öffentlichen Schlüssel, der zur Überprüfung der Nachrichtensignatur benötigt wird. Zum Beispiel:

```
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
```

3. Stellen Sie sicher, `SigningCertURL` dass das von einer vertrauenswürdigen AWS Domain stammt (z. B. https://sns.us-east-1.amazonaws.com). Lehnen Sie aus Sicherheitsgründen alle URLs externen AWS Domains ab.
4. Laden Sie das X.509-Zertifikat von der angegebenen URL herunter. Zum Beispiel:

```
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
```

5. Extrahieren Sie den öffentlichen Schlüssel aus dem heruntergeladenen X.509-Zertifikat. Mit dem öffentlichen Schlüssel können Sie die Signatur der Nachricht entschlüsseln und ihre Integrität überprüfen. Zum Beispiel:

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. Verschiedene Nachrichtentypen erfordern zum Signieren unterschiedliche Schlüssel-Wert-Paare in der Zeichenfolge. Identifizieren Sie den Nachrichtentyp (`Type`Feld in der Amazon SNS SNS-Nachricht), um zu bestimmen, welche Schlüssel-Wert-Paare eingeschlossen werden sollen:
   - Benachrichtigungsnachricht — Beinhaltet `Message``MessageId`,, `Subject` (falls vorhanden),`Timestamp`, `TopicArn` und. `Type`
   - SubscriptionConfirmationoder UnsubscribeConfirmation Nachricht — Beinhaltet `Message` `MessageId``SubscribeURL`,`Timestamp`,`Token`,`TopicArn`, und`Type`.
7. Amazon SNS verlangt, dass die Zeichenfolge signiert wird, um eine strikte, feste Reihenfolge zur Überprüfung einzuhalten. Nur die ausdrücklich erforderlichen Felder müssen enthalten sein — es können keine zusätzlichen Felder hinzugefügt werden. Optionale Felder, wie z. B.`Subject`, dürfen nur enthalten sein, wenn sie in der Nachricht enthalten sind, und müssen genau an der Position erscheinen, die durch die erforderliche Feldreihenfolge definiert ist. Zum Beispiel:

```
KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
```

> **Important**
>
> Fügen Sie am Ende der Zeichenfolge kein Zeilenumbruchzeichen hinzu.

8. Ordnen Sie die Schlüssel-Wert-Paare in Byte-Sortierreihenfolge an (alphabetisch nach dem Schlüsselnamen).
9. Konstruieren Sie die zu signierende Zeichenfolge mithilfe des folgenden Formatbeispiels:

```
STRING_TO_SIGN=""
for FIELD in "${FIELDS[@]}"; do
    VALUE=$(jq -r --arg field "$FIELD" '.[$field]' "$MESSAGE_FILE")
    STRING_TO_SIGN+="$FIELD\n$VALUE"
    # Append a newline after each field except the last one
    if [[ "$FIELD" != "Type" ]]; then
        STRING_TO_SIGN+="\n"
    fi
done
```

Beispiel für eine Benachrichtigung:

```
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 Beispiel:

```
Message
Please confirm your subscription
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
```

```
https://sns.us-east-2.amazonaws.com/...
Timestamp
2024-01-01T00:00:00.000Z
Token
abc123...
TopicArn
arn:aws:sns:us-east-2:123456789012:MyTopic
Type
SubscriptionConfirmation
```

10. Das `Signature` Feld in der Nachricht ist Base64-codiert. Sie müssen es dekodieren, um seine rohe Binärform mit dem abgeleiteten Hash zu vergleichen. Zum Beispiel:

```
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
echo "$SIGNATURE" | base64 -d > signature.bin
```

11. Verwenden Sie das `SignatureVersion` Feld, um den Hash-Algorithmus auszuwählen:
    - Verwenden Sie für `SignatureVersion` 1 SHA1(zum Beispiel`-sha1`).
    - Verwenden Sie für `SignatureVersion` 2 SHA256(zum Beispiel`-sha256`).
12. Um die Echtheit der Amazon SNS SNS-Nachricht zu bestätigen, generieren Sie einen Hash der erstellten Zeichenfolge und überprüfen Sie die Signatur mithilfe des öffentlichen Schlüssels.

```
openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
 "$STRING_TO_SIGN"
```

    Wenn die Signatur gültig ist, ist `Verified OK` die Ausgabe. Andernfalls ist die Ausgabe`Verification Failure`.

Beispielskript mit Fehlerbehandlung

Das folgende Beispielskript automatisiert den Überprüfungsprozess:

```
#!/bin/bash

# Path to the local message file
MESSAGE_FILE="message.json"

# Extract the SigningCertURL and Signature from the message
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
```

```
# Fetch the X.509 certificate
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem

# Extract the public key from the certificate
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem

# Define the fields to include in the string to sign
FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")

# Initialize the string to sign
STRING_TO_SIGN=""

# Iterate over the fields to construct the string to sign
for FIELD in "${FIELDS[@]}"; do
    VALUE=$(jq -r --arg field "$FIELD" '.[$field]' "$MESSAGE_FILE")
    STRING_TO_SIGN+="$FIELD\n$VALUE"
    # Append a newline after each field except the last one
    if [[ "$FIELD" != "Type" ]]; then
        STRING_TO_SIGN+="\n"
    fi
done

# Verify the signature
echo -e "$STRING_TO_SIGN" | openssl dgst -sha256 -verify public_key.pem -signature
 <(echo "$SIGNATURE" | base64 -d)
```

Analysieren von Amazon SNS SNS-Nachrichtenformaten

Wenn Amazon SNS Nachrichten an HTTP/HTTPS-Endpunkte sendet, enthalten sie sowohl HTTP-Header als auch einen JSON-Nachrichtentext. Diese Nachrichten folgen einem strukturierten Format, das Metadaten wie Nachrichtentyp, Themen-ARN, Zeitstempel und digitale Signaturen enthält. Durch die korrekte Analyse von Amazon SNS SNS-Nachrichten können Sie feststellen, ob es sich bei einer Nachricht um eine Abonnementbestätigung, Benachrichtigung oder Abmeldebestätigung handelt, relevante Daten extrahieren und die Echtheit mithilfe der Signaturvalidierung überprüfen.

HTTP/HTTPS-Header

Sobald Amazon SNS eine Abonnementbestätigungs-, Benachrichtigungs- oder Abmeldebestätigungsnachricht an HTTP/HTTPS-Endpunkte sendet, sendet es eine POST-Nachricht mit einer Reihe an Amazon–SNS-spezifischen Header-Werten. Sie können Header-Werte für Aufgaben wie das Identifizieren des Nachrichtentyps verwenden, ohne den JSON-

Nachrichtentext zu analysieren, um den `Type`Wert zu lesen. Standardmäßig sendet Amazon SNS alle Benachrichtigungen an HTTP/S-Endpunkte, wobei `Content-Type` auf `text/plain; charset=UTF-8` festgelegt ist. Informationen zur Auswahl eines `Content-Type`, der nicht text/plain (Standard) ist, finden Sie unter `headerContentType` in Erstellung HTTP/S einer Lieferrichtlinie.

`x-amz-sns-message-type`

Der Nachrichtentyp. Die möglichen Werte sind `SubscriptionConfirmation`, `Notification` und `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Benachrichtigung, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`x-amz-sns-topic-arn`

Amazon Resource Name (ARN) des Themas, in dem die Nachricht veröffentlicht wurde.

`x-amz-sns-subscription-arn`

Die ARN für das Abonnement für diesen Endpunkt.

Der folgende „HTTP POST“-Header ist ein Beispiel für einen Header für eine `Notification`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
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
```

JSON-Format für die HTTP/HTTPS-Abonnement-Bestätigung

Nachdem Sie einen HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS Endpunkt abonniert haben. Diese Nachricht enthält einen

`SubscribeURL`-Wert, den Sie besuchen müssen, um das Abonnement zu bestätigen (alternativ können Sie den `Token`-Wert mit der Aktion `ConfirmSubscription` verwenden).

> **Note**
>
> Amazon SNS sendet keine Benachrichtigungen an diesen Endpunkt, solange das Abonnement nicht bestätigt wurde

Bei der Abonnement-Bestätigung handelt es sich um eine POST-Nachricht mit einem Nachrichtentext, der ein JSON-Dokument mit den folgenden Name-Wert-Paaren enthält.

`Type`

Der Nachrichtentyp. Für eine Abonnement-Bestätigung ist der Typ `SubscriptionConfirmation`.

`MessageId`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Nachricht, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`Token`

Ein Wert, den Sie mit der Aktion `ConfirmSubscription` verwenden können, um das Abonnement zu bestätigen. Sie können aber auch einfach nur den Parameter besuchen `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) des Themas, für das dieser Endpunkt abonniert wurde.

`Message`

Eine Zeichenfolge, die die Nachricht beschreibt. Bei einer Abonnement-Bestätigung sieht diese Zeichenfolge wie folgt aus:

```
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`

Die URL, die Sie besuchen müssen, um das Abonnement zu bestätigen. Alternativ können Sie auch den Parameter `Token` mit der Aktion `ConfirmSubscription`verwenden, um das Abonnement zu bestätigen.

`Timestamp`

Die Uhrzeit (GMT), zu der die Abonnement-Bestätigung gesendet wurde.

`SignatureVersion`

Version der verwendeten Amazon-SNS-Signatur.

- Wenn `SignatureVersion` 1 lautet, ist `Signature` eine Base64-kodierte `SHA1withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.
- Wenn `SignatureVersion` 2 lautet, ist `Signature` eine Base64-kodierte `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`Signature`

Eine Base64-kodierte `SHA1withRSA`- oder `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`SigningCertURL`

Die URL, die den Zugriff auf das Zertifikat ermöglicht, das zum Signieren der Nachricht verwendet wurde.

Die folgende „HTTP POST“-Nachricht ist ein Beispiel für eine `SubscriptionConfirmation`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
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"
}
```

JSON-Format für eine HTTP/HTTPS-Benachrichtigung

Wenn Amazon SNS eine Benachrichtigung an einen abonnierten HTTP- oder HTTPS-Endpunkt sendet, enthält der Nachrichtentext der POST-Nachricht, die an den Endpunkt gesendet wurde, ein JSON-Dokument mit den folgenden Name-Wert-Paaren.

`Type`

Der Nachrichtentyp. Für eine Bestätigung ist der Typ `Notification`.

`MessageId`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Benachrichtigung, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`TopicArn`

Amazon Resource Name (ARN) des Themas, in dem die Nachricht veröffentlicht wurde.

`Subject`

Der Parameter `Subject`, der bei der Veröffentlichung der Benachrichtigung im Thema angegeben wurde.

> **Note**
>
> Dieser Parameter ist optional. Wenn kein `Subject` angegeben wurde, wird dieses Name-Wert-Paar nicht in diesem JSON-Dokument angezeigt.

`Message`

Der Nachrichtenwert `Message`, der bei der Veröffentlichung der Benachrichtigung im Thema angegeben wurde.

`Timestamp`

Uhrzeit (GMT), zu der die Benachrichtigung veröffentlicht wurde.

`SignatureVersion`

Version der verwendeten Amazon-SNS-Signatur.

- Wenn `SignatureVersion` 1 lautet, ist `Signature` eine Base64-kodierte `SHA1withRSA`-Signatur der Werte `Message`, `MessageId`, `Subject` (sofern vorhanden), `Type`, `Timestamp` und `TopicArn`.
- Wenn `SignatureVersion` 2 lautet, ist `Signature` eine Base64-kodierte `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Subject` (sofern vorhanden), `Type`, `Timestamp` und `TopicArn`.

`Signature`

Eine Base64-kodierte `SHA1withRSA`- oder `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Subject` (sofern vorhanden), `Type`, `Timestamp` und `TopicArn`.

`SigningCertURL`

Die URL, die den Zugriff auf das Zertifikat ermöglicht, das zum Signieren der Nachricht verwendet wurde.

`UnsubscribeURL`

Eine URL, die Sie verwenden können, um den Endpunkt aus diesem Thema abzumelden. Wenn Sie diese URL besuchen, meldet Amazon SNS den Endpunkt ab und es werden keine weiteren Benachrichtigungen an diesen Endpunkt gesendet.

Die folgende „HTTP POST“-Nachricht ist ein Beispiel für eine `Notification`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
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"
  }
```

JSON-Format für die HTTP/HTTPS-Abonnement-Abmeldung

Nachdem ein HTTP/HTTPS-Endpunkt aus einem Thema abgemeldet wurde, sendet Amazon SNS eine Abmelde-Bestätigung an den Endpunkt.

Bei der Abmelde-Bestätigung handelt es sich um eine POST-Nachricht mit einem Nachrichtentext, der ein JSON-Dokument mit den folgenden Name-Wert-Paaren enthält.

`Type`

Der Nachrichtentyp. Für eine Abmelde-Bestätigung ist der Typ `UnsubscribeConfirmation`.

`MessageId`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Nachricht, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`Token`

Ein Wert, den Sie mit der Aktion `ConfirmSubscription` verwenden können, um das Abonnement erneut zu bestätigen. Sie können aber auch einfach nur den Parameter besuchen `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) des Themas, von dem dieser Endpunkt abgemeldet wurde.

`Message`

Eine Zeichenfolge, die die Nachricht beschreibt. Bei einer Abmelde-Bestätigung sieht diese Zeichenfolge wie folgt aus:

```
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`

Die URL, die Sie besuchen müssen, um das Abonnement erneut zu bestätigen. Alternativ können Sie auch den Parameter `Token` mit der Aktion `ConfirmSubscription`verwenden, um das Abonnement erneut zu bestätigen.

`Timestamp`

Uhrzeit (GMT), zu der die Abmelde-Bestätigung gesendet wurde.

`SignatureVersion`

Version der verwendeten Amazon-SNS-Signatur.

- Wenn `SignatureVersion` 1 lautet, ist `Signature` eine Base64-kodierte `SHA1withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.
- Wenn `SignatureVersion` 2 lautet, ist `Signature` eine Base64-kodierte `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`Signature`

Eine Base64-kodierte `SHA1withRSA`- oder `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`SigningCertURL`

Die URL, die den Zugriff auf das Zertifikat ermöglicht, das zum Signieren der Nachricht verwendet wurde.

Die folgende „HTTP POST“-Nachricht ist ein Beispiel für eine `UnsubscribeConfirmation`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
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"
```

```
}
```

SetSubscriptionAttributesJSON-Format für Lieferrichtlinien

Wenn Sie eine Anforderung an die Aktion `SetSubscriptionAttributes` senden und den Parameter `AttributeName` auf den Wert `DeliveryPolicy` festlegen, muss der Wert des Parameters `AttributeValue` ein gültiges JSON-Objekt sein. Im folgenden Beispiel wird die Zustellungsrichtlinie auf insgesamt 5 Wiederholversuche festgelegt.

```
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}}
...
```

Verwenden Sie das folgende JSON-Format für den Wert des Parameters `AttributeValue`.

```
{
    "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"
    }
}
```

Weitere Informationen zu der `SetSubscriptionAttribute` Aktion finden Sie SetSubscriptionAttributesin der Amazon Simple Notification Service API-Referenz. Weitere Hinweise zu den unterstützten HTTP-Inhaltstyp-Headern finden Sie unter Erstellung HTTP/S einer Lieferrichtlinie.

SetTopicAttributes JSON-Format für Lieferrichtlinien

Wenn Sie eine Anforderung an die Aktion `SetTopicAttributes` senden und den Parameter `AttributeName` auf den Wert `DeliveryPolicy` festlegen, muss der Wert des Parameters `AttributeValue` ein gültiges JSON-Objekt sein. Im folgenden Beispiel wird die Zustellungsrichtlinie auf insgesamt 5 Wiederholversuche festgelegt.

```
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}}}
...
```

Verwenden Sie das folgende JSON-Format für den Wert des Parameters `AttributeValue`.

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

Weitere Informationen zu der `SetTopicAttribute` Aktion finden Sie SetTopicAttributesin der Amazon Simple Notification Service API-Referenz. Weitere Hinweise zu den unterstützten HTTP-Inhaltstyp-Headern finden Sie unter Erstellung HTTP/S einer Lieferrichtlinie.

Fanout von Amazon SNS SNS-Ereignissen in AWS Event Fork-Pipelines

Für die Archivierung und Analyse von Ereignissen empfiehlt Amazon SNS jetzt, die native Integration mit Amazon Data Firehose zu verwenden. Sie können Firehose-Lieferstreams für SNS-Themen abonnieren, sodass Sie Benachrichtigungen an Archivierungs- und Analyseendpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden können. Die Verwendung von Amazon SNS mit Firehose Delivery Streams ist eine vollständig verwaltete und codelose Lösung, für die Sie keine Funktionen verwenden müssen. AWS Lambda Weitere Informationen finden Sie unter Lieferstreams von Fanout nach Firehose.

Sie können Amazon SNS zum Entwickeln ereignisgesteuerter Anwendungen verwenden, die Abonnenten-Services für die automatische Ausführung von Aufgaben als Antwort auf von Herausgeber-Services ausgelöste Ereignisse verwenden. Dieses Architekturmuster kann eine bessere Wiederverwendbarkeit, Interoperabilität und Skalierbarkeit von Services unterstützen. Es kann jedoch arbeitsaufwändig sein, die Verarbeitung von Ereignissen auf Pipelines aufzuteilen, die gemeinsame Anforderungen an die Ereignisbehandlung erfüllen, wie Speicherung, Backup, Suche, Analytik und Replay von Events.

Um die Entwicklung Ihrer ereignisgesteuerten Anwendungen zu beschleunigen, können Sie Event-Handling-Pipelines — unterstützt von AWS Event Fork Pipelines — für Amazon SNS SNS-Themen abonnieren. AWS Event Fork Pipelines ist eine Suite verschachtelter Open-Source-Anwendungen, die auf dem AWS Serverless Application Model (AWS SAM) basieren und die Sie direkt aus der AWS Event Fork Pipelines Suite (wählen Sie Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) in Ihrem Konto bereitstellen können. AWS

Einen Anwendungsfall für AWS Event Fork Pipelines finden Sie unter. Bereitstellen und Testen der Beispielanwendung Amazon SNS Event Fork Pipelines

Themen

- So funktionieren AWS Event Fork-Pipelines
- Bereitstellung von AWS Event Fork-Pipelines
- Bereitstellen und Testen der Beispielanwendung Amazon SNS Event Fork Pipelines
- AWS Event Fork-Pipelines für ein Amazon SNS SNS-Thema abonnieren

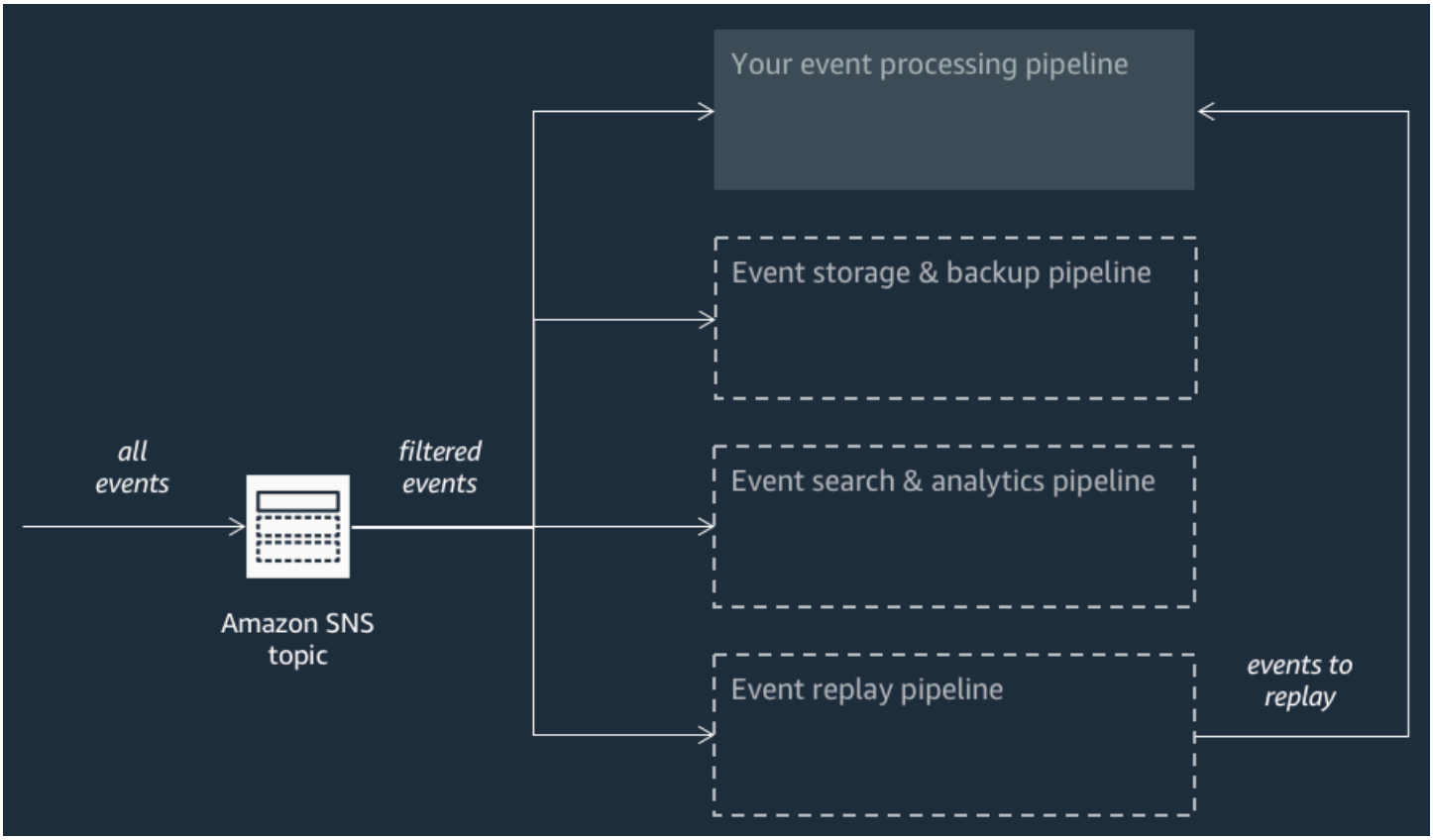
So funktionieren AWS Event Fork-Pipelines

AWS Event Fork Pipelines ist ein serverloses Entwurfsmuster. Es handelt sich jedoch auch um eine Suite verschachtelter serverloser Anwendungen, die auf AWS SAM basieren (die Sie direkt vom AWS Serverless Application Repository (AWS SAR) auf Ihrem Computer bereitstellen können, um Ihre AWS-Konto ereignisgesteuerten Plattformen zu erweitern). Sie können diese verschachtelten Anwendungen einzeln bereitstellen wie von Ihrer Architektur gefordert.

Themen

- Die Pipeline für die Speicherung und Sicherung von Ereignissen
- Die Pipeline für die Suche und Analyse von Ereignissen
- Die Pipeline für das Replay von Events

Das folgende Diagramm zeigt eine AWS Event Fork Pipelines-Anwendung, die durch drei verschachtelte Anwendungen ergänzt wird. Sie können jede der Pipelines aus der AWS Event Fork Pipelines Suite unabhängig voneinander auf der AWS SAR bereitstellen, wie es Ihre Architektur erfordert.



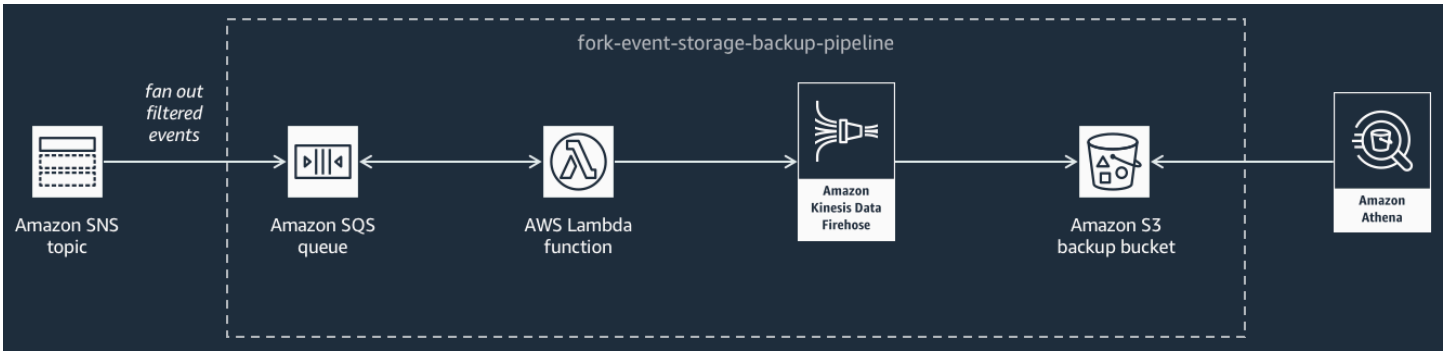
Jede Pipeline hat dasselbe Amazon SNS-Thema abonniert, um Ereignisse parallel verarbeiten zu können, wenn diese Ereignisse zum Thema veröffentlicht werden. Jede Pipeline ist unabhängig und Sie können eine eigene Abonnementfilterrichtlinie festlegen. So ist es möglich, dass eine Pipeline nur eine Teilmenge der Ereignisse verarbeitet - die von Interesse (anstelle aller zum Thema veröffentlichter Ereignisse).

> **Note**
>
> Da Sie die drei AWS Event Fork-Pipelines neben Ihren regulären Event-Verarbeitungspipelines platzieren (möglicherweise haben Sie Ihr Amazon SNS SNS-Thema bereits abonniert), müssen Sie keinen Teil Ihres aktuellen Nachrichtenherausgebers ändern, um die AWS Event Fork-Pipelines in Ihren bestehenden Workloads zu nutzen.

Die Pipeline für die Speicherung und Sicherung von Ereignissen

Das folgende Diagramm zeigt die Pipeline für die Speicherung und Sicherung von Ereignissen. Sie können für diese Pipeline ein Abonnement Ihres Amazon SNS-Themas erstellen, um die Ereignisse in Ihrem System automatisch zu sichern.

Diese Pipeline besteht aus einer Amazon SQS SQS-Warteschlange, die die vom Amazon SNS SNS-Thema übermittelten Ereignisse zwischenspeichert, einer AWS Lambda Funktion, die automatisch nach diesen Ereignissen in der Warteschlange abfragt und sie in einen Stream überträgt, und einem Amazon S3 S3-Bucket, der die vom Stream geladenen Ereignisse dauerhaft sichert.

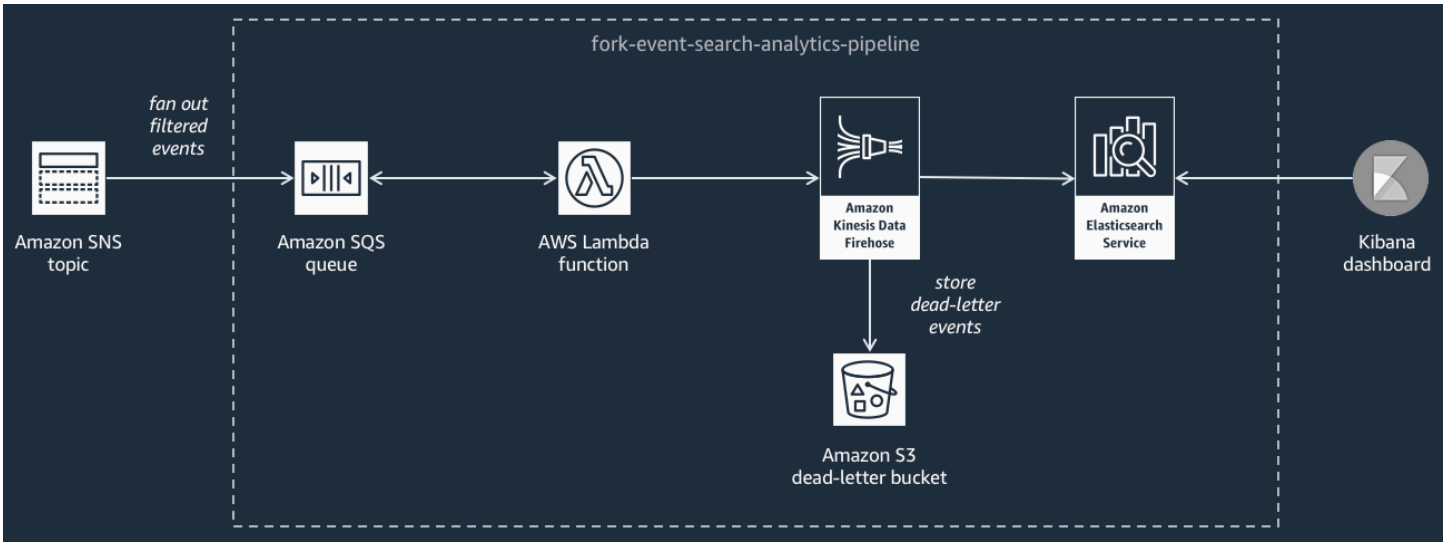


Um das Verhalten Ihres Firehose-Streams zu optimieren, können Sie ihn so konfigurieren, dass Ihre Ereignisse gepuffert, transformiert und komprimiert werden, bevor sie in den Bucket geladen werden. Während die Ereignisse geladen werden, können Sie Amazon Athena verwenden, um im Bucket SQL-Standardabfragen zu stellen. Sie können die Pipeline auch für die Wiederverwendung eines vorhandenen Amazon S3-Buckets konfigurieren oder einen neuen Bucket erstellen.

Die Pipeline für die Suche und Analyse von Ereignissen

Das folgende Diagramm zeigt die Pipeline für die Suche und Analyse von Ereignissen. Sie können für diese Pipeline ein Abonnement Ihres Amazon SNS-Themas erstellen, um die Ereignisse in Ihrem System in einer Suchdomäne zu indizieren und anschließend zu analysieren.

Diese Pipeline besteht aus einer Amazon SQS SQS-Warteschlange, die die vom Amazon SNS SNS-Thema übermittelten Ereignisse zwischenspeichert, einer AWS Lambda Funktion, die Ereignisse aus der Warteschlange abfragt und in einen Stream überträgt, einer Amazon OpenSearch Service-Domain, die die vom Firehose-Stream geladenen Ereignisse indexiert, und einem Amazon S3 S3-Bucket, der die Ereignisse speichert, die nicht in der Such-Domain indexiert werden können.



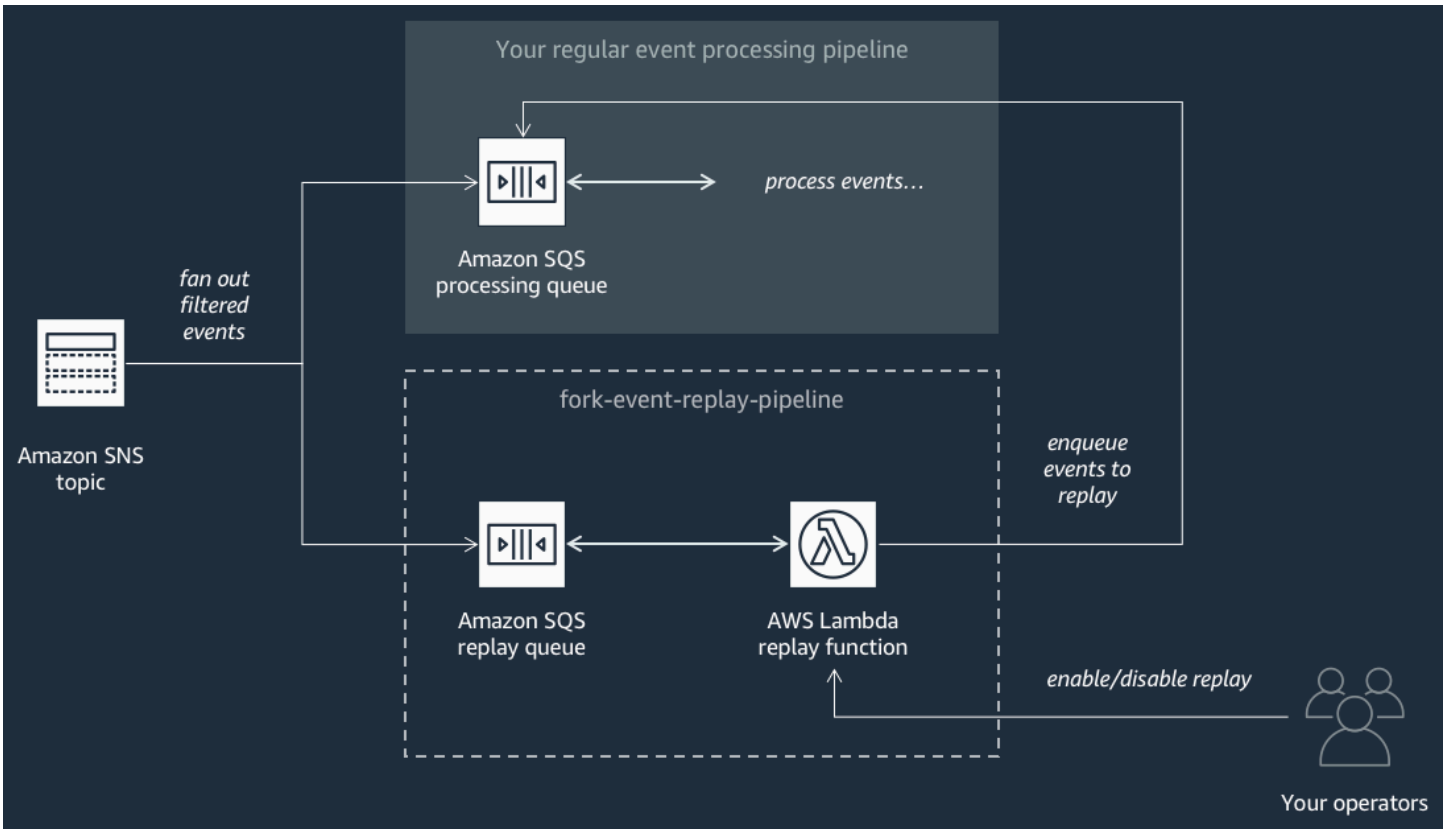
Um Ihren Firehose-Stream in Bezug auf die Pufferung, Transformierung und Komprimierung von Ereignissen zu optimieren, können Sie diese Pipeline konfigurieren.

Sie können auch konfigurieren, ob die Pipeline eine bestehende OpenSearch Domain in Ihrer wiederverwenden AWS-Konto oder eine neue für Sie erstellen soll. Während der Indizierung von Ereignissen in der Suchdomäne können Sie mit Kibana Analysen für Ihre Ereignisse ausführen und visuelle Dashboards in Echtzeit aktualisieren.

Die Pipeline für das Replay von Events

Das folgende Diagramm zeigt die Pipeline für das Replay von Events. Um die Ereignisse aufzuzeichnen, die von Ihrem System in den letzten 14 Tagen verarbeitet wurden (beispielsweise für den Fall, dass Ihre Plattform nach einem Fehler wiederhergestellt werden muss), können Sie für diese Pipeline ein Abonnement Ihres Amazon SNS-Themas erstellen und anschließend die Ereignisse neu verarbeiten.

Diese Pipeline besteht aus einer Amazon SQS SQS-Warteschlange, die die vom Amazon SNS SNS-Thema übermittelten Ereignisse zwischenspeichert, und einer AWS Lambda Funktion, die Ereignisse aus der Warteschlange abfragt und sie in Ihre reguläre Ereignisverarbeitungspipeline zurückleitet, die auch Ihr Thema abonniert hat.



> **Note**
>
> Standardmäßig ist die Funktion für das erneute Abspielen deaktiviert und schiebt Ihre Ereignisse nicht erneut in die reguläre Pipeline. Wenn Sie Ereignisse erneut verarbeiten müssen, müssen Sie die Amazon-SQS-Warteschlange für das erneute Abspielen als Ereignisquelle für die AWS Lambda -Funktion für das erneute Abspielen aktivieren.

Bereitstellung von AWS Event Fork-Pipelines

Die AWS Event Fork Pipelines Suite (wählen Sie Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) ist als Gruppe von öffentlichen Anwendungen in der verfügbar AWS Serverless Application Repository, von wo aus Sie sie mithilfe der AWS Lambda Konsole manuell bereitstellen und testen können. Informationen zur Bereitstellung von Pipelines mithilfe der

AWS Lambda Konsole finden Sie unter. AWS Event Fork-Pipelines für ein Amazon SNS SNS-Thema abonnieren

In einem Produktionsszenario empfehlen wir, AWS Event Fork-Pipelines in die SAM-Vorlage Ihrer gesamten Anwendung einzubetten. AWS Mit der Funktion für verschachtelte Anwendungen können Sie dies tun, indem Sie die Ressource `AWS::Serverless::Application` zu Ihrer AWS SAM-Vorlage hinzufügen und dabei auf die AWS SAR `ApplicationId` und die der verschachtelten Anwendung verweisen. `SemanticVersion`

Sie können beispielsweise die Event Storage and Backup Pipeline als verschachtelte Anwendung verwenden, indem Sie dem `Resources` Abschnitt Ihrer SAM-Vorlage den folgenden YAML-Snippet hinzufügen. AWS

```
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
```

Wenn Sie Parameterwerte angeben, können Sie AWS CloudFormation systeminterne Funktionen verwenden, um auf andere Ressourcen in Ihrer Vorlage zu verweisen. Im obigen YAML-Snippet verweist der `TopicArn` Parameter beispielsweise auf die `AWS::SNS::Topic` Ressource, die an anderer Stelle in der Vorlage `MySNSTopic` definiert ist. AWS SAM Weitere Informationen finden Sie in der Referenz für intrinsische Funktion im AWS CloudFormation -Benutzerhandbuch.

> **Note**
>
> Die AWS Lambda Konsolenseite für Ihre AWS SAR-Anwendung enthält die Schaltfläche Als SAM-Ressource kopieren, mit der das für die Verschachtelung einer AWS SAR-Anwendung erforderliche YAML in die Zwischenablage kopiert wird.

Bereitstellen und Testen der Beispielanwendung Amazon SNS Event Fork Pipelines

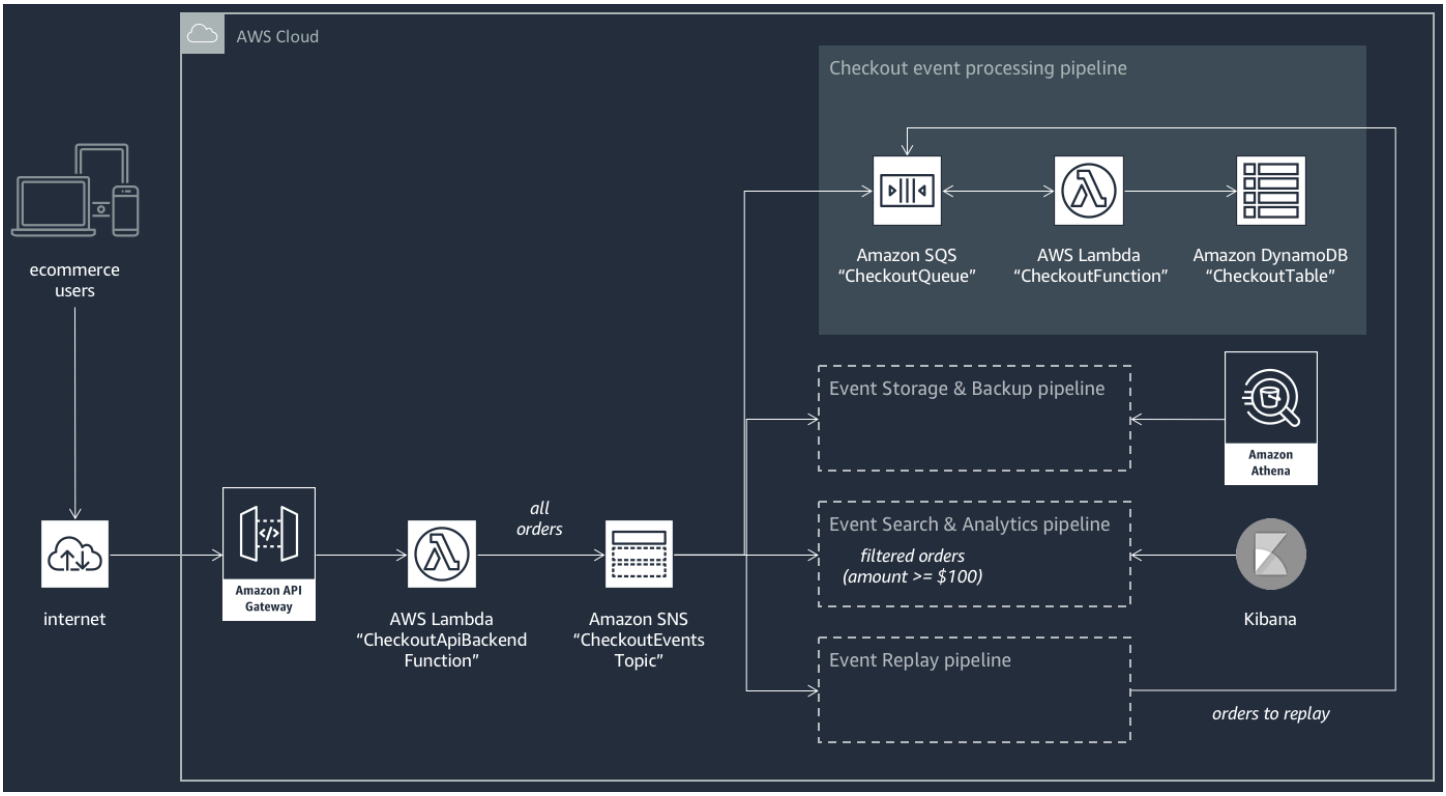
Um die Entwicklung Ihrer ereignisgesteuerten Anwendungen zu beschleunigen, können Sie Event-Handling-Pipelines — unterstützt von AWS Event Fork Pipelines — für Amazon SNS SNS-Themen abonnieren. AWS Event Fork Pipelines ist eine Suite verschachtelter Open-Source-Anwendungen, die auf dem AWS Serverless Application Model (AWS SAM) basieren und die Sie direkt aus der AWS Event Fork Pipelines Suite (wählen Sie Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) in Ihrem Konto bereitstellen können. AWS Weitere Informationen finden Sie unter So funktionieren AWS Event Fork-Pipelines.

Auf dieser Seite wird gezeigt, wie Sie die Beispielanwendung Event Fork AWS Management Console Pipelines bereitstellen und testen können. AWS

> **Important**
>
> Um zu vermeiden, dass Ihnen nach Abschluss der Bereitstellung der AWS Event Fork Pipelines-Beispielanwendung unerwünschte Kosten entstehen, löschen Sie den zugehörigen Stack. AWS CloudFormation Weitere Informationen finden Sie unter Löschen eines Stacks in der AWS CloudFormation -Konsole im AWS CloudFormation -Benutzerhandbuch.

AWS Anwendungsbeispiel für Event Fork Pipelines

Das folgende Szenario beschreibt eine ereignisgesteuerte, serverlose E-Commerce-Anwendung, die Event Fork-Pipelines verwendet AWS . Sie können diese E-Commerce-Beispielanwendung in der verwenden AWS Serverless Application Repository und sie dann AWS-Konto mithilfe der AWS Lambda Konsole bereitstellen, wo Sie sie testen und ihren Quellcode untersuchen können. GitHub



Diese E-Commerce-Anwendung nimmt Bestellungen von Käufern über eine RESTful API entgegen, die von API Gateway gehostet und von der AWS Lambda Funktion unterstützt wird`CheckoutApiBackendFunction`. Diese Funktion veröffentlicht alle empfangenen Aufträge zu einem Amazon SNS-Thema mit dem Namen `CheckoutEventsTopic`, das seinerseits die Aufträge an vier unterschiedliche Pipelines weitergibt.

Die erste Pipeline ist die reguläre Checkout-Verarbeitungs-Pipeline, die vom Eigentümer der e-Commerce-Anwendung entworfen und implementiert wurde. Diese Pipeline hat die Amazon SQS SQS-Warteschlange`CheckoutQueue`, die alle eingegangenen Bestellungen zwischenspeichert, eine AWS Lambda Funktion namens, `CheckoutFunction` die die Warteschlange abfragt, um diese Bestellungen zu verarbeiten, und die DynamoDB-Tabelle, in der alle aufgegebenen Bestellungen `CheckoutTable` sicher gespeichert werden.

Event AWS Fork-Pipelines anwenden

Die Komponenten der e-Commerce-Anwendung kümmern sich um die zentrale Geschäftslogik. Der Eigentümer der e-Commerce-Anwendung muss sich aber auch um Folgendes kümmern:

- Compliance – sichere, komprimierte Backups, die im Ruhezustand verschlüsselt sind und die Bereinigung sensibler Informationen

- Ausfallsicherheit – Wiederholung aktueller Bestellungen im Falle einer Unterbrechung des Fulfillment-Prozesses
- Durchsuchbarkeit – Ausführen von Analytik und Generierung von Metriken für aufgegebene Aufträge

Anstatt diese Ereignisverarbeitungslogik zu implementieren, kann der Anwendungsbesitzer AWS Event Fork-Pipelines für das `CheckoutEventsTopic` Amazon SNS SNS-Thema abonnieren

- Die Pipeline für die Speicherung und Sicherung von Ereignissen ist für die Transformation von Daten konfiguriert, sodass Kreditkartendaten entfernt, Daten 60 Sekunden gepuffert, Daten mit GZIP komprimiert und mit dem verwalteten Standardschlüssel für Amazon S3 verschlüsselt werden. Dieser Schlüssel wird von AWS Key Management Service ()AWS KMS verwaltet AWS und unterstützt.

  Weitere Informationen finden Sie unter Wählen Sie Amazon S3 für Ihr Ziel, Amazon Data Firehose Data Transformation und Configure Settings im Amazon Data Firehose Developer Guide.
- Die Pipeline für die Suche und Analyse von Ereignissen ist mit einer Index-Wiederholungsdauer von 30 Sekunden konfiguriert, sowie mit einem Bucket für die Speicherung von Aufträgen, die in der Suchdomäne nicht indiziert werden können, und einer Filterrichtlinie zur Einschränkung der indizierten Aufträge.

  Weitere Informationen finden Sie unter Wählen Sie den OpenSearch Service für Ihr Ziel im Amazon Data Firehose Developer Guide.
- Die Pipeline für das Replay von Events ist mit dem Amazon-SQS-Warteschlangenteil der normalen Auftragsverarbeitungspipeline konfiguriert, die von Eigentümer der e-Commerce-Anwendung entworfen und implementiert wurde.

  Weitere Informationen finden Sie unter Name der Warteschlange und URL im Amazon Simple Queue Service Leitfaden für Entwickler.

Die folgende JSON-Filterrichtlinie ist in der Konfiguration für die Ereignissuche- und Analysepipeline eingestellt. Sie gleicht nur eingehende Aufträge mit einem Gesamtbetrag von mindestens 100 USD ab. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

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

Mithilfe des AWS Event Fork-Pipelines-Musters kann der Eigentümer der E-Commerce-Anwendung den Entwicklungsaufwand vermeiden, der häufig mit der Programmierung einer undifferenzierten Logik für die Ereignisbehandlung verbunden ist. Stattdessen kann sie AWS Event Fork-Pipelines direkt von ihr aus bereitstellen. AWS Serverless Application Repository AWS-Konto

Schritt 1: Bereitstellen der Amazon SNS SNS-Beispielanwendung

1. Melden Sie sich an der AWS Lambda -Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:
   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).
   b. Suchen Sie nach `fork-example-ecommerce-checkout-api` und wählen Sie dann die Anwendung aus.
4. Gehen Sie auf der fork-example-ecommerce-checkout-api-Seite wie folgt vor:
   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `fork-example-ecommerce-my-app`).

      > **Note**
      > - Um Ihre Ressourcen später leichter finden zu können, behalten Sie das Präfix `fork-example-ecommerce`.
      > - Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation Stack (anstatt einen neuen zu erstellen).

   b. (Optional) Geben Sie eine der folgenden LogLevelEinstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:
      - `DEBUG`
      - `ERROR`

- `INFO` (Standard)
- `WARNING`

5. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann unten auf der Seite Deploy (Bereitstellen) aus.

Auf der *`my-app`* Seite Bereitstellungsstatus für fork-example-ecommerce — zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

AWS CloudFormation Beginnt im Abschnitt Ressourcen mit der Erstellung des Stacks und zeigt den CREATE_IN_PROGRESS-Status für jede Ressource an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

> **Note**
>
> Es kann 20 bis 30 Minuten dauern, bis alle Ressourcen bereitgestellt sind.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Schritt 2: Ausführen der mit SNS verknüpften Beispielanwendung

1. Wählen Sie in der AWS Lambda Konsole im Navigationsbereich Applications aus.
2. Suchen Sie auf der Seite Applications (Anwendungen) im Suchfeld nach `serverlessrepo-fork-example-ecommerce-`*`my-app`* und wählen Sie dann die Anwendung aus.
3. Gehen Sie im Abschnitt Resources (Ressourcen) wie folgt vor:

   a. Um die Ressource zu finden, deren Typ ist ApiGatewayRestApi, sortieren Sie die Ressourcen `ServerlessRestApi` beispielsweise nach Typ und erweitern Sie dann die Ressource.

   b. Es werden zwei verschachtelte Ressourcen der Typen ApiGatewayDeployment und ApiGatewayStage angezeigt.

   c. Kopieren Sie den Link Prod API endpoint (Prod-API-Endpunkt) und fügen Sie diesem `/checkout` an, z. B.:

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

4. Kopieren Sie den folgenden JSON-Code in eine Datei mit dem Namen `test_event.json`.

```
{
   "id": 15311,
   "date": "2019-03-25T23:41:11-08:00",
   "status": "confirmed",
   "customer": {
      "id": 65144,
  "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. Um eine HTTPS-Anfrage an Ihren API-Endpunkt zu senden, übergeben Sie die Nutzlast des Beispielereignisses als Eingabe, indem Sie einen `curl`-Befehl ausführen, zum Beispiel:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

   Die API gibt die folgende leere Antwort unter Angabe einer erfolgreichen Ausführung zurück:

```
{ }
```

Schritt 3: Überprüfen der Leistung der Amazon SNS SNS-Anwendung und -Pipeline

Schritt 1: Überprüfen der Ausführung der Muster-Checkout-Pipeline

1. Melden Sie sich bei der Amazon DynamoDB-Konsole an.
2. Wählen Sie im Navigationsbereich Tables (Tabellen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie `CheckoutTable` aus.
4. Wählen Sie auf der Tabellendetailseite Items (Elemente) und dann das erstellte Element aus.

   Die gespeicherten Attribute werden angezeigt.

Schritt 2: Überprüfen der Ausführung der Ereignisspeicher- und Backup-Pipeline

1. Melden Sie sich bei der Amazon S3-Konsole an.
2. Wählen Sie im Navigationsbereich Buckets (Buckets) aus.
3. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie dann `CheckoutBucket`.
4. Navigieren Sie durch die Verzeichnishierarchie, bis Sie eine Datei mit der Erweiterung finden `.gz`.
5. Um die Datei herunterzuladen, wählen Sie Actions (Aktionen), Open (Öffnen).
6. Die Pipeline ist mit einer Lambda-Funktion konfiguriert, die Kreditkarteninformationen aus Gründen der Compliance schützt.

   Um zu überprüfen, ob die gespeicherte JSON-Nutzlast Kreditkarteninformationen enthält, dekomprimieren Sie die Datei.

Schritt 3: Überprüfen der Ausführung der Event-Such- und Analysepipeline

1. Melden Sie sich bei der OpenSearch Servicekonsole an.
2. Wählen Sie im Navigationsbereich in My domains (Meine Domains) die Domäne mit dem Präfix `serverl-analyt` aus.
3. Die Pipeline ist mit einer Amazon SNS-Abonnement-Filterrichtlinie konfiguriert, die eine numerische Abgleichbedingung einrichtet.

   Um zu überprüfen, ob das Ereignis indexiert ist, weil es sich auf eine Bestellung bezieht, deren Wert über 100 USD liegt, wählen Sie auf der ***abcdefgh1ijk*** Seite serverl-analyst die Option Indices, checkout_events aus.

Schritt 4: Überprüfen der Ausführung der Event-Replay-Pipeline

1. Melden Sie sich bei der Amazon SQS-Konsole an.
2. Suchen Sie in der Liste der Warteschlangen nach `serverlessrepo-fork-example` und wählen Sie `ReplayQueue` aus.
3. Wählen Sie Nachrichten senden und empfangen.
4. Wählen Sie im ***123ABCD4E5F6*** Dialogfeld Nachrichten senden und empfangen in fork-example-ecommerce -***my-app***... ReplayP- ReplayQueue - die Option Umfrage für Nachrichten aus.

5. Um zu prüfen, ob sich das Ereignis in der Warteschlange befindet, wählen Sie More Details (Weitere Einzelheiten) neben der Nachricht, die in der Warteschlange angezeigt wird.

Schritt 4: Simulieren eines Problems und Wiederholung der Ereignisse zur Wiederherstellung

Schritt 1: Aktivieren Sie das simulierte Problem und senden Sie eine zweite API-Anfrage

1. Melden Sie sich an der AWS Lambda -Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example`und wählen Sie `CheckoutFunction` aus.
4. Auf dem fork-example-ecommerce- ***my-app*** - CheckoutFunction -***ABCDEF***... Setzen Sie auf der Seite im Abschnitt Umgebungsvariablen die Variable BUG_ENABLED auf true und wählen Sie dann Speichern aus.
5. Kopieren Sie den folgenden JSON-Code in eine Datei mit dem Namen `test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
"quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Um eine HTTPS-Anfrage an Ihren API-Endpunkt zu senden, übergeben Sie die Nutzlast des Beispielereignisses als Eingabe, indem Sie einen `curl`-Befehl ausführen, zum Beispiel:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

Die API gibt die folgende leere Antwort unter Angabe einer erfolgreichen Ausführung zurück:

```
{ }
```

Schritt 2: Überprüfen Sie die simulierte Datenbeschädigung

1. Melden Sie sich bei der Amazon DynamoDB-Konsole an.
2. Wählen Sie im Navigationsbereich Tables (Tabellen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie `CheckoutTable` aus.
4. Wählen Sie auf der Tabellendetailseite Items (Elemente) und dann das erstellte Element aus.

   Die gespeicherten Attribute werden angezeigt, einige markiert als CORRUPTED (BESCHÄDIGT!)

Schritt 3: Deaktivieren Sie das simulierte Problem

1. Melden Sie sich an der AWS Lambda -Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example`und wählen Sie `CheckoutFunction` aus.
4. Auf dem fork-example-ecommerce- ***my-app*** - CheckoutFunction -***ABCDEF***... Setzen Sie auf der Seite im Abschnitt Umgebungsvariablen die Variable BUG_ENABLED auf false und wählen Sie dann Speichern aus.

Schritt 4: Aktivieren Sie die Wiedergabefunktion, um das Problem zu beheben

1. Wählen Sie in der AWS Lambda Konsole im Navigationsbereich die Option Funktionen aus.
2. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie `ReplayFunction` aus.
3. Erweitern Sie den Abschnitt Designer (Designer), wählen Sie das Feld SQS (SQS) und dann im Abschnitt SQS (SQS) Enabled (Aktiviert) aus.

   > **Note**
   >
   > Es dauert ca. 1 Minute, bis der Amazon SQS-Ereignisquellenauslöser aktiviert wird.

4. Wählen Sie Save (Speichern) aus.
5. Zur Anzeige der wiederhergestellten Attribute kehren Sie zur Amazon–DynamoDB-Konsole zurück.
6. Um die Wiedergabe zu deaktivieren, kehren Sie zur AWS Lambda Konsole zurück und deaktivieren Sie den Amazon SQS SQS-Ereignisquellenauslöser für. `ReplayFunction`

AWS Event Fork-Pipelines für ein Amazon SNS SNS-Thema abonnieren

Um die Entwicklung Ihrer ereignisgesteuerten Anwendungen zu beschleunigen, können Sie Event-Handling-Pipelines — unterstützt von AWS Event Fork Pipelines — für Amazon SNS SNS-Themen abonnieren. AWS Event Fork Pipelines ist eine Suite verschachtelter Open-Source-Anwendungen, die auf dem AWS Serverless Application Model (AWS SAM) basieren und die Sie direkt aus der AWS Event Fork Pipelines Suite (wählen Sie Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) in Ihrem Konto bereitstellen können. AWS Weitere Informationen finden Sie unter So funktionieren AWS Event Fork-Pipelines.

In diesem Abschnitt wird gezeigt, wie Sie die verwenden können AWS Management Console , um eine Pipeline bereitzustellen und anschließend AWS Event Fork-Pipelines für ein Amazon SNS SNS-Thema zu abonnieren. Bevor Sie beginnen, erstellen Sie ein Amazon SNS-Thema.

Um die Ressourcen zu löschen, aus denen eine Pipeline besteht, suchen Sie die Pipeline auf der Anwendungsseite von in der AWS Lambda Konsole, erweitern Sie den Abschnitt SAM-Vorlage, wählen Sie CloudFormationStack und dann Andere Aktionen, Stack löschen aus.

Bereitstellung und Abonnement der Event Storage and Backup Pipeline für Amazon SNS

> Für die Archivierung und Analyse von Ereignissen empfiehlt Amazon SNS jetzt, die native Integration mit Amazon Data Firehose zu verwenden. Sie können Firehose-Lieferstreams für SNS-Themen abonnieren, sodass Sie Benachrichtigungen an Archivierungs- und Analyseendpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden können. Die Verwendung von Amazon SNS mit Firehose Delivery Streams ist eine vollständig verwaltete und codelose Lösung, für die Sie keine Funktionen verwenden müssen. AWS Lambda Weitere Informationen finden Sie unter Lieferstreams von Fanout nach Firehose.

In diesem Tutorial wird gezeigt, wie Sie die Event Storage Backup Pipeline bereitstellen und ein Amazon-SNS-Thema abonnieren. Dieser Prozess wandelt die der Pipeline zugeordnete AWS SAM Vorlage automatisch in einen AWS CloudFormation Stack um und stellt den Stack dann in Ihrem Stack bereit. AWS-Konto Dieser Prozess erstellt und konfiguriert außerdem eine Reihe von Ressourcen, den Ereignisspeicher und die Sicherungspipeline umfassen, darunter:

- Amazon-SQS-Warteschlange

- Lambda-Funktion
- Firehose-Bereitstellungsdat
- Amazon S3 Backup Bucket

Weitere Informationen zur Konfiguration eines Streams mit einem Amazon S3 S3-Bucket als Ziel finden Sie `S3DestinationConfiguration` in der Amazon Data Firehose API-Referenz.

Weitere Informationen zur Transformation von Ereignissen und zur Konfiguration von Ereignispufferung, Ereigniskomprimierung und Ereignisverschlüsselung finden Sie unter Creating an Delivery Stream im Amazon Data Firehose Developer Guide.

Weitere Informationen zur Filterung von Ereignissen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements in diesem Leitfaden.

1. Melden Sie sich an der AWS Lambda -Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:
   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).
   b. Suchen Sie nach `fork-event-storage-backup-pipeline` und wählen Sie dann die Anwendung aus.
4. Gehen Sie auf der Seite fork-event-storage-backup-pipeline wie folgt vor:
   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `my-app-backup`).

   > **Note**
   >
   > - Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation Stack (anstatt einen neuen zu erstellen).

b. (Optional) Geben Sie für BucketArnden ARN des Amazon S3 S3-Buckets ein, in den eingehende Ereignisse geladen werden. Wenn Sie keinen Wert eingeben, wird ein neuer Amazon S3 S3-Bucket in Ihrem AWS Konto erstellt.

c. (Optional) Geben Sie für DataTransformationFunctionArnden ARN der Lambda-Funktion ein, über die die eingehenden Ereignisse transformiert werden. Wenn Sie keinen Wert eingeben, wird die Datentransformation deaktiviert.

d. (Optional) Geben Sie eine der folgenden LogLevelEinstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:

   - `DEBUG`
   - `ERROR`
   - `INFO` (Standard)
   - `WARNING`

e. Geben Sie für TopicArnden ARN des Amazon SNS SNS-Themas ein, für das diese Instance der Fork-Pipeline abonniert werden soll.

f. (Optional) Geben Sie für StreamBufferingIntervalInSecondsund StreamBufferingSizeInMBsdie Werte für die Konfiguration der Pufferung eingehender Ereignisse ein. Wenn Sie keine Werte eingeben, 300 Sekunden und 5 MB verwendet.

g. (Optional) Geben Sie eine der folgenden StreamCompressionFormatEinstellungen für die Komprimierung eingehender Ereignisse ein:

   - `GZIP`
   - `SNAPPY`
   - `UNCOMPRESSED` (Standard)
   - `ZIP`

h. (Optional) Geben Sie für das Zeichenkettenpräfix ein StreamPrefix, um Dateien zu benennen, die im Amazon S3 S3-Backup-Bucket gespeichert sind. Wenn Sie keinen Wert eingeben, wird kein Präfix verwendet.

i. (Optional) Geben Sie für SubscriptionFilterPolicydie Amazon SNS SNS-Abonnementfilterrichtlinie im JSON-Format ein, die zum Filtern eingehender Ereignisse verwendet werden soll. Die Filterrichtlinie entscheidet, welche Ereignisse im OpenSearch Service-Index indexiert werden. Wenn Sie keinen Wert eingeben, wird keine Filterung verwendet wird (alle Ereignisse werden indiziert).

j. (Optional) Geben Sie für die Zeichenfolge ein SubscriptionFilterPolicyScope, `MessageBody` oder `MessageAttributes` um die nutzlast- oder attributbasierte Nachrichtenfilterung zu aktivieren.

k. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann Deploy (Bereitstellen).

Auf der *my-app* Seite Bereitstellungsstatus für zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

AWS CloudFormation Beginnt im Abschnitt Ressourcen mit der Erstellung des Stacks und zeigt den CREATE_IN_PROGRESS-Status für jede Ressource an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Zu Ihrem Amazon SNS SNS-Thema veröffentlichte Nachrichten werden im Amazon S3 S3-Backup-Bucket gespeichert, der von der Event Storage and Backup-Pipeline automatisch bereitgestellt wird.

Bereitstellung und Abonnement der Event Search and Analytics-Pipeline für Amazon SNS

> Für die Archivierung und Analyse von Ereignissen empfiehlt Amazon SNS jetzt, die native Integration mit Amazon Data Firehose zu verwenden. Sie können Firehose-Lieferstreams für SNS-Themen abonnieren, sodass Sie Benachrichtigungen an Archivierungs- und Analyseendpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden können. Die Verwendung von Amazon SNS mit Firehose Delivery Streams ist eine vollständig verwaltete und codelose Lösung, für die Sie keine Funktionen verwenden müssen. AWS Lambda Weitere Informationen finden Sie unter Lieferstreams von Fanout nach Firehose.

In diesem Tutorial wird gezeigt, wie Sie die Ereignissuche- und Analytik-Pipeline bereitstellen und von diesen ein Amazon SNS-Thema abonnieren lassen. Dieser Prozess wandelt die der Pipeline zugeordnete AWS SAM Vorlage automatisch in einen AWS CloudFormation Stack um und stellt den

Stack dann in Ihrem Stack bereit. AWS-Konto Dieser Prozess erstellt und konfiguriert außerdem eine Reihe von Ressourcen, die die Ereignissuche und die Analysepipeline umfassen, darunter:

- Amazon-SQS-Warteschlange
- Lambda-Funktion
- Firehose-Bereitstellungsdat
- Amazon OpenSearch Service-Domäne
- Amazon S3 Bucket für unzustellbare Nachrichten

Weitere Informationen zur Konfiguration eines Streams mit einem Index als Ziel finden Sie `ElasticsearchDestinationConfiguration` in der Amazon Data Firehose API-Referenz.

Weitere Informationen zur Transformation von Ereignissen und zur Konfiguration von Ereignispufferung, Ereigniskomprimierung und Ereignisverschlüsselung finden Sie unter Creating an Delivery Stream im Amazon Data Firehose Developer Guide.

Weitere Informationen zur Filterung von Ereignissen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements in diesem Leitfaden.

1. Melden Sie sich an der AWS Lambda -Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:
   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).
   b. Suchen Sie nach `fork-event-search-analytics-pipeline` und wählen Sie dann die Anwendung aus.
4. Gehen Sie auf der Seite fork-event-search-analytics-pipeline wie folgt vor:
   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `my-app-search`).

> **Note**
>
> Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation Stack (anstatt einen neuen zu erstellen).

b. (Optional) Geben Sie für den ARN der Lambda-Funktion ein DataTransformationFunctionArn, die für die Transformation eingehender Ereignisse verwendet wird. Wenn Sie keinen Wert eingeben, wird die Datentransformation deaktiviert.

c. (Optional) Geben Sie eine der folgenden LogLevelEinstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:

   - `DEBUG`
   - `ERROR`
   - `INFO` (Standard)
   - `WARNING`

d. (Optional) Geben Sie für SearchDomainArnden ARN der OpenSearch Service-Domain ein, einen Cluster, der die benötigten Rechen- und Speicherfunktionen konfiguriert. Wenn Sie keinen Wert eingeben, wird eine neue Domäne mit der Standard-Konfiguration erstellt.

e. Geben Sie für TopicArnden ARN des Amazon SNS SNS-Themas ein, für das diese Instance der Fork-Pipeline abonniert werden soll.

f. Geben Sie für SearchIndexNameden Namen des OpenSearch Service-Index für die Suche und Analyse von Ereignissen ein.

> **Note**
>
> Die folgenden Kontingente gelten für Indexnamen:
>
> - Sie dürfen keine Großbuchstaben enthalten.
> - Es dürfen keine der folgenden Zeichen enthalten sein: `\ / * ? " < > | ` , #`
> - Sie dürfen nicht mit den folgenden Zeichen beginnen: `- + _`
> - Sie dürfen nicht folgende Zeichen sein: `. ..`
> - Sie dürfen nicht mehr als 80 Zeichen enthalten.
> - Sie dürfen nicht mehr als 255 Bytes enthalten.

- Darf keinen Doppelpunkt enthalten (ab OpenSearch Service 7.0)

g. (Optional) Geben Sie eine der folgenden SearchIndexRotationPeriodEinstellungen für den Rotationszeitraum des OpenSearch Serviceindex ein:

- `NoRotation` (Standard)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

Die Indexrotation hängt dem Indexnamen einen Zeitstempel an, was den Ablauf veralteter Daten ermöglicht.

h. Geben Sie für SearchTypeNameden Namen des OpenSearch Servicetyps für die Organisation der Ereignisse in einem Index ein.

> **Note**
>
> - OpenSearch Namen von Servicetypen können ein beliebiges Zeichen (außer Null-Bytes) enthalten, dürfen aber nicht mit beginnen_.
> - Für OpenSearch Service 6.x kann es nur einen Typ pro Index geben. Wenn Sie einen neuen Typ für einen vorhandenen Index angeben, der bereits einen anderen Typ hat, gibt Firehose einen Laufzeitfehler zurück.

i. (Optional) Geben Sie für StreamBufferingIntervalInSecondsund StreamBufferingSizeInMBsdie Werte für die Konfiguration der Pufferung eingehender Ereignisse ein. Wenn Sie keine Werte eingeben, 300 Sekunden und 5 MB verwendet.

j. (Optional) Geben Sie eine der folgenden StreamCompressionFormatEinstellungen für die Komprimierung eingehender Ereignisse ein:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (Standard)
- `ZIP`

k. (Optional) Geben Sie für das Zeichenkettenpräfix ein StreamPrefix, um Dateien zu benennen, die im Amazon S3 S3-Dead-Letter-Bucket gespeichert sind. Wenn Sie keinen Wert eingeben, wird kein Präfix verwendet.

l. (Optional) Geben Sie für die Wiederholungsdauer für Fälle ein StreamRetryDurationInSecons, in denen Firehose keine Ereignisse im OpenSearch Serviceindex indizieren kann. Wenn Sie keinen Wert eingeben, wird 300 Sekunden verwendet.

m. (Optional) Geben Sie für SubscriptionFilterPolicydie Amazon SNS SNS-Abonnementfilterrichtlinie im JSON-Format ein, die zum Filtern eingehender Ereignisse verwendet werden soll. Die Filterrichtlinie entscheidet, welche Ereignisse im OpenSearch Service-Index indexiert werden. Wenn Sie keinen Wert eingeben, wird keine Filterung verwendet wird (alle Ereignisse werden indiziert).

n. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann Deploy (Bereitstellen).

Auf der `my-app-search` Seite Bereitstellungsstatus für zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

AWS CloudFormation Beginnt im Abschnitt Ressourcen mit der Erstellung des Stacks und zeigt den CREATE_IN_PROGRESS-Status für jede Ressource an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Nachrichten, die zu Ihrem Amazon SNS SNS-Thema veröffentlicht wurden, werden automatisch im OpenSearch Service-Index indexiert, der von der Event Search and Analytics-Pipeline bereitgestellt wird. Wenn die Pipeline ein Ereignis nicht indizieren kann, speichert sie es in einem Amazon S3 S3-Dead-Letter-Bucket.

Bereitstellung der Event Replay Pipeline mit Amazon SNS SNS-Integration

In diesem Tutorial wird gezeigt, wie Sie die Ereignis-Wiederholungspipeline bereitstellen und von dieser ein Amazon-SNS-Thema abonnieren lassen. Dieser Prozess wandelt die der Pipeline zugeordnete AWS SAM Vorlage automatisch in einen AWS CloudFormation Stack um und stellt den

Stack dann in Ihrem bereit. AWS-Konto Dieser Prozess erstellt und konfiguriert außerdem eine Reihe von Ressourcen, die die Ereignis-Wiederholungs-Pipeline umfassen, darunter eine Amazon-SQS-Warteschlange und eine Lambda-Funktion.

Weitere Informationen zur Filterung von Ereignissen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements in diesem Leitfaden.

1. Melden Sie sich an der AWS Lambda -Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:
   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).
   b. Suchen Sie nach `fork-event-replay-pipeline` und wählen Sie dann die Anwendung aus.
4. Auf derfork-event-replay-pipeline-Seite, machen Sie das Folgende:
   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `my-app-replay`).

      > **Note**
      > Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation Stack (anstatt einen neuen zu erstellen).

   b. (Optional) Geben Sie eine der folgenden LogLevelEinstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:
      - `DEBUG`
      - `ERROR`
      - `INFO` (Standard)
      - `WARNING`

c. (Optional) Geben Sie für ReplayQueueRetentionPeriodInSecondsden Zeitraum in Sekunden ein, für den die Amazon SQS SQS-Wiedergabewarteschlange die Nachricht aufbewahrt. Wenn Sie keinen Wert eingeben, wird 1.209.600 Sekunden (14 Tage) verwendet.

d. Geben Sie für TopicArnden ARN des Amazon SNS SNS-Themas ein, für das diese Instance der Fork-Pipeline abonniert werden soll.

e. Geben Sie für DestinationQueueNameden Namen der Amazon SQS SQS-Warteschlange ein, an die die Lambda-Wiedergabefunktion Nachrichten weiterleitet.

f. (Optional) Geben Sie für SubscriptionFilterPolicydie Amazon SNS SNS-Abonnementfilterrichtlinie im JSON-Format ein, die zum Filtern eingehender Ereignisse verwendet werden soll. Die Filterrichtlinie legt fest, welche Ereignisse für die Wiederholung gepuffert werden. Wenn Sie keinen Wert eingeben, wird keine Filterung verwendet wird (alle Ereignisse werden für die Wiederholung gepuffert).

g. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann Deploy (Bereitstellen).

Auf der `my-app-replay` Seite Bereitstellungsstatus für zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

AWS CloudFormation Beginnt im Abschnitt Ressourcen mit der Erstellung des Stacks und zeigt den CREATE_IN_PROGRESS-Status für jede Ressource an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Für Ihr Amazon SNS-Thema veröffentlichte Nachrichten werden in der von der Ereignis-Wiederholungspipeline bereitgestellten Amazon SQS-Warteschlange automatisch gepuffert.

> **Note**
>
> Standardmäßig ist die Wiederholung deaktiviert. Um die Wiederholung zu aktivieren, navigieren Sie zur Seite der Funktion in der Lambda-Konsole, erweitern Sie den Abschnitt Designer, wählen Sie das Feld SQS und dann im Abschnitt SQS die Option Enabled (Aktiviert).

Amazon EventBridge Scheduler mit Amazon SNS verwenden

Amazon EventBridge Scheduler ist ein serverloser Scheduler, mit dem Sie Aufgaben von einem zentralen, verwalteten Service aus erstellen, ausführen und verwalten können. Mit EventBridge Scheduler können Sie Zeitpläne mithilfe von Cron und Preisausdrücken für wiederkehrende Muster erstellen oder einmalige Aufrufe konfigurieren. Sie können flexible Zeitfenster für die Zustellung einrichten, Wiederholungslimits definieren und die maximale Aufbewahrungszeit für fehlgeschlagene API-Aufrufe festlegen.

Auf dieser Seite wird erklärt, wie Sie EventBridge Scheduler verwenden, um eine Nachricht aus einem Amazon SNS SNS-Thema nach einem Zeitplan zu veröffentlichen.

Einrichtung der Ausführungsrolle

Wenn Sie einen neuen Zeitplan erstellen, muss EventBridge Scheduler über die Berechtigung verfügen, seinen API-Zielvorgang in Ihrem Namen aufzurufen. Sie gewähren EventBridge Scheduler diese Berechtigungen mithilfe einer Ausführungsrolle. Die Berechtigungsrichtlinie, die Sie der Ausführungsrolle Ihres Zeitplans hinzufügen, definiert die erforderlichen Berechtigungen. Diese Berechtigungen hängen von der Ziel-API ab, die EventBridge Scheduler aufrufen soll.

Wenn Sie die EventBridge Scheduler-Konsole verwenden, um einen Zeitplan zu erstellen, wie im folgenden Verfahren beschrieben, richtet EventBridge Scheduler automatisch eine Ausführungsrolle auf der Grundlage Ihres ausgewählten Ziels ein. Wenn Sie einen Zeitplan mit einem der EventBridge Scheduler SDKs, dem AWS CLI, oder erstellen möchten, müssen Sie über eine vorhandene Ausführungsrolle verfügen AWS CloudFormation, die die Berechtigungen gewährt, die EventBridge Scheduler zum Aufrufen eines Ziels benötigt. Weitere Informationen zum manuellen Einrichten einer Ausführungsrolle für Ihren Zeitplan finden Sie unter Einrichten einer Ausführungsrolle im Scheduler-Benutzerhandbuch. EventBridge

Erstellen eines Zeitplans

So erstellen Sie einen Zeitplan mithilfe der Konsole

1. Öffnen Sie die Amazon EventBridge Scheduler-Konsole zu https://console.aws.amazon.com/scheduler/Hause.
2. Wählen Sie auf der Seite Zeitpläne die Option Zeitplan erstellen aus.
3. Gehen Sie auf der Seite Zeitplandetails angeben im Abschnitt Zeitplanname und -beschreibung wie folgt vor:

a. Geben Sie unter Zeitplanname einen Namen für Ihren Zeitplan ein. Beispiel, **`MyTestSchedule`**.

b. (Optional) Geben Sie unter Beschreibung eine Beschreibung für Ihren Zeitplan ein. Beispiel, **`My first schedule`**.

c. Wählen Sie für Zeitplangruppe eine Zeitplangruppe aus der Dropdown-Liste aus. Wenn Sie noch keine Gruppe haben, wählen Sie Standard. Um eine Zeitplangruppe zu erstellen, wählen Sie Eigenen Zeitplan erstellen.

   Sie verwenden Zeitplangruppen, um Tags zu Zeitplangruppen hinzuzufügen.

4. • Wählen Sie Ihre Zeitplanoptionen.

Vorkommen	Vorgehensweise	
Einmaliger Zeitplan Ein einmaliger Zeitplan ruft ein Ziel nur einmal zu dem von Ihnen angegebenen Datum und der angegeben en Uhrzeit auf.	Gehen Sie für Datum und Uhrzeit wie folgt vor: • Geben Sie ein gültiges Datum im `YYYY/MM/D D` -Format ein. • Geben Sie einen Zeitstempel im 24-Stunden-Format (`hh:mm`) ein. • Wählen Sie unter Zeitzone die Zeitzone aus.	
Wiederkehrender Zeitplan Ein wiederkehrender Zeitplan ruft ein Ziel mit einer Rate auf, die Sie mit einem cron-Ausdruck oder einem Rate-Ausdruck angeben.	a. Gehen Sie bei Zeitplant yp wie folgt vor: • Um den Zeitplan mithilfe eines Cron-Ausdrucks zu definiere n, wählen Sie Cron-basierter Zeitplan und	

Vorkommen	Vorgehensweise	
	geben Sie den Cron-Ausdruck ein. • Um den Zeitplan mithilfe eines Rate-Ausdrucks zu definiere n, wählen Sie Rate-Ausdruck ein. Weitere Informationen zu Cron- und Rate-Ausdrücken finden Sie unter Zeitplant ypen auf EventBrid ge Scheduler im Amazon EventBridge Scheduler-Benutzer handbuch. b. Wählen Sie für Flexibles Zeitfenster die Option Aus, um die Option zu deaktivie ren, oder wählen Sie eines der vordefinierten Zeitfenster aus. Wenn Sie beispielsweise 15 Minuten auswählen und einen wiederkehrenden Zeitplan festlegen, der sein Ziel einmal pro Stunde aufruft, wird der Zeitplan innerhalb von 15 Minuten nach Beginn jeder Stunde ausgeführt.	

5. (Optional) Wenn Sie im vorherigen Schritt Wiederkehrender Zeitplan ausgewählt haben, gehen Sie im Abschnitt Zeitrahmen wie folgt vor:

   a. Wählen Sie unter Zeitzone eine Zeitzone aus.

   b. Geben Sie für Startdatum und -uhrzeit ein gültiges Datum im `YYYY/MM/DD`-Format ein und geben Sie dann einen Zeitstempel im 24-Stunden-Format (`hh:mm`) an.

   c. Geben Sie für Enddatum und -uhrzeit ein gültiges Datum im `YYYY/MM/DD`-Format ein und geben Sie dann einen Zeitstempel im 24-Stunden-Format (`hh:mm`) an.

6. Wählen Sie Weiter.

7. Wählen Sie auf der Seite „Ziel auswählen“ den AWS API-Vorgang aus, den der Scheduler aufruft EventBridge :

   a. Wählen Sie Amazon SNS Publish (Amazon SNS Veröffentlichung).

   b. Wählen Sie im Abschnitt Veröffentlichen eine SNS Thema oder wählen Sie Neu erstellen SNS Thema.

   c. (Optional) Geben Sie ein JSON Nutzlast. Wenn Sie keine Nutzlast eingeben, verwendet EventBridge Scheduler ein leeres Ereignis, um die Funktion aufzurufen.

8. Wählen Sie Weiter.

9. Führen Sie auf der Seite Settings (Einstellungen) die folgenden Schritte aus:

   a. Um den Zeitplan zu aktivieren, schalten Sie unter Zeitplanstatus die Option Zeitplan aktivieren ein.

   b. Um eine Wiederholungsrichtlinie für Ihren Zeitplan zu konfigurieren, gehen Sie unter Wiederholungsrichtlinie und Warteschlange für unzustellbare Nachrichten (DLQ) wie folgt vor:

      - Aktivieren Sie die Option Wiederholen.
      - Geben Sie unter Höchstalter des Ereignisses die maximale (n) Stunde (n) und Minute (n) ein, die der EventBridge Scheduler für ein unbearbeitetes Ereignis speichern muss.
      - Die Höchstdauer beträgt 24 Stunden.
      - Geben Sie unter Maximale Anzahl an Wiederholungen ein, wie oft der EventBridge Scheduler den Zeitplan maximal wiederholt, falls das Ziel einen Fehler zurückgibt.

        Der Maximalwert beträgt 185 Wiederholungen.

Bei Wiederholungsrichtlinien führt der Scheduler den Zeitplan erneut aus, wenn ein Zeitplan sein Ziel nicht aufrufen kann. EventBridge Falls konfiguriert, müssen Sie die maximale Aufbewahrungszeit und Wiederholungsversuche für den Zeitplan festlegen.

c. Wählen Sie aus, wo der EventBridge Scheduler nicht zugestellte Ereignisse speichert.

Option für Warteschlange für unzustellbare Nachrichten (DLQ)	Vorgehensweise
Nicht speichern	Wählen Sie None.
Speichern Sie das Ereignis dort, AWS-Konto wo Sie den Zeitplan erstellen	a. Wählen Sie Wählen Sie eine Amazon SQS SQS-Warteschlange in meiner AWS-Konto als DLQ aus. b. Wählen Sie den Amazon-Ressourcennamen (ARN) der Amazon SQS-Warteschlange.
Speichern Sie das Ereignis an einem anderen Ort als AWS-Konto dem, an dem Sie den Zeitplan erstellen	a. Wählen Sie AWS-Konten Als DLQ eine Amazon SQS SQS-Warteschlange angeben aus. b. Geben Sie den Amazon-Ressourcennamen (ARN) der Amazon-SQS-Warteschlange ein.

d. Um einen kundenverwalteten Schlüssel zur Verschlüsselung Ihrer Zieleingabe zu verwenden, wählen Sie unter Verschlüsselung die Option Verschlüsselungseinstellungen anpassen (erweitert).

Wenn Sie diese Option wählen, geben Sie einen vorhandenen CMK-ARN ein oder wählen Sie Erstellen eines AWS KMS key, um zur AWS KMS -Konsole zu navigieren. Weitere Informationen darüber, wie EventBridge Scheduler Ihre Daten im Ruhezustand verschlüsselt, finden Sie unter Verschlüsselung im Ruhezustand im Amazon EventBridge Scheduler-Benutzerhandbuch.

e. Damit EventBridge Scheduler eine neue Ausführungsrolle für Sie erstellt, wählen Sie Neue Rolle für diesen Zeitplan erstellen. Geben Sie dann einen Namen für Rollenname ein. Wenn Sie diese Option wählen, ordnet EventBridge Scheduler der Rolle die erforderlichen Berechtigungen zu, die für Ihr vorgegebenes Ziel erforderlich sind.

10. Wählen Sie Weiter.
11. Überprüfen Sie auf der Seite Zeitplan überprüfen und erstellen die Details Ihres Zeitplans. Wählen Sie in jedem Abschnitt Bearbeiten aus, um zu diesem Schritt zurückzukehren und seine Details zu bearbeiten.
12. Wählen Sie Zeitplan erstellen.

    Auf der Seite Zeitpläne können Sie eine Liste Ihrer neuen und vorhandenen Zeitpläne anzeigen. Überprüfen Sie in der Spalte Status, ob Ihr neuer Zeitplan aktiviert ist.

Zugehörige Ressourcen

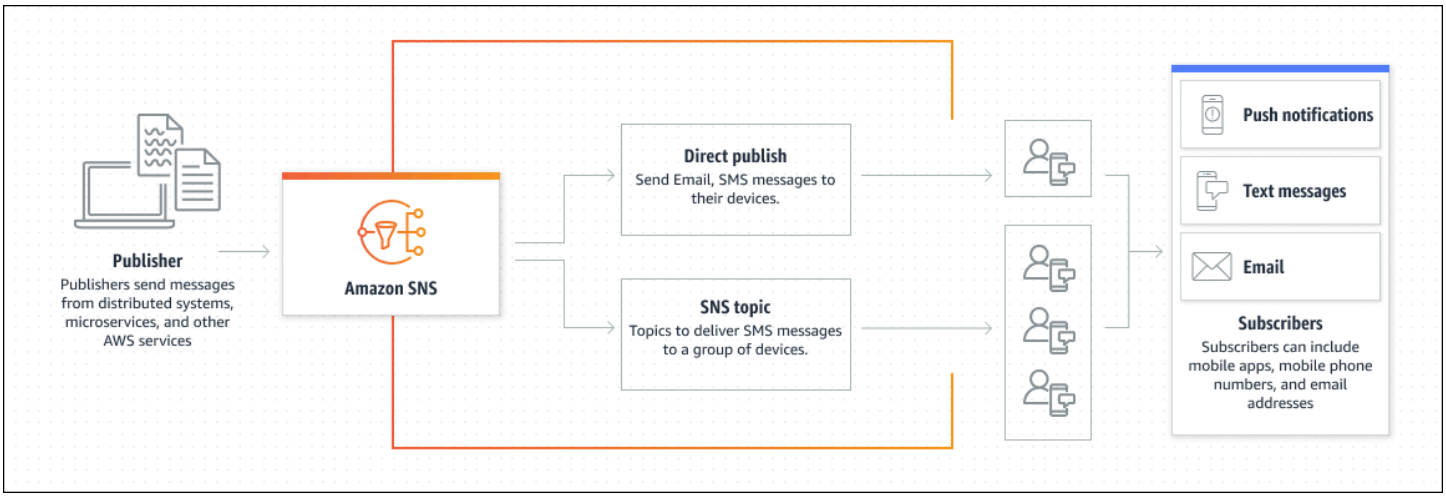
Weitere Informationen zu EventBridge Scheduler finden Sie im Folgenden:

- EventBridge Scheduler-Benutzerhandbuch
- EventBridge Scheduler-API-Referenz
- EventBridge Preise für Scheduler

Amazon SNS für application-to-person den Nachrichtenversand verwenden

Mit Amazon SNS application-to-person (A2P) können Sie Benachrichtigungen und Benachrichtigungen per SMS (Short Message Service) direkt an die Mobilgeräte Ihrer Kunden senden. Mit dieser Funktion können Sie Push-Benachrichtigungen an mobile Apps, Textnachrichten an Handynummern und Klartext-E-Mails an E-Mail-Adressen senden. Darüber hinaus haben Sie die Flexibilität, Nachrichten mithilfe von Themen zu verteilen, um mehrere Empfänger zu erreichen, oder Nachrichten direkt an einzelne mobile Endpunkte zu veröffentlichen, um personalisierte Kommunikation zu ermöglichen.

In diesem Thema wird erklärt, wie Amazon SNS für Benutzerbenachrichtigungen mit Abonnenten verwendet wird, z. B. für mobile Anwendungen, Handynummern und E-Mail-Adressen.



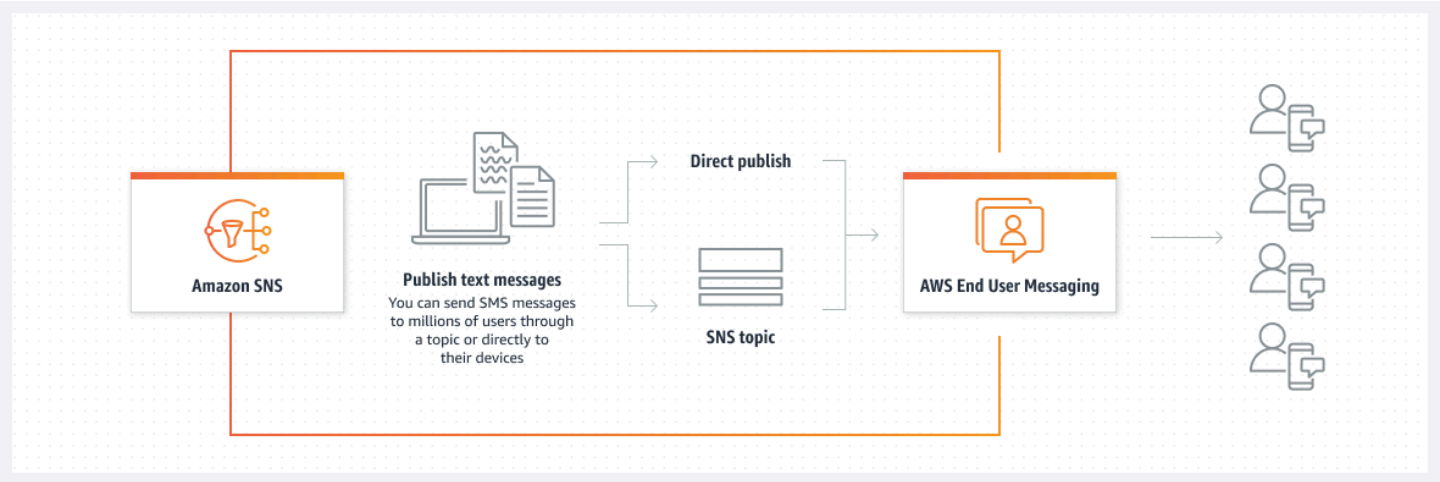
Mobiles Textnachrichten mit Amazon SNS

> **Important**
>
> Der Amazon SNS SMS Developer Guide wurde aktualisiert. Amazon SNS hat sich AWS Endbenutzer-Messaging SMSfür die Zustellung von SMS-Nachrichten integriert. Dieses Handbuch enthält die neuesten Informationen zur Erstellung, Konfiguration und Verwaltung Ihrer Amazon SNS SNS-SMS-Nachrichten.

Amazon SNS Mobile Text Messaging (SMS) wurde entwickelt, um die Nachrichtenzustellung an verschiedene Plattformen zu erleichtern, z. B. Web-, Mobil- und Geschäftsanwendungen, die SMS unterstützen. Benutzer können Nachrichten an eine oder mehrere Telefonnummern senden, indem sie ihnen ein Thema abonnieren, was den Verteilungsprozess vereinfacht.

Amazon SNS SNS-Nachrichten werden von zugestellt AWS Endbenutzer-Messaging SMS, wodurch eine zuverlässige Nachrichtenübertragung gewährleistet ist. In Amazon SNS APIs können Sie verschiedene Eigenschaften wie Nachrichtentypen (Werbe- oder Transaktionsnachrichten), monatliche Ausgabenlimits, Abmeldelisten und die Optimierung der Nachrichtenzustellung festlegen.

AWS Endbenutzer-Messaging SMS wickelt die Übertragung von Nachrichten an die Zieltelefonnummer über das globale SMS-Versorgungsnetz ab. Es verwaltet das Routing, den Lieferstatus und die erforderliche Einhaltung der regionalen Vorschriften. Informationen zum Zugriff auf zusätzliche SMS-Funktionen wie detaillierte Berechtigungen, Telefonpools, Konfigurationssätze, SMS-Simulator und Länderregeln finden Sie im AWS Endbenutzer-Messaging SMS Benutzerhandbuch.



Die folgenden Hauptfunktionen helfen Ihnen beim Senden von Amazon SNS SNS-SMS-Nachrichten, die skalierbar und leicht erweiterbar sind:

Passen Sie die Nachrichteneinstellungen an

Passen Sie SMS-Lieferungen an Ihre Bedürfnisse an, AWS-Konto indem Sie SMS-Einstellungen einrichten, die auf Ihrem Budget und Anwendungsfall basieren. Sie können beispielsweise wählen, ob Ihre Nachrichten der Kosteneffizienz oder der zuverlässigen Zustellung Priorität einräumen.

Legen Sie Ausgabenquoten fest

Passen Sie Ihre SMS-Lieferungen an, indem Sie Ausgabenkontingente oder für einzelne Nachrichtenzustellungen und monatliche Ausgabenkontingente für Sie angeben AWS-Konto. Sofern dies nach lokalen Gesetzen und Vorschriften (z. B. in den USA und Kanada) erforderlich ist, können sich SMS-Empfänger abmelden, was bedeutet, dass sie sich dafür entscheiden, keine SMS-Nachrichten mehr von Ihnen zu erhalten AWS-Konto. Nachdem sich ein Empfänger vom Empfang von Nachrichten abgemeldet hat, können Sie die Telefonnummer mit Einschränkungen erneut angeben, sodass Sie mit dem Senden von Nachrichten fortfahren können.

Senden Sie SMS-Nachrichten weltweit

Amazon SNS unterstützt SMS-Nachrichten in mehreren Regionen, sodass Sie Nachrichten in über 240 Länder und Regionen senden können.

Wie stellt Amazon SNS meine SMS-Nachrichten zu?

Wenn Sie Amazon SNS bitten, in Ihrem Namen SMS zu senden, werden die Nachrichten über AWS Endbenutzer-Messaging SMS. Die Integration zwischen Amazon SNS und AWS Endbenutzer-Messaging SMS bietet die folgenden Vorteile:

IAM-Richtlinien

Sie können IAM- und Ressourcenrichtlinien nutzen, um den Zugriff auf Ihre SMS-Ressourcen zu kontrollieren und auf andere AWS Dienste und Regionen zu verteilen.

AWS Endbenutzer-Messaging SMS -Konfigurationen

Alle Konfigurationen im Zusammenhang mit der Origination ID (Erstellung, Aktualisierung der Konfiguration, Bereitstellung neuer Originationen IDs, Änderung von Registrierungsvorlagen) verwenden. AWS Endbenutzer-Messaging SMS

AWS Endbenutzer-Messaging SMS Abrechnung

Die gesamte SMS-Abrechnung ist jedoch abgeschlossen AWS Endbenutzer-Messaging SMS. Sie können Ihre AWS Ausgaben für Ihre SMS-Workloads konsolidieren und gleichzeitig Ihre SMS-Ressourcen an einem zentralen Ort beschaffen und verwalten.

Erste Schritte mit Amazon SNS SMS

> **Important**
>
> Der Amazon SNS SMS Developer Guide wurde aktualisiert. Amazon SNS hat sich AWS Endbenutzer-Messaging SMSfür die Zustellung von SMS-Nachrichten integriert. Dieses Handbuch enthält die neuesten Informationen zur Erstellung, Konfiguration und Verwaltung Ihrer Amazon SNS SNS-SMS-Nachrichten.

Dieses Thema führt Sie durch die Verwaltung Ihrer SMS-Sandbox und die Konfiguration von IAM und ressourcenbasierten Richtlinien, um Amazon SNS die erforderlichen Berechtigungen für den Zugriff und die Nutzung von zu gewähren. AWS Endbenutzer-Messaging SMS APIs

Voraussetzungen

Amazon SNS empfiehlt, Ihre IAM-Richtlinie um die folgenden Maßnahmen zu aktualisieren, um eine umfassende Kontrolle und Transparenz Ihrer Amazon SNS SNS-Ressourcen zu gewährleisten:

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

Verwenden der Amazon SNS SMS-Sandbox

Neu erstellte Amazon SNS SNS-SMS-Konten werden automatisch in die SMS-Sandbox aufgenommen, um die Sicherheit von AWS Kunden und Empfängern zu gewährleisten, indem das Risiko von Betrug und Missbrauch minimiert wird. Diese Umgebung dient als sicherer Bereich für Test- und Entwicklungszwecke. Während Sie in der SMS-Sandbox arbeiten, haben Sie Zugriff auf alle Amazon SNS SNS-Funktionen, unterliegen jedoch bestimmten Einschränkungen:

- Sie können SMS-Nachrichten nur an verifizierte Zieltelefonnummern senden.
- Sie können über bis zu 10 verifizierte Zielrufnummern verfügen.
- Sie können Ziel-Telefonnummern erst löschen, wenn seit der Überprüfung oder dem letzten Überprüfungsversuch mindestens 24 Stunden vergangen sind.

Sobald Ihr Konto die Sandbox verlassen hat, werden diese Einschränkungen aufgehoben, und Sie können SMS-Nachrichten an jeden Empfänger senden.

Erste Schritte

Neue Amazon SNS SNS-SMS-Konten werden in einer SMS-Sandbox platziert. Gehen Sie wie folgt vor, um Telefonnummern in Ihrer Sandbox zu erstellen und zu verwalten, Ausgangsnummern und Absender zu erstellen und Ihr Unternehmen zu IDs registrieren.

1. Fügen Sie der SMS-Sandbox eine Zieltelefonnummer hinzu. Einzelheiten zum Hinzufügen, Verwalten und Verschieben von Telefonnummern aus der Amazon SNS SMS-Sandbox finden Sie unter. Hinzufügen und Überprüfen von Telefonnummern in der Amazon SNS SMS-Sandbox
2. Erstellen Sie eine Absenderidentität, die Ihre Empfänger auf ihren Geräten sehen, wenn Sie ihnen eine SMS-Nachricht senden. Weitere Informationen zu Herkunftsidentitäten, einschließlich der verschiedenen Typen, die Sie verwenden können, finden Sie in der Dokumentation. Herkunftsidentitäten für Amazon SNS SNS-SMS-Nachrichten
3. Registrieren Sie Ihr Unternehmen. In einigen Ländern müssen Sie die Identität Ihres Unternehmens registrieren, um Telefonnummern oder Absender kaufen IDs und die Nachrichten überprüfen zu können, die Sie an Empfänger in ihrem Land senden. Informationen darüber, in welchen Ländern eine Registrierung erforderlich ist, finden Sie AWS Endbenutzer-Messaging SMS im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Unterstützte Länder und Regionen für SMS-Nachrichten.
4. Senden Sie Ihre Nachrichten an ein Thema oder ein Mobiltelefon. Weitere Informationen finden Sie unter Senden von SMS-Nachrichten mit Amazon SNS.

Hinzufügen und Überprüfen von Telefonnummern in der Amazon SNS SMS-Sandbox

Bevor Sie mit dem Senden von SMS-Nachrichten von Ihrem Computer aus beginnen können, AWS-Konto während Sie sich in der SMS-Sandbox befinden, müssen Sie die folgenden Einrichtungsschritte ausführen. Dadurch wird sichergestellt, dass Ihr Konto für SMS-Nachrichten bereit ist und dass Ihre Zieltelefonnummern ordnungsgemäß verifiziert sind.

1. Erstellen Sie eine Quell-ID. Ähnlich wie bei Konten außerhalb der SMS-Sandbox ist eine Ursprungs-ID erforderlich, bevor Sie SMS-Nachrichten an Empfänger in einigen Ländern oder Regionen senden können.
2. Fügen Sie die Zieltelefonnummern, an die Sie Nachrichten senden möchten, innerhalb der SMS-Sandbox hinzu.
3. Überprüfen Sie die Telefonnummern, um sicherzustellen, dass die Zieltelefonnummern für die Verwendung in Ihren SMS-Nachrichten gültig sind.

Fügen Sie Ziel-Telefonnummern hinzu und überprüfen Sie sie

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Konsolenmenü eine Region aus, die SMS-Nachrichten unterstützt.
3. Wählen Sie im Navigationsbereich Text-Messaging (SMS) aus.
4. Wählen Sie im Abschnitt Sandbox-Zielrufnummern die Option Telefonnummer hinzufügen aus.
5. Geben Sie unter Zieldetails die folgenden Informationen ein und wählen Sie dann Telefonnummer hinzufügen aus:
   - Landesvorwahl und Telefonnummer des Ziels.
   - Die Sprache, in der die Bestätigungsnachricht gesendet werden soll.
6. Nach dem Hinzufügen der Telefonnummer sendet Amazon SNS ein OTP an die angegebene Zieltelefonnummer. Dieses OTP ist für die Überprüfung erforderlich.
7. Sie erhalten das OTP als Standard-SMS-Nachricht an die von Ihnen angegebene Zieltelefonnummer.
   - Wenn Sie das Einmalpasswort nicht innerhalb von 15 Minuten erhalten, wählen Sie in der Amazon SNS SNS-Konsole die Option Bestätigungscode erneut senden aus.
   - Sie können das OTP innerhalb von 24 Stunden bis zu fünf Mal erneut senden.
8. Sobald Sie das OTP erhalten haben, geben Sie es in das Feld Bestätigungscode ein und wählen Sie Telefonnummer verifizieren aus.
9. Überprüfen Sie den Bestätigungsstatus.
   - Nach erfolgreicher Überprüfung der Telefonnummer werden die Telefonnummer und ihr Bestätigungsstatus im Abschnitt Sandbox-Zieltelefonnummern angezeigt.
   - Wenn der Status Ausstehend lautet, war die Überprüfung nicht erfolgreich. Dies kann beispielsweise der Fall sein, wenn Sie die Landesvorwahl nicht korrekt eingegeben haben.
   - Sie können ausstehende oder verifizierte Telefonnummern erst löschen, wenn seit dem letzten Bestätigungsversuch mindestens 24 Stunden vergangen sind.
10. Wenn Sie dieselbe Zieltelefonnummer in anderen Regionen verwenden möchten, wiederholen Sie die vorherigen Schritte für jede Region, in der Sie sie verwenden möchten.

Behebung von Problemen beim Nichtempfang eines OTP-Textes

Beheben Sie häufig auftretende Probleme, die verhindern können, dass eine Telefonnummer OTP-Texte empfängt.

- Amazon SNS SMS-Ausgabenlimit: Wenn Sie das Ausgabenlimit für den Versand von SMS-Nachrichten AWS-Konto überschritten haben, werden weitere Nachrichten, einschließlich OTP-Texte, möglicherweise erst zugestellt, wenn das Limit erhöht oder das Abrechnungsproblem gelöst ist.
- Telefonnummer, die nicht für SMS-Benachrichtigungen aktiviert wurde: In einigen Ländern oder Regionen müssen sich Empfänger dafür entscheiden, SMS-Nachrichten über Kurzcodes zu erhalten, die üblicherweise für OTP-Texte verwendet werden. Wenn die Telefonnummer des Empfängers nicht aktiviert ist, erhält er den OTP-Text nicht.
- Einschränkungen oder Filterung durch Mobilfunkanbieter: Einige Mobilfunkanbieter verfügen möglicherweise über Einschränkungen oder Filtermechanismen, die die Zustellung bestimmter Arten von SMS-Nachrichten, einschließlich OTP-Texten, verhindern. Dies könnte auf Sicherheitsrichtlinien oder Anti-Spam-Maßnahmen des Mobilfunkanbieters zurückzuführen sein.
- Ungültige oder falsche Telefonnummer: Wenn die vom Empfänger angegebene Telefonnummer falsch oder ungültig ist, wird der OTP-Text nicht zugestellt.
- Netzwerkprobleme: Vorübergehende Netzwerkprobleme oder Ausfälle können die Zustellung von SMS-Nachrichten, einschließlich OTP-Texten, an das Telefon des Empfängers verhindern.
- Verspätete Zustellung: In einigen Fällen kann es bei SMS-Nachrichten aufgrund von Netzwerküberlastungen oder anderen Faktoren zu Verzögerungen bei der Zustellung kommen. Der OTP-Text kann irgendwann zugestellt werden, er kann sich jedoch über den erwarteten Zeitrahmen hinaus verzögern.
- Sperrung oder Kündigung Ihres Kontos: Wenn es Probleme mit Ihrem Konto gibt AWS-Konto, wie z. B. Nichtzahlung oder Verstoß gegen die AWS Nutzungsbedingungen, können die Nachrichtenfunktionen von Amazon SNS, einschließlich OTP-Texten, gesperrt oder beendet werden.

Löschen von Telefonnummern aus der Amazon SNS SMS-Sandbox

Sie können sowohl ausstehende als auch verifizierte Zielrufnummern aus der SMS-Sandbox löschen.

> **Important**
>
> Sie können eine Telefonnummer nur 24 Stunden nach der Bestätigung der Telefonnummer oder 24 Stunden nach Ihrem letzten Bestätigungsversuch löschen.

So löschen Sie Zieltelefonnummern aus der SMS-Sandbox

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Konsolenmenü eine Region aus, die SMS-Nachrichten unterstützt und in der Sie eine Zieltelefonnummer hinzugefügt haben.
3. Wählen Sie im Navigationsbereich Textnachrichten (SMS) aus.
4. Navigieren Sie auf der Seite Mobile Textnachrichten (SMS) zum Abschnitt Sandbox-Zieltelefonnummern.
5. Wählen Sie die spezifische Telefonnummer aus, die Sie löschen möchten, und wählen Sie dann Telefonnummer löschen aus.
6. Geben Sie **`delete me`** ein, und wählen Sie dann Delete (Löschen), um zu bestätigen, dass Sie die Telefonnummer löschen möchten.

   Stellen Sie sicher, dass mindestens 24 Stunden vergangen sind, seit Sie die Zieltelefonnummer verifiziert oder versucht haben, sie zu verifizieren, bevor Sie mit dem Löschen fortfahren.
7. Wiederholen Sie diese Schritte in jeder Region, in der Sie die Zieltelefonnummer hinzugefügt haben und nicht mehr verwenden möchten.

Verlassen der Amazon SNS SMS-Sandbox

Um Sie AWS-Konto aus der SMS-Sandbox zu entfernen, müssen Sie zunächst Zielrufnummern hinzufügen, verifizieren und testen. Erstellen Sie anschließend einen Fall mit AWS -Support.

Um zu beantragen, dass Ihr AWS Konto aus der SMS-Sandbox entfernt wird

1. Überprüfen Sie die Telefonnummern
   a. Öffnen Sie AWS-Konto die Amazon SNS-Konsole, während Sie sich in der SMS-Sandbox befinden.
   b. Wählen Sie im Navigationsbereich unter Mobil die Option Textnachrichten (SMS) aus.

   c. Fügen Sie im Abschnitt Sandbox-Zielrufnummern eine oder mehrere Zielrufnummern hinzu und überprüfen Sie sie. Diese Überprüfung stellt sicher, dass Sie erfolgreich Nachrichten senden und empfangen können.

2. Testen Sie das Veröffentlichen von SMS

   - Vergewissern Sie sich, dass Sie Nachrichten an mindestens eine verifizierte Telefonnummer senden und empfangen können. Ausführlichere Anweisungen zum Veröffentlichen von SMS-Nachrichten finden Sie unterVeröffentlichen von SMS-Nachrichten auf einem Mobiltelefon mithilfe von Amazon SNS.

3. Initiieren Sie die Sandbox-Bearbeitung

   - Wählen Sie auf der Seite Amazon SNS-Konsole Text-Messaging (SMS) unter Kontoinformationen SMS-Sandbox verlassen. Diese Aktion leitet Sie zum Amazon Support Center weiter und erstellt automatisch einen Support-Fall, bei dem die Option zur Erhöhung des Servicekontingents ausgewählt ist.

4. Füllen Sie das Formular aus

   - Gehen Sie im Support-Formular unter Erhöhung der Servicequote wie folgt vor:

     i. Wählen Sie „SNS-Textnachrichten“ als Dienst aus.

     ii. Geben Sie die URL der Website oder den Namen der App an, von der aus Sie SMS-Nachrichten senden möchten.

     iii. Geben Sie die Art der Nachrichten an, die Sie senden möchten: Einmalpasswort, Werbe - oder Transaktionsnachricht.

     iv. Wählen Sie aus, AWS-Regionvon wo aus Sie SMS-Nachrichten senden möchten.

     v. Listen Sie die Länder oder Regionen auf, in denen Sie SMS-Nachrichten senden möchten.

     vi. Beschreiben Sie, wie sich Ihre Kunden für den Empfang von Nachrichten anmelden.

     vii. Fügen Sie alle Nachrichtenvorlagen hinzu, die Sie verwenden möchten.

5. Geben Sie Kontingent und Region an

   - Gehen Sie unter Requests (Anfragen) wie folgt vor:

     i. Wählen Sie AWS-Region, wohin Sie Ihre verschieben möchten AWS-Konto.

     ii. Wählen Sie Allgemeine Grenzwerte für den Ressourcentyp.

     iii. Wählen Sie für Quota die Option Exit SMS Sandbox aus.

iv. (Optional) Um zusätzliche Erhöhungen oder andere Anpassungen anzufordern, wählen Sie „Weitere Anfrage hinzufügen“ und geben Sie die erforderlichen Details an.

v. Geben Sie unter Neuer Kontingentwert das Limit in USD ein, das Sie beantragen.

6. Zusätzliche Details

   a. Geben Sie in der Fallbeschreibung alle zusätzlichen Details an, die für Ihre Anfrage relevant sind.

   b. Wählen Sie unter Kontaktoptionen Ihre bevorzugte Kontaktsprache aus.

7. Reichen Sie die Anfrage ein

   - Wählen Sie Senden, um Ihre Anfrage an zu senden Support.

Das Support Team gibt innerhalb von 24 Stunden eine erste Antwort auf Ihre Anfrage.

Da wir verhindern möchten, dass unerwünschte oder schädliche Inhalte in unseren Systemen eingehen, müssen wir jede Anfrage sorgfältig prüfen. Nach einer Prüfung kommen wir Ihrer Anfrage innerhalb dieses 24-Stunden-Zeitraums nach. Für den Fall, dass wir weitere Informationen von Ihnen benötigen, kann die Bearbeitung Ihrer Anfrage länger dauern.

Wenn Ihr Anwendungsfall gegen unsere Richtlinien verstößt, können wir Ihrer Anfrage möglicherweise nicht nachkommen.

Herkunftsidentitäten für Amazon SNS SNS-SMS-Nachrichten

> **Important**
>
> Der Amazon SNS SMS Developer Guide wurde aktualisiert. Amazon SNS hat sich AWS Endbenutzer-Messaging SMSfür die Zustellung von SMS-Nachrichten integriert. Dieses Handbuch enthält die neuesten Informationen zur Erstellung, Konfiguration und Verwaltung Ihrer Amazon SNS SNS-SMS-Nachrichten.

Herkunftsidentitäten für SMS-Nachrichten sind Identifikatoren, die zur Darstellung des Absenders einer SMS-Nachricht verwendet werden. Sie können sich gegenüber Ihren Empfängern anhand der folgenden Arten von Absenderidentitäten identifizieren:

Ursprungsnummern

Eine numerische Zeichenfolge, die die Telefonnummer eines SMS-Absenders identifiziert. Es gibt verschiedene Arten von Ausgangsnummern, darunter Langcodes (Standardtelefonnummern, die normalerweise 10 oder mehr Ziffern haben), 10-stellige Langcodes (10DLC), gebührenfreie Nummern (TFN) und Kurzcodes (Telefonnummern, die zwischen vier und sieben Ziffern enthalten).

In Ländern, in denen lokale Gesetze die Verwendung von Absendernummern IDs anstelle von Ursprungsnummern vorschreiben, ist die Support für Ursprungsnummern nicht verfügbar. Wenn Sie eine SMS-Nachricht mit einer Ursprungsnummer senden, zeigt das Gerät des Empfängers die Ursprungsnummer als Telefonnummer des Senders an. Sie können verschiedene Ursprungsnummern nach Anwendungsfall angeben.

Weitere Informationen finden Sie im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Telefonnummern.

> **Tip**
>
> Um eine Liste aller vorhandenen Ursprungsnummern in Ihrem AWS Konto anzuzeigen, wählen Sie im Navigationsbereich der Amazon SNS SNS-Konsole Originationsnummern aus.

Absender IDs

Ein alphabetischer Name, der den Absender einer SMS-Nachricht identifiziert. Wenn Sie eine SMS-Nachricht mit einer Absender-ID senden und sich der Empfänger in einem Bereich befindet, in dem die Absender-ID-Authentifizierung unterstützt wird, wird Ihre Absender-ID auf dem Gerät des Empfängers anstelle Ihrer Telefonnummer angezeigt. Eine Sender-ID stellt SMS-Empfängern mehr Informationen über den Sender zur Verfügung als eine Telefonnummer, Langwahl- oder Kurzwahlnummer.

Absender IDs werden in mehreren Ländern und Regionen auf der ganzen Welt unterstützt. In einigen Regionen müssen Sie als Unternehmen, das SMS-Nachrichten an einzelne Kunden sendet, eine Sender-ID verwenden, die vorab bei einer Regulierungsbehörde oder einem Branchengremium registriert wurde. Eine vollständige Liste der Länder und Regionen, die Sender unterstützen oder erfordern IDs, finden Sie AWS Endbenutzer-Messaging SMS im AWS

Endbenutzer-Messaging SMS Benutzerhandbuch unter Unterstützte Länder und Regionen für SMS-Nachrichten mit.

Für die Verwendung von Sender fallen keine zusätzlichen Gebühren an IDs. Die Unterstützung und die Anforderungen für die Absender-ID-Authentifizierung sind jedoch von Land zu Land unterschiedlich. Mehrere wichtige Märkte (darunter Kanada, China und die Vereinigte Staaten) unterstützen die Verwendung von Sender nicht IDs. In einigen Regionen müssen Sie als Unternehmen, das SMS-Nachrichten an einzelne Kunden sendet, eine Sender-ID verwenden, die vorab bei einer Regulierungsbehörde oder einem Branchengremium registriert wurde.

Weitere Informationen finden Sie IDs im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Sender.

Konfiguration von SMS-Nachrichten in Amazon SNS

> **Important**
>
> Der Amazon SNS SMS Developer Guide wurde aktualisiert. Amazon SNS hat sich AWS Endbenutzer-Messaging SMSfür die Zustellung von SMS-Nachrichten integriert. Dieses Handbuch enthält die neuesten Informationen zur Erstellung, Konfiguration und Verwaltung Ihrer Amazon SNS SNS-SMS-Nachrichten.

Sie können die Konfigurationen in Amazon SNS SMS verwenden, um SMS-Einstellungen an Ihre Anforderungen anzupassen, wie z. B. die Anpassung von Ausgabenkontingenten und die Einrichtung der Versandstatusprotokollierung. In diesem Thema erfahren Sie auch, wie Sie mithilfe der Amazon SNS SNS-Konsole und des AWS SDK SMS-Nachrichten zu Themen veröffentlichen, Kontingente effizient verwalten und detaillierte Statistiken zu SMS-Aktivitäten abrufen können.

Senden von SMS-Nachrichten mit Amazon SNS

In diesem Abschnitt wird beschrieben, wie SMS-Nachrichten mithilfe von Amazon SNS gesendet werden, einschließlich der Veröffentlichung zu einem Thema, des Abonnierens von Telefonnummern für Themen, dem Festlegen von Attributen für Nachrichten und dem direkten Veröffentlichen auf Mobiltelefonen.

Veröffentlichen von SMS-Nachrichten zu einem Amazon SNS SNS-Thema

Sie können eine einzelne SMS-Nachricht gleichzeitig an viele Telefonnummern veröffentlichen, indem Sie das Amazon SNS-Thema für diese Telefonnummern abonnieren. Ein SNS-Thema ist ein Kommunikationskanal, dem Sie Abonnenten hinzufügen können. Anschließend können Sie an all diese Abonnenten Nachrichten veröffentlichen. Ein Abonnent erhält alle Nachrichten, die zu dem Thema veröffentlicht wurden, bis Sie das Abonnement kündigen oder bis der Abonnent den Empfang von SMS-Nachrichten von Ihrem AWS Konto ablehnt.

Senden einer Nachricht zu einem Thema mithilfe der Konsole AWS

Erstellen Sie ein Thema wie folgt

Führen Sie die folgenden Schritte aus, wenn Sie noch kein Thema besitzen, an das Sie SMS-Nachrichten senden möchten.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Konsolenmenü eine Region aus, die SMS-Nachrichten unterstützt.
3. Wählen Sie im Navigationsbereich Themen aus.
4. Klicken Sie auf der Seite Themen auf Thema erstellen.
5. Gehen Sie auf der Seite Thema erstellen unter Details wie folgt vor:
   a. Wählen Sie unter Type (Typ) die Option Standard aus.
   b. Geben Sie für Name einen Themennamen ein.
   c. (Optional) Geben Sie unter Display name (Anzeigename) einen benutzerdefinierten Präfix für Ihre SMS-Nachrichten ein. Wenn Sie eine Nachricht an ein Thema senden, stellt Amazon SNS den Anzeigenamen voran, gefolgt von einer spitzen Klammer rechts (**>**) und einer Leerstelle. Anzeigenamen beachteten die Groß- und Kleinschreibung nicht und Amazon SNS konvertiert Anzeigenamen zu Großbuchstaben. Beispiel: Wenn der Anzeigename eines Themas `MyTopic` ist und die Nachricht lautet `Hello World!`, wird die Nachricht folgendermaßen angezeigt:

      ```
      MYTOPIC> Hello World!
      ```
6. Wählen Sie Thema erstellen aus. Der Name des Themas und der Amazon-Ressourcenname (ARN) werden auf der Registerkarte Themen angezeigt.

So erstellen Sie ein SMS-Abonnement:

Anhand von Abonnements können Sie eine SMS-Nachricht an mehrere Empfänger senden, indem Sie die Nachricht nur einmal zu dem Thema veröffentlichen.

> **Note**
>
> Wenn Sie beginnen, Amazon SNS zum Senden von SMS-Nachrichten zu verwenden, befindet sich Ihr AWS Konto in der SMS-Sandbox. Die SMS-Sandbox bietet Ihnen eine sichere Umgebung, in der Sie Amazon SNS Funktionen ausprobieren können, ohne Ihren Ruf als SMS-Sender zu gefährden. Während sich Ihr Konto in der SMS-Sandbox befindet, können Sie alle Funktionen von Amazon SNS verwenden. Sie können jedoch SMS-Nachrichten nur an verifizierte Zieltelefonnummern senden. Weitere Informationen finden Sie unter Verwenden der Amazon SNS SMS-Sandbox.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions aus.
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Gehen Sie auf der Seite Abonnement erstellen unter Details wie folgt vor:
   a. Für Thema-ARN geben Sie ein oder wählen Sie den Amazon-Ressourcennamen (ARN) des Themas, an das Sie SMS-Nachrichten senden möchten.
   b. Wählen Sie für Protokoll SMS aus.
   c. Geben Sie für Endpunkt die Telefonnummer ein, die mit Ihrem Thema abonniert werden soll.
5. Wählen Sie Create subscription (Abonnement erstellen) aus. Die Abonnementinformationen erscheint auf der Seite Abonnements.

   Wiederholen Sie diese Schritte, um weitere Telefonnummern hinzuzufügen. Sie können auch andere Abonnementtypen, wie E-Mail, hinzufügen.

So senden Sie eine Nachricht

Wenn Sie eine Nachricht veröffentlichen, versucht Amazon SNS diese Nachricht an alle Telefonnummern zu schicken, die das Thema abonniert haben.

1. Wählen Sie in der Amazon SNS-Konsole auf der Seite Themen den Namen des Themas, an das Sie SMS-Nachrichten senden möchten.
2. Wählen Sie auf der Seite mit den Details des Themas Publish message (Nachricht veröffentlichen) aus.
3. Geben Sie auf der Seite Nachricht für Thema veröffentlichen unter Nachrichtendetails wie folgt vor:

   a. Lassen Sie unter Subject (Betreff) das Feld leer, es sei denn, Ihr Thema enthält E-Mail-Abonnements und Sie möchten sowohl an E-Mail- und SMS-Abonnements Nachrichten veröffentlichen. Amazon SNS verwendet den Betreff, den Sie eingeben, als E-Mail-Betreffzeile.

   b. (Optional) Geben Sie für Time to Live (TTL) eine Anzahl an Sekunden an, in denen Amazon SNS Ihre SMS-Nachricht an Abonnenten des Mobilanwendungs-Endpunkts senden muss.
4. Gehen Sie unter Nachrichtentext wie folgt vor:

   a. Wählen Sie für Nachrichtenstruktur Identische Nutzlast für alle Bereitstellungsprotokolle, um die gleiche Nachricht an alle Protokolltypen zu schicken, die Ihr Thema abonniert haben. Oder wählen Sie Benutzerdefinierte Nutzlast für die einzelnen Bereitstellungsprotokolle, um die Nachricht für Abonnenten verschiedener Protokolltypen anzupassen. Sie können beispielsweise eine Standardnachricht für Telefonnummernabonnenten und eine benutzerdefinierte Nachricht für E-Mail-Abonnenten eingeben.

   b. Geben Sie für Nachrichtentext zum Senden an den Endpunkt Ihre Nachricht oder Ihre benutzerdefinierten Nachrichten je Zustellungsprotokoll ein.

      Wenn Ihr Thema über einen Anzeigenamen verfügt, fügt Amazon SNS diesen der Nachricht hinzu, wodurch sich die Nachrichtenlänge verlängert. Die Länge des Anzeigenamens setzt sich aus der Anzahl der Zeichen des Namens sowie zwei Zeichen für die spitze Klammer (**>**) und von Amazon SNS hinzugefügte Leerzeichen zusammen.

      Weitere Informationen über das Größenkontingent für SMS-Nachrichten finden Sie unter Veröffentlichen von SMS-Nachrichten auf einem Mobiltelefon mithilfe von Amazon SNS.
5. (Optional) Fügen Sie für Nachrichtenattribute Nachrichtenmetadaten wie Zeitstempel, Signaturen und hinzu. IDs
6. Wählen Sie Publish message (Nachricht veröffentlichen) aus. Amazon SNS sendet die SMS-Nachricht und zeigt eine Erfolgsmeldung an.

Senden einer Nachricht an ein Thema mithilfe des AWS SDKs

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.
- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

Java

SDK für Java 2.x

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
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 "";
    }
}
```

Abonnieren eines Endpunkts für ein Thema.

```
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);
        }
    }
}
```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```
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);
        }
    }
}
```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```
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);
        }
    }
}
```

Veröffentlichen von SMS-Nachrichten auf einem Mobiltelefon mithilfe von Amazon SNS

Sie können Amazon SNS verwenden, um SMS-Nachrichten direkt an ein Mobiltelefon zu senden, ohne die Telefonnummer für ein Amazon SNS-Thema zu abonnieren.

> **Note**
>
> Das Abonnieren von Telefonnummern zu einem Thema kann nach wie vor nützlich sein, wenn Sie jede Nachricht in mehreren Telefonnummern zugleich senden möchten. Weitere Anleitungen zum Veröffentlichen einer SMS-Nachricht in einem Thema finden Sie unter Veröffentlichen von SMS-Nachrichten zu einem Amazon SNS SNS-Thema.

Wenn Sie eine Nachricht senden, können Sie überprüfen, ob der Versand in Bezug auf Kosten und Verlässlichkeit optimiert ist. Sie können auch eine Sender-ID oder Ursprungsnummer angeben. Wenn Sie die Nachricht programmgesteuert über die Amazon SNS SNS-API oder die senden AWS SDKs, können Sie einen Höchstpreis für die Nachrichtenzustellung angeben.

Eine SMS-Nachricht kann maximal 140 Byte groß sein, wobei das Zeichenkontingent vom Codierungsschema abhängig ist. Die Anzahl der Zeichen einer SMS-Nachricht ist entsprechend wie folgt begrenzt:

- 160 GSM-Zeichen
- 140 ASCII-Zeichen

- 70 UCS-2-Zeichen

Wenn Sie eine Nachricht veröffentlichen, die das Größenkontingent überschreitet, sendet Amazon SNS mehrere Teilnachrichten, die jeweils dem Größenkontingent entsprechen. Nachrichten werden niemals inmitten eines Wortes geteilt, sondern immer zwischen zwei Wörtern. Für die gesamte veröffentlichte und in mehreren Teilen versendete SMS-Nachricht gilt ein Größenkontingent von 1600 Byte.

Wenn Sie eine SMS-Nachricht senden, geben Sie die Telefonnummer im E.164-Format an – eine Standardnummernstruktur für die Telefonnummerierung, die für die internationale Telekommunikation verwendet wird. Telefonnummern in diesem Format bestehen aus maximal 15 Zeichen sowie einem vorangestellten Plus-Zeichen (+) und der Ländervorwahl. Beispielsweise wird eine US-Telefonnummer im E.164-Format als +1 0100 angezeigt. XXX555

Senden einer Nachricht (Konsole)

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Konsolenmenü eine Region aus, die SMS-Nachrichten unterstützt.
3. Wählen Sie im Navigationsbereich Text-Messaging (SMS) aus.
4. Wählen Sie auf der Seite Text-Messaging (SMS) Textnachricht veröffentlichen.
5. Wählen Sie auf der Seite Textnachricht veröffentlichen für Nachrichtentyp eines der Folgenden:
   - Werbenachrichten – Unkritische Nachrichten, wie beispielsweise Marketing-Nachrichten.
   - Transaktionsnachrichten: Sensible Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-Authentifizierung (MFA).

   > **Note**
   >
   > Diese Einstellung auf Nachrichtenebene setzt den Standardnachrichtentyp auf Kontoebene außer Kraft. Auf der Seite Textnachrichten (SMS) im Abschnitt Text-Messaging-Einstellungen können SIe einen Standardnachrichtentyp auf Kontoebene festlegen.

   Informationen zu den Gebühren für Aktions- und Transaktionsnachrichten finden Sie unter Weltweite SMS-Preise.

6. Geben Sie unter Zieltelefonnummer die Telefonnummer ein, an die Sie die Nachricht senden möchten.
7. Geben Sie im Feld Message (Nachricht) die Art der zu versendenden Nachricht ein.
8. (Optional) Legen Sie unter Ursprungsidentitäten fest, wie Sie sich selbst gegenüber Ihren Empfängern identifizieren möchten:

   - Geben Sie im Feld Sender ID (Sender-ID) eine Kunden-ID ein, die 3 bis 11 alphanumerische Zeichen enthält, einschließlich mindestens einem Buchstaben und ohne Leerzeichen. Die Sender-ID wird auf dem Empfänger-Gerät als Sender der Nachricht angezeigt. Sie können z. B. Ihre Unternehmensmarke verwenden, um die Nachrichtenquelle leichter erkennbar zu machen.

     Die Support für den Absender IDs ist je nach Land und and/or Region unterschiedlich. Beispiel: Für Nachrichten, die an eine US-Telefonnummer gesendet werden, wird keine Sender-ID angezeigt. Informationen zu den Ländern und Regionen, die Sender unterstützen IDs, finden Sie AWS Endbenutzer-Messaging SMS im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Unterstützte Länder und Regionen für SMS-Nachrichten mit.

     Wenn Sie keine Sender-ID angeben, wird eines der folgenden Elemente als Ursprungsidentität angezeigt:

     - In Ländern, die Langwahlnummern unterstützen, wird die Langwahlnummer angezeigt.
     - In Ländern, in denen nur Absender unterstützt IDs werden, wird NOTICE angezeigt.

     Diese Sender-ID auf Nachrichtenebene ersetzt die standardmäßige Sender-ID, die Sie auf der Seite Text messaging preferences (SMS-Präferenzen) konfigurieren.

   - Um eine Ursprungsnummer festzulegen, geben Sie eine Zeichenfolge von 5-14 Nummern ein, die als Telefonnummer des Senders auf dem Gerät des Empfängers angezeigt werden soll. Diese Zeichenfolge muss mit einer Ausgangsnummer übereinstimmen, die in Ihrem AWS-Konto Zielland konfiguriert ist. Bei der Ausgangsnummer kann es sich um eine 10DLC-Nummer, eine gebührenfreie Nummer, einen person-to-person Langcode oder eine Kurzvorwahl handeln. Weitere Informationen finden Sie unter Herkunftsidentitäten für Amazon SNS SNS-SMS-Nachrichten.

     Wenn Sie keine Ursprungsnummer angeben, wählt Amazon SNS eine Ursprungsnummer aus, die für die SMS-Textnachricht verwendet werden soll, basierend auf Ihrer AWS-Konto - Konfiguration.

9. Wenn Sie SMS-Nachrichten an Empfänger in Indien senden, erweitern Sie Länderspezifische Attribute und legen Sie die folgenden Attribute fest:

   - Entity-ID – Die Entity ID oder Principal Entity (PE) -ID für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige Zeichenfolge von 1 bis 50 Zeichen, die die Telecom Regulatory Authority of India (TRAI) zur Identifizierung der Entität bereitstellt, die Sie bei der TRAI registriert haben.
   - ID der Vorlage – Die Vorlagen-ID für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige, von TRAI bereitgestellte Zeichenfolge mit 1 bis 50 Zeichen, die die Vorlage identifiziert, die Sie bei TRAI registriert haben. Die Vorlagen-ID muss der Sender-ID zugeordnet sein, die Sie für die Nachricht angegeben haben.

   Weitere Informationen zum Senden von SMS-Nachrichten an Empfänger in Indien, Indien, Indien, finden Sie im AWS Endbenutzer-Messaging SMS Benutzerhandbuch.
10. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

> **Tip**
>
> Um SMS-Nachrichten von einer Ursprungsnummer zu senden, können Sie auch Ursprungsnummern im Navigationsbereich in der Amazon SNS-Konsole eingeben. Wählen Sie eine Ursprungsnummer, die in der Spalte Funktionen SMS enthält und wählen Sie dann Textnachricht veröffentlichen.

Eine Nachricht senden (AWS SDKs)

Um eine SMS-Nachricht mit einem der zu senden AWS SDKs, verwenden Sie den API-Vorgang in diesem SDK, der der `Publish` Anfrage in der Amazon SNS SNS-API entspricht. Mit dieser Anfrage können Sie eine SMS-Nachricht direkt an eine Telefonnummer senden. Sie können auch den `MessageAttributes`-Parameter verwenden, um Werte für die folgenden Attributnamen zu definieren:

`AWS.SNS.SMS.SenderID`

Eine benutzerdefinierte ID, die 3 bis 11 alphanumerische Zeichen oder Bindestriche (-) enthält, einschließlich mindestens einem Buchstaben und ohne Leerzeichen. Die Sender-ID wird auf dem

Empfänger-Gerät als Sender der Nachricht angezeigt. Sie können z. B. Ihre Unternehmensmarke verwenden, um die Nachrichtenquelle leichter erkennbar zu machen.

Die Support für den Absender IDs ist je nach Land oder Region unterschiedlich. Beispiel: Für Nachrichten, die an eine US-Telefonnummer gesendet werden, wird keine Sender-ID angezeigt. Eine Liste der Länder oder Regionen, die Absender unterstützen IDs, finden Sie AWS Endbenutzer-Messaging SMS im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Unterstützte Länder und Regionen für SMS-Nachrichten mit.

Wenn Sie keine Sender-ID angeben, erscheint eine Langwahlnummer als Sender-ID in den unterstützten Ländern oder Regionen. Für Länder oder Regionen, die eine alphabetische Sender-ID erfordern, wird NOTICE als Sender-ID angezeigt.

Dieses Attribut auf Nachrichtenebene ersetzt das `DefaultSenderID`-Attribut auf Kontoebene, das Sie bei der Nutzung der `SetSMSAttributes`-Anfrage festlegen.

`AWS.MM.SMS.OriginationNumber`

Eine benutzerdefinierte Zeichenfolge von 5 bis 14 Zahlen, die ein optionales vorzeitiges Pluszeichen (+) enthalten. Diese Zahlenfolge wird als Telefonnummer des Senders auf dem empfangenden Gerät angezeigt. Die Zeichenfolge muss mit einer Absendernummer übereinstimmen, die in Ihrem AWS Konto für das Zielland konfiguriert ist. Bei der Ausgangsnummer kann es sich um eine 10DLC-Nummer, eine gebührenfreie Nummer, einen person-to-person (P2P) -Langcode oder einen Kurzcode handeln. Weitere Informationen finden Sie im Benutzerhandbuch unter Telefonnummern.AWS Endbenutzer-Messaging SMS

Wenn Sie keine Ursprungsnummer angeben, wählt Amazon SNS auf der Grundlage Ihrer AWS Kontokonfiguration eine Ursprungsnummer aus.

`AWS.SNS.SMS.MaxPrice`

Höchstpreis in USD, den Sie zu zahlen bereit sind, um die SMS-Nachricht zu senden. Wenn Amazon SNS festlegt, dass das Senden der Nachricht zu Kosten führen würde, die Ihren Maximalpreis übersteigen, wird die Nachricht nicht gesendet.

Dieses Attribut hat keine Auswirkung, wenn Ihre month-to-date SMS-Kosten das für das Attribut festgelegte Kontingent bereits überschritten haben. `MonthlySpendLimit` Sie können das `MonthlySpendLimit`-Attribut mit der `SetSMSAttributes`-Anfrage festlegen.

Wenn Sie die Nachricht an ein Amazon SNS-Thema senden, gilt bei jeder Nachrichtenübertragung an alle Telefonnummern, die das Thema abonniert haben, der Höchstpreis.

`AWS.SNS.SMS.SMSType`

Der Typ der gesendeten Nachricht:

- **`Promotional`** (Standard): Unkritische Nachrichten, wie beispielsweise Marketing-Nachrichten.
- **`Transactional`** – Sensible Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-Authentifizierung.

Dieses Attribut auf Nachrichtenebene ersetzt das `DefaultSMSType`-Attribut auf Kontoebene, das Sie bei der Nutzung der `SetSMSAttributes`-Anfrage festlegen.

`AWS.MM.SMS.EntityId`

Dieses Attribut ist nur für das Senden von SMS-Nachrichten an Empfänger in Indien erforderlich.

Das ist Ihre Entity ID oder Principal Entity (PE) -ID für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige Zeichenfolge von 1 bis 50 Zeichen, die die Telecom Regulatory Authority of India (TRAI) zur Identifizierung der Entität bereitstellt, die Sie bei der TRAI registriert haben.

`AWS.MM.SMS.TemplateId`

Dieses Attribut ist nur für das Senden von SMS-Nachrichten an Empfänger in Indien erforderlich.

Dies ist Ihre Vorlage für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige, von TRAI bereitgestellte Zeichenfolge mit 1 bis 50 Zeichen, die die Vorlage identifiziert, die Sie bei TRAI registriert haben. Die Vorlagen-ID muss der Sender-ID zugeordnet sein, die Sie für die Nachricht angegeben haben.

Senden einer Nachricht

In den folgenden Codebeispielen wird veranschaulicht, wie Sie SMS-Nachrichten über Amazon SNS veröffentlichen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * 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();
```

```
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für C++ -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Einstellungen für SMS-Nachrichten in Amazon SNS einrichten

Verwenden Sie Amazon SNS, um Einstellungen für SMS-Messaging anzugeben. Sie können beispielsweise festlegen, ob Zustellung zu Kosten oder zur Zuverlässigkeit optimiert werden, Ihr monatliches Ausgabelimit, wie Zustellungen protokolliert werden und ob Sie tägliche SMS-Nutzungsberichte erhalten möchten.

Diese Einstellungen gelten für jede SMS-Nachricht, die Sie von Ihrem Konto senden. Sie können jedoch auch beim Senden einer einzelnen Nachricht überschrieben werden. Weitere Informationen finden Sie unter Veröffentlichen von SMS-Nachrichten auf einem Mobiltelefon mithilfe von Amazon SNS.

Festlegen von SMS-Messaging-Einstellungen mithilfe der AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie eine Region aus, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Mobil und dann Textnachrichten (SMS) aus.
4. Wählen Sie auf der Seite Mobile text messaging (SMS) (Mobile Texttnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.
5. Führen Sie auf der Seite Edit text messaging preferences (SMS-Einstellungen bearbeiten) im Abschnitt Details Folgendes durch:

    a. Wählen Sie für Default message type (Standardnachrichtentyp) eine der folgenden Optionen aus:

        - Werbenachrichten – Nicht kritische Nachrichten (zum Beispiel Marketing). Amazon SNS optimiert die Übertragung von Nachrichten im Hinblick auf möglichst niedrige Kosten.
        - Transaktionsnachrichten (Standard) – Kritische Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-

Authentifizierung (MFA). Amazon SNS optimiert die Übertragung von Nachrichten im Hinblick auf höchste Zuverlässigkeit.

Informationen zu den Gebühren für Aktions- und Transaktionsnachrichten finden Sie unter Global SMS Pricing.

b. (Optional) Geben Sie für Account spend limit (Kontoausgabenlimit) den Betrag in US-Dollar ein, den Sie pro Kalendermonat für SMS-Nachrichten ausgeben möchten.

> **Important**
>
> - Standardmäßig ist das Ausgabenkontingent auf 1,00 US-Dollar festgelegt. Wenn Sie das Service-Kontingent erhöhen möchten, senden Sie eine Anfrage.
> - Wenn die in der Konsole festgelegte Menge Ihr Service-Kontingent überschreitet, stoppt Amazon SNS die Veröffentlichung von SMS-Nachrichten.
> - Da Amazon SNS ein verteiltes System ist, beendet es das Senden von SMS-Nachrichten innerhalb von einigen Minuten nach der Überschreitung des Ausgabenkontingents. Wenn Sie während dieses Zeitraums weiter SMS-Nachrichten senden, können Kosten entstehen, die Ihr Kontingent überschreiten.

6. (Optional) Geben Sie im Feld Default sender ID (Standard-Sender-ID) eine benutzerdefinierte ID ein, wie z. B. Ihr Unternehmen, die als Sender des Empfängergeräts angezeigt wird.

> **Note**
>
> Die Support für den Absender IDs ist je nach Land unterschiedlich.

7. (Optional) Geben Sie den Namen des Amazon S3 Bucket Namen für Nutzungsberichte ein.

> **Note**
>
> Die Amazon S3 S3-Bucket-Richtlinie muss Schreibzugriff auf Amazon SNS gewähren.

8. Wählen Sie Änderungen speichern.

Einstellungen festlegen ()AWS SDKs

Um Ihre SMS-Einstellungen mithilfe einer der festzulegen AWS SDKs, verwenden Sie die Aktion in diesem SDK, die der `SetSMSAttributes` Anfrage in der Amazon SNS SNS-API entspricht. Mit dieser Anforderung können Sie Werte zu den verschiedenen SMS-Attributen zuweisen – beispielsweise das monatliche Ausgabenkontingent und Ihren Standard-SMS-Typ (Werbung oder Transaktionen). Informationen zu allen SMS-Attributen finden Sie unter Set SMSAttributes in der Amazon Simple Notification Service API-Referenz.

Die folgenden Code-Beispiele zeigen, wie `SetSMSAttributes` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

So verwenden Sie Amazon SNS, um das SMSType Standardattribut festzulegen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So legen Sie SMS-Nachrichtenattribute fest

Im folgenden `set-sms-attributes`-Beispiel wird die standardmäßige Absender-ID für SMS-Nachrichten auf `MyName` festgelegt.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für PHP API-Referenz.

Einstellung der SMS-Nachrichteneinstellungen für die länderspezifische Zustellung

Sie können Ihren SMS-Verkehr verwalten und kontrollieren, indem Sie Nachrichten nur an bestimmte Zielländer senden. Dadurch wird sichergestellt, dass Ihre Nachrichten nur in zugelassene Länder gesendet werden, wodurch unerwünschte SMS-Gebühren vermieden werden. Die folgenden Anweisungen verwenden die Protect-Konfiguration von Amazon Pinpoint, um die Länder anzugeben, die Sie zulassen oder blockieren möchten.

1. Öffnen Sie die AWS SMS Konsole unter. https://console.aws.amazon.com/sms-voice/
2. Wählen Sie im Navigationsbereich unter Übersicht im Abschnitt Schnellstart die Option Create a protect configuration aus.
3. Geben Sie unter Protect-Konfigurationsdetails einen unternehmensfreundlichen Namen für Ihre Protect-Konfiguration ein (z. B. Allow-Only-AU).
4. Wählen Sie unter SMS-Länderregeln das Kontrollkästchen Region/Land aus, um das Senden von Nachrichten in alle unterstützten Länder zu blockieren.
5. Deaktivieren Sie die Kontrollkästchen für die Länder, in die Sie Nachrichten senden möchten. Um beispielsweise nur Nachrichten nach Australien zuzulassen, deaktivieren Sie das Kontrollkästchen für Australien.

6. Wählen Sie im Abschnitt Konfigurationszuordnungen schützen unter Zuordnungstyp die Option Kontostandard aus. Dadurch wird sichergestellt, dass sich die AWS Endbenutzer-Messaging SMS Protect-Konfiguration auf alle Nachrichten auswirkt, die über Amazon SNS, Amazon Cognito und den Amazon Pinpoint `SendMessages`API-Aufruf gesendet werden.
7. Wählen Sie Create Protect configuration aus, um Ihre Einstellungen zu speichern.

   Die folgende Bestätigungsmeldung wird angezeigt:

   ```
   Success Protect configuration protect-abc0123456789 has been created.
   ```

8. Melden Sie sich bei der Amazon-SNS-Konsole an.
9. Veröffentlichen Sie eine Nachricht in einem der blockierten Länder, z. B. Indien.

   Die Nachricht wird nicht zugestellt. Sie können dies in den Zustellungsfehlerprotokollen mit überprüfen CloudWatch. Suchen Sie nach einer Protokollgruppe sns/region/AccountID/DirectPublishToPhoneNumber/Failurefür eine Antwort, die dem folgenden Beispiel ähnelt:

   ```
   {
   "notification": {
   "messageId": "bd59a509-XXXX-XXXX-82f8-fbdb8cb68217",
   "timestamp": "YYYY-MM-DD XX:XX:XX.XXXX“
   },
   "delivery": {
   "destination": "+91XXXXXXXXXX",
   "smsType": "Transactional",
   "providerResponse": "Cannot deliver message to the specified destination country",
   "dwellTimeMs": 85
   },
   "status": "FAILURE"
   }
   ```

Verwaltung von Amazon SNS SNS-Telefonnummern und Abonnements

Amazon SNS bietet mehrere Optionen, mit denen Sie verwalten können, wer von Ihrem Konto SMS-Nachrichten erhält. Unter Berücksichtigung einer begrenzten Häufigkeit können Sie Telefonnummern, die den Empfang von SMS-Nachrichten von Ihrem Konto deaktiviert haben, hinzufügen. Um Nachrichten nicht länger an SMS-Abonnements zu senden, können Sie die Abonnements oder die Themen, die an diese Abonnements veröffentlichen, entfernen.

Deaktivieren des Empfangs von SMS-Nachrichten

Sofern dies nach lokalen Gesetzen und Vorschriften (z. B. in den USA und Kanada) erforderlich ist, können SMS-Empfänger ihre Geräte verwenden, um sich abzumelden, indem sie auf die Nachricht mit einer der folgenden Antworten antworten:

- ARRET (Französisch)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

Um sich abzumelden, muss der Empfänger auf dieselbe Absendernummer antworten, die Amazon SNS für die Zustellung der Nachricht verwendet hat. Nach der Abmeldung erhält der Empfänger keine SMS-Nachrichten mehr von Ihnen, AWS-Konto es sei denn, Sie geben die Telefonnummer an.

Wenn die Telefonnummer ein Amazon SNS SNS-Thema abonniert hat, wird das Abonnement durch die Abmeldung nicht entfernt, aber SMS-Nachrichten können dieses Abonnement nicht zugestellt werden, es sei denn, Sie haben die Telefonnummer aktiviert.

Verwaltung von Telefonnummern und Abonnements mit der Amazon SNS SNS-Konsole

Sie können über die Amazon SNS-Konsole steuern, welche Telefonnummern SMS-Nachrichten von Ihrem Konto empfangen.

Anmeldung einer Telefonnummer, für die die Amazon SNS-Konsole deaktiviert wurde

Sie können sehen, welche Telefonnummern für den Empfang von SMS-Nachrichten von Ihrem Konto deaktiviert wurden, und Sie können diese Telefonnummern aktivieren, um weiterhin Nachrichten an diese Personen zu senden.

Sie können eine Telefonnummer nur einmal alle 30 Tage aktivieren.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
4. Auf der Seite SMS (Mobile Textnachrichten) werden im Abschnitt Abgemeldete Telefonnummern angezeigt.
5. Aktivieren Sie das Kontrollkästchen für die Telefonnummer, für die Sie sich anmelden möchten, und wählen Sie Anmelden aus. Die Telefonnummer ist nicht mehr deaktiviert und empfängt SMS-Nachrichten, die Sie an sie senden.

Löschen eines SMS-Abonnements: Die Amazon SNS SNS-Konsole

Löschen Sie ein SMS-Abonnement, um nicht länger SMS-Nachrichten über Veröffentlichungen zu Ihren Themen an diese Telefonnummer zu senden.

1. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
2. Aktivieren Sie die Kontrollkästchen für die Abonnements, die Sie löschen möchten. Wählen Sie anschließend Actions (Aktionen) und Delete Subscriptions (Abonnements löschen) aus.
3. Wählen Sie im Fenster Löschen Löschen aus. Amazon SNS löscht das Abonnement und zeigt eine Erfolgsmeldung an.

Löschen eines Themas in der Amazon SNS SNS-Konsole

Löschen Sie ein Thema, wenn Sie nicht länger Nachrichten an die Endpunkte veröffentlichen möchten, die dieses Thema abonniert haben.

1. Wählen Sie im Navigationsbereich Topics (Themen) aus.
2. Aktivieren Sie die Kontrollkästchen der Themen, die Sie löschen möchten. Wählen Sie anschließend Actions (Aktionen) und Delete Topics (Themen löschen) aus.
3. Wählen Sie im Fenster Löschen Löschen aus. Amazon SNS löscht das Thema und zeigt eine Erfolgsmeldung an.

Verwaltung von Telefonnummern und Abonnements mithilfe des SDK AWS

Sie können den verwenden AWS SDKs , um programmatische Anfragen an Amazon SNS zu stellen und zu verwalten, welche Telefonnummern SMS-Nachrichten von Ihrem Konto empfangen können.

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Dateien mit gemeinsam genutzten Konfigurationen und Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Alle Telefonnummern, für die Sie sich abgemeldet haben, mithilfe des SDK anzeigen AWS

Um alle abgemeldeten Telefonnummern einzusehen, senden Sie eine `ListPhoneNumbersOptedOut` Anfrage mit der Amazon SNS SNS-API.

Die folgenden Code-Beispiele zeigen, wie `ListPhoneNumbersOptedOut` verwendet wird.

CLI

AWS CLI

So führen Sie Abmeldungen für SMS-Nachrichten auf

Das folgende `list-phone-numbers-opted-out`-Beispiel listet die Telefonnummern auf, bei denen der Empfang von SMS-Nachrichten abbestellt wurde.

```
aws sns list-phone-numbers-opted-out
```

Ausgabe:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Einzelheiten zur API finden Sie unter ListPhoneNumbersOptedOutBefehlsreferenz.AWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK für PHP API-Referenz.

Mithilfe des SDK wird überprüft, ob eine Telefonnummer deaktiviert wurde AWS

Um zu überprüfen, ob eine Telefonnummer deaktiviert ist, reichen Sie eine `CheckIfPhoneNumberIsOptedOut` Anfrage mit der Amazon SNS SNS-API ein.

Die folgenden Code-Beispiele zeigen, wie `CheckIfPhoneNumberIsOptedOut` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So überprüfen Sie SMS-Nachrichten-Abmeldungen für eine Telefonnummer

Im folgenden `check-if-phone-number-is-opted-out` Beispiel wird geprüft, ob die angegebene Telefonnummer den Empfang von SMS-Nachrichten vom AWS Girokonto deaktiviert hat.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Ausgabe:

```
{
    "isOptedOut": false
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für PHP API-Referenz.

Anmeldung einer Telefonnummer, die über die Amazon SNS-API deaktiviert wurde

Um sich für eine Telefonnummer anzumelden, reichen Sie eine `OptInPhoneNumber` Anfrage mit der Amazon SNS SNS-API ein.

Sie können sich nur einmal alle 30 Tage für eine Telefonnummer anmelden.

Löschen eines SMS-Abonnements mithilfe des SDK AWS

Um ein SMS-Abonnement von einem Amazon SNS-Thema zu löschen, müssen Sie zunächst anhand einer `ListSubscriptions`-Anfrage über das Amazon SNS-API den ARN des Abonnements abrufen und anschließend den ARN einer `Unsubscribe`-Anfrage übergeben.

Die folgenden Code-Beispiele zeigen, wie `Unsubscribe` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Melden Sie sich mit einem Abonnement-ARN von einem Thema ab.

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

- Details zu API finden Sie unter Abmelden in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So melden Sie sich von einem Thema ab

Im folgenden `unsubscribe`-Beispiel wird das angegebene Abonnement aus einem Thema gelöscht.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Details finden Sie unter Unsubscribe in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {
```

```
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Details zu API finden Sie unter Unsubscribe (Abmelden) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Details zu API finden Sie unter Abmelden in der AWS API-Referenz zu SDK for Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Details zu API finden Sie unter Abmelden in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Einzelheiten zur API finden Sie unter Unsubscribe in AWS SDK für die Swift-API-Referenz.

Löschen eines Themas mithilfe des AWS SDK

Um ein Thema und alle seine Abonnements zu löschen, müssen Sie zunächst anhand einer `ListTopics`-Anfrage über die Amazon SNS-API den ARN des Abonnements einholen und anschließend den ARN einer `DeleteTopic`-Anfrage übergeben.

Die folgenden Code-Beispiele zeigen, wie `DeleteTopic` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-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;
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

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

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



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

```
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

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

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

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

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Swift.

Überwachung der SMS-Aktivitäten von Amazon SNS

Durch die Überwachung der SMS-Aktivitäten behalten Sie Zieltelefonnummern, erfolgreiche oder fehlgeschlagene Zustellungen, Gründe für Ausfälle, Kosten und andere Informationen im Auge. Amazon SNS hilft, indem es Statistiken in der Konsole zusammenfasst, Informationen an Amazon CloudWatch sendet und tägliche SMS-Nutzungsberichte an einen von Ihnen angegebenen Amazon S3 S3-Bucket sendet.

Amazon SNS SNS-SMS-Versandstatistiken anzeigen

Sie können Statistiken zu Ihren aktuellen SMS-Zustellungen über die Amazon SNS-Konsole anzeigen.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
4. Zeigen Sie auf der Seite Text messaging (SMS) im Abschnitt Account stats (Kontostatistik) die Diagramme zu den Zustellungen Ihrer transaktions- und werbebezogenen SMS-Nachrichten an. Jedes Diagramm zeigt die folgenden Daten für die letzten 15 Tage:

   - Zustellungsrate (Prozentsatz der erfolgreichen Zustellungen)
   - Gesendet (Anzahl der Zustellversuche)
   - Fehlgeschlagen (Anzahl der fehlgeschlagenen Zustellungen)

Auf dieser Seite können Sie auch die Schaltfläche Usage (Nutzung) wählen, um zum Amazon S3-Bucket zu wechseln, in dem Ihre täglichen Nutzungsberichte gespeichert sind. Weitere Informationen finden Sie unter Abonnieren der täglichen SMS-Nutzungsberichte von Amazon SNS.

Amazon SNS SMS-Versandüberwachung mit CloudWatch Amazon-Metriken und Protokollen

Sie können Amazon CloudWatch und Amazon CloudWatch Logs verwenden, um Ihre SMS-Nachrichtenzustellungen zu überwachen.

CloudWatch Amazon-Metriken anzeigen

Amazon SNS sammelt automatisch Messwerte über Ihre SMS-Nachrichtenzustellungen und leitet sie an Amazon weiter. CloudWatch Sie können diese Messwerte überwachen und Alarme einrichten, CloudWatch um Sie zu benachrichtigen, wenn eine Metrik einen Schwellenwert überschreitet. Sie können beispielsweise CloudWatch Messwerte überwachen, um Ihre SMS-Zustellrate und Ihre month-to-date SMS-Gebühren zu ermitteln.

Informationen zur Überwachung von CloudWatch Messwerten, zur Einstellung von CloudWatch Alarmen und zu den verfügbaren Messwerttypen finden Sie unterÜberwachung von Amazon SNS SNS-Themen mit CloudWatch.

CloudWatch Protokolle anzeigen

Sie können Informationen über erfolgreiche und erfolglose SMS-Nachrichtenzustellungen sammeln, indem Sie Amazon SNS aktivieren, in Amazon CloudWatch Logs zu schreiben. Für jede SMS-Nachricht, die Sie senden, schreibt Amazon SNS ein Protokoll, das die Kosten für die Nachricht, den

Erfolgs- oder Fehlerstatus, den Grund für das Fehlschlagen (wenn die Nachricht fehlgeschlagen ist), die Leerlaufzeit für die Nachricht und andere Informationen enthält.

Um CloudWatch Protokolle für Ihre SMS-Nachrichten zu aktivieren und anzuzeigen

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
4. Wählen Sie auf der Seite Mobile text messaging (SMS) (Mobile Texttnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.
5. Erweitern Sie auf der nächsten Seite den Abschnitt Protokollierung des Zustellungsstatus.
6. Geben Sie für Success sample rate den Prozentsatz der erfolgreichen SMS-Lieferungen an, für die Amazon SNS CloudWatch Protokolle in Logs schreibt. Zum Beispiel:

   - Um nur Protokolle für fehlgeschlagene Zustellungen zu schreiben, setzen Sie diesen Wert auf 0.
   - Um Protokolle für 10 % der erfolgreichen Zustellungen zu schreiben, setzen Sie den Wert auf 10.

   Wenn Sie keinen Prozentsatz angeben, schreibt Amazon SNS Protokolle für alle erfolgreichen Zustellungen.
7. Gehen Sie wie folgt vor, um die angeforderten Genehmigungen bereitzustellen:

   - Zum Erstellen einer neuen Servicerolle wählen Sie Neue Servicerolle erstellen und dann Neue Rollen erstellen. Wählen Sie auf der nächsten Seite Erlauben, um Amazon SNS Schreibzugriff auf die Ressourcen Ihres Kontos zu geben.
   - Um eine bestehende Servicerolle zu verwenden, wählen Sie Vorhandene Servicerolle verwenden und fügen Sie dann den ARN-Namen im Feld IAM-Rolle für erfolgreiche und fehlgeschlagene Übermittlungen.

     Die angegebene Servicerolle muss Schreibzugriff auf die Ressourcen Ihres Kontos ermöglichen. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Creating a role for an AWS service im IAM-Benutzerhandbuch.
8. Wählen Sie Änderungen speichern.

9. Zurück auf der Seite Text-Messaging (SMS) gehen Sie zum Abschnitt Zustellungsstatusprotokolle, um verfügbare Protokolle anzuzeigen.

> **Note**
>
> Je nach Carrier der Zieltelefonnummer kann es bis zu 72 Stunden dauern, bis Zustellungsprotokolle in der Amazon-SNS-Konsole angezeigt werden.

Beispielprotokoll für eine erfolgreiche SMS-Zustellung

Das Zustellungsstatusprotokoll für eine erfolgreiche SMS-Zustellung ähnelt dem folgenden Beispiel:

```
{
    "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"
}
```

Beispielprotokoll für eine fehlgeschlagene SMS-Zustellung

Das Zustellungsstatusprotokoll für eine fehlgeschlagene SMS-Zustellung ähnelt dem folgenden Beispiel:

```
{
    "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"
}
```

Gründe für das Fehlschlagen der SMS-Zustellung

Der Grund für das Fehlschlagen wird mit dem `providerResponse`-Attribut angegeben. SMS-Nachrichten können aus den folgenden Gründen fehlschlagen:

- Telefonnetzbetreiber hat die Nachricht als Spam blockiert
- Ziel befindet sich auf einer blockierten Liste
- Telefonnummer ist ungültig
- Nachrichtentext ist ungültig
- Telefonnentzbetreiber hat die Nachricht blockiert
- Telefonnetzbetreiber ist derzeit nicht erreichbar/nicht verfügbar
- Telefon blockiert SMS
- Telefon befindet sich auf einer blockierten Liste
- Telefon ist derzeit nicht erreichbar/nicht verfügbar
- Telefonnummer lehnt Empfang ab
- Zustellung würde den Maximalpreis überschreiten
- Unbekannter Fehler versucht das Telefon zu erreichen

Abonnieren der täglichen SMS-Nutzungsberichte von Amazon SNS

Sie können Ihre SMS-Zustellungen überwachen, indem Sie tägliche Nutzungsberichte von Amazon SNS abonnieren. Für jeden Tag, an dem Sie mindestens eine SMS-Nachricht senden, sendet

Amazon SNS einen Nutzungsbericht als CSV-Datei an den angegebenen Amazon S3-Bucket. Es dauert 24 Stunden, bis der SMS-Nutzungsbericht im Amazon S3 S3-Bucket verfügbar ist.

Informationen im täglichen Nutzungsbericht

Der Nutzungsbericht umfasst die folgenden Informationen für jede SMS-Nachricht, die Sie von Ihrem Konto senden.

Beachten Sie, dass der Bericht keine Nachrichten enthält, die an Empfänger gesendet werden, die sich vom Nachrichtenempfang abgemeldet haben.

- Zeitpunkt der Veröffentlichung für die Nachricht (in UTC)
- Nachrichten-ID
- Zieltelefonnummer
- Nachrichtentyp
- Zustellungsstatus
- Kosten für die Nachricht (in US-Dollar)
- Segmentnummer (eine Nachricht wird in mehrere Segmente aufgeteilt, wenn sie für eine einzelne Nachricht zu lang ist)
- Gesamtanzahl der Segmente

> **Note**
>
> Wenn Amazon SNS die Segmentnummer nicht erhalten hat, setzen wir deren Wert auf Null.

Abonnieren der täglichen Nutzungsberichte

Um tägliche Nutzungsberichte zu abonnieren, müssen Sie einen Amazon S3-Bucket mit den entsprechenden Berechtigungen erstellen.

So erstellen Sie einen Amazon-S3-Bucket für Ihre täglichen Nutzungsberichte

1. Melden Sie sich über den AWS-Konto , der SMS-Nachrichten sendet, bei der Amazon S3-Konsole an.
2. Wählen Sie Create Bucket (Bucket erstellen) aus.

3. Unter Bucket Name (Bucket-Name) empfiehlt es sich, einen für Ihr Konto und Ihre Organisation eindeutigen Namen einzugeben. Verwenden Sie zum Beispiel das Muster `<my-bucket-prefix>-<account_id>-<org-id>`.

   Weitere Informationen über Konventionen und Einschränkungen für Bucket-Namen finden Sie unter Regeln für die Bucket-Benennung im Benutzerhandbuch zu Amazon Simple Storage Service.

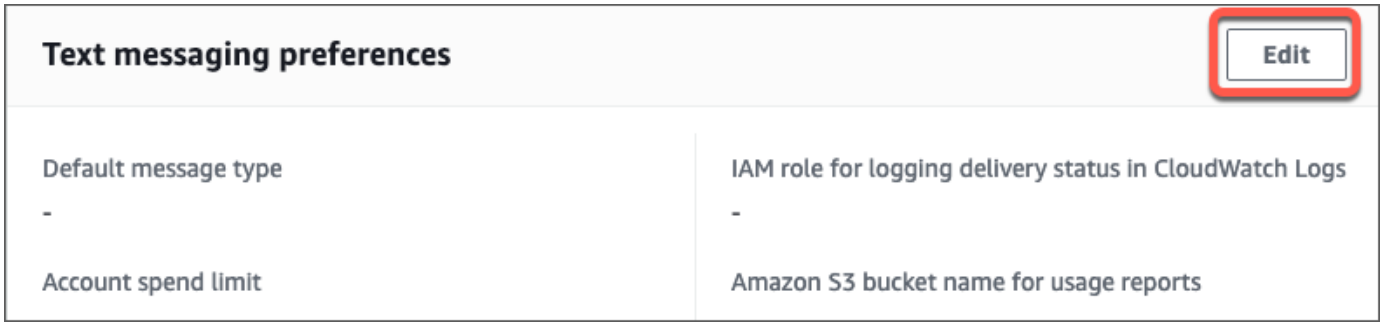
4. Wählen Sie Erstellen aus.
5. Wählen Sie in der Tabelle Alle Buckets den Bucket aus.
6. Wählen Sie in der Registerkarte Genehmigungen Bucket-Richtlinie.
7. Geben Sie im Fenster Bucket Policy Editor (Bucket-Richtlinieneditor) eine Richtlinie ein, die den Amazon SNS-Serviceprinzipal berechtigt, in Ihren Bucket zu schreiben. Ein Beispiel finden Sie unter Beispiel einer Bucket-Richtlinie.

   Wenn Sie die Beispielrichtlinie verwenden, denken Sie daran, sie durch den Bucket-Namen zu *`my-s3-bucket`* ersetzen, den Sie in Schritt 3 ausgewählt haben.

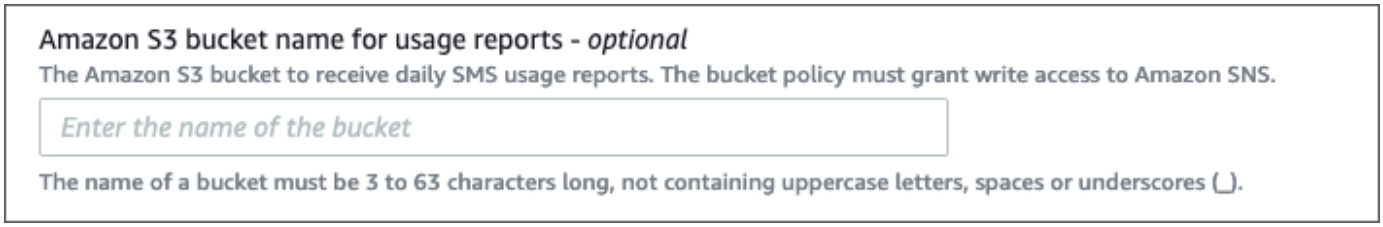
8. Wählen Sie Speichern.

So abonnieren Sie tägliche Nutzungsberichte

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
3. Wählen Sie auf der Seite Text messaging (SMS) (Textnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.



4. Geben Sie auf der Seite Edit text messaging preferences (Textnachrichteneinstellungen bearbeiten) im Abschnitt Details Amazon S3 bucket name for usage reports (Amazon S3-Bucket-Name für Nutzungsberichte) an.



5. Wählen Sie Änderungen speichern aus.

Beispiel einer Bucket-Richtlinie

Die folgende Richtlinie berechtigt den Amazon SNS-Service, die Aktionen `s3:PutObject`, `s3:GetBucketLocation` und `s3:ListBucket` durchzuführen.

AWS bietet Tools für alle Dienste mit Service Principals, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde. Wenn der Principal in einer Amazon S3 S3-Bucket-Richtlinienerklärung ein verwirrtes stellvertretendes Problem ist. Um zu begrenzen, welche Region und welches Konto der Bucket tägliche Nutzungsberichte erhalten kann, verwenden Sie `aws:SourceArn`, wie im folgenden Beispiel gezeigt. Wenn Sie die Regionen, die diese Berichte genieren können, nicht einschränken wollen, können Sie mit `aws:SourceAccount` einschränken, welche Konten die Berichte generieren. Wenn Sie den ARN der Ressource nicht kennen, verwenden Sie `aws:SourceAccount`.

Verwenden Sie das folgende Beispiel, das einen Schutz vor dem Problem des verwirrten Stellvertreters enthält, um eine Amazon-S3-Bucket für den Empfang von täglichen SMS-Nutzungsberichten von Amazon SNS zu erstellen.

JSON

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:us-east-1:account_id:*"
                }
            }
        },
        {
            "Sid": "AllowGetBucketLocation",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:GetBucketLocation",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:us-east-1:account_id:*"
                }
            }
        },
        {
            "Sid": "AllowListBucket",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:us-east-1:account_id:*"
                }
            }
        }
    ]
}
```

> **Note**
>
> Sie können Nutzungsberichte in Amazon-S3-Buckets veröffentlichen, die sich im Besitz von AWS-Konto befinden, die in der Amazon S3-Richtlinie im `Condition`-Element festgelegt sind. Informationen zum Veröffentlichen von Nutzungsberichten in einem Amazon S3 S3-Bucket, der einem anderen AWS-Konto gehört, finden Sie unter Wie kann ich Amazon S3 S3-Objekte von einem anderen kopieren AWS-Konto? .

Beispiel eines täglichen Nutzungsberichts

Nachdem Sie die täglichen Nutzungsberichte abonniert haben, speichert Amazon SNS jeden Tag eine CSV-Datei mit Nutzungsdaten am folgenden Speicherort:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Jede Datei kann bis zu 50.000 Datensätze enthalten. Wenn die Berichte für einen Tag dieses Kontingent übersteigen, fügt Amazon SNS mehrere Dateien hinzu. Nachstehend finden Sie einen Beispielbericht:

```
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
```

Unterstützung für Amazon SNS SMS-Messaging anfordern

> **Important**
>
> Der Amazon SNS SMS Developer Guide wurde aktualisiert. Amazon SNS hat sich AWS Endbenutzer-Messaging SMSfür die Zustellung von SMS-Nachrichten integriert. Dieses Handbuch enthält die neuesten Informationen zur Erstellung, Konfiguration und Verwaltung Ihrer Amazon SNS SNS-SMS-Nachrichten.

Bestimmte SMS-Optionen mit Amazon SNS sind für Ihr AWS Konto erst verfügbar, wenn Sie Kontakt aufnehmen Support. Erstellen Sie einen Fall im AWS -Support Center, um eine der folgenden Optionen anzufordern:

- Eine Erhöhung Ihres monatlichen Ausgabenschwellenwerts für SMS

  Standardmäßig liegt der monatliche Ausgabenschwellenwert bei 1,00 USD. Ihr Ausgabenschwellenwert bestimmt die Anzahl der Nachrichten, die Sie mit Amazon SNS senden können. Sie können einen Ausgabenschwellenwert anfordern, der der erwarteten monatlichen Menge an Nachrichten für Ihren SMS-Anwendungsfall entspricht.
- Eine Verschiebung aus der SMS-Sandbox, damit Sie SMS-Nachrichten ohne Einschränkungen versenden können. Weitere Informationen finden Sie unter Verlassen der Amazon SNS SMS-Sandbox.
- Eine engagierte Ursprungsnummer
- Eine spezielle Absender-ID. Eine Sender-ID ist eine benutzerdefinierte ID, die auf dem Gerät des Empfängers als Sender angezeigt wird. Sie können z. B. Ihre Unternehmensmarke verwenden, um die Nachrichtenquelle leichter erkennbar zu machen. Die Support für den Absender IDs ist je nach Land oder Region unterschiedlich. Weitere Informationen finden Sie AWS Endbenutzer-Messaging SMS im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Unterstützte Länder und Regionen für SMS-Nachrichten mit.

Erhöhung Ihres monatlichen Amazon SNS SMS-Ausgabenkontingents beantragen

Amazon SNS stellt Ausgabenkontingente zur Verfügung, mit denen Sie die maximalen Kosten pro Monat verwalten können, die durch das Senden von SMS über Ihr Konto entstehen. Das Ausgabenkontingent begrenzt Ihr Risiko im Falle eines böswilligen Angriffs und verhindert, dass Ihre Upstream-Anwendung mehr Nachrichten sendet als erwartet. Sie können Amazon SNS so konfigurieren, dass die Veröffentlichung von SMS-Nachrichten eingestellt wird, wenn festgestellt wird, dass beim Senden einer SMS-Nachricht Kosten anfallen, die Ihr Ausgabenkontingent für den aktuellen Monat übersteigen.

Um sicherzustellen, dass Ihr Betrieb nicht beeinträchtigt wird, empfehlen wir, eine Ausgabenquote anzufordern, die hoch genug ist, um Ihre Produktions-Workloads zu unterstützen. Weitere Informationen finden Sie unter Schritt 1: Öffnen eines Amazon-SNS-SMS-Falls. Sobald Sie das Kontingent erhalten haben, können Sie Ihr Risiko verwalten, indem Sie das volle Kontingent oder einen kleineren Wert anwenden, wie unter Schritt 2: Aktualisieren Ihrer SMS-Einstellungen

beschrieben. Wenn Sie einen kleineren Wert anwenden, können Sie Ihre monatlichen Ausgaben mit der Option steuern, bei Bedarf hochzuskalieren.

> **Important**
>
> Da Amazon SNS ein verteiltes System ist, beendet es das Senden von SMS-Nachrichten innerhalb von einigen Minuten, falls das Ausgabenkontingent überschritten wurde. Wenn Sie in diesem Zeitraum weiter SMS-Nachrichten senden, können Kosten entstehen, die Ihr Kontingent überschreiten.

Wir setzen das Ausgabenkontingent für alle neuen Konten auf 1,00 USD pro Monat fest. Dieses Kontingent soll Ihnen erlauben, die Möglichkeiten des Nachrichtenversands von Amazon SNS zu testen. Um eine Erhöhung des SMS-Ausgabenkontingents für Ihr Konto zu beantragen, öffnen Sie im AWS Support Center einen Fall zur Erhöhung des Kontingents.

Themen

- Schritt 1: Öffnen eines Amazon SNS SMS-Falls
- Schritt 2: Aktualisieren Ihrer SMS-Einstellungen in der Amazon SNS-Konsole

Schritt 1: Öffnen eines Amazon SNS SMS-Falls

Sie können eine Erhöhung Ihres monatlichen Ausgabenkontingents beantragen, indem Sie im AWS Support Center einen Fall zur Erhöhung des Kontingents eröffnen.

> **Note**
>
> Einige der Felder im Antragsformular sind als „optional“ markiert. Allerdings benötigt Support alle Informationen, die in den folgenden Schritten aufgeführt sind, um Ihre Anfrage zu bearbeiten. Wenn Sie nicht alle erforderlichen Informationen bereitstellen, kann es bei der Bearbeitung Ihrer Anfrage zu Verzögerungen kommen.

1. Melden Sie sich unter https://console.aws.amazon.com/ AWS Management Consoleean.
2. Klicken Sie im Menü Support (Support) auf Support Center (Support-Center).
3. Wählen Sie auf der Registerkarte Offene Support-Fälle die Option Fall erstellen aus.

4. Wählen Sie den Link Erhöhung des Servicelimits? und gehen Sie dann wie folgt vor:
   - Wählen Sie als Limittyp SNS Textnachrichten.
   - (Optional) Geben Sie unter Link zur Website oder App angeben, die die SMS-Nachrichten senden wird Informationen über die Website, die Anwendung oder den Service an, die bzw. der SMS-Nachrichten senden wird.
   - (Optional) Wählen Sie für Art der Nachrichten, die gesendet werden sollen die Art der Nachrichten aus, die Sie mit Ihren Langwahlnummern senden möchten:
     - One-time Password (Einmaliges Passwort) – Nachrichten, die für Ihre Kunden Passwörter zur Authentifizierung bei Ihrer Website oder Anwendung bereitstellen.
     - Promotional (Werbung) – Nicht kritische Nachrichten, die Ihr Unternehmen oder Ihren Service bewerben, wie beispielsweise Sonderangebote oder Ankündigungen.
     - Transactional (Transaktionsnachrichten) – Wichtige Informationsmeldung, die Kundentransaktionen unterstützen, wie beispielsweise Bestellbestätigungen oder Kontowarnungen. Transaktionsnachrichten dürfen keine Werbeaktionen oder Marketinginhalte enthalten.
   - (Optional) Wählen Sie für welche AWS Region Sie Nachrichten senden möchten, die Region aus, aus der Sie Nachrichten senden möchten.
   - (Optional) Geben Sie für In welche Länder möchten Sie Nachrichten senden das Land oder die Region ein, in dem bzw. der Sie Kurzwahlnummern erwerben möchten.
   - (Optional) Geben Sie unter Wie entscheiden sich Ihre Kunden dafür, Nachrichten von Ihnen zu erhalten Einzelheiten zu Ihrem Anmeldeverfahren an.
   - (Optional) Geben Sie im Feld Bitte geben Sie die Nachrichtenvorlage an, die Sie verwenden möchten, um Nachrichten an Ihre Kunden zu senden die Vorlage ein, die Sie verwenden werden.
5. Führen Sie unter Requests (Anforderungen) die folgenden Abschnitte aus:
   - Wählen Sie für die Region die Region aus, aus der Sie Nachrichten senden möchten.

> **Note**
>
> Die Region ist im Abschnitt Anforderungen erforderlich. Auch wenn Sie diese Informationen im Abschnitt Falldetails angegeben haben, müssen Sie sie auch hier angeben.

- Wählen Sie für Resource Type (Ressourcentyp) die Option General Limits (Allgemeine Limits) aus.
- Wählen Sie für Limit Account Spend Threshold Increase (Erhöhung des Ausgabenschwellenwerts) aus.

6. Geben Sie für den Neuen Grenzwert den Höchstbetrag in USD ein, den Sie pro Kalendermonat für SMS-Nachrichten ausgeben möchten.
7. Machen Sie unter Case description (Fallbeschreibung) für Use case description (Anwendungsfall-Beschreibung) die folgenden Angaben:

   - Die Website oder App des Unternehmens oder Services, das/der SMS-Nachrichten sendet.
   - Den Service, der von Ihrer Website oder App bereitgestellt wird, und wie Ihre SMS-Nachrichten zu diesem Service beitragen.
   - Wie sich Benutzer bei Ihrer Website, App oder an einem anderen Ort anmelden, um Ihre SMS-Nachrichten freiwillig zu erhalten.

   Wenn Ihr angefordertes Ausgabenkontingent (der Wert, den Sie unter New quota value (Neuer Kontingentwert) angegeben haben), 10.000 USD überschreitet, geben Sie für jedes Zielland Ihrer Nachrichten die folgenden zusätzlichen Einzelheiten an:

   - Ob Sie eine Sender-ID oder eine Kurzwahlnummer verwenden. Wenn Sie eine Sender-ID verwenden, geben Sie an:
     - Die Sender-ID.
     - Ob die Sender-ID bei Mobilnetzanbietern in dem betreffenden Land registriert ist.
   - Das erwartete Maximum transactions-per-second (TPS) für Ihre Nachrichten.
   - Die durchschnittliche Nachrichtengröße.
   - Die Vorlage für die in das betreffende Land gesendeten Nachrichten.
   - (Optional) Eventuelle Zeichenverschlüsselungsanforderungen.
8. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Wenn Sie mehrere Anfragen durchführen, geben Sie für jede davon die erforderlichen Informationen an. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Unterstützung für Amazon SNS SMS-Messaging anfordern.
9. Wählen Sie unter Kontaktoptionen für Bevorzugte Kontaktsprache die Sprache aus, in der Sie Mitteilungen zu diesem Fall erhalten möchten.

10. Wenn Sie fertig sind, klicken Sie auf Submit (Absenden).

Das Support Team gibt innerhalb von 24 Stunden eine erste Antwort auf Ihre Anfrage.

Da wir verhindern möchten, dass unerwünschte oder schädliche Inhalte in unseren Systemen eingehen, müssen wir jede Anfrage sorgfältig prüfen. Nach einer Prüfung kommen wir Ihrer Anfrage innerhalb dieses 24-Stunden-Zeitraums nach. Für den Fall, dass wir weitere Informationen von Ihnen benötigen, kann die Bearbeitung Ihrer Anfrage länger dauern.

Wenn Ihr Anwendungsfall gegen unsere Richtlinien verstößt, können wir Ihrer Anfrage möglicherweise nicht nachkommen.

Schritt 2: Aktualisieren Ihrer SMS-Einstellungen in der Amazon SNS-Konsole

Nachdem wir Sie darüber informiert haben, dass Ihr monatliches Ausgabenkontingent erhöht wurde, müssen Sie das Ausgabenkontingent für Ihr Konto in der Amazon SNS-Konsole anpassen.

> **Important**
>
> Sie müssen die folgenden Schritte ausführen, sonst wird Ihr SMS-Ausgabenlimit nicht erhöht.

So passen Sie Ihr Ausgabenkontingent in der Konsole an

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Öffnen Sie das linke Navigationsmenü, erweitern Sie Mobil und wählen Sie dann SMS.
3. Wählen Sie auf der Seite Mobile text messaging (SMS) (Mobile Texttnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.
4. Geben Sie auf der Seite SMS-Voreinstellungen bearbeiten unter Details Ihr neues SMS-Ausgabenlimit unter Kontoausgabenlimit ein.

   > **Note**
   >
   > Möglicherweise erhalten Sie eine Warnung, dass der eingegebene Wert größer als das Standardausgabenlimit ist. Sie können dies ignorieren.

5. Wählen Sie Änderungen speichern.

> **Note**
>
> Wenn der Fehler „Invalid Parameter (Ungültiger Parameter)" angezeigt wird, überprüfen Sie den Kontakt beim AWS Support, und prüfen Sie, ob Sie das richtige neue SMS-Ausgabenlimit eingegeben haben. Wenn Sie immer noch ein Problem haben, öffnen Sie einen Fall im AWS Support Center.

Wenn Sie Ihren Fall im Support Center erstellen, achten Sie darauf, dass Sie alle erforderlichen Informationen für die Art der Anfrage angeben, die Sie einreichen. Andernfalls Support müssen Sie sich mit Ihnen in Verbindung setzen, um diese Informationen zu erhalten, bevor Sie fortfahren. Indem Sie einen detaillierten Fall übermitteln, tragen Sie dazu bei, dass Ihr Fall ohne Verzögerungen gelöst werden kann. Informationen zu den Details, die für bestimmte Arten von SMS-Anfragen benötigt werden, finden Sie in den folgenden Themen.

Weitere Informationen zum Absender IDs finden Sie in der folgenden Dokumentation im AWS Endbenutzer-Messaging SMS Benutzerhandbuch:

AWS Endbenutzer-Messaging SMS Thema	Beschreibung
Eine Erhöhung der Ausgabenquote beantragen	Ihr Ausgabenkontingent bestimmt, wie viel Geld Sie AWS Endbenutzer-Messaging SMS jeden Monat für den Versand von SMS-Nachrichten ausgeben können.
Eröffnen Sie im Support Center einen Fall für eine Absender-ID	Wenn Sie beabsichtigen, Nachrichten an Empfänger in einem Land zu senden, in dem Absender erforderlich IDs sind, können Sie eine Absender-ID anfordern, indem Sie im Support Center einen neuen Fall erstellen.

Bewährte Methoden für Amazon SNS SMS-Messaging

> **Important**
>
> Der Amazon SNS SMS Developer Guide wurde aktualisiert. Amazon SNS hat sich AWS Endbenutzer-Messaging SMSfür die Zustellung von SMS-Nachrichten integriert. Dieses Handbuch enthält die neuesten Informationen zur Erstellung, Konfiguration und Verwaltung Ihrer Amazon SNS SNS-SMS-Nachrichten.

Mobiltelefonbenutzer neigen zu einer sehr geringen Toleranz für unerwünschte SMS-Nachrichten. Die Antwortraten für unaufgeforderte SMS-Kampagnen sind fast immer sehr niedrig, was gleichzeitig auch eine schlechte Investitionsrendite bedeutet.

Außerdem überwachen Mobilfunkanbieter Massen-SMS-Sender kontinuierlich. Nummern, die offenbar unerwünschte Nachrichten senden, werden gedrosselt oder gesperrt.

Das Senden von unerwünschten Inhalten stellt auch einen Verstoß gegen die AWS Acceptable Use Policy dar. Das Amazon SNS-Team überprüft SMS-Kampagnen regelmäßig und kann Ihre Möglichkeiten zum Senden von Nachrichten drosseln oder sperren, wenn Sie offensichtlich unerwünschte Nachrichten senden.

Schließlich gelten in vielen Ländern, Regionen und Rechtsbereichen empfindliche Strafen für das Senden unerwünschter SMS-Nachrichten. So bestimmt beispielsweise in den USA der Telephone Consumer Protection Act (TCPA), dass Verbraucher einen Anspruch auf eine (vom Sender zu leistende) Zahlung von 500 bis 1.500 USD für jede unerwünschte Nachricht haben, die sie erhalten.

Dieser Abschnitt beschreibt verschiedene bewährte Methoden, die Ihnen dabei helfen können, die Einbeziehung Ihrer Kunden zu verbessern und hohe Strafzahlungen zu vermeiden. Beachten Sie jedoch, dass diesem Abschnitt keine Rechtsberatung darstellt. Wenden Sie sich immer an einen Rechtsanwalt, um juristischen Rat einzuholen.

Einhaltung von Gesetzen, Vorschriften und Anforderungen der Anbieter

Für Verstöße gegen die Gesetze und Bestimmungen der Länder, in denen sich Ihre Kunden befinden, drohen erhebliche Geldbußen und Strafen. Daher müssen Sie die Gesetze kennen, die SMS-Messaging in den Ländern und Regionen regeln, in denen Sie geschäftlich tätig sind.

Die folgende Liste enthält Links auf die wichtigsten Gesetze zu SMS-Kommunikation in den größten Märkten der Welt.

- USA: Der Telephone Consumer Protection Act (TCPA) von 1991 gilt für bestimmte Arten von SMS-Nachrichten. Weitere Informationen finden Sie unter Rules and Regulations (Regeln und Vorschriften) auf der Website der Federal Communication Commission.
- Großbritannien und Nordirland: Die Privacy and Electronic Communications (EG-Richtlinie) (PECR) von 2003 gilt für bestimmte Arten von SMS-Nachrichten. Weitere Informationen finden Sie unter What are PECR? (Was sind PECR?) auf der Website des UK Information Commissioner's Office.
- Europäische Union: Die Datenschutzrichtlinie für elektronische Kommunikation von 2002 (auch ePrivacy-Richtlinie) gilt für einige Typen von SMS-Nachrichten. Weitere Informationen finden Sie im vollständigen Gesetzestext auf der Website Europa.eu.
- Kanada: Der Fighting Internet and Wireless Spam Act, auch als kanadisches Antispamgesetz (CASL) bekannt, gilt für bestimmte Arten von SMS-Nachrichten. Weitere Informationen finden Sie im vollständigen Gesetzestext auf der Website des kanadischen Parlaments.
- Japan: Das Gesetz zur Regulierung der Übermittlung von bestimmten elektronischen Mails kann für bestimmte Arten von SMS-Nachrichten gelten. Weitere Informationen finden Sie unter Japan's Countermeasures Against Spam auf der Website des japanischen Ministeriums für innere Angelegenheiten und Kommunikation.

Diese Gesetze können für Sie als Absender auch dann gelten, wenn Ihr Unternehmen oder Ihre Organisation nicht in einem dieser Länder ansässig ist. Einige der Gesetze in dieser Liste wurden ursprünglich für unerwünschte E-Mails oder Anrufe erlassen, wurden seither jedoch auch auf SMS-Nachrichten erweitert. In anderen Ländern und Regionen können andere Gesetze für die Übermittlung von SMS-Nachrichten gelten. Fragen Sie einen Anwalt in einem Land oder einer Region, in der sich Ihre Kunden befinden, um sich entsprechend beraten zu lassen.

In vielen Ländern haben die lokalen Netzbetreiber letztendlich die Befugnis, zu bestimmen, welche Art von Datenverkehr ihre Netze passiert. Dies bedeutet, dass die Netzbetreiber möglicherweise Beschränkungen für SMS-Inhalte auferlegen, die über die Mindestanforderungen der örtlichen Gesetzgebung hinausgehen.

Einholen von Berechtigungen

Senden Sie niemals Nachrichten an Empfänger, die nicht ausdrücklich darum gebeten haben, die Arten von Nachrichten zu erhalten, die Sie senden möchten. Teilen Sie keine Opt-in-Listen, auch nicht in Organisationen innerhalb desselben Unternehmens.

Wenn Empfänger über ein Onlineformular angeben können, dass sie Ihre Nachrichten erhalten möchten, fügen Sie dem Formular Systeme hinzu, die verhindern, dass automatisierte Scripts ohne

Wissen der Benutzer Abonnements für sie abschließen. Sie sollten auch die Häufigkeit begrenzen, mit der ein Benutzer in einer einzelnen Sitzung eine Telefonnummer angeben kann.

Wenn Sie eine SMS-Opt-in-Anfrage erhalten, senden Sie dem Empfänger eine Nachricht mit der Bitte, zu bestätigen, dass er Nachrichten von Ihnen erhalten möchte. Senden Sie diesem Empfänger keine weiteren Nachrichten, bis er sein Abonnement bestätigt hat. Eine Bestätigungsnachricht für ein Abonnement kann wie folgt aussehen:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

Führen Sie Unterlagen mit Datum, Uhrzeit und Quelle der einzelnen Opt-in-Anfragen und Bestätigungen. Dies kann hilfreich sein, wenn ein Anbieter oder eine Regulierungsbehörde danach fragt, oder auch für Routineprüfungen Ihrer Kundenliste.

Opt-in-Workflow

In einigen Fällen (wie bei der gebührenfreien Registrierung in den USA oder der Kurzwahl-Registrierung) verlangen Mobilfunkanbieter, dass Sie Mockups oder Screenshots Ihres gesamten Opt-in-Workflows zur Verfügung stellen. Die Mockups oder Screenshots müssen dem Opt-in-Workflow, den Ihre Empfänger ausführen werden, sehr ähnlich sein.

Ihre Mockups oder Screenshots sollten alle unten aufgeführten erforderlichen Angaben enthalten, um ein Höchstmaß an Compliance zu gewährleisten.

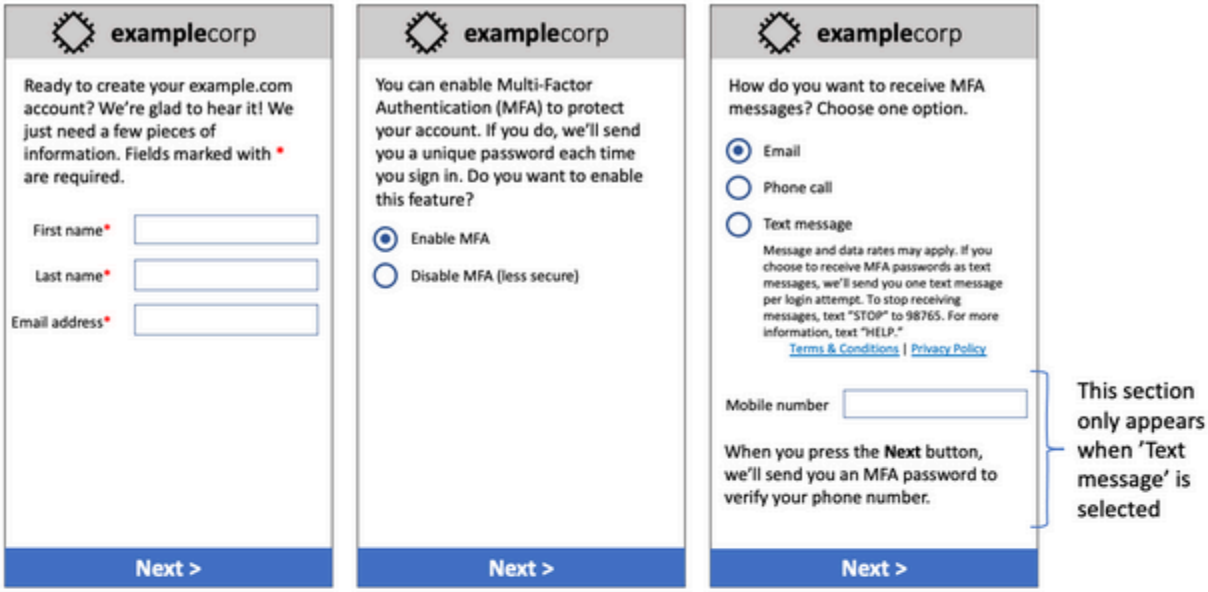
Erforderliche Angaben

- Eine Beschreibung des Messaging-Anwendungsfalls, den Sie über Ihr Programm senden.
- Der Satz „Es können Gebühren für Nachrichten und Daten anfallen“.
- Ein Hinweis darauf, wie oft Empfänger Nachrichten von Ihnen erhalten. Beispielsweise könnte die Angabe für ein wiederkehrendes Messaging-Programm „eine Nachricht pro Woche“ lauten. Ein Anwendungsfall für ein Einmalpasswort oder eine Multi-Faktor-Authentifizierung könnte lauten: „Nachrichtenfrequenz variiert“ oder „eine Nachricht pro Anmeldeversuch“.
- Links zu Ihren Allgemeinen Geschäftsbedingungen und Datenschutzrichtlinien.

Häufige Ablehnungsgründe für nicht konforme Opt-ins

- Wenn der angegebene Firmenname nicht mit dem übereinstimmt, was im Mockup oder Screenshot angegeben ist. Alle nicht offensichtlichen Zusammenhänge sollten in der Beschreibung des Opt-in-Workflows erklärt werden.
- Wenn es den Anschein hat, dass eine Nachricht an den Empfänger gesendet wird, zuvor jedoch keine ausdrückliche Zustimmung eingeholt wurde. Die ausdrückliche Zustimmung ist eine Voraussetzung für alle Nachrichten.
- Wenn es den Anschein hat, dass der Empfang einer Textnachricht erforderlich ist, um sich für einen Service anzumelden. Dies ist nicht konform, wenn der Workflow keine Alternative zum Empfang einer Opt-in-Nachricht in anderer Form wie E-Mail oder Sprachanruf bietet.
- Wenn die Opt-in-Formulierungen vollständig in den Nutzungsbedingungen angegeben ist. Die Angaben sollten dem Empfänger immer zum Opt-in-Zeitpunkt vorgelegt werden und nicht in einem verknüpften Richtliniendokument enthalten sein.
- Wenn ein Kunde zugestimmt hat, eine bestimmte Art von Nachricht von Ihnen zu erhalten, und Sie ihm andere Arten von Textnachrichten senden. Der Kunde stimmt beispielsweise zu, Einmalpasswörter zu erhalten, erhält aber auch Umfragenachrichten.
- Wenn die erforderlichen Angaben (siehe oben) den Empfängern nicht vorgelegt werden.

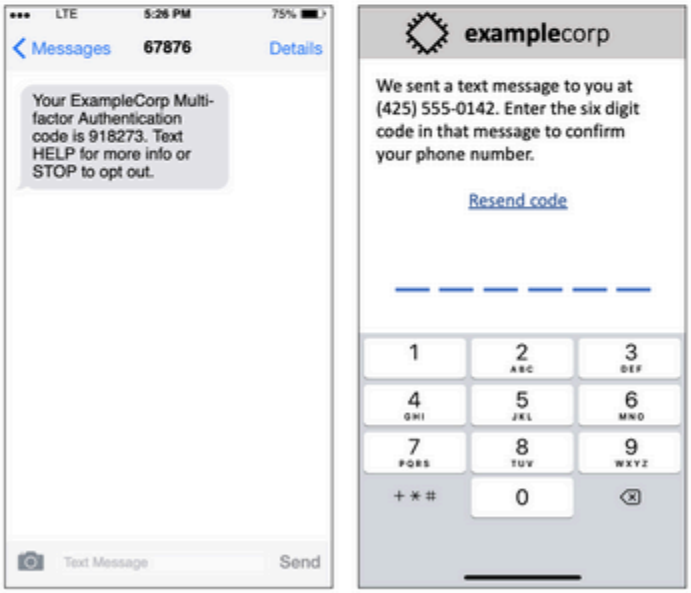
Das folgende Beispiel erfüllt die Anforderungen der Mobilfunkanbieter für einen Anwendungsfall mit Multi-Faktor-Authentifizierung.



1. User provides basic account information.

2. User decides whether to enable MFA.

3. If MFA enabled, user chooses how to receive MFA token.



4. If user chooses to receive MFA token by text, send a token.

5. User enters MFA token to verify phone number.

Modell eines Anwendungsfalls für die Multi-Faktor-Authentifizierung

Es enthält finalisierten Text und Bilder und zeigt den gesamten Opt-in-Workflow mit Anmerkungen. Im Opt-in-Workflow muss der Kunde eindeutige, absichtliche Maßnahmen ergreifen, um seine

Zustimmung zum Erhalt von Textnachrichten zu erteilen. Der Workflow enthält außerdem alle erforderlichen Angaben.

Andere Opt-in-Workflowtypen

Mobilfunkanbieter akzeptieren auch Opt-in-Workflows außerhalb von Anwendungen und Websites wie mündliches oder schriftliches Opt-in, wenn dies den oben genannten Anforderungen entspricht. Ein konformer Opt-in-Workflow und ein mündliches oder schriftliches Skript holen die ausdrückliche Zustimmung des Empfängers ein, einen bestimmten Nachrichtentyp zu erhalten. Beispiele hierfür sind das mündliche Skript, das ein Kundendienstmitarbeiter verwendet, um die Zustimmung einzuholen, bevor er die Daten in eine Servicedatenbank aufnimmt, oder eine Telefonnummer, die auf einem Werbeflyer aufgeführt ist. Um ein Modell dieser Arten von Opt-in-Workflows zu erstellen, können Sie einen Screenshot Ihres Anmeldeskripts, Marketingmaterials oder Ihrer Datenbank, in der Zahlen gesammelt werden, bereitstellen. Mobilfunkanbieter haben möglicherweise zusätzliche Fragen zu diesen Anwendungsfällen, wenn ein Opt-in nicht klar ist oder der Anwendungsfall bestimmte Volumina überschreitet.

Nicht an alte Listen senden

Empfänger ändern ihre Telefonnummer häufig. Eine Telefonnummer, für die Sie vor zwei Jahren eine Zustimmung zur Kontaktaufnahme eingeholt haben, könnte heute jemand anderem gehören. Verwenden Sie keine alte Liste von Telefonnummern für ein neues Messaging-Programm. Andernfalls werden wahrscheinlich einige Nachrichten ihren Empfänger nicht erreichen, weil die Nummer nicht mehr vergeben ist, und einige Personen werden sich abmelden, weil sie sich nicht erinnern können, Ihnen ihre Zustimmung gegeben zu haben.

Prüfung Ihrer Kundenlisten

Wenn Sie wiederkehrende SMS-Kampagnen versenden, sollten Sie Ihre Kundenlisten regelmäßig überprüfen. Dadurch stellen Sie sicher, dass nur Kunden, die an Ihren Nachrichten interessiert sind, diese auch erhalten.

Senden Sie bei der Prüfung Ihrer Liste jedem angemeldeten Kunden eine Nachricht, die diesen an sein Abonnement erinnert, begleitet von Anleitungen zum eventuellen Abbestellen der Nachrichten. Eine solche Erinnerungsnachricht kann wie folgt aussehen:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

Aufbewahren von Aufzeichnungen

Führen Sie Aufzeichnungen, aus denen hervorgeht, wann ein Kunde den Erhalt von SMS-Nachrichten von Ihnen angefordert hat und welche Nachrichten Sie an welche Kunden senden. Viele Ländern und Regionen auf der Welt fordern Sender von SMS-Nachrichten auf, dieses so aufzubewahren, dass Sie problemlos abgerufen werden können. Auch Mobilfunkanbieter können diese Informationen jederzeit von Ihnen anfordern. Die genauen Informationen, die Sie bereitstellen müssen, variieren je nach Land oder Region. Weitere Informationen über Aufbewahrungsanforderungen finden Sie in den Bestimmungen zu kommerziellem SMS Messaging in den Ländern oder Regionen, in denen Ihre Kunden ansässig sind.

Es kann vorkommen, dass ein Anbieter oder eine Regulierungsbehörde von uns verlangt, dass wir nachweisen, dass Kunden sich dafür entschieden haben, Nachrichten von Ihnen zu empfangen. Support Kontaktiert Sie in diesen Situationen mit einer Liste der Informationen, die der Transporteur oder die Agentur benötigt. Wenn Sie die notwendigen Informationen nicht bereitstellen können, schränken wir Ihre Möglichkeit, weitere SMS-Nachrichten zu senden, möglicherweise ein.

Formulieren Sie Ihre Nachrichten klar, ehrlich und präzise

SMS ist ein einzigartiges Medium. Das character-per-message 160-Limit bedeutet, dass Ihre Nachrichten präzise sein müssen. Techniken, die Sie möglicherweise in anderen Kommunikationskanälen wie E-Mail verwenden, gelten möglicherweise nicht für den SMS-Kanal und wirken möglicherweise sogar unehrlich oder irreführend, wenn sie mit SMS-Nachrichten verwendet werden. Wenn der Inhalt Ihrer Nachrichten nicht mit bewährten Methoden übereinstimmt, ignorieren die Empfänger Ihre Nachrichten möglicherweise. Im schlimmsten Fall identifizieren die Mobilfunkanbieter Ihre Nachrichten möglicherweise als Spam und blockieren künftige Nachrichten von Ihrer Telefonnummer.

Dieser Abschnitt enthält einige Tipps und Anregungen zum Erstellen eines effektiven SMS-Nachrichtentexts.

Identifizieren Sie sich als Absender

Ihre Empfänger sollten sofort erkennen können, dass eine Nachricht von Ihnen stammt. Absender, die diese bewährte Methode befolgen, geben am Anfang jeder Nachricht einen identifizierenden Namen („Programmname“) an.

Vermeiden Sie Folgendes:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

Versuchen Sie nicht, Ihre Nachricht wie eine Nachricht aussehen zu person-to-person lassen

Einige Vermarkter sind versucht, ihren SMS-Nachrichten eine persönliche Note zu verleihen, indem sie den Eindruck erwecken, dass ihre Nachrichten von einer Person stammen. Diese Strategie kann jedoch dazu führen, dass Ihre Nachricht wie ein Phishing-Versuch erscheint.

Vermeiden Sie Folgendes:

```
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.
```

Versuchen Sie stattdessen dieses Linkformat:

```
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.
```

Seien Sie vorsichtig, wenn Sie über Geld sprechen.

Betrüger machen sich oft den Wunsch der Menschen zunutze, Geld zu sparen und zu erhalten. Lassen Sie Angebote nicht so erscheinen, als seien sie zu gut, um wahr zu sein. Nutzen Sie die Verlockung des Geldes nicht, um Menschen zu täuschen. Verwenden Sie keine Währungssymbole, um Geldbeträge anzugeben.

Vermeiden Sie Folgendes:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Versuchen Sie stattdessen dieses Linkformat:

```
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.
```

Verwenden Sie nur die notwendigen Zeichen

Unternehmen neigen oft dazu, ihre eingetragenen Marken zu schützen, indem sie Markensymbole wie ™ oder ® in ihre Nachrichten aufnehmen. Diese Symbole sind jedoch nicht Teil des Standardzeichensatzes (bekannt als GSM-Alphabet), der in einer SMS-Nachricht mit 160 Zeichen enthalten sein kann. Wenn Sie eine Nachricht senden, die eines dieser Zeichen enthält, wird Ihre Nachricht automatisch mit einem anderen Zeichencodierungssystem gesendet, das nur 70 Zeichen pro Nachricht unterstützt. Infolgedessen könnte Ihre Nachricht in mehrere Teile zerlegt werden. Da Ihnen jeder gesendete Nachrichtenteil in Rechnung gestellt wird, kann die Nachricht Sie mehr kosten als erwartet. Darüber hinaus erhalten Ihre Empfänger statt einer einzigen Nachricht möglicherweise mehrere aufeinanderfolgende Nachrichten von Ihnen. Weitere Informationen zur SMS-Zeichencodierung erhalten Sie unter SMS-Zeichenbeschränkungen in Amazon SNS.

Vermeiden Sie Folgendes:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> Die beiden vorherigen Beispiele sind fast identisch. Das erste Beispiel enthält jedoch ein Symbol für eine eingetragene Marke (®), das nicht Bestandteil des GSM-Alphabets ist. Dadurch wird das erste Beispiel in zwei Nachrichtenteilen gesendet, während das zweite Beispiel als ein Nachrichtenteil seinen Empfänger erreicht.

Verwenden gültiger, sicherer Links

Wenn Ihre Nachricht Links enthält, überprüfen Sie die Links, um sicherzustellen, dass sie funktionieren. Testen Sie Ihre Links auf einem Gerät außerhalb Ihres Unternehmensnetzwerks, um sicherzustellen, dass sie ordnungsgemäß aufgelöst werden. Aufgrund der Beschränkung von 160 Zeichen für SMS-Nachrichten URLs kann sehr lange Nachrichten auf mehrere Nachrichten aufgeteilt werden. Sie sollten Umleitungsdomänen verwenden, um verkürzte Inhalte bereitzustellen. URLs Sie

sollten jedoch keine kostenlosen Linkverkürzungsdienste, wie tinyurl.com oder bitly.com, verwenden, da Netzbetreiber dazu neigen, Nachrichten zu filtern, die Links zu diesen Domänen enthalten. Sie können jedoch kostenpflichtige Linkverkürzungsdienste nutzen, solange Ihre Links auf eine Domäne verweisen, die ausschließlich der Nutzung Ihres Unternehmens oder Ihrer Organisation vorbehalten ist.

Vermeiden Sie Folgendes:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Versuchen Sie stattdessen dieses Linkformat:

```
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.
```

Verwenden Sie nur eine begrenzte Anzahl von Abkürzungen

Das Limit von 160 Zeichen für den SMS-Kanal verleitet einige Absender dazu, in ihren Nachrichten verstärkt Abkürzungen zu verwenden. Die übermäßige Verwendung von Abkürzungen kann jedoch vielen Lesern unprofessionell erscheinen und dazu führen, dass einige Benutzer Ihre Nachricht als Spam melden. Es ist durchaus möglich, eine zusammenhängende Nachricht zu schreiben, ohne übermäßig viele Abkürzungen zu verwenden.

Vermeiden Sie Folgendes:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Angemessene Antworten

Wenn ein Empfänger auf Ihre Nachrichten antwortet, stellen Sie sicher, dass Sie mit nützlichen Informationen reagieren. Zum Beispiel: Wenn ein Kunde auf eine Ihrer Nachrichten mit dem Schlüsselwort „HILFE“ antwortet, senden Sie diesem Informationen zu dem von ihm abonnierten Programm, zu der Anzahl der pro Monat gesendeten Nachrichten und Möglichkeiten zur

Kontaktaufnahme mit Ihnen für weitere Informationen zu. Eine „HILFE“-Antwort kann wie folgt aussehen:

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Wenn ein Kunde mit dem Schlüsselwort „STOPP“ antwortet, teilen Sie ihm mit, dass er fortan keine weiteren Nachrichten mehr erhalten wird. Eine „STOPP“-Antwort kann wie folgt aussehen:

```
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.
```

Anpassen Ihres Sendens basierend auf der Kundenbeteiligung

Die Prioritäten Ihrer Kunden können sich mit der Zeit ändern. Wenn Kunden Ihre Nachrichten nicht mehr nützlich finden, bestellen sie sie möglicherweise ganz ab oder melden sie sogar als unerwünscht. Daher ist es wichtig, dass Sie Ihr Sendeverhalten auf der Grundlage der Kundenbeteiligung anpassen.

Für Kunden, die nur selten mit Ihren Nachrichten interagieren, sollten Sie die Häufigkeit Ihrer Nachrichten entsprechend anpassen. Wenn Sie z. B. wöchentliche Nachrichten an interessierte Kunden senden, können Sie für weniger interessierte Kunden einen monatlichen Kurzbericht erstellen.

Entfernen Sie schließlich Kunden, die niemals mit Ihren Nachrichten interagieren, vollständig aus Ihren Kundenlisten. Dieser Schritt wird verhindert, dass die Kunden irgend wann verärgert auf Ihre Nachrichten reagieren. Außerdem sparen Sie dadurch Geld und schützen Ihren guten Ruf als Sender.

Senden zu angemessenen Zeiten

Senden Sie Nachrichten nur während üblicher Arbeitszeiten am Tage. Wenn Sie Nachrichten zur Mittagszeit oder mitten in der Nacht senden, ist es sehr wahrscheinlich, dass Kunden Ihre Nachrichten abbestellen, um davon nicht mehr gestört zu werden. Darüber hinaus ist es nicht sinnvoll, SMS-Nachrichten dann zu senden, wenn Ihre Kunden nicht sofort darauf reagieren können.

Wenn Sie Kampagnen oder Journeys an sehr große Zielgruppen senden, überprüfen Sie die Durchsatzraten für Ihre Ursprungsnummern. Teilen Sie die Anzahl der Empfänger durch Ihre Durchsatzrate, um zu bestimmen, wie lange es dauert, Nachrichten an alle Ihre Empfänger zu senden.

Vermeiden Sie Ermüdungen durch die Nutzung mehrerer Kanäle

Wenn Sie in Ihren Kampagnen mehrere Kommunikationskanäle (z. B. E-Mail, SMS und Push-Nachrichten) verwenden, senden Sie nicht die gleiche Nachricht über jeden Kanal. Wenn Sie dieselbe Nachricht gleichzeitig über mehrere Kanäle senden, fühlen sich Ihre Kunden sehr wahrscheinlich dadurch eher gestört.

Verwenden dedizierter Kurzwahlnummern

Wenn Sie Kurzwahlnummern verwenden, unterhalten Sie eine separate Kurzwahlnummer für jede Marke und jeden Nachrichtentyp. Wenn Ihr Unternehmen z. B. zwei Marken hat, verwenden Sie für jede davon eine eigene Kurzwahlnummer. Wenn Sie Transaktions- und Werbenachrichten versenden, verwenden Sie auch dafür jeweils separate Kurzwahlnummern. Weitere Informationen zum Anfordern von Kurzcodes finden Sie unter Anfordern von Kurzcodes für SMS-Nachrichten mit AWS Endbenutzer-Messaging SMS im AWS Endbenutzer-Messaging SMS Benutzerhandbuch.

Bestätigen Sie Ihre Ziel-Telefonnummern

Wenn Sie SMS-Nachrichten über Amazon SNS senden, wird Ihnen jeder Nachrichtenteil, den Sie senden, in Rechnung gestellt. Der Preis, den Sie pro Nachrichtenteil zahlen, hängt vom Land oder der Region des Empfängers ab. Weitere Informationen zu SMS-Preisen finden Sie unter AWS Weltweite SMS-Preise.

Wenn Amazon SNS eine Anfrage zum Senden einer SMS-Nachricht akzeptiert (als Ergebnis eines SendMessagesAPI-Aufrufs oder als Ergebnis einer gestarteten Kampagne oder Journey), wird Ihnen das Senden dieser Nachricht in Rechnung gestellt. Diese Aussage gilt auch dann, wenn der beabsichtigte Empfänger die Nachricht nicht erhält. Wenn beispielsweise die Telefonnummer des Empfängers nicht mehr vergeben ist oder wenn die Nummer, an die Sie die Nachricht gesendet haben, keine gültige Mobiltelefonnummer war, wird Ihnen das Senden der Nachricht dennoch in Rechnung gestellt.

Amazon SNS akzeptiert gültige Anfragen zum Senden von SMS-Nachrichten und versucht, diese zuzustellen. Aus diesem Grund sollten Sie überprüfen, ob die Telefonnummern, an die Sie Nachrichten senden, gültige Mobiltelefonnummern sind. Sie können AWS Endbenutzer-Messaging SMS damit eine Testnachricht senden, um festzustellen, ob eine Telefonnummer gültig ist und um welche Art von Nummer es sich handelt (z. B. Mobilfunk, Festnetz oder VoIP). Weitere Informationen finden Sie im AWS Endbenutzer-Messaging SMS Benutzerhandbuch unter Senden einer Testnachricht mit dem SMS-Simulator.

Erstellen Sie Entwürfe unter Beachtung der Redundanz

Für geschäftskritische Messaging-Programme empfehlen wir, Amazon SNS in mehr als einer AWS-Region zu konfigurieren. Amazon SNS ist in mehreren AWS-Regionen Varianten erhältlich. Eine Liste der Regionen, in denen Amazon SNS erhältlich ist, finden Sie in der Allgemeine AWS-Referenz.

Die Telefonnummern, die Sie für SMS-Nachrichten verwenden – einschließlich Kurzwahl-, Langwahl-, gebührenfreien Nummern und 10DLC-Nummern – können in AWS-Regionen nicht repliziert werden. Damit Sie Amazon SNS in mehreren Regionen verwenden können, müssen Sie daher in jeder Region, in der Sie Amazon SNS verwenden möchten, separate Telefonnummern anfordern. Wenn Sie beispielsweise eine Kurzvorwahl verwenden, um Textnachrichten an Empfänger in den Vereinigte Staaten zu senden, müssen Sie in jeder, AWS-Region die Sie verwenden möchten, separate Kurzcodes anfordern.

In einigen Ländern können Sie auch mehrere Arten von Telefonnummern verwenden, um die Redundanz zu erhöhen. In den USA können Sie beispielsweise Kurzwahlnummern, 10DLC-Nummern und gebührenfreie Nummern anfordern. Jeder dieser Telefonnummerntypen nimmt einen anderen Weg zum Empfänger. Die Verfügbarkeit mehrerer Rufnummerntypen — entweder in derselben AWS-Region oder auf mehrere verteilt AWS-Regionen— bietet eine zusätzliche Redundanzebene, die zur Verbesserung der Ausfallsicherheit beitragen kann.

SMS-Limits und -Einschränkungen

Informationen zu den Beschränkungen und Beschränkungen für SMS finden Sie unter Beschränkungen und Einschränkungen für SMS und MMS im AWS Endbenutzer-Messaging SMS Benutzerhandbuch.

Verwalten von Schlüsselwörtern für die Abmeldung

SMS-Empfänger können ihre Geräte verwenden, um Nachrichten abzubestellen, indem sie mit einem Schlüsselwort antworten. Weitere Informationen finden Sie unter Deaktivieren des Empfangs von SMS-Nachrichten.

CreatePool

Verwenden Sie die API-Aktion `CreatePool`, um einen neuen Pool zu erstellen und diesem Pool eine festgelegte Ursprungsidentität zuzuordnen. Weitere Informationen finden Sie CreatePoolunter AWS Endbenutzer-Messaging SMS API-Referenz.

PutKeyword

Verwenden Sie die `PutKeyword` API-Aktion, um eine Schlüsselwortkonfiguration für eine ursprüngliche Telefonnummer oder einen Pool zu erstellen oder zu aktualisieren. Weitere Informationen finden Sie PutKeywordunter AWS Endbenutzer-Messaging SMS API-Referenz.

Verwalten von Nummerneinstellungen

Informationen zum Verwalten der Einstellungen für die speziellen Kurz- und Langcodes, die Sie beim AWS Support angefordert und Ihrem Konto zugewiesen haben, finden Sie unter Ändern der Funktionen einer Telefonnummer mit AWS CLI der Eingabe AWS Endbenutzer-Messaging SMS.

SMS-Zeichenbeschränkungen in Amazon SNS

Eine einzelne SMS-Nachricht kann bis zu 140 Bytes enthalten. Die Anzahl der Zeichen für eine einzelne SMS-Nachricht hängt vom Typ der in der Nachricht verwendeten Zeichen ab.

Wenn Ihre Nachricht nur Zeichen aus dem Zeichensatz GSM 03.38, auch als GSM 7-Bit-Alphabet bekannt, enthält, sind bis zu 160 Zeichen zulässig. Enthält Ihre Nachricht nicht im Zeichensatz GSM 03.38 enthaltene Zeichen, sind 70 Zeichen zulässig. Wenn Sie eine SMS-Nachricht senden, bestimmt Amazon SNS automatisch die effizienteste Kodierung.

Enthält eine Nachricht mehr als die maximale Anzahl von Zeichen, wird die Nachricht in mehrere Teile aufgeteilt. Wenn Nachrichten in mehrere Teile aufgeteilt werden, enthält jedes Teil zusätzliche Informationen über den Nachrichtenteil, der ihm vorangestellt ist. Empfängt das Gerät des Empfängers dann die so getrennten Nachrichtenteile, verwendet es diese zusätzlichen Informationen, um alle Nachrichtenteile in der richtigen Reihenfolge anzuzeigen. Je nach Mobilfunkanbieter und Gerät des Empfängers können mehrere Nachrichten als eine Nachricht oder als Sequenz von separaten Nachrichten angezeigt werden. Daher verringert sich die Anzahl der Zeichen in den einzelnen Nachrichten auf 153 (für Nachrichten, die nur GSM 03.38 Zeichen enthalten) oder auf 67 (für Nachrichten, die andere Zeichen enthalten). Sie können vor dem Senden schätzen, wie viele Nachrichtenteile Ihre Nachricht enthält, indem Sie einen SMS-Längenrechner verwenden (es sind mehrere online verfügbar). Die maximal unterstützte Größe einer Nachricht beträgt 1600 GSM-Zeichen oder 630 Nicht-GSM-Zeichen. Weitere Informationen zum Durchsatz und zur Nachrichtengröße finden Sie unter Begrenzungen für SMS-Zeichen in Amazon Pinpoint im Amazon-Pinpoint-Benutzerhandbuch.

Wenn Sie die Anzahl der Nachrichtenteile für jede gesendete Nachricht anzeigen möchten, sollten Sie zuerst die Ereignisstreamingeinstellungen aktivieren. Dann erzeugt Amazon

SNS ein `_SMS.SUCCESS`-Ereignis, sobald die Nachricht an den Mobilfunkanbieter des Empfängers zugestellt wurde. Der `_SMS.SUCCESS`-Ereignisdatensatz enthält ein Attribut namens `attributes.number_of_message_parts`. Dieses Attribut gibt die Anzahl der Nachrichtenteile an, die in der Nachricht enthalten sind.

> **Important**
>
> Wenn Sie eine Nachricht mit mehr als einem Nachrichtenteil senden, wird Ihnen die Anzahl der enthaltenen Nachrichtenteile in Rechnung gestellt.

Zeichensatz GSM 03.38

In der folgenden Tabelle sind alle Zeichen aus dem Zeichensatz GSM 03.38 aufgeführt. Wenn Sie eine Nachricht senden, die nur Zeichen aus der folgenden Tabelle enthält, kann die Nachricht bis zu 160 Zeichen enthalten.

GSM 03.38-Standardzeichen												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	i
n	o	p	q	r	S	t	u	V	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

Der Zeichensatz GSM 03.38 enthält neben den in der vorherigen Tabelle gezeigten Symbole noch weitere. Diese Zeichen zählen jedoch als zwei Zeichen, da sie alle ein unsichtbares Escape-Zeichen enthalten:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Darüber hinaus enthält der Zeichensatz GSM 03.38 die folgenden Zeichen, die nicht gedruckt werden:

- Ein Leerzeichen.
- Ein Zeilenvorschub-Steuerzeichen, das das Ende eine Textzeile und den Beginn einer neuen markiert.
- Ein Wagenrücklauf-Steuerzeichen, das zum Anfang einer Textzeile führt (folgt in der Regel einem Zeilenvorschubzeichen).
- Ein Escape-Steuerzeichen, das automatisch zu den Zeichen in der vorherigen Liste hinzugefügt wird.

Beispielnachrichten

Dieser Abschnitt enthält Beispiele für SMS-Nachrichten. In diesem Abschnitt werden für jedes Beispiel die Gesamtzahl der Zeichen sowie die Anzahl der Nachrichtenteile einer Nachricht angezeigt.

Beispiel 1: Lange Nachricht, die nur Zeichen im GSM 03.38-Alphabet enthält

Die folgende Nachricht enthält nur Zeichen, die im GSM 03.38-Alphabet enthalten sind.

```
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.
```

Die obige Nachricht enthält 180 Zeichen und muss daher in mehrere Nachrichtenteile aufgeteilt werden. Wenn eine Nachricht in mehrere Nachrichtenteile aufgeteilt wird, kann jedes Teil 153 GSM 03.38-Zeichen enthalten. Daher wird diese Nachricht in zwei Teilen gesendet.

Beispiel 2: Nachricht, die Multibyte-Zeichen enthält

Die folgende Nachricht enthält chinesische Zeichen, die nicht dem GSM 03.38-Alphabet angehören.

```
##################################################·####1994#7#########
```

Die obige Nachricht enthält 71 Zeichen. Da jedoch fast alle Zeichen in der Nachricht nicht im GSM 03.38-Alphabet enthalten sind, werden zwei Nachrichtenteile gesendet. Jedes dieser Nachrichtenteile kann maximal 67 Zeichen enthalten.

Beispiel 3: Nachricht, die ein einzelnes Nicht-GSM-Zeichen enthält

Die folgende Nachricht enthält ein Zeichen, das nicht Teil des GSM 03.38-Alphabets ist. In diesem Beispiel ist das Zeichen ein schließendes einfaches Anführungszeichen (’), das kein regulärer Apostroph (') ist. Textverarbeitungsanwendungen wie Microsoft Word ersetzen häufig automatisch Apostrophe durch schließende einfache Anführungszeichen. Wenn Sie Ihre SMS-Nachrichten in Microsoft Word entwerfen und in Amazon SNS einfügen, sollten Sie diese Sonderzeichen entfernen und durch Apostrophe ersetzen.

```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

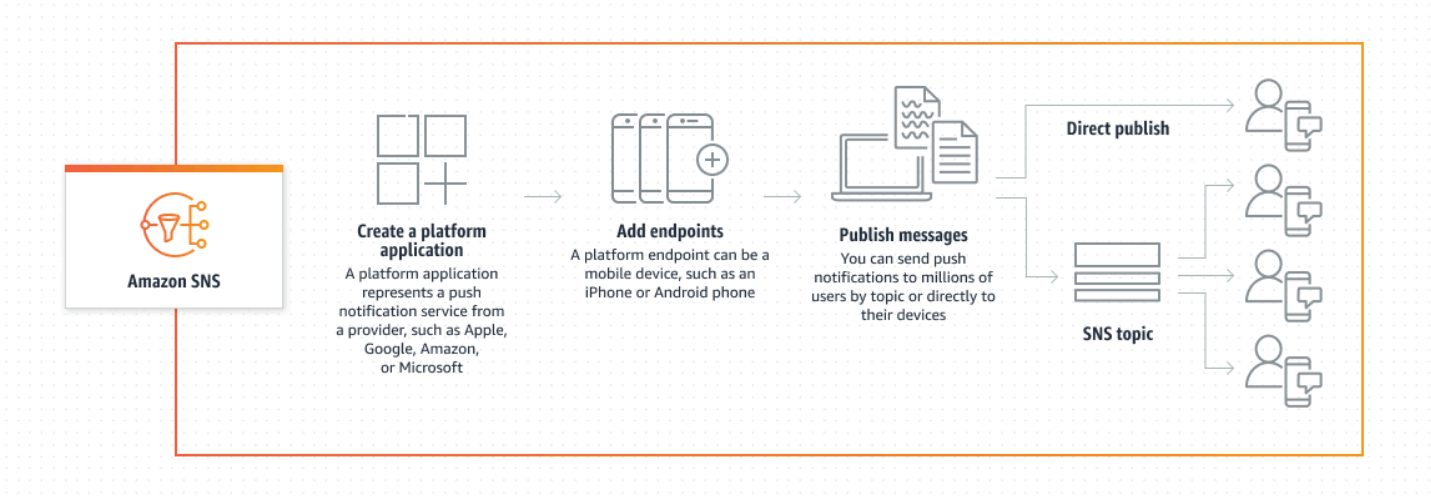
Die obige Nachricht enthält 130 Zeichen. Da sie jedoch das schließende einfache Anführungszeichen enthält, das nicht Teil des GSM 03.38-Alphabets ist, werden zwei Nachrichtenteile gesendet.

Wenn Sie das schließende einfache Anführungszeichen in dieser Nachricht durch ein Apostroph (das im GSM 03.38-Alphabet enthalten ist) ersetzen, wird nur eine Nachricht gesendet.

Senden von mobilen Push-Benachrichtigungen mit Amazon SNS

Sie können Amazon SNS verwenden, um Push-Benachrichtigungen direkt an Apps auf Mobilgeräten zu senden. An einen mobilen Endpunkt gesendete Push-Benachrichtigungen können in der mobilen

App als Nachrichtenwarnungen, Badge-Updates oder akustische Benachrichtigungen angezeigt werden.



Themen

- So funktionieren Amazon SNS SNS-Benutzerbenachrichtigungen
- Push-Benachrichtigungen mit Amazon SNS einrichten
- Eine mobile App in Amazon SNS einrichten
- Amazon SNS für mobile Push-Benachrichtigungen verwenden
- Attribute der mobilen Amazon SNS SNS-App
- Benachrichtigungen über Amazon SNS SNS-Anwendungsereignisse für mobile Anwendungen
- Mobile Push-API-Aktionen
- Häufige Amazon SNS SNS-Push-API-Fehler für Mobilgeräte
- Verwendung des Time-to-Live-Nachrichten-Attributs von Amazon SNS für mobile Push-Benachrichtigungen
- Unterstützte Regionen für mobile Amazon SNS SNS-Anwendungen
- Bewährte Methoden für die Verwaltung von mobilen Amazon SNS SNS-Push-Benachrichtigungen

So funktionieren Amazon SNS SNS-Benutzerbenachrichtigungen

Senden Sie Push-Benachrichtigungen an mobile Geräte und Desktops mithilfe einer der folgenden unterstützten Push-Benachrichtigungsservices:

- Amazon Device Messaging (ADM)
- Apple Push Notification Service (APNs) für iOS und Mac OS X

- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Microsoft Push Notification Service für Windows Phone (MPNS)
- Windows Push Notification Services (WNS)

Push-Benachrichtigungsdienste wie FCM APNs und FCM halten eine Verbindung zu jeder App und jedem zugehörigen Mobilgerät aufrecht, die für die Nutzung ihres Dienstes registriert sind. Wenn eine App oder ein Mobilgerät registriert werden, gibt der Push-Benachrichtigungsservice ein Gerätetoken zurück. Amazon SNS verwendet das Gerätetoken, um einen mobilen Endpunkt zu erstellen, zu dem mobile Push-Benachrichtigungen direkt gesendet werden können. Damit Amazon SNS mit den verschiedenen Push-Benachrichtigungs-Services kommunizieren kann, müssen Sie Amazon SNS Ihre Anmeldeinformationen für die Push-Benachrichtigungs-Services mitteilen, um sie in Ihrem Namen zu verwenden. Weitere Informationen finden Sie unter Push-Benachrichtigungen mit Amazon SNS einrichten.

Zusätzlich zum Senden von direkten Push-Benachrichtigungen, können Sie mit Amazon SNS auch Benachrichtigungen an mobile Endpunkte senden, die ein Thema abonniert haben. Das Konzept ist das gleiche, als wenn andere Endpunkttypen, wie Amazon SQS, HTTP/S, E-Mail und SMS ein Thema abonnieren, wie unter Was ist Amazon SNS? beschrieben. Der Unterschied besteht darin, dass Amazon SNS über Push-Benachrichtigungsservices kommuniziert, damit die abonnierten mobilen Endpunkte die an das Thema gesendeten Benachrichtigungen empfangen können.

Push-Benachrichtigungen mit Amazon SNS einrichten

1. Rufen Sie die Anmeldeinformationen und das Geräte-Token für die mobilen Plattformen ab, die Sie unterstützen möchten.
2. Verwenden Sie die Anmeldeinformationen, um ein Plattformanwendungsobjekt (`PlatformApplicationArn`) mit Amazon SNS zu erstellen. Weitere Informationen finden Sie unter Erstellen einer Amazon SNS-Plattformanwendung.
3. Verwenden Sie die zurückgegebenen Anmeldeinformationen, um für Ihre mobile App und Ihr Gerät ein Geräte-Token von Push-Benachrichtigungs-Service anzufordern. Das erhaltene Token repräsentiert Ihre mobile App und Ihr Gerät.
4. Erstellen Sie unter Verwendung des Geräte-Tokens und des `PlatformApplicationArn` mit Amazon SNS ein Endpunkt-Objekt für die Plattform (`EndpointArn`). Weitere Informationen finden Sie unter Einrichtung eines Amazon SNS SNS-Plattformendpunkts für mobile Benachrichtigungen.

5. Verwenden Sie den `EndpointArn`, um eine Nachricht an eine App auf einem Mobilgerät zu veröffentlichen. Weitere Informationen finden Sie unter Direkte Amazon SNS SNS-Nachrichtenübermittlung auf Mobilgeräten und der API Veröffentlichen in der Amazon Simple Notification Service API-Referenz.

Eine mobile App in Amazon SNS einrichten

In diesem Thema wird beschrieben, wie Sie mobile Anwendungen AWS Management Console mithilfe der unter beschriebenen Informationen einrichtenVoraussetzungen für Amazon SNS-Benutzerbenachrichtigungen.

Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen

Um mit der Nutzung von mobilen Amazon SNS SNS-Push-Benachrichtigungen zu beginnen, benötigen Sie Folgendes:

- Eine Reihe von Anmeldeinformationen für die Verbindung mit einem der unterstützten Push-Benachrichtigungsdienste: ADM, Baidu APNs, FCM, MPNS oder WNS.
- Ein Geräte-Token oder eine Registrierungs-ID für die mobile App und das Gerät.
- Entsprechende Amazon SNS-Konfiguration, sodass Push-Benachrichtigungen an mobile Endpunkte gesendet werden.
- Eine mobile App, die registriert und so konfiguriert ist, dass sie einen der unterstützten Push-Benachrichtigungsservices verwenden kann.

Für die Registrierung Ihrer Anwendung mit einem Push-Benachrichtigungsservice sind mehrere Schritte erforderlich. Amazon SNS benötigt einige Informationen, die Sie dem Push-Benachrichtigungsservice mitgeteilt haben, um Push-Benachrichtigungen direkt an den mobilen Endpunkt zu senden. Im Allgemeinen benötigen Sie die erforderlichen Anmeldeinformationen zur Verbindungsherstellung mit dem Push-Benachrichtigungsservice, das vom Push-Benachrichtigungsservice bereitgestellte Geräte-Token oder eine Registrierungs-ID (die Ihr Mobilgerät und Ihre mobile App repräsentieren) sowie die mobile App, die mit dem Push-Benachrichtigungsservice registriert wurde.

Die genaue Art der Anmeldeinformationen kann zwischen den mobilen Plattformen variieren, jedoch müssen sie in jedem Fall beim Erstellen einer Verbindung mit der Plattform angegeben werden. An jede mobile App wird ein Satz an Anmeldeinformationen ausgegeben und dieser muss verwendet werden, um eine Nachricht an eine beliebige Instance dieser App zu senden.

Die spezifischen Namen variieren, je nachdem welcher Push-Benachrichtigungsservice verwendet wird. Wenn Sie es beispielsweise APNs als Push-Benachrichtigungsdienst verwenden, benötigen Sie ein Geräte-Token. Alternativ wird das Geräte-Token bei Verwendung von FCM Registrierungs-ID genannt. Beim Geräte-Token oder bei der Registrierungs-ID handelt es sich um eine Zeichenfolge, die vom Betriebssystem des Mobilgeräts an die Anwendung gesendet wird. Auf diese Weise wird eine Instance einer mobilen App auf einem bestimmten Mobilgerät gekennzeichnet und fungiert so als eindeutige Bezeichnung dieser App/Geräte-Paarung.

Amazon SNS speichert die Anmeldeinformationen (sowie einige andere Einstellungen) als Plattformanwendungsressource. Die Geräte-Token (wiederum mit einigen zusätzlichen Einstellungen) werden als Objekte dargestellt, die als Plattformendpunkte bezeichnet werden. Jeder Plattformendpunkt gehört zu einer bestimmten Plattformanwendung und mit jedem Plattformendpunkt kann mithilfe der Anmeldeinformationen, die in der entsprechenden Plattformanwendung gespeichert sind, kommuniziert werden.

Die folgenden Abschnitte beschreiben die Voraussetzungen für jeden der unterstützten Push-Benachrichtigungsservices. Sobald Sie die erforderlichen Informationen erhalten haben, können Sie eine Push-Benachrichtigung mit dem AWS Management Console oder dem Amazon SNS SNS-Push APIs für Mobilgeräte senden. Weitere Informationen finden Sie unter Push-Benachrichtigungen mit Amazon SNS einrichten.

Erstellen einer Amazon SNS-Plattformanwendung

Um Benachrichtigungen von Amazon SNS an mobile Endpunkte zu senden — ob direkt oder über Abonnements für ein Thema — müssen Sie zunächst eine Plattformanwendung erstellen. Nach der Registrierung der App bei AWS müssen Sie einen Endpunkt sowohl für die App als auch für das Mobilgerät erstellen. Dieser Endpunkt ermöglicht Amazon SNS, Nachrichten an das Gerät zu senden.

So erstellen Sie eine Plattformanwendung

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Push-Benachrichtigungen aus.
3. Wählen Sie im Abschnitt Platform applications (Plattformanwendungen) die Option Create platform applications (Plattformanwendung erstellen) aus.
4. Wählen Sie Ihre AWS-Region. Eine Liste der AWS -Regionen, in denen Sie mobile Anwendungen erstellen können, finden Sie unter Unterstützte Regionen für mobile Amazon SNS SNS-Anwendungen.
5. Geben Sie die folgenden Anwendungsdetails ein:

- Anwendungsname — Geben Sie einen Namen für Ihre Plattformanwendung ein. Der Name muss zwischen 1 und 256 Zeichen lang sein und kann Groß- und Kleinbuchstaben, Zahlen, Unterstriche, Bindestriche und Punkte enthalten.
- Plattform für Push-Benachrichtigungen — Wählen Sie den entsprechenden Benachrichtigungsdienst aus, bei dem die App registriert ist (z. B. Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM)).

6. Abhängig von der ausgewählten Plattform müssen Sie bestimmte Anmeldeinformationen angeben:

   - Für APNs(Apple Push Notification Service) — Wählen Sie zwischen tokenbasierter oder zertifikatsbasierter Authentifizierung.
     - Laden Sie für die tokenbasierte Authentifizierung eine P8-Datei hoch (generiert über Keychain Access).
     - Laden Sie für die zertifikatsbasierte Authentifizierung eine .p12-Datei hoch (ebenfalls aus Keychain Access exportiert).
   - Für FCM (Firebase Cloud Messaging) — Geben Sie den Serverschlüssel aus der Firebase-Konsole ein.
   - Für andere Plattformen (wie ADM oder GCM) — Geben Sie die entsprechenden API-Schlüssel oder Anmeldeinformationen ein.
7. Nachdem Sie die erforderlichen Details eingegeben haben, wählen Sie Plattformanwendung erstellen. Diese Aktion registriert die App bei Amazon SNS und erstellt das entsprechende Plattformanwendungsobjekt.
8. Bei der Erstellung generiert Amazon SNS einen `PlatformApplicationArn`(Amazon-Ressourcennamen) und gibt ihn zurück. Dieser ARN identifiziert Ihre Plattformanwendung eindeutig und wird bei der Erstellung von Endpunkten für mobile Geräte verwendet.

Einrichtung eines Amazon SNS SNS-Plattformendpunkts für mobile Benachrichtigungen

Wenn sich eine App und ein Mobilgerät bei einem Push-Benachrichtigungsdienst (wie APNs Firebase Cloud Messaging) registrieren, gibt der Push-Benachrichtigungsdienst ein Geräte-Token zurück. Amazon SNS verwendet dieses Geräte-Token, um einen Plattformendpunkt zu erstellen, der als Ziel für das Senden direkter Push-Benachrichtigungen an die App auf dem Gerät dient. Der Plattformendpunkt dient als Brücke und leitet Nachrichten, die von Amazon SNS gesendet

werden, an den Push-Benachrichtigungsdienst weiter, um sie an das entsprechende Mobilgerät weiterzuleiten. Weitere Informationen erhalten Sie unter Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen und Push-Benachrichtigungen mit Amazon SNS einrichten.

Geräte-Token und Plattformendpunkte verstehen

Ein Geräte-Token identifiziert eindeutig ein Mobilgerät, das bei einem Push-Benachrichtigungsdienst registriert ist (z. APNs B. Firebase Cloud Messaging). Wenn sich eine App beim Push-Benachrichtigungsdienst registriert, generiert sie ein Geräte-Token, das für diese App und dieses Gerät spezifisch ist. Amazon SNS verwendet dieses Geräte-Token, um einen Plattformendpunkt innerhalb der entsprechenden Plattformanwendung zu erstellen.

Der Plattformendpunkt ermöglicht es Amazon SNS, über den Push-Benachrichtigungsdienst Push-Benachrichtigungen an das Gerät zu senden und so die Verbindung zwischen Ihrer App und dem Gerät des Benutzers aufrechtzuerhalten.

Erstellen eines Plattformendpunkts

Um Push-Benachrichtigungen zu einer App per Amazon SNS zu senden, muss zuerst das Geräte-Token der App bei Amazon SNS registriert werden, indem die "create platform endpoint"-Aktion aufgerufen wird. Diese Aktion verwendet den Amazon Resource Name (ARN) der Plattformanwendung und das Geräte-Token als Parameter und gibt den ARN des erstellten Plattformendpunkts zurück.

Die `CreatePlatformEndpoint`Aktion bewirkt Folgendes:

- Wenn der Plattformendpunkt bereits existiert, erstellen Sie ihn nicht erneut. Dem Aufrufer wird der ARN des vorhandenen Plattformendpunkts zurückgegeben.
- Wenn der Plattformendpunkt mit demselben Geräte-Token, aber unterschiedlichen Einstellungen bereits vorhanden ist, erstellen Sie ihn nicht erneut. Es wird eine Ausnahme für den Aufrufer ausgelöst.
- Wenn der Plattformendpunkt nicht existiert, erstellen Sie ihn. Dem Aufrufer wird der ARN des neu erstellten Plattformendpunkts zurückgegeben.

Sie sollten die Aktion nicht direkt bei jedem Start der Anwendung aufrufen. Sie sorgt nicht immer für die Bereitstellung eines funktionierenden Endpunkts. Dies kann zum Beispiel der Fall sein, wenn eine App auf einem Gerät deinstalliert und neu installiert wird und der Endpunkt für diese bereits vorhanden aber deaktiviert ist. Ein erfolgreicher Registrierungsprozess sollte folgendermaßen aussehen:

1. Stellen Sie sicher, dass ein Plattformendpunkt für diese App-Geräte-Kombination vorhanden ist.
2. Stellen Sie sicher, dass das Geräte-Token im Plattformendpunkt das neueste gültige Geräte-Token ist.
3. Stellen Sie sicher, dass der Plattformendpunkt aktiviert und bereit ist.

Pseudo-Code

Der folgende Pseudo-Code beschreibt ein empfohlenes Verfahren für die Erstellung eines funktionierenden, aktuellen, aktivierten Plattformendpunkts in einer Vielzahl von Startbedingungen. Dieser Ansatz funktioniert, wenn die App zum ersten Mal oder erneut registriert wird. Es ist egal, ob der Plattformendpunkt für diese App bereits vorhanden ist, ob der Plattformendpunkt aktiviert ist, ob er das richtige Geräte-Token hat und so weiter. Der Code kann mehrfach hintereinander aufgerufen werden. Er wird keine doppelten Plattformendpunkte erstellen oder einen vorhandenen Plattformendpunkt ändern, der bereits auf dem neuesten Stand und aktiviert ist.

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
  # this is a first-time registration
  call create platform endpoint
  store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
  # the platform endpoint was deleted
  call create platform endpoint with the latest device token
  store the returned platform endpoint ARN
else
  if (the device token in the endpoint does not match the latest one) or
      (GetEndpointAttributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

Dieser Ansatz kann jederzeit verwendet werden, wenn die App sich registrieren oder erneut registrieren möchte. Er kann auch beim Benachrichtigen von Amazon SNS über eine Änderung des Geräte-Tokens verwendet werden. In diesem Fall können Sie einfach eine Aktion mit dem neuesten Geräte-Token-Wert aufrufen. Es gibt einige wichtige Punkte zu diesem Ansatz:

- Es gibt zwei Fälle, in denen die „create platform endpoint“-Aktion aufgerufen werden kann. Sie kann direkt zu Beginn aufgerufen werden, wenn die App ihren eigenen Plattformendpunkt-ARN nicht kennt (z. B. während der Erstregistrierung). Es wird auch aufgerufen, wenn der erste `GetEndpointAttributes` Aktionsaufruf mit einer nicht gefundenen Ausnahme fehlschlägt, was der Fall wäre, wenn die Anwendung ihren Endpunkt-ARN kennt, dieser aber gelöscht wurde.
- Die `GetEndpointAttributes` Aktion wird aufgerufen, um den Status des Plattformendpunkts zu überprüfen, auch wenn der Plattformendpunkt gerade erstellt wurde. Dies geschieht, wenn der Plattformendpunkt bereits vorhanden, aber deaktiviert ist. In diesem Fall ist die „create platform endpoint“-Aktion erfolgreich, aber der Plattformendpunkt wird nicht aktiviert. Daher müssen Sie den Status des Plattformendpunkts vor der Rückgabe eines Erfolgs prüfen.

AWS SDK-Beispiel

Der folgende Code zeigt, wie der vorherige Pseudocode mithilfe der Amazon SNS SNS-Clients implementiert wird, die von der bereitgestellt werden. AWS SDKs

Um ein AWS SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

CLI

AWS CLI

So erstellen Sie ein Plattformanwendungsendpunkt

Im folgenden `create-platform-endpoint`-Beispiel wird mithilfe des angegebenen Tokens ein Endpunkt für die angegebene Plattformanwendung erstellt.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Ausgabe:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
```

```
}
```

Java

SDK für Java 2.x

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>
```

```
            Where:
               token - The device token or registration ID of the mobile device.
This is a unique
               identifier provided by the device platform (e.g., Apple Push
Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
push notifications.

               platformApplicationArn - The ARN value of platform application.
You can get this value from the AWS Management Console.\s

            """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
response.endpointArn());

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

```
            }
        }
    }
```

Weitere Informationen finden Sie unter Mobile Push-API-Aktionen.

Fehlerbehebung

Wiederholtes Aufrufen von „Create Platform Endpoint“ mit einem veralteten Geräte-Token

Insbesondere bei FCM-Endpunkten denken Sie vielleicht, dass es am besten ist, das erste Geräte-Token zu speichern, das für die Anwendung ausgestellt wurde, und dann bei jedem Start der Anwendung den Create Platform-Endpunkt mit diesem Geräte-Token aufzurufen. Dies scheint sinnvoll, da die App sich nicht mehr um den Status des Geräte-Tokens kümmern muss und Amazon SNS das Geräte-Token automatisch auf den aktuellen Wert aktualisiert. Diese Lösung hat jedoch mehrere ernsthafte Probleme:

- Amazon SNS stützt sich auf das Feedback von FCM, um abgelaufene Geräte-Tokens durch neue Geräte-Tokens zu aktualisieren. FCM erhält einige Zeit Informationen zu alten Geräte-Tokens. Dies gilt jedoch nicht unbegrenzt. Sobald FCM die Verbindung zwischen dem alten Geräte-Token und dem neuen Geräte-Token vergessen hat, ist Amazon SNS nicht mehr in der Lage, das im Plattformendpunkt gespeicherte Geräte-Token auf den korrekten Wert zu aktualisieren. Stattdessen wird der Plattformendpunkt einfach deaktiviert.
- Die Plattformanwendung enthält mehrere Plattformendpunkte mit demselben Geräte-Token.
- Amazon SNS arbeitet mit einem Kontingent an Plattformendpunkten, die gleichzeitig mit demselben Geräte-Token gestartet werden können. Irgendwann wird die Erstellung neuer Endpunkte mit einer „Invalid Parameter“-Ausnahme und der Fehlermeldung „This endpoint is already registered with a different token.“ fehlschlagen.

Weitere Informationen zur Verwaltung von FCM-Endpunkten finden Sie unter. Amazon SNS SNS-Verwaltung von Firebase Cloud Messaging-Endpunkten

Erneutes Aktivieren eines Plattformendpunktes mit einem ungültigen Geräte-Token

Wenn eine mobile Plattform (wie APNs oder FCM) Amazon SNS darüber informiert, dass das in der Veröffentlichungsanfrage verwendete Geräte-Token ungültig war, deaktiviert Amazon SNS den mit diesem Geräte-Token verknüpften Plattformendpunkt. Amazon SNS weist dann nachfolgende

Veröffentlichungen für das Geräte-Token zurück. Möglicherweise glauben Sie, Sie könnten einfach den Plattformendpunkt neu aktivieren und so weiter veröffentlichen, doch wird dies in den meisten Fällen nicht funktionieren. Die veröffentlichten Nachrichten werden nicht zugestellt und der Plattformendpunkt wird bald wieder deaktiviert.

Der Grund hierfür ist, dass das Geräte-Token des Plattformendpunkts wirklich ungültig ist. Zustellungen können nicht abgeschlossen werden, da es nicht mehr mit einer installierten App übereinstimmt. Bei der nächsten Veröffentlichung informiert die mobile Plattform Amazon SNS erneut darüber, dass das Geräte-Token ungültig ist. Amazon SNS wird den Plattformendpunkt wieder deaktivieren.

Um einen deaktivierten Plattformendpunkt zu reaktivieren, muss dieser einem gültigen Geräte-Token zugeordnet (über einen „set endpoint attributes“-Aktionsaufruf) sein und dann aktiviert werden. Nur dann werden Zustellungen an den Plattformendpunkt erfolgreich sein. Das erneute Aktivieren eines Plattformendpunkts ohne Aktualisierung des Geräte-Tokens funktioniert nur, wenn ein dem Endpunkt zugeordnetes Geräte-Token ungültig ist und wieder gültig wird. Dies kann zum Beispiel der Fall sein, wenn eine App deinstalliert und dann auf demselben Mobilgerät neu installiert wurde und dasselbe Geräte-Token erhält. Der oben gezeigte Ansatz geht so vor. Er stellt sicher, dass der Plattformendpunkt erst dann neu aktiviert wird, wenn überprüft wurde, dass das ihm zugeordnete Geräte-Token im Moment verfügbar ist.

Integration von Geräte-Token mit Amazon SNS für mobile Benachrichtigungen

Wenn Sie eine App und ein Mobilgerät zum ersten Mal bei einem Benachrichtigungsdienst wie Apple Push Notification Service (APNs) und Firebase Cloud Messaging (FCM) registrieren, IDs werden Geräte-Token oder Registrierungen vom Service zurückgegeben. Diese Tokens/ IDs werden zu Amazon SNS hinzugefügt, um mithilfe der API einen Endpunkt für die App und das Gerät zu erstellen. `PlatformApplicationArn` Sobald der Endpunkt erstellt wurde, `EndpointArn`wird ein zurückgegeben, das Amazon SNS verwendet, um Benachrichtigungen an die richtige App/das richtige Gerät weiterzuleiten.

Sie können Geräte-Tokens oder Registrierungen IDs auf folgende Weise zu Amazon SNS hinzufügen:

- Fügen Sie manuell ein einzelnes Token hinzu über AWS Management Console
- Laden Sie mehrere Token mithilfe der API `CreatePlatformEndpoint` hoch
- Tokens für future Geräte registrieren

Um manuell ein Geräte-Token oder eine Registrierungs-ID hinzuzufügen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Push-Benachrichtigungen aus.
3. Wählen Sie im Abschnitt Plattformanwendungen Ihre Anwendung aus und klicken Sie dann auf Bearbeiten. Wenn Sie noch keine Plattformanwendung erstellt haben, folgen Sie der Erstellen einer Amazon SNS-Plattformanwendung Anleitung, um dies jetzt zu tun.
4. Klicken Sie auf Endpunkt erstellen.
5. Geben Sie im Feld Endpunkt-Token das Token oder die Registrierungs-ID ein, je nachdem, welchen Benachrichtigungsdienst Sie verwenden (z. B. die FCM-Registrierungs-ID).
6. (Optional) Geben Sie zusätzliche Daten in das Feld Benutzerdaten ein. Diese Daten müssen UTF-8-codiert sein und weniger als 2 KB groß sein.
7. Klicken Sie auf Endpunkt erstellen.

Sobald der Endpunkt erstellt ist, können Sie Nachrichten direkt an das Mobilgerät oder an Mobilgeräte senden, die ein Amazon SNS SNS-Thema abonniert haben.

Um mehrere Token mithilfe der API hochzuladen **`CreatePlatformEndpoint`**

Die folgenden Schritte zeigen, wie Sie die von bereitgestellte Beispiel-Java-App (`bulkuploadPaket`) verwenden AWS , um mehrere Token (Geräte-Token oder Registrierung IDs) auf Amazon SNS hochzuladen. Sie können diese Beispielanwendung verwenden, um die ersten Schritte zum Hochladen der vorhandenen Token zu unternehmen.

> **Note**
>
> Die folgenden Schritte verwenden die IDE Eclipse Java. Bei den Schritten wird davon ausgegangen, dass Sie die installiert haben AWS SDK für Java und über die AWS Sicherheitsanmeldeinformationen für Ihre AWS-Konto verfügen. Weitere Informationen finden Sie unter AWS SDK für Java. Weitere Informationen zu Anmeldeinformationen finden Sie unter AWS Sicherheitsanmeldedaten im IAM-Benutzerhandbuch.

1. Laden Sie die Datei snsmobilepush.zip herunter und entpacken Sie sie.
2. Erstellen Sie ein neues Java-Projekt in Eclipse und importieren Sie den `SNSSamples` Ordner in das Projekt.

3. Laden Sie die OpenCSV-Bibliothek herunter und fügen Sie sie dem Build-Pfad hinzu.
4. Geben Sie in der `BulkUpload.properties` Datei Folgendes an:
   - Ihr `ApplicationArn` (ARN der Plattformanwendung).
   - Der absolute Pfad zu Ihrer CSV-Datei, die die Token enthält.
   - Protokollierung von Dateinamen für erfolgreiche und fehlgeschlagene Token. Beispiel: `goodTokens.csv` und `badTokens.csv`.
   - (Optional) Eine Konfiguration für das Trennzeichen, das Anführungszeichen und die Anzahl der zu verwendenden Threads.

   Ihre vollständigen `BulkUpload.properties` sollten wie folgt aussehen:

   ```
   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
   ```
5. Führen Sie die BatchCreatePlatformEndpointSample.java-Anwendung aus, um die Token auf Amazon SNS hochzuladen. Erfolgreich hochgeladene Token werden angemeldet`goodTokens.csv`, während falsch formatierte Token angemeldet werden. `badTokens.csv`

Um Tokens von Geräten für future App-Installationen zu registrieren

Für diesen Vorgang haben Sie zwei Möglichkeiten:

Verwenden Sie den Amazon Cognito-Service

Ihre mobile App kann temporäre Sicherheitsanmeldedaten verwenden, um Endgeräte zu erstellen. Amazon Cognito wird empfohlen, temporäre Anmeldeinformationen zu generieren. Weitere Informationen finden Sie im Amazon Cognito Developer Guide

Um App-Registrierungen nachzuverfolgen, verwenden Sie Amazon SNS SNS-Ereignisse, um Benachrichtigungen zu erhalten, wenn neue Endpunkte erstellt ARNs werden.

Alternativ können Sie die `ListEndpointByPlatformApplication`API verwenden, um die Liste der registrierten Endpunkte abzurufen.

Verwenden eines Proxy-Servers

Wenn Ihre App-Infrastruktur die Geräteregistrierung bei der Installation bereits unterstützt, können Sie Ihren Server als Proxy verwenden. Es leitet Geräte-Token über die `CreatePlatformEndpoint`API an Amazon SNS weiter.

Der von Amazon SNS erstellte Endpunkt-ARN wird zurückgegeben und kann von Ihrem Server für die future Nachrichtenveröffentlichung gespeichert werden.

Amazon SNS Authentifizierungsmethoden für Apple-Push-Benachrichtigungen

Sie können Amazon SNS ermächtigen, Push-Benachrichtigungen an Ihre iOS- oder macOS-App zu senden, indem Sie Informationen bereitstellen, die Sie als Entwickler der App identifizieren. Um sich zu authentifizieren, geben Sie entweder einen Schlüssel oder ein Zertifikat beim Erstellen einer Plattformanwendung an, die Sie beide von Ihrem Apple Developer-Konto aus erhalten können.

Signierschlüssel für Token

Ein privater Signaturschlüssel, den Amazon SNS zum Signieren von Apple Push Notification Service (APNs) -Authentifizierungstoken verwendet.

Wenn Sie einen Signaturschlüssel angeben, verwendet Amazon SNS APNs für jede von Ihnen gesendete Push-Benachrichtigung ein Token zur Authentifizierung. Mit Ihrem Signaturschlüssel können Sie Push-Benachrichtigungen an APNs Produktions- und Sandbox-Umgebungen senden.

Ihr Signierschlüssel läuft nicht ab und Sie können denselben Signierschlüssel für mehrere Apps verwenden. Weitere Informationen finden Sie unter Kommunizieren mit APNs mithilfe von Authentifizierungstoken in der Hilfe zum Entwicklerkonto auf der Apple-Website.

Zertifikat

Ein TLS-Zertifikat, mit dem sich Amazon SNS authentifiziert, APNs wenn Sie Push-Benachrichtigungen senden. Sie erhalten das Zertifikat aus Ihrer Apple-Entwicklerkonto.

Zertifikate laufen nach einem Jahr ab. Wenn dies geschieht, müssen Sie ein neues Zertifikat erstellen und es Amazon SNS bereitstellen. Weitere Informationen finden Sie unter Herstellen einer zertifikatsbasierten Verbindung zu APNs auf der Apple Developer-Website.

So verwalten Sie APNs Einstellungen mit der AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Push-Benachrichtigungen aus.
3. Wählen Sie im Abschnitt Plattformanwendungen die Anwendung aus, deren APNs Einstellungen Sie bearbeiten möchten, und wählen Sie dann Bearbeiten aus. Wenn Sie noch keine Plattformanwendung erstellt haben, folgen Sie der Erstellen einer Amazon SNS-Plattformanwendung Anleitung, um dies jetzt zu tun.
4. Wählen Sie Bearbeiten, um die Einstellungen für Ihre Plattformanwendung zu ändern.
5. Wählen Sie im Abschnitt Authentifizierungstyp eine der folgenden Optionen aus:
   - Tokenbasierte Authentifizierung (empfohlen für moderne APNs Integrationen)
   - Zertifikatsbasierte Authentifizierung (ältere Methode)
6. Konfigurieren Sie Ihre Anmeldeinformationen basierend auf dem Authentifizierungstyp:
   - Für die tokenbasierte Authentifizierung:
     - Laden Sie die .p8-Datei hoch. Dabei handelt es sich um den Signaturschlüssel für das Authentifizierungstoken, den Sie von Ihrem Apple Developer-Konto heruntergeladen haben.
     - Geben Sie die Signaturschlüssel-ID ein, die Sie in Ihrem Apple-Entwicklerkonto finden. Navigieren Sie zu Certificates, IDs& Profiles, Keys und wählen Sie den Schlüssel aus, den Sie verwenden möchten.
     - Geben Sie die Team-ID aus Ihrem Apple Developer-Konto ein. Sie finden diese auf der Mitgliedschaftsseite.
     - Geben Sie die Bundle-ID ein, die Ihrer App zugewiesen ist. Sie finden diese unter Zertifikate IDs und Profile, App IDs.
   - Für die zertifikatsbasierte Authentifizierung:
     - Laden Sie die P12-Datei für Ihr TLS-Zertifikat hoch. Diese Datei kann aus Keychain Access auf macOS exportiert werden, nachdem Sie das Zertifikat von Ihrem Apple Developer-Konto heruntergeladen haben.
     - Wenn Sie Ihrem .p12-Zertifikat ein Passwort zugewiesen haben, geben Sie es hier ein.
7. Nachdem Sie die erforderlichen Anmeldeinformationen eingegeben haben, wählen Sie Änderungen speichern, um die Einstellungen zu aktualisieren.

Amazon SNS SNS-Integration mit Firebase Cloud Messaging-Authentifizierungseinrichtung

In diesem Thema wird beschrieben, wie Sie die erforderlichen FCM-API-Anmeldeinformationen (HTTP v1) von Google erhalten, um sie mit der AWS API zu verwenden, AWS CLI und die. AWS Management Console

> **Important**
>
> 26. März 2024 — Amazon SNS unterstützt die FCM-HTTP-v1-API für Apple-Geräte und Webpush-Ziele. Wir empfehlen Ihnen, Ihre vorhandenen mobilen Push-Anwendungen am oder vor dem 1. Juni 2024 auf die neueste FCM-HTTP-v1-API zu migrieren, um Anwendungsunterbrechungen zu vermeiden.
> 18. Januar 2024 — Amazon SNS hat die Unterstützung für die FCM-HTTP-v1-API für die mobile Übermittlung von Push-Benachrichtigungen an Android-Geräte eingeführt.
> 20. Juni 2023 — Google hat seine ältere HTTP-API für Firebase Cloud Messaging (FCM) als veraltet eingestuft. Amazon SNS unterstützt jetzt die Lieferung an alle Gerätetypen mithilfe der FCM-HTTP-v1-API. Wir empfehlen Ihnen, Ihre vorhandenen mobilen Push-Anwendungen am oder vor dem 1. Juni 2024 auf die neueste FCM-HTTP-v1-API zu migrieren, um Unterbrechungen zu vermeiden.

Sie können Amazon SNS ermächtigen, Push-Benachrichtigungen an Ihre Anwendungen zu senden, indem Sie Informationen bereitstellen, die Sie als Entwickler der App identifizieren. Geben Sie zur Authentifizierung entweder einen API-Schlüssel oder ein Token an, wenn Sie eine Plattformanwendung erstellen. Sie können die folgenden Informationen von Ihrer Firebase-Anwendungskonsole abrufen:

API-Schlüssel

Der API-Schlüssel ist eine Anmeldeinformation, die beim Aufrufen der Legacy-API von Firebase verwendet wird. Das FCM-Vermächtnis APIs wird am 20. Juni 2024 von Google entfernt. Wenn Sie derzeit einen API-Schlüssel als Plattformanmeldeinformation verwenden, können Sie die Plattformanmeldeinformation aktualisieren, indem Sie Token als Option auswählen und die zugehörige JSON-Datei für Ihre Firebase-Anwendung hochladen.

Token

Beim Aufrufen der HTTP v1-API wird ein kurzlebiges Zugriffstoken verwendet. Dies ist die von Firebase vorgeschlagene API zum Senden von Push-Benachrichtigungen. Zum Generieren von Zugriffstoken stellt Firebase Entwicklern eine Reihe von Anmeldeinformationen in Form einer privaten Schlüsseldatei (auch als Datei service.json bezeichnet) zur Verfügung.

Voraussetzung

Sie benötigen Ihre FCM-service.json-Anmeldeinformationen, bevor Sie mit der Verwaltung von FCM-Einstellungen in Amazon SNS beginnen können. Informationen zum Abrufen Ihrer service.json-Anmeldeinformationen finden Sie in der Google Firebase-Dokumentation unter Migration von Legacy FCM APIs zu HTTP v1.

Verwaltung der FCM-Einstellungen mit der CLI

Sie können FCM-Push-Benachrichtigungen mithilfe der AWS API erstellen. Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie im Allgemeine AWS-Referenz Handbuch unter Endpunkte und Kontingente von Amazon Simple Notification Service.

Um eine FCM-Push-Benachrichtigung zusammen mit einem Amazon SNS SNS-Thema (AWS API) zu erstellen

Bei Verwendung von Schlüssel-Anmeldeinformationen lautet `PlatformCredential` `API key`. Bei Verwendung von Token-Anmeldeinformationen ist `PlatformCredential` eine private Schlüsseldatei im JSON-Format:

- `CreatePlatformApplication`

Um einen FCM-Anmeldeinformationstyp für ein vorhandenes Amazon SNS SNS-Thema (API) abzurufen AWS

Ruft den Anmeldeinformationstyp `"AuthenticationMethod": "Token"` oder `"AuthenticationMethod": "Key"` ab:

- GetPlatformApplicationAttributes

So legen Sie ein FCM-Attribut für ein vorhandenes Amazon-SNS-Thema fest (AWS -API)

Legt das FCM-Attribut fest:

- SetPlatformApplicationAttributes

Verwaltung der FCM-Einstellungen über die Konsole

Sie können FCM-Push-Benachrichtigungen mit der AWS Command Line Interface (CLI) erstellen. Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

So erstellen Sie eine FCM-Push-Benachrichtigung zusammen mit einem Amazon-SNS-Thema (AWS CLI)

Bei Verwendung von Schlüssel-Anmeldeinformationen lautet `PlatformCredential` `API key`. Bei Verwendung von Token-Anmeldeinformationen ist `PlatformCredential` eine private Schlüsseldatei im JSON-Format. Bei Verwendung der AWS CLI muss die Datei im Zeichenkettenformat vorliegen und Sonderzeichen müssen ignoriert werden. Um die Datei korrekt zu formatieren, empfiehlt Amazon SNS die Verwendung des folgenden Befehls: `SERVICE_JSON=` `jq @json <<< cat service.json` `

- create-platform-application

So rufen Sie einen FCM-Anmeldeinformationstyp für ein vorhandenes Amazon-SNS-Thema ab (AWS CLI)

Ruft den Anmeldeinformationstyp `"AuthenticationMethod": "Token"` oder `"AuthenticationMethod": "Key"` ab:

- get-platform-application-attributes

So legen Sie ein FCM-Attribut für ein vorhandenes Amazon-SNS-Thema fest (AWS CLI)

Legt das FCM-Attribut fest:

- set-platform-application-attributes

Verwalten von FCM-Einstellungen (Konsole)

Gehen Sie wie folgt vor, um Ihre Firebase Cloud Messaging (FCM) -Anmeldeinformationen in Amazon SNS einzugeben und zu verwalten.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Push-Benachrichtigungen aus.
3. Wählen Sie im Abschnitt Plattformanwendungen die FCM-Plattformanwendung aus, deren Anmeldeinformationen Sie bearbeiten möchten, und wählen Sie dann Bearbeiten aus.
4. Wählen Sie im Abschnitt Firebase Cloud Messaging-Anmeldeinformationen eine der folgenden Optionen aus:
   - Tokenbasierte Authentifizierung (empfohlene Methode) — Laden Sie die private Schlüsseldatei (JSON) hoch, die Sie von der Firebase-Konsole heruntergeladen haben. Diese Datei enthält die Anmeldeinformationen, die zum Generieren kurzlebiger Zugriffstoken für FCM-Benachrichtigungen erforderlich sind. Um diese Datei zu erhalten:
     1. Gehen Sie zu Ihrer Firebase-Anwendungskonsole.
     2. Wählen Sie in den Projekteinstellungen Cloud Messaging aus.
     3. Laden Sie die JSON-Datei mit dem privaten Schlüssel herunter (zur Verwendung in der tokenbasierten Authentifizierungsmethode).
   - API-Schlüsselauthentifizierung — Wenn Sie lieber die ältere API-Schlüssel-Authentifizierungsmethode verwenden möchten, geben Sie den Google-API-Schlüssel in das dafür vorgesehene Feld ein. Um diese Datei zu erhalten:
     1. Gehen Sie zu Ihrer Firebase-Anwendungskonsole.
     2. Wählen Sie in den Projekteinstellungen Cloud Messaging aus.
     3. Kopieren Sie den Serverschlüssel (API-Schlüssel), den Sie für das Senden von Benachrichtigungen verwenden möchten.
5. Wenn Sie die gewünschten Änderungen vorgenommen haben, wählen Sie Save changes (Änderungen speichern) aus.

Verwandte Themen

- Verwenden von Google Firebase Cloud Messaging v1-Nutzlasten in Amazon SNS

Amazon SNS SNS-Verwaltung von Firebase Cloud Messaging-Endpunkten

Verwaltung und Pflege von Geräte-Token

Sie können die Zustellbarkeit der Push-Benachrichtigungen Ihrer mobilen Anwendung sicherstellen, indem Sie die folgenden Schritte ausführen:

1. Speichern Sie alle Geräte-Token, den entsprechenden Amazon SNS SNS-Endpunkt ARNs und die Zeitstempel auf Ihrem Anwendungsserver.
2. Entfernen Sie alle veralteten Token und löschen Sie den entsprechenden Amazon SNS SNS-Endpunkt. ARNs

Beim ersten Start Ihrer App erhalten Sie ein Geräte-Token (auch als Registrierungstoken bezeichnet) für das Gerät. Dieses Geräte-Token wird vom Betriebssystem des Geräts geprägt und ist an Ihre FCM-Anwendung gebunden. Sobald Sie dieses Geräte-Token erhalten haben, können Sie es bei Amazon SNS als Plattformendpunkt registrieren. Wir empfehlen Ihnen, das Geräte-Token, den ARN des Amazon SNS SNS-Plattformendpunkts und den Zeitstempel zu speichern, indem Sie sie auf Ihrem Anwendungsserver oder einem anderen persistenten Speicher speichern. Informationen zum Einrichten Ihrer FCM-Anwendung zum Abrufen und Speichern von Gerätetokens finden Sie in der Firebase-Dokumentation von Google unter Registrierungstoken abrufen und speichern.

Es ist wichtig, dass Sie Tokens verwalten up-to-date. Die Geräte-Token Ihres Benutzers können sich unter den folgenden Bedingungen ändern:

1. Die mobile Anwendung wird auf einem neuen Gerät wiederhergestellt.
2. Der Benutzer deinstalliert oder aktualisiert die Anwendung.
3. Der Benutzer löscht die Anwendungsdaten.

Wenn sich Ihr Geräte-Token ändert, empfehlen wir Ihnen, den entsprechenden Amazon SNS SNS-Endpunkt mit dem neuen Token zu aktualisieren. Dadurch kann Amazon SNS die Kommunikation mit dem registrierten Gerät fortsetzen. Sie können dies tun, indem Sie den folgenden Pseudocode in Ihrer mobilen Anwendung implementieren. Es beschreibt eine empfohlene Vorgehensweise für die Erstellung und Wartung aktivierter Plattformendpunkte. Dieser Ansatz kann bei jedem Start der mobilen Anwendungen oder als geplante Aufgabe im Hintergrund ausgeführt werden.

Pseudo-Code

Verwenden Sie den folgenden FCM-Pseudocode, um Gerätetokens zu verwalten und zu verwalten.

```
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
```

Weitere Informationen zu den Anforderungen für Token-Updates finden Sie unter Regelmäßiges Aktualisieren von Tokens in der Firebase-Dokumentation von Google.

Erkennung ungültiger Token

Wenn eine Nachricht mit einem ungültigen Geräte-Token an einen FCM v1-Endpunkt gesendet wird, erhält Amazon SNS eine der folgenden Ausnahmen:

- `UNREGISTERED`(HTTP 404) — Wenn Amazon SNS diese Ausnahme empfängt, erhalten Sie ein Ereignis mit einem Zustellungsfehler mit dem Wert `FailureType` von`InvalidPlatformToken`, und ein `FailureMessage` dem Endpunkt zugeordnetes Plattform-Token ist ungültig. Amazon SNS deaktiviert Ihren Plattformendpunkt, wenn eine Lieferung mit dieser Ausnahme fehlschlägt.
- `INVALID_ARGUMENT`(HTTP 400) — Wenn Amazon SNS diese Ausnahme empfängt, bedeutet dies, dass das Geräte-Token oder die Nachrichtennutzlast ungültig ist. Weitere Informationen finden Sie ErrorCodein der Firebase-Dokumentation von Google.

Da Amazon SNS in beiden Fällen zurückgesendet werden `INVALID_ARGUMENT` kann, gibt Amazon SNS ein of zurück`InvalidNotification`, und ein `FailureType` `FailureMessage` of der Benachrichtigungstext ist ungültig. Wenn Sie diesen Fehler erhalten, überprüfen Sie, ob Ihre

Payload korrekt ist. Wenn sie korrekt ist, überprüfen Sie, ob das Geräte-Token korrekt ist up-to-date. Amazon SNS deaktiviert Ihren Plattformendpunkt nicht, wenn eine Lieferung fehlschlägt, mit dieser Ausnahme.

Ein weiterer Fall, in dem ein `InvalidPlatformToken` Zustellungsfehler auftritt, ist, wenn das registrierte Geräte-Token nicht zu der Anwendung gehört, die versucht, diese Nachricht zu senden. In diesem Fall gibt Google einen SENDER_ID_MISMATCH-Fehler zurück. Amazon SNS deaktiviert Ihren Plattformendpunkt, wenn eine Lieferung mit dieser Ausnahme fehlschlägt.

Alle beobachteten Fehlercodes, die von der FCM v1-API empfangen wurden, stehen Ihnen zur Verfügung, CloudWatch wenn Sie die Versandstatusprotokollierung für Ihre Anwendung einrichten.

Informationen zum Empfang von Zustellungsereignissen für Ihre Anwendung finden Sie unterVerfügbare Anwendungsereignisse.

Veraltete Token entfernen

Token gelten als veraltet, sobald die Nachrichtenzustellung an das Endgerät fehlschlägt. Amazon SNS legt diese veralteten Token als deaktivierte Endpunkte für Ihre Plattformanwendung fest. Wenn Sie auf einem deaktivierten Endpunkt veröffentlichen, gibt Amazon SNS ein `EventDeliveryFailure` Ereignis mit dem Wert `FailureType` of zurück`EndpointDisabled`, und `FailureMessage` der Endpunkt ist deaktiviert. Informationen zum Empfang von Zustellungsereignissen für Ihre Anwendung finden Sie unterVerfügbare Anwendungsereignisse.

Wenn Sie diesen Fehler von Amazon SNS erhalten, müssen Sie das veraltete Token in Ihrer Plattformanwendung entfernen oder aktualisieren.

Amazon SNS für mobile Push-Benachrichtigungen verwenden

In diesem Abschnitt wird beschrieben, wie Sie mobile Push-Benachrichtigungen senden.

So veröffentlichen Sie in einem Thema

Sie können Amazon SNS auch verwenden, um Nachrichten an mobile Endpunkte zu senden, die ein Thema abonniert haben. Das Konzept ist das gleiche, als wenn andere Endpunkttypen, wie Amazon SQS, HTTP/S, E-Mail und SMS ein Thema abonnieren, wie unter Was ist Amazon SNS? beschrieben. Der Unterschied besteht darin, dass Amazon SNS über Benachrichtigungsdienste wie Apple Push Notification Service (APNS) und Google Firebase Cloud Messaging (FCM) kommuniziert. Durch den Benachrichtigungsservice erhalten die abonnierten mobilen Endpunkte Benachrichtigungen, die an das Thema gesendet wurden.

Direkte Amazon SNS SNS-Nachrichtenübermittlung auf Mobilgeräten

Senden Sie Amazon SNS-Push-Benachrichtigungen direkt an einen Endpunkt, der eine App auf einem mobilen Gerät repräsentiert.

So senden Sie eine direkte Nachricht

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie auf der Seite Push-Benachrichtigungen für Mobilgeräte im Abschnitt Plattformanwendungen beispielsweise ***MyApp*** den Namen der Anwendung aus.
4. Wählen Sie auf der ***MyApp*** Seite im Abschnitt Endpoints einen Endpoint aus und klicken Sie dann auf Nachricht veröffentlichen.
5. Geben Sie auf der Seite Publish message to endpoint (Nachricht am Endpunkt veröffentlichen) die Nachricht ein, die in der Anwendung auf dem mobilen Gerät erscheinen wird, und wählen Sie dann Publish message (Nachricht veröffentlichen).

   Amazon SNS sendet die Benachrichtigung an den Plattformbenachrichtigungsservice, der sie dann wiederum an die Anwendung sendet.

Veröffentlichen von Amazon SNS SNS-Benachrichtigungen mit plattformspezifischen Payloads

Sie können das AWS Management Console oder Amazon SNS verwenden APIs , um benutzerdefinierte Nachrichten mit plattformspezifischen Nutzlasten an mobile Geräte zu senden. Informationen zur Verwendung von Amazon SNS finden Sie APIs unter Mobile Push-API-Aktionen und die `SNSMobilePush.java` Datei in`snsmobilepush.zip`.

Senden von Nachrichten im JSON-Format

Wenn Sie plattformspezifische Nutzlasten senden, müssen die Daten als Schlüssel-Wert-Paar-Zeichenfolgen im JSON-Format vorliegen und mit Anführungszeichen maskiert sein.

Die folgenden Beispiele zeigen eine benutzerdefinierte Nachricht für die FCM-Plattform.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
```

```
}
```

Senden plattformspezifischer Nachrichten

Sie können nicht nur benutzerdefinierte Daten als Schlüssel-Wert-Paare, sondern auch plattformspezifische Schlüssel-Wert-Paare senden.

Das folgende Beispiel zeigt den Einschluss der FCM-Parameter `time_to_live` und `collapse_key` nach den Schlüsselwertpaaren der benutzerdefinierten Daten im FCM `data`-Parameter .

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Eine Liste der von jedem der Push-Benachrichtigungsservices in Amazon SNS unterstützten Schlüssel-Wert-Paare finden Sie unter:

> **Important**
>
> Amazon SNS unterstützt jetzt die Firebase Cloud Messaging (FCM) HTTP v1-API zum Senden von mobilen Push-Benachrichtigungen an Android-Geräte.
> 26. März 2024 — Amazon SNS unterstützt die FCM-HTTP-v1-API für Apple-Geräte und Webpush-Ziele. Wir empfehlen Ihnen, Ihre vorhandenen mobilen Push-Anwendungen am oder vor dem 1. Juni 2024 auf die neueste FCM-HTTP-v1-API zu migrieren, um Anwendungsunterbrechungen zu vermeiden.

- Payload Key-Referenz in der Dokumentation APNs
- Firebase Cloud Messaging-HTTP-Protokoll in der FCM-Dokumentation
- Senden einer Nachricht in der ADM-Dokumentation

Senden von Nachrichten an eine Anwendung auf mehreren Plattformen

Um eine Nachricht an eine Anwendung zu senden, die auf Geräten für mehrere Plattformen installiert ist, z. B. FCM und APNs, müssen Sie zuerst die mobilen Endgeräte für ein Thema in Amazon SNS abonnieren und dann die Nachricht zum Thema veröffentlichen.

Das folgende Beispiel zeigt eine Nachricht, die an abonnierte mobile Endpunkte auf FCM und ADM gesendet werden soll APNs:

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

Senden von Nachrichten als Warn- oder APNs Hintergrundbenachrichtigung

Amazon SNS kann Nachrichten `alert` oder `background` Benachrichtigungen an APNs Werbeanzeigen senden (weitere Informationen finden Sie in der APNs Dokumentation unter Pushing Background Updates to Your App).

- Eine `alert` APNs Benachrichtigung informiert den Benutzer, indem er eine Warnmeldung anzeigt, einen Ton abspielt oder dem Symbol Ihrer Anwendung ein Badge hinzufügt.
- Eine `background` APNs Benachrichtigung wird aktiviert oder Ihre Anwendung angewiesen, auf den Inhalt der Benachrichtigung zu reagieren, ohne den Benutzer zu informieren.

Angeben von benutzerdefinierten Header-Werten APNs

Wir empfehlen, benutzerdefinierte Werte für das `AWS.SNS.MOBILE.APNS.PUSH_TYPE` reservierte Nachrichtenattribut mithilfe der Amazon SNS `Publish` SNS-API-Aktion oder der AWS CLI anzugeben. AWS SDKs Im folgenden CLI-Beispiel wird für das angegebene Thema für `content-available` 1 und für `apns-push-type` `background` festgelegt.

```
aws sns publish \
--endpoint-url https://sns.us-east-1.amazonaws.com \
--target-arn arn:aws:sns:us-east-1:123456789012:endpoint/APNS_PLATFORM/MYAPP/1234a567-
bc89-012d-3e45-6fg7h890123i \
--message '{"APNS_PLATFORM":"{\"aps\":{\"content-available\":1}}"}' \
--message-attributes '{ \
```

```
  "AWS.SNS.MOBILE.APNS.TOPIC":
{"DataType":"String","StringValue":"com.amazon.mobile.messaging.myapp"}, \
  "AWS.SNS.MOBILE.APNS.PUSH_TYPE":{"DataType":"String","StringValue":"background"}, \
  "AWS.SNS.MOBILE.APNS.PRIORITY":{"DataType":"String","StringValue":"5"}}' \
--message-structure json
```

> **Note**
>
> Stellen Sie sicher, dass die JSON-Struktur gültig ist. Fügen Sie nach jedem Schlüssel-Wert-Paar, außer dem letzten, ein Komma hinzu.

Ableiten des APNs Push-Typ-Headers aus der Nutzlast

Wenn Sie den `apns-push-type` APNs Header nicht festlegen, setzt Amazon SNS den Header auf `alert` oder `background` abhängig vom `content-available` Schlüssel im `aps` Wörterbuch Ihrer JSON-formatierten APNs Payload-Konfiguration.

> **Note**
>
> Amazon SNS kann nur `alert`- oder `background`-Header ableiten, auch wenn der Header `apns-push-type` auf andere Werte gesetzt werden kann.

- `apns-push-type` wird auf `alert` gesetzt
  - Wenn das `aps`-Verzeichnis einen auf 1 gesetzten `content-available`-Schlüssel und einen oder mehrere Schlüssel enthält, die Benutzerinteraktionen auslösen
  - Wenn das `aps`-Verzeichnis einen auf `0` gesetzten `content-available`-Schlüssel enthält oder wenn der `content-available`-Schlüssel nicht vorhanden ist
  - Wenn der Wert des `content-available`-Schlüssels keine Ganzzahl und kein boolescher Wert ist
- `apns-push-type` wird auf `background` gesetzt
  - Wenn das `aps`-Verzeichnis nur auf 1 gesetzte `content-available` und keine anderen Schlüssel enthält, die Benutzerinteraktionen auslösen.

> **Important**
>
> Wenn Amazon SNS ein unformatiertes Konfigurationsobjekt für APNs als reine Hintergrundbenachrichtigung sendet, müssen Sie `content-available` set to 1 in das Wörterbuch aufnehmen. `aps` Auch wenn benutzerdefinierte Schlüssel eingeschlossen werden können, darf das `aps`-Verzeichnis keine Schlüssel enthalten, die Benutzerinteraktionen (z. B. Warnmeldungen, Abzeichen oder Audiosignale) auslösen.

Es folgt ein Beispiel für ein unformatiertes Konfigurationsobjekt.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

In diesem Beispiel legt Amazon SNS den `apns-push-type` APNs Header für die Nachricht auf `background` fest. Wenn Amazon SNS erkennt, dass das `apn`-Verzeichnis den `content-available` Schlüssel festgelegt auf `1` enthält, und keine anderen Schlüssel enthält, die Benutzerinteraktionen auslösen können, legt es den Header auf `background`.

Verwenden von Google Firebase Cloud Messaging v1-Nutzlasten in Amazon SNS

Amazon SNS unterstützt die Verwendung der FCM-HTTP-v1-API zum Senden von Benachrichtigungen an Android-, iOS- und Webpush-Ziele. Dieses Thema enthält Beispiele für die Payload-Struktur beim Veröffentlichen von mobilen Push-Benachrichtigungen über die CLI oder die Amazon SNS SNS-API.

Sie können die folgenden Nachrichtentypen in Ihre Payload aufnehmen, wenn Sie eine FCM-Benachrichtigung senden:

- Datennachricht — Eine Datennachricht wird von Ihrer Client-App verarbeitet und enthält benutzerdefinierte Schlüssel-Wert-Paare. Wenn Sie eine Datennachricht erstellen, müssen Sie den `data` Schlüssel mit einem JSON-Objekt als Wert angeben und dann Ihre benutzerdefinierten Schlüssel-Wert-Paare eingeben.
- Benachrichtigungsnachricht oder Anzeigenachricht — Eine Benachrichtigung enthält einen vordefinierten Satz von Schlüsseln, die vom FCM SDK verarbeitet werden. Diese Schlüssel variieren je nach Gerätetyp, an den Sie liefern. Weitere Informationen zu plattformspezifischen Benachrichtigungsschlüsseln finden Sie im Folgenden:

- Android-Benachrichtigungstasten
- APNS-Benachrichtigungsschlüssel
- Schlüssel für Webpush-Benachrichtigungen

Weitere Informationen zu FCM-Nachrichtentypen finden Sie unter Nachrichtentypen in der Firebase-Dokumentation von Google.

Verwenden der FCM v1-Payload-Struktur zum Senden von Nachrichten

Wenn Sie zum ersten Mal eine FCM-Anwendung erstellen oder die Funktionen von FCM v1 nutzen möchten, können Sie sich dafür entscheiden, eine FCM v1-formatierte Payload zu senden. Dazu müssen Sie den Schlüssel der obersten Ebene angeben. `fcmV1Message` Weitere Informationen zum Erstellen von FCM v1-Payloads finden Sie unter Migration von Legacy-FCM APIs zu HTTP v1 und Plattformübergreifendes Anpassen einer Nachricht in der Firebase-Dokumentation von Google.

FCM v1-Beispielnutzlast, die an Amazon SNS gesendet wurde:

> **Note**
>
> Der im folgenden Beispiel verwendete `GCM` Schlüsselwert muss als Zeichenfolge kodiert werden, wenn eine Benachrichtigung mit Amazon SNS veröffentlicht wird.

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

```
}
```

Achten Sie beim Senden einer JSON-Nutzlast darauf, das `message-structure` Attribut in Ihre Anfrage aufzunehmen und es auf zu setzen. `json`

CLI-Beispiel:

```
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
```

Weitere Informationen zum Senden von Payloads im FCM v1-Format finden Sie in der Firebase-Dokumentation von Google:

- Migrieren Sie von Legacy-FCM zu HTTP v1 APIs
- Über FCM-Nachrichten
- REST-Ressource: projects.messages

Verwenden der veralteten Payload-Struktur zum Senden von Nachrichten an die FCM v1-API

Bei der Migration zu FCM v1 müssen Sie die Payload-Struktur, die Sie für Ihre älteren Anmeldeinformationen verwendet haben, nicht ändern. Amazon SNS wandelt Ihre Payload in die neue FCM v1-Payload-Struktur um und sendet sie an Google.

Payload-Format der Eingabe-Nachricht:

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

Nachricht an Google gesendet:

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

Potenzielle Risiken

- Die Zuordnung von Legacy zu Version 1 unterstützt weder den Apple Push Notification Service (APNS) `headers` noch die `fcm_options` Schlüssel. Wenn Sie diese Felder verwenden möchten, senden Sie eine FCM v1-Payload.
- In einigen Fällen benötigt FCM v1 Nachrichtenkopfzeilen, um stille Benachrichtigungen an Ihre Geräte zu senden. APNs Wenn Sie derzeit stille Benachrichtigungen an Ihre APNs Geräte senden, funktionieren diese mit dem älteren Ansatz nicht. Stattdessen empfehlen wir, die FCM v1-Payload zu verwenden, um unerwartete Probleme zu vermeiden. Eine Liste der APNs Header und wofür sie verwendet werden, findest du unter Kommunizieren mit APNs im Apple Developer Guide.
- Wenn Sie beim Senden Ihrer Benachrichtigung das `TTL` Amazon SNS SNS-Attribut verwenden, wird es nur im `android` Feld aktualisiert. Wenn Sie das `TTL` APNS-Attribut festlegen möchten, verwenden Sie die FCM v1-Payload.

- Die `webpush` Schlüssel `androidapns`, und werden zugeordnet und mit allen bereitgestellten relevanten Schlüsseln gefüllt. Wenn Sie beispielsweise angeben, was ein Schlüssel ist`title`, der von allen drei Plattformen gemeinsam genutzt wird, füllt die FCM-v1-Zuordnung alle drei Plattformen mit dem von Ihnen angegebenen Titel auf.
- Bei einigen von Plattformen gemeinsam genutzten Schlüsseln werden unterschiedliche Werttypen erwartet. Beispielsweise `apns` erwartet der `badge` Schlüssel, an den übergeben wurde, einen Integer-Wert, während der `badge` Schlüssel, an den übergeben wurde, einen String-Wert `webpush` erwartet. In Fällen, in denen Sie den `badge` Schlüssel angeben, füllt die FCM v1-Zuordnung nur den Schlüssel auf, für den Sie einen gültigen Wert angegeben haben.

Ereignisse, die bei der FCM-Zustellung fehlschlagen

Die folgende Tabelle enthält den Amazon SNS SNS-Fehlertyp, der den Fehler-/Statuscodes entspricht, die von Google für FCM v1-Benachrichtigungsanfragen erhalten wurden. Alle beobachteten Fehlercodes, die von der FCM v1-API empfangen wurden, stehen Ihnen zur Verfügung, CloudWatch wenn Sie die Versandstatusprotokollierung für Ihre Anwendung einrichten.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`UNREGISTERED`	`InvalidPlatformToken`	Das dem Endpunkt zugeordnete Plattform token ist nicht gültig.	Das an Ihren Endpunkt angehängte Geräte-Token ist veraltet oder ungültig. Amazon SNS hat Ihren Endpunkt deaktiviert. Aktualisieren Sie den Amazon SNS SNS-Endpunkt auf das neueste Geräte-Token.
`INVALID_ARGUMENT`	`InvalidNotification`	Der Text der Benachrichtigung ist ungültig.	Das Geräte-Token oder die Nachrichten-Payload sind möglicherweise ungültig. Stellen Sie

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
			sicher, dass Ihre Nachrichten-Payload gültig ist. Wenn die Nachrichten-Payload gültig ist, aktualisieren Sie den Amazon SNS SNS-Endpunkt auf das neueste Geräte-Token.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	Das dem Endpunkt zugeordnete Plattform-Token ist nicht gültig.	Die mit dem Geräte-Token verknüpfte Plattformanwendung ist nicht berechtigt, an das Geräte-Token zu senden. Stellen Sie sicher, dass Sie die richtigen FCM-Anmeldeinformationen in Ihrer Amazon SNS-Plattformanwendung verwenden.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`UNAVAILABLE`	`DependencyUnavailable`	Die Abhängigkeit ist nicht verfügbar.	FCM konnte die Anfrage nicht rechtzeitig bearbeiten. Alle von Amazon SNS ausgeführten Wiederholungen sind fehlgeschlagen. Sie können diese Nachrichten in einer Warteschlange (Dead-Letter Queue, DLQ) speichern und sie zu einem späteren Zeitpunkt erneut versenden.
`INTERNAL`	`UnexpectedFailure`	Unerwarteter Fehler. Bitte wenden Sie sich an Amazon. Fehlerphrase [Interner Fehler].	Beim Versuch, Ihre Anfrage zu bearbeiten, ist auf dem FCM-Server ein Fehler aufgetreten. Alle von Amazon SNS ausgeführten Wiederholungen sind fehlgeschlagen. Sie können diese Nachrichten in einer Warteschlange (Dead-Letter Queue, DLQ) speichern und sie zu einem späteren Zeitpunkt erneut versenden.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	Die Anmeldeinformationen für die Plattformanwendung sind nicht gültig.	Eine Nachricht, die an ein iOS-Gerät oder ein Webpush-Gerät gerichtet war, konnte nicht gesendet werden. Stellen Sie sicher, dass Ihre Entwicklungs- und Produktionsanmeldedaten gültig sind.
`QUOTA_EXCEEDED`	`Throttled`	Die Anfrage wurde von [gcm] gedrosselt.	Ein Nachrichtenratenkontingent, ein Gerätenachrichtenratenkontingent oder ein Themen-Nachrichtenratenkontingent wurde überschritten. Informationen zur Behebung dieses Problems finden Sie ErrorCodein der Firebase-Dokumentation von Google.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`PERMISSION_DENIED`	`InvalidNotification`	Der Text der Benachrichtigung ist ungültig.	Im `PERMISSION_DENIED` Ausnahmefall ist der Aufrufer (Ihre FCM-Anwendung) nicht berechtigt, den angegebenen Vorgang in der Nutzlast auszuführen. Navigieren Sie zu Ihrer FCM-Konsole und vergewissern Sie sich, dass in Ihren Anmeldeinformationen die erforderlichen API-Aktionen aktiviert sind.

Attribute der mobilen Amazon SNS SNS-App

Amazon Simple Notification Service (Amazon SNS) unterstützt die Protokollierung des Zustellungsstatus von Push-Benachrichtigungen. Nachdem Sie Anwendungsattribute konfiguriert haben, werden Protokolleinträge für Nachrichten, die von Amazon SNS an mobile Endpunkte gesendet wurden, an CloudWatch Logs gesendet. Die Protokollierung des Zustellungsstatus von Nachrichten verhilft Ihnen zu besseren betrieblichen Einblicken, wie beispielsweise den Folgenden:

- Sie erfahren, ob eine Push-Benachrichtigungsmitteilung von Amazon SNS an den Push-Benachrichtigungsservice zugestellt wurde.
- Identifizieren Sie die Antwort, die vom Push-Benachrichtigungsservice an den Amazon SNS gesendet wurde.
- Ermitteln Sie die dwell-Zeit der Nachricht (das Zeitfenster zwischen dem Zeitstempel der Veröffentlichung bis direkt vor der Übergabe an einen Push-Benachrichtigungsservice).

Um Anwendungsattribute für den Nachrichtenzustellungsstatus zu konfigurieren, können Sie die AWS Management Console AWS Software Development Kits (SDKs) oder die Abfrage-API verwenden.

Konfiguration von Nachrichtenzustellungsstatusattributen mit dem AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Mobile (Mobil) und dann Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie im Abschnitt Plattformanwendungen die Anwendung aus, die die Endpoints enthält, für die Sie CloudWatch Protokolle erhalten möchten.
4. Wählen Sie Application Actions (Anwendungsaktionen) und anschließend Delivery status (Zustellungsstatus).
5. Wählen Sie im Dialogfeld Delivery Status (Zustellungsstatus)die Option Create IAM Roles (IAM-Rollen erstellen).

   Sie werden anschließend zur IAM-Konsole umgeleitet.
6. Wählen Sie Zulassen, um Amazon SNS Schreibzugriff auf die Nutzung von CloudWatch Logs in Ihrem Namen zu gewähren.
7. Kehren Sie nun zum Dialogfeld „Lieferstatus“ zurück und geben Sie im Feld Erfolgsquote bis zur Stichprobe (0-100) eine Zahl für den Prozentsatz der erfolgreich gesendeten Nachrichten ein, für die Sie Logs erhalten CloudWatch möchten.

   > **Note**
   >
   > Nachdem Sie die Anwendungsattribute für den Nachrichtenzustellungsstatus konfiguriert haben, werden für alle fehlgeschlagenen Nachrichtenzustellungen CloudWatch Protokolle generiert.
8. Wählen Sie anschließend Save Configuration (Konfiguration speichern). Sie können jetzt die CloudWatch Protokolle mit dem Status der Nachrichtenzustellung anzeigen und analysieren. Weitere Informationen zur Verwendung CloudWatch finden Sie in der CloudWatchDokumentation.

Beispiele für das CloudWatch Protokoll zum Status der Nachrichtenzustellung in Amazon SNS

Nachdem Sie die Attribute für den Nachrichtenzustellungsstatus für einen Anwendungsendpunkt konfiguriert haben, werden CloudWatch Protokolle generiert. Beispielprotokolle im JSON-Format werden wie folgt angezeigt:

ERFOLG

```
{
  "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"
  }
}
```

FEHLER

```
{
  "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"
  }
}
```

Eine Liste der Antwortcodes für den Push-Benachrichtigungsservice finden Sie unter Antwortcodes der Plattformen.

Konfiguration von Statusattributen für die Nachrichtenzustellung mit dem AWS SDKs

AWS SDKsSie bieten APIs in mehreren Sprachen die Verwendung von Nachrichtenzustellungsstatusattributen mit Amazon SNS an.

Das folgende Java-Beispiel zeigt, wie Sie mit der `SetPlatformApplicationAttributes`-API Anwendungsattribute für den Status von Nachrichtenübermittlungen von Push-Benachrichtigungsmitteilungen konfigurieren. Sie können die folgenden Attribute für den Status von Nachrichtenübermittlungen verwenden: `SuccessFeedbackRoleArnFailureFeedbackRoleArn` und `SuccessFeedbackSampleRate`. Die `FailureFeedbackRoleArn` Attribute `SuccessFeedbackRoleArn` und werden verwendet, um Amazon SNS Schreibzugriff auf die Nutzung von CloudWatch Logs in Ihrem Namen zu gewähren. Das Attribut `SuccessFeedbackSampleRate` dient zum Festlegen des Prozentsatzes der Samplerate (0-100) der erfolgreich zugestellten Nachrichten. Nachdem Sie das `FailureFeedbackRoleArn` Attribut konfiguriert haben, werden bei allen fehlgeschlagenen Nachrichtenzustellungen CloudWatch Protokolle generiert.

```
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);
```

Weitere Informationen zu SDK für Java finden Sie unter Erste Schritte mit dem AWS SDK für Java.

Antwortcodes der Plattformen

Die folgende Liste enthält Links für die Antwortcodes des Push-Benachrichtigungsservice:

Push-Benachrichtigungsservice	Antwortcodes
Amazon Device Messaging (ADM)	Weitere Informationen finden Sie unter Antwortformat in der ADM Dokumentation.
Apple-Push-Benachrichtigungsdienst () APNs	Weitere Informationen finden Sie unter HTTP/2-Antwort von APNs in Kommunizieren mit APNs im Programmierhandbuch für lokale und ferngesteuerte Benachrichtigungen.
Firebase Cloud Messaging (FCM)	Weitere Informationen finden Sie unter Downstream Message Error Response Codes in der Firebase Cloud Messaging Dokumenta tion.
Microsoft Push Notification Service für Windows Phone (MPNS)	Weitere Informationen finden Sie unter Push Notification Service Response Codes for Windows Phone 8 in der Windows 8 Development Dokumentation.
Windows Push Notification Services (WNS)	Weitere Informationen finden Sie unter "Response codes" in Push Notification Service Request and Response Headers (Windows Runtime Apps) in der Windows 8 Development-Dokumentation.

Benachrichtigungen über Amazon SNS SNS-Anwendungsereignisse für mobile Anwendungen

Amazon SNS löst bei Auftreten bestimmter Anwendungsereignisse Benachrichtigungen aus. Im Anschluss daran können Sie eine programmgesteuerte Aktion für das Ereignis definieren. Ihre Anwendung muss Unterstützung für einen Push-Benachrichtigungsdienst wie Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) und Windows Push Notification Services (WNS)

enthalten. Sie richten Benachrichtigungen über Anwendungsereignisse mithilfe der Amazon SNS SNS-Konsole oder der AWS SDKs ein. AWS CLI

Verfügbare Anwendungsereignisse

Ereignisbenachrichtigungen einer Anwendung können nachverfolgen, wann Endpunkte für individuelle Plattformen erstellt, gelöscht oder aktualisiert wurden. Sie können außerdem Versandfehler nachverfolgen. Im Folgenden sind die Attributnamen für die Anwendungsereignisse aufgeführt.

Attributname	Benachrichtigungsauslöser
`EventEndpointCreated`	Ein neuer Plattformendpunkt wird Ihrer Anwendung hinzugefügt.
`EventEndpointDeleted`	Alle Plattformendpunkte, die Ihrer Anwendung zugewiesen sind, werden gelöscht.
`EventEndpointUpdated`	Alle Attribute der Plattformendpunkte, die Ihrer Anwendung zugewiesen sind, werden geändert.
`EventDeliveryFailure`	Bei einer Übermittlung an einen der Plattformendpunkte, die Ihrer Anwendung zugewiesen wurden, ist ein dauerhafter Fehler aufgetreten. **Note** Um von der Plattform-Anwendung ausgehende Fehler bei der Übermittlung zu verfolgen, abonnieren Sie für die Anwendung Statusereignisse für die Nachrichtenübermittlung. Weitere Informationen finden Sie unter Verwendung von Amazon SNS-Anwendungsattributen für den Status von Nachrichtenübermittlungen.

Sie können jedes Attribut einer Anwendung zuordnen, die dann diese Ereignisbenachrichtigungen empfangen kann.

Senden mobiler Push-Benachrichtigungen

Um Ereignisbenachrichtigungen für Anwendungen zu versenden, geben Sie ein Thema an, das die Benachrichtigungen für den jeweiligen Ereignistyp erhalten soll. Sobald Amazon SNS die Benachrichtigungen sendet, kann das Themas diese an Endpunkte weiterleiten, die eine programmgesteuerte Aktion durchführen werden.

> **Important**
>
> Anwendungen mit großem Volumen rufen häufige Ereignisbenachrichtigungen für Anwendungen hervor (z. B. Zehntausende), durch die Endpunkte, die für die Verwendung durch Menschen vorgesehen sind, überflutet werden (z. B. E-Mail-Adressen, Telefonnummern und mobile Anwendungen). Beachten Sie folgende Richtlinien, wenn Sie Ereignisbenachrichtigungen für Anwendungen an ein Thema senden:
>
> - Jedes Thema, das Benachrichtigungen erhält, sollte nur Abonnements für programmatische Endpunkte wie HTTP- oder HTTPS-Endpunkte, Amazon SQS SQS-Warteschlangen oder Funktionen enthalten. AWS Lambda
> - Um den Bearbeitungsaufwand, der durch Benachrichtigungen ausgelöst wird, zu minimieren, reduzieren Sie die Anzahl der Abonnements eines jeden Themas (z. B. auf fünf oder weniger).

Sie können Benachrichtigungen über Anwendungsereignisse über die Amazon SNS SNS-Konsole, die AWS Command Line Interface (AWS CLI) oder die AWS SDKs senden.

AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Mobile (Mobil) und dann Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie auf der Seite Mobile Push-Benachrichtigungen im Abschnitt Plattformanwendungen eine Anwendung aus und klicken Sie dann auf Bearbeiten.
4. Erweitern Sie den Abschnitt Event notifications (Ereignisbenachrichtigungen).
5. Wählen Sie Actions (Aktionen) und Configure events (Ereignisse konfigurieren) aus.
6. Geben Sie die ARNs vier Themen ein, die für die folgenden Ereignisse verwendet werden sollen:
   - Ersteller Endpunkt

- Gelöschter Endpunkt
- Aktualisierter Endpunkt
- Übermittlungsfehler

7. Wählen Sie Änderungen speichern.

AWS CLI

Führen Sie den Befehl set-platform-application-attributes aus.

Im folgenden Beispiel wird das gleiche Amazon SNS-Thema für alle vier Anwendungsereignisse definiert:

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDKs

Richten Sie Benachrichtigungen über Anwendungsereignisse ein, indem Sie mithilfe eines AWS SDK eine `SetPlatformApplicationAttributes` Anfrage mit der Amazon SNS SNS-API einreichen.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele, einschließlich Hilfe bei den ersten Schritten und Informationen zu früheren Versionen, finden Sie unterAmazon SNS mit einem AWS SDK verwenden.

Mobile Push-API-Aktionen

Um Amazon SNS Mobile Push verwenden zu können APIs, müssen Sie zunächst die Voraussetzungen für den Push-Benachrichtigungsdienst wie Apple Push Notification Service (APNs) und Firebase Cloud Messaging (FCM) erfüllen. Weitere Informationen zu diesen Voraussetzungen finden Sie unter Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen.

Um mithilfe von eine Push-Benachrichtigung an eine mobile App und ein Gerät zu senden APIs, müssen Sie zunächst die `CreatePlatformApplication` Aktion verwenden, die ein Attribut zurückgibt. `PlatformApplicationArn` Das `PlatformApplicationArn`-Attribut wird dann von `CreatePlatformEndpoint` verwendet. So wird ein `EndpointArn`-Attribut zurückgeben. Anschließend können Sie mit dem `EndpointArn`-Attribut mit der `Publish`-Aktion eine Benachrichtigung an eine mobile App und ein Gerät senden oder Sie können das `EndpointArn`-Attribut mit der `Subscribe`-Aktion zum Abonnieren eines Themas verwenden. Weitere Informationen finden Sie unter Push-Benachrichtigungen mit Amazon SNS einrichten.

Die mobilen Amazon SNS SNS-Pushs APIs lauten wie folgt:

`CreatePlatformApplication`

Erstellt ein Plattformanwendungsobjekt für einen der unterstützten Push-Benachrichtigungsdienste wie APNs FCM, bei dem sich Geräte und mobile Apps registrieren können. Gibt ein `PlatformApplicationArn`-Attribut zurück, das für die `CreatePlatformEndpoint`-Aktion verwendet wird.

`CreatePlatformEndpoint`

Erstellt einen Endpunkt für ein Gerät und eine mobile App für einen der unterstützten Push-Benachrichtigungsservices. `CreatePlatformEndpoint` verwendet das `PlatformApplicationArn`-Attribut, das der `CreatePlatformApplication`-Aktion zurückgegeben wurde. Das `EndpointArn`-Attribut, das bei der Verwendung von `CreatePlatformEndpoint` zurückgegeben wird, wird mit der `Publish`-Aktion verwendet, um eine Benachrichtigung an die mobile App und das Gerät zu senden.

`CreateTopic`

Erstellt ein Thema, über das die Nachrichten veröffentlicht werden können.

`DeleteEndpoint`

Löscht den Endpunkt für ein Gerät und eine mobile App für einen der unterstützten Push-Benachrichtigungsservices.

`DeletePlatformApplication`

Löscht ein Plattformanwendungsobjekt.

`DeleteTopic`

Löscht ein Thema und alle seine Abonnements.

`GetEndpointAttributes`

Ruft die Endpunktattribute für ein Gerät und eine mobile App ab.

`GetPlatformApplicationAttributes`

Ruft die Attribute des Plattformanwendungsobjekts ab.

`ListEndpointsByPlatformApplication`

Listet die Endpunkte und Endpunktattribute für Geräte und mobile Apps in einem unterstützten Push-Benachrichtigungsservice auf.

`ListPlatformApplications`

Listet die Plattformanwendungsobjekte für die unterstützten Push-Benachrichtigungsservices auf.

`Publish`

Sendet eine Benachrichtigung an alle abonnierten Endpunkte des Themas.

`SetEndpointAttributes`

Legt die Attribute für einen Endpunkt für ein Gerät und eine mobile App fest.

`SetPlatformApplicationAttributes`

Legt die Attribute des Plattformanwendungsobjekts fest.

`Subscribe`

Bereitet über das Senden einer Bestätigungsnachricht an den Endpunkt das Abonnieren eines Endpunkts vor. Um tatsächlich ein Abonnement zu erstellen, muss der Endpunktbesitzer die ConfirmSubscription Aktion mit dem Token aus der Bestätigungsnachricht aufrufen.

`Unsubscribe`

Löscht ein Abonnement.

Häufige Amazon SNS SNS-Push-API-Fehler für Mobilgeräte

Fehler, die von Amazon SNS APIs for Mobile Push zurückgegeben werden, sind in der folgenden Tabelle aufgeführt. Weitere Informationen über Amazon SNS APIs for Mobile Push finden Sie unterMobile Push-API-Aktionen.

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
Anwendungsname ist eine Null-Zeichenfolge	Der erforderliche Anwendungsname ist auf Null festgelegt.	400	`CreatePlatformApplication`
Plattformname ist eine Null-Zeichenfolge	Der erforderliche Plattformname ist auf Null festgelegt.	400	`CreatePlatformApplication`
Plattformname ist ungültig	Für den Plattform namen wurde ein ungültiger out-of-range Wert oder angegeben.	400	`CreatePlatformApplication`
APNs — Principal ist kein gültiges Zertifikat	Für den APNs Prinzipal, das SSL-Zertifikat, wurde ein ungültiges Zertifikat bereitgestellt. Weitere Informationen finden Sie CreatePlatformApplicationin der Amazon Simple Notification Service API-Referenz.	400	`CreatePlatformApplication`
APNs — Principal ist ein gültiges Zertifikat, aber nicht im PEM-Format	Für den APNs Prinzipal, das SSL-Zertifikat, wurde ein gültiges Zertifikat bereitgestellt, das nicht im PEM-Format vorliegt.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs — Principal ist ein abgelaufenes Zertifikat	Für den APNs Prinzipal, das SSL-Zertifikat, wurde ein abgelaufenes Zertifikat bereitgestellt.	400	`CreatePlatformApplication`
APNs — Principal ist kein von Apple ausgestelltes Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein nicht von Apple ausgestelltes Zertifikat angegeben.	400	`CreatePlatformApplication`
APNs — Der Schulleiter ist nicht angegeben	Der APNs Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde nicht bereitgestellt.	400	`CreatePlatformApplication`
APNs — Der Berechtigungsnachweis wurde nicht bereitgestellt	Die APNs Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, wurden nicht bereitgestellt. Weitere Informationen finden Sie CreatePlatformApplicationin der Amazon Simple Notification Service API-Referenz.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs — Die Anmeldeinformationen haben kein gültiges PEM-Format	Die APNs Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, haben kein gültiges PEM-Format.	400	`CreatePlatformApplication`
FCM — Server wird nicht bereitgestellt APIKey	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, wurden nicht angegeben. Weitere Informationen finden Sie CreatePlatformApplicationin der Amazon Simple Notification Service API-Referenz.	400	`CreatePlatformApplication`
FCM — Server APIKey ist leer	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind leer.	400	`CreatePlatformApplication`
FCM — Server APIKey ist eine Nullzeichenfolge	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind Null.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
FCM — Server APIKey ist ungültig	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind ungültig.	400	`CreatePlatformApplication`
ADM – clientsecret wurde nicht angegeben	Der erforderliche Clientschlüssel liegt nicht vor.	400	`CreatePlatformApplication`
ADM – clientsecret ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für den Clientschlüssel ist Null.	400	`CreatePlatformApplication`
ADM – client_secret ist eine leere Zeichenfolge	Die erforderliche Zeichenfolge für den Clientschlüssel ist leer.	400	`CreatePlatformApplication`
ADM – client_secre ist nicht gültig	Die erforderliche Zeichenfolge für den Clientschlüssel ist nicht gültig.	400	`CreatePlatformApplication`
ADM – client_id ist eine leere Zeichenfolge	Die erforderliche Zeichenfolge für die Client-ID ist leer.	400	`CreatePlatformApplication`
ADM – clientId wurde nicht angegeben	Die erforderliche Zeichenfolge für die Client-ID liegt nicht vor.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
ADM – clientid ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für die Client-ID ist Null.	400	`CreatePlatformApplication`
ADM – client_id ist nicht gültig	Die erforderliche Zeichenfolge für die Client-ID ist nicht gültig.	400	`CreatePlatformApplication`
EventEndpointCreated hat ein ungültiges ARN-Format	EventEndpointCreated hat ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventEndpointDeleted hat ein ungültiges ARN-Format	EventEndpointDeleted hat ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventEndpointUpdated hat ein ungültiges ARN-Format	EventEndpointUpdated hat ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure hat ein ungültiges ARN-Format	EventDeliveryAttemptFailure hat ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventDeliveryFailure hat ein ungültiges ARN-Format	EventDeliveryFailure hat ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventEndpointCreated ist kein vorhandenes Thema	EventEndpointCreated ist kein vorhandenes Thema.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
EventEndpointDeleted ist kein vorhandenes Thema	EventEndpointDeleted ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
EventEndpointUpdated ist kein vorhandenes Thema	EventEndpointUpdated ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure ist kein vorhandenes Thema	EventDeliveryAttemptFailure ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
EventDeliveryFailure ist kein vorhandenes Thema	EventDeliveryFailure ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
ARN der Plattform ist ungültig	ARN der Plattform ist ungültig.	400	`SetPlatformAttributes`
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	400	`SetPlatformAttributes`
APNs — Principal ist kein gültiges Zertifikat	Für den APNs Prinzipal, das SSL-Zertifikat, wurde ein ungültiges Zertifikat bereitgestellt. Weitere Informationen finden Sie CreatePlatformApplicationin der Amazon Simple Notification Service API-Referenz.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs — Principal ist ein gültiges Zertifikat, aber nicht im PEM-Format	Für den APNs Prinzipal, das SSL-Zertifikat, wurde ein gültiges Zertifikat bereitgestellt, das nicht im PEM-Format vorliegt.	400	`SetPlatformAttributes`
APNs — Principal ist ein abgelaufenes Zertifikat	Für den APNs Prinzipal, das SSL-Zertifikat, wurde ein abgelaufenes Zertifikat bereitgestellt.	400	`SetPlatformAttributes`
APNs — Principal ist kein von Apple ausgestelltes Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein nicht von Apple ausgestelltes Zertifikat angegeben.	400	`SetPlatformAttributes`
APNs — Der Schulleiter ist nicht angegeben	Der APNs Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde nicht bereitgestellt.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs — Der Berechtigungsnachweis wurde nicht bereitgestellt	Die APNs Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, wurden nicht bereitgestellt. Weitere Informationen finden Sie CreatePlatformApplicationin der Amazon Simple Notification Service API-Referenz.	400	`SetPlatformAttributes`
APNs — Die Anmeldeinformationen haben kein gültiges PEM-Format	Die APNs Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, haben kein gültiges PEM-Format.	400	`SetPlatformAttributes`
FCM — Server wird nicht bereitgestellt APIKey	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, wurden nicht angegeben. Weitere Informationen finden Sie CreatePlatformApplicationin der Amazon Simple Notification Service API-Referenz.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
FCM — Server APIKey ist eine Nullzeichenfolge	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind Null.	400	`SetPlatformAttributes`
ADM – clientId wurde nicht angegeben	Die erforderliche Zeichenfolge für die Client-ID liegt nicht vor.	400	`SetPlatformAttributes`
ADM – clientid ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für die Client-ID ist Null.	400	`SetPlatformAttributes`
ADM – clientsecret wurde nicht angegeben	Der erforderliche Clientschlüssel liegt nicht vor.	400	`SetPlatformAttributes`
ADM – clientsecret ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für den Clientschlüssel ist Null.	400	`SetPlatformAttributes`
EventEndpointUpdated hat ein ungültiges ARN-Format	EventEndpointUpdated hat ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventEndpointDeleted hat ein ungültiges ARN-Format	EventEndpointDeleted hat ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventEndpointUpdated hat ein ungültiges ARN-Format	EventEndpointUpdated hat ein ungültiges ARN-Format.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
EventDeliveryAttemptFailure hat ein ungültiges ARN-Format	EventDeliveryAttemptFailure hat ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventDeliveryFailure hat ein ungültiges ARN-Format	EventDeliveryFailure hat ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventEndpointCreated ist kein vorhandenes Thema	EventEndpointCreated ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventEndpointDeleted ist kein vorhandenes Thema	EventEndpointDeleted ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventEndpointUpdated ist kein vorhandenes Thema	EventEndpointUpdated ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure ist kein vorhandenes Thema	EventDeliveryAttemptFailure ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventDeliveryFailure ist kein vorhandenes Thema	EventDeliveryFailure ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`GetPlatformApplicationAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	403	`GetPlatformApplicationAttributes`
Der angegebene Token ist ungültig	Das angegebene Token ist ungültig.	400	`ListPlatformApplications`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`ListEndpointsByPlatformApplication`
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	404	`ListEndpointsByPlatformApplication`
Der angegebene Token ist ungültig	Das angegebene Token ist ungültig.	400	`ListEndpointsByPlatformApplication`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`DeletePlatformApplication`
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	403	`DeletePlatformApplication`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`CreatePlatformEndpoint`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	404	`CreatePlatformEndpoint`
Der Token wurde nicht angegeben	Das Token wurde nicht angegeben.	400	`CreatePlatformEndpoint`
Token hat nicht die richtige Länge	Das Token hat nicht die richtige Länge.	400	`CreatePlatformEndpoint`
Kunden-/Benutzerdaten sind zu groß	Die Kunden-/Benutzerdaten dürfen in der UTF-8-Codierung maximal 2048 Byte lang ein.	400	`CreatePlatformEndpoint`
Endpunkt-ARN ist ungültig	Der Endpunkt-ARN ist ungültig.	400	`DeleteEndpoint`
Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer	Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`DeleteEndpoint`
Endpunkt-ARN ist ungültig	Der Endpunkt-ARN ist ungültig.	400	`SetEndpointAttributes`
Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer	Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`SetEndpointAttributes`
Der Token wurde nicht angegeben	Das Token wurde nicht angegeben.	400	`SetEndpointAttributes`
Token hat nicht die richtige Länge	Das Token hat nicht die richtige Länge.	400	`SetEndpointAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
Kunden-/Benutzerdaten sind zu groß	Die Kunden-/Benutzerdaten dürfen in der UTF-8-Codierung maximal 2048 Byte lang ein.	400	`SetEndpointAttributes`
Endpunkt-ARN ist ungültig	Der Endpunkt-ARN ist ungültig.	400	`GetEndpointAttributes`
Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer	Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`GetEndpointAttributes`
Ziel-ARN ist ungültig	Der Ziel-ARN ist ungültig.	400	`Publish`
Der Ziel-ARN ist gültig, gehört aber nicht dem Benutzer	Der Ziel-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`Publish`
Nachrichtenformat ist ungültig	Das Nachrichtenformat ist ungültig.	400	`Publish`
Nachrichtengröße ist größer als die vom Protokoll/End-Service unterstützte Größe	Die Nachrichtengröße ist größer als die vom Protokoll/End-Service unterstützte Größe.	400	`Publish`

Verwendung des Time-to-Live-Nachrichten-Attributs von Amazon SNS für mobile Push-Benachrichtigungen

Amazon Simple Notification Service (Amazon SNS) unterstützt das Festlegen eines Time To Live (TTL)-Nachrichtenattributs für Mobile Push-Benachrichtigungen. Dies gilt zusätzlich zu der bestehenden Möglichkeit, TTL innerhalb des Amazon SNS SNS-Nachrichtentexts für die mobilen

Push-Benachrichtigungsdienste festzulegen, die dies unterstützen, wie Amazon Device Messaging (ADM) und Firebase Cloud Messaging (FCM) beim Senden an Android.

Das TTL-Nachrichtenattribut wird verwendet, um Ablauf-Metadaten über eine Nachricht anzugeben. Auf diese Weise können Sie angeben, wie viel Zeit der Push-Benachrichtigungsdienst wie Apple Push Notification Service (APNs) oder FCM benötigt, um die Nachricht an den Endpunkt zu übermitteln. Wenn die Nachricht aus irgendwelchen Gründen (z. B. bei Ausschalten des Mobilgeräts) innerhalb des angegebenen TTL nicht zustellbar ist, wird die Nachricht gelöscht und es werden keine weiteren Zustellversuche unternommen. Um TTL innerhalb von Nachrichtenattributen anzugeben, können Sie die AWS Management Console AWS Software Development Kits (SDKs) oder die Abfrage-API verwenden.

TTL-Nachrichtenattribute für Push-Benachrichtigungsservices

Im Folgenden finden Sie eine Liste der TTL-Nachrichtenattribute für Push-Benachrichtigungsdienste, die Sie bei Verwendung der API AWS SDKs oder der Abfrage-API festlegen können:

Push-Benachrichtigungsservice	TTL-Nachrichtenattribut
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple-Push-Benachrichtigungsdienst () APNs	`AWS.SNS.MOBILE.APNS.TTL`
Sandbox für den Apple-Push-Benachrichtigungsdienst (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCM beim Senden an Android)	`AWS.SNS.MOBILE.FCM.TTL`
Windows Push Notification Services (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Jeder Push-Benachrichtigungsservice geht anders mit TTL um. Amazon SNS bietet eine abstrakte Ansicht des TTL über alle Push-Benachrichtigungsservices, was das Festlegen des TTL erleichtert. Wenn Sie die AWS Management Console TTL (in Sekunden) angeben, müssen Sie den TTL-Wert nur einmal eingeben. Amazon SNS berechnet dann die TTL für jeden der ausgewählten Push-Benachrichtigungsdienste, wenn die Nachricht veröffentlicht wird.

Der TTL ist relativ zum Veröffentlichungszeitpunkt. Bevor eine Push-Benachrichtigung an einen bestimmten Push-Benachrichtigungsservice übergeben wird, berechnet Amazon SNS die dwell-Zeit (die Zeit zwischen dem Veröffentlichungszeitstempel und unmittelbar vor der Übergabe an einen Push-Benachrichtigungsservice) für die Push-Benachrichtigung und leitet den verbleibenden TTL an den betreffenden Push-Benachrichtigungsservice weiter. Wenn der TTL kürzer ist als die dwell-Zeit, versucht Amazon SNS keine Veröffentlichung.

Wenn Sie eine TTL für eine Push-Benachrichtigung angeben, muss der TTL-Wert eine positive Ganzzahl sein, es sei denn, der Wert von `0` hat eine spezifische Bedeutung für den Push-Benachrichtigungsdienst, z. B. with APNs und FCM (beim Senden an Android). Wenn der TTL-Wert auf `0` festgelegt ist und der Push-Benachrichtigungsservice für `0` keine spezifische Bedeutung vorsieht, wird die Nachricht von Amazon SNS verworfen. Weitere Informationen zum TTL-Parameter, der `0` bei der Verwendung auf eingestellt ist APNs, finden Sie in Tabelle A-3 Elementkennungen für Fernbenachrichtigungen in der Binary Provider-API-Dokumentation.

Rangfolge bei der Bestimmung von TTL

Die Rangfolge, die Amazon SNS zur Bestimmung des TTL-Werts für eine Push-Benachrichtigung verwendet, basiert auf der folgenden Reihenfolge, wobei die niedrigste Zahl die höchste Priorität hat:

1. TTL-Nachrichtenattribut
2. TTL-Nachrichtentext
3. Standard-TTL des Push-Benachrichtigungsservice (variiert je nach Service)
4. Standard-TTL von Amazon SNS (4 Wochen)

Wenn Sie für die gleiche Nachricht verschiedene TTL-Werte festlegen (einen in den Nachrichtenattributen und einen anderen im Nachrichtentext), ändert Amazon SNS den TTL im Nachrichtentext so, dass er dem in dem Nachrichtenattribut angegebenen TTL-Wert entspricht.

Angabe von TTL mit dem AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Mobile (Mobil) und dann Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie auf der Seite Mobile push notifications (Mobile Push-Benachrichtigungen) im Abschnitt Platform applications (Plattformanwendungen) eine Anwendung aus.

4. Wählen Sie auf der ***MyApplication*** Seite im Abschnitt Endpoints einen Anwendungsendpunkt aus und klicken Sie dann auf Nachricht veröffentlichen.
5. Geben Sie im Abschnitt Message details (Nachrichtendetails) die TTL ein (die Anzahl an Sekunden, die dem Push-Benachrichtigungs-Service zur Verfügung stehen, um die Nachricht an den Endpunkt zu übermitteln).
6. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

Unterstützte Regionen für mobile Amazon SNS SNS-Anwendungen

Derzeit können Sie mobile Anwendungen in den folgenden Regionen erstellen:

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Africa (Cape Town)
- Asien-Pazifik (Hongkong)
- Asien-Pazifik (Jakarta)
- Asien-Pazifik (Mumbai)
- Asien-Pazifik (Osaka)
- Asien-Pazifik (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- Europe (Frankfurt)
- Europa (Irland)
- Europa (London)
- Europa (Milan)
- Europe (Paris)
- Europe (Stockholm)
- Middle East (Bahrain)

- Naher Osten (VAE)
- Südamerika (São Paulo)
- AWS GovCloud (USA, Westen)

Bewährte Methoden für die Verwaltung von mobilen Amazon SNS SNS-Push-Benachrichtigungen

In diesem Abschnitt werden bewährte Methoden beschrieben, die Ihnen dabei helfen können, die Kundenbindung zu verbessern.

Endpunktverwaltung

Zustellungsprobleme können in Situationen auftreten, in denen sich Gerätetokens aufgrund einer Benutzeraktion auf dem Gerät ändern (z. B. wenn eine App auf dem Gerät neu installiert wird) oder wenn Zertifikatsupdates sich auf Geräte auswirken, auf denen eine bestimmte iOS-Version ausgeführt wird. Es wird von Apple empfohlen, sich bei jedem Start Ihrer App bei APNs jedem Start zu registrieren.

Da sich das Geräte-Token nicht jedes Mal ändert, wenn eine App von einem Benutzer geöffnet wird, kann die idempotente API `CreatePlatformEndpoint` verwendet werden. Dies kann jedoch zu Duplikaten für dasselbe Gerät führen, wenn das Token selbst ungültig ist oder wenn der Endpunkt gültig, aber deaktiviert ist (z. B. wenn Produktions- und Sandbox-Umgebungen nicht übereinstimmen).

Es kann ein Mechanismus zur Verwaltung von Geräte-Token wie der im Pseudo-Code verwendet werden.

Informationen zur Verwaltung und Pflege von FCM v1-Gerätetokens finden Sie unter. Amazon SNS SNS-Verwaltung von Firebase Cloud Messaging-Endpunkten

Protokollieren des Zustellungsstatus

Für die Überwachung des Zustellungsstatus von Push-Benachrichtigungen empfehlen wir, die Protokollierung des Zustellungsstatus für Ihre Amazon-SNS-Plattformanwendung zu aktivieren. Dies hilft bei der Behebung von Zustellungsfehlern, da die Protokolle Anbieter-Antwortcodes enthalten, die vom Push-Plattformservice zurückgegeben wurden. Weitere Informationen zum Aktivieren der Protokollierung des Zustellungsstatus finden Sie unter Wie greife ich auf Zustellungsprotokolle für Amazon SNS-Themen für Push-Benachrichtigungen zu?.

Ereignis-Benachrichtigungen

Um Endpunkte auf ereignisgesteuerte Weise zu verwalten, können Sie die Funktion Ereignisbenachrichtigungen verwenden. Dies ermöglicht es dem konfigurierten Amazon-SNS-Thema, Ereignisse an die Abonnenten zu verteilen, wie beispielsweise eine Lambda-Funktion für Plattformanwendungsereignisse zur Erstellung, Löschung, Aktualisierung und fehlgeschlagenen Zustellung von Endpunkten.

Einrichtung und Verwaltung von Amazon SNS SNS-E-Mail-Abonnements

Mit dem, oder AWS SDK für .NET können Sie eine E-Mail-Adresse für ein Amazon SNS SNS-Thema abonnieren. AWS Management Console AWS SDK für Java

> **Hinweise**
>
> - Die Anpassung des E-Mail-Nachrichtentexts wird nicht unterstützt. Die E-Mail-Übermittlungsfunktion dient dazu, interne Systemwarnungen bereitzustellen, keine Marketingnachrichten.
> - Das direkte Abonnieren von E-Mail-Endpunkten wird nur für Standardthemen unterstützt.
> - Der Durchsatz der E-Mail-Zustellung wird gedrosselt. Weitere Informationen finden Sie unter Amazon SNS SNS-Kontingente.

> **Important**
>
> Informationen zum Verhindern, dass Empfänger von Mailinglisten alle Empfänger von E-Mails zu Amazon-SNS-Themen abmelden, finden Sie unter Einrichten eines E-Mail-Abonnements, das zum Abbestellen eine Authentifizierung voraussetzt vom AWS -Support.

Abonnieren einer E-Mail-Adresse für ein Amazon SNS SNS-Thema mit dem AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.

2. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:
   a. Für das Thema-ARN wählen Sie den Amazon-Ressourcennamen (ARN) eines Themas.
   b. Wählen Sie unter Protocol (Protokoll) die Option Email (E-Mail) aus.
   c. Geben Sie unter Endpunkt Ihre E-Mail-Adresse ein.
   d. (Optional) Um eine Filterrichtlinie zu konfigurieren, erweitern Sie den EintragAbonnement-FilterrichtlinieAbschnitts erstellt. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.
   e. (Optional) Zur Aktivierung der nutzlastbasierten Filterung konfigurieren Sie `Filter Policy Scope` auf `MessageBody`. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich.
   f. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für das Abonnement zu konfigurieren, erweitern Sie den Abschnitt Redrive-Richtlinie (Warteschlange für unzustellbare Nachrichten). Weitere Informationen finden Sie unter Amazon SNS SNS-Warteschlangen für unzustellbare Briefe.
   g. Wählen Sie Create subscription (Abonnement erstellen) aus.

      Die Konsole erstellt das Abonnement und öffnet die Abodetails.

Sie müssen das Abonnement bestätigen, bevor die E-Mail-Adresse Nachrichten empfangen kann.

Ein Abonnement bestätigen

1. Überprüfen Sie Ihren E-Mail-Posteingang und wählen Sie Abonnement bestätigen in der E-Mail von Amazon SNS.
2. Amazon SNS öffnet Ihren Webbrowser und zeigt eine Abonnementbestätigung mit Ihrer Abonnement-ID an.

Abonnieren einer E-Mail-Adresse für ein Amazon SNS SNS-Thema mithilfe eines SDK AWS

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Die Dateien für die gemeinsame Konfiguration und die Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Die folgenden Code-Beispiele zeigen, wie `Subscribe` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

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

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

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

- Details zu API finden Sie unter Abonnieren in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
//! 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();
}
```

Abonnieren Sie ein Thema mit einer mobilen Anwendung.

```
//! 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();
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
//! 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();
```

```
}
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        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;
            }
```

Abonnieren Sie ein Thema mit einem Filter.

```
        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;
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So abonnieren Sie ein Thema

Der folgende `subscribe`-Befehl abonniert das angegebene Thema mit eine E-Mail-Adresse.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Ausgabe:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API-Details finden Sie unter Subscribe in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
import (
 "context"
 "encoding/json"
 "log"

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

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
```

```
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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);
        }
    }
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
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);
        }
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
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 "";
    }
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

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

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
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;
};
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
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;
};
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
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;
};
```

Abonnieren Sie ein Thema mit einem Filter.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
```

```
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Details zu API finden Sie unter Subscribe (Abonnieren) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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());
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
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());
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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
```

- Details zu API finden Sie unter Abonnieren in der AWS API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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(())
}
```

- Details zu API finden Sie unter Abonnieren in der AWS -SDK-für-Rust-API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
   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.
```

- Details zu API finden Sie unter Subscribe (Anmelden) in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Abonnieren Sie eine Telefonnummer für ein Thema, um Benachrichtigungen per SMS zu erhalten.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Einzelheiten zur API finden Sie unter Abonnieren im AWS SDK für die Swift-API-Referenz.

Codebeispiele für Amazon SNS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon SNS mit einem AWS Software Development Kit (SDK) verwendet wird.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarios anzeigen.

Szenarien sind Code-Beispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte

Hello Amazon SNS

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon SNS.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.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})
```

Code für die Quelldatei „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;
}
```

- Einzelheiten zur API finden Sie ListTopicsunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Initialisieren Sie einen SNS-Client und listen Sie Themen in Ihrem Konto auf.

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

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API-Details finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
```

```
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Das Hauptprogramm von Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
```

```
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}
```

```
/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Swift.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon SNS AWS SDKs
  - Hello Amazon SNS
  - Aktionen für Amazon SNS mit AWS SDKs
    - Verwendung CheckIfPhoneNumberIsOptedOut mit einem AWS SDK oder CLI
    - Verwendung ConfirmSubscription mit einem AWS SDK oder CLI
    - Verwendung CreateTopic mit einem AWS SDK oder CLI
    - Verwendung DeleteTopic mit einem AWS SDK oder CLI
    - Verwendung GetSMSAttributes mit einem AWS SDK oder CLI
    - Verwendung GetTopicAttributes mit einem AWS SDK oder CLI
    - Verwendung ListPhoneNumbersOptedOut mit einem AWS SDK oder CLI
    - Verwendung ListSubscriptions mit einem AWS SDK oder CLI
    - Verwendung ListTopics mit einem AWS SDK oder CLI
    - Verwendung Publish mit einem AWS SDK oder CLI
    - Verwendung SetSMSAttributes mit einem AWS SDK oder CLI
    - Verwendung SetSubscriptionAttributes mit einem AWS SDK oder CLI
    - SetSubscriptionAttributesRedrivePolicyMit einem AWS SDK verwenden

    - Verwendung SetTopicAttributes mit einem AWS SDK oder CLI
    - Verwendung Subscribe mit einem AWS SDK oder CLI
    - Verwendung TagResource mit einem AWS SDK oder CLI
    - Verwendung Unsubscribe mit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon SNS AWS SDKs
  - Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
  - Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
  - Erstellen Sie mithilfe eines SDK einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen AWS
  - Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
  - Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
  - Erstellen und Veröffentlichen in einem FIFO-Amazon-SNS-Thema mithilfe eines SDK AWS
  - Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS
  - Veröffentlichen Sie SMS-Nachrichten mithilfe eines SDK zu einem Amazon SNS SNS-Thema AWS
  - Veröffentlichen Sie mit Amazon S3 mithilfe eines SDK eine AWS große Nachricht in Amazon SNS
  - Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem SDK AWS
  - Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS
  - Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
  - Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion
- Serverlose Beispiele für Amazon SNS
  - Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Grundlegende Beispiele für die Verwendung von Amazon SNS AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Simple Notification Service mit verwenden können AWS SDKs.

Beispiele

- Hello Amazon SNS
- Aktionen für Amazon SNS mit AWS SDKs
  - Verwendung CheckIfPhoneNumberIsOptedOut mit einem AWS SDK oder CLI
  - Verwendung ConfirmSubscription mit einem AWS SDK oder CLI
  - Verwendung CreateTopic mit einem AWS SDK oder CLI
  - Verwendung DeleteTopic mit einem AWS SDK oder CLI
  - Verwendung GetSMSAttributes mit einem AWS SDK oder CLI
  - Verwendung GetTopicAttributes mit einem AWS SDK oder CLI
  - Verwendung ListPhoneNumbersOptedOut mit einem AWS SDK oder CLI
  - Verwendung ListSubscriptions mit einem AWS SDK oder CLI
  - Verwendung ListTopics mit einem AWS SDK oder CLI
  - Verwendung Publish mit einem AWS SDK oder CLI
  - Verwendung SetSMSAttributes mit einem AWS SDK oder CLI
  - Verwendung SetSubscriptionAttributes mit einem AWS SDK oder CLI
  - SetSubscriptionAttributesRedrivePolicyMit einem AWS SDK verwenden
  - Verwendung SetTopicAttributes mit einem AWS SDK oder CLI
  - Verwendung Subscribe mit einem AWS SDK oder CLI
  - Verwendung TagResource mit einem AWS SDK oder CLI
  - Verwendung Unsubscribe mit einem AWS SDK oder CLI

Hello Amazon SNS

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon SNS.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.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})
```

Code für die Quelldatei „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;
}
```

- Einzelheiten zur API finden Sie ListTopicsunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Initialisieren Sie einen SNS-Client und listen Sie Themen in Ihrem Konto auf.

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

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API-Details finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
```

```
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Das Hauptprogramm von Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
```

```
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}
```

```
/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Amazon SNS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne Amazon SNS SNS-Aktionen mit AWS SDKs ausführen. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-SNS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon SNS AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Simple Notification Service.

Beispiele

- Verwendung CheckIfPhoneNumberIsOptedOut mit einem AWS SDK oder CLI
- Verwendung ConfirmSubscription mit einem AWS SDK oder CLI
- Verwendung CreateTopic mit einem AWS SDK oder CLI

- Verwendung DeleteTopic mit einem AWS SDK oder CLI
- Verwendung GetSMSAttributes mit einem AWS SDK oder CLI
- Verwendung GetTopicAttributes mit einem AWS SDK oder CLI
- Verwendung ListPhoneNumbersOptedOut mit einem AWS SDK oder CLI
- Verwendung ListSubscriptions mit einem AWS SDK oder CLI
- Verwendung ListTopics mit einem AWS SDK oder CLI
- Verwendung Publish mit einem AWS SDK oder CLI
- Verwendung SetSMSAttributes mit einem AWS SDK oder CLI
- Verwendung SetSubscriptionAttributes mit einem AWS SDK oder CLI
- SetSubscriptionAttributesRedrivePolicyMit einem AWS SDK verwenden
- Verwendung SetTopicAttributes mit einem AWS SDK oder CLI
- Verwendung Subscribe mit einem AWS SDK oder CLI
- Verwendung TagResource mit einem AWS SDK oder CLI
- Verwendung Unsubscribe mit einem AWS SDK oder CLI

Verwendung **`CheckIfPhoneNumberIsOptedOut`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CheckIfPhoneNumberIsOptedOut` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So überprüfen Sie SMS-Nachrichten-Abmeldungen für eine Telefonnummer

Im folgenden `check-if-phone-number-is-opted-out` Beispiel wird geprüft, ob die angegebene Telefonnummer den Empfang von SMS-Nachrichten vom AWS Girokonto deaktiviert hat.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Ausgabe:

```
{
    "isOptedOut": false
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ConfirmSubscription`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ConfirmSubscription` verwendet wird.

CLI

AWS CLI

So bestätigen Sie ein Abonnement

Mit dem folgenden `confirm-subscription`-Befehl wird der Bestätigungsvorgang abgeschlossen, der gestartet wurde, als Sie ein SNS-Thema mit dem Namen `my-topic` abonniert haben. Der --token-Parameter stammt aus der Bestätigungsnachricht, die an den im Abonnementaufruf angegebenen Benachrichtigungsendpunkt gesendet wurde.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
```

```
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Ausgabe:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
       }
   }
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK für PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CreateTopic`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTopic` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema mit einem bestimmten Namen.

```
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;
        }
    }
```

Erstellen Sie ein neues Thema mit einem Namen und spezifischen FIFO- und Deduplizierungsattributen.

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

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein SNS-Thema

Das folgende `create-topic`-Beispiel erstellt ein SNS-Thema namens `my-topic`.

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

Ausgabe:

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

Weitere Informationen finden Sie unter Verwenden der AWS Befehlszeilenschnittstelle mit Amazon SQS und Amazon SNS im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter CreateTopic AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

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

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
```

```
 }

 return topicArn, err
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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 "";
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

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

- API-Details finden Sie CreateTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie CreateTopicin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   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.
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(name: name)
```

```
        )

        guard let arn = output.topicArn else {
            print("No topic ARN returned by Amazon SNS.")
            return
        }
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteTopic`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTopic` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-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;
    }
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
```

```
 "log"

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

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



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

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

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

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

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

```
    }
}
```

- API-Details finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
```

```
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetSMSAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSMSAttributes` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie unter Get SMSAttributes in AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So führen Sie die Standard-SMS-Nachrichtenattribute auf

Das folgende `get-sms-attributes`-Beispiel führt die Standardattribute für das Senden von SMS-Nachrichten auf.

```
aws sns get-sms-attributes
```

Ausgabe:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Einzelheiten zur API finden Sie unter Get SMSAttributes in AWS CLI Command Reference.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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");
    }
}
```

- Einzelheiten zur API finden Sie unter Get SMSAttributes in AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie unter Get SMSAttributes in AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie unter Get SMSAttributes in AWS SDK für PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetTopicAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetTopicAttributes` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie die Attribute eines Themas ab

Im folgenden `get-topic-attributes`-Beispiel werden die Attribute für das angegebene Thema angezeigt.

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

Ausgabe:

```
{
    "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"
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

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

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- API-Details finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für PHP API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListPhoneNumbersOptedOut`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListPhoneNumbersOptedOut` verwendet wird.

CLI

AWS CLI

So führen Sie Abmeldungen für SMS-Nachrichten auf

Das folgende `list-phone-numbers-opted-out`-Beispiel listet die Telefonnummern auf, bei denen der Empfang von SMS-Nachrichten abbestellt wurde.

```
aws sns list-phone-numbers-opted-out
```

Ausgabe:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK für PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListSubscriptions`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListSubscriptions` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So führen Sie Ihre SNS-Abonnements auf

Im folgenden `list-subscriptions` Beispiel wird eine Liste der SNS-Abonnements in Ihrem AWS Konto angezeigt.

```
aws sns list-subscriptions
```

Ausgabe:

```
{
```

```
    "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"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- API-Details finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie ListSubscriptionsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListTopics`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTopics` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So führen Sie Ihre SNS-Themen auf

Das folgende `list-topics` Beispiel listet alle SNS-Themen in Ihrem AWS Konto auf.

```
aws sns list-topics
```

Ausgabe:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- API-Details finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie ListTopicsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
```

```
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Publish`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Publish` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen einer SMS-Nachricht
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen einer Nachricht für ein Thema.

```
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}");
        }
```

```
    }
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

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

Wenden Sie die Benutzerauswahl auf die Veröffentlichungsaktion an.

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

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        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;
        }
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So veröffentlichen Sie eine Nachricht für ein Thema

Das folgende `publish`-Beispiel veröffentlicht die angegebene Nachricht im angegebenen SNS-Thema. Die Nachricht stammt aus einer Textdatei, in der Sie Zeilenumbrüche einfügen können.

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

Inhalt von `message.txt`:

```
Hello World
Second Line
```

Ausgabe:

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

Beispiel 2: So veröffentlichen Sie eine SMS-Nachricht an eine Telefonnummer

Im folgenden `publish`-Beispiel wird Nachricht `Hello world!` an Telefonnummer `+1-555-555-0100` veröffentlicht.

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

Ausgabe:

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

- API-Details finden Sie unter Publish in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

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

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

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

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

    let groupId;
    let deduplicationId;
    let choices;

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

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

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

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

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

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

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

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

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

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für PHP -API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

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

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$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
```

- Einzelheiten zur API finden Sie unter In AWS -Tools für PowerShell Cmdlet-Referenz veröffentlichen (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

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

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$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
```

- Einzelheiten zur API finden Sie unter In AWS -Tools für PowerShell Cmdlet-Referenz veröffentlichen (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen Sie eine Nachricht mit Attributen, damit ein Abonnement basierend auf Attributen filtern kann.

```
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
```

Veröffentlichen Sie eine Nachricht, die basierend auf dem Protokoll des Abonnenten unterschiedliche Formen annimmt.

```
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
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Details zu API finden Sie unter Veröffentlichen in der Referenz für das AWS -SDK für Rust-API.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.publish(
            input: PublishInput(
                message: message,
                topicArn: arn
            )
        )

        guard let messageId = output.messageId else {
            print("No message ID received from Amazon SNS.")
            return
        }

        print("Published message with ID \(messageId)")
```

- Einzelheiten zur API finden Sie unter Im AWS SDK veröffentlichen für die Swift-API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SetSMSAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetSMSAttributes` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

So verwenden Sie Amazon SNS, um das SMSType Standardattribut festzulegen.

```
//! 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();
}
```

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So legen Sie SMS-Nachrichtenattribute fest

Im folgenden `set-sms-attributes`-Beispiel wird die standardmäßige Absender-ID für SMS-Nachrichten auf `MyName` festgelegt.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SetSubscriptionAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetSubscriptionAttributes` verwendet wird.

CLI

AWS CLI

So legen Sie Abonnementattribute fest

Im folgenden `set-subscription-attributes`-Beispiel wird das `RawMessageDelivery`-Attribut auf ein SQS-Abonnement festgelegt.

```
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
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird ein `FilterPolicy`-Attribut auf ein SQS-Abonnement festgelegt.

```
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\"] }"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird das `FilterPolicy`-Attribut von einem SQS-Abonnement entfernt.

```
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 "{}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon SNS mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

`SetSubscriptionAttributesRedrivePolicy`Mit einem AWS SDK verwenden

Das folgende Codebeispiel zeigt, wie es verwendet wird`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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);
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SetTopicAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetTopicAttributes` verwendet wird.

CLI

AWS CLI

So legen Sie ein Attribut für ein Thema fest

Im folgenden `set-topic-attributes`-Beispiel wird das `DisplayName`-Attribute für das angegebene Thema festgelegt.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- API-Details finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für PHP API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# 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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Subscribe`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Subscribe` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

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

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

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

- Details zu API finden Sie unter Abonnieren in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
//! 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();
}
```

Abonnieren Sie ein Thema mit einer mobilen Anwendung.

```
//! 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();
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
//! 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();
}
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        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;
           }
```

Abonnieren Sie ein Thema mit einem Filter.

```
       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;
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So abonnieren Sie ein Thema

Der folgende `subscribe`-Befehl abonniert das angegebene Thema mit eine E-Mail-Adresse.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Ausgabe:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API-Details finden Sie unter Subscribe in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
import (
 "context"
 "encoding/json"
 "log"

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

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
```

```
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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);
```

```
        }
    }
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
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);
        }
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
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 "";
    }
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

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

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
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;
};
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
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;
};
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
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;
};
```

Abonnieren Sie ein Thema mit einem Filter.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Details zu API finden Sie unter Subscribe (Abonnieren) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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());
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
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());
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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
```

- Details zu API finden Sie unter Abonnieren in der AWS API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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(())
}
```

- Details zu API finden Sie unter Abonnieren in der AWS -SDK-für-Rust-API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
   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.
```

- Details zu API finden Sie unter Subscribe (Anmelden) in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Abonnieren Sie eine Telefonnummer für ein Thema, um Benachrichtigungen per SMS zu erhalten.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Einzelheiten zur API finden Sie unter Abonnieren im AWS SDK für die Swift-API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`TagResource`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Thema ein Tag hinzu

Das folgende `tag-resource`-Beispiel fügt dem angegebenen Amazon-SNS-Thema ein Metadaten-Tag hinzu.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- API-Details finden Sie TagResourcein der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Unsubscribe`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Unsubscribe` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Melden Sie sich mit einem Abonnement-ARN von einem Thema ab.

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

- Details zu API finden Sie unter Abmelden in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So melden Sie sich von einem Thema ab

Im folgenden `unsubscribe`-Beispiel wird das angegebene Abonnement aus einem Thema gelöscht.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Details finden Sie unter Unsubscribe in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
```

```
        }
    }
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
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({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
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;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
```

```
}
```

- Details zu API finden Sie unter Unsubscribe (Abmelden) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Details zu API finden Sie unter Abmelden in der AWS API-Referenz zu SDK for Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   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.
```

- Details zu API finden Sie unter Abmelden in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Einzelheiten zur API finden Sie unter Unsubscribe in AWS SDK für die Swift-API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für die Verwendung von Amazon SNS AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon SNS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb von Amazon SNS oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
- Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
- Erstellen Sie mithilfe eines SDK einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen AWS
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erstellen und Veröffentlichen in einem FIFO-Amazon-SNS-Thema mithilfe eines SDK AWS
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS
- Veröffentlichen Sie SMS-Nachrichten mithilfe eines SDK zu einem Amazon SNS SNS-Thema AWS
- Veröffentlichen Sie mit Amazon S3 mithilfe eines SDK eine AWS große Nachricht in Amazon SNS
- Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem SDK AWS
- Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Code-Beispiele zeigen, wie man eine Anwendung erstellt, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

Java

SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Das Beispiel zeigt, wie man eine App erstellt, die es Benutzern ermöglicht, Daten an eine Amazon-DynamoDB-Tabelle zu übermitteln und eine Textnachricht an den Administrator mit Amazon Simple Notification Service (Amazon SNS) zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt

Die folgende Code-Beispiele zeigen, wie man eine Anwendung erstellt, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

.NET

SDK für .NET

Zeigt, wie man die .NET-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionalität verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Java

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung des Beispiels, das die Java Async API verwendet, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Kotlin

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie mithilfe eines SDK einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen AWS

Das folgende Codebeispiel zeigt, wie ein Plattformendpunkt für Amazon-SNS-Push-Benachrichtigungen erstellt werden.

CLI

AWS CLI

So erstellen Sie ein Plattformanwendungsendpunkt

Im folgenden `create-platform-endpoint`-Beispiel wird mithilfe des angegebenen Tokens ein Endpunkt für die angegebene Plattformanwendung erstellt.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Ausgabe:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
```

```
              token - The device token or registration ID of the mobile device.
This is a unique
              identifier provided by the device platform (e.g., Apple Push
Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
              for Android devices) when the mobile app is registered to receive
push notifications.

              platformApplicationArn - The ARN value of platform application.
You can get this value from the AWS Management Console.\s

           """;

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

       String token = args[0];
       String platformApplicationArn = args[1];
       SnsClient snsClient = SnsClient.builder()
           .region(Region.US_EAST_1)
           .build();

       createEndpoint(snsClient, token, platformApplicationArn);
   }
   public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
       System.out.println("Creating platform endpoint with token " + token);
       try {
           CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
               .token(token)
               .platformApplicationArn(platformApplicationArn)
               .build();

           CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
           System.out.println("The ARN of the endpoint is " +
response.endpointArn());

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

```
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchron APIs , um einen Job zu starten, der nach Abschluss des Jobs eine Benachrichtigung in einem Amazon Simple Notification Service (Amazon SNS) -Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen und Veröffentlichen in einem FIFO-Amazon-SNS-Thema mithilfe eines SDK AWS

Die folgenden Code-Beispiele zeigen, wie man ein FIFO-Amazon-SNS-Thema erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
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",
                "FifoThroughputScope", "MessageGroup");

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

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

            return topicArn;

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

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

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
```

```
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

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

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

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

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

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

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

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
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),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            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
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen

- Abonnieren

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " 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.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie SMS-Nachrichten mithilfe eines SDK zu einem Amazon SNS SNS-Thema AWS

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.

- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
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 "";
    }
}
```

Abonnieren eines Endpunkts für ein Thema.

```
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);
        }
    }
}
```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```
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);
        }
    }
}
```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```
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);
        }
    }
```

```
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie mit Amazon S3 mithilfe eines SDK eine AWS große Nachricht in Amazon SNS

Das folgende Code-Beispiel zeigt, wie man mit Amazon S3 eine große Nachricht an Amazon SNS veröffentlicht, um die Nachrichtennutzlast zu speichern.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Java. Die Nachricht, die Sie senden, verweist auf ein Amazon S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

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

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem SDK AWS

In den folgenden Codebeispielen wird veranschaulicht, wie Sie SMS-Nachrichten über Amazon SNS veröffentlichen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}");
```

```
            }
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * 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();
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für C++ -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Console application to run a feature 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 scenario 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 scenario.
    /// </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 scenario, 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;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SQS-Operationen einschließt.

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

Erstellen Sie eine Klasse, die Amazon–SNS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
```

```
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

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

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

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

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
```

```
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

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

```
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

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


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
```

```
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;

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

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or
 automatically generated "
```

```
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
```

```
        }

        request.SetName(topicName);

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

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

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

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
```

```
                        << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                  "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

```
                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
```

```
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
```

```
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
```

```
                            << "\"'s subscription to the topic \""
                            << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

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

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
```

```
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

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

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

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

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

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
```

```
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

```
                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
```

```
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
```

```
                        << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

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

        if (outcome.IsSuccess()) {
```

```
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                           const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";
```

```
    return policyStream.str();
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

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

```
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
```

```
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
```

```
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
```

```
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
   for askAboutTones {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelections = append(toneSelections, ToneChoices[toneIndex])
    askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
 the filter? (y/n) ", "y")
   }
   log.Printf("Your subscription will be filtered to only pass the following
 tones: %v\n", toneSelections)
   filterPolicy = map[string][]string{TONE_KEY: toneSelections}
  }
 }

 subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
 filterPolicy)
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
 string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
```

```
    log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
     "All messages within the same group will be received in the order they were
 published.")
    groupId = runner.questioner.Ask("Enter a message group ID: ")
    if !contentBasedDeduplication {
     log.Println("Because you are not using content-based deduplication,\n" +
      "you must enter a deduplication ID.")
     dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
    }
   }
   if usingFilters {
    if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
     toneIndex := runner.questioner.AskChoice(
      "Enter the number of the tone you want to filter by:\n", ToneChoices)
     toneSelection = ToneChoices[toneIndex]
    }
   }

   err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
   if err != nil {
    panic(err)
   }
   log.Println(("Your message was published."))

   publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
```

```
    }
    messages = append(messages, currentMsgs...)
   }
   if len(messages) == 0 {
    log.Printf("No messages were received by queue %v.\n", queueUrl)
   } else if len(messages) == 1 {
    log.Printf("One message was received by queue %v:\n", queueUrl)

   } else {
    log.Printf("%v messages were received by queue %v:\n", len(messages),
  queueUrl)
   }
   for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
     panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
   }

   if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
     panic(err)
    }
   }
  }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
```

```
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
```

```
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die die in diesem Beispiel verwendeten Amazon SNS SNS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
```

```
)

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
```

```
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Definieren Sie eine Struktur, die die in diesem Beispiel verwendeten Amazon SQS SQS-Aktionen umschließt.

```
import (
```

```
 "context"
 "encoding/json"
 "fmt"
 "log"

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

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}
```

```
// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
```

```
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
```

```
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management
 Console\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
```

```
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

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

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
```

```
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
```

```
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,
\n"
                +
                "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
 must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
```

```
            ]
        }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
```

```
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
```

```
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
 duplication, groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
 continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
 Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
```

```
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

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

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

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

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

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

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
```

```
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

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

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

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

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
```

```
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

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

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

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

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

```
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
```

```
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
```

```
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

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

    // Attach a policy to the queue.
```

```
   public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
       try {
           Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
           attrMap.put(QueueAttributeName.POLICY, policy);

           SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
               .queueUrl(queueUrl)
               .attributes(attrMap)
               .build();

           sqsClient.setQueueAttributes(attributesRequest);
           System.out.println("The policy has been successfully attached.");

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

   public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
       // Specify the attributes to retrieve.
       List<QueueAttributeName> atts = new ArrayList<>();
       atts.add(QueueAttributeName.QUEUE_ARN);

       GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
           .queueUrl(queueUrl)
           .attributeNames(atts)
           .build();

       GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
       Map<String, String> queueAtts = response.attributesAsStrings();
       for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
           return queueAtt.getValue();

       return "";
   }

   public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
```

```
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

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

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

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

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

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

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

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

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

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateQueue

- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt für dieses Szenario.

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

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
```

```
};
```

Der vorherige Code stellt die erforderlichen Abhängigkeiten bereit und startet das Szenario. Der nächste Abschnitt enthält den Großteil des Beispiels.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
```

```
    this.prompter = prompter;
    this.logger = logger;
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );
```

```
    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
```

```
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
```

```
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
```

```
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

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

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
```

```
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

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

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

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );
```

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

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

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
```

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

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
```

```
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
```

```
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)
```

```
    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
```

```
        },
        "Action": "sqs:SendMessage",
                "Resource": "$sqsQueueArn",
                "Condition": {
            "ArnEquals": {
                "aws:SourceArn": "$topicArn"
            }
        }
    }
    ]
    }"""
   setQueueAttr(sqsQueueUrl, policy)
   println(DASHES)

   println(DASHES)
   println("6. Subscribe to the SQS queue.")
   if (selectFIFO) {
       println(
           """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
       )
       println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
       val filterAns: String = input.nextLine()
       if (filterAns.compareTo("y") == 0) {
           var moreAns = false
           println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
           println("1. cheerful")
           println("2. funny")
           println("3. serious")
           println("4. sincere")
           while (!moreAns) {
               println("Select a number or choose 0 to end.")
               val ans: String = input.nextLine()
               when (ans) {
                   "1" -> filterList.add("cheerful")
                   "2" -> filterList.add("funny")
                   "3" -> filterList.add("serious")
                   "4" -> filterList.add("sincere")
                   else -> moreAns = true
```

```
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
```

```
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

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

```
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
```

```
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

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

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
```

```
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
```

```
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }
```

```
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
```

```
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

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

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
```

```
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
            str = readLine()

            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
```

```
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
```

```
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
 attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
```

```
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the
 'tone' attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
 throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]
```

```
            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
String? {
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }
```

```
        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }
```

```
    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
                                ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
```

```
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
            }
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues
example.
              In this workflow, you'll create an SNS topic, then create two SQS
```

```
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well
 as
              the queue subscriptions. You can then post messages to the topic
 and
              receive the results on the queues.
              \(rowOfStars)\n
              """
       )

       // 0. Create SNS and SQS clients.

       let snsConfig = try await SNSClient.SNSClientConfiguration(region:
 region)
       let snsClient = SNSClient(config: snsConfig)

       let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
 region)
       let sqsClient = SQSClient(config: sqsConfig)

       // 1. Ask the user whether to create a FIFO topic. If so, ask whether
       //    to use content-based deduplication instead of requiring a
       //    deduplication ID.

       let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
       var isContentBasedDeduplication = false

       if isFIFO {
           print("""
                 \(rowOfStars)
                 Because you've chosen to create a FIFO topic, deduplication is
                 supported.

                 Deduplication IDs are either set in the message or are
 automatically
                 generated from the content using a hash function.

                 If a message is successfully published to an SNS FIFO topic,
 any
                 message published and found to have the same deduplication ID
                 (within a five-minute deduplication interval), is accepted but
                 not delivered.
```

```
                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
 deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
 create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
```

```
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")
```

```
        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
 messages will
                be received in the queue. For information about message
 filtering, see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's
 subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
```

```
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                  queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil

        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }
```

```
        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
 publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.
\n
                        """
                    )
                }
```

```
                publishInput.messageGroupId = stringRequest(prompt: "Enter a
message group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based
deduplication, you
                              must enter a deduplication ID. If other messages
with the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue]
= [:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType:
"String", stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }
```

```
            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )
```

```
        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine von Amazon API Gateway aufgerufene AWS Lambda Funktion erstellt wird.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.

- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Serverlose Beispiele für Amazon SNS

Die folgenden Codebeispiele zeigen, wie Sie Amazon SNS mit AWS SDKs verwenden.

Beispiele

- Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von .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 für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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 für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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 für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
```

```
    throw err;
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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 für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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 für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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 für Rust

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Amazon-SNS-Sicherheit

Das AWS Modell der gilt für den Datenschutz in Amazon Simple Notification Service. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt umfasst die Sicherheitskonfiguration und die Verwaltungsaufgaben für die AWS Dienste, die Sie verwenden. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und individuelle Benutzerkonten mit AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir empfehlen TLS 1.2 oder höher.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen innerhalb der AWS Dienste.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu sichern.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.
- Nachrichtendatenschutz
  - Der Nachrichtendatenschutz ist eine neue Hauptfunktion von Amazon SNS.
  - Verwenden Sie MDP, um Nachrichten nach vertraulichen oder sensiblen Informationen zu durchsuchen.
  - Stellen Sie Nachrichtenüberwachung für den gesamten Inhalt bereit, der das Thema passiert.
  - Stellen Sie Inhaltszugriffskontrollen für Nachrichten zur Verfügung, die im Thema veröffentlicht wurden, und für Nachrichten, die vom Thema zugestellt wurden.

> **Important**
>
> Wir empfehlen dringend, in Freitextfeldern wie z. B. im Feld Name keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon SNS oder anderen Amazon Web Services über die Konsole AWS CLI, API oder AWS SDKs arbeiten. Alle Daten, die Sie in Tags (Markierungen) oder Freiformfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Amazon SNS SNS-Datenverschlüsselung

Dieser Datenschutz bezieht sich auf Daten bei der Übertragung (wenn sie zu Amazon SNS oder von diesem geschickt werden), ebenso wie auf ruhende Daten (die in Amazon-S3-Rechenzentren auf Datenträgern gespeichert sind). Sie können Daten mithilfe von Secure Sockets Layer (SSL) oder einer clientseitigen Verschlüsselung während der Übertragung schützen. Standardmäßig speichert Amazon SNS Nachrichten und Dateien mit Festplattenverschlüsselung. Sie können Daten im Ruhezustand schützen, indem Sie Amazon SNS auffordern, Ihre Nachrichten zu verschlüsseln, bevor Sie sie im verschlüsselten Dateisystem in den Rechenzentren speichern. Amazon SNS empfiehlt die Verwendung von SSE für eine optimierte Datenverschlüsselung.

Sicherung von Amazon SNS SNS-Daten mit serverseitiger Verschlüsselung

Mit serverseitiger Verschlüsselung (SSE) können Sie vertrauliche Daten in verschlüsselten Themen speichern, indem Sie den Inhalt von Nachrichten in Amazon SNS SNS-Themen mithilfe von Schlüsseln schützen, die in AWS Key Management Service ()AWS KMS verwaltet werden.

SSE verschlüsselt Nachrichten, sobald sie bei Amazon SNS eingehen. Die Nachrichten werden in verschlüsselter Form gespeichert und erst entschlüsselt, wenn sie gesendet werden.

- Hinweise zur Verwaltung von SSE mithilfe von AWS Management Console oder AWS SDK für Java (durch Festlegen des `KmsMasterKeyId` Attributs mithilfe der `SetTopicAttributes` API-Aktionen `CreateTopic` und) finden Sie unterAmazon SNS SNS-Themenverschlüsselung mit serverseitiger Verschlüsselung einrichten.

- Informationen zum Erstellen verschlüsselter Themen mithilfe von AWS CloudFormation (durch Einstellen der `KmsMasterKeyId` Eigenschaft mithilfe der `AWS::SNS::Topic` Ressource) finden Sie im AWS CloudFormation Benutzerhandbuch.

> **Important**
>
> Alle Anfragen zu Themen mit aktiviertem SSE müssen HTTPS und Signature Version 4 verwenden.
> Weitere Informationen zur Kompatibilität anderer Dienste mit verschlüsselten Themen finden Sie in Ihrer Service-Dokumentation.
> Amazon SNS unterstützt nur KMS-Schlüssel mit symmetrischer Verschlüsselung.
> Sie können keinen anderen KMS-Schlüsseltyp für die Verschlüsselung Ihre Service-Ressourcen verwenden. Wie Sie feststellen, ob ein KMS-Schlüssel ein symmetrisch Verschlüsselungsschlüssel ist, erfahren Sie unter Erkennen asymmetrischer Schlüssel.

AWS KMS kombiniert sichere, hochverfügbare Hardware und Software zu einem für die Cloud skalierten Schlüsselverwaltungssystem. Wenn Sie Amazon SNS mit verwenden AWS KMS, werden die Datenschlüssel, die Ihre Nachrichtendaten verschlüsseln, ebenfalls verschlüsselt und zusammen mit den Daten gespeichert, die sie schützen.

Vorteile von AWS KMS:

- Sie können den AWS KMS key selbst erstellen und verwalten.
- Sie können auch AWS verwaltete KMS-Schlüssel für Amazon SNS verwenden, die für jedes Konto und jede Region einzigartig sind.
- Die AWS KMS Sicherheitsstandards können Ihnen dabei helfen, die Compliance-Anforderungen im Zusammenhang mit der Verschlüsselung zu erfüllen.

Weitere Informationen finden Sie unter Was ist? AWS Key Management Service im AWS Key Management Service Entwicklerhandbuch.

Verschlüsselungsumfang

SSE verschlüsselt den Nachrichtentext in einer Amazon SNS-Thema.

Folgendes wird von SSE nicht verschlüsselt:

- Themenmetadaten (Name und Attribute des Themas)
- Metadaten der Nachrichten (Betreff, ID, Zeitstempel und Attribute)
- Datenschutzrichtlinie
- Metriken nach Themen

> **Note**
>
> - Eine Nachricht ist nur dann verschlüsselt, wenn sie gesendet wurde, nachdem die Verschlüsselung einer Queue aktiviert wurde. Amazon SNS verschlüsselt keine Nachrichten, die sich bereits in der Queue befinden.
> - Verschlüsselte Nachrichten bleiben auch dann verschlüsselt, wenn die Verschlüsselung des Themas deaktiviert wird.

Wichtige Begriffe

Die folgenden wichtigen Begriffe vermitteln Ihnen ein besseres Verständnis für die Funktionalität von SSE. Detaillierte Beschreibungen finden Sie in derAmazon Simple Notification Service-API-Referenz.

Datenschlüssel

Der Datenverschlüsselungsschlüssel (Data Encryption Key, DEK), der dafür zuständig ist, den Inhalt von Amazon SNS-Nachrichten zu verschlüsseln.

Weitere Informationen finden Sie unter Datenschlüssel im AWS Key Management Service Entwicklerhandbuch und Envelope-Verschlüsselung im AWS Encryption SDK Entwicklerhandbuch.

AWS KMS key ID (ID)

Der Alias, Alias-ARN, die Schlüssel-ID oder der Schlüssel-ARN eines AWS KMS key oder eines benutzerdefinierten Kontos AWS KMS— in Ihrem Konto oder in einem anderen Konto. Während der Alias des AWS KMS für Amazon AWS verwalteten SNS immer gilt`alias/aws/sns`, AWS KMS kann der Alias eines Benutzernamens beispielsweise sein`alias/`*`MyAlias`*. Sie können diese AWS KMS -Schlüssel zum Schutz der Nachrichten in Amazon-SNS-Themen verwenden.

> **Note**
>
> Beachten Sie Folgendes:
>
> - Wenn Sie das erste Mal verwenden AWS Management Console , um das AWS verwaltete KMS für Amazon SNS für ein Thema anzugeben, AWS KMS wird das AWS verwaltete KMS für Amazon SNS erstellt.
> - Alternativ können Sie, wenn Sie die `Publish` Aktion zum ersten Mal für ein Thema mit aktivierter SSE verwenden, das AWS KMS AWS verwaltete KMS für Amazon SNS erstellen.

Sie können AWS KMS Schlüssel erstellen, die Richtlinien definieren, die steuern, wie AWS KMS Schlüssel verwendet werden können, und die AWS KMS Verwendung mithilfe des AWS KMS keysAbschnitts der AWS KMS Konsole oder der `CreateKey` AWS KMS Aktion überprüfen. Weitere Informationen finden Sie unter AWS KMS keys und Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch. Weitere Beispiele für AWS KMS Identifikatoren finden Sie KeyIdin der AWS Key Management Service API-Referenz. Informationen zum Suchen von AWS KMS Identifikatoren finden Sie unter Suchen der Schlüssel-ID und des ARN im AWS Key Management Service Entwicklerhandbuch.

> **Important**
>
> Für die Nutzung AWS KMS fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter Schätzung der Kosten AWS KMS und Preise zu AWS Key Management Service.

Verwaltung von Amazon SNS SNS-Verschlüsselungsschlüsseln und Kosten

Die folgenden Abschnitte enthalten Informationen zum Arbeiten mit Schlüsseln, die in AWS Key Management Service (AWS KMS) verwaltet werden.

> **Note**
>
> Amazon SNS unterstützt nur KMS-Schlüssel mit symmetrischer Verschlüsselung.
> Sie können keinen anderen KMS-Schlüsseltyp für die Verschlüsselung Ihre Service-

Ressourcen verwenden. Wie Sie feststellen, ob ein KMS-Schlüssel ein symmetrisch Verschlüsselungsschlüssel ist, erfahren Sie unter Erkennen asymmetrischer Schlüssel.

Schätzung der Kosten AWS KMS

Um Kosten vorherzusagen und Ihre AWS Rechnung besser zu verstehen, möchten Sie vielleicht wissen, wie oft Amazon SNS Ihre AWS KMS key verwendet.

> **Note**
>
> Mit der nachstehenden Formel erhalten Sie eine gute Vorstellung davon, welche Kosten auf Sie zukommen. Allerdings können die tatsächlichen Kosten aufgrund der verteilten Struktur von Amazon SNS höher liegen.

Zur Berechnung der Anzahl der API-Anfragen (`R`) pro Thema verwenden Sie folgende Formel:

```
R = B / D * (2 * P)
```

`B` ist der Abrechnungszeitraum (in Sekunden).

`D` ist die Wiederverwendungszeit des Datenschlüssels (in Sekunden—Amazon SNS verwendet einen Datenschlüssel bis zu 5 Minuten lang).

`P` ist die Anzahl der Veröffentlichungen der Prinzipale, die Nachrichten an das Amazon-SNS-Thema senden.

Es folgen Beispielberechnungen: Preisinformationen finden Sie unter AWS Key Management Service -Preise.

Beispiel 1: Berechnung der Anzahl der AWS KMS API-Aufrufe für einen Herausgeber und ein Thema

In diesem Beispiel wird Folgendes angenommen:

- Der Abrechnungszeitraum ist 1. bis 31. Januar (2 678 400 Sekunden).
- Der Wiederverwendungszeitraum für den Datenschlüssel ist 5 Minuten (300 Sekunden).
- Es gibt 1 Thema.
- Es gibt 1 Veröffentlichungsprinzipal.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Beispiel 2: Berechnung der Anzahl der AWS KMS -API-Aufrufe für mehrere Herausgeber und 2 Themen

In diesem Beispiel wird Folgendes angenommen:

- Der Abrechnungszeitraum ist 1. bis 28. Februar (2 419 200 Sekunden).
- Der Wiederverwendungszeitraum für den Datenschlüssel ist 5 Minuten (300 Sekunden).
- Es gibt 2 Themen.
- Das erste Thema hat 3 Veröffentlichungsprinzipale.
- Das zweite Thema hat 5 Veröffentlichungsprinzipale.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Berechtigungen konfigurieren AWS KMS

Bevor Sie SSE verwenden können, müssen Sie AWS KMS key Richtlinien konfigurieren, die die Verschlüsselung von Themen sowie die Verschlüsselung und Entschlüsselung von Nachrichten ermöglichen. Weitere Informationen zu Berechtigungen für AWS KMS -Aktionen finden Sie unter AWS KMS Glue-API-Berechtigungen: Referenz für Aktionen und Ressourcen im AWS Key Management Service Entwicklerhandbuch. Einzelheiten zur Einrichtung eines Amazon-SNS-Themas mit serverseitiger Verschlüsselung finden Sie unter Zusätzliche Informationen.

> **Note**
>
> Sie können Berechtigungen für symmetrische KMS-Verschlüsselungsschlüssel auch mit IAM-Richtlinien verwalten. Weitere Informationen finden Sie unter Verwenden von IAM-Richtlinien mit. AWS KMS
> Sie können zwar globale Berechtigungen für das Senden an und Empfangen von Amazon SNS konfigurieren, AWS KMS erfordert jedoch die ausdrückliche Benennung des KMSs vollständigen ARN bestimmter Regionen im `Resource` Abschnitt einer IAM-Richtlinie.

Sie müssen außerdem sicherstellen, dass die wichtigsten Richtlinien von die erforderlichen AWS KMS key Berechtigungen zulassen. Geben Sie dazu die Namen der Prinzipale an, die

verschlüsselte Nachrichten in Amazon SNS als Benutzer in der KMS-Schlüsselrichtlinie produzieren und verbrauchen.

Alternativ können Sie die erforderlichen AWS KMS Aktionen und den KMS-ARN in einer IAM-Richtlinie angeben, die den Prinzipalen zugewiesen ist, die verschlüsselte Nachrichten in Amazon SNS veröffentlichen und abonnieren. Weitere Informationen finden Sie unter Verwalten des Zugriffs auf AWS KMS im AWS Key Management Service -Entwicklerhandbuch.

Wenn Sie einen vom Kunden verwalteten Schlüssel für Ihr Amazon-SNS-Thema auswählen und Sie Aliase verwenden, um den Zugriff auf KMS-Schlüssel mithilfe von IAM-Richtlinien oder KMS-Schlüsselrichtlinien mit dem Bedingungsschlüssel `kms:ResourceAliases` zu steuern, stellen Sie sicher, dass dem ausgewählten vom Kunden verwalteten Schlüssel auch ein Alias zugeordnet ist. Weitere Informationen zur Verwendung von Aliasen zur Steuerung des Zugriffs auf KMS-Schlüssel finden Sie unter Verwenden von Aliasen zur Steuerung des Zugriffs auf KMS-Schlüssel im Entwicklerhandbuch zu AWS Key Management Service .

Einem Benutzer erlauben, Nachrichten an ein bestimmtes Thema mit SSE zu senden

Der Publisher muss die Berechtigungen `kms:GenerateDataKey*` und `kms:Decrypt` für den AWS KMS key besitzen.

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

Aktivieren Sie die Kompatibilität zwischen Ereignisquellen aus AWS Diensten und verschlüsselten Themen

Verschiedene AWS Dienste veröffentlichen Veranstaltungen zu Amazon SNS SNS-Themen. Damit diese Ereignisquellen mit verschlüsselten Themen arbeiten können, müssen Sie die folgenden Schritte ausführen:

1. Verwenden Sie einen vom Kunden verwalteten Schlüssel. Weitere Informationen finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch.
2. Damit der AWS Service über die `kms:Decrypt` Berechtigungen `kms:GenerateDataKey*` und verfügt, fügen Sie der KMS-Richtlinie die folgende Erklärung hinzu.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Ereignisquelle	Dienstauftraggeber
Amazon CloudWatch	`cloudwatch.amazonaws.com`
CloudWatch Amazon-Veranstaltungen	`events.amazonaws.com`
AWS CodeCommit	`codecommit.amazonaws.com`
AWS Database Migration Service	`dms.amazonaws.com`
AWS Directory Service	`ds.amazonaws.com`
Amazon-DynamoDB	`dynamodb.amazonaws.com`

Ereignisquelle	Dienstauftraggeber
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`
Amazon Simple Email Service	`ses.amazonaws.com`
Amazon Simple Storage Service	`s3.amazonaws.com`
AWS Snowball Edge	`importexport.amazonaws.com`
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager besteht aus zwei Serviceprinzipien: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Bei einigen Amazon SNS SNS-Ereignisquellen müssen Sie in der Richtlinie eine IAM-Rolle (und nicht den Service Principal) angeben: AWS KMS key
>
> - Amazon EC2 Auto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. Fügen Sie die `aws:SourceAccount`- und `aws:SourceArn`-Bedingungsschlüssel der KMS-Ressourcenrichtlinie hinzu, um den KMS-Schlüssel vor Confused-Deputy-Angriffen zu schützen.

Genaue Einzelheiten zu jedem Fall finden Sie in der jeweiligen Liste der servicespezifischen Dokumentation (oben).

> **Important**
>
> Das Hinzufügen von `aws:SourceAccountaws:SourceArn`, und `aws:SourceOrgID` zu einer AWS KMS Richtlinie wird für EventBridge-to-encrypted Themen nicht unterstützt.

```
{
  "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. Aktivieren Sie SSE für Ihr Thema mithilfe Ihres KMS.
5. Stellen Sie der Ereignisquelle den ARN des verschlüsselten Themas zur Verfügung.

AWS KMS Fehler

Wenn Sie mit Amazon SNS und arbeiten AWS KMS, können Fehler auftreten. Die folgende Liste beschreibt die Fehler und möglichen Lösungen.

KMSAccessDeniedException

Der Verschlüsselungstext verweist auf einen Schlüssel, der nicht vorhanden ist oder auf den Sie keinen Zugriff haben.

HTTP Status Code: 400

KMSDisabledAusnahme

Die Anforderung wurde abgelehnt, da der angegebene KMS nicht aktiviert ist.

HTTP Status Code: 400

KMSInvalidStateException

Die Anforderung wurde abgelehnt, da der Status der angegebenen Ressource für diese Anforderung ungültig ist. Weitere Informationen finden Sie unter Schlüsselstatus von AWS KMS keys im AWS Key Management Service -Entwicklerhandbuch.

HTTP Status Code: 400

KMSNotFoundException

Die Anforderung wurde abgelehnt, da die angegebene Entity oder Ressource nicht gefunden wurde.

HTTP Status Code: 400

KMSOptInRequired

Für die AWS Zugriffsschlüssel-ID ist ein Abonnement für den Dienst erforderlich.

HTTP Status Code: 403

KMSThrottlingAusnahme

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt. Weitere Informationen zur Drosselung finden Sie unter Kontingente im AWS Key Management Service -Entwicklerhandbuch.

HTTP Status Code: 400

Amazon SNS SNS-Themenverschlüsselung mit serverseitiger Verschlüsselung einrichten

Amazon SNS unterstützt serverseitige Verschlüsselung (SSE), um den Inhalt von Nachrichten mit AWS Key Management Service (AWS KMS) zu schützen. Folgen Sie den nachstehenden Anweisungen, um SSE über die Amazon SNS SNS-Konsole oder das CDK zu aktivieren.

Option 1: Aktivieren Sie die Verschlüsselung mit dem AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Navigieren Sie zur Themenseite, wählen Sie Ihr Thema aus und klicken Sie auf Bearbeiten.
3. Erweitern Sie den Abschnitt Encryption (Verschlüsselung) und gehen Sie wie folgt vor:
   - Schalten Sie die Verschlüsselung auf Aktivieren um.
   - Wählen Sie das verwaltete AWS SNS Schlüssel (alias/aws/sns) als Verschlüsselungsschlüssel. Diese Option ist standardmäßig aktiviert.
4. Wählen Sie Änderungen speichern.

> **Note**
>
> - Der Von AWS verwalteter Schlüssel wird automatisch erstellt, falls er noch nicht existiert.
> - Wenn Sie den Schlüssel nicht sehen oder nicht über ausreichende Berechtigungen verfügen, fragen Sie Ihren Administrator nach `kms:ListAliases` und`kms:DescribeKey`.

Option 2: Aktivieren Sie die Verschlüsselung mit AWS CDK

Um das verwaltete zu verwenden AWS SNS Geben Sie in Ihrer CDK-Anwendung den folgenden Codeausschnitt ein:

```
import software.amazon.awscdk.services.sns.*;
import software.amazon.awscdk.services.kms.*;
import software.amazon.awscdk.core.*;

public class SnsEncryptionExample extends Stack {
    public SnsEncryptionExample(final Construct scope, final String id) {
```

```
        super(scope, id);

        // Define the managed SNS key
        IKey snsKey = Alias.fromAliasName(this, "helloKey", "alias/aws/sns");

        // Create the SNS Topic with encryption enabled
        Topic.Builder.create(this, "MyEncryptedTopic")
            .masterKey(snsKey)
            .build();
    }
}
```

Zusätzliche Informationen

- Benutzerdefinierter KMS-Schlüssel — Sie können bei Bedarf einen benutzerdefinierten Schlüssel angeben. Wählen Sie in der Amazon SNS SNS-Konsole Ihren benutzerdefinierten KMS-Schlüssel aus der Liste aus oder geben Sie den ARN ein.
- Berechtigungen für benutzerdefinierte KMS-Schlüssel — Wenn Sie einen benutzerdefinierten KMS-Schlüssel verwenden, nehmen Sie Folgendes in die Schlüsselrichtlinie auf, damit Amazon SNS Nachrichten ver- und entschlüsseln kann:

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.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"
        }
    }
```

```
}
```

Auswirkungen auf die Verbraucher

Die Aktivierung von SSE ändert nichts daran, wie Abonnenten Nachrichten konsumieren. AWS verwaltet die Verschlüsselung und Entschlüsselung auf transparente Weise. Nachrichten bleiben im Ruhezustand verschlüsselt und werden vor der Zustellung an Abonnenten automatisch entschlüsselt. Für optimale Sicherheit AWS empfiehlt es sich, HTTPS für alle Endpunkte zu aktivieren, um eine sichere Übertragung von Nachrichten zu gewährleisten.

Amazon SNS SNS-Themenverschlüsselung mit verschlüsseltem Amazon SQS SQS-Warteschlangenabonnement einrichten

Sie können die serverseitige Verschlüsselung (SSE) für ein Thema aktivieren, um die entsprechenden Daten zu schützen. Damit Amazon SNS Nachrichten an verschlüsselte Amazon-SQS-Warteschlangen senden kann, muss der mit der Amazon-SQS-Warteschlange verknüpfte verwaltete Schlüssel über eine Richtlinienanweisung verfügen, die dem Amazon-SNS-Dienstprinzipal Zugriff auf die AWS KMS -API-Aktionen `GenerateDataKey` und `Decrypt` gewährt. Weitere Informationen zur Verwendung von der SSE finden Sie unter Sicherung von Amazon SNS SNS-Daten mit serverseitiger Verschlüsselung.

In diesem Thema wird erklärt, wie Sie SSE für ein Amazon SNS SNS-Thema mit einem verschlüsselten Amazon SQS SQS-Warteschlangenabonnement mithilfe von aktivieren. AWS Management Console

Schritt 1: Erstellen eines benutzerdefinierten KMS-Schlüssels

1. Melden Sie sich bei der AWS KMS -Konsole mit einem Benutzer an, der mindestens über die `AWSKeyManagementServicePowerUser`-Richtlinie verfügt.
2. Klicken Sie auf Create a key.
3. Um einen KMS-Schlüssel mit symmetrischer Verschlüsselung zu erstellen, wählen Sie für Key type (Schlüsseltyp) die Option Symmetric (Symmetrisch) aus.

   Weitere Informationen zum Erstellen eines asymmetrischen KMS-Schlüssels in der AWS KMS -Konsole finden Sie unter Erstellen asymmetrischer KMS-Schlüssel (Konsole).
4. Unter Key usage (Schlüsselverwendung) ist die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) für Sie ausgewählt.

Weitere Informationen wie Sie KMS-Schlüssel erstellen, die MAC-Codes generieren und überprüfen, finden Sie unter Erstellen von HMAC-KMS-Schlüssel.

Weitere Hinweise zu den Erweiterten Optionen finden Sie unter Schlüssel für spezielle Zwecke.

5. Wählen Sie Weiter.
6. Geben Sie einen Alias für den Replikatschlüssel ein. Der Aliasname darf nicht mit **`aws/`** beginnen. Das **`aws/`** Präfix ist von Amazon Web Services reserviert, um es Von AWS verwaltete Schlüssel in Ihrem Konto darzustellen.

> **Note**
>
> Wenn Sie einen Alias hinzufügen, löschen oder aktualisieren, wird dadurch möglicherweise eine Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Einzelheiten finden Sie unter ABAC für AWS KMS und Verwenden von Aliassen zur Steuerung des Zugriffs auf KMS-Schlüssel.

Ein Alias ist ein Anzeigename, den Sie verwenden können, um einen KMS-Schlüssel zu identifizieren. Wir empfehlen, dass Sie einen Alias wählen, der auf die Art von Daten, die Sie schützen möchten, oder die Anwendung, die Sie mit dem KMS-Schlüssel verwenden möchten, hindeutet.

Zum Erstellen eines KMS-Schlüssels in der Konsole benötigen Sie Aliase AWS Management Console. Sie sind optional, wenn Sie den CreateKeyVorgang verwenden.

7. (Optional) Geben Sie eine Beschreibung für den KMS-Schlüssel ein.

   Sie können jetzt eine Beschreibung hinzufügen oder sie jederzeit aktualisieren, es sei denn, der Schlüsselstatus lautet `Pending Deletion` oder `Pending Replica Deletion`. Um die Beschreibung eines vorhandenen, vom Kunden verwalteten Schlüssels hinzuzufügen, zu ändern oder zu löschen, bearbeiten Sie die Beschreibung im AWS Management Console oder verwenden Sie den UpdateKeyDescriptionVorgang.
8. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Add tag (Tag hinzufügen), wenn Sie mehr als ein Tag zum KMS-Schlüssel hinzufügen möchten.

> **Note**
>
> Wenn Sie einen KMS-Schlüssel markieren oder entmarkieren, wird dadurch möglicherweise die Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Einzelheiten finden Sie unter ABAC für AWS KMS und Verwenden von Tags zur Steuerung des Zugriffs auf KMS-Schlüssel.

Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Markieren von Schlüsseln und ABAC für AWS KMS.

9. Wählen Sie Weiter.
10. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel verwalten können.

> **Note**
>
> Diese Schlüsselrichtlinie ermöglicht die AWS-Konto vollständige Kontrolle über diesen KMS-Schlüssel. Kontoadministratoren können damit anderen Prinzipalen mithilfe von IAM-Richtlinien die Berechtigung zum Verwalten des KMS-Schlüssels erteilen. Details dazu finden Sie unter Standardschlüsselrichtlinie.
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

11. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
12. Wählen Sie Weiter.
13. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können. Wählen Sie Weiter.

14. Fügen Sie auf der Seite Review and edit key policy (Schlüsselrichtlinie überprüfen und bearbeiten) die folgende Anweisung zur Schlüsselrichtlinie hinzu. Wählen Sie anschließend Finish (Fertig stellen) aus.

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

Ihr neues benutzerdefinierter kundenverwalteter Schlüssel wird in der Liste der Schlüssel angezeigt.

Schritt 2: Erstellen eines verschlüsselten Amazon SNS-Themas

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie Thema erstellen aus.
4. Geben Sie auf der Seite Create new topic (Neues Thema erstellen) in Topic name (Name des Themas) einen Namen für das Thema ein (z. B. `MyEncryptedTopic`). Wählen Sie anschließend Create topic (Thema erstellen) aus.
5. Erweitern Sie den Abschnitt Encryption (Verschlüsselung) und gehen Sie wie folgt vor:

   a. Wählen Sie Enable server-side encryption.

   b. Erstellen des kundenverwalteten Schlüssels. Weitere Informationen finden Sie unter Wichtige Begriffe.

      Für jeden kundenverwalteten Schlüsseltyp werden die Beschreibung, das Konto und der ARN des kundenverwalteten Schlüssels angezeigt.

> **Important**
>
> Wenn Sie nicht der Besitzer des kundenverwalteten Schlüssels sind oder wenn Sie sich mit einem Konto anmelden, das über keine `kms:ListAliases`- und `kms:DescribeKey`-Berechtigungen verfügt, können Sie auf der Amazon-SNS-Konsole keine Informationen über den kundenverwalteten Schlüssel aufrufen. Bitten Sie den Inhaber des kundenverwalteten Schlüssels, Ihnen diese Berechtigungen zu erteilen. Beispiele und weitere Informationen zu AWS KMS - Berechtigungen finden Sie unter API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen im AWS Key Management Service Benutzerhandbuch.

c. Wählen Sie für den vom Kunden verwalteten Schlüssel den Schlüssel aus MyCustomKey, den Sie zuvor erstellt haben, und wählen Sie dann Serverseitige Verschlüsselung aktivieren aus.

6. Wählen Sie Änderungen speichern.

   SSE ist für Ihr Thema aktiviert und die MyTopicSeite wird angezeigt.

   Auf der Registerkarte Verschlüsselung werden der Verschlüsselungsstatus des Themas, das AWS -Konto, der kundenverwaltete Schlüssel, der CMK-ARN und die Beschreibung angezeigt.

Ihr neues verschlüsseltes Thema wird in der Themenliste angezeigt.

Schritt 3: Erstellen und Abonnieren verschlüsselter Amazon-SQS-Warteschlangen

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Klicken Sie auf Create New Queue (Neue Queue erstellen).
3. Führen Sie auf der Seite Create New Queue (Neue Queue erstellen) die folgenden Schritte aus:

   a. Geben Sie einen Queue Name (Queuesname) ein (z. B. `MyEncryptedQueue1`).

   b. Klicken Sie auf Standard Queue (StandardQueue) und wählen Sie dann Configure Queue (Queue konfigurieren) aus.

   c. Wählen Sie Use SSE.

   d. Wählen Sie für AWS KMS key, MyCustomKeywelches Sie zuvor erstellt haben, und wählen Sie dann Warteschlange erstellen aus.

4. Wiederholen Sie den Vorgang, um eine zweite Queue (z. B. namens) zu erstellen. `MyEncryptedQueue2`).

   Ihre neuen verschlüsselten Queues werden in der Liste der Queues angezeigt.

5. Wählen Sie in der Amazon SQS-Konsole `MyEncryptedQueue1` und `MyEncryptedQueue2` aus. Wählen Sie anschließend (Queuesaktionen) und Subscribe Queues to SNS Topic (Abonniere SNS-Thema für Queues erstellen) aus.
6. Wählen Sie im Dialogfeld „Thema abonnieren“ die Option Thema auswählen aus und wählen Sie MyEncryptedTopicdann Abonnieren aus.

   Die Abonnements Ihrer verschlüsselten Queues für Ihr verschlüsseltes Thema werden im Dialogfeld Topic Subscription Result (Ergebnis für Thema-Abonnement angezeigt.

7. Wählen Sie OK aus.

Schritt 4: Veröffentlichen einer Nachricht in Ihrem verschlüsselten Thema

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie aus der Themenliste die Option Nachricht veröffentlichen aus MyEncryptedTopicund klicken Sie dann auf Nachricht veröffentlichen.
4. Führen Sie auf der Seite Publish a message (Ziel auswählen) folgende Schritte durch:

   a. (Optional) Geben Sie im Abschnitt Message details (Nachrichtendetails) den Subject (Betreff) ein (z. B. `Testing message publishing`).

   b. Geben Sie im Abschnitt Message body (Nachrichtentext) den Nachrichtentext ein (z. B. `My message body is encrypted at rest.`).

   c. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

Ihre Nachricht wird in Ihren abonnierten verschlüsselten Queues veröffentlicht.

Schritt 5: Überprüfen der Zustellung Ihrer Nachrichten

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Wählen Sie aus der Liste der Warteschlangen MyEncryptedQueue1 aus und wählen Sie dann Nachrichten senden und empfangen aus.

3. Wählen Sie auf der Seite Nachrichten senden und empfangen in MyEncryptedQueue 1 die Option Umfrage für Nachrichten aus.

   Die Nachricht die Sie zuvor gesendet haben wird angezeigt.
4. Wählen Sie Weitere Details, um Ihre Nachricht anzuzeigen.
5. Wählen Sie Close, wenn Sie fertig sind.
6. Wiederholen Sie den Vorgang für MyEncryptedQueue2.

Sicherung des Amazon SNS SNS-Datenverkehrs mit VPC-Endpunkten

Ein Amazon Virtual Private Cloud (Amazon VPC)-Endpunkt für Amazon SNS ist eine logische Einheit innerhalb einer VPC, die nur Konnektivität mit Amazon SNS ermöglicht. Die VPC leitet Anforderungen an Amazon SNS weiter und Antworten an die VPC zurück. In den folgenden Abschnitten finden Sie Informationen zum Arbeiten mit VPC-Endpunkten und zum Erstellen von VPC-Endpunktrichtlinien.

Wenn Sie Amazon Virtual Private Cloud (Amazon VPC) zum Hosten Ihrer AWS Ressourcen verwenden, können Sie eine private Verbindung zwischen Ihrer VPC und Amazon SNS herstellen. Mit dieser Verbindung können Sie Nachrichten in Ihren Amazon SNS-Themen veröffentlichen, ohne sie über das öffentliche Internet senden zu müssen.

Amazon VPC ist ein AWS Service, mit dem Sie AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk starten können. Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways. Zum Herstellen einer Verbindung Ihrer VPC mit Amazon SNS definieren Sie einen Schnittstellen-VPC-Endpunkt. Dieser Endpunkttyp ermöglicht es Ihnen, Ihre VPC mit AWS Diensten zu verbinden. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu Amazon SNS, ohne dass ein Internet-Gateway, eine NAT-Instance (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen finden Sie unter Zugreifen und AWS-Service Verwenden eines VPC-Endpunkts mit einer Schnittstelle im Amazon VPC-Benutzerhandbuch.

Die Informationen in diesem Abschnitt sind für Benutzer von Amazon VPC vorgesehen. Weitere Informationen und Informationen zu den ersten Schritten beim Erstellen einer VPC finden Sie unter Planen Ihrer VPC im Amazon VPC-Benutzerhandbuch.

> **Note**
>
> VPC-Endpunkte erlauben es nicht, ein Amazon SNS-Thema an eine private IP-Adresse zu abonnieren.

Erstellen eines Amazon VPC Endpunkts für Amazon SNS

Zum Veröffentlichen von Nachrichten in Ihren Amazon SNS-Themen von einer Amazon VPC aus, erstellen Sie einen VPC-Endpunkt. Anschließend können Sie Nachrichten in Ihren Themen veröffentlichen und gleichzeitig den Datenverkehr innerhalb des Netzwerks behalten, das Sie mit der VPC verwalten.

Verwenden Sie die folgenden Informationen, um den Endpunkt zu erstellen und die Verbindung zwischen Ihrer VPC und Amazon SNS zu testen. Außerdem finden Sie ein Tutorial, mit dem Sie von Grund auf neu starten, unter Veröffentlichen einer Amazon SNS Nachricht aus Amazon VPC.

Erstellen des Endpunkts

Sie können einen Amazon SNS SNS-Endpunkt in Ihrer VPC mit dem AWS Management Console, dem, einem AWS SDK AWS CLI, der Amazon SNS SNS-API oder erstellen. AWS CloudFormation

Informationen zum Erstellen und Konfigurieren eines Endpunkts über die Amazon-VPC-Konsole oder die AWS CLI finden Sie unter Creating an Interface Endpoint (Erstellen eines Schnittstellenendpunkts) im Amazon-VPC-Benutzerhandbuch.

> **Important**
>
> Sie können Amazon Virtual Private Cloud nur mit HTTPS-Amazon-SNS-Endpunkten verwenden.
> Wenn Sie einen Endpunkt erstellen, geben Sie an, dass Amazon SNS der Service ist, mit dem Ihre VPC eine Verbindung einrichten soll. In der Amazon VPC-Konsole variieren die Servicenamen abhängig von der ausgewählten Region. Wenn Sie beispielsweise USA Ost (Nord-Virginia) wählen, lautet der Servicename com.amazonaws.us-east-1.states.
> Wenn Sie Amazon SNS konfigurieren, um Nachrichten von Amazon VPC zu senden, müssen Sie privates DNS aktivieren und Endpunkte im `sns.`*`us-east-2`*`.amazonaws.com`-Format angeben.

Ein privates DNS unterstützt keine Legacy-Endpunkte wie z. B. `queue.amazonaws.com` oder `us-east-2.queue.amazonaws.com`.

Informationen zum Erstellen und Konfigurieren eines Endpunkts mithilfe AWS CloudFormation von finden Sie in der *`AWS::EC2::VPCEndpoint`*Ressource im AWS CloudFormation Benutzerhandbuch.

Testen der Verbindung zwischen Ihrer VPC und Amazon SNS

Nachdem Sie einen Endpunkt für Amazon SNS erstellt haben, können Sie Nachrichten von Ihrer VPC zu Ihrem Amazon SNS-Thema senden. Zum Testen dieser Verbindung gehen Sie wie folgt vor:

1. Stellen Sie eine Connect zu einer EC2 Amazon-Instance her, die sich in Ihrer VPC befindet. Informationen zum Herstellen einer Verbindung finden Sie unter Connect to Your Linux Instance oder Connecting to Your Windows Instance in der EC2 Amazon-Dokumentation.

   Um Beispielsweise eine Verbindung mit einer Linux-Instance über einen SSH-Client herzustellen, führen Sie von einem Terminal aus den folgenden Befehl aus:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Wobei gilt:

   - ec2-key-pair.pem ist die Datei, die das key pair enthält, das Amazon bei der Erstellung der Instance bereitgestellt hat. EC2
   - instance-hostname ist der öffentliche Hostname der Instance. Um den Hostnamen in der EC2Amazon-Konsole abzurufen: Wählen Sie Instances, wählen Sie Ihre Instance aus und suchen Sie den Wert für Public DNS.
2. Verwenden Sie von Ihrer Instance aus Amazon SNS-Befehl `publish` in der AWS CLI. Mit dem folgenden Befehl können eine einfache Nachricht für ein Thema senden:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   Wobei gilt:

   - aws-region ist die AWS Region, in der sich das Thema befindet.

- sns-topic-arnist der Amazon-Ressourcenname (ARN) des Themas. So erhalten Sie den ARN von der Amazon-SNS-Konsole: Wählen Sie Topics (Themen), suchen Sie Ihr Thema und verwenden Sie den Wert in der Spalte ARN.

Wenn die Nachricht vom Amazon SNS erfolgreich empfangen wurde, gibt das Terminal eine Nachrichten-ID wie die folgende aus:

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Erstellen einer Amazon-VPC-Endpunktrichtlinie für Amazon SNS

Sie können eine Richtlinie für Amazon VPC Endpunkte für Amazon SNS erstellen, in der Sie Folgendes angeben:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon VPC User Guide.

Im folgenden Beispiel für eine VPC-Endpunktrichtlinie wird angegeben, dass der IAM-Benutzer`MyUser` zum Amazon SNS-Thema `MyTopic` veröffentlichen darf.

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

Folgendes wird abgelehnt:

- Andere Amazon-SNS-API-Aktionen, z. B. `sns:Subscribe` und `sns:Unsubscribe`.
- Andere IAM-Benutzer und -Regeln, die versuchen, diesen VPC-Endpunkt zu verwenden.
- `MyUser`-Veröffentlichungen in einem anderen Amazon SNS Amazon SNS-Thema.

> **Note**
>
> Der IAM-Benutzer kann nach wie vor andere Amazon SNS-API-Aktionen von außerhalb der VPC verwenden.

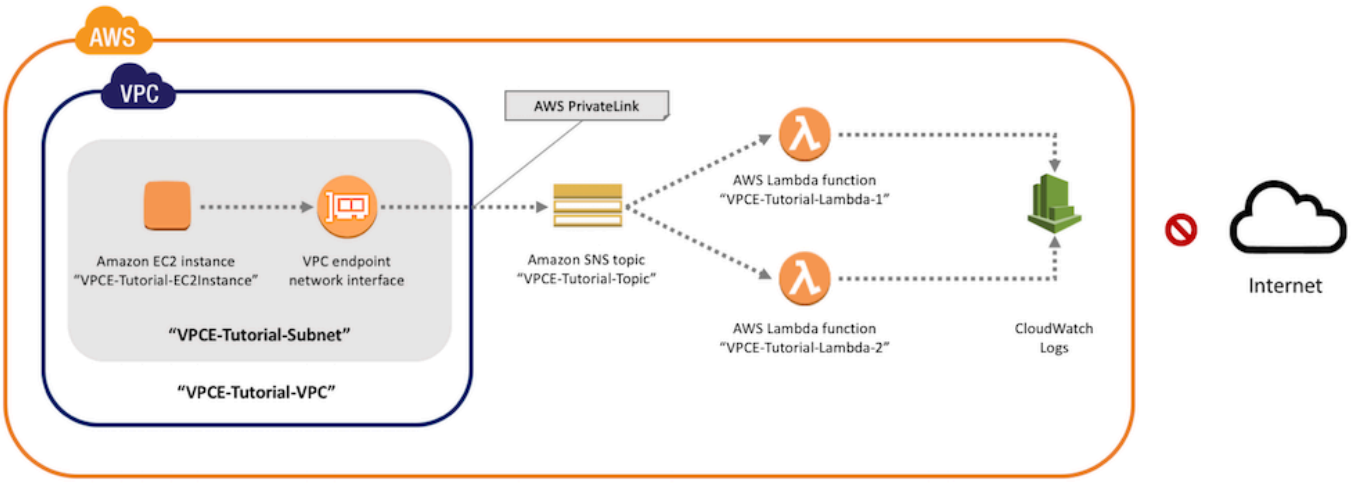
Veröffentlichen einer Amazon SNS Nachricht aus Amazon VPC

In diesem Abschnitt erfahren Sie, wie Sie ein Amazon SNS Amazon SNS-Thema veröffentlichen und gleichzeitig die Nachrichten in einem privaten Netzwerk geschützt halten. Sie veröffentlichen eine Nachricht von einer EC2 Amazon-Instance, die in Amazon Virtual Private Cloud (Amazon VPC) gehostet wird. Die Nachricht bleibt im AWS Netzwerk, ohne über das öffentliche Internet zu gelangen. Durch die private Veröffentlichung von Nachrichten von einer VPC aus können Sie die Sicherheit des Datenverkehrs zwischen Ihren Anwendungen und Amazon SNS verbessern. Diese Sicherheit ist wichtig, wenn Sie personenbezogene Informationen (PII) über Ihre Kunden veröffentlichen oder wenn Ihre Anwendung Marktregulierungen unterliegt. Beispiel: Die private Veröffentlichung ist hilfreich bei einem Gesundheitssystem, das den Health Insurance Portability and Accountability Act (HIPAA) erfüllen muss, oder bei einem Finanzsystem, das den Payment Card Industry Data Security Standard (PCI DSS) erfüllen muss.

Gehen Sie wie folgt vor:

- Verwenden Sie eine AWS CloudFormation Vorlage, um automatisch ein temporäres privates Netzwerk in Ihrem zu erstellen AWS-Konto.
- Erstellen Sie einen VPC-Endpunkt, der die VPC mit Amazon SNS verbindet.
- Melden Sie sich bei einer EC2 Amazon-Instance an und veröffentlichen Sie eine Nachricht privat zu einem Amazon SNS-Thema.
- Vergewissern Sie sich, dass die Nachricht erfolgreich zugestellt wurde.
- Löschen Sie die Ressourcen, die Sie während dieses Vorgangs erstellt haben, damit sie nicht in Ihren AWS-Konto Ressourcen verbleiben.

Das folgende Diagramm zeigt das private Netzwerk, das Sie in Ihrem AWS Konto erstellen, wenn Sie diese Schritte ausführen:



Dieses Netzwerk besteht aus einer VPC, die eine EC2 Amazon-Instance enthält. Die Instance stellt über einen Schnittstellen-VPC-Endpunkt eine Verbindung mit Amazon SNS her. Dieser Endpunkttyp stellt eine Verbindung zu Diensten her, die von AWS PrivateLink betrieben werden. Wenn diese Verbindung hergestellt ist, können Sie sich bei der EC2 Amazon-Instance anmelden und Nachrichten zum Amazon SNS-Thema veröffentlichen, obwohl das Netzwerk vom öffentlichen Internet getrennt ist. Das Thema verteilt die Nachrichten, die es empfängt, auf zwei AWS Lambda Abonnementfunktionen. Diese Funktionen protokollieren die Nachrichten, die sie erhalten, in Amazon CloudWatch Logs.

Es dauert ungefähr 20 Minuten, bis diese Schritte fertig ausgeführt sind.

Themen

- Bevor Sie beginnen
- Schritt 1: Erstellen Sie ein EC2 Amazon-Schlüsselpaar
- Schritt 2: Erstellen Sie die AWS Ressourcen
- Schritt 3: Stellen Sie sicher, dass Ihre EC2 Amazon-Instance keinen Internetzugang hat
- Schritt 4: Erstellen eines Amazon VPC-Endpunkts für Amazon SNS
- Schritt 5: Veröffentlichen einer Nachricht in Ihrem Amazon SNS-Thema
- Schritt 6: Überprüfen Sie die Zustellung Ihrer Nachrichten
- Schritt 7: Bereinigen
- Zugehörige Ressourcen

Bevor Sie beginnen

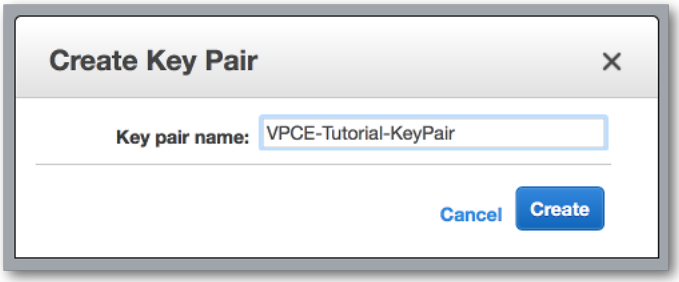
Bevor Sie beginnen, benötigen Sie einen Amazon Web Services (AWS)-Konto. Wenn Sie sich registrieren, wird Ihr Konto automatisch für alle Dienste angemeldet AWS, einschließlich Amazon SNS und Amazon VPC. Wenn Sie nicht bereits ein Konto erstellt haben, wechseln Sie zu https://aws.amazon.com/ und wählen Sie Create a Free Account (Kostenloses Konto erstellen) aus.

Schritt 1: Erstellen Sie ein EC2 Amazon-Schlüsselpaar

Ein key pair wird verwendet, um sich bei einer EC2 Amazon-Instance anzumelden. Es besteht aus einem öffentlichen Schlüssel zum Verschlüsseln Ihrer Anmeldeinformationen und einem privaten Schlüssel zum Entschlüsseln. Wenn Sie ein Schlüsselpaar erstellen, laden eine Kopie des privaten Schlüssels herunter. Später verwenden Sie das key pair, um sich bei einer EC2 Amazon-Instance anzumelden. Zum Anmelden geben Sie den Namen des Schlüsselpaares und den privaten Schlüssel an.

So erstellen Sie ein Schlüsselpaar

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Suchen Sie im Navigationsmenü auf der linken Seite den Abschnitt Network & Security (Netzwerk und Sicherheit). Wählen Sie dann Key Pairs (Schlüsselpaare).
3. Wählen Sie Create Key Pair aus.
4. Geben Sie im Fenster Create Key Pair (Schlüsselpaar erstellen) für Key pair name (Schlüsselpaar-Name) den Wert **`VPCE-Tutorial-KeyPair`** ein. Wählen Sie dann die Option Erstellen.



5. Die private Schlüsseldatei wird von Ihrem Browser automatisch runtergeladen. Bewahren Sie es an einem sicheren Ort auf. Amazon EC2 gibt der Datei eine Erweiterung von `.pem`.

6. (Optional) Wenn Sie einen SSH-Client auf Ihrem Mac- oder Linux-Computer verwenden, um eine Verbindung mit der Instance herzustellen, legen Sie die Berechtigungen für die private Schlüsseldatei mit dem Befehl `chmod` fest, sodass nur Sie diese lesen können:

   a. Öffnen Sie ein Terminal und navigieren Sie zu dem Verzeichnis, das den privaten Schlüssel enthält:

   ```
   $ cd /filepath_to_private_key/
   ```

   b. Legen Sie mit folgendem Befehl die Berechtigungen fest:

   ```
   $ chmod 400 VPCE-Tutorial-KeyPair.pem
   ```

Schritt 2: Erstellen Sie die AWS Ressourcen

Um die Infrastruktur einzurichten, verwenden Sie eine AWS CloudFormation Vorlage. Eine Vorlage ist eine Datei, die als Vorlage für die Erstellung von AWS Ressourcen wie EC2 Amazon-Instances und Amazon SNS-Themen dient. Die Vorlage für diesen Prozess steht Ihnen GitHub zum Herunterladen zur Verfügung.

Sie stellen die Vorlage zur Verfügung und stellen AWS CloudFormation die Ressourcen AWS CloudFormation, die Sie benötigen, als Stapel in Ihrem bereit AWS-Konto. Bei einem Stack handelt es sich um eine Sammlung von Ressourcen, die Sie als einzelne Einheit verwalten. Wenn Sie diese Schritte abgeschlossen haben, können Sie AWS CloudFormation damit alle Ressourcen im Stack auf einmal löschen. Diese Ressourcen verbleiben nicht in Ihrem AWS-Konto, es sei denn, Sie möchten, dass sie es tun.

Der Stack für dieses Tutorial enthält die folgenden Ressourcen:

- Eine VPC und die zugehörigen Netzwerkressourcen, einschließlich eines Subnetzes, einer Sicherheitsgruppe, eines Internet-Gateways und einer Routing-Tabelle.
- Eine EC2 Amazon-Instance, die im Subnetz in der VPC gestartet wird.
- Amazon SNS-Thema.
- Zwei Funktionen AWS Lambda . Diese Funktionen empfangen Nachrichten, die im Amazon SNS SNS-Thema veröffentlicht werden, und protokollieren Ereignisse in CloudWatch Logs.
- CloudWatch Amazon-Metriken und -Protokolle.

- Eine IAM-Rolle, die es der EC2 Amazon-Instance ermöglicht, Amazon SNS zu verwenden, und eine IAM-Rolle, die es den Lambda-Funktionen ermöglicht, in Protokolle zu schreiben. CloudWatch

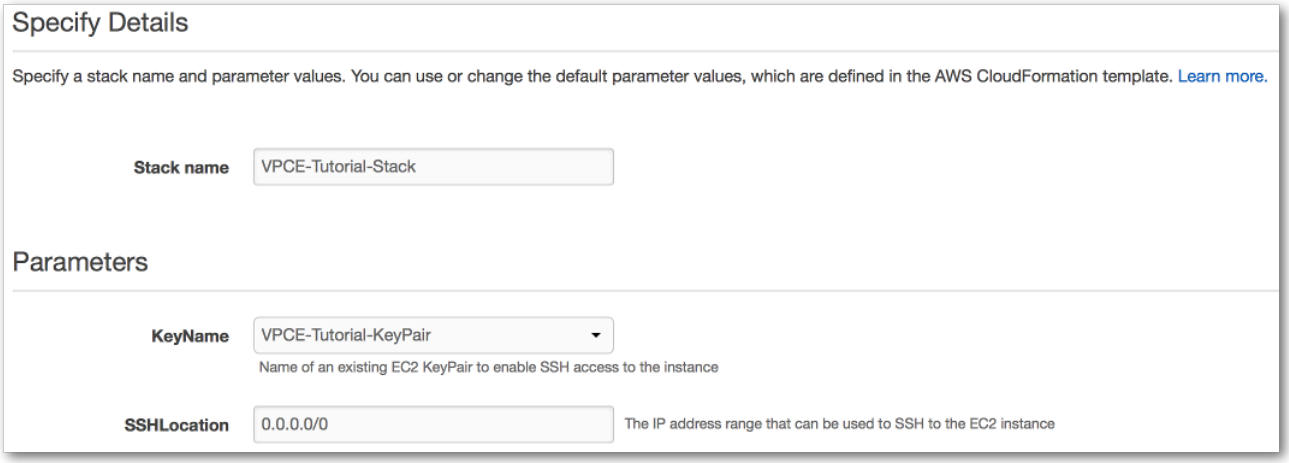
Um die Ressourcen zu erstellen AWS

1. Laden Sie die Vorlagendatei von der GitHub Website herunter.
2. Melden Sie sich an der AWS CloudFormation -Konsole an.
3. Wählen Sie Stapel erstellen aus.
4. Wählen Sie auf der Seite Select Template (Vorlage auswählen) die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen) aus. Wählen Sie dann Ihre Datei und Next (Weiter) aus.
5. Geben Sie auf der Seite Specify Details (Details angeben) Stack- und Schlüsselnamen an:

   a. Geben Sie für Stack name **`VPCE-Tutorial-Stack`** ein.

   b. Wählen Sie für KeyNameVPCE-Tutorial -. KeyPair

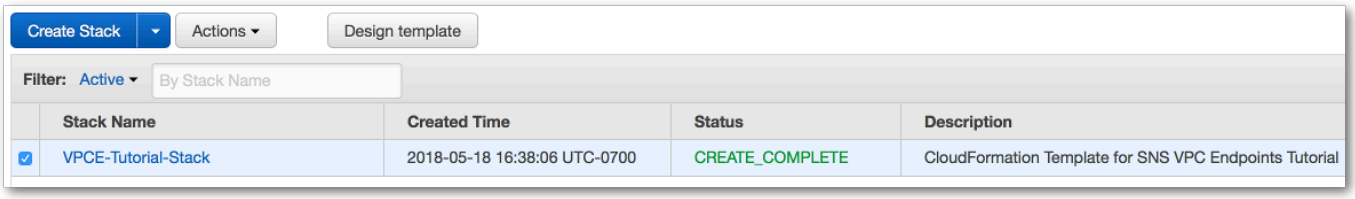
   c. Behalten Sie für SSHLocationden Standardwert von bei. **`0.0.0.0/0`**



   d. Wählen Sie Weiter aus.

6. Behalten Sie auf der Seite Options (Optionen) alle Standardwerte bei, und wählen Sie dann Next (Weiter).
7. Überprüfen Sie die Einstellungen auf der Seite Review (Überprüfen) die Stack-Details.
8. Bestätigen Sie unter Funktionen, dass AWS CloudFormation möglicherweise IAM-Ressourcen mit benutzerdefinierten Namen erstellt werden.
9. Wählen Sie Erstellen aus.

Die AWS CloudFormation Konsole öffnet die Seite Stacks. Der VPCE-Tutorial-Stack hat den Status CREATE_IN_PROGRESS. Nach Abschluss des Erstellungsprozesses ändert sich der Status in wenigen Minuten in CREATE_COMPLETE.



> **Tip**
>
> Wählen Sie die Schaltfläche Refresh (Aktualisieren), um den neuesten Stack-Status zu sehen.

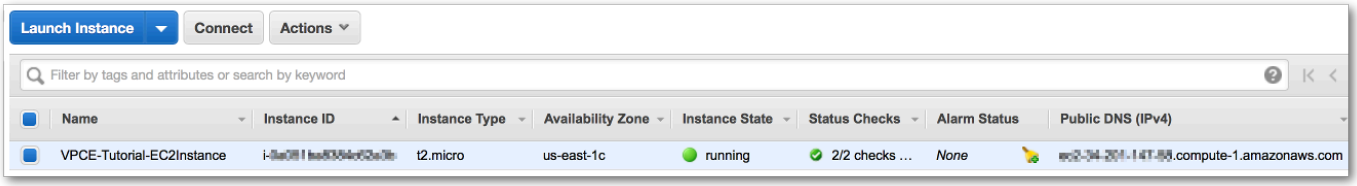
Schritt 3: Stellen Sie sicher, dass Ihre EC2 Amazon-Instance keinen Internetzugang hat

Der EC2 Amazon-Instance, die im vorherigen Schritt in Ihrer VPC gestartet wurde, fehlt der Internetzugang. Sie verbietet ausgehenden Datenverkehr und kann keine Nachrichten in Amazon SNS veröffentlichen. Überprüfen Sie dies, indem Sie sich bei der Instance anmelden. Anschließend versuchen Sie, eine Verbindung zu einem öffentlichen Endpunkt herzustellen, und versuchen, eine Nachricht an Amazon SNS zu senden.

An diesem Punkt schlägt der Veröffentlichungsversuch fehl. In einem späteren Schritt nach dem Erstellen eines VPC-Endpunkts für Amazon SNS ist der Veröffentlichungsversuch erfolgreich.

So stellen Sie eine Verbindung zu Ihrer EC2 Amazon-Instance her

1. Öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Suchen Sie im Navigationsmenü auf der linken Seite den Abschnitt Instances. Wählen Sie anschließend Instances.
3. Wählen Sie in der Liste der Instances VPCE- Tutorial-EC2Instance aus.
4. Kopieren Sie den Hostnamen, der in der Spalte Public DNS angegeben ist.



5. Öffnen Sie ein -Terminalfenster. Stellen Sie von dem Verzeichnis aus, das das key pair enthält, mithilfe des folgenden Befehls eine Verbindung zur Instance her. Dabei *instance-hostname* handelt es sich um den Hostnamen, den Sie von der EC2 Amazon-Konsole kopiert haben:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

So überprüfen Sie, ob die Instance eine Verbindung zum Internet hat

- Versuchen Sie in Ihrem Terminal, eine Verbindung mit einem öffentlichen Endpunkt herzustellen, wie z. B. amazon.com:

  ```
  $ ping amazon.com
  ```

  Da der Verbindungsversuch fehlschlägt, können Sie ihn jederzeit stornieren (Strg+C unter Windows oder Befehl+C unter macOS).

Überprüfen, dass die Instance keine Verbindung zu Amazon SNS hat

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsmenü auf der linken Seite Topics (Themen) aus.
3. Kopieren Sie auf der Seite Topics (Themen) den Amazon-Ressourcennamen (ARN) für das Thema VPCE-Tutorial-Topic.
4. Versuchen Sie in Ihrem Terminal eine Nachricht an das Thema zu veröffentlichen:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   Da der Veröffentlichungsversuch fehlschlägt, können Sie ihn jederzeit abbrechen.

Schritt 4: Erstellen eines Amazon VPC-Endpunkts für Amazon SNS

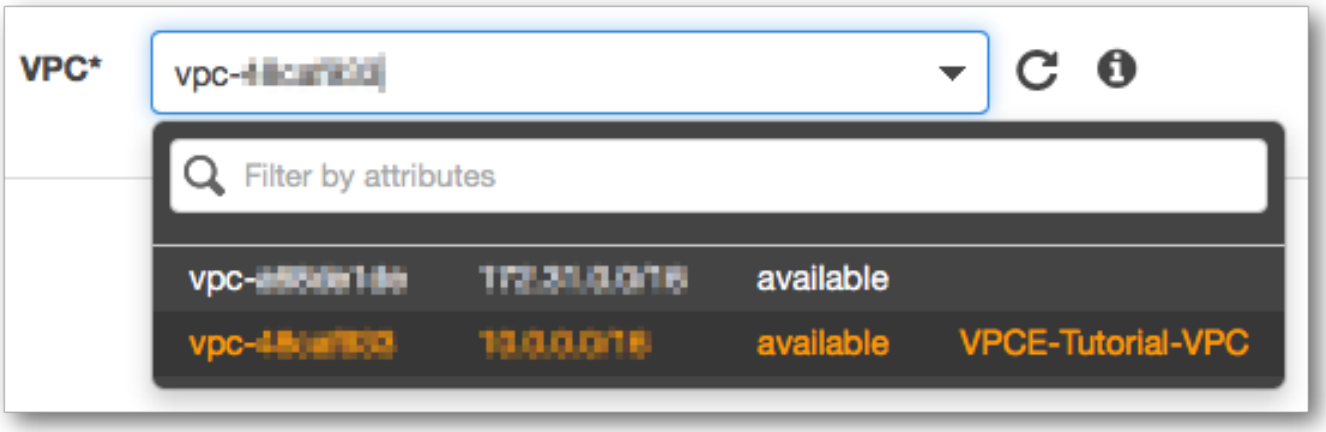
Zum Herstellen einer Verbindung der VPC mit Amazon SNS definieren Sie einen Schnittstellen-VPC-Endpunkt. Nachdem Sie den Endpunkt hinzugefügt haben, können Sie sich bei der EC2 Amazon-Instance in Ihrer VPC anmelden und von dort aus die Amazon SNS-API verwenden. Sie können Nachrichten in dem Thema veröffentlichen. Die Nachrichten werden privat veröffentlicht. Sie bleiben im AWS Netzwerk und nutzen nicht das öffentliche Internet.

> **Note**
>
> Der Instanz fehlt immer noch der Zugriff auf andere AWS Dienste und Endpunkte im Internet.

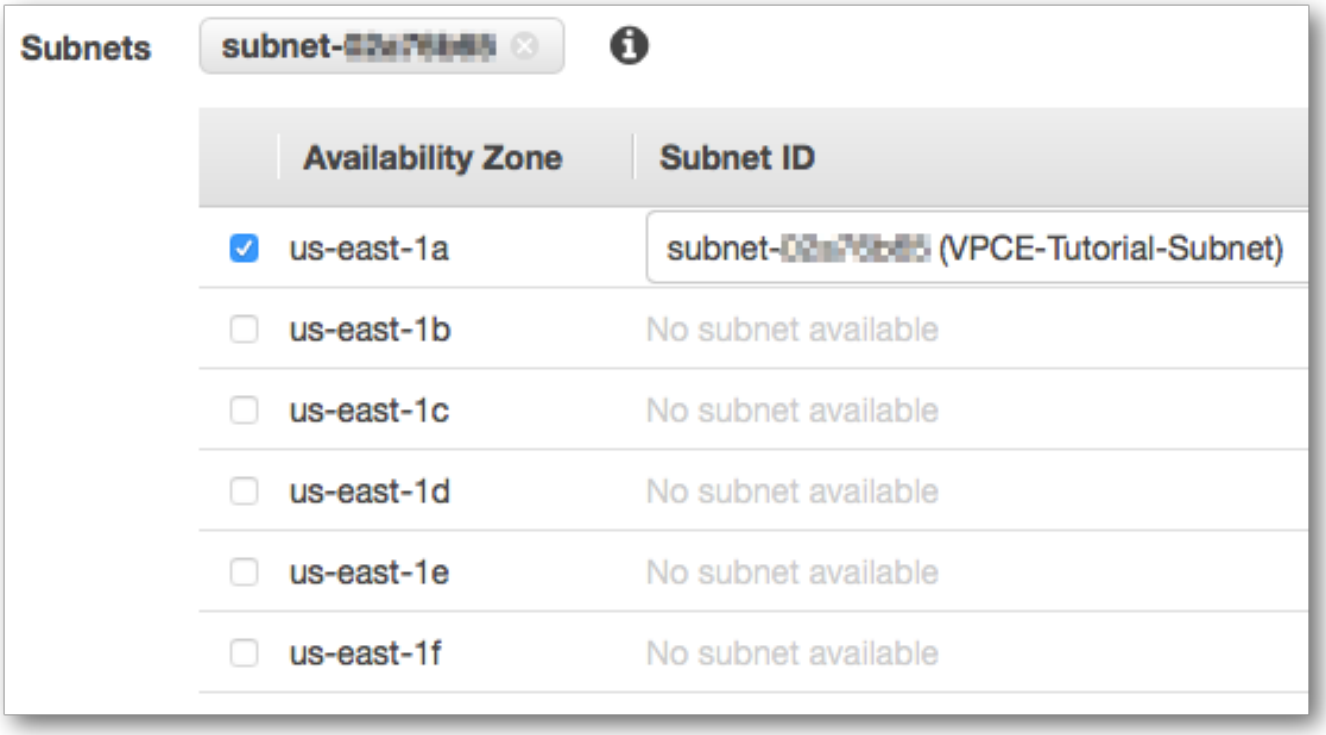
So erstellen Sie den Endpunkt

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsmenü auf der linken Seite Endpoints (Endpunkte) aus.
3. Klicken Sie auf Endpunkt erstellen.
4. Behalten Sie auf der Seite Create Endpoint (Endpunkt erstellen) für Service category (Servicekategorie) die Standardauswahl von AWS Services bei.
5. Wählen Sie für Service Name (Service-Name) den Service-Namen für Amazon SNS.

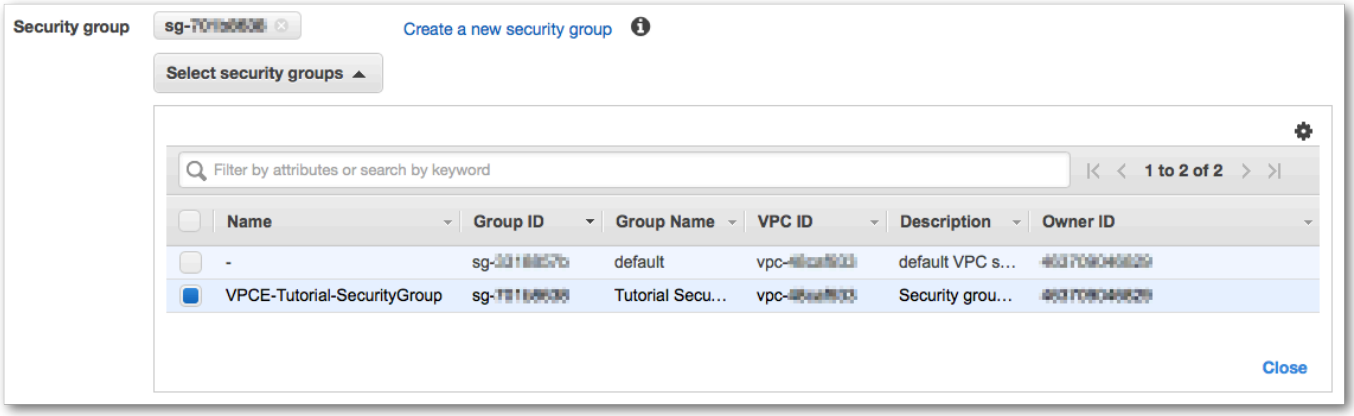
   Die Servicenamen variieren abhängig von der ausgewählten Region. Wenn Sie beispielsweise US East (Nord-Virginia) ausgewählt haben, lautet der Dienstname com.amazonaws.***us-east-1***.sns.
6. Wählen Sie für VPC die VPC mit dem Namen VPCE-Tutorial-VPC.



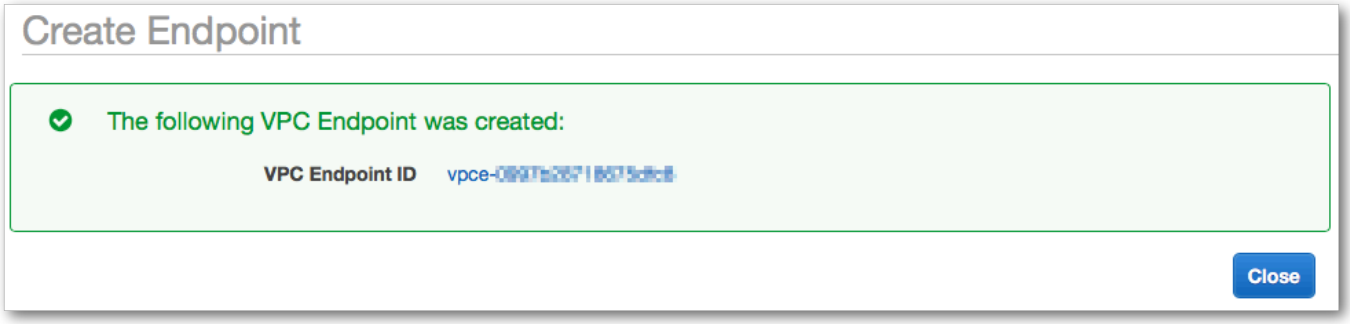
7. Wählen Sie unter Subnets (Subnetze) die Subnetze aus, deren Subnetz-ID VPCE-Tutorial-Subnet enthält.



8. Wählen Sie für Enable Private DNS Name (Privaten DNS-Namen aktivieren) die Option Enable for this endpoint (Für diesen Endpunkt aktivieren) aus.
9. Wählen Sie für Sicherheitsgruppe die Option Sicherheitsgruppe auswählen und anschließend VPCE-Tutorial - aus. SecurityGroup

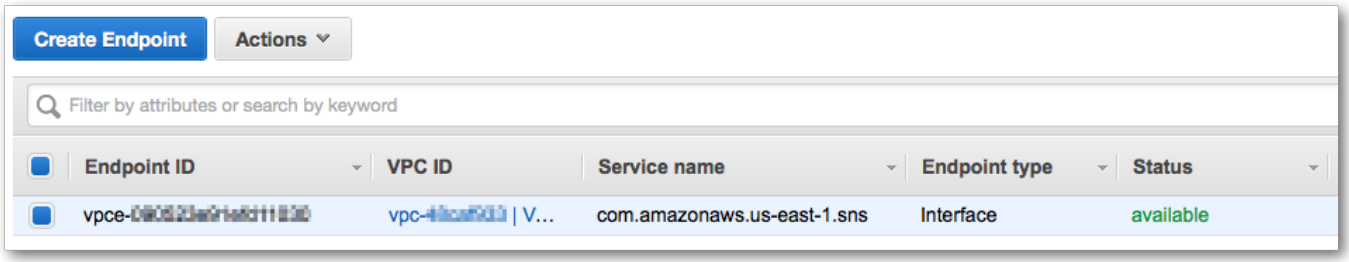


10. Wählen Sie Endpunkt erstellen aus. Die Amazon VPC-Konsole bestätigt, dass ein VPC-Endpunkt erstellt wurde.



11. Klicken Sie auf Schließen.

    Die Amazon-VPC-Konsole öffnet die Seite Endpunkte. Der neue Endpunkt weist den Status pending (ausstehend) auf. Nach Abschluss des Erstellungsprozesses ändert sich der Status in wenigen Minuten in available (verfügbar).



Schritt 5: Veröffentlichen einer Nachricht in Ihrem Amazon SNS-Thema

Da Ihre VPC nun einen Endpunkt für Amazon SNS enthält, können Sie sich bei der EC2 Amazon-Instance anmelden und Nachrichten zum Thema veröffentlichen.

So veröffentlichen Sie eine Nachricht

1. Wenn Ihr Terminal nicht mehr mit Ihrer EC2 Amazon-Instance verbunden ist, stellen Sie erneut eine Verbindung her:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Führen Sie den gleichen Befehl aus wie zuvor, um eine Nachricht in Ihrem Amazon SNS-Thema zu veröffentlichen. Dieses Mal ist der Veröffentlichungsversuch erfolgreich und Amazon SNS gibt eine Mitteilungs-ID zurück:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

```
{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

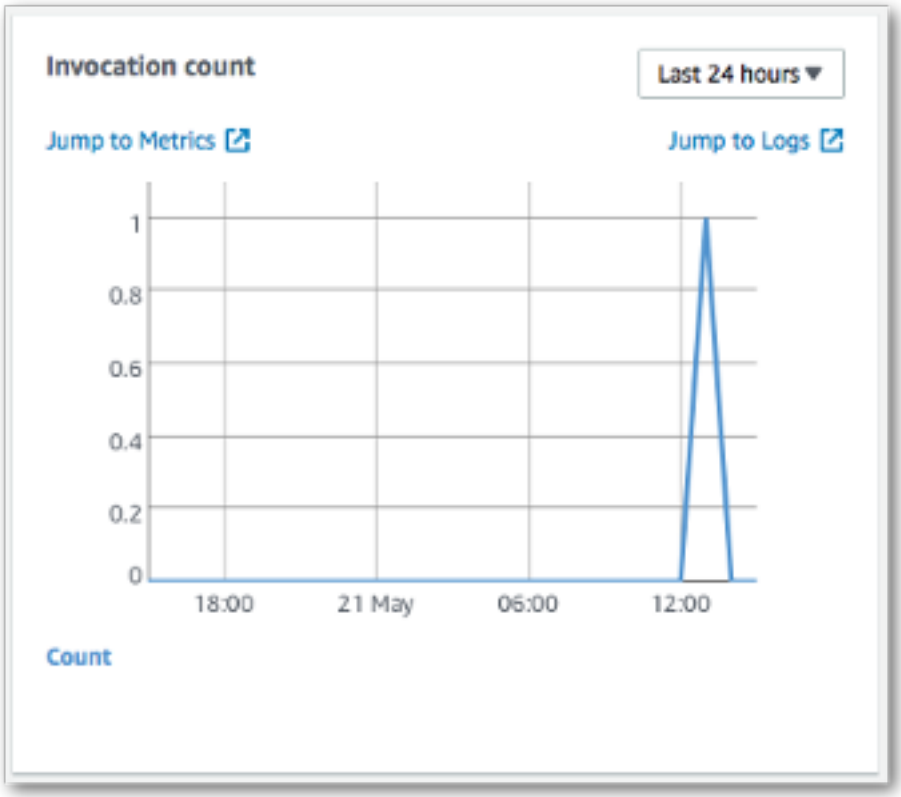
Schritt 6: Überprüfen Sie die Zustellung Ihrer Nachrichten

Wenn das Amazon SNS-Thema eine Nachricht empfängt, wird die Nachricht durch Senden an die zwei Lambda-Funktionen mit Abonnement verbreitet. Wenn diese Funktionen die Nachricht erhalten, protokollieren sie das Ereignis in CloudWatch Protokollen. Um zu überprüfen, ob die Nachrichtenzustellung erfolgreich war, überprüfen Sie, ob die Funktionen aufgerufen wurden, und überprüfen Sie, ob die CloudWatch Protokolle aktualisiert wurden.

So überprüfen Sie, dass die Lambda-Funktionen aufgerufen wurden

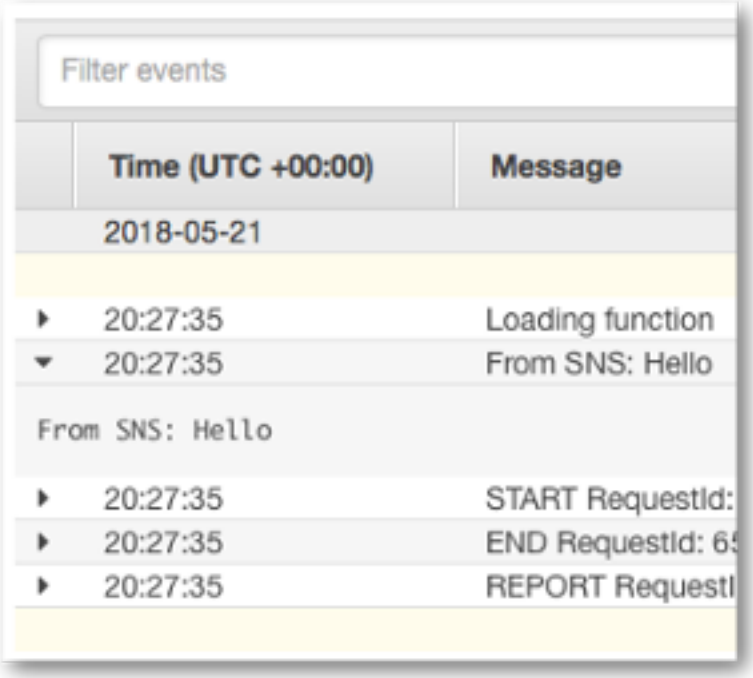
1. Öffnen Sie die AWS Lambda Konsole unter. https://console.aws.amazon.com/lambda/
2. Wählen Sie auf der Seite Functions (Funktionen) die Option VPCE-Tutorial-Lambda-1.
3. Wählen Sie Monitoring.
4. Überprüfen Sie das Diagramm Invocation count (Anzahl der Aufrufe). Dieses Diagramm zeigt die Anzahl der Ausführungen der Lambda-Funktion.

   Die Anzahl der Aufrufe entspricht der Anzahl der Veröffentlichungen einer Nachricht im Thema.



Um zu überprüfen, ob die CloudWatch Protokolle aktualisiert wurden

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsmenü links Logs (Protokolle).
3. Überprüfen Sie die Protokolle, die von den Lambda-Funktionen geschrieben wurden:
   a. Wählen Sie die Loggruppe/aws/lambda/VPCE-Tutorial-Lambda-1/.
   b. Wählen Sie den Protokollstream.
   c. Prüfen Sie, dass das Protokoll den Eintrag `From SNS: Hello` enthält.



d. Wählen Sie Log Groups (Protokollgruppen) oben auf der Konsole, um zur Seite Log Groups (Protokollgruppen) zurückzukehren. Wiederholen Sie dann die vorherigen Schritte für die Protokollgruppe the /aws/lambda/VPCE -Tutorial-Lambda-2/.

Herzlichen Glückwunsch! Durch das Hinzufügen eines Endpunkts für Amazon SNS zu einer VPC konnten Sie eine Nachricht in einem Thema innerhalb des Netzwerks veröffentlichen, das von der VPC verwaltet wird. Die Nachricht wurde privat veröffentlicht, ohne im Internet veröffentlicht zu werden.

Schritt 7: Bereinigen

Sofern Sie die Ressourcen, die Sie für dieses Tutorial erstellt haben, nicht behalten möchten, können Sie sie nun löschen. Durch das Löschen von AWS Ressourcen, die Sie nicht mehr verwenden, vermeiden Sie unnötige Kosten für Ihre AWS-Konto.

Löschen Sie zuerst Ihren VPC-Endpunkt mithilfe der Amazon VPC-Konsole. Löschen Sie anschließend die anderen Ressourcen, die Sie erstellt haben, indem Sie den Stack in der AWS CloudFormation Konsole löschen. Wenn Sie einen Stapel löschen, AWS CloudFormation werden die Ressourcen des Stacks aus Ihrem entfernt AWS-Konto.

So löschen Sie Ihren VPC-Endpunkt

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsmenü auf der linken Seite Endpoints (Endpunkte) aus.
3. Wählen Sie den Endpunkt aus, den Sie erstellt haben.
4. Wählen Sie Actions (Aktionen) und anschließend Delete Endpoint (Endpunkt löschen).
5. Wählen Sie im Fenster Delete Endpoint (Endpunkt löschen) Yes, Delete (Ja, löschen).

   Der Status des Endpunkts wird in deleting (Wird gelöscht...) geändert. Wenn der Löschvorgang abgeschlossen ist, wird der Endpunkt von der Seite entfernt.

Um deinen AWS CloudFormation Stapel zu löschen

1. Öffnen Sie die AWS CloudFormation Konsole unter https://console.aws.amazon.com/cloudformation.
2. Wählen Sie den Stack VPCE-Tutorial-Stack aus.
3. Wählen Sie die Option Actions (Aktionen) und anschließend Delete Stack (Stack löschen) aus.
4. Wählen Sie im Fenster Delete Stack (Stack löschen) Yes, Delete (Ja, löschen).

   Der Stack-Status ändert sich in DELETE_IN_PROGRESS. Wenn der Löschvorgang abgeschlossen ist, wird der Stack von der Seite entfernt.

Zugehörige Ressourcen

Weitere Informationen finden Sie in den folgenden Ressourcen.

- AWS Sicherheitsblog: Schützen von Nachrichten, die auf Amazon SNS veröffentlicht wurden, mit AWS PrivateLink
- Was ist Amazon VPC?
- VPC-Endpunkte
- Was ist Amazon EC2?
- AWS CloudFormation Konzepte

Stellen Sie mithilfe von Dual-Stack IPv4 - (und IPv6) Endpunkten eine Connect zu Amazon SNS her

Dual-Stack-Endpunkte unterstützen sowohl IPv4 den Datenverkehr als auch. IPv6 Wenn Sie eine Anfrage an einen Dual-Stack-Endpunkt stellen, wird die Endpunkt-URL in eine Adresse oder eine IPv4 Adresse aufgelöst. IPv6 Weitere Informationen zu Dual-Stack- und FIPS-Endpunkten finden Sie im SDK-Referenzhandbuch.

Amazon SNS unterstützt regionale Dual-Stack-Endpunkte, was bedeutet, dass Sie die AWS Region als Teil des Endpunktnamens angeben müssen. Dual-Stack-Endpunktnamen verwenden die folgende Benennungskonvention:. `sns.Region.amazonaws.com` Beispielsweise ist der Dual-Stack-Endpunktname für die Region `eu-west-1` `sns.eu-west-1.amazonaws.com`.

Die vollständige Liste der Amazon SNS SNS-Endpunkte finden Sie in der AWS Allgemeinen Referenz.

Verbesserung der Amazon SNS SNS-Sicherheit mit Message Data Protection

- Der Nachrichtendatenschutz ist eine Funktion in Amazon SNS, mit der Sie Ihre eigenen Regeln und Richtlinien definieren können, um den Inhalt für Daten in Bewegung im Gegensatz zu Daten im Ruhezustand zu überprüfen und zu kontrollieren.
- Der Nachrichtendatenschutz bietet Governance-, Compliance- und Auditing-Services für Unternehmensanwendungen, die auf Nachrichten ausgerichtet sind, sodass der Dateneingang und -ausgang vom Amazon-SNS-Themenbesitzer kontrolliert und Inhaltsabläufe verfolgt und protokolliert werden können.
- Sie können nutzlastbasierte Governance-Regeln erstellen, um zu verhindern, dass nicht autorisierte Nutzlastinhalte in Ihre Nachrichten-Streams gelangen.
- Sie können einzelnen Abonnenten unterschiedliche Zugriffsberechtigungen für Inhalte erteilen und den gesamten Inhaltsablaufprozess überwachen.

Identity and Access Management in Amazon SNS

Für den Zugriff auf Amazon SNS sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. Diese Anmeldeinformationen müssen über Berechtigungen

für den Zugriff auf AWS Ressourcen wie Amazon SNS SNS-Themen und -Nachrichten verfügen. In den folgenden Abschnitten erfahren Sie, wie Sie Ihre Ressourcen mithilfe von AWS Identity and Access Management (IAM) und Amazon SNS schützen können, indem Sie den Zugriff auf sie kontrollieren.

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) ist, Amazon-SNS-Ressourcen zu nutzen. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Amazon SNS ausführen.

Service-Benutzer – wenn Sie den Amazon–SNS-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie zur Ausführung von Aufgaben weitere Amazon-SNS-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Wenn Sie auf eine Funktion in Amazon SNS nicht zugreifen können, finden Sie Informationen dazu unter Beheben von Identitäts- und Zugriffsfehlern bei Amazon Simple Notification Service.

Service-Administrator – wenn Sie in Ihrem Unternehmen für die Amazon-SNS-Ressourcen zuständig sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon SNS. Ihre Aufgabe besteht darin, zu bestimmen, auf welche Amazon-SNS-Funktionen und -Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Amazon SNS verwenden kann, finden Sie unter So funktioniert Amazon SNS mit IAM.

IAM-Administrator – wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht Details darüber erfahren, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Amazon SNS erstellen können. Beispiele für identitätsbasierte Amazon-SNS-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als AWS-Konto Root-Benutzer, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle authentifiziert (angemeldet AWS) sein.

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter AWS Signature Version 4 für API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center -Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung (MFA) in IAM im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-

Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Methoden für die Übernahme einer Rolle im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Service aufrufen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es mit einer Identität oder Ressource verknüpft ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto.

Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter Auswählen zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der

identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.

- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich jedes AWS-Konto Root-Benutzers. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten, einschließlich des AWS-Konto Root-Benutzers, auswirken, unabhängig davon, ob diese zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

Zugriffskontrolle

Amazon SNS verfügt über ein eigenes ressourcenbasiertes Berechtigungssystem, das Richtlinien verwendet, die in derselben Sprache verfasst sind, die für AWS Identity and Access Management (IAM) -Richtlinien verwendet wird. Das bedeutet, dass Sie mit Amazon SNS- und IAM-Richtlinien ähnliche Ziele erreichen können.

> **Note**
>
> Es ist wichtig zu verstehen, dass alle ihre Berechtigungen an Benutzer unter ihren Konten delegieren AWS-Konten können. Der kontenübergreifende Zugriff ermöglicht Ihnen den gemeinsamen Zugriff auf AWS -Ressourcen, ohne zusätzliche Benutzer verwalten zu müssen. Weitere Informationen über die Verwendung des kontenübergreifenden Zugriffs finden Sie unter Enabling Cross-Account Access im IAM-Benutzerhandbuch.

Anwendungsfälle für die Amazon SNS SNS-Zugriffskontrolle

Sie sind sehr flexibel, was die Erteilung bzw. Verweigerung des Zugriffs auf eine Ressource anbelangt. Die typischen Anwendungsfälle sind jedoch recht einfach:

- Sie möchten AWS-Konto einem anderen eine bestimmte Art von Themenaktion gewähren (z. B. Veröffentlichen). Weitere Informationen finden Sie unter Gewähren AWS-Konto Sie Zugriff auf ein Thema.
- Sie möchten die Abonnements für Ihr Thema nur auf das HTTPS-Protokoll beschränken. Weitere Informationen finden Sie unter Beschränken von Abonnements auf HTTPS.
- Sie möchten, dass Amazon SNS in der Lage ist, Nachrichten für Ihre Amazon SQS-Queue zu veröffentlichen. Weitere Informationen finden Sie unter Veröffentlichen in einer Amazon SQS-Queue..

Die wichtigsten Konzepte der Amazon SNS SNS-Zugriffsrichtlinie

In den folgenden Abschnitten werden die Konzepte beschrieben, die Sie in Bezug auf die Verwendung der Zugriffskontrollsprache kennen sollten. Sie sind logisch angeordnet, das heißt, die wichtigsten Konzepte befinden sich ganz oben in der Liste.

Berechtigung

Eine Berechtigung ist das Konzept, eine bestimmte Art des Zugriffs auf eine bestimmte Ressource zu erteilen bzw. zu verweigern. Berechtigungen folgen im Allgemeinen dieser Form: "A darf (bzw. darf nicht) B bei C tun, wenn D gilt". Beispiel: Jane (A) hat die Berechtigung zum Veröffentlichen (B) zum Thema A (C), sofern sie das HTTP-Protokoll verwendet (D). Jedes Mal, wenn Jane eine Veröffentlichung zum Thema A vornimmt, prüft der Service, ob Jane dazu berechtigt ist und ob die Anforderung die Bedingungen erfüllt, die für die Berechtigung vorgegeben sind.

Statement

Eine Anweisungist die formelle Beschreibung einer einzelnen Berechtigung, die in der Zugriffskontrollsprache verfasst wird. Sie schreiben eine Anweisung stets als Teil eines umfassenderen Containerdokuments, das als Richtlinie bezeichnet wird (siehe nächstes Konzept).

Richtlinie

Eine Richtlinie ist ein Dokument (verfasst in der Zugriffskontrollsprache), das als Container für eine oder mehrere Anweisungen fungiert. Eine Richtlinie könnte beispielsweise zwei Anweisungen enthalten: eine Anweisung, die besagt, dass Jane unter Verwendung des E-Mail-Protokolls ein Abonnement durchführen darf, und eine andere, die besagt, dass Bob nicht zum Thema A veröffentlichen darf. Die folgende Abbildung zeigt ein analoges Szenario mit zwei Richtlinien: Eine besagt, dass Jane unter Verwendung des E-Mail-Protokolls abonnieren darf, und eine andere, dass Bob nicht zum Thema A veröffentlichen darf.

In Richtliniendokumenten sind nur ASCII-Zeichen zulässig. Sie können das Szenario verwenden `aws:SourceAccount` und umgehen`aws:SourceOwner`, in dem Sie andere AWS Dienste einbinden müssen, ARNs die Nicht-ASCII-Zeichen enthalten. Sehen Sie sich die Unterschiede zwischen `aws:SourceAccount` im Vergleich zu `aws:SourceOwner` an.

Aussteller

Der Aussteller ist die Person, die eine Richtlinie schreibt, um Berechtigungen für eine Ressource zu gewähren. Der Emittent ist (per Definition) immer der Eigentümer der Ressource. AWS erlaubt AWS Dienstbenutzern nicht, Richtlinien für Ressourcen zu erstellen, die ihnen nicht gehören. Wenn John der Eigentümer der Ressource ist, AWS authentifiziert er Johns Identität, wenn er die Richtlinie einreicht, die er zur Erteilung von Berechtigungen für diese Ressource verfasst hat.

Auftraggeber

Der Prinzipal ist die Person bzw. sind die Personen, die die Berechtigung in der Richtlinie erhalten. In der Anweisung "A ist berechtigt, B bei C zu tun, sofern D gilt" ist der Prinzipal "A". Sie können den Prinzipal in einer Richtlinie auf "alle" festlegen (d. h. Sie können einen Platzhalter angeben, der alle Personen umfasst). Sie können dies z. B. tun, wenn Sie den Zugriff nicht basierend auf der tatsächlichen Identität des Anforderers einschränken möchten, sondern aufgrund anderer Identifikationsmerkmale (z. B. der IP-Adresse des Anforderers).

Aktion

Die Aktion ist die Aktivität, die der Prinzipal ausführen darf. Die Aktion ist das „B“ in der Aussage „A ist berechtigt, B bei C zu tun, sofern D gilt“. In der Regel handelt es sich bei der Aktion lediglich um den Vorgang in der Anfrage an AWS. Beispiel: Jane sendet mit `Action=Subscribe`. eine Anfrage an Amazon SNS Sie können in einer Richtlinie eine oder mehrere Aktionen angeben.

Ressource

Die Ressource ist das Objekt, für die der Prinzipal Zugriff anfordert. In der Anweisung "A ist berechtigt, B bei C zu tun, sofern D gilt" ist die Ressource das "C".

Bedingungen und Schlüssel

Bedingungen sind jegliche Einschränkungen oder Details hinsichtlich der Berechtigung. Die Bedingung ist das D in der Aussage „A ist berechtigt, B bei C zu tun, sofern D gilt“. Der Teil der Richtlinie, der die Bedingungen festlegt, kann der zugleich der ausführlichste und komplexeste aller Bestandteile sein. Typische Bedingungen betreffen:

- Datum und Uhrzeit (z. B.: Die Anfrage muss vor einem bestimmten Tag eintreffen.)
- IP-Adresse (z. B. muss die IP-Adresse des Anforderers Teil eines bestimmten CIDR-Bereichs sein)

Ein Schlüssel ist das besondere Merkmal, das die Grundlage für die Einschränkung des Zugriffs bildet, z. B. das Datum und die Uhrzeit der Anfrage.

Sie verwenden Bedingungen und Schlüssel zusammen, um die Einschränkung auszudrücken. Am besten lässt sich anhand eines Beispiel zeigen, wie Sie eine Einschränkung implementieren: Wenn Sie den Zugriff auf die Zeit vor dem 30. Mai 2010 einschränken möchten, verwenden Sie die Bedingung `DateLessThan`. Sie verwenden den Schlüssel namens`aws:CurrentTime`und setzen Sie es auf den Wert`2010-05-30T00:00:00Z`. AWS definiert die Bedingungen und Schlüssel, die Sie verwenden können. Der AWS Service selbst (z. B. Amazon SQS oder Amazon SNS) kann auch dienstspezifische Schlüssel definieren. Weitere Informationen finden Sie unter Amazon SNS-API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen.

Auftraggeber

Der Anforderer ist die Person, die eine Anfrage an einen AWS Dienst sendet und um Zugriff auf eine bestimmte Ressource bittet. Der Anforderer sendet eine Anfrage an AWS , in der es im Wesentlichen heißt: „Erlauben Sie mir, B bis C zu tun, wo D zutrifft?“

Bewertung

Bei der Bewertung entscheidet der AWS Service anhand der geltenden Richtlinien, ob eine eingehende Anfrage abgelehnt oder zugelassen werden sollte. Informationen zur Bewertungslogik finden Sie unter Auswertungslogik.

Auswirkung

Die Auswirkung ist das Ergebnis, das eine Richtlinienanweisung zum Bewertungszeitpunkt zurückgeben soll. Sie geben diesen Wert beim Schreiben der Anweisungen in einer Richtlinie an. Die mögliche Werte sind Zugriffsverweigerung und Zugriffserlaubnis.

Sie können beispielsweise eine Richtlinie schreiben, die eine Anweisung enthält, die alle Anfragen von der Antarktis (Auswirkung = Zugriffsverweigerung, da in der Anfrage eine IP-Adresse verwendet wird, die der Antarktis zugewiesen ist) ablehnt. Sie könnten auch eine Richtlinie schreiben, die eine Anweisung enthält, die alle Anfragen, die nicht von der Antarktis stammen, zulässt (Auswirkung = Zugriffserlaubnis, da die Anfrage nicht von der Antarktis stammt). Diese beiden Anweisungen scheinen zwar das Gleiche zu tun, unterscheiden sich aber nach der Logik der Zugriffskontrollsprache. Weitere Informationen finden Sie unter Auswertungslogik.

Zwar gibt es für die Angabe der Auswirkung (Zulassen oder Ablehnen) nur zwei mögliche Werte, dennoch kann es bei der Bewertung der Richtlinie drei unterschiedliche Ergebnisse geben:

standardmäßige Ablehnung, Zugriffserlaubnis oder explizite Zugriffsverweigerung. Weitere Informationen finden Sie in den folgenden Konzepten und unter Auswertungslogik.

Standardmäßige Ablehnung

Eine standardmäßige Ablehnung ist das Standardergebnis einer Richtlinie, wenn weder "Zugriffserlaubnis" noch "explizite Zugriffsverweigerung" vorhanden sind.

Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf

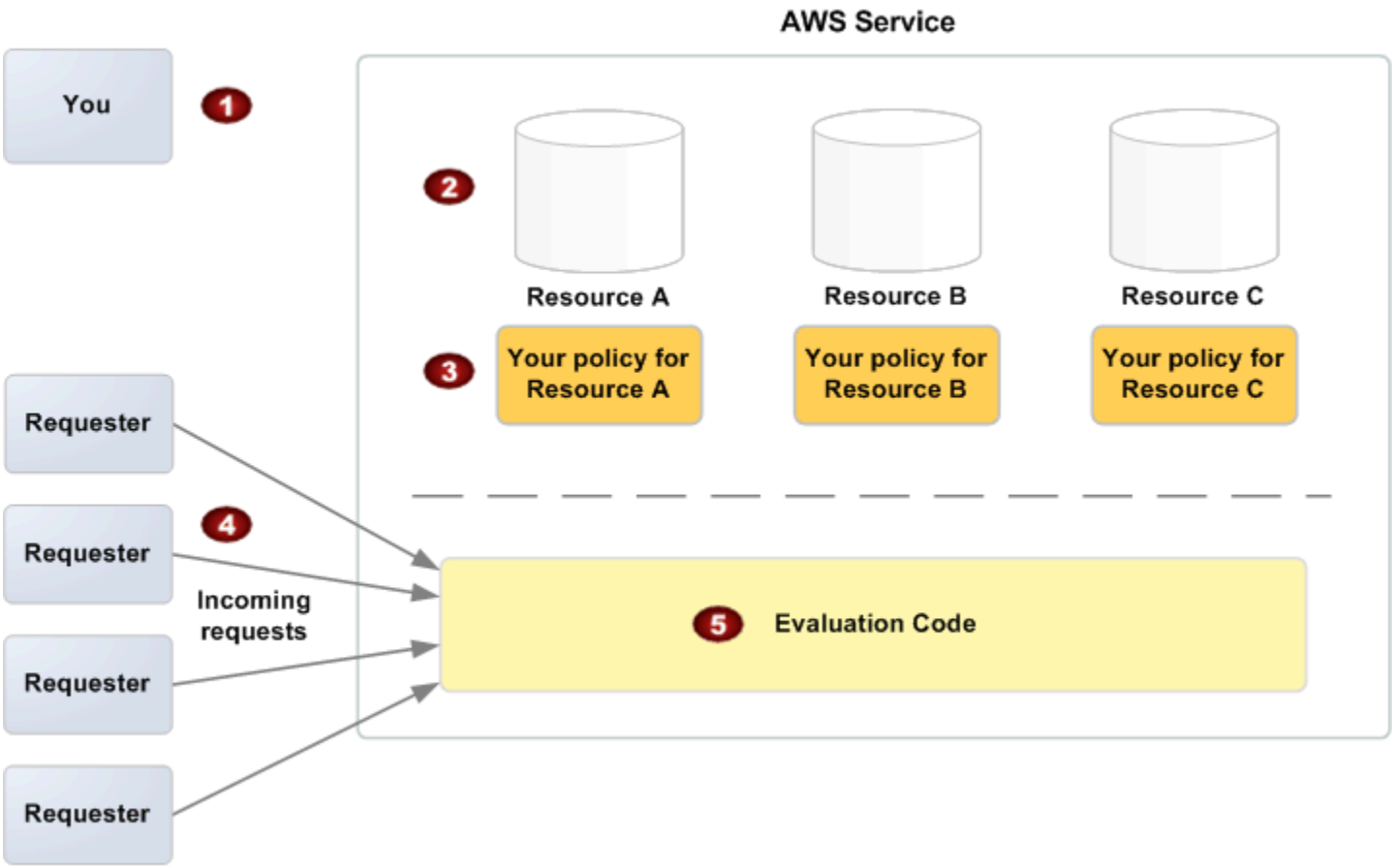
Zugriffserlaubnis resultiert aus einer Anweisung, die die Auswirkung "Zugriffserlaubnis" hat, davon ausgehend, dass alle angegebenen Bedingungen erfüllt sind. Beispiel: Alle Anfragen zulassen, wenn sie am 30. April 2010 vor 13:00 Uhr eingehen. Durch "Zugriffserlaubnis" werden alle standardmäßigen Ablehnungen überschrieben, niemals jedoch eine explizite Zugriffsverweigerung.

Explizite Zugriffsverweigerung

Eine explizite Zugriffsverweigerung resultiert aus einer Anweisung, die die Auswirkung "Zugriffsverweigerung" hat, davon ausgehend, dass alle angegebenen Bedingungen erfüllt sind. Beispiel: Alle Anfragen ablehnen, wenn sie von der Antarktis stammen. Alle Anfragen von der Antarktis werden immer abgelehnt, unabhängig davon, was andere Richtlinien ggf. erlauben.

Überblick über die Architektur der Amazon SNS SNS-Zugriffskontrolle

In der folgenden Abbildung und Tabelle werden die Hauptkomponenten beschrieben, die interagieren, um eine Zugriffskontrolle für Ihre Ressourcen zu bieten.

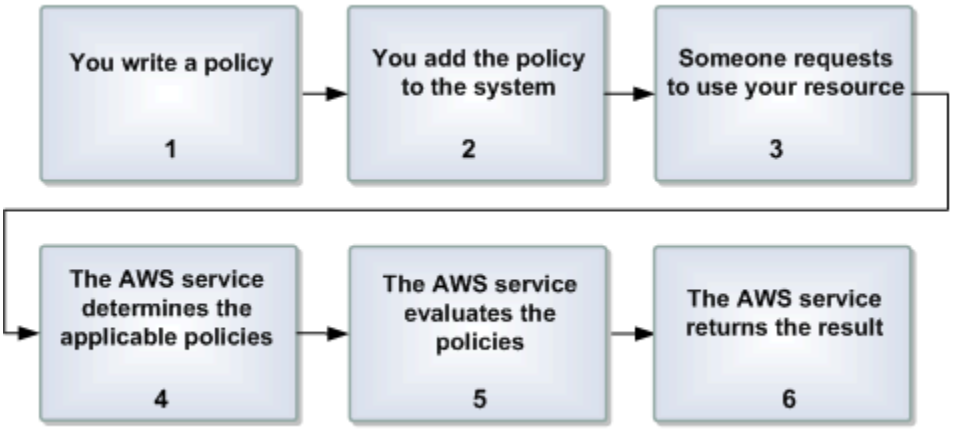


1	Sie selbst als Ressourceneigentümer.
2	Ihre Ressourcen (im AWS Service enthalten, z. B. Amazon SQS SQS-Warteschlangen).
3	Ihre Richtlinien. In der Regel verfügen Sie über eine Richtlinie pro Ressource. Sie können aber auch mehrere haben. Der AWS Service selbst stellt eine API bereit, mit der Sie Ihre Richtlinien hochladen und verwalten können.
4	Anfragende und ihre eingehenden Anfragen an den AWS Dienst.
5	Code für die Auswertung der Zugriffskontrollsprache Dies ist der Codesatz innerhalb des AWS Dienstes, der eingehende Anfragen anhand der geltenden Richtlinien bewertet und bestimmt, ob dem Anforderer Zugriff auf die Ressource

gewährt wird. Weitere Informationen dazu, wie der Service die Entscheidung trifft, finden Sie unter Auswertungslogik.

Verwenden der Sprache der Zugriffsrichtlinien in Amazon SNS

Die folgende Abbildung und Tabelle beschreiben den allgemeinen Prozess der Zugriffskontrolle in der Zugriffskontrollsprache.



Verfahren für die Verwendung der Zugriffsrichtliniensprache

1	Sie schreiben eine Richtlinie für Ihre Ressource. Sie schreiben z. B. eine Richtlinie, um die Berechtigungen für Ihre Amazon-SNS-Themen anzugeben.
2	Sie laden Ihre Richtlinie auf hoch. AWS Der AWS Dienst selbst bietet eine API, mit der Sie Ihre Richtlinien hochladen. Sie verwenden beispielsweise die Amazon SNS `SetTopicAttributes` -Aktion zum Hochladen einer Richtlinie für ein bestimmtes Amazon SNS-Thema.
3	Jemand sendet eine Anfrage zur Verwendung Ihrer Ressource. Beispielsweise sendet ein Benutzer eine Anfrage an Amazon SNS zur Verwendung eines Ihrer Themen.
4	Der AWS Dienst bestimmt, welche Richtlinien für die Anfrage gelten.

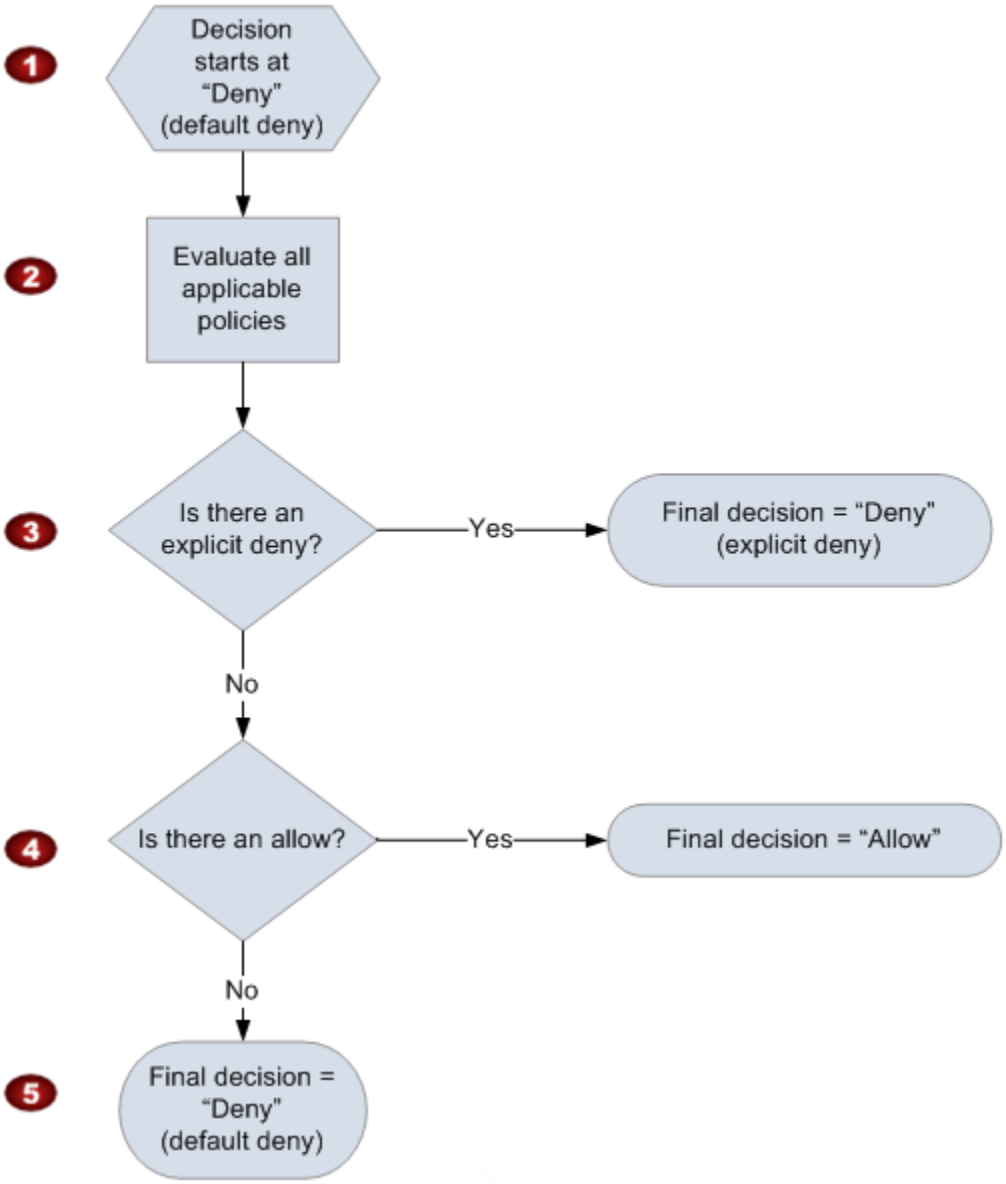
	Zum Beispiel überprüft Amazon SNS alle verfügbaren Amazon SNS-Richtlinien und bestimmt dann, welche anwendbar sind (abhängig davon, um welche Ressource es sich handelt, wer der Anforderer ist usw.).
5	Der AWS Dienst bewertet die Richtlinien. Zum Beispiel bewertet Amazon SNS die Richtlinien und bestimmt, ob der Anforderer Ihr Thema verwenden darf oder nicht. Informationen zur Entscheidungslogik finden Sie unter Auswertungslogik.
6	Der AWS Dienst lehnt die Anfrage entweder ab oder bearbeitet sie weiter. Basierend auf dem Ergebnis der Richtlinienbewertung gibt der Service z. B. entweder den Fehler "Zugriff verweigert" an den Anforderer zurück oder fährt mit der Verarbeitung der Anfrage fort.

Auswertungslogik

Das Ziel zum Bewertungszeitpunkt ist, zu entscheiden, ob eine bestimmte Anfrage gewährt oder abgelehnt werden soll. Die Auswertungslogik unterliegt mehreren Grundregeln:

- Standardmäßig werden alle Anforderungen zur Verwendung Ihrer Ressourcen, die nicht von Ihnen stammen, verweigert
- Eine Zugriffserlaubnis überschreibt jedwede standardmäßige Ablehnung.
- Eine explizite Zugriffsverweigerung überschreibt jedwede Zugriffserlaubnis
- Es spielt keine Rolle, in welcher Reihenfolge die Richtlinien ausgewertet werden

Das folgende Flussdiagramm stellt genauer dar, wie die Entscheidung getroffen wird.



1	Die Entscheidung beginnt mit einer standardmäßigen Ablehnung.
2	Durch den Durchführungscode werden alle Richtlinien ausgewertet, die auf die Anfrage anwendbar sind (basierend auf der Ressource, dem Prinzipal, der Aktion und den Bedingungen). Es spielt keine Rolle, in welcher Reihenfolge die Richtlinien ausgewertet werden.
3	Es wird in allen Richtlinien nach expliziten Anweisungen für Zugriffsverweigerungen gesucht, die auf die Anforderung zutreffen.

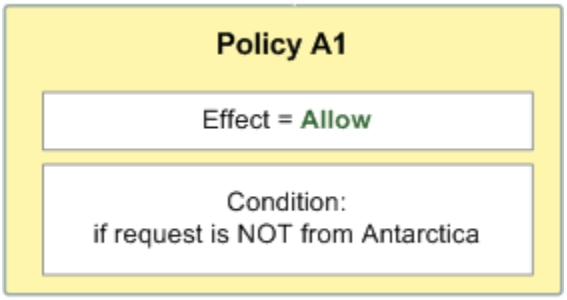
	Wenn auch nur eine gefunden wird, gibt der Durchführungscode eine Ablehnungsentscheidung zurück und der Prozess wird beendet (dies ist eine explizite Zugriffsverweigerung; weitere Informationen siehe Explizite Zugriffsverweigerung).
4	Wenn keine explizite Zugriffsverweigerung gefunden wird, wird nach "allow"-Anweisungen gesucht, die auf die Anfrage zutreffen. Wenn auch nur eine gefunden wird, gibt der Durchführungscode eine Zugriffserlaubnis zurück und der Prozess ist beendet (bzw. der Service setzt die Verarbeitung der Anfrage fort).
5	Wenn kein "allow" gefunden wird, ist die endgültige Entscheidung "deny" (da es keine explizite Zugriffsverweigerung bzw. Zugriffserlaubnis gab, gilt dies als standardmäßige Ablehnung) (weitere Informationen siehe Standardmäßige Ablehnung).

Das Zusammenspiel von expliziten und standardmäßigen Zugriffsverweigerungen

Wenn eine Richtlinie nicht direkt auf eine Anfrage anwendbar ist, wird der Zugriff standardmäßig verweigert ("standardmäßige Ablehnung"). Wenn ein Benutzer beispielsweise die Verwendung von Amazon SNS anfordert, sich die Richtlinie zu diesem Thema jedoch AWS-Konto überhaupt nicht auf die des Benutzers bezieht, führt diese Richtlinie zu einer Standardverweigerung.

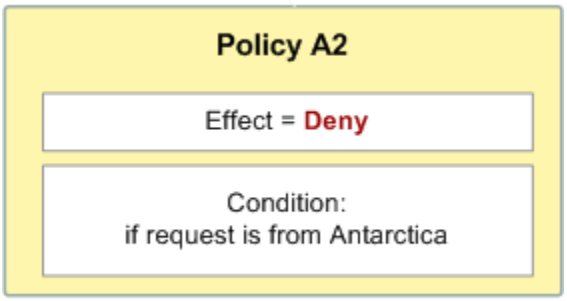
Eine Richtlinie hat außerdem eine standardmäßige Ablehnung zur Folge, wenn eine Bedingung in einer Anweisung nicht erfüllt ist. Wenn alle Bedingungen einer Anweisung erfüllt sind, hat dies für diese Richtlinie abhängig vom Wert des Auswirkungselements in der Richtlinie entweder eine Zugriffserlaubnis oder eine explizite Zugriffsverweigerung zur Folge. In Richtlinien ist nicht festgelegt, was geschieht, wenn eine Bedingung nicht erfüllt ist, daher ist das Ergebnis in diesem Fall eine standardmäßige Ablehnung.

Angenommen, Sie möchten Anforderungen verhindern, die aus der Antarktis stammen. Sie erstellen eine Richtlinie ("Richtlinie A1"), um Zugriff nur dann zu gewähren, wenn Anfragen von außerhalb von Antarktis kommen. Diese Richtlinie ist in der folgenden Abbildung dargestellt.



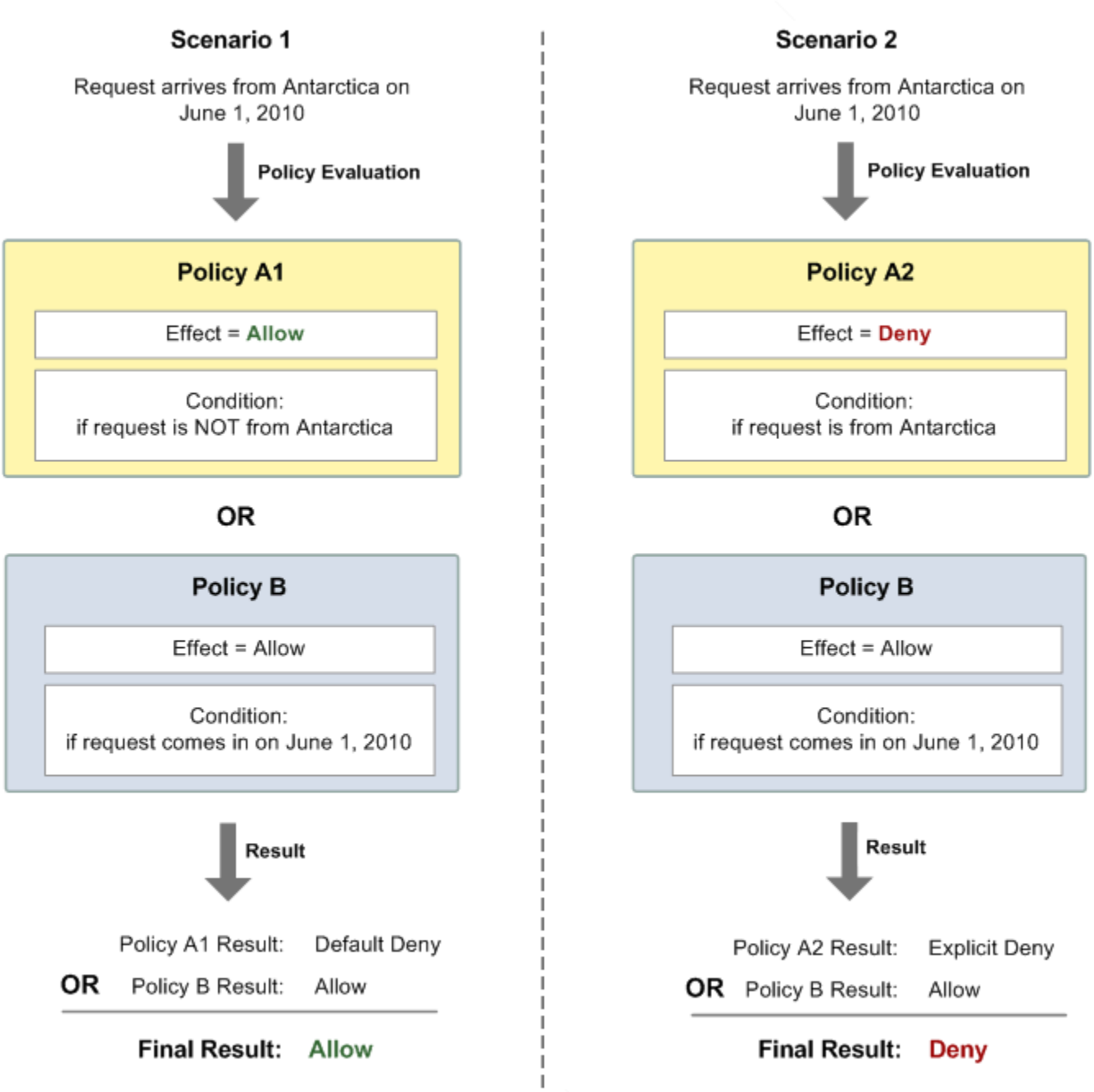
Wenn eine Anforderung aus den USA gesendet wird, ist die Bedingung erfüllt (die Anforderung stammt nicht aus der Antarktis). Daher wird die Anforderung zugelassen. Wenn jedoch eine Anforderung aus der Antarktis eingeht, ist die Bedingung nicht erfüllt und der Zugriff wird standardmäßig abgelehnt.

Sie könnten das Ergebnis in eine explizite Zugriffsverweigerung umwandeln, indem Sie die Richtlinie ("Richtlinie A2") wie im folgenden Diagramm dargestellt neu schreiben. Hier lehnt die Richtlinie eine Anfrage explizit ab, wenn sie von der Antarktis stammt.



Wenn eine Anforderung aus der Antarktis eingeht, ist die Bedingung erfüllt und das Richtlinienergebnis ist daher eine explizite Zugriffsverweigerung.

Die Unterscheidung zwischen "standardmäßiger Ablehnung" und "expliziter Zugriffsverweigerung" ist wichtig, da eine standardmäßige Ablehnung durch eine Zugriffserlaubnis überschrieben werden kann, eine explizite Zugriffsverweigerung jedoch nicht. Angenommen, es gibt eine andere Richtlinie, die Anforderungen zulässt, wenn diese am 1. Juni 2010 gesendet werden. Zu welchem Gesamtergebnis führt diese Richtlinie, wenn sie gemeinsam mit der Richtlinie ausgewertet wird, die Zugriff aus der Antarktis verweigert? Wir vergleichen das Gesamtergebnis, wenn diese datenbasierte Richtlinie (nennen wir sie "Richtlinie B") zusammen mit den vorherigen Richtlinien (A1 und A2) ausgewertet wird. In Szenario 1 werden die Richtlinien A1 und B zusammen ausgewertet, in Szenario 2 die Richtlinien A2 und B. Die folgende Abbildung zeigt die Ergebnisse, wenn eine Anfrage am 1. Juni 2010 aus der Antarktis eingeht.



In Szenario 1 gibt Richtlinie A1 eine standardmäßige Ablehnung wie zuvor beschrieben zurück. Richtlinie B gibt eine Zugriffserlaubnis zurück, da die Richtlinie (laut Definition) Anfragen zulässt, die am 1. Juni 2010 eingehen. Die Zugriffserlaubnis aus Richtlinie B überschreibt die standardmäßige Zugriffsverweigerung aus Richtlinie A1 und die Anforderung wird zugelassen.

In Szenario 2 gibt Richtlinie A2 eine explizite Zugriffsverweigerung wie zuvor beschrieben zurück. Richtlinie B gibt auch hier "allow" zurück. Die Zugriffserlaubnis aus Richtlinie B wird durch die explizite Zugriffsverweigerung aus Richtlinie A2 überschrieben und die Anforderung wird abgelehnt.

Beispiele für die Zugriffskontrolle in Amazon SNS

In diesem Abschnitt finden Sie einige typische Anwendungsfälle für die Zugriffskontrolle.

Gewähren AWS-Konto Sie Zugriff auf ein Thema

Nehmen wir an, Sie haben ein Thema in Amazon SNS und möchten einem oder mehreren Personen erlauben, eine bestimmte Aktion AWS-Konten zu diesem Thema auszuführen, z. B. das Veröffentlichen von Nachrichten. Sie können dies mithilfe der Amazon SNS SNS-API-Aktion `AddPermission` erreichen.

Mit der `AddPermission` Aktion können Sie ein Thema AWS-Konto IDs, eine Liste von Aktionen und eine Bezeichnung angeben. Amazon SNS generiert dann automatisch eine neue Richtlinienerklärung und fügt sie der Zugriffskontrollrichtlinie des Themas hinzu. Sie müssen die Grundsatzerklärung nicht selbst verfassen — Amazon SNS erledigt das für Sie. Wenn Sie die Richtlinie später entfernen müssen, können Sie dies tun, indem Sie uns anrufen `RemovePermission` und das Label angeben, das Sie beim Hinzufügen der Genehmigung verwendet haben.

Wenn Sie beispielsweise das `AddPermission` Thema arn:aws:sns:us-east- 2:444455556666: aufrufen und die AWS-Konto ID 1111-2222-3333, die `Publish` Aktion und das Label `grant-1234-publish` angebenMyTopic, generiert Amazon SNS die folgende Richtlinienerklärung und fügt sie in die Zugriffskontrollrichtlinie des Themas ein:

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Nachdem diese Erklärung hinzugefügt wurde, ist 1111-2222-3333 berechtigt, Nachrichten zu diesem Thema zu veröffentlichen. AWS-Konto

Zusätzliche Informationen:

- Benutzerdefiniertes Richtlinienmanagement: Es `AddPermission` ist zwar praktisch, um Berechtigungen zu erteilen, aber in komplexeren Szenarien wie dem Hinzufügen von Bedingungen oder dem Erteilen von Berechtigungen für bestimmte IAM-Rollen oder -Dienste ist es oft nützlich, die Zugriffskontrollrichtlinie des Themas manuell zu verwalten. Sie können dies tun, indem Sie die `SetTopicAttributes` API verwenden, um das Richtlinienattribut direkt zu aktualisieren.
- Bewährte Sicherheitsmethoden: Seien Sie vorsichtig, wenn Sie Berechtigungen erteilen, um sicherzustellen, dass nur vertrauenswürdige AWS-Konten oder Entitäten Zugriff auf Ihre Amazon SNS SNS-Themen haben. Überprüfen und überprüfen Sie regelmäßig die mit Ihren Themen verknüpften Richtlinien, um die Sicherheit zu gewährleisten.
- Richtlinienbeschränkungen: Beachten Sie, dass der Umfang und die Komplexität der Amazon SNS SNS-Richtlinien begrenzt sind. Wenn Sie viele Berechtigungen oder komplexe Bedingungen hinzufügen müssen, stellen Sie sicher, dass Ihre Richtlinie diese Grenzen nicht überschreitet.

Beschränken von Abonnements auf HTTPS

Um das Benachrichtigungszustellungsprotokoll für Ihr Amazon SNS SNS-Thema auf HTTPS zu beschränken, müssen Sie eine benutzerdefinierte Richtlinie erstellen. Mit der `AddPermission` Aktion in Amazon SNS können Sie keine Protokolleinschränkungen angeben, wenn Sie Zugriff auf Ihr Thema gewähren. Daher müssen Sie manuell eine Richtlinie schreiben, die diese Einschränkung durchsetzt, und dann die `SetTopicAttributes` Aktion verwenden, um die Richtlinie auf Ihr Thema anzuwenden.

So können Sie eine Richtlinie erstellen, die Abonnements auf HTTPS beschränkt:

1. Schreiben Sie die Richtlinie. In der Richtlinie muss die AWS-Konto ID angegeben werden, der Sie Zugriff gewähren möchten, und die Bedingung durchsetzen, dass nur HTTPS-Abonnements zulässig sind. Im Folgenden finden Sie eine Beispielrichtlinie, die der AWS-Konto ID 1111-2222-3333 die Erlaubnis erteilt, das Thema zu abonnieren, jedoch nur, wenn das verwendete Protokoll HTTPS ist.

```
{
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
```

```
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
      }
    }
  }]
}
```

2. Wenden Sie die Richtlinie an. Verwenden Sie die `SetTopicAttributes` Aktion in der Amazon SNS SNS-API, um diese Richtlinie auf Ihr Thema anzuwenden. Legen Sie das `Policy` Attribut des Themas auf die von Ihnen erstellte JSON-Richtlinie fest.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Zusätzliche Informationen:

- Anpassen der Zugriffskontrolle. Mit diesem Ansatz können Sie detailliertere Zugriffskontrollen durchsetzen, z. B. die Einschränkung von Abonnementprotokollen, was mit dieser Aktion allein nicht möglich ist. `AddPermission` Benutzerdefinierte Richtlinien bieten Flexibilität für Szenarien, die bestimmte Bedingungen erfordern, wie z. B. die Durchsetzung von Protokollen oder IP-Adressbeschränkungen.
- Bewährte Sicherheitsmethoden. Die Beschränkung von Abonnements auf HTTPS erhöht die Sicherheit Ihrer Benachrichtigungen, indem sichergestellt wird, dass Daten während der Übertragung verschlüsselt werden. Überprüfen Sie Ihre Themenrichtlinien regelmäßig, um sicherzustellen, dass sie Ihren Sicherheits- und Compliance-Anforderungen entsprechen.
- Testen von Richtlinien. Bevor Sie die Richtlinie in einer Produktionsumgebung anwenden, testen Sie sie in einer Entwicklungsumgebung, um sicherzustellen, dass sie sich erwartungsgemäß verhält. Dies trägt dazu bei, versehentliche Zugriffsprobleme oder unbeabsichtigte Einschränkungen zu vermeiden.

Veröffentlichen in einer Amazon SQS-Queue.

Um Nachrichten aus Ihrem Amazon SNS SNS-Thema in einer Amazon SQS SQS-Warteschlange zu veröffentlichen, müssen Sie die richtigen Berechtigungen für die Amazon SQS SQS-Warteschlange konfigurieren. Sowohl Amazon SNS als auch Amazon AWS SQS verwenden zwar die Sprache der Zugriffskontrollrichtlinien, Sie müssen jedoch explizit eine Richtlinie für die Amazon SQS SQS-Warteschlange festlegen, damit Nachrichten aus dem Amazon SNS SNS-Thema gesendet werden können.

Sie können dies erreichen, indem Sie die `SetQueueAttributes` Aktion verwenden, um eine benutzerdefinierte Richtlinie auf die Amazon SQS SQS-Warteschlange anzuwenden. Im Gegensatz zu Amazon SNS unterstützt Amazon SQS die `AddPermission` Aktion zur Erstellung von Richtlinienerklärungen mit Bedingungen nicht. Daher müssen Sie die Richtlinie manuell schreiben.

Im Folgenden finden Sie ein Beispiel für eine Amazon SQS SQS-Richtlinie, die Amazon SNS die Erlaubnis erteilt, Nachrichten an Ihre Warteschlange zu senden. Beachten Sie, dass diese Richtlinie mit der Amazon SQS SQS-Warteschlange und nicht mit dem Amazon SNS SNS-Thema verknüpft ist. Die angegebenen Aktionen sind Amazon SQS SQS-Aktionen, und die Ressource ist der Amazon-Ressourcenname (ARN) der Warteschlange. Sie können den ARN der Warteschlange mithilfe der `GetQueueAttributes` Aktion abrufen.

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

Diese Richtlinie verwendet die `aws:SourceArn` Bedingung, um den Zugriff auf die SQS-Warteschlange auf der Grundlage der Quelle der gesendeten Nachrichten einzuschränken. Dadurch

wird sichergestellt, dass nur Nachrichten, die aus dem angegebenen SNS-Thema stammen (in diesem Fall arn:aws:sns:us-east- 2:444455556666:), an die Warteschlange zugestellt werden dürfen. MyTopic

Zusätzliche Informationen:

- ARN in die Warteschlange stellen. Stellen Sie sicher, dass Sie mit der `GetQueueAttributes` Aktion den richtigen ARN Ihrer Amazon SQS SQS-Warteschlange abrufen. Dieser ARN ist wichtig, um die richtigen Berechtigungen festzulegen.
- Bewährte Sicherheitsmethoden. Halten Sie sich bei der Einrichtung von Richtlinien immer an das Prinzip der geringsten Rechte. Gewähren Sie dem Amazon SNS SNS-Thema nur die erforderlichen Berechtigungen, um mit der Amazon SQS SQS-Warteschlange zu interagieren, und überprüfen Sie regelmäßig Ihre Richtlinien, um sicherzustellen, dass sie sicher sind up-to-date
- Standardrichtlinien in Amazon SNS. Amazon SNS gewährt nicht automatisch eine Standardrichtlinie, die anderen Benutzern AWS-Services oder Konten den Zugriff auf neu erstellte Themen ermöglicht. Standardmäßig werden Amazon SNS SNS-Themen ohne Berechtigungen erstellt, was bedeutet, dass sie privat sind und nur für das Konto zugänglich sind, das sie erstellt hat. Um den Zugriff für andere AWS-Services, Konten oder Principals zu ermöglichen, müssen Sie explizit eine Zugriffsrichtlinie definieren und an das Thema anhängen. Dies entspricht dem Prinzip der geringsten Rechte und stellt sicher, dass standardmäßig kein unbeabsichtigter Zugriff gewährt wird.
- Testen und Validieren. Nachdem Sie die Richtlinie festgelegt haben, testen Sie die Integration, indem Sie Nachrichten im Amazon SNS SNS-Thema veröffentlichen und überprüfen, ob sie erfolgreich an die Amazon SQS SQS-Warteschlange zugestellt wurden. Auf diese Weise kann bestätigt werden, dass die Richtlinie korrekt konfiguriert ist.

Erlauben Sie Amazon S3 Ereignisbenachrichtigungen für ein Thema

Damit ein Amazon S3 S3-Bucket aus einem anderen AWS-Konto Bucket Ereignisbenachrichtigungen zu Ihrem Amazon SNS SNS-Thema veröffentlichen kann, müssen Sie die Zugriffsrichtlinie des Themas entsprechend konfigurieren. Dazu müssen Sie eine benutzerdefinierte Richtlinie schreiben, die dem Amazon S3-Service die Genehmigung für den jeweiligen Amazon S3-Service erteilt, AWS-Konto und diese Richtlinie dann auf Ihr Amazon SNS SNS-Thema anwenden.

So können Sie es einrichten:

1. Schreiben Sie die Richtlinie. Die Richtlinie sollte dem Amazon S3 S3-Service (s3.amazonaws.com) die erforderlichen Berechtigungen für die Veröffentlichung unter Ihrem Amazon SNS SNS-Thema gewähren. Sie verwenden die `SourceAccount` Bedingung, um sicherzustellen, dass nur der angegebene Benutzer AWS-Konto, dem der Amazon S3 S3-Bucket gehört, Benachrichtigungen zu Ihrem Thema veröffentlichen kann.

   Folgendes ist eine Beispielrichtlinie:

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

   - Eigentümer des Themas — 111122223333 ist die AWS-Konto ID, der das Amazon SNS SNS-Thema gehört.
   - Besitzer des Amazon S3 S3-Buckets — 444455556666 ist die AWS-Konto ID, der der Amazon S3 S3-Bucket gehört, der Benachrichtigungen sendet.

2. Wenden Sie die Richtlinie an. Verwenden Sie die `SetTopicAttributes` Aktion, um diese Richtlinie für Ihr Amazon SNS SNS-Thema festzulegen. Dadurch wird die Zugriffskontrolle für das Thema aktualisiert und umfasst nun die in Ihrer benutzerdefinierten Richtlinie angegebenen Berechtigungen.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Zusätzliche Informationen:

- **`SourceAccount`**Bedingung wird verwendet. Die `SourceAccount` Bedingung stellt sicher, dass nur Ereignisse, die von der angegebenen Adresse stammen AWS-Konto (in diesem Fall 444455556666), das Amazon SNS SNS-Thema auslösen können. Dies ist eine Sicherheitsmaßnahme, um zu verhindern, dass nicht autorisierte Konten Benachrichtigungen zu Ihrem Thema senden.
- Andere unterstützende Dienste**`SourceAccount`**. Die `SourceAccount` Bedingung wird durch die folgenden Dienste unterstützt. Es ist wichtig, diese Bedingung zu verwenden, wenn Sie den Zugriff auf Ihr Amazon SNS SNS-Thema basierend auf dem ursprünglichen Konto einschränken möchten.
  - Amazon API Gateway
  - Amazon CloudWatch
  - DevOpsAmazon-Guru
  - Amazon EventBridge
  - GameLift Amazon-Server
  - Amazon Pinpoint-SMS- und Sprachnachrichten-API
  - Amazon RDS
  - Amazon Redshift
  - Amazon S3 Glacier
  - Amazon SES
  - Amazon Simple Storage Service
  - AWS CodeCommit
  - AWS Directory Service
  - AWS Lambda
  - AWS Systems Manager Incident Manager
- Testen und Validieren. Nachdem Sie die Richtlinie angewendet haben, testen Sie das Setup, indem Sie ein Ereignis im Amazon S3 S3-Bucket auslösen und bestätigen, dass es erfolgreich in Ihrem Amazon SNS SNS-Thema veröffentlicht wurde. Auf diese Weise können Sie sicherstellen, dass Ihre Richtlinie korrekt konfiguriert ist.
- Bewährte Sicherheitsmethoden. Überprüfen und prüfen Sie regelmäßig Ihre Amazon SNS SNS-Themenrichtlinien, um sicherzustellen, dass sie Ihren Sicherheitsanforderungen entsprechen. Die Beschränkung des Zugriffs auf vertrauenswürdige Konten und Dienste ist für die Aufrechterhaltung eines sicheren Betriebs unerlässlich.

Erlauben Sie Amazon SES, in einem Thema zu veröffentlichen, das einem anderen Konto gehört

Sie können einer anderen Person erlauben AWS-Service , Beiträge zu einem Thema zu veröffentlichen, das einem anderen gehört AWS-Konto. Angenommen, Sie haben sich beim 111122223333 Konto angemeldet, Amazon SES geöffnet und eine E-Mail erstellt. Um Benachrichtigungen zu dieser E-Mail an ein Amazon SNS-Thema zu veröffentlichen, das das Konto 444455556666 besitzt, erstellen Sie eine Richtlinie wie die folgende. Um dies zu tun, müssen Sie Informationen über den Prinzipal (den anderen Dienst) und den Besitz jeder Ressource angeben. Die`Resource`-Anweisung enthält das Thema ARN, das die Konto-ID des Topic-Besitzers 444455556666 enthält. Die`"aws:SourceOwner": "111122223333"`-Anweisung gibt an, dass Ihr Konto Eigentümer der E-Mail ist.

JSON

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

Beim Veröffentlichen von Ereignissen in Amazon SNS unterstützen die folgenden Services `aws:SourceOwner`:

- Amazon API Gateway
- Amazon CloudWatch
- DevOpsAmazon-Guru
- GameLift Amazon-Server
- Amazon Pinpoint-SMS- und Sprachnachrichten-API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` im Vergleich zu `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner` ist veraltet, und neue Services können nur über `aws:SourceArn` und `aws:SourceAccount` in Amazon SNS integriert werden. Amazon SNS behält weiterhin die Abwärtskompatibilität für bestehende Services bei, die derzeit `aws:SourceOwner` unterstützen.

Die Bedingungsschlüssel `aws:SourceAccount` und `aws:SourceOwner` werden jeweils von einigen AWS-Services -Services verwendet, wenn sie in einem Amazon-SNS-Thema veröffentlicht werden. Sofern unterstützt, entspricht der Wert der 12-stelligen AWS Konto-ID, in deren Namen der Dienst Daten veröffentlicht. Einige Dienste unterstützen einen, andere unterstützen den anderen.

- Hier finden Sie Informationen Erlauben Sie Amazon S3 Ereignisbenachrichtigungen für ein Thema zur Verwendung von Amazon S3 S3-Benachrichtigungen `aws:SourceAccount` und eine Liste der AWS Dienste, die diese Bedingung unterstützen.
- Hier finden Sie Informationen Erlauben Sie Amazon SES, in einem Thema zu veröffentlichen, das einem anderen Konto gehört zur Verwendung von Amazon SES `aws:SourceOwner` und eine Liste der AWS Services, die diese Bedingung unterstützen.

Erlauben Sie Konten in einer Organisation AWS Organizations , Beiträge zu einem Thema in einem anderen Konto zu veröffentlichen

Der AWS Organizations Service hilft Ihnen dabei, die Abrechnung zentral zu verwalten, den Zugriff und die Sicherheit zu kontrollieren und Ressourcen gemeinsam zu AWS-Konten nutzen.

Sie finden Ihre Organisations-ID in der Organisationskonsole. Weitere Informationen finden Sie unter Anzeigen von Details zu einer Organisation vom Masterkonto aus.

In diesem Beispiel `myOrgId` kann jede AWS-Konto interne Organisation ein Thema `MyTopic` im Konto `444455556666` auf Amazon SNS veröffentlichen. Die Richtlinie überprüft den Organisations-ID-Wert mithilfe des globalen Bedingungsschlüssels `aws:PrincipalOrgID`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

Erlauben Sie jedem CloudWatch Alarm, zu einem Thema in einem anderen Konto zu veröffentlichen

Gehen Sie wie folgt vor, um ein Amazon SNS SNS-Thema mit einem CloudWatch Alarm in verschiedenen Bereichen aufzurufen. AWS-Konten In diesem Beispiel werden zwei Konten verwendet:

- Konto A wird verwendet, um den CloudWatch Alarm zu erstellen.
- Konto B wird verwendet, um ein SNS-Thema zu erstellen.

Erstellen Sie ein SNS-Thema in Konto B

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) und Create topic (Thema erstellen).
3. Wählen Sie Standard als Thementyp und erstellen Sie dann einen Namen für das Thema.
4. Wählen Sie Create topic aus und kopieren Sie dann den ARN des Themas.
5. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) und dann Create subscription (Abonnement erstellen) aus.
6. Fügen Sie den ARN des Themas im Abschnitt Themen-ARN hinzu, wählen Sie E-Mail als Protokoll aus und geben Sie dann eine E-Mail-Adresse ein.
7. Wählen Sie Abonnement erstellen und überprüfen Sie dann Ihre E-Mails, um das Abonnement zu bestätigen.

Erstellen Sie einen CloudWatch Alarm in Konto A

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Alarme und dann Alarme erstellen aus.
3. Wenn Sie noch keinen Alarm erstellt haben, erstellen Sie jetzt einen. Wählen Sie andernfalls Ihre Metrik aus und geben Sie dann Details für den Schwellenwert und die Vergleichsparameter an.
4. Wählen Sie unter Aktionen konfigurieren unter Benachrichtigungen die Option Thema ARN verwenden, um andere Konten zu benachrichtigen, und geben Sie dann das Thema ARN von Konto B ein.
5. Erstellen Sie einen Namen für den Alarm und wählen Sie dann Alarm erstellen aus.

Aktualisieren Sie die Zugriffsrichtlinie für das SNS-Thema in Konto B

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Themen und dann das Thema aus.
3. Wählen Sie Bearbeiten und fügen Sie der Richtlinie dann Folgendes hinzu:

> **Note**
>
> Ersetzen Sie die Beispielwerte in der Richtlinie unten durch Ihre eigenen.

JSON

```
{
    "Version": "2008-10-17",
    "Id": "__default_policy_ID",
    "Statement": [
        {
            "Sid": "__default_statement_ID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "SNS:GetTopicAttributes",
                "SNS:SetTopicAttributes",
                "SNS:AddPermission",
                "SNS:RemovePermission",
                "SNS:DeleteTopic",
                "SNS:Subscribe",
                "SNS:ListSubscriptionsByTopic",
                "SNS:Publish"
            ],
            "Resource": "example-topic-arn-account-b",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:"
                }
            }
        }
    ]
}
```

Testen Sie den Alarm

Um den Alarm zu testen, ändern Sie entweder den Alarmschwellenwert auf der Grundlage der metrischen Datenpunkte oder ändern Sie den Alarmstatus manuell. Wenn Sie den Alarmschwellenwert oder den Alarmstatus ändern, erhalten Sie eine E-Mail-Benachrichtigung.

Problemumgehung für die Verwendung eines lokalen Amazon SNS SNS-Themas und die Weiterleitung von Nachrichten

Gehen Sie wie folgt vor, um kontoübergreifende Amazon SNS SNS-Benachrichtigungen für CloudWatch Alarme zu aktivieren:

1. Erstellen Sie ein Amazon SNS SNS-Thema in demselben Konto wie der CloudWatchAlarm (111122223333).
2. Abonnieren Sie eine Lambda-Funktion oder eine EventBridgeAmazon-Regel für dieses Amazon SNS-Thema.
3. Die Lambda-Funktion oder EventBridge -Regel kann die Nachricht dann im Amazon SNS SNS-Thema im Zielkonto (444455556666) veröffentlichen.

Die Veröffentlichung von einem bestimmten VPC-Endpunkt aus auf nur ein Amazon SNS -Thema beschränken

In diesem Fall darf das Thema im Konto 444455556666 nur vom VPC-Endpunkt mit der ID `vpce-1ab2c34d` veröffentlicht werden.

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

So funktioniert Amazon SNS mit IAM

Bevor Sie mit IAM den Zugriff auf Amazon SNS verwalten können, sollten Sie sich darüber informieren, welche IAM-Funktionen Sie mit Amazon SNS verwenden können.

IAM-Funktionen, die Sie mit Amazon Simple Notification Service verwenden können

IAM-Feature	Amazon-SNS-Support
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Ja
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (services pezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie Amazon SNS und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren.

AWS verwaltete Richtlinien für Amazon Simple Notification Service

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet AWS wird. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur

Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinie: Amazon SNSFull Access

`AmazonSNSFullAccess`bietet vollen Zugriff auf Amazon SNS mit dem AWS Management Console. Diese Richtlinie umfasst auch die folgenden Lese- und Schreibaktionen AWS Endbenutzer-Messaging SMS für Aufrufe über Amazon SNS. Sie können diese Richtlinie an Ihre Benutzer, Gruppen oder Rollen anhängen.

Details zu Berechtigungen

Die folgenden Berechtigungen gelten nur bei Verwendung von Amazon SNS APIs:

- `sns:*`— Erlaubt volle Berechtigungen zur Ausführung aller Aktionen im Zusammenhang mit Amazon SNS. Dieser Platzhalter (*) bedeutet, dass der Benutzer alle möglichen Amazon SNS SNS-Aktionen ausführen kann.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Ermöglicht das Abrufen einer Liste von Telefonnummern, die für den Versand von SMS-Nachrichten innerhalb von verifiziert wurden. AWS-Konto
- `sms-voice:CreateVerifiedDestinationNumber`— Ermöglicht es Ihnen, eine neue Telefonnummer für die Verwendung mit SMS-Messaging-Diensten innerhalb von zu verifizieren AWS.
- `sms-voice:SendDestinationNumberVerificationCode`— Ermöglicht das Senden eines Bestätigungscodes an eine Telefonnummer, die gerade für SMS-Nachrichten verifiziert wird AWS.
- `sms-voice:SendTextMessage`— Ermöglicht es Ihnen, eine neue Textnachricht zu erstellen und diese an die Telefonnummer eines Empfängers zu senden. `SendTextMessage`sendet bei jedem Aufruf nur eine SMS-Nachricht an einen Empfänger.

- `sms-voice:DeleteVerifiedDestinationNumber`— Ermöglicht das Entfernen einer Telefonnummer aus der Liste der verifizierten Nummern innerhalb der AWS-Konto
- `sms-voice:VerifyDestinationNumber`— Ermöglicht es Ihnen, den Überprüfungsprozess für eine Telefonnummer einzuleiten und abzuschließen, die für SMS-Nachrichtendienste innerhalb von AWS.
- `sms-voice:DescribeAccountAttributes`— Ermöglicht das Abrufen detaillierter Informationen zu den Attributen auf Kontoebene im Zusammenhang mit den darin enthaltenen SMS-Nachrichtendiensten. AWS
- `sms-voice:DescribeSpendLimits`— Ermöglicht das Abrufen von Informationen zu den Ausgabenlimits für SMS-Messaging-Dienste innerhalb der AWS-Konto
- `sms-voice:DescribePhoneNumbers`— Ermöglicht das Abrufen detaillierter Informationen zu den Telefonnummern, die mit SMS-Messaging-Diensten verknüpft sind, innerhalb der AWS-Konto
- `sms-voice:SetTextMessageSpendLimitOverride`— Ermöglicht das Festlegen oder Überschreiben des Ausgabenlimits für SMS-Textnachrichten innerhalb der AWS-Konto
- `sms-voice:DescribeOptedOutNumbers`— Ermöglicht das Abrufen einer Liste von Telefonnummern, die sich gegen den Empfang von SMS-Nachrichten von Ihrem AWS Konto entschieden haben.
- `sms-voice:DeleteOptedOutNumber`— Ermöglicht das Entfernen einer Telefonnummer aus der Liste der abgemeldeten Nummern innerhalb der AWS-Konto

`AmazonSNSFullAccess`Beispiel für eine Richtlinie

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSFullAccess",
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
```

```
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:CreateVerifiedDestinationNumber",
                "sms-voice:SendDestinationNumberVerificationCode",
                "sms-voice:SendTextMessage",
                "sms-voice:DeleteVerifiedDestinationNumber",
                "sms-voice:VerifyDestinationNumber",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:SetTextMessageSpendLimitOverride",
                "sms-voice:DescribeOptedOutNumbers",
                "sms-voice:DeleteOptedOutNumber"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Die Berechtigungen für diese Richtlinie finden Sie unter Amazon SNSFull Access in der Referenz für AWS verwaltete Richtlinien.

AWS verwaltete Richtlinie: Amazon SNSRead OnlyAccess

`AmazonSNSReadOnlyAccess`bietet schreibgeschützten Zugriff auf Amazon SNS mit dem. AWS Management Console Diese Richtlinie umfasst auch die folgenden schreibgeschützten Aktionen für AWS Endbenutzer-Messaging SMS Anrufe über Amazon SNS. Sie können diese Richtlinie Ihren Benutzern, Gruppen und Rollen zuordnen.

Details zu Berechtigungen

Die folgenden Berechtigungen gelten nur bei Verwendung von Amazon SNS APIs:

- `sns:GetTopicAttributes`— Ermöglicht das Abrufen der Attribute eines Amazon SNS SNS-Themas. Dazu gehören Informationen wie der ARN (Amazon Resource Name) des Themas, die Abonnentenliste, Lieferrichtlinien, Zugriffskontrollrichtlinien und alle anderen Metadaten, die mit dem Thema verknüpft sind.

- `sns:List*`— Ermöglicht Ihnen, jeden Vorgang auszuführen, der mit `List` Amazon SNS SNS-Ressourcen beginnt. Dazu gehören Berechtigungen zum Auflisten verschiedener Elemente im Zusammenhang mit Amazon SNS, wie z. B.:
  - `sns:ListTopics`— Ermöglicht das Abrufen einer Liste aller Amazon SNS SNS-Themen in der AWS-Konto.
  - `sns:ListSubscriptions`— Ermöglicht das Abrufen einer Liste aller Abonnements für Amazon SNS SNS-Themen.
  - `sns:ListSubscriptionsByTopic`— Ermöglicht es Ihnen, alle Abonnements für ein bestimmtes Amazon SNS SNS-Thema aufzulisten.
  - `sns:ListPlatformApplications`— Ermöglicht es Ihnen, alle Plattformanwendungen aufzulisten, die für mobile Push-Benachrichtigungen erstellt wurden.
  - `sns:ListEndpointsByPlatformApplication`— Ermöglicht es Ihnen, alle Endpunkte aufzulisten, die einer Plattformanwendung zugeordnet sind.
- `sns:CheckIfPhoneNumberIsOptedOut`— Ermöglicht es Ihnen zu überprüfen, ob eine bestimmte Telefonnummer den Empfang von SMS-Nachrichten über Amazon SNS deaktiviert hat.
- `sns:GetEndpointAttributes`— Ermöglicht das Abrufen der Attribute eines Endpunkts, der mit einer Amazon SNS SNS-Plattformanwendung verknüpft ist. Dies kann Attribute wie den aktivierten Status des Endpunkts, benutzerdefinierte Benutzerdaten und alle anderen mit dem Endpunkt verknüpften Metadaten beinhalten.
- `sns:GetDataProtectionPolicy`— Ermöglicht es Ihnen, die mit einem Amazon SNS SNS-Thema verknüpfte Datenschutzrichtlinie abzurufen.
- `sns:GetPlatformApplicationAttributes`— Ermöglicht das Abrufen der Attribute einer Amazon SNS-Plattformanwendung. Plattformanwendungen werden in Amazon SNS verwendet, um Push-Benachrichtigungen über Dienste wie Apple Push Notification Service (APNS) oder Firebase Cloud Messaging (FCM) an mobile Geräte zu senden.
- `sns:GetSMSAttributes`— Ermöglicht das Abrufen der Standard-SMS-Einstellungen für. AWS-Konto
- `sns:GetSMSSandboxAccountStatus`— Ermöglicht es Ihnen, den aktuellen Status der SMS-Sandbox für Sie AWS-Konto abzurufen.
- `sns:GetSubscriptionAttributes`— Ermöglicht das Abrufen der Attribute eines bestimmten Abonnements für ein Amazon SNS SNS-Thema.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Ermöglicht das Anzeigen oder Abrufen einer Liste von Telefonnummern, die für den Versand von SMS-Nachrichten verifiziert wurden, innerhalb der AWS-Konto

- `sms-voice:DescribeAccountAttributes`— Ermöglicht das Anzeigen oder Abrufen von Informationen zu den Attributen auf Kontoebene im Zusammenhang mit den darin enthaltenen SMS-Nachrichtendiensten. AWS
- `sms-voice:DescribeSpendLimits`— Ermöglicht das Anzeigen oder Abrufen von Informationen zu den Ausgabenlimits für SMS-Nachrichtendienste in Ihrem AWS-Konto
- `sms-voice:DescribePhoneNumbers`— Ermöglicht das Anzeigen oder Abrufen von Informationen zu den Telefonnummern, die für SMS-Messaging-Dienste verwendet werden, innerhalb der AWS-Konto
- `sms-voice:DescribeOptedOutNumbers`— Ermöglicht das Anzeigen oder Abrufen einer Liste von Telefonnummern, für die der Empfang von SMS-Nachrichten von Ihrem AWS-Konto

`AmazonSNSReadOnlyAccess`Beispiel für eine Richtlinie

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "sns:GetTopicAttributes",
                "sns:List*",
                "sns:CheckIfPhoneNumberIsOptedOut",
                "sns:GetEndpointAttributes",
                "sns:GetDataProtectionPolicy",
                "sns:GetPlatformApplicationAttributes",
                "sns:GetSMSAttributes",
                "sns:GetSMSSandboxAccountStatus",
                "sns:GetSubscriptionAttributes"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
```

```
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:DescribeOptedOutNumbers"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Die Berechtigungen für diese Richtlinie finden Sie unter Amazon SNSFull Access in der Referenz für AWS verwaltete Richtlinien.

Amazon SNS SNS-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon SNS an, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Amazon SNS-Dokumentverlaufsseite.

Änderung	Beschreibung	Datum
AmazonSNSFullAccess – Aktualisierung auf eine bestehende Richtlinie	Amazon SNS hat neue Berechtigungen hinzugefügt, um vollen Zugriff auf Amazon SNS mithilfe von zu ermöglichen. AWS Management Console	24.09.2024
Amazon SNSRead OnlyAccess — Aktualisierung einer bestehenden Richtlinie	Amazon SNS hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf Amazon SNS	24.09.2024

Änderung	Beschreibung	Datum
	mithilfe von zu ermöglichen. AWS Management Console	
Amazon SNS hat mit der Nachverfolgung von Änderungen begonnen	Amazon SNS hat damit begonnen, Änderungen an seinen AWS verwalteten Richtlinien nachzuverfolgen.	27.08.2024

Richtlinienaktionen für Amazon SNS

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Amazon-SNS-Aktionen finden Sie unter Von Amazon Simple Notification Service definierte Ressourcen in der Service-Autorisierungs-Referenz.

Richtlinienaktionen in Amazon SNS verwenden das folgende Präfix vor der Aktion:

```
sns
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "sns:action1",
```

```
        "sns:action2"
           ]
```

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Richtlinienressourcen für Amazon SNS

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Amazon SNS SNS-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von Amazon Simple Notification Service definierte Aktionen in der Service Authorization Reference. Um zu erfahren, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, lesen Sie von Von Amazon Simple Notification Service definierte Ressourcen.

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Richtlinien-Bedingungsschlüssel für Amazon SNS

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der Amazon-SNS-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon Simple Notification Service in der Service-Authorization-Referenz. Informationen dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon Simple Notification Service definierte Ressourcen.

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

ACLs auf Amazon SNS

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon SNS

Unterstützt ABAC (Tags in Richtlinien): Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. AWS In werden diese Attribute Tags genannt. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit Amazon SNS

Unterstützt temporäre Anmeldeinformationen: Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit IAM.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch

automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln von einer Benutzerrolle zu einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipalberechtigungen für Amazon SNS

Unterstützt Forward Access Sessions (FAS): Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Amazon SNS

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle könnte die Funktionalität von Amazon SNS beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon SNS dazu Anleitungen gibt.

Serviceverknüpfte Rollen für Amazon SNS

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service

Benutzer besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Amazon-SNS-Ressourcen. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Amazon SNS definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Simple Notification Service in der Service Authorization Reference.

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand Amazon-SNS-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon-SNS-Konsole

Um auf die Konsole von Amazon Simple Notification Service zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon SNS SNS-Ressourcen in Ihrem AWS-Konto aufzulisten und anzuzeigen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Amazon SNS-Konsole weiterhin verwenden können, fügen Sie den Entitäten auch die Amazon SNS `ConsoleAccess` - oder `ReadOnly` AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS

Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich jedes AWS-Konto Root-Benutzers. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.

- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten, einschließlich des AWS-Konto Root-Benutzers, auswirken, unabhängig davon, ob diese zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Logik zur Richtlinienbewertung.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder AWS .

```
{
```

```
    "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": "*"
        }
    ]
}
```

Identitätsbasierte Richtlinien für Amazon SNS

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon SNS

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Ressourcenbasierte Richtlinien in Amazon ECS

Unterstützt ressourcenbasierte Richtlinien	Ja

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM-Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwenden von identitätsbasierten Richtlinien mit Amazon SNS

Amazon Simple Notification Service ist in AWS Identity and Access Management (IAM) integriert, sodass Sie angeben können, welche Amazon SNS SNS-Aktionen ein Benutzer in Ihrem Umfeld mit Amazon SNS SNS-Ressourcen ausführen AWS-Konto kann. Sie können ein bestimmtes Thema in der Richtlinie festlegen. Sie können z. B. beim Erstellen einer IAM-Richtlinie Variablen verwenden, um bestimmten Benutzern in Ihrer Organisation die Verwendung der `Publish`-Aktion für bestimmte Themen in Ihrem AWS-Konto zu ermöglichen. Weitere Informationen finden Sie unter Richtlinienvariablen im Handbuch Verwenden von IAM.

> **Important**
>
> Durch die Nutzung von IAM mit Amazon SNS ändert sich die Verwendung von Amazon SNS nicht. Es gibt keine geänderten Amazon SNS-Aktionen und keine neuen -Aktionen im Zusammenhang mit Benutzern und Zugriffskontrolle.

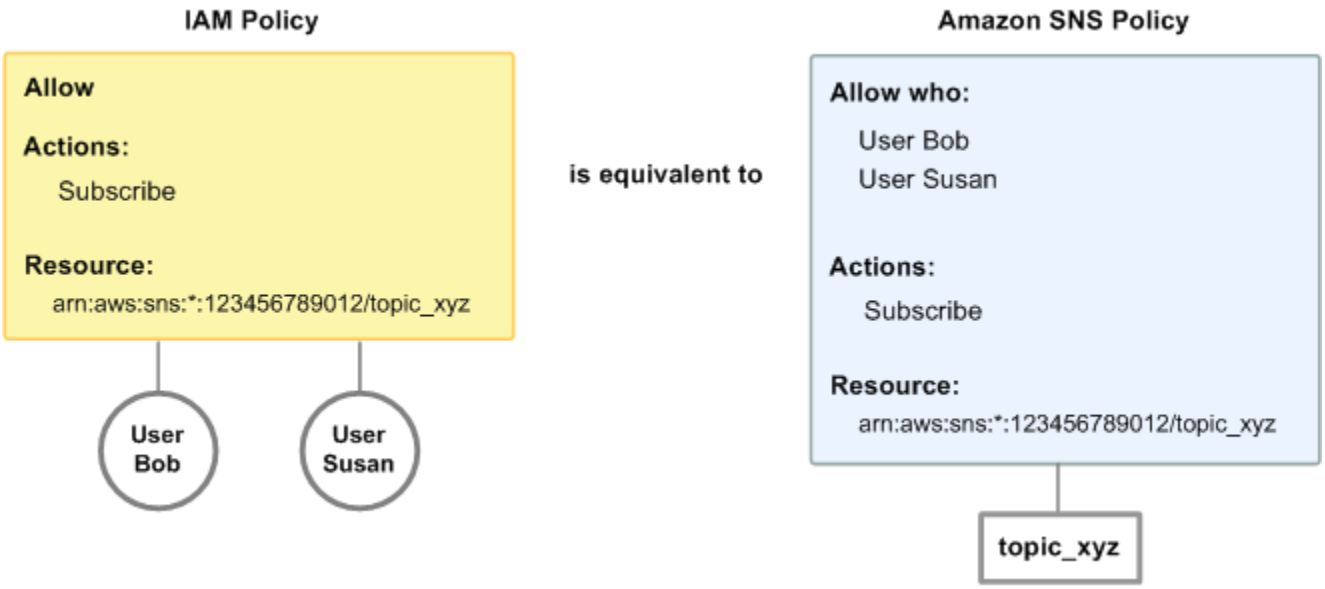
Beispiele von Richtlinien, die Amazon SNS Aktionen und Ressourcen abdecken, finden Sie unter Beispielrichtlinien für Amazon SNS.

Gemeinsame Nutzung von IAM- und Amazon SNS-Richtlinien

Sie verwenden eine IAM-Richtlinie, um den Zugriff des Benutzers auf Amazon SNS-Aktionen und -Themen einzuschränken. Eine IAM-Richtlinie kann den Zugriff nur auf Benutzer in Ihrem AWS Konto beschränken, nicht auf andere AWS-Konten.

Sie können eine Amazon SNS-Richtlinie mit einem bestimmten Thema verwenden, um einzuschränken, wer mit diesem Thema arbeiten kann (z. B., wer Nachrichten veröffentlichen kann, wer es abonnieren kann usw.). Amazon SNS SNS-Richtlinien können anderen AWS-Konten oder Benutzern innerhalb Ihres eigenen AWS-Konto Bereichs Zugriff gewähren.

Um Ihren Benutzern Berechtigungen für Ihre Amazon SNS-Themen zu gewähren, können Sie IAM-Richtlinien, Amazon SNS-Richtlinien oder beide verwenden. In den meisten Fällen erzielen Sie mit beiden Systemen dieselben Ergebnisse. Die folgende Abbildung zeigt eine IAM-Richtlinie und eine entsprechende Amazon SNS-Richtlinie. Die IAM-Richtlinie ermöglicht die Amazon SNS `Subscribe` SNS-Aktion für das Thema topic_xyz in Ihrem AWS-Konto Die IAM-Richtlinie ist den Benutzern Bob und Susan zugeordnet (was bedeutet, dass Bob und Susan über die in der Richtlinie angegebenen Berechtigungen verfügen). Die Amazon SNS-Richtlinie bietet Bob und Susan ebenfalls die Berechtigung für den Zugriff auf `Subscribe` für "topic_xyz".



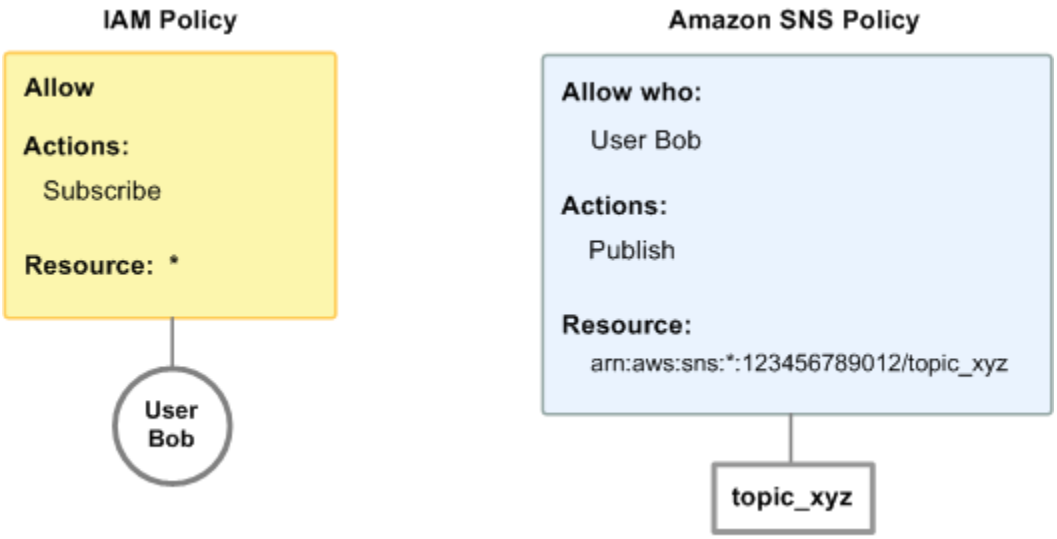
> **Note**
>
> Das vorherige Beispiel veranschaulicht einfache Richtlinien ohne Bedingungen. Sie können eine bestimmte Bedingung in einer der Richtlinien angeben und erhalten dasselbe Ergebnis.

Es gibt einen Unterschied zwischen AWS IAM- und Amazon SNS SNS-Richtlinien: Mit dem Amazon SNS SNS-Richtliniensystem können Sie anderen eine Genehmigung erteilen AWS-Konten, während dies bei der IAM-Richtlinie nicht der Fall ist.

Sie entscheiden je nach Ihren Anforderungen, wie Sie beide Systeme zum Verwalten von Berechtigungen verwenden. Die folgenden Beispiele zeigen, wie die beiden Richtliniensysteme zusammenarbeiten.

Example 1

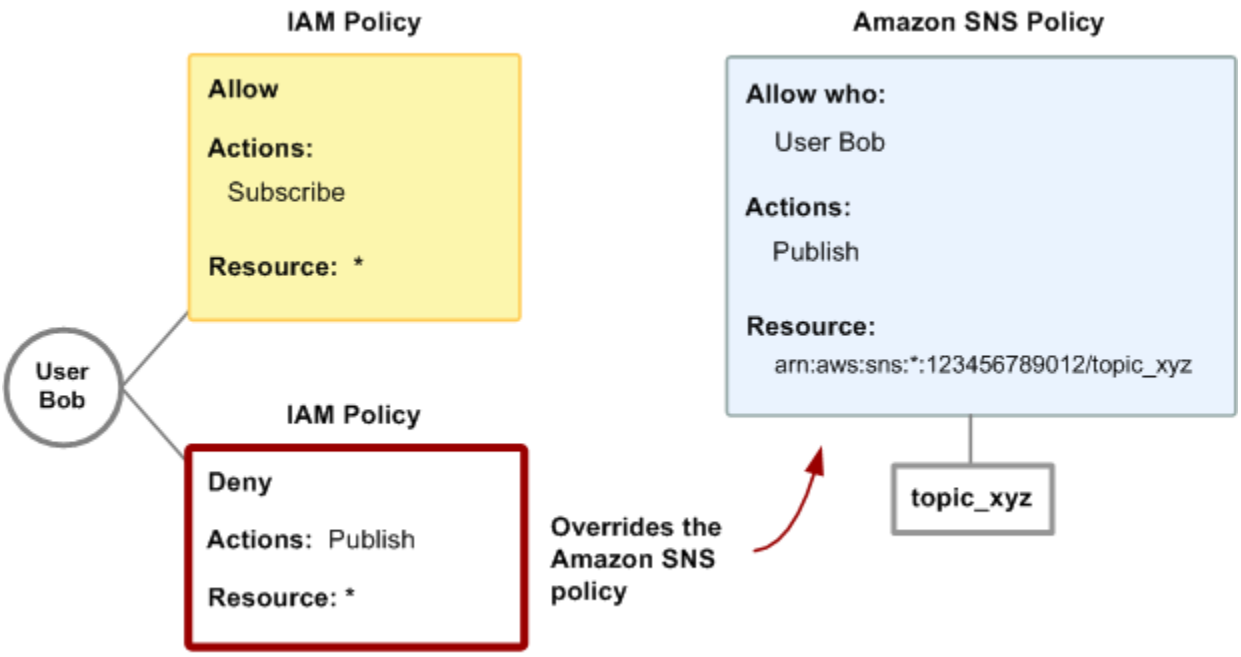
In diesem Beispiel wird eine IAM-Richtlinie und eine Amazon SNS-Richtlinie auf Bob angewendet. Die IAM-Richtlinie gewährt ihm die Erlaubnis für `Subscribe` jedes der Themen, wohingegen die AWS-Konto Amazon SNS SNS-Richtlinie ihm die Erlaubnis zur Verwendung `Publish` zu einem bestimmten Thema (topic_xyz) gewährt. Das folgende Diagramm verdeutlicht das Konzept.



Wenn Bob eine Anfrage zum Abonnieren eines beliebigen Themas im AWS Konto senden würde, würde die IAM-Richtlinie die Aktion zulassen. Wenn Bob eine Anforderung zum Veröffentlichen einer Nachricht an „topic_xyz“ sendet, lässt die Amazon-SNS-Richtlinie diese Aktion zu.

Example 2

In diesem Beispiel bauen wir auf Beispiel 1 auf (wobei Bob zwei für ihn geltende Richtlinien hat). Angenommen, Bob veröffentlicht Nachrichten für das Thema „topic_xyz“, und sollte daran gehindert werden. In diesem Fall entfernen Sie seine Möglichkeit zur Veröffentlichung für diese Themen vollständig. Der einfachste Weg ist, eine IAM-Richtlinie hinzuzufügen, die ihm den Zugriff auf die `Publish`-Aktionen für alle Themen verweigert. Diese dritte Richtlinie überschreibt die Amazon SNS-Richtlinie, die ihm ursprünglich die Berechtigung zur Veröffentlichung in das Thema „topic_xyz“ gewährt hat. Eine explizite Verweigerung überschreibt grundsätzlich eine gewährte Berechtigung (weitere Informationen über die Logik der Richtlinienauswertung finden Sie unter Auswertungslogik). Das folgende Diagramm verdeutlicht das Konzept.



Beispiele von Richtlinien, die Amazon SNS-Aktionen und -Ressourcen abdecken, finden Sie unter Beispielrichtlinien für Amazon SNS.

Amazon SNS Ressourcen - ARN Format

Bei Amazon SNS, können Sie in einer Richtlinie nur den Ressourcentyp „Thema“ festlegen. Im Folgenden finden Sie das Amazon-Ressourcenname (ARN)-Format für Themen:

```
arn:aws:sns:region:account_ID:topic_name
```

Weitere Informationen zu ARNs finden Sie ARNsim IAM-Benutzerhandbuch.

Example

Im Folgenden finden Sie einen ARN für ein Thema, das MyTopic in der Region us-east-2 benannt ist und zu AWS-Konto 123456789012 gehört.

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

Example

Wenn Sie MyTopic in jeder der verschiedenen Regionen, die Amazon SNS unterstützt, ein Thema benannt hätten, könnten Sie die Themen mit dem folgenden ARN angeben.

```
arn:aws:sns:*:123456789012:MyTopic
```

Sie können im Namen des Themas die Platzhalterzeichen * und ? verwenden. Der folgende Namen könnte z. B. alle Themen bezeichnen, die von Bob erstellt wurden und den Präfix `bob_`. haben.

```
arn:aws:sns:*:123456789012:bob_*
```

Zur Vereinfachung gibt Amazon SNS beim Erstellen eines Themas den ARN in der Antwort zurück.

Amazon SNS API-Aktionen

In einer IAM-Richtlinie können Sie eine beliebige, von Amazon SNS unterstützte Aktion angeben. Die Aktionen `ConfirmSubscription` und `Unsubscribe` erfordern jedoch keine Authentifizierung. Dies bedeutet, dass &IAM; die Benutzer auch dann nicht beim Zugriff beschränkt, wenn Sie diese Aktionen in einer Richtlinie festlegen.

Jeder in einer Richtlinie angegebenen Aktion muss die Zeichenfolge `sns:` in Kleinbuchstaben vorangestellt werden. Zum Angeben von allen Amazon SNS -Aktionen würden Sie beispielsweise `sns:*` verwenden. Eine Liste der Aktionen finden Sie unterAmazon Simple Notification Service-API-Referenz.

Amazon SNS Richtlinienschlüssel

Amazon SNS implementiert die folgenden AWS allgemeinen Richtlinienschlüssel sowie einige dienstspezifische Schlüssel.

Eine Liste der von ihnen unterstützten Bedingungsschlüssel finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS-Services im IAM-Benutzerhandbuch. AWS-Service Eine Liste der Bedingungsschlüssel, die in mehreren Fällen verwendet werden können AWS-Services, finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Amazon SNS verwendet die folgenden servicespezifischen Schlüssel. Verwenden Sie diese Schlüssel in Richtlinien, die den Zugriff auf `Subscribe`-Anfragen beschränken.

- sns:Endpoint: Die URL, die E-Mail-Adresse oder der ARN von einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um den Zugriff auf bestimmte Endpunkte (z. B. *@yourcompany.com) zu beschränken.

- sns:Protocol: Der `protocol`-Wert einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um die Veröffentlichung auf bestimmte Bereitstellungsprotokolle (z. B. HTTPS) zu beschränken.

Beispielrichtlinien für Amazon SNS

Dieser Abschnitt zeigt mehrere einfache Richtlinien zum Steuern des Benutzerzugriffs auf Amazon SNS .

> **Note**
>
> In Zukunft können in Amazon SNS neue Aktionen hinzugefügt werden, die logisch in einer der folgenden Richtlinien eingeschlossen werden müssen, basierend auf den formulierten Zielen der Richtlinie.

Example 1: Zulassen der Erstellung und Verwaltung von Themen für eine Gruppe

In diesem Beispiel erstellen wir eine Richtlinie, die Zugriff auf `CreateTopic`, `ListTopics`, `SetTopicAttributes` und `DeleteTopic` gewährt.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: Zulassen der Veröffentlichung von Nachrichten zu einem bestimmten Thema für die IT-Gruppe

In diesem Beispiel erstellen wir eine Gruppe für die IT, die eine Richtlinie zuweist, die Zugriff auf `Publish` für das definierte Thema gewährt.

```
{
  "Statement": [{
```

```
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3: Geben Sie Benutzern die AWS-Konto Möglichkeit, Themen zu abonnieren

In diesem Beispiel erstellen wir eine Richtlinie mit der Bedingungen für die Zeichenfolgenübereinstimmungen für die Richtlinienschlüssel `sns:Protocol` und `sns:Endpoint`, die den Zugriff auf die `Subscribe`-Aktion gewährt.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "sns:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: Zulassen der Veröffentlichung von Nachrichten zu einem bestimmten Thema für einen Partner

Sie können eine Amazon SNS-Richtlinie oder eine IAM-Richtlinie verwenden, um einem Partner das Veröffentlichen zu einem bestimmten Thema zu gewähren. Wenn Ihr Partner über eine verfügt AWS-Konto, ist es möglicherweise einfacher, eine Amazon SNS SNS-Richtlinie zu verwenden. Jeder Mitarbeiter des Partners, der über die AWS Sicherheitsnachweise verfügt, könnte jedoch Nachrichten zu diesem Thema veröffentlichen. In diesem Beispiel wird davon ausgegangen, dass Sie den Zugriff auf eine bestimmte Person (oder Anwendung) begrenzen möchten. Zu diesem Zweck müssen Sie den Partner wie einen Benutzer innerhalb Ihres eigenen Unternehmens behandeln. Verwenden Sie eine &IAM;-Richtlinie anstelle einer Amazon SNS-Richtlinie.

In diesem Beispiel erstellen wir eine Gruppe mit dem Namen WidgetCo , die das Partnerunternehmen repräsentiert. Wir erstellen einen Benutzer für die bestimmte Person (oder Anwendung) im Partnerunternehmen, die Zugriff benötigt, und fügen den Benutzer dann der Gruppe hinzu.

Anschließend fügen wir eine Richtlinie hinzu, die der Gruppe `Publish` Zugriff auf das angegebene Thema gewährt WidgetPartnerTopic.

Wir möchten auch verhindern, dass die WidgetCo Gruppe etwas anderes mit Themen macht. Deshalb fügen wir eine Erklärung hinzu, die allen Amazon SNS SNS-Aktionen die Genehmigung verweigert, außer zu anderen `Publish` Themen als. WidgetPartnerTopic Dies ist nur erforderlich, wenn eine umfassendere Richtlinie im System verwendet wird, die Benutzern einen breiten Zugriff auf Amazon SNS-Ressourcen ermöglicht.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Verwaltung benutzerdefinierter Amazon SNS IAM-Richtlinien

Mit benutzerdefinierten IAM-Richtlinien können Sie Berechtigungen für einzelne IAM-Benutzer, -Gruppen oder -Rollen angeben und so den Zugriff auf bestimmte Ressourcen und Aktionen gewähren oder einschränken. AWS Bei der Verwaltung von Amazon SNS SNS-Ressourcen ermöglichen Ihnen benutzerdefinierte IAM-Richtlinien, die Zugriffsberechtigungen an die Sicherheits- und Betriebsanforderungen Ihres Unternehmens anzupassen.

Gehen Sie wie folgt vor, um benutzerdefinierte IAM-Richtlinien für Amazon SNS zu verwalten:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich Richtlinien aus.

3. Um eine neue benutzerdefinierte IAM-Richtlinie zu erstellen, wählen Sie Richtlinie erstellen und dann. SNS Um eine bestehende Richtlinie zu bearbeiten, wählen Sie die Richtlinie aus der Liste aus und klicken Sie auf Richtlinie bearbeiten.
4. Definieren Sie im Richtlinieneditor die Berechtigungen für den Zugriff auf Amazon SNS SNS-Ressourcen. Sie können Aktionen, Ressourcen und Bedingungen auf der Grundlage Ihrer spezifischen Anforderungen angeben.
5. Um Berechtigungen für Amazon SNS SNS-Aktionen zu erteilen, nehmen Sie relevante Amazon SNS SNS-Aktionen wie `sns:Publishsns:Subscribe`, und `sns:DeleteTopic` in Ihre IAM-Richtlinie auf. Definieren Sie den ARN (Amazon Resource Name) der Amazon SNS SNS-Themen, für die die Berechtigungen gelten.
6. Geben Sie die IAM-Benutzer, -Gruppen oder -Rollen an, denen die Richtlinie zugeordnet werden soll. Sie können die Richtlinie direkt an IAM-Benutzer oder -Gruppen anhängen oder sie IAM-Rollen zuordnen, die von AWS-Services unseren Anwendungen verwendet werden.
7. Überprüfen Sie die IAM-Richtlinienkonfiguration, um sicherzustellen, dass sie Ihren Anforderungen an die Zugriffskontrolle entspricht. Speichern Sie die Richtlinienänderungen nach der Überprüfung.
8. Fügen Sie die benutzerdefinierte IAM-Richtlinie den entsprechenden IAM-Benutzern, -Gruppen oder -Rollen in Ihrem hinzu. AWS-Konto Dadurch erhalten sie die in der Richtlinie für die Verwaltung von Amazon SNS SNS-Ressourcen definierten Berechtigungen.

Verwenden von temporären Sicherheitsanmeldeinformationen mit Amazon SNS

AWS Identity and Access Management (IAM) ermöglicht es Ihnen, Benutzern und Anwendungen, die Zugriff auf Ihre AWS Ressourcen benötigen, temporäre Sicherheitsanmeldedaten zu gewähren. Diese temporären Sicherheitsanmeldedaten werden hauptsächlich für IAM-Rollen und Verbundzugriff über branchenübliche Protokolle wie SAML und OpenID Connect (OIDC) verwendet.

Um den Zugriff auf AWS Ressourcen effektiv zu verwalten, ist es wichtig, die folgenden Schlüsselkonzepte zu verstehen:

- IAM-Rollen — Rollen werden verwendet, um den Zugriff auf Ressourcen zu AWS delegieren. Rollen können von Entitäten wie EC2 Amazon-Instances, Lambda-Funktionen oder Benutzern von anderen AWS-Kontenübernommen werden.

- Verbundbenutzer — Dies sind Benutzer, die über externe Identitätsanbieter (IdPs) mithilfe von SAML oder OIDC authentifiziert wurden. Der Verbundzugriff wird für menschliche Benutzer empfohlen, während IAM-Rollen für Softwareanwendungen verwendet werden sollten.
- Roles Anywhere — Für externe Anwendungen, die AWS Zugriff benötigen, können Sie IAM Roles Anywhere verwenden, um den Zugriff sicher zu verwalten, ohne langfristige Anmeldeinformationen erstellen zu müssen.

Sie können temporäre Sicherheitsanmeldedaten verwenden, um Anfragen an Amazon SNS zu stellen. Die Bibliotheken SDKs und die API berechnen anhand dieser Anmeldeinformationen die erforderliche Signatur, um Ihre Anfragen zu authentifizieren. Anfragen mit abgelaufenen Anmeldeinformationen werden von Amazon SNS abgelehnt.

Weitere Informationen zu temporären Sicherheitsanmeldedaten finden Sie unter Verwenden von IAM-Rollen und Bereitstellen des Zugriffs für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.

Example Beispiel für eine HTTPS-Anfrage

Das folgende Beispiel zeigt, wie eine Amazon SNS SNS-Anfrage mithilfe temporärer Sicherheitsanmeldedaten authentifiziert wird, die von AWS Security Token Service (STS) abgerufen wurden.

```
https://sns.us-east-2.amazonaws.com/
?Action=CreateTopic
&Name=My-Topic
&SignatureVersion=4
&SignatureMethod=AWS4-HMAC-SHA256
&Timestamp=2023-07-05T12:00:00Z
&X-Amz-Security-Token=SecurityTokenValue
&X-Amz-Date=20230705T120000Z
&X-Amz-Credential=<your-access-key-id>/20230705/us-east-2/sns/aws4_request
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Schritte zur Authentifizierung der Anfrage

1. Temporäre Sicherheitsanmeldedaten abrufen — Verwenden Sie diese AWS STS Option, um eine Rolle zu übernehmen oder Verbundbenutzeranmeldeinformationen abzurufen.

Dadurch erhalten Sie eine Zugriffsschlüssel-ID, einen geheimen Zugriffsschlüssel und ein Sicherheitstoken.

2. Konstruieren Sie die Anfrage — Geben Sie die erforderlichen Parameter für Ihre Amazon SNS SNS-Aktion an (z. B. CreateTopic) und stellen Sie sicher, dass Sie HTTPS für die sichere Kommunikation verwenden.
3. Signieren Sie die Anfrage — Verwenden Sie den AWS Signature Version 4-Prozess, um Ihre Anfrage zu signieren. Dazu gehört die Erstellung einer kanonischen Anfrage und die string-to-sign anschließende Berechnung der Signatur. Weitere Informationen zu AWS Signature Version 4 finden Sie unter Verwenden der Signatur mit Signature Version 4 im Amazon EBS-Benutzerhandbuch.
4. Anfrage senden — Fügen Sie das X-Amz-Security-Token in Ihren Anforderungsheader ein, um die temporären Sicherheitsanmeldedaten an Amazon SNS weiterzuleiten.

Amazon SNS-API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen

Die folgende Liste enthält Informationen zur Amazon SNS-Implementierung der Zugriffskontrolle:

- Jede Richtlinie darf nur ein einzelnes Thema betreffen (bei der Erstellung einer Richtlinie dürfen Sie keine Anweisungen zu anderen Themen schreiben).
- Jede Richtlinie muss eine eindeutige `Id` haben.
- Jede Anweisung in einer Richtlinie muss eine eindeutige Anweisungs-`sid` haben.

Richtlinienkontingente

In der folgenden Tabelle sind die Kontingente für eine Richtlinienanweisung aufgeführt.

Name	Höchstkontingent
Bytes	30 KB
Anweisungen	100
Prinzipale	1 bis 200 (0 ist ungültig)

Name	Höchstkontingent
Ressource	1 (0 ist ungültig. Der Wert muss mit dem ARN des Themas der Richtlinie übereinstimmen.)

Gültige Amazon SNS-Richtlinienaktionen

Amazon SNS unterstützt die in der folgenden Tabelle aufgeführten Aktionen.

Aktion	Beschreibung
sns: AddPermission	Erteilt die Berechtigung zum Hinzufügen von Berechtigungen zur Themarichtlinie.
sns: DeleteTopic	Erteilt die Berechtigung zum Löschen eines Themas.
sns: GetDataProtectionPolicy	Erteilt die Berechtigung zum Abrufen der Datenschutzrichtlinie eines Themas.
sns: GetTopicAttributes	Erteilt die Berechtigung zum Empfangen aller Themenattribute.
sns: ListSubscriptionsByTopic	Erteilt die Berechtigung zum Abrufen aller Abonnements für ein bestimmtes Thema.
sns: ListTagsForResource	Gewährt die Berechtigung zum Auflisten aller Tags, die einem bestimmten Thema hinzugefügt wurden.
sns:Publish	Erteilt die Berechtigung zum Batch-Veröffentlichen in einem Thema oder Endpunkt. Weitere Informationen finden Sie unter Veröffentlichen und PublishBatchin der Amazon Simple Notificat ion Service API-Referenz.
sns: PutDataProtectionPolicy	Erteilt die Berechtigung zum Festlegen der Datenschutzrichtlinie eines Themas.
sns: RemovePermission	Erteilt die Berechtigung zum Entfernen von Berechtigungen in der Themarichtlinie.

Aktion	Beschreibung
sns: SetTopicAttributes	Erteilt die Berechtigung zum Festlegen der Attribute eines Themas.
sns:Subscribe	Erteilt die Berechtigung zum Abonnieren eines Themas.

Servicespezifische Schlüssel

Amazon SNS verwendet die folgenden servicespezifischen Schlüssel. Sie können diese in Richtlinien verwenden, die den Zugriff auf `Subscribe`-Anfragen beschränken.

- sns:Endpoint: Die URL, die E-Mail-Adresse oder der ARN von einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um den Zugriff auf bestimmte Endpunkte (z. B. *@beispiel.de) zu beschränken.
- sns:Protocol: Der `protocol`-Wert einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um die Veröffentlichung auf bestimmte Bereitstellungsprotokolle (z. B. HTTPS) zu beschränken.

> **Important**
>
> Wenn Sie eine Richtlinie verwenden, um den Zugriff nach sns:Endpoint zu beschränken, beachten Sie, dass künftig DNS-Probleme die Namensauflösung des Endpunkts beeinträchtigen können.

Beheben von Identitäts- und Zugriffsfehlern bei Amazon Simple Notification Service

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von Amazon SNS und IAM auftreten können.

Ich bin nicht autorisiert, eine Aktion in Amazon SNS auszuführen

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen einer Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um die Aktion durchführen zu können.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `sns:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Mateo-Richtlinie aktualisiert werden, damit er mit der `sns:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon SNS übergeben zu können.

In einigen AWS-Services Fällen können Sie eine bestehende Rolle an diesen Dienst übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon SNS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon SNS SNS-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon SNS diese Funktionen unterstützt, finden Sie unter So funktioniert Amazon SNS mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen in AWS-Konten Ihrem Besitz gewähren können, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , dem Sie gehören.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Protokollierung und Überwachung in Amazon SNS

Mit Amazon SNS können Sie Messaging-Aktivitäten verfolgen und überwachen, indem Sie API-Aufrufe protokollieren CloudTrail und Themen mit CloudWatch überwachen. Diese Tools helfen Ihnen dabei, Einblicke in die Nachrichtenzustellung zu gewinnen, Probleme zu beheben und die Integrität Ihrer Messaging-Workflows sicherzustellen. In diesem Thema wird Folgendes behandelt:

- Protokollieren AWS von SNS-API-Aufrufen mit AWS CloudTrail. Diese Protokollierung ermöglicht es Ihnen, die Aktionen zu verfolgen, die an Ihren Amazon SNS SNS-Themen durchgeführt wurden, wie z. B. Themenerstellung, Abonnementverwaltung und Nachrichtenveröffentlichung. Durch die Analyse von CloudTrail Protokollen können Sie ermitteln, wer bestimmte API-Anfragen gestellt

hat und wann diese Anfragen gestellt wurden. So können Sie Ihre Nutzung von Amazon SNS überprüfen und Fehler beheben.

- Überwachung von Amazon SNS SNS-Themen mit CloudWatch. CloudWatch bietet Kennzahlen, mit denen Sie die Leistung und den Zustand Ihrer Amazon SNS SNS-Themen in Echtzeit beobachten können. Richten Sie Alarme auf der Grundlage dieser Kennzahlen ein, sodass Sie umgehend auf Anomalien wie Zustellungsfehler oder hohe Nachrichtenlatenz reagieren können. Diese Überwachungsfunktion stellt sicher, dass Sie die Zuverlässigkeit Ihres SNS-basierten Nachrichtensystems aufrechterhalten können, indem Sie potenzielle Probleme proaktiv angehen.

Protokollieren AWS von SNS-API-Aufrufen mit AWS CloudTrail

AWS SNS ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt. AWS-Service CloudTrail erfasst alle API-Aufrufe für SNS als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der SNS-Konsole und Codeaufrufen für die SNS-API-Operationen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an SNS gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS Management Console sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter Erstellen eines Trails für Ihr AWS-Konto und Erstellen eines Trails für eine Organisation im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3 – Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise.

SNS-Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden (z. B. Lesen oder Schreiben in ein Amazon-S3-Objekt). Sie

werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Protokolliert standardmäßig CloudTrail keine Datenereignisse. Der CloudTrail Ereignisverlauf zeichnet keine Datenereignisse auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preisgestaltung.

Sie können Datenereignisse für die SNS-Ressourcentypen mithilfe der CloudTrail Konsole oder CloudTrail API-Operationen protokollieren. AWS CLI Weitere Informationen zum Protokollieren von Datenereignissen finden Sie unter Protokollieren von Datenereignissen mit dem AWS Management Console und Protokollieren von Datenereignissen mit dem AWS Command Line Interface im AWS CloudTrail -Benutzerhandbuch.

In der folgenden Tabelle sind die SNS-Ressourcentypen aufgeführt, für die Sie Datenereignisse protokollieren können. In der Spalte Datenereignistyp (Konsole) wird der Wert angezeigt, der aus der Liste Datenereignistyp auf der CloudTrail Konsole ausgewählt werden kann. In der Wertspalte resources.type wird der `resources.type` Wert angezeigt, den Sie bei der Konfiguration erweiterter Event-Selektoren mithilfe von oder angeben würden. AWS CLI CloudTrail APIs In der CloudTrail Spalte APIs Protokollierte Daten werden die API-Aufrufe angezeigt, die CloudTrail für den Ressourcentyp protokolliert wurden.

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
SNS-Thema	`AWS::SNS::Topic`	• `Publish` • `PublishBatch`
SNS-Plattformendpunkt	`AWS::SNS::PlatformEndpoint`	• Publish Weitere Informationen finden Sie `AdvancedEventSelector` in der AWS CloudTrail API-Referenz.

> **Note**
>
> Der SNS-Ressourcentyp `AWS::SNS::PhoneNumber` wird nicht protokolliert CloudTrail.

Sie können erweiterte Event-Selektoren so konfigurieren, dass sie nach den Feldern `eventName`, `readOnly` und `resources.ARN` filtern, sodass nur die Ereignisse protokolliert werden, die für Sie wichtig sind. Weitere Informationen zu diesen Feldern finden Sie unter AdvancedFieldSelector in der API-Referenz zu AWS CloudTrail

Informationen zum Protokollieren von Datenereignissen finden Sie unter Protokollieren von Datenereignissen mit dem AWS Management Console und Protokollieren von Datenereignissen mit dem AWS CLI im CloudTrail Benutzerhandbuch.

SNS-Verwaltungsereignisse in CloudTrail

Verwaltungsereignisse enthalten Informationen zu Verwaltungsvorgängen, die an Ressourcen in Ihrem AWS-Konto ausgeführt werden. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. CloudTrail Protokolliert standardmäßig Verwaltungsereignisse.

AWS SNS protokolliert die folgenden Vorgänge auf der SNS-Steuerungsebene CloudTrail als Verwaltungsereignisse.

- `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**
>
> Wenn Sie nicht bei Amazon Web Services angemeldet sind (nicht authentifizierter Modus) und entweder die `Unsubscribe`Aktionen `ConfirmSubscription`oder aufgerufen werden, werden sie nicht angemeldet. CloudTrail Ebenso wird, wenn Sie den angegebenen Link in der E-Mail-Benachrichtigung wählen, um ein ausstehendes Abonnement für ein Thema zu bestätigen, die Aktion `ConfirmSubscription` im nicht authentifizierten Modus aufgerufen. In diesem Beispiel würde die `ConfirmSubscription` Aktion nicht protokolliert werden. CloudTrail

Beispiele für SNS-Ereignisse

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das die **`DeleteTopic`** Aktionen **`ListTopicsCreateTopic`**, und demonstriert.

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

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das die **`Publish`** Aktion demonstriert.

```
{
  "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"
  }
}
```

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das die **`PublishBatch`** Aktion demonstriert.

```
{
  "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"
  }
}
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter CloudTrailDatensatzinhalte.

Überwachung von Amazon SNS SNS-Themen mit CloudWatch

Amazon SNS und Amazon CloudWatch sind integriert, sodass Sie Metriken für jede aktive Amazon SNS SNS-Benachrichtigung sammeln, anzeigen und analysieren können. Sobald Sie CloudWatch für Amazon SNS konfiguriert haben, können Sie einen besseren Einblick in die Leistung Ihrer Amazon SNS SNS-Themen, Push-Benachrichtigungen und SMS-Lieferungen erhalten. Sie können z. B. einen Alarm festlegen, sodass Ihnen eine E-Mail-Benachrichtigung gesendet wird, sobald ein bestimmter Schwellenwert für eine Amazon-SNS-Metrik erreicht wird, wie z. B. `NumberOfNotificationsFailed`. Eine Liste aller Metriken, an die Amazon SNS sendet

CloudWatch, finden Sie unterAmazon SNS-Metriken. Weitere Informationen über Amazon-SNS-Push-Benachrichtigungen finden Sie unter Senden von mobilen Push-Benachrichtigungen mit Amazon SNS.

> **Note**
>
> Die Metriken, die Sie CloudWatch für Ihre Amazon SNS SNS-Themen konfigurieren, werden automatisch erfasst und in Intervallen von 1 CloudWatch Minute abgerufen. Diese Metriken werden zu allen Themen erfasst, die den CloudWatch Richtlinien für aktive Aktivitäten entsprechen. Ein Thema gilt bis zu sechs Stunden nach der letzten Aktivität (d. h. jedem API-Aufruf) zu diesem Thema als aktiv. CloudWatch
> Für die unter angegebenen Amazon SNS SNS-Metriken fallen keine Gebühren an CloudWatch. Sie werden als Teil des Amazon SNS SNS-Service bereitgestellt.

CloudWatch Metriken für Amazon SNS anzeigen

Sie können Metriken für Amazon SNS mithilfe der CloudWatch Konsole, CloudWatch der eigenen Befehlszeilenschnittstelle (CLI) oder programmgesteuert mithilfe der API überwachen. CloudWatch Die folgenden Verfahren zeigen Ihnen, wie Sie mithilfe der AWS Management Console auf die Metriken zugreifen können.

Um Metriken mit der Konsole anzuzeigen CloudWatch

1. Melden Sie sich an der CloudWatch-Konsole an.
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Wählen Sie auf der Registerkarte All metrics (Alle Metriken) die Option SNS und danach eine der folgenden Dimensionen aus:
   - Country, SMS Type (Land, SMS-Typ)
   - PhoneNumber
   - Topics Metrics (Themen-Metriken)
   - Metrics with no dimensions (Metriken ohne Dimensionen)
4. Um weitere Details anzuzeigen, wählen Sie ein bestimmtes Element aus. Wenn Sie beispielsweise Topic Metrics auswählen und dann auswählen NumberOfMessagesPublished, wird die durchschnittliche Anzahl veröffentlichter Amazon SNS SNS-Nachrichten für einen Zeitraum von 1 Minute über einen Zeitraum von 6 Stunden angezeigt.

5. Wenn Sie die Amazon-SNS-Nutzungsmetriken anzeigen möchten, wählen Sie auf der Registerkarte All metrics (Alle Metriken) die Option Usage (Nutzung) und die target Amazon SNS usage metric (Amazon-SNS-Zielmetrik) aus (z. B. `NumberOfMessagesPublishedPerAccount`).

CloudWatch Alarme für Amazon SNS SNS-Metriken einrichten

CloudWatch ermöglicht es Ihnen auch, Alarme einzustellen, wenn ein Schwellenwert für eine Metrik erreicht wird. Sie könnten beispielsweise einen Alarm für die Metrik einrichten, sodass NumberOfNotificationsFailed, wenn der angegebene Schwellenwert innerhalb des Probenahmezeitraums erreicht wird, eine E-Mail-Benachrichtigung gesendet wird, um Sie über das Ereignis zu informieren.

Um Alarme über die CloudWatch Konsole einzustellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Alarme und dann die Schaltfläche Alarm erstellen aus. Dadurch wird der Assistent Alarm erstellen gestartet.
3. Scrollen Sie durch die Amazon-SNS-Metriken, um die Metrik zu finden, auf die Sie einen Alarm setzen möchten. Wählen Sie die Metrik aus, um einen Alarm zu erstellen, und wählen Sie dann Fortfahren.
4. Geben Sie für die Metrik die Werte Name, Beschreibung, Schwellenwert und Zeit an und wählen Sie dann Fortfahren.
5. Wählen Sie als Alarmzustand Alarm aus. Wenn Sie Ihnen eine E-Mail senden CloudWatch möchten, wenn der Alarmstatus erreicht ist, wählen Sie entweder ein vorhandenes Amazon SNS SNS-Thema oder wählen Sie Neues E-Mail-Thema erstellen. Wenn Sie Neues E-Mail-Thema erstellen wählen, können Sie den Namen und die E-Mail Adressen für ein neues Thema einrichten. Diese Liste wird gespeichert und im Dropdown-Menü für künftige Alarme angezeigt. Klicken Sie auf Fortfahren.

> **Note**
>
> Wenn Sie Neues E-Mail-Thema erstellen verwenden, um ein neues Amazon SNS-Thema zu erstellen, müssen die E-Mail Adressen überprüft werden, bevor die Empfänger Benachrichtigungen erhalten können. E-Mail Nachrichten werden nur gesendet, wenn der Alarm in einen Alarmzustand wechselt. Wenn es zu dieser Änderung des

Alarmzustands kommt, bevor die E-Mail Adressen überprüft wurden, erhalten die Empfänger keine Benachrichtigung.

6. An diesem Punkt bietet Ihnen der Assistent Alarm erstellen die Möglichkeit, den Alarm, den Sie gerade erstellen, zu überprüfen. Wenn Sie Änderungen vornehmen müssen, können Sie die Links Bearbeiten auf der rechten Seite verwenden. Wenn Sie zufrieden sind, wählen Sie Alarm erstellen.

Weitere Informationen zur Verwendung CloudWatch und zu Alarmen finden Sie in der CloudWatchDokumentation.

Amazon SNS-Metriken

Amazon SNS sendet die folgenden Messwerte an CloudWatch.

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfMessagesPublished`	Die Anzahl der zu Ihren Amazon SNS-Themen veröffentlichten Mitteilungen. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe
AWS/SNS	`NumberOfNotificationsDelivered`	Die Anzahl der Mitteilungen, die von Ihren Amazon SNS-Themen erfolgreich an Endpunkte übermittelt wurden. Damit ein Zustellungsversuch erfolgreich ausgeführt wird, muss das Abonnement des Endpunkts die Mitteilung entgegennehmen. Ein Abonnement akzeptiert eine

Namespace	Metrik	Beschreibung
		Nachricht, wenn a.) eine Filterric htlinie fehlt oder b.) die Filterric htlinie Attribute enthält, die mit denjenigen übereinstimmen, die der Nachricht zugewiesen wurden. Falls das Abonnement die Mitteilun g ablehnt, wird der Zustellun gsversuch für diese Metrik nicht gezählt. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsFailed`	Die Anzahl von Nachrichten, die Amazon SNS nicht zustellen konnte. Bei Amazon SQS-, E-Mail-, SMS- oder mobilen Push-Endpunkten steigt diese Metrik um 1, wenn Amazon SNS die Nachrichtenzustellungsversuche einstellt. Bei HTTP- oder HTTPS-Endpunkten umfasst die Metrik jeden fehlgeschlagenen Zustellversuch, einschließlich Wiederholungen nach dem ursprünglichen Versuch. Für alle anderen Endpunkte steigt die Zahl um 1, wenn die Mitteilung nicht zugestellt wird (unabhängig von der Anzahl der Versuche). Diese Metrik enthält keine Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden. Sie können die Anzahl der Wiederholungen für HTTP-Endpunkte kontrollieren. Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName

Namespace	Metrik	Beschreibung
		Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden. Eine Filterrichtlinie lehnt eine Nachricht ab, wenn die Nachrichtenattribute nicht mit den Richtlinienattributen übereinstimmen. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien für die attributbasierte Filterung abgelehnt wurden. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien für die nutzlastbasierte Filterung abgelehnt wurden. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden, weil die Nachrichtenattribute ungültig sind zum Beispiel, weil der JSON des Attributs falsch formatiert ist. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden, weil die Nachrichten keine Attribute enthalten. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien aufgrund eines für die Filterung ungültigen Nachrichtentexts abgelehnt wurden, zum Beispiel ein ungültiger JSON-Nachrichtentext. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Die Anzahl der Nachrichten, die in eine Warteschlange für unzustellbare Nachrichten verschoben wurden. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Die Anzahl der Nachrichten, die nicht in eine Warteschlange für unzustellbare Nachrichten verschoben werden konnten. Einheiten: Anzahl Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`PublishSize`	Die Größe von veröffentlichten Nachrichten. Einheiten: Byte Gültige Dimensionen: Anwendung PhoneNumber, Plattform und TopicName Gültige Statistiken: Minimum, Maximum, Durchschnitt und Anzahl

Namespace	Metrik	Beschreibung
AWS/SNS	`SMSMonthToDateSpentUSD`	Ihre Gebühren für das Senden von SMS-Nachrichten seit Beginn des aktuellen Kalendermonats. Sie können einen Alarm für diese Kennzahl einrichten, damit Sie informiert werden, wenn Ihre month-to-date Gebühren fast das monatliche SMS-Ausgabenkontingent für Ihr Konto erreichen. Wenn Amazon SNS feststellt, dass durch den Versand einer SMS-Nachricht Kosten entstehen, die dieses Kontingent überschreiten, wird die Veröffentlichung von SMS-Nachrichten innerhalb von Minuten beendet. Weitere Informationen zum Einrichten Ihres monatlichen Ausgabenkontingents für SMS oder zum Anfordern einer Erhöhung des Ausgabenkontingents bei AWS finden Sie unter Einstellungen für SMS-Nachrichten in Amazon SNS einrichten. Einheiten: USD Gültige Abmessungen: Keine Gültige Statistiken: Summe

Namespace	Metrik	Beschreibung
AWS/SNS	`SMSSuccessRate`	Die Rate der erfolgreichen SMS-Nachrichtenzustellungen Einheiten: Anzahl Gültige Abmessungen: PhoneNumber Gültige Statistiken: Summe, Durchschnitt, Datenproben

Dimensionen für Amazon-SNS-Metriken

Amazon Simple Notification Service sendet die folgenden Dimensionen an CloudWatch.

Dimension	Beschreibung
`Application`	Filtert nach Anwendungsobjekten, die eine App und ein Gerät darstellen, die bei einem der unterstützten Push-Benachrichtigungsdienste wie APNs FCM registriert sind.
`Application,Platform`	Filtert nach Anwendungs- und Plattformobjekten, wobei die Plattformobjekte für die unterstützten Push-Benachrichtigungsdienste wie FCM APNs bestimmt sind.
`Country`	Filtert das Zielland oder die Zielregion einer SMS-Mitteilung. Das Land oder die Region werden durch den zugehörigen ISO 3166-1 Alpha-2-Code dargestellt.
`PhoneNumber`	Filtert die Telefonnummer, wenn Sie SMS direkt an eine Telefonnummer (ohne Thema) veröffentlichen.
`Platform`	Filtert nach Plattformobjekten für die Push-Benachrichtigungsdienste wie APNs und FCM.
`TopicName`	Filtert Amazon-SNS-Themennamen.

Dimension	Beschreibung
`SMSType`	Filtert den Nachrichtentyp einer SMS-Mitteilung. Mögliche Werte sind promotional oder transactional.

Amazon-SNS-Nutzungsmetriken

Amazon Simple Notification Service sendet die folgenden Nutzungsmetriken an CloudWatch.

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
AWS/Verwendung	SNS	`ResourceCount`	`NumberOfMessagesPublishedPerAccount`	Ressource	• Die Anzahl der Nachrichten, die zu Ihren Amazon SNS SNS-Themen in Ihrem AWS Konto veröffentlicht wurden. • Einheiten: keine • Gültige Statistiken: Summe
AWS/Verwendung	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Ressource	• Die ungefähre Anzahl der Themen in Ihrem AWS Konto.

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
					• Einheiten: keine • Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt
AWS/Verwendung	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Ressource	• Die ungefähre Anzahl von Filterrichtlinien in Ihrem AWS -Konto. • Einheiten: keine • Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
AWS/Verwendung	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Ressource	• Die ungefähre Anzahl ausstehender Abonnements in Ihrem AWS Konto. • Einheiten: keine • Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
AWS/Verwendung	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber` • `DeleteTopic`	API	• Die Anzahl der API-Aufrufe für die ausgewählte Amazon SNS SNS-API in Ihrem AWS Konto. • Inhalt ist im abschließenden Abschnitt nicht zulässig. Einheiten: keine • Gültige Statistiken: Summe

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
			• `Subscribe` • `Unsubscribe` • `UntagResource` • `VerifySMSSandboxPhoneNumber`		

Compliance-Validierung für Amazon SNS

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Compliance und Governance im Bereich Sicherheit – In diesen Anleitungen für die Lösungsimplementierung werden Überlegungen zur Architektur behandelt. Außerdem werden Schritte für die Bereitstellung von Sicherheits- und Compliance-Features beschrieben.
- Referenz für berechtigte HIPAA-Services – Listet berechtigte HIPAA-Services auf. Nicht alle AWS-Services sind HIPAA-fähig.
- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.

- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Die Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerelementreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Ausfallsicherheit bei Amazon SNS

Die Resilienz von Amazon SNS wird durch die Nutzung der AWS globalen Infrastruktur gewährleistet, die sich um Availability Zones AWS-Regionen dreht. AWS-Regionen bieten physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Diese Architektur ermöglicht einen nahtlosen Failover zwischen Availability Zones ohne Unterbrechung, wodurch Anwendungen und Datenbanken im Vergleich zu herkömmlichen Rechenzentrumsinfrastrukturen von Natur aus fehlertoleranter und skalierbarer sind. Durch die Verwendung von Availability Zones profitieren Amazon SNS SNS-Abonnenten von verbesserter Verfügbarkeit und Zuverlässigkeit, sodass die Nachrichtenzustellung trotz potenzieller Unterbrechungen gewährleistet ist. Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Darüber hinaus können Abonnements für Amazon SNS SNS-Themen mit Wiederholungsversuchen und Warteschlangen für unzustellbare Briefe konfiguriert werden, sodass vorübergehende Ausfälle automatisch behandelt werden und sichergestellt wird, dass Nachrichten zuverlässig ihren Bestimmungsort erreichen.

Amazon SNS unterstützt auch Nachrichtenfilterung und Nachrichtenattribute, sodass Sie Resilienzstrategien an ihre spezifischen Anwendungsfälle anpassen und so die allgemeine Robustheit Ihrer Anwendungen verbessern können.

Infrastruktursicherheit in Amazon SNS

Als verwalteter Service ist Amazon SNS durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die in der Dokumentation Best Practices for Security, Identity and Compliance beschrieben sind.

Verwenden Sie AWS API-Aktionen, um über das Netzwerk auf Amazon SNS zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Clients müssen außerdem Cipher Suites mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen.

Anforderungen müssen sowohl mit einer Zugriffsschlüssel-ID als auch mit einem geheimen Zugriffsschlüssel signiert sein, die mit einem IAM-Prinzipal verknüpft sind. Alternativ können Sie die AWS Security Token Service (AWS STS) verwenden, um temporäre Anmeldeinformationen für Signaturanforderungen zu generieren.

Sie können API-Aktionen von jedem Netzwerkstandort aus aufrufen. Amazon SNS unterstützt jedoch ressourcenbasierte Zugriffsrichtlinien, die Einschränkungen auf der Basis der Quell-IP-Adresse enthalten können. Sie können auch Amazon SNS SNS-Richtlinien verwenden, um den Zugriff von bestimmten Amazon VPC-Endpunkten oder bestimmten zu kontrollieren. VPCs Dadurch wird der Netzwerkzugriff auf ein bestimmtes Amazon SNS SNS-Thema effektiv nur von der spezifischen VPC innerhalb des Netzwerks isoliert. AWS Weitere Informationen finden Sie unter Die Veröffentlichung von einem bestimmten VPC-Endpunkt aus auf nur ein Amazon SNS -Thema beschränken.

Bewährte Methoden für die Sicherheit in Amazon SNS

AWS bietet viele Sicherheitsfunktionen für Amazon SNS. Überprüfen Sie diese Sicherheitsfunktionen im Kontext Ihrer eigenen Sicherheitsrichtlinie.

> **Note**
>
> Der Leitlinien für diese Sicherheitsfunktionen gilt für allgemeine Anwendungsfälle und Implementierungen. Wir empfehlen Ihnen, diese bewährten Methoden im Kontext Ihres spezifischen Anwendungsfalls, der Architektur und des Bedrohungsmodells zu überprüfen.

Vorbeugende bewährte Methoden

Im Folgenden werden bewährte vorbeugende Sicherheitsmethoden für Amazon SNS beschrieben.

Themen

- Stellen Sie sicher, dass Themen nicht öffentlich zugänglich sind
- Implementieren der geringstmöglichen Zugriffsrechte
- Verwenden Sie IAM-Rollen für Anwendungen und AWS Services, für die Amazon SNS SNS-Zugriff erforderlich ist
- Implementieren serverseitiger Verschlüsselung
- Erzwingen der Verschlüsselung von Daten während der Übertragung
- Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SNS
- Stellen Sie sicher, dass Abonnements nicht für die Bereitstellung an unformatierte HTTP-Endpunkte konfiguriert sind

Stellen Sie sicher, dass Themen nicht öffentlich zugänglich sind

Sofern Sie nicht ausdrücklich verlangen, dass jemand im Internet Ihr Amazon SNS SNS-Thema lesen oder darauf schreiben kann, sollten Sie sicherstellen, dass Ihr Thema nicht öffentlich zugänglich ist (für jeden auf der Welt oder für jeden authentifizierten AWS Benutzer zugänglich).

- Vermeiden Sie das Erstellen von Richtlinien mit auf `""` festgelegtem `Principal`.
- Vermeiden Sie die Verwendung eines Platzhalters (`*`). Benennen Sie stattdessen einen oder mehrere bestimmte Benutzer.

Implementieren der geringstmöglichen Zugriffsrechte

Wenn Sie Berechtigungen erteilen, entscheiden Sie, wer sie erhält, für welche Themen die Berechtigungen gelten und welche bestimmten API-Aktionen für diese Themen zugelassen werden.

Die Umsetzung des Prinzips der geringsten Rechte ist für die Reduzierung von Sicherheitsrisiken ausschlaggebend. Es hilft auch, die negativen Auswirkungen von Fehlern oder böswilligen Absichten zu reduzieren.

Folgen Sie den standardmäßigen Sicherheitshinweisen zur Erteilung von geringsten Privilegien. Das heißt: erteilen Sie nur die Berechtigungen, die zum Ausführen einer bestimmten Aufgabe erforderlich sind. Sie können die geringsten Rechte implementieren, indem Sie eine Kombination von Sicherheitsrichtlinien für den Benutzerzugriff verwenden.

Amazon SNS verwendet das Herausgeber-Abonnenten-Modell, das drei Arten von Benutzerkontenzugriff erfordert:

- Administratoren – Zugriff auf das Erstellen, Ändern und Löschen von Themen. Administratoren steuern auch Themenrichtlinien.
- Herausgeber – Zugriff auf das Senden von Nachrichten an Themen.
- Abonnenten – Zugriff auf das Abonnieren von Themen.

Weitere Informationen finden Sie in den folgenden Abschnitten:

- Identity and Access Management in Amazon SNS
- Amazon SNS-API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen

Verwenden Sie IAM-Rollen für Anwendungen und AWS Services, für die Amazon SNS SNS-Zugriff erforderlich ist

Damit Anwendungen oder AWS Dienste wie Amazon auf Amazon EC2 SNS-Themen zugreifen können, müssen sie in ihren AWS API-Anfragen gültige AWS Anmeldeinformationen verwenden. Da diese Anmeldeinformationen nicht automatisch rotiert werden, sollten Sie die AWS Anmeldeinformationen nicht direkt in der Anwendung oder EC2 Instance speichern.

Sie sollten mithilfe einer IAM-Rolle temporäre Anmeldeinformationen für Anwendungen und Services verwalten, die Zugriff auf Amazon SNS benötigen. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen (wie einen Benutzernamen, ein Passwort und Zugriffsschlüssel) an eine EC2 Instanz oder einen AWS Dienst verteilen, z. AWS Lambda B. Stattdessen stellt die Rolle temporäre Berechtigungen bereit, die Anwendungen verwenden können, wenn sie andere AWS Ressourcen aufrufen.

Weitere Informationen finden Sie unter IAM-Rollen und Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services im IAM Benutzerhandbuch.

Implementieren serverseitiger Verschlüsselung

Probleme durch Datenlecks lassen sich verringern, indem Sie die Verschlüsselung im Ruhezustand verwenden. Dabei verschlüsseln Sie Ihre Nachrichten mithilfe eines Schlüssels, der an einem anderen Speicherort gespeichert ist als Ihre Nachrichten. Serverseitige Verschlüsselung (SSE) bietet Datenverschlüsselung im Ruhezustand. Amazon SNS verschlüsselt Ihre Daten auf Nachrichtenebene bei der Speicherung und entschlüsselt die Nachrichten für Sie bei Zugriff darauf. SSE verwendet Schlüssel, die in verwaltet werden. AWS Key Management Service Solange Sie Ihre Anfrage authentifizieren und über Zugriffsberechtigungen verfügen, besteht kein Unterschied zwischen dem Zugriff auf verschlüsselte und unverschlüsselte Themen.

Weitere Informationen erhalten Sie unter Sicherung von Amazon SNS SNS-Daten mit serverseitiger Verschlüsselung und Verwaltung von Amazon SNS SNS-Verschlüsselungsschlüsseln und Kosten.

Erzwingen der Verschlüsselung von Daten während der Übertragung

Es ist möglich, aber nicht zu empfehlen, Nachrichten zu veröffentlichen, die während der Übertragung über HTTP nicht verschlüsselt sind. Wenn ein Thema jedoch im Ruhezustand verschlüsselt wird, ist es erforderlich AWS KMS, HTTPS für die Veröffentlichung von Nachrichten zu verwenden, um die Verschlüsselung sowohl im Ruhezustand als auch bei der Übertragung sicherzustellen. Das Thema lehnt HTTP-Nachrichten zwar nicht automatisch ab, die Verwendung von HTTPS ist jedoch erforderlich, um die Sicherheitsstandards aufrechtzuerhalten.

AWS empfiehlt, HTTPS anstelle von HTTP zu verwenden. Wenn Sie HTTPS verwenden, werden Nachrichten während des Transports automatisch verschlüsselt, selbst wenn das SNS-Thema selbst nicht verschlüsselt ist. Ohne HTTPS kann ein netzwerkbasierter Angreifer den Netzwerkverkehr abhören oder ihn mithilfe eines Angriffs wie manipulieren. man-in-the-middle

Um nur verschlüsselte Verbindungen über HTTPS zu erzwingen, fügen Sie die`aws:SecureTransport`-Bedingung in der IAM-Richtlinie hinzu, die unverschlüsselten SNS-Themen zugeordnet ist. Dies zwingt Nachrichtenherausgeber, HTTPS anstelle von HTTP zu verwenden. Sie können folgendes Beispiel nutzen:

JSON

```
{
```

```
    "Id": "ExamplePolicy",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPublishThroughSSLOnly",
            "Action": "SNS:Publish",
            "Effect": "Deny",
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:test-topic"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            },
            "Principal": "*"
        }
    ]
}
```

Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SNS

Verwenden Sie bei Themen, mit denen eine Interaktion erforderlich ist, die jedoch dem Internet absolut nicht zugänglich gemacht werden dürfen, VPC-Endpunkte, um den Zugriff auf Themen ausschließlich auf die Hosts innerhalb einer bestimmten VPC einzuschränken. Sie können Themenrichtlinien verwenden, um den Zugriff auf Themen von bestimmten Amazon VPC-Endpunkten oder von bestimmten aus zu steuern. VPCs

Amazon SNS-VPC-Endpunkte bieten zwei Möglichkeiten zur Kontrolle des Zugriffs auf Ihre Nachrichten:

- Sie können steuern, welche Anfragen, Benutzer oder Gruppen durch einen spezifischen VPC-Endpunkt erlaubt sind.
- Mithilfe einer Themenrichtlinie können Sie steuern, welche VPCs oder VPC-Endpunkte Zugriff auf Ihr Thema haben.

Weitere Informationen erhalten Sie unter Erstellen des Endpunkts und Erstellen einer Amazon-VPC-Endpunktrichtlinie für Amazon SNS.

Stellen Sie sicher, dass Abonnements nicht für die Bereitstellung an unformatierte HTTP-Endpunkte konfiguriert sind

Vermeiden Sie das Konfigurieren von Abonnements für die Bereitstellung an unformatierte HTTP-Endpunkte. Stellen Sie immer Abonnements bereit, die an einen Endpunkt-Domänennamen gesendet werden. Beispiel: Ein Abonnement, das für die Bereitstellung an einen Endpunkt konfiguriert ist, `http://1.2.3.4/my-path`, sollte geändert werden in `http://my.domain.name/my-path`.

Problembehebung Amazon SNS SNS-Themen mithilfe von AWS X-Ray

AWS X-Ray sammelt Daten über Anfragen, die Ihre Anwendung bearbeitet, und ermöglicht es Ihnen, Daten anzuzeigen und zu filtern, um potenzielle Probleme und Optimierungsmöglichkeiten zu identifizieren. Für jede verfolgte Anfrage an Ihre Anwendung können Sie detaillierte Informationen über die Anfrage, die Antwort und die Aufrufe einsehen, die Ihre Anwendung an nachgelagerte AWS Ressourcen, Microservices, Datenbanken und das HTTP-Web sendet. APIs

Sie können X-Ray mit Amazon SNS verwenden, um die in Ihrer Anwendung weitergeleiteten Nachrichten nachzuverfolgen und zu analysieren. Sie können den verwenden AWS Management Console , um die Karte der Verbindungen zwischen Amazon SNS und anderen Diensten anzuzeigen, die Ihre Anwendung verwendet. Sie können mithilfe der Konsole auch Metriken, wie durchschnittliche Latenz- und Ausfallraten, anzuzeigen. Weitere Informationen finden Sie unter Amazon SNS und AWS X-Ray im AWS X-Ray Entwicklerhandbuch.

Aktives Tracing in Amazon SNS

Wird verwendet AWS X-Ray , um Benutzeranfragen zu verfolgen und zu analysieren, während sie Ihre Amazon SNS SNS-Themen an Amazon Data Firehose- AWS Lambda, Amazon SQS - und HTTP/S-Endpunktabonnements weiterleiten.

Mit X-Ray erhalten Sie einen end-to-end Überblick über jede Anfrage, sodass Sie:

- Identifizieren Sie, was Ihr Amazon SNS SNS-Thema ist und welche Dienste den Abonnements nachgelagert sind.
- Analysieren Sie Latenzen, wie z. B.:
  - Zeit, die vor der Verarbeitung mit dem Amazon SNS SNS-Thema verbracht wurde.
  - Lieferzeiten für jeden abonnierten Endpunkt.

> **Important**
>
> Amazon-SNS-Themen mit zahlreichen Abonnements erreichen möglicherweise eine Größenbeschränkung und können nicht vollständig verfolgt werden. Informationen zu

Größenbeschränkungen für Trace-Dokumente finden Sie unter X-Ray Service-Kontingente in der AWS allgemeinen Referenz.

Wenn Sie eine Amazon-SNS-API von einem Service aufrufen, der bereits nachverfolgt wird, führt Amazon SNS die Nachverfolgung auch dann durch, wenn X-Ray-Tracing auf der API nicht aktiviert ist.

Amazon SNS unterstützt X-Ray-Tracing für Standard- und FIFO-Themen. Sie können X-Ray für ein Amazon-SNS-Thema aktivieren, indem Sie die Amazon-SNS-Konsole, die Amazon-SNS–API `SetTopicAttributes`, die CLI-Referenz zu Amazon Simple Notification Service oder AWS CloudFormation verwenden.

Weitere Informationen zur Verwendung von Amazon SNS mit X-Ray finden Sie unter Amazon SNS und AWS X-Ray im AWS X-Ray -Entwicklerhandbuch.

Berechtigungen für aktives Tracing

Wenn Sie die Amazon-SNS-Konsole verwenden, versucht Amazon SNS, die erforderlichen Berechtigungen für das Amazon-SNS-Thema zum Aufrufen von X-Ray zu erstellen. Der Versuch kann abgelehnt werden, wenn Sie nicht über ausreichende Berechtigungen verfügen, um die Amazon-SNS-Konsole zu verwenden. Weitere Informationen erhalten Sie unter Identity and Access Management in Amazon SNS und Beispiele für die Zugriffskontrolle in Amazon SNS.

Wenn Sie die CLI verwenden, müssen Sie die Berechtigungen manuell konfigurieren. Diese Berechtigungen werden mithilfe von Ressourcenrichtlinien konfiguriert. Weitere Informationen zur Verwendung der erforderlichen Berechtigungen in X-Ray finden Sie unter Amazon SNS und AWS X-Ray.

Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe der Konsole AWS

Wenn das aktive Tracing für ein Amazon-SNS-Thema aktiviert ist, liest es die Trace-ID, sendet die Daten anhand der Trace-ID an den Kunden und gibt die Trace-ID an nachgelagerte Services weiter.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Thema aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Themen finden Sie unter Erstellen eines Amazon-SNS-Themas.

3. Wählen Sie auf der Seite Thema erstellen im Abschnitt Details einen Thementyp aus: FIFO oder Standard.
   a. Geben Sie den Namen des neuen Themas ein.
   b. (Optional) Geben Sie einen Anzeigenamen für Ihr Thema ein.
4. Erweitern Sie Aktives Tracing und wählen Sie Aktives Tracing verwenden.

Sobald Sie X-Ray für Ihr Amazon SNS SNS-Thema aktiviert haben, können Sie die X-Ray-Servicekarte verwenden, um die end-to-end Traces und Service Maps für das Thema anzuzeigen.

Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe des SDK AWS

Das folgende Codebeispiel zeigt, wie die aktive Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe des AWS SDK for Java aktiviert wird.

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

Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe der CLI AWS

Das folgende Codebeispiel zeigt, wie die aktive Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe der AWS CLI aktiviert wird.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Aktivieren der aktiven Ablaufverfolgung für ein Amazon SNS SNS-Thema mit AWS CloudFormation

Der folgende AWS CloudFormation Stack zeigt, wie Sie die aktive Ablaufverfolgung für ein Amazon SNS SNS-Thema aktivieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

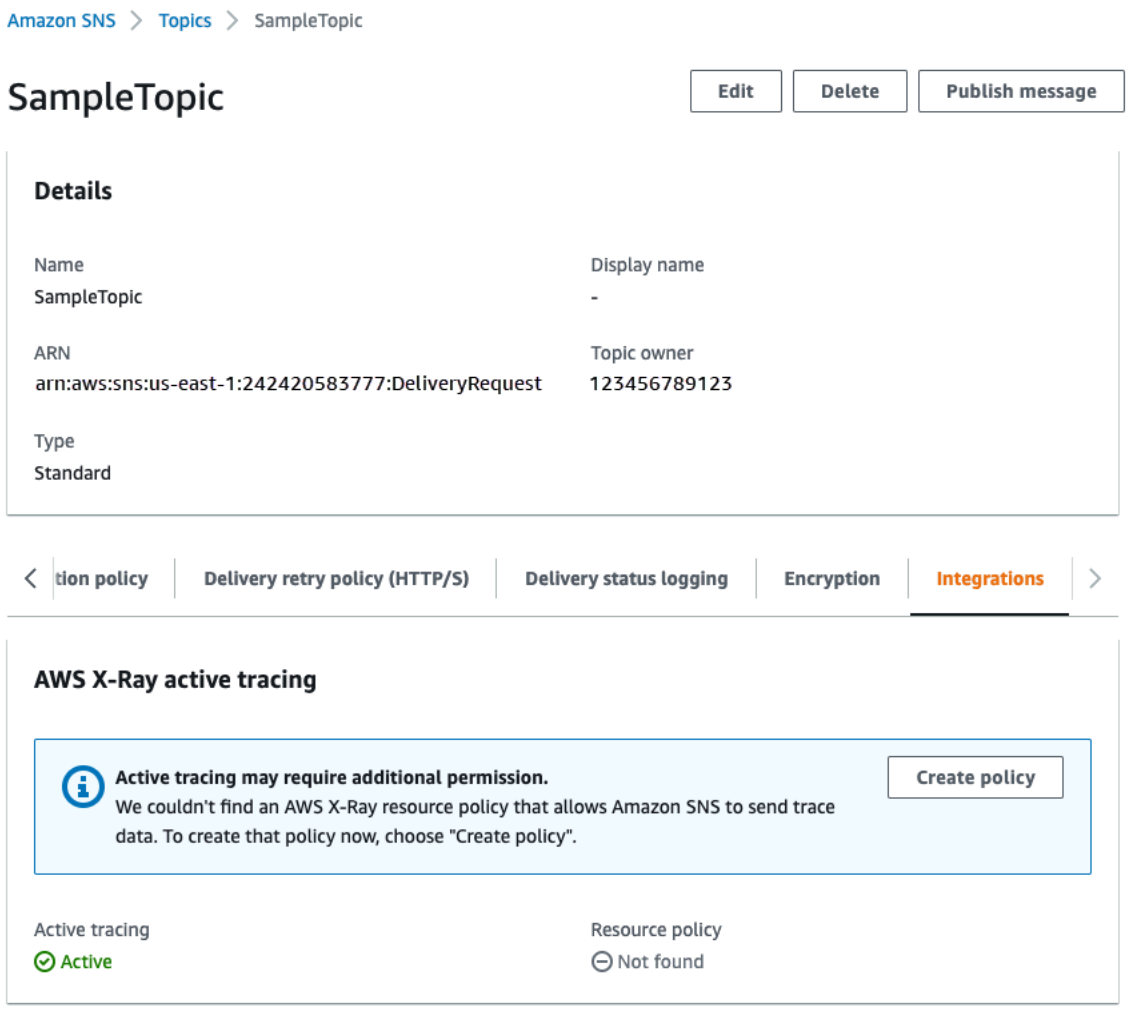
Überprüfen, ob das aktive Tracing für Ihr Thema aktiviert ist

Sie können die Amazon-SNS-Konsole verwenden, um zu überprüfen, ob das aktive Tracing für Ihr Thema aktiviert ist oder ob die Ressourcenrichtlinie nicht hinzugefügt werden konnte.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Wählen Sie auf der Seite Themen ein Thema aus.
4. Wählen Sie die Registerkarte Integrationen aus.

   Wenn das aktive Tracing aktiviert ist, wird ein grünes Aktiv-Symbol angezeigt.

5. Wenn Sie das aktive Tracing aktiviert haben und nicht sehen, dass die Ressourcenrichtlinie hinzugefügt wurde, wählen Sie Richtlinie erstellen aus, um die zusätzlichen erforderlichen Berechtigungen hinzuzufügen.



Aktives Tracing testen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Erstellen Sie ein Amazon-SNS-Thema. Weitere detaillierte Informationen hierzu finden Sie unter Um ein Thema mit dem zu erstellen AWS Management Console.
3. Erweitern Sie Aktives Tracing und wählen Sie Aktives Tracing verwenden.

4. Veröffentlichen Sie eine Nachricht im Amazon-SNS-Thema. Weitere detaillierte Informationen hierzu finden Sie unter Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen.
5. Verwenden Sie die X-Ray-Servicekarte, um die end-to-end Traces und Service Maps für das Thema anzuzeigen.

Verlauf der Amazon SNS SNS-Dokumentation

In der folgenden Tabelle werden die letzten Änderungen am Amazon Simple Notification Service-Entwicklerhandbuch beschrieben.

Servicefunktionen werden manchmal schrittweise in den AWS Regionen eingeführt, in denen ein Dienst verfügbar ist. Wir aktualisieren diese Dokumentation nur für die erste Version. Wir stellen keine Informationen über die Verfügbarkeit von Regionen zur Verfügung und kündigen auch keine späteren Rollouts von Regionen an. Informationen zur regionalen Verfügbarkeit von Servicefunktionen und zum Abonnieren von Benachrichtigungen über Updates finden Sie unter Was gibt's Neues bei AWS? .

Änderung	Beschreibung	Datum
Unterstützung für Dual-Stack IPv4 - (und IPv6) Endpunkte hinzugefügt	Amazon SNS unterstützt IPv6 jetzt API-Anfragen, sodass Kunden mithilfe IPv4 von,, oder Dual Stack eine Verbindung zu öffentlichen Endpunkten herstellen können. IPv6	3. April 2025
Amazon SNS hat das `FifoThroughputScope` Attribut für Amazon SNS FIFO-Themen hinzugefügt	Amazon SNS unterstützt das FifoThroughputScope Attribut, das die Durchsatzquote und das Deduplizierungsverhalten angibt, die für das FIFO-Thema gelten sollen. Gültige Werte sind `Topic` oder `MessageGroup` .	21. Januar 2025
`AmazonSNSFullAccess` und verwaltete Richtlinienaktualisierungen `AmazonSNSReadOnlyAccess`	Amazon SNS hat neue Berechtigungen hinzugefügt AmazonSNSFullAccess und Richtlinien AmazonSNS ReadOnlyAccess verwaltet , was zusätzlichen Zugriff	24. September 2024

	auf Amazon SNS über die ermöglicht. AWS Management Console	
Amazon SNS SNS-Integration mit AWS Endbenutzer-Messaging SMS für die Zustellung von SMS-Nachrichten	Amazon SNS unterstützt neue Funktionen wie SMS-Ressourcenverwaltung, bidirektionales Messaging, detaillierte Ressourcenberechtigungen, länderspezifische Sperrregeln und zentrale Abrechnung für alle AWS SMS-Nachrichten, ohne dass Änderungen an den Konfigurationen oder dem von Amazon SNS verwendeten globalen AWS SMS-Netzwerk vorgenommen werden müssen.	24. September 2024
Unterstützung für FIFO-Themen in Kanada West (Calgary)	Amazon SNS unterstützt das FIFO-Thema in Kanada West (Calgary).	28. März 2024
Amazon SNS SMS-Unterstützung in fünf neuen Regionen	Amazon SNS hat SMS-Unterstützung für die folgenden Regionen hinzugefügt: Asien-Pazifik (Hyderabad), Asien-Pazifik (Melbourne), Naher Osten (VAE), Europa (Zürich). und Europa (Spanien).	8. Februar 2024
Unterstützung für Firebase Cloud Messaging (FCM) HTTP v1	Amazon SNS unterstützt FCM v1-Anmeldeinformationen.	18. Januar 2024
Amazon SNS-SMS werden in Asien-Pazifik (Jakarta) unterstützt	Amazon-SNS unterstützt SMS-Nachrichten in Asien-Pazifik (Jakarta)	14. Dezember 2023

AWS CloudFormation Unterstützung für die Konfiguration von `DeliveryStatusLogging` Amazon SNS SNS-Themen	AWS CloudFormation Support steht für die Konfiguration `DeliveryStatusLogging` bei der Erstellung oder Aktualisierung von Amazon SNS SNS-Themen zur Verfügung.	07. Dezember 2023
Neue Nachrichtenfilter-Operatoren hinzugefügt	Sie können jetzt beim Filtern von Amazon-SNS-Nachrichten Suffix-Abgleich, Gleich-Groß- und Kleinschreibung ignorieren und OR-Operatoren verwenden.	16. November 2023
Support für Nachrichtenarchivierung und -wiederholung hinzugefügt	Themeninhaber können Nachrichten bis zu 365 Tage lang zu einem Thema archivieren. Themen-Subscriber können die archivierten Nachrichten an einem abonnierten Endpunkt wiederholen, um Nachrichten wiederherzustellen, die auf einen Fehler in einer nachgelagerten Anwendung zurückzuführen sind, oder um den Status einer vorhandenen Anwendung zu replizieren.	26. Oktober 2023

Support für das Abonnieren einer Standard-Warteschlange für ein FIFO-Thema hinzugefügt	Sie können eine Amazon-SQS-FIFO-Warteschlange oder eine Standard-Warteschlange für ein Amazon-SNS-FIFO-Thema abonnieren. Nur Amazon-SQS-FIFO-Warteschlangen garantieren, dass Nachrichten in der richtigen Reihenfolge und ohne Duplikate empfangen werden.	14. September 2023
SMS-Unterstützung für Israel (Tel Aviv) hinzugefügt	Amazon-SNS-SMS wird jetzt in der Region Israel (Tel Aviv) unterstützt.	28. August 2023
Support für aktives X-Ray-Tracing zu FIFO-Themen hinzugefügt	Bisher nur mit Amazon SNS SNS-Standardthemen unterstützt, verfolgt und analysiert AWS X-Ray jetzt Benutzeranfragen auf ihrem Weg durch Ihre FIFO-Themen zu Ihren Amazon Data Firehose- AWS Lambda, Amazon SQS- und Endpoint-Abonnements. HTTP/S	31. Mai 2023
Verbesserte Unterstützung für Content-Type-Header	Sie können in der Anforderungsrichtlinie den Content-Type-Header festlegen, um den Medientyp der Benachrichtigung anzugeben.	23. März 2023

Aktive Tracing-Unterstützung hinzugefügt	AWS X-Ray verfolgt und analysiert Benutzeranfragen auf ihrem Weg durch Ihre Amazon SNS SNS-Standardthemen zu Ihren Amazon Data Firehose- AWS Lambda, Amazon SQS- und HTTP/S Endpoint-Abonnements.	8. Februar 2023
Registrierung einer Sender-ID für Singapur	Anweisungen zur Registrierung des Absenders IDs in Singapur hinzugefügt.	10. Januar 2023
Nutzlastbasierte Nachrichtenfilterung	Mit der nutzlastbasierten Filterung können Sie Nachrichten anhand der Nachrichtennutzlast filtern und so die Kosten vermeiden, die mit der Verarbeitung unerwünschter Daten verbunden sind.	22. November 2022
SHA256 Hash-Algorithmus für Amazon SNS SNS-Nachrichtensignierung hinzugefügt	Support für den SHA256 Hash-Algorithmus bei Verwendung der Amazon SNS SNS-Nachrichtensignatur hinzugefügt.	15. September 2022
Zusätzliche Regionen zum SMS-Messaging hinzugefügt	Amazon SNS unterstützt SMS-Nachrichten in den folgenden Regionen: Afrika (Kapstadt), Asien-Pazifik (Osaka), Europa (Mailand) und AWS GovCloud (USA-Ost).	09. September 2022

Unterstützung für den Nachrichtendatenschutz hinzugefügt	Der Nachrichtendatenschutz schützt die Daten, die in Ihren Amazon SNS SNS-Themen veröffentlicht werden, indem die vertraulichen Informationen, die zwischen Anwendungen oder AWS Diensten übertragen werden, mithilfe von Datenschutzrichtlinien geprüft und blockiert werden.	8. September 2022
Neues Registrierungsverfahren für gebührenfreie Nummern	Support für das Senden von Amazon-SNS-Nachrichten über gebührenfreie Telefonnummern (TFN) an Empfänger in den Vereinigten Staaten hinzugefügt.	1. August 2022
Unterstützung für attributbasierte Zugriffssteuerungen (ABAC)	Zusätzliche Unterstützung für attributbasierte Zugriffssteuerung (ABAC) für API-Aktionen, einschließlich `Publish` und `PublishBatch` . ABAC ist eine Autorisierungsstrategie, die Zugriffsberechtigungen auf der Grundlage von Tags definiert, die an IAM-Ressourcen wie IAM-Benutzer und -Rollen und an Ressourcen wie Amazon SNS AWS SNS-Themen angehängt werden können, um die Berechtigungsverwaltung zu vereinfachen.	10. Januar 2022

Support für Apple-Token-basierte Authentifizierung für Push-Benachrichtigungen	Sie können Amazon SNS ermächtigen, Push-Benachrichtigungen an Ihre iOS- oder macOS-App zu senden, indem Sie Informationen bereitstellen, die Sie als Entwickler der App identifizieren.	28. Oktober 2021
Neue Absender von SMS-Nachrichten werden in der SMS-Sandbox platziert	Um Betrug und Missbrauch zu vermeiden und Ihren Ruf als Absender zu schützen, ist die SMS-Sandbox vorhanden. Solange sich Ihr AWS Konto in der SMS-Sandbox befindet, können Sie SMS-Nachrichten nur an verifizierte Zieltelefonnummern senden.	1. Juni 2021
Neue Absender von SMS-Nachrichten werden in der SMS-Sandbox platziert	Um Betrug und Missbrauch zu vermeiden und Ihren Ruf als Absender zu schützen, ist die SMS-Sandbox vorhanden. Solange sich Ihr AWS Konto in der SMS-Sandbox befindet, können Sie SMS-Nachrichten nur an verifizierte Zieltelefonnummern senden.	1. Juni 2021
Neue Attribute für das Senden von SMS-Nachrichten an Empfänger in Indien	Zwei neue Attribute Entity-ID und ID der Vorlage um SMS-Nachrichten an Empfänger in Indien zu senden, sind jetzt erforderlich.	22. April 2021

Aktualisierungen von Nachrichtenfilter-Operatoren	Ein neuer Operator, `cidr`, ist für die übereinstimmenden IP-Adressen und Subnetze der Nachrichtenquelle verfügbar . Sie können nun auch auf das Fehlen eines Attributs chlüssels prüfen und ein Präfix mit dem `anything-but` -Operator für Attribut-Zeichenfolgenabgleich.	7. April 2021
Beenden der Unterstützung für P2P-LongCodes für US-Destinationen	Ab dem 1. Juni 2021 unterstützen die US-amerikanischen Telekommunikationsanbieter die Verwendung von person-to-person (P2P) -Langcodes für application-to-person (A2P) -Verbindungen zu Zielen in den USA nicht mehr. Stattdessen können Sie Kurzcodes, gebührenfreie Nummern oder eine neue Art von Ursprungsnummer, 10DLC genannt, benutzen.	16. Februar 2021
Support für einminütige Amazon-Metriken CloudWatch	Die CloudWatch 1-Minuten -Metrik für Amazon SNS ist jetzt in allen AWS Regionen verfügbar.	28. Januar 2021

Support für Endgeräte	Sie können Firehose-Lieferstreams für SNS-Themen abonnieren. Auf diese Weise können Sie Benachrichtigungen an Archivierungs- und Analyseen dpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden.	12. Januar 2021
Ursprungsnummern sind verfügbar	Beim Senden von Textnachr ichten (SMS) können Sie Originalnummern verwenden.	23. Oktober 2020
Support für Amazon-SNS-FIFO-Themen	Um verteilte Anwendungen, die Datenkonsistenz in nahezu Echtzeit erfordern, zu integrier en, können Sie Amazon SNS First-In-First-Out-Themen (FIFO) in Amazon SQS FIFO-Queues verwenden.	22. Oktober 2020
Die Amazon SNS Extended Client Library für Java ist verfügbar	Sie können diese Bibliothek verwenden, um große Amazon SNS-Nachrichten zu veröffent lichen.	25. August 2020
SSE ist in den China-Reg ionen verfügbar	Serverseitige Verschlüsselung (SSE) für Amazon SNS ist in den chinesischen Regionen verfügbar.	20. Januar 2020

Support für die Erfassung DLQs unzustellbarer Nachrichten	Sie können eine Amazon SQS-Queue für unzustellbare Nachrichten mit einem Amazon SNS-Abonnement verwenden, um unzustellbare Nachrichten zu erfassen.	14. November 2019
Support für die Angabe benutzerdefinierter APNs Header-Werte	Sie können einen benutzerdefinierten APNs Header-Wert angeben.	18. Oktober 2019
Support für das Header-Feld `apns-push-type` " für APNs	Sie können das `apns-push-type` Header-Feld für mobile Benachrichtigungen verwenden, die über gesendet werden APNs.	10. September 2019
Support bei der Problembehandlung mit AWS X-Ray	Sie können mithilfe von X-Ray Fehler bei Nachrichten beheben, die über SNS-Themen weitergeleitet werden.	24. Juli 2019
Support für die Attribut-Schlüsselzuordnung mit dem „`exists`“-Operator	Sie können mithilfe des `exists`-Operators überprüfen, ob eine eingehende Nachricht über ein Attribut verfügt, dessen Schlüssel in der Filterrichtlinie aufgeführt ist.	5. Juli 2019
Unterstützung für den „Alles außer“-Abgleich mehrerer numerischer Werte	Zusätzlich zu mehreren Zeichenfolgen erlaubt Amazon SNS den „Alles außer“-Abgleich mehrerer numerischer Werte.	5. Juli 2019

Amazon-SNS-Version shinweise sind als RSS-Feed verfügbar	Nach dem Titel auf dieser Seite (Dokumentationsverlauf), wählen Sie RSS-FEED aus.	22. Juni 2019

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.