亚马逊云科技

开发人员指南

Amazon Simple Notification Service

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

Amazon Simple Notification Service: 开发人员指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆或者贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅 中国的 Amazon Web Services 服务入门 (PDF)。

Table of Contents

什么是 Amazon SNS？ ... 1
- 工作方式 ... 1
- 访问 Amazon SNS ... 2
- Amazon SNS 常见场景 ... 3
  - 应用程序集成 ... 3
  - 应用程序提示 ... 4
  - 用户通知 ... 4
  - 移动推送通知 ... 4
- Amazon SNS 的定价 ... 4
- Amazon SNS 特征和功能 ... 4
- 常用共享服务 ... 6
- 与 Amazon SDKs ... 8

创建主题并发布消息 ... 10
- 设置 ... 10
  - 创建账户和 IAM 用户 ... 10
  - 后续步骤 ... 11
- 步骤 1：创建主题 ... 11
  - Amazon Web Services Management Console ... 12
  - Amazon SDKs ... 14
- 步骤 2：创建主题订阅 ... 29
  - 要为终端节点订阅 Amazon SNS 主题 ... 30
- 步骤 3：发布消息 ... 31
  - Amazon Web Services Management Console ... 31
  - Amazon SDKs ... 32
  - 大型消息负载 ... 56
  - 消息属性 ... 67
  - 消息批处理 ... 71
- 第 4 步：删除订阅和主题 ... 74
  - Amazon Web Services Management Console ... 74
  - Amazon SDKs ... 75
- 后续步骤 ... 85

使用 FIFO 主题的消息排序和重复数据删除 ... 86
- 高吞吐量 FIFO 主题 ... 86
  - 使用案例 ... 86

分区和数据分布 ... 87
分区和数据分布 ... 87
启用高吞吐量 ... 87
为 Amazon SQS 队列启用高吞吐量模式 ... 88
FIFO 主题使用案例 ... 88
消息顺序详细信息 ... 89
消息分组 ... 93
按消息组分发数据 IDs 以提高性能 ... 94
消息传输 ... 95
消息筛选 ... 95
消息重复数据删除 ... 97
消息安全性 ... 98
消息持久性 ... 99
消息归档与重播功能 ... 101
什么是消息归档与重播功能 ... 101
对于主题拥有者 ... 102
对于主题订阅用户 ... 106
代码示例 ... 109
FIFO 示例 (Amazon SDKs) ... 109
FIFO 示例 (Amazon CloudFormation) ... 122
消息筛选 ... 127
订阅筛选策略范围 ... 127
订阅筛选策略 ... 127
Amazon SNS 筛选策略示例 ... 128
筛选策略限制 ... 131
AND/OR 逻辑 ... 134
键匹配 ... 138
数值匹配 ... 140
字符串值匹配 ... 143
应用订阅筛选策略 ... 149
Amazon Web Services Management Console ... 150
Amazon CLI ... 151
Amazon SDKs ... 152
Amazon SNS API ... 156
Amazon CloudFormation ... 156
删除订阅筛选策略 ... 157

使用 Amazon Web Services Management Console 157
使用 Amazon CLI 157
使用 Amazon SNS API 158
消息数据保护 159
什么是消息数据保护 159
为什么使用消息数据保护？ 159
数据保护策略 160
什么是数据保护策略？ 160
数据保护策略结构概览 160
如何确定 IAM 主体 163
数据保护策略操作 163
数据保护策略示例 171
创建数据保护策略 178
删除数据保护策略 186
数据标识符 187
托管数据标识符 187
消息传输 226
原始消息传输 226
利用 Amazon Web Services Management Console实现原始消息传输 226
消息格式示例 227
Amazon SQS 订阅的消息属性和原始消息传送 228
跨账户传输 228
队列拥有者创建订阅 228
非队列所有者用户创建订阅 230
如何强制订阅要求对取消订阅请求进行身份验证？ 232
跨区域传输 233
选择加入的区域 233
消息传输状态 235
记录交付状态的先决条件 236
使用 Amazon Web Services Management Console配置传输状态日志记录 237
使用配置传送状态日志 Amazon SDKs 238
Amazon 用于配置主题属性的 SDK 示例 240
使用 Amazon CloudFormation配置传输状态日志记录 248
消息传输重试 250
传输协议和策略 250
传输策略阶段 251

创建 HTTP/S 传输策略 252
死信队列 256
为什么消息传输会失败？ 257
死信队列的工作方式 258
如何将消息移至死信队列中？ 258
如何将消息移出死信队列？ 258
如何监控和记录死信队列？ 258
配置死信队列 259
消息归档和分析 264
资源管理和优化 265
标记 265
成本分配的标记 265
访问控制的标记 266
进行标记以便进行资源搜索和筛选 267
配置标签 268
Amazon SNS 事件来源和目标 274
事件源 274
Analytics 274
应用程序集成 275
账单和成本管理 276
业务应用程序 276
计算 277
容器 278
客户互动 278
数据库 279
开发人员工具 280
前端 Web 和移动 281
游戏开发 281
物联网 282
机器学习 283
管理与治理 283
媒体 285
迁移与传输 285
网络和内容分发 286
安全性、身份与合规性 287
无服务器 288

存储 ... 289
其他事件来源 ... 290
事件目标 ... 291
A2A 目标 ... 291
A2P 目标 ... 292
Application-to-application 消息传递 ... 294
扇出到 Firehose 传输流 ... 294
先决条件 ... 295
将传输流订阅到主题 ... 296
管理跨多个传送流目标的消息 ... 297
消息归档和分析的示例应用场景 ... 310
扇出到 Lambda 函数 ... 321
先决条件 ... 322
将函数订阅到主题 ... 322
扇出到 Amazon SQS 队列 ... 323
为队列订阅主题 ... 324
通过以下方式自动发送亚马逊 SNS 到亚马逊 SQS 的消息 Amazon CloudFormation ... 330
将通知扇出到 HTTPS 端点 ... 338
为终端节点订阅主题 ... 339
验证消息签名 ... 347
解析消息格式 ... 352
Fanout 事件到 Amazon 事件分叉管道 ... 361
Amazon 事件分叉管道的工作原理 ... 362
部署 Amazon 事件分叉管道 ... 365
部署和测试 Event Fork Pipelines 示例应用程序 ... 366
为事件管道订阅主题 ... 374
使用 EventBridge 调度器 ... 381
设置执行角色 ... 382
创建计划 ... 382
相关资源 ... 385
Application-to-person 消息传递 ... 387
移动文本消息 ... 387
Amazon SNS 如何传送我的短信？ ... 388
入门 ... 389
发起身份 ... 394
配置 ... 395

短信的最佳实践 ... 469
发送移动推送通知 ... 483
Amazon SNS 用户通知的工作原理 ... 484
使用 Amazon SNS 设置推送通知 ... 484
设置移动应用程序 ... 485
使用 Amazon SNS 发送移动推送通知 ... 501
移动应用程序属性 ... 513
移动应用程序事件 ... 516
移动推送 API 操作 ... 519
移动推送 API 常见错误 ... 520
移动推送 TTL ... 530
支持的区域 ... 532
移动推送通知最佳实践 ... 533
电子邮件订阅设置和管理 ... 533
Amazon Web Services Management Console ... 534
Amazon SDKs ... 535
代码示例 ... 567
基本功能 ... 580
开始使用 Amazon SNS ... 581
操作 ... 593
场景 ... 764
构建应用程序以将数据提交到 DynamoDB 表 ... 765
构建 Amazon SNS 应用程序 ... 766
为推送通知创建平台终端节点 ... 767
创建无服务器应用程序来管理照片 ... 770
创建 Amazon Textract 浏览器应用程序 ... 774
创建并发布到 FIFO 主题 ... 775
检测视频中的人物和对象 ... 787
将短信发布到主题 ... 788
发布大型消息 ... 795
发布 SMS 文本消息 ... 798
将消息发布到队列 ... 806
使用 API Gateway 调用 Lambda 函数 ... 921
使用计划的事件调用 Lambda 函数 ... 923
无服务器示例 ... 925
通过 Amazon SNS 触发器调用 Lambda 函数 ... 925

安全性 935
数据加密 936
使用服务器端加密保护数据安全 936
密钥管理 938
使用服务器端加密设置主题加密 945
使用加密的 Amazon SQS 队列订阅设置主题加密 947
使用 VPC 端点保护流量 951
创建 VPC 终端节点 952
创建 VPC 策略 954
从 VPC 发布消息 955
消息数据保护安全性 966
身份和访问管理 967
受众 967
使用身份进行身份验证 967
使用策略管理访问 970
访问控制 972
访问控制应用场景 972
访问策略关键概念 972
架构概述 975
使用访问策略语言 977
评估逻辑 978
用于 Amazon SNS 访问控制的示例案例 983
Amazon SNS 如何与 IAM 配合使用 993
Amazon 托管策略 994
策略操作 999
策略资源 1000
策略条件键 1000
ACLs 1001
ABAC 1001
临时凭证 1002
主体权限 1002
服务角色 1002
服务相关角色 1003
基于身份的策略示例 1003
基于身份的策略 1006
基于资源的策略 1007

使用基于身份的策略 1007
管理自定义 IAM 策略 1014
使用临时凭证 1014
API 权限参考 1016
日志记录和监控 1019
CloudTrail 日志 1019
使用监控主题 CloudWatch 1029
合规性验证 1042
恢复能力 1043
基础结构安全性 1043
安全最佳实践 1043
预防性最佳实践 1044
使用疑难解答主题 Amazon X-Ray 1048
主动跟踪 1048
权限 1049
启用主动跟踪 1049
使用软件开发工具包对 Amazon SNS 主题启用主动跟踪 Amazon 1049
使用 CLI 在 Amazon SNS 主题上启用主动跟踪 Amazon 1050
使用在 Amazon SNS 主题上启用主动跟踪 Amazon CloudFormation 1050
验证已启用主动跟踪 1050
测试 1051
Amazon SNS 文档历史记录 1053
..... mlx

什么是 Amazon SNS？

亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 是一项完全托管的服务，可提供从发布者（制作者）到订阅者（消费者）的消息传送。发布者通过将消息发送至主题与订阅者进行异步交流，主题是一个逻辑访问点和通信渠道。

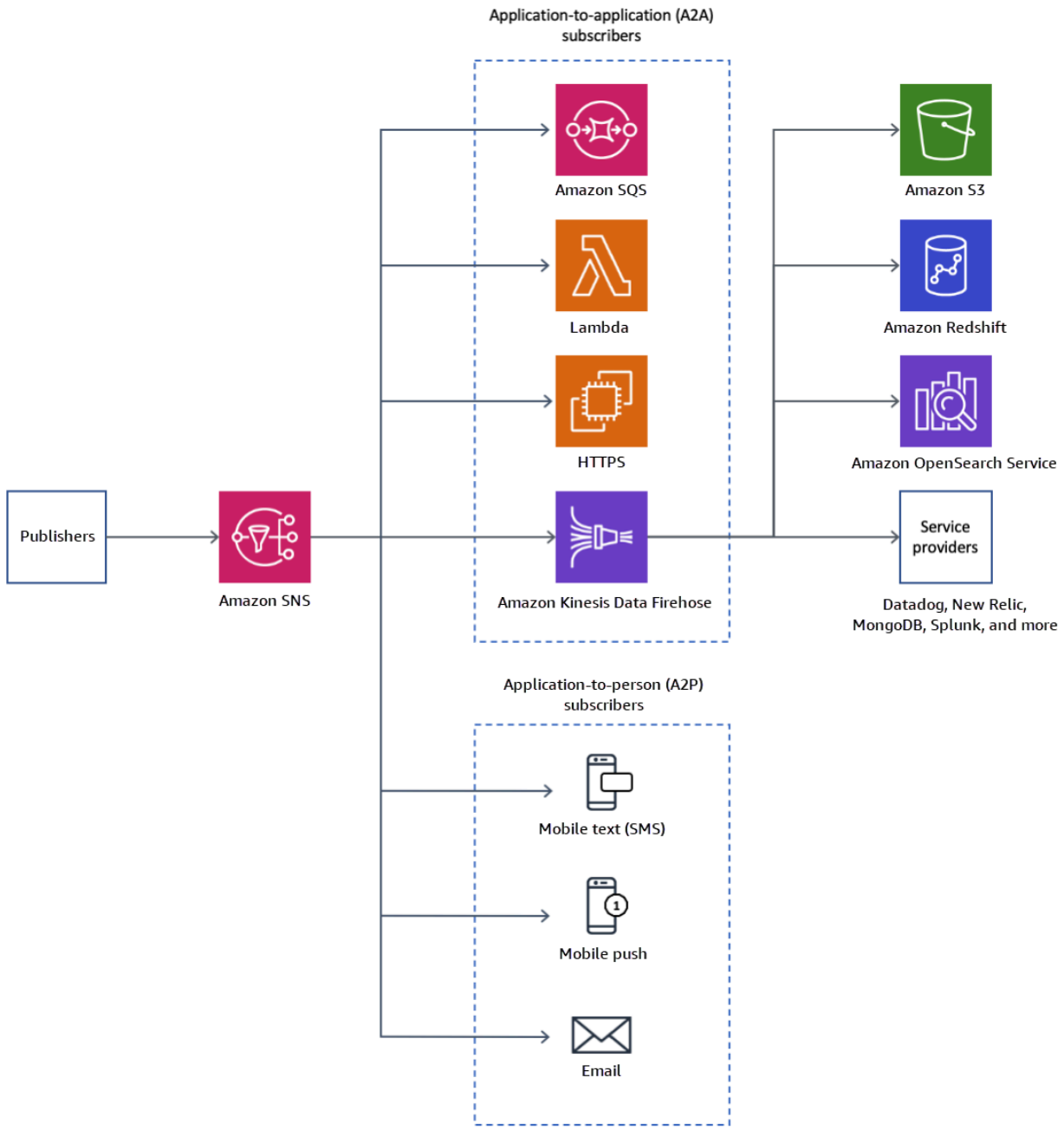
工作方式

在 SNS 中，发布者向主题发送消息，该主题充当沟通渠道。该主题充当逻辑接入点，确保跨不同平台将消息传递给多个订阅者。

SNS 主题的订阅者可以通过不同的终端节点接收消息，具体取决于他们的用例，例如：

- Amazon SQS
- Lambda
- HTTP (S) 端点
- 电子邮件
- 移动推送通知
- 移动短信 (SMS)
- Amazon Data Firehose
- 服务提供商（例如 Datadog、MongoDB、Splunk）

SNS 支持 Application-to-Application (A2A) 和 Application-to-Person (A2P) 消息传递，因此可以灵活地在不同的应用程序之间发送消息，也可以直接向手机、电子邮件地址等发送消息。



访问 Amazon SNS

您可以通过控制台、或访问和管理 Amazon SNS Amazon CLI Amazon SDKs，具体取决于您的首选交互方式。该控制台为基本任务提供了图形界面，而 Amazon CLI 和则为更复杂的用例 SDKs 提供了高级配置和自动化功能。

- Amazon SNS 控制台提供了一个方便的用户界面，用于创建主题和订阅、发送和接收消息以及监控事件和日志。
- Amazon Command Line Interface (Amazon CLI) 允许您直接访问 Amazon SNS API 以获取高级配置和自动化用例。有关更多信息，请参阅将 Amazon SNS 与 Amazon CLI结合使用。
- Amazon SDKs 以多种语言提供。有关更多信息，请参阅SDKs 和工具包。

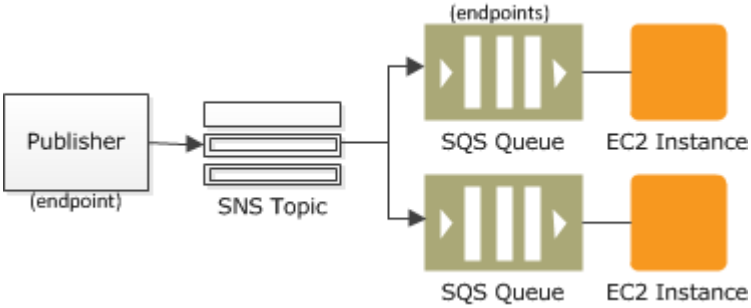
Amazon SNS 常见场景

使用这些常见的 Amazon SNS 场景可实现可扩展的事件驱动型架构，并确保应用程序和用户之间实现可靠的实时通信。

应用程序集成

扇出场景是复制发布到 SNS 主题的消息并将其推送到多个端点（例如 Firehose 传输流、Amazon SQS 队列、HTTP(S) 端点和 Lambda 函数）的场景。这允许进行并行异步处理。

例如，您可以开发一个应用程序，以在产品被下单的任何时候发布消息至 SNS 主题。然后，订阅 SNS 主题的 SQS 队列会接收到新订单的相同通知。连接到其中一个 SQS 队列的亚马逊弹性计算云 (Amazon EC2) 服务器实例可以处理或配送订单。您还可以将另一个 Amazon EC2 服务器实例附加到数据仓库中，以分析收到的所有订单。



您还可以通过扇出复制使用您的测试环境复制发送到生产环境的数据。对前一个示例进行展开讨论，您还可以为同一个 SNS 主题订阅另一个 SQS 队列，以处理新来的订单。然后，可将这个新 SQS 队列附加到测试环境中，您可以继续使用从生产环境中接收到的数据改进和测试应用程序。

> **Important**
>
> 在将任何生产数据发送到测试环境之前，请确保您考虑数据隐私和安全性。

有关更多信息，请参阅以下资源：

- 扇出到 Firehose 传输流
- 将 Amazon SNS 通知扇出到 Lambda 函数以进行自动处理
- 将 Amazon SNS 通知扇出到 Amazon SQS 队列进行异步处理
- 将 Amazon SNS 通知扇出到 HTTPS 端点
- 使用 Amazon SNS 以及计算、存储、数据库 Amazon 和网络服务实现事件驱动型计算

应用程序提示

应用程序和系统提示是由预定义阈值触发的通知。Amazon SNS 可以通过 SMS 和或电子邮件将这些通知发送给指定用户。例如，当事件发生时，您可以立即收到通知，例如您的 Amazon A EC2 uto Scaling 组的特定更改、上传到 Amazon S3 存储桶的新文件或亚马逊的指标阈值被突破。 CloudWatch 有关更多信息，请参阅亚马逊 CloudWatch 用户指南中的设置 Amazon SNS 通知。

用户通知

Amazon SNS 可以向个人或组发送推送电子邮件和短信（SMS 消息）。例如，您可以将电子商务订单确认作为用户通知发送。有关使用 Amazon SNS 发送 SMS 消息的更多信息，请参阅 移动文本消息与 Amazon SNS。

移动推送通知

使用移动推送通知，可将消息直接推送到移动应用程序。例如，您可以使用 Amazon SNS 向应用程序发送更新通知。通知消息可以包含下载和安装更新的链接。有关使用 Amazon SNS 发送推送通知消息的更多信息，请参阅 使用 Amazon SNS 发送移动推送通知。

Amazon SNS 的定价

Amazon SNS 没有前期成本。您将根据您发布的消息数量、传输的通知数量以及用于管理主题和订阅的任何其他 API 调用付费。传输定价因终端节点类型而异。您可以免费开始使用 Amazon SNS 免费套餐。有关信息，请参阅全球短信定价。

Amazon SNS 特征和功能

Amazon SNS 提供了一套全面的功能，旨在增强应用程序和用户之间的消息传递。这些功能可实现无缝通信、安全消息传送和强大的消息管理，确保各种消息传递用例的高可用性、持久性和灵活性。

Application-to-application 消息传递

pplication-to-application消息支持订阅者，例如 Amazon Data Firehose 交付流、Lambda 函数、亚马逊 SQS 队列、HTTP/S 终端节点和事件分叉管道。 Amazon 这允许在事件驱动型架构中实现高效消息传输。

Application-to-person notifications

pplication-to-person通知向订阅者提供用户通知，例如移动应用程序、移动电话号码和电子邮件地址。

标准主题和 FIFO 主题

FIFO 主题可确保严格的消息排序、消息分组和重复数据删除，允许 FIFO 和标准队列订阅消息处理。当消息排序和可能的重复并不重要时，会使用标准主题，支持更广泛的用例的所有传输协议。

消息持久性

Amazon SNS 使用多种策略协同工作来提供消息持久性：

- 已发布的消息存储在多个地理位置分隔的服务器和数据中心之间。
- 如果订阅的终端节点不可用，Amazon SNS 将运行传输重试策略。
- 要保留在传输重试策略结束之前未传输的任何消息，您可以创建死信队列。

消息归档、重播和分析

您可以通过多种方式使用 Amazon SNS 来归档消息，包括将 Firehose 传输流订阅到 SNS 主题，这使您能够向分析端点 [如 Amazon Simple Storage Service（Amazon S3）存储桶、Amazon Redshift 表等] 发送通知。此外，Amazon SNS FIFO 主题支持将消息归档与重播功能作为无代码、就地消息归档功能，这可让主题所有者在其主题中存储（或归档）消息。然后，主题订阅用户可以将归档的消息检索（或重播）回订阅的端点。有关更多信息，请参阅FIFO 主题的 Amazon SNS 消息归档与重播功能。

消息属性

Amazon SNS 消息属性让您可以提供有关消息的任意元数据。

消息筛选

默认情况下，每个订阅者会收到发布到该主题的每条消息。要仅接收一部分消息，订阅者必须将筛选策略分配给主题订阅。订阅者还可以定义筛选策略范围，以启用基于有效负载或基于属性的筛选。筛选策略范围的默认值为 `MessageAttributes`。当传入消息属性与筛选策略属性匹配时，消息将传输到订阅的终端节点。否则，消息将被筛选掉。当筛选策略范围为 `MessageBody` 时，筛选策略属性将与有效负载进行匹配。有关更多信息，请参阅 消息筛选。

消息安全性

服务器端加密使用提供的加密密钥保护存储在 Amazon SNS 主题中的消息内容。 Amazon KMS有关更多信息，请参阅the section called “使用服务器端加密保护数据安全”您还可以在 Amazon SNS 和您的虚拟私有云 (VPC) 之间建立私有连接。有关更多信息，请参阅。the section called “使用 VPC 端点保护流量”

Amazon 亚马逊 SNS 常用的服务

将 Amazon SNS 与多种 Amazon Web Services 云 服务集成，以增强消息处理、改善访问控制、启用事件驱动型处理并实现资源自动化。这种集成优化了性能，增强了安全性，并简化了操作。

Amazon CloudWatch

Amazon 为 Amazon SNS CloudWatch 提供监控和可观察性，帮助您跟踪消息传送、检测异常情况和解决问题。通过使用 CloudWatch，您可以：

- 监控 Amazon SNS 指标，例如跨主题和订阅发布、传送或失败的消息数量。
- 设置 CloudWatch 警报，以便在 Amazon SNS 指标超过预定义阈值（例如高交付失败率或限制）时触发自动操作。
- 使用 CloudWatch 日志捕获发送到 HTTP/S、Lambda 和 Amazon SQS 终端节点的消息的 Amazon SNS 传输状态，以进行调试和审计。

有关更多信息，请参阅 使用监控亚马逊 SNS 主题 CloudWatch。

Amazon SQS

Amazon SQS 是一项完全托管的消息队列服务，可在分布式软件组件之间实现安全、持久且可扩展的通信。它通过缓冲消息、确保可靠传送和防止因消息丢失而导致系统故障，帮助分离应用程序架构。亚马逊 SQS 通过以下方式与亚马逊 SNS 集成：

- 死信队列 — Amazon SNS 可以将无法送达的消息路由到 Amazon SQS 死信队列，以进行故障排除和重新处理。
- 主题订阅 — 您可以为亚马逊 SQS 队列订阅亚马逊 SNS 主题，从而允许亚马逊 SNS 使用亚马逊 SQS 将消息分发给多个消费者。
- FIFO 队列支持 — Amazon SQS FIFO 队列可以订阅 Amazon SNS FIFO 主题，从而确保严格的消息排序和一次性处理。标准 Amazon SQS 队列也可以订阅 Amazon SNS 主题，但不能保证按顺序传送消息或删除重复数据。

Amazon CloudFormation

Amazon CloudFormation 使用基础设施即代码 (IaC) 自动配置和管理 Amazon 资源，包括 Amazon SNS 主题和订阅。借助 Amazon CloudFormation，您可以：

- 在可重复使用、受版本控制的模板中定义 Amazon SNS 主题、订阅和权限。
- 确保跨多个地区一致部署 Amazon SNS 资源 Amazon Web Services 账户 。
- 使用更改集@@ 更新或修改 Amazon SNS 配置，无需人工干预。

有关更多信息，请参阅 用户指南。Amazon CloudFormation

Amazon CloudTrail

CloudTrail 提供对 Amazon SNS 的 API 活动的可见性，帮助您监控和审核对亚马逊 SNS 主题、订阅和消息的访问权限。借助 CloudTrail，您可以：

- 跟踪对 Amazon SNS 的 API 调用，包括谁访问或修改了主题、订阅和权限。
- 出于安全和合规目的，通过分析日志来@@ 检测未经授权或意外的活动。
- 与亚马逊集成 Amazon Security Hub， CloudWatch 或者根据异常的亚马逊 SNS 操作创建提醒。

有关更多信息，请参阅 使用记录 Amazon SNS API 调用 Amazon CloudTrail。

Amazon Lambda

Amazon Lambda 是一项无服务器计算服务，可根据事件自动运行您的代码，无需预置或管理服务器。它允许您构建事件驱动的应用程序，这些应用程序可以自动扩展并在高度可用的计算环境中执行。

亚马逊 SNS 允许您订阅 Lambda 函数来订阅亚马逊 SNS 主题，从而与 Lambda 集成。当 Amazon SNS 主题收到消息时，它可以触发 Lambda 函数，从而实现实时处理、自动化和应用程序逻辑执行。这种集成通常用于：

- 事件驱动处理-自动触发函数以响应 Amazon SNS 消息。
- 数据转换-在将 Amazon SNS 消息转发到其他服务之前，对其进行修改或筛选。
- 自动化工作流程-处理应用程序警报、系统监控或事件编排的通知。

Amazon Identity and Access Management (IAM)

IAM 为 Amazon 资源提供安全的访问控制，允许您管理谁可以访问您的 Amazon SNS 主题、他们可以执行哪些操作以及在什么条件下可以执行哪些操作。使用 IAM，您可以：

- 在@@ 用户和服务与 Amazon SNS 主题进行交互之前，对其进行身份验证。

- 定义精细权限以指定用户或角色可以发布、订阅或管理哪些 Amazon SNS 主题。
- 使用基于身份的策略来实施安全最佳实践，例如限制对特定 Amazon Web Services 账户的 IP 地址或条件的访问。

有关更多信息，请参阅 将基于身份的策略用于 Amazon SNS。

Amazon Key Management Service (Amazon KMS)

Amazon KMS 通过启用服务器端加密 (SSE) 来提高消息机密性，从而增强 Amazon SNS 的安全性。借助 Amazon KMS，您可以：

- 使用 Amazon托管或客户管理的@@ 加密密钥（ ）对静态的 Amazon SNS 消息进行加密。CMKs
- 通过定义限制谁可以发布或订@@ 阅的精细密钥策略来控制对 Amazon SNS 主题的访问权限。
- 通过审核密钥使用情况，确保遵守安全和监管要求 Amazon CloudTrail。

有关更多信息，请参阅 管理 Amazon SNS 加密密钥和成本。

Amazon X-Ray

X-Ray 提供对 Amazon SNS 的跟踪，帮助您分析和调试通过事件驱动架构的消息流。使用 X-Ray，您可以：

- 追踪跨多个终端节点（例如 Lambda Amazon Web Services 服务、亚马逊 SQS 和 HTTP/S 终端节点）的亚马逊 SNS 消息传送情况。
- 通过可视化消息发布、交付和处理所需的时间来@@ 识别延迟瓶颈。
- 检测 Amazon SNS 消息流中的错误和重试次数，以排除传输失败或处理时间缓慢的问题。

有关更多信息，请参阅 Amazon SNS 中的主动跟踪。

将 Amazon SNS 与软件开发工具包配合使用 Amazon

Amazon 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档
Amazon CLI
适用于 Java 的 Amazon SDK

SDK 文档
适用于 JavaScript 的 Amazon SDK
适用于 .NET 的 Amazon SDK
适用于 PHP 的 Amazon SDK
Amazon Tools for PowerShell
适用于 Python (Boto3) 的 Amazon SDK
适用于 Ruby 的 Amazon SDK
适用于 SAP ABAP 的 Amazon SDK

有关特定于 Amazon SNS 的示例，请参阅 使用 Amazon SNS 的代码示例 Amazon SDKs。

创建 Amazon SNS 主题，并发布消息

本主题提供了管理 Amazon SNS 资源的基本步骤，特别侧重于主题、订阅和消息发布。首先，您将为 Amazon SNS 设置必要的访问权限，确保您拥有创建和管理 Amazon SNS 资源的正确权限。接下来，您将创建一个新的 Amazon SNS 主题，该主题将作为管理消息和向订阅用户发送消息的中央枢纽。创建主题后，您将继续创建对此主题的订阅，允许特定的端点接收发布到该主题的消息。

主题和订阅完成后，您将向该主题发布一条消息，观察 Amazon SNS 如何高效地将消息传送到所有已订阅的端点。最后，您将学习如何删除订阅和主题，从而完成您管理的 Amazon SNS 资源的生命周期。这种方法可以让您清楚地了解 Amazon SNS 中的基本操作，使您具备使用 Amazon SNS 控制台管理消息工作流程所需的实用技能。

设置 Amazon SNS 的访问权限

在首次使用 Amazon SNS 之前，您必须完成以下步骤。

创建一个 Amazon Web Services 账户 和一个 IAM 用户

要访问任何 Amazon 服务，必须先创建一个Amazon Web Services 账户。您可以使用 Amazon Web Services 账户 来查看您的活动和使用情况报告，以及管理身份验证和访问权限。

注册获取 Amazon Web Services 账户

如果您没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

报名参加 Amazon Web Services 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话，要求使用电话键盘输入一个验证码。

   当您注册时 Amazon Web Services 账户，将创建一个 Amazon Web Services 账户 root 用户。根用户有权访问该账户中的所有 Amazon Web Services 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

Amazon 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择 “我的账户”，查看您当前的账户活动并管理您的账户。

保护 IAM 用户

注册后 Amazon Web Services 账户，请开启多重身份验证 (MFA)，保护您的管理用户。有关说明，请参阅《IAM 用户指南》中的 为 IAM 用户启用虚拟 MFA 设备（控制台）。

要允许其他用户访问您的 Amazon Web Services 账户 资源，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅《IAM 用户指南》中的以下主题：

- 在你的 IAM 用户中创建 Amazon Web Services 账户
- 适用于 Amazon 资源的访问权限管理
- 基于 IAM 身份的策略示例

后续步骤

现在，您已准备好使用 Amazon SNS，可以从以下入手：

1. 创建 Amazon SNS 主题
2. 在 Amazon SNS 主题中创建订阅
3. 发布 Amazon SNS 消息
4. 删除 Amazon SNS 订阅和主题

创建 Amazon SNS 主题

Amazon SNS 主题是一个逻辑访问点，可充当通信通道。主题允许您对多个终端节点（例如 Amazon SQS Amazon Lambda、HTTP/S 或电子邮件地址）进行分组。

要广播使用需要其消息的多个其他服务（例如，结算和执行系统）的消息创建器系统（例如，电子商务网站）的消息，您可以为创建器系统创建主题。

第一个也是最常见的 Amazon SNS 任务是创建主题。本页显示了如何使用 Amazon Web Services Management Console 适用于 Java 的 Amazon SDK、和， 适用于 .NET 的 Amazon SDK 来创建主题。

在创建过程中，您可以选择主题类型（标准或 FIFO）并命名主题。创建主题后，无法更改主题类型或名称。在创建主题期间，所有其他配置选项都是可选的，您可以稍后对其进行编辑。

> **Important**
>
> 请勿在主题名称中添加个人身份信息 (PII) 或其他机密或敏感信息。其他 Amazon Web Services 可以访问主题名称，包括 CloudWatch 日志。主题名称不适合用于私有或敏感数据。

要使用创建主题 Amazon Web Services Management Console

在 Amazon SNS 中创建主题为消息分发奠定了基础，使您能够发布可以扇出给多个订阅用户的消息。此步骤对于配置主题的类型、加密设置和访问策略至关重要，可确保主题符合组织的安全、合规性和运营要求。

1. 登录 Amazon SNS 控制台。
2. 请执行以下操作之一：
   - 如果 Amazon Web Services 账户 之前未在您的主题下创建过任何主题，请阅读主页上对 Amazon SNS 的描述。
   - 如果 Amazon Web Services 账户 之前已在您的下方创建过主题，请在导航面板上选择主题。
3. 在 Topics（主页）页面上，选择 Create topic（创建主题）。
4. 在 Create topic（创建主题）页面上，在 Details（详细信息）部分中，执行以下操作：
   a. 对于 Type（类型），选择主题类型（标准或者FIFO）。
   b. 输入主题的名称。对于 FIFO 主题，将 .fifo 添加到名称的末尾。
   c. （可选）输入主题的显示名称。

      > **Important**
      >
      > 订阅电子邮件端点时，Amazon SNS 主题显示名称和发送电子邮件地址（例如 no-reply@sns.amazonaws.com）的组合字符计数不得超过 320 个 UTF-8 字符。在为 Amazon SNS 主题配置显示名称之前，您可以使用第三方编码工具验证发送地址的长度。

   d. （可选）对于 FIFO 主题，您可以选择基于内容的消息重复数据删除以启用默认的消息重复数据删除。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息重复数据删除。
5. （可选）展开加密部分并执行以下操作。有关更多信息，请参阅 使用服务器端加密保护 Amazon SNS 数据安全。

a. 选择启用加密。

b. 指定密 Amazon KMS 钥。有关更多信息，请参阅 关键术语。

对于每个 KMS 类型，都会显示 Description（描述）、Account（账户）和 KMS ARN。

> **Important**
>
> 如果您不是 KMS 的拥有者，或者您登录的账户没有 `kms:ListAliases` 和 `kms:DescribeKey` 权限，则无法在 Amazon SNS 控制台上查看有关 KMS 的信息。
> 要求 KMS 的拥有者授予您这些权限。有关更多信息，请参阅 Amazon Key Management Service 开发人员指南中的 Amazon KMS API 权限：操作和资源参考。

- 默认选择适用于 Amazon SNS 的 Amazon 托管 KMS（默认）alias/aws/sns。

> **Note**
>
> 记住以下内容：
>
> - 首次使用为主题指定适用于 Amazon SNS 的 Amazon 托管 KMS 时， Amazon KMS 会为亚马逊 SNS 创建 Amazon 托管 KMS。 Amazon Web Services Management Console
> - 或者，在启用 SSE 的情况下首次对主题使用`Publish`操作时， Amazon KMS 会为 Amazon SNS 创建 Amazon 托管 KMS。

- 要使用 Amazon 账户中的自定义 KMS，请选择 KMS 密钥字段，然后从列表中选择自定义 KMS。

> **Note**
>
> 有关创建自定义密钥的说明 KMSs，请参阅《Amazon Key Management Service 开发者指南》中的创建密钥

- 要使用来自您的 Amazon 账户或其他账户的自定义 KMS ARN，请将其输入到 KMS 密钥字段中。 Amazon

6. （可选）默认情况下，只有主题拥有者才能发布或订阅主题。要配置其他访问权限，请展开访问策略部分。有关更多信息，请参阅Amazon SNS 中的 Identity and Access Management和用于 Amazon SNS 访问控制的示例案例。

> **Note**
>
> 使用控制台创建主题时，默认策略使用 `aws:SourceOwner` 条件键。此密钥类似于 `aws:SourceAccount`。

7. （可选）要配置 Amazon SNS 重试失败消息传输尝试的方式，请展开 Delivery retry policy (HTTP/S)（传输重试策略 (HTTP/S)）部分。有关更多信息，请参阅 Amazon SNS 消息传输重试。
8. （可选）要配置 Amazon SNS 如何记录向其发送的消息 CloudWatch，请展开传送状态记录部分。有关更多信息，请参阅 Amazon SNS 消息传输状态。
9. （可选）要将元数据标签添加到主题中，请展开标签部分，输入一个键和值（可选），然后选择添加标签。有关更多信息，请参阅 Amazon SNS 主题标记。
10. 选择创建主题。

    主题已创建并显示***MyTopic***页面。

    主题的名称、ARN、（可选）显示名称和主题所有者的 Amazon 账户 ID 显示在详细信息部分中。
11. 将主题 ARN 复制到剪贴板，例如：

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

使用 Amazon SDK 创建主题

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置Amazon SDKs 和凭据文件。

以下代码示例演示如何使用 `CreateTopic`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用特定的名称创建主题。

```
    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;
        }
    }
```

创建一个包含名称以及特定 FIFO 和重复数据消除属性的新主题。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考CreateTopic中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考CreateTopic中的。

CLI

Amazon CLI

创建 SNS 主题

以下 `create-topic` 示例将创建名为 `my-topic` 的 SNS 主题。

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

输出：

```
{
    "ResponseMetadata": {
```

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

有关更多信息，请参阅《Amazon 命令行界面用户指南》中的在 Amazon SQS 和 Amazon SNS 中使用命令Amazon 行界面。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateTopic中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考CreateTopic中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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 "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateTopic中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

```
import { snsClient } from "../libs/snsClient.js";

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

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CreateTopic中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

```
        CreateTopicRequest {
            name = topicName
        }

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

- 有关 API 的详细信息，请参阅适用CreateTopic于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考CreateTopic中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用CreateTopic于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考CreateTopic中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用CreateTopic于 Rust 的Amazon SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   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.
```

- 有关 API 的详细信息，请参阅适用CreateTopic于 S AP 的Amazon SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
        }
```

- 如需了解 API 的详细信息，请参阅适用CreateTopic于 S wift 的Amazon SDK API 参考。

在 Amazon SNS 主题中创建订阅

要接收发布至某个主题的消息，您必须订阅一个到该主题的终端节点。在为终端节点订阅主题后，此终端节点会开始接收发布到关联主题的消息。

> **Note**
>
> HTTP (S) 端点、电子邮件地址和其他 Amazon 资源 Amazon Web Services 账户 需要确认订阅后才能接收消息。

要为终端节点订阅 Amazon SNS 主题

通过为端点订阅 Amazon SNS 主题，可以将消息传送到指定的端点，从而确保当向该主题发布消息时，正确的系统或用户会收到通知。此步骤对于将主题与使用者（无论他们是应用程序、电子邮件收件人还是其他服务）联系起来至关重要，从而实现跨系统的无缝通信。

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择订阅。
3. 在订阅页面上，选择创建订阅。
4. 在创建订阅页上的详细信息部分，执行以下操作：

   a. 对于 Topic ARN（主题 ARN），选择主题的 Amazon Resource Name (ARN)。例如，此值是您创建 Amazon SNS 主题时生成的 Amazon ARN。`arn:aws:sns:us-east-2:123456789012:your_topic`

   b. 对于 Protocol（协议），选择终端节点类型。可用的终端节点类型包括：

      - HTTP/HTTPS
      - 电子邮件/电子邮件-JSON
      - Amazon Data Firehose
      - Amazon SQS

      > **Note**
      > 要订阅到 SNS FIFO 主题，请选择该选项。

      - Amazon Lambda
      - 平台应用程序终端节点
      - 短信

   c. 对于 Endpoint（终端节点），输入终端节点值，例如电子邮件地址或 Amazon SQS 队列的 ARN。

   d. 仅限于 Firehose 端点：对于订阅角色 ARN，指定您为写入到 Firehose 传输流创建的 IAM 角色的 ARN。有关更多信息，请参阅 订阅 Firehose 传输流到 Amazon SNS 主题的先决条件。

   e. （可选）对于 Firehose、Amazon SQS、HTTP/S 端点，您还可以启用原始消息传输。有关更多信息，请参阅 Amazon SNS 原始消息传输。

f. （可选）要配置筛选策略，请展开 Subscription filter policy（订阅筛选策略）部分。有关更多信息，请参阅 Amazon SNS 订阅筛选策略。

g. （可选）要启用基于有效负载的筛选，请将 `Filter Policy Scope` 配置为 `MessageBody`。有关更多信息，请参阅 Amazon SNS 订阅筛选策略范围。

h. （可选）要为订阅配置死信队列，请展开 Redrive policy (dead-letter queue)（重新驱动策略（死信队列））部分。有关更多信息，请参阅 Amazon SNS 死信队列。

i. 选择创建订阅。

   控制台将创建订阅并打开订阅的 Details（详细信息）页面。

发布 Amazon SNS 消息

在创建 Amazon SNS 主题并为终端节点订阅主题后，可以将消息发布到主题。发布消息时，Amazon SNS 会尝试将消息传输给订阅的终端节点。

要使用 Amazon Web Services Management Console将消息发布到 Amazon SNS 主题

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页上，选择一个主题，然后选择Publish message（发布主题）。

   控制台将打开 Publish message to topic（将消息发布到主题）页面。
4. 在 Message details（消息详细信息）部分中，执行以下操作：

   a. （可选）输入消息 Subject（主题）。

   b. 对于 FIFO topic（FIFO 主题），输入 Message group ID（消息组 ID）。同一消息组中的消息按消息的发布顺序传输。

   c. 对于 FIFO 主题，请输入 Message deduplication ID（消息重复数据删除 ID）。如果您为主题启用了 Content-based message deduplication（基于内容的消息重复数据删除）设置，则此 ID 为可选项。

   d. （可选）对于 mobile push notifications（移动推送通知），输入 Time to Live (TTL)（存活时间 (TTL)）值（以秒为单位）。这是推送通知服务（例如 Apple 推送通知服务 (APNs) 或 Firebase Cloud Messaging (FCM)）将消息传递到端点所需的时间。
5. 在 Message body (消息正文) 部分中，执行以下操作之一：

a. 选择 Identical payload for all delivery protocols（完全相同负载用于所有交付协议），然后输入消息。

b. 选择 Custom payload for each delivery protocol（对每个交付协议使用自定义负载），然后输入 JSON 对象定义要发送给每个协议的消息。

有关更多信息，请参阅 使用特定于平台的有效载荷发布 Amazon SNS 通知。

6. 在 Message attributes（消息属性）部分中，添加您希望 Amazon SNS 与订阅属性 `FilterPolicy` 相匹配的任何属性，以确定订阅的终端节点是否对发布的消息感兴趣。

a. 对于 Type（类型），选择属性类型，例如 String.Array。

> **Note**
>
> 对于属性类型 String.Array，请将该数组放入方括号 (`[]`) 内。在该数组内，将字符串值加入双引号内。数字以及关键字 `true`、`false` 和 `null` 无需加引号。

b. 输入属性名称，例如 `customer_interests`。

c. 输入属性值，例如 `["soccer", "rugby", "hockey"]`。

如果属性类型为 String、String.Array 或 Number，Amazon SNS 会首先依据订阅的筛选策略（如果存在）来评估该消息属性，然后再将消息发送至该订阅，前提是筛选策略范围没有明确设置为 `MessageBody`。

有关更多信息，请参阅 Amazon SNS 消息属性。

7. 选择发布消息。

消息将发布到主题，且控制台将打开主题的 Details（详细信息）页面。

使用 Amazon SDK 将消息发布到主题

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置Amazon SDKs 和凭据文件。

以下代码示例演示如何使用 `Publish`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

向主题发布消息。

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

使用组、复制和属性选项向主题发布消息。

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

将用户的选择应用于发布操作。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 Publish。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

发布带有属性的消息。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 Publish。

CLI

Amazon CLI

示例 1：向主题发布消息

以下 publish 示例将指定消息发布到指定 SNS 主题。该消息来自一个文本文件，您可以在该文件中包含换行符。

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

message.txt 的内容：

```
Hello World
```

```
Second Line
```

输出：

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

示例 2：向电话号码发布 SMS 消息

以下 `publish` 示例将消息 `Hello world!` 发布到电话号码 `+1-555-555-0100`。

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

输出：

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

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 Publish。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 Publish。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 Publish。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                    if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
```

```
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

使用组、复制和属性选项向主题发布消息。

```
  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();
    }
  }
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考中的 Publish。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

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

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 Publish。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 Publish。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例显示发布一条 MessageAttribute 声明为内联的消息。

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

示例 2：此示例显示发布一条事先 MessageAttributes 声明了多条消息的消息。

```
$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
```

- 有关 API 的详细信息，请参阅在 Amazon Tools for PowerShell Cmdlet 参考中发布 (V4)。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

发布包含属性的消息，以便订阅可以根据属性进行筛选。

```
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
```

发布基于订阅用户的协议采取不同形式的消息。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 Publish。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 Publish。

Rust

SDK for Rust

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 Publish。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的发布。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)")
```

- 有关 API 的详细信息，请参阅在 Amazon Swift 开发工具包中发布 API 参考。

通过 Amazon SNS 和 Amazon S3 发布大型消息

要发布很大的 Amazon SNS 消息，您可以使用适用于 Java 的 Amazon SNS 扩展型客户端库或适用于 Python 的 Amazon SNS 扩展型客户端库。对于大于当前最大值 256KB（最大为 2GB）的消息，这些库非常有用。这两个库将实际有效负载保存到 Amazon S3 桶，并将存储的 Amazon S3 对象的引用发布到 Amazon SNS 主题。订阅的 Amazon SQS 队列可以使用适用于 Java 的 Amazon SQS 扩展客户端库从 Amazon S3 中取消引用并检索负载。其他端点（如 Lambda）可以使用 Amazon的有效负载卸载 Java 公共库来取消引用并检索有效负载。

> **Note**
>
> Amazon SNS 扩展型客户端库与标准主题和 FIFO 主题都兼容。

适用于 Java 的 Amazon SNS 扩展型客户端库

先决条件

以下是使用适用于 Java 的 Amazon SNS 扩展型客户端库的先决条件：

- 一个 Amazon 软件开发工具包。本页上的示例使用 Amazon Java 开发工具包。要安装和设置 SDK，请参阅《适用于 Java 的 Amazon SDK 开发者指南》中的 “设置 Amazon 适用于 Java 的 SDK”。
- 并 Amazon Web Services 账户 具有正确的凭据。要创建 Amazon Web Services 账户，请导航到Amazon 主页，然后选择创建 Amazon 帐户。按照说明进行操作。

有关证书的信息，请参阅《适用于 Java 的 Amazon SDK 开发人员指南》中的 “设置用于开发的 Amazon 凭证和区域”。

- Java 8 或更高版本。
- 适用于 Java 的 Amazon SNS 扩展型客户端库（也可从 Maven 中获得）。

配置消息存储

Amazon SNS 扩展客户端库使用负载卸载 Java 公共库 Amazon 进行消息存储和检索。您可以配置以下 Amazon S3 消息存储选项：

- 自定义消息大小阈值-负载和属性超过此大小的消息将自动存储在 Amazon S3 中。
- `alwaysThroughS3fla@@ g` — 将此值设置为`true`，以强制将所有消息负载存储在 Amazon S3 中。例如：

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

- 自定义 KMS 密钥 — 用于在 Amazon S3 存储桶中进行服务器端加密的密钥。
- 存储桶名称-用于存储消息负载的 Amazon S3 存储桶的名称。

示例：使用存储在 Amazon S3 中的负载将消息发布到 Amazon SNS

以下代码示例展示了如何：

- 创建示例主题和队列。
- 订阅队列以接收来自主题的消息。
- 发布测试消息。

消息负载存储在 Amazon S3，以及发布到的引用中。Amazon SQS 扩展客户端用于接收消息。

适用于 Java 的 SDK 1.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

要发布大型消息，请使用适用于 Java 的 Amazon SNS 扩展客户端库。您发送的消息将引用包含实际消息内容的 Amazon S3 对象。

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

其他终端节点协议

Amazon SNS 和 Amazon SQS 库都使用 Amazon的负载卸载 Java 公共库通过 Amazon S3 存储和检索消息负载。任何启用 Java 的终端节点（例如，在 Java 中实施的 HTTPS 终端节点）都可以使用相同的库来取消引用消息内容。

无法使用有效负载卸载 Java 公共库的终端节点仍然 Amazon 可以发布存储在 Amazon S3 中的有效负载的消息。以下是由上面的代码示例发布的 Amazon S3 引用的示例：

```
[
  "software.amazon.payloadoffloading.PayloadS3Pointer",
```

```
  {
    "s3BucketName": "extended-client-bucket",
    "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
  }
]
```

适用于 Python 的 Amazon SNS 扩展型客户端库

先决条件

以下是使用适用于 Python 的 Amazon SNS 扩展型客户端库的先决条件：

- 一个 Amazon 软件开发工具包。本页上的示例使用 Amazon Python SDK Boto3。要安装和设置 SDK，请参阅 Amazon SDK for Python 文档。
- 并 Amazon Web Services 账户 具有正确的凭据。要创建 Amazon Web Services 账户，请导航到Amazon 主页，然后选择创建 Amazon 帐户。按照说明进行操作。

  有关凭证的信息，请参阅《Amazon SDK for Python 开发人员指南》中的凭证。
- Python 3.x（或更高版本）和 pip。
- 适用于 Python 的 Amazon SNS 扩展型客户端库（也可从 PyPI 中获得）。

配置消息存储

Boto3 Amazon SN S 客户端、主题PlatformEndpoint和对象上提供了以下属性，用于配置 Amazon S3 消息存储选项。

- **`large_payload_support`**— 用于存储大型消息的 Amazon S3 存储桶名称。
- **`use_legacy_attribute`**— 如果`True`，则所有已发布的消息都使用旧版保留消息属性 (`SQSLargePayloadSize`)，而不是当前的保留消息属性 (`ExtendedPayloadSize`)。
- **`message_size_threshold`** – 在大型消息桶中存储消息的阈值。不能小于或`0`大于`262144`。默认值为 `262144`。
- **`always_through_s3`** – 如果为 `True`，则所有消息都存储在 Amazon S3 中。默认值为 `False`。
- **`s3_client`**— 用于将`client`对象存储到亚马逊 S3 的 Boto3 Amazon S3 对象。如果您想控制 Amazon S3 客户端（例如，自定义 Amazon S3 配置或证书），请使用此选项。如果之前未设置，`boto3.client("s3")`则默认为首次使用时。

示例：使用存储在 Amazon S3 中的有效负载将消息发布到 Amazon SNS

以下代码示例展示了如何：

- 创建示例 Amazon SNS 主题和 Amazon SQS 队列。
- 将策略附加到亚马逊 SQS 队列以接收来自亚马逊 SNS 主题的消息。
- 订阅队列以接收来自主题的消息。
- 使用 Amazon SNS 扩展客户端、主题资源和 PlatformEndpoint 资源发布测试消息。
- 消息有效负载存储在 Amazon S3 中，并发布对它的引用。
- 打印队列中已发布的消息以及从 Amazon S3 检索到的原始消息。

要发布大型消息，请使用适用于 Python 的 Amazon SNS 扩展型客户端库。您发送的消息将引用包含实际消息内容的 Amazon S3 对象。

```
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)
```

输出

```
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 消息属性

Amazon SNS 支持传输消息属性，这些属性支持您提供与消息相关的结构化元数据项目（如时间戳、地理空间数据、签名和标识符）。对于 SQS 订阅，在启用 Raw Message Delivery（原始消息传输）时，最多可以发送 10 个消息属性。要发送 10 个以上的消息属性，必须禁用 Raw Message Delivery（原始消息传输）。所具有的消息属性（定向到启用了原始消息传输的 Amazon SQS 订阅）超过 10 个的消息将作为客户端错误被丢弃。

消息属性是可选的，并独立于消息正文（但随之一起发送）。接收方可以使用此信息来决定如何处理消息，而不必先处理消息正文。

有关使用 Amazon Web Services Management Console 或发送带有属性的消息的信息 适用于 Java 的 Amazon SDK，请参阅要使用 Amazon Web Services Management Console将消息发布到 Amazon SNS 主题教程。

> **Note**
>
> 当消息结构为 String 而不是 JSON 时，仅发送消息属性。

您还可以使用消息属性，帮助构造移动终端节点的推送通知消息。在这种情况下，消息属性仅用于帮助构造推送通知消息。这些属性不会被传递到终端节点，就像在发送带有消息属性的消息到 Amazon SQS 终端节点时一样。

您还可以使用消息属性来让消息变为可通过订阅筛选策略进行筛选。可以将筛选策略应用于主题订阅。应用了筛选策略且筛选策略范围设置为 `MessageAttributes`（默认值）时，订阅将只接收具有策略接受的属性的那些消息。有关更多信息，请参阅 Amazon SNS 消息筛选。

> **Note**
>
> 使用消息属性进行筛选时，该值必须是有效的 JSON 字符串。这样做可以确保将消息传送到启用了消息属性筛选的订阅。

消息属性项目和验证

每个消息属性包含以下项目：

- Name – 消息属性的名称可以包含以下字符：A-Z、a-z、0-9、下划线 (_)、连字符 (-) 和句点 (.)。名称不得以句点开头或结尾，并且不应包含连续句点。名称区分大小写，且必须在消息的所有属性名

称中是唯一的。名称最多可以有 256 个字符。名称不能以 `AWS.` 或 `Amazon.`（或任何大小写变化形式）开头，因为这些前缀已预留以供 Amazon Web Services 使用。

- Type – 受支持的消息属性数据类型有 `String`、`String.Array`、`Number` 和 `Binary`。数据类型在内容方面具有与消息正文相同的限制。想要了解更多信息，请参阅消息属性数据类型和验证部分。
- Value – 用户指定的消息属性值。对于字符串数据类型，value 属性必须遵循与消息正文相同的内容限制。但是，如果使用消息属性进行筛选，则该值必须是有效的 JSON 字符串，以确保与 Amazon SNS 订阅筛选策略兼容。有关更多信息，请参阅 Amazon Simple Notification Service API 参考中的发布操作。

名称、类型和值都不得为空或 null。此外，消息正文也不应为空或 null。消息属性的所有部分 (包括名称、类型和值) 都包含在消息大小限制中，该限制当前是 256 KB。

消息属性数据类型和验证

消息属性数据类型指示 Amazon SNS 处理消息属性值的方式。例如，如果类型是数字，则 Amazon SNS 会验证它是否为数字。

除非另有说明，否则 Amazon SNS 支持所有终端节点的以下逻辑数据类型：

- String – 字符串是使用 UTF-8 二进制编码的 Unicode。有关代码值的列表，请参阅 http://en.wikipedia。 org/wiki/ASCII#ASCII_printable_characters。

> **Note**
>
> 消息属性中不支持代理值。例如，使用代理值来表示表情符号将收到以下错误：`Invalid attribute value was passed in for message attribute`。

- String.Array – 格式为字符串的阵列，可以包含多个值。这些值可以是字符串、数字或关键字 `true`、`false` 和 `null`。数字或布尔类型的 String.Array 不需要引号。多个 String.Array 值用逗号分隔。

  Amazon Lambda 订阅不支持此数据类型。如果您为 Lambda 终端节点指定此数据类型，它会作为 Amazon SNS 传输给 Lambda 的 JSON 负载中的 `String` 数据类型传递。
- Number – 数字是正或负整数或是浮点数。数字具有足够的范围和精度，以便包含整数、浮点数和双精度数通常支持的大多数可能值。数字的值可以介于 -10^9 到 10^9 之间，精确至小数点后 5 位数。系统会删减开头和结尾的 0。

Amazon Lambda 订阅不支持此数据类型。如果您为 Lambda 终端节点指定此数据类型，它会作为 Amazon SNS 传输给 Lambda 的 JSON 负载中的 `String` 数据类型传递。

- Binary – 二进制类型属性可以存储任何二进制数据，例如压缩数据、加密数据或图像。

为移动推送通知预留的消息属性

下表列出了可用于构造推送通知消息的移动推送通知服务的预留消息属性：

推送通知服务	预留的消息属性
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`

推送通知服务	预留的消息属性
	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] 如果消息属性不符合要求，Apple 将拒绝 Amazon SNS 通知。如需了解更多详情，请参阅 Apple 开发者网站上的 “向发送通知请求”。 APNs

Amazon SNS 消息批处理

什么是消息批处理？

在各个 Publish API 请求中向标准或 FIFO 主题发布消息的替代方法，该方法使用 Amazon SNS PublishBatch API 在单个 API 请求中最多发布 10 条消息。批量发送消息可以帮助您利用 Amazon SNS 将与连接分布式应用程序（A2A 消息收发）或者向人们发送通知（A2P 消息收发）相关的成本最高降低 10 倍。根据您运营所在区域，Amazon SNS 对每秒可以向某个主题发布的消息数量设定了配额。有关 API 限额的更多信息，请参阅《Amazon Web Services 一般参考》中的 Amazon SNS 端点和限额页面。

> **Note**
>
> 您在单个 PublishBatch API 请求中发送的所有消息的总大小不能超过 262,144 字节 (256 KiB)。
>
> PublishBatch API 将相同的 Publish API 操作用于 IAM 策略。

消息批处理是如何工作的？

使用 PublishBatch API 发布消息与使用 Publish API 发布消息类似。主要的区别在于，需要给 PublishBatch API 请求中的每条消息分配一个唯一的批处理 ID（最多 80 个字符）。这样，Amazon SNS 可以为批处理中的每条消息返回单独的 API 响应，以确认每条消息均已发布或发生故障。对于发布到 FIFO 主题的消息，除了包括分配唯一的批处理 ID 之外，还需要为每条单独的消息包括 MessageDeduplicationID 和 MessageGroupId。

示例

向标准主题发布一批 10 条的消息

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

向 FIFO 主题发布一批 10 条的消息

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

删除 Amazon SNS 订阅和主题

删除主题后，其关联订阅会异步删除。虽然客户仍然可以访问这些订阅，但即使您使用相同的名称重新创建主题，这些订阅也不再与该主题相关联。如果发布者尝试向已删除的主题发布消息，发布者将收到一条错误消息，指出该主题不存在。同样，任何订阅已删除主题的尝试也会导致错误消息。您无法删除正在等待确认的订阅。Amazon SNS 会在 48 小时后自动删除未确认的订阅。

要删除 Amazon SNS 主题或订阅，请使用 Amazon Web Services Management Console

删除 Amazon SNS 主题或订阅可确保高效的资源管理、防止不必要的使用并保持 Amazon SNS 控制台井然有序。此步骤有助于避免闲置资源带来的潜在成本，并通过移除不再需要的主题或订阅来简化管理。

要使用删除主题 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页面上，选择一个主题，然后选择 Delete（删除）。
4. 在 Delete topic（删除主题）对话框中，输入 `delete me`，然后选择 Delete（删除）。

   控制台将删除主题。

要删除订阅，请使用 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择订阅。
3. 在订阅页面上，选择状态为已确认的订阅，然后选择删除。
4. 在 Delete subscription（删除订阅）对话框中，选择 Delete（删除）。

   控制台删除订阅。

要使用 Amazon 开发工具包删除订阅和主题

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅和工具参考指南中的共享配置Amazon SDKs 和凭据文件。

以下代码示例演示如何使用 `DeleteTopic`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按主题 ARN 删除主题。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DeleteTopic中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DeleteTopic中的。

CLI

Amazon CLI

删除 SNS 主题

以下 `delete-topic` 示例将删除指定的 SNS 主题。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteTopic中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考DeleteTopic中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteTopic中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

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

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteTopic中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

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

- 有关 API 的详细信息，请参阅适用DeleteTopic于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DeleteTopic中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 Python 的Amazon SDK (Boto3) API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 S AP 的Amazon SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSSNS

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

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

- 有关 API 的详细信息，请参阅适用于 S wift 的Amazon SDK API 参考DeleteTopic中。

后续步骤

现在，您已经使用订阅创建了一个主题，并向该主题发送了消息，您可能希望尝试以下操作：

- 浏览 Amazon 开发人员中心。
- 在 Security（安全性）部分中了解如何保护您的数据。
- 为主题启用服务器端加密。
- 使用已订阅的加密 Amazon Simple Queue Service (Amazon SQS) 队列为主题启用服务器端加密。
- 将 Amazon Event Fork Pipelines 订阅到主题。

使用 Amazon SNS FIFO 主题的消息排序和重复数据删除策略

本主题提供有关 Amazon SNS FIFO（先进先出）主题的特征和功能以及它们如何与 Amazon SQS FIFO 队列集成的信息。您将学习如何将这些服务结合使用，以确保严格的消息排序和重复数据删除，这对于需要数据一致性的应用程序来说至关重要。本内容涵盖了 Amazon SNS FIFO 主题有益的特定用例，提供了对消息顺序和唯一性至关重要的场景的见解。

您还将了解消息排序、消息分组的技术细节，以及它们如何影响消息传送。消息重复数据删除主题说明了防止重复消息的机制，确保每条消息只处理一次。此外，您还将了解消息筛选、安全性和耐久性，这些对于维护消息收发系统的完整性和可靠性非常重要。本内容还包括有关消息归档和重播的信息，提供了管理消息历史记录的策略。还提供了实用的代码示例，协助您在自己的应用程序中实现这些特征，让您亲身体验 Amazon SNS FIFO 主题及其与 Amazon SQS FIFO 队列的集成。

Amazon SNS 中的高吞吐量 FIFO 主题

Amazon SNS 中的高吞吐量 FIFO 主题可有效管理高消息吞吐量，同时保持严格的消息顺序，确保处理大量消息的应用程序的可靠性和可扩展性。该解决方案非常适合要求高吞吐量和有序消息传递的场景。要使用高吞吐量 FIFO 主题提高消息吞吐量，建议增加消息组的数量。有关高吞吐量消息配额的更多信息，请参阅中的 Amazon SNS 服务配额。Amazon Web Services 一般参考

Amazon SNS FIFO 主题的高吞吐量用例

以下用例重点介绍了高吞吐量 FIFO 主题的多样化应用，展示了它们在各行业和场景中的有效性：

- 实时数据处理：处理实时数据流（例如事件处理或遥测数据摄取）的应用程序可以受益于高吞吐量 FIFO 主题，以处理持续涌入的消息，同时保留消息顺序以进行准确分析。
- 电子商务订单处理：在维护客户交易顺序至关重要的电子商务平台中，高吞吐量FIFO主题可确保订单按顺序交付，即使在购物旺季也是如此。
- 金融服务：处理高频交易或交易数据的金融机构依靠高吞吐量的FIFO话题以最小的延迟处理市场数据和交易，同时遵守严格的报文排序监管要求。
- 媒体流：流媒体平台和媒体分发服务利用高吞吐量 FIFO 主题来管理媒体文件和流媒体内容的交付，确保用户获得流畅的播放体验，同时保持正确的内容交付顺序

Amazon SNS FIFO 主题的高吞吐量分区和数据分配

对于高吞吐量主题，Amazon SNS 可以跨分区分发 FIFO 主题数据。分区是指主题的容量分配，该容量会在内的多个可用区之间自动复制 Amazon Web Services 区域。您无需管理分区。相反，Amazon SNS 会根据入口速率自动代表您管理分区。

对于 FIFO 主题，在以下情况下，Amazon SNS 会修改主题中的分区数量：

- 如果当前的发布速率接近或超过现有分区所能支持的速度，则会分配额外的分区，直到主题达到区域配额。有关配额的信息，请参阅中的 Amazon SNS 服务配额。Amazon Web Services 一般参考
- 如果当前分区的利用率较低，则分区的数量可能会减少。

分区管理在后台自动进行，对程序是透明的。您的主题和消息随时可用。

> **Note**
>
> 如果您突然大幅增加主题的流量，同时发送量是平时数倍的流量，则可能会出现临时发布 API 限制。这种限制可以持续到重复数据删除窗口的持续时间，而主题则可以向上扩展以适应增加的流量。

按消息组分发数据 IDs

向 FIFO 主题发布消息时，Amazon SNS 使用每条消息的消息组 ID 的值作为内部哈希函数的输入。哈希函数的输出值决定哪个分区处理消息，一个或多个消息组 IDs 可以由给定分区处理。

> **Note**
>
> Amazon SNS 经过优化，可以在 FIFO 主题的分区中均匀分配项目，无论分区数量如何。Amazon 建议您使用可 IDs 包含大量不同值的消息组。

在 Amazon SNS FIFO 主题上启用高吞吐量

默认情况下，Amazon SNS FIFO 主题配置为主题级重复数据删除，这由`FifoThroughputScope`设置为的主题属性控制，吞吐量配额更受限制，请参阅中的 Amazon SNS 服务配额。`Topic`Amazon Web Services 一般参考

要为您的 Amazon SNS FIFO 主题启用高吞吐量，请将`FifoThroughputScope`属性更新为。`MessageGroup`此更改可以通过控制台或使用 Amazon CLI 和 SDK 完成，也可以在主题创建期间进行设置，Amazon SNS 建议这样做，以获得最佳的客户体验并减少主题被限制的机会。

> **Important**
>
> 启用主题后`MessageGroup`，就`FifoThroughputScope`无法将其恢复为`Topic`吞吐量。

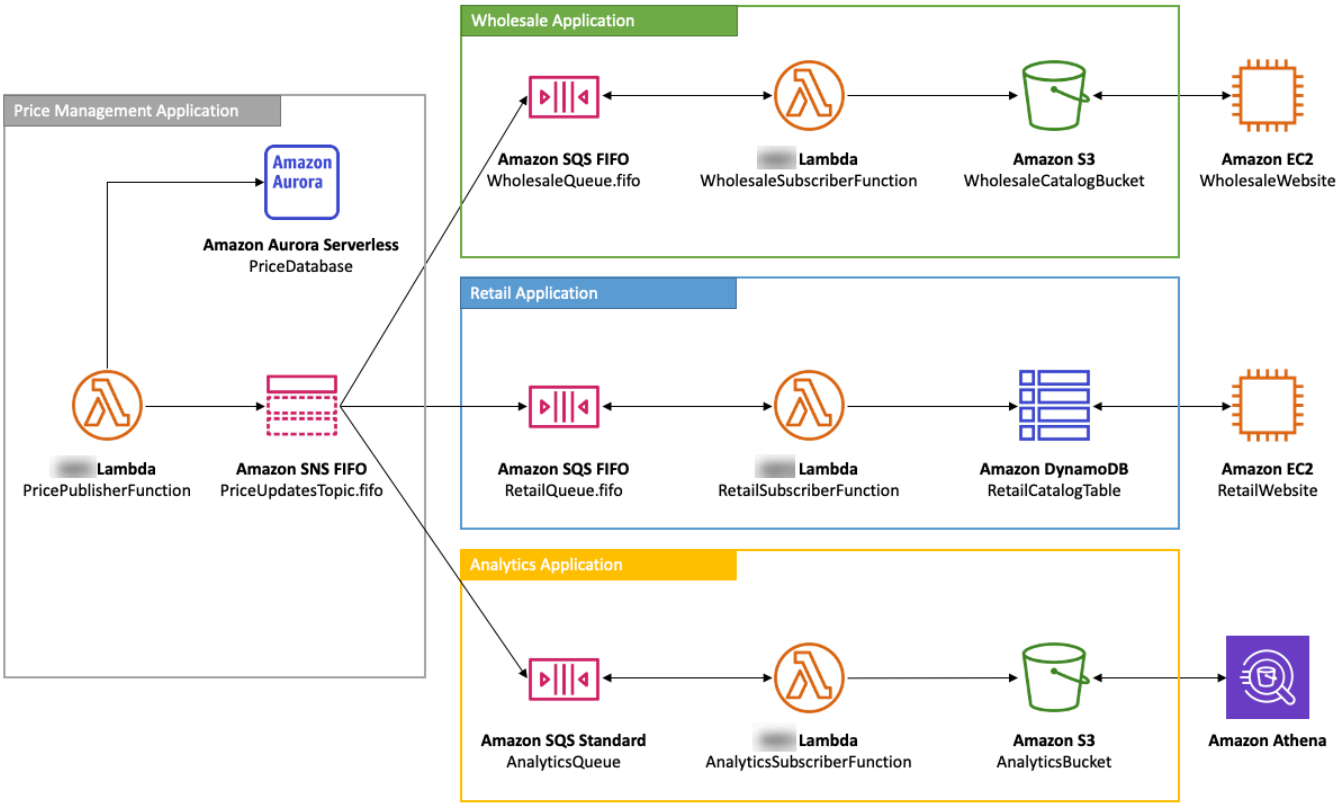
为任何已订阅的 Amazon SQS FIFO 队列启用高吞吐量模式

在启用高吞吐量的情况下发布到您的 Amazon SNS FIFO 主题并且订阅了一个或多个 Amazon SQS FIFO 队列时，建议您在亚马逊 SQS FIFO 队列上启用高吞吐量，以使您的 Amazon SNS FIFO 高吞吐量主题能够顺利交付。有关更多信息，请参阅《Amazon Simple Queue Service 开发人员指南》中的 FIFO 队列的高吞吐量。

Amazon SNS FIFO 主题示例使用案例

以下示例介绍了一个由汽车零部件制造商使用 Amazon SNS FIFO 主题和 Amazon SQS 队列构建的电子商务平台。该平台包含四个无服务器应用程序：

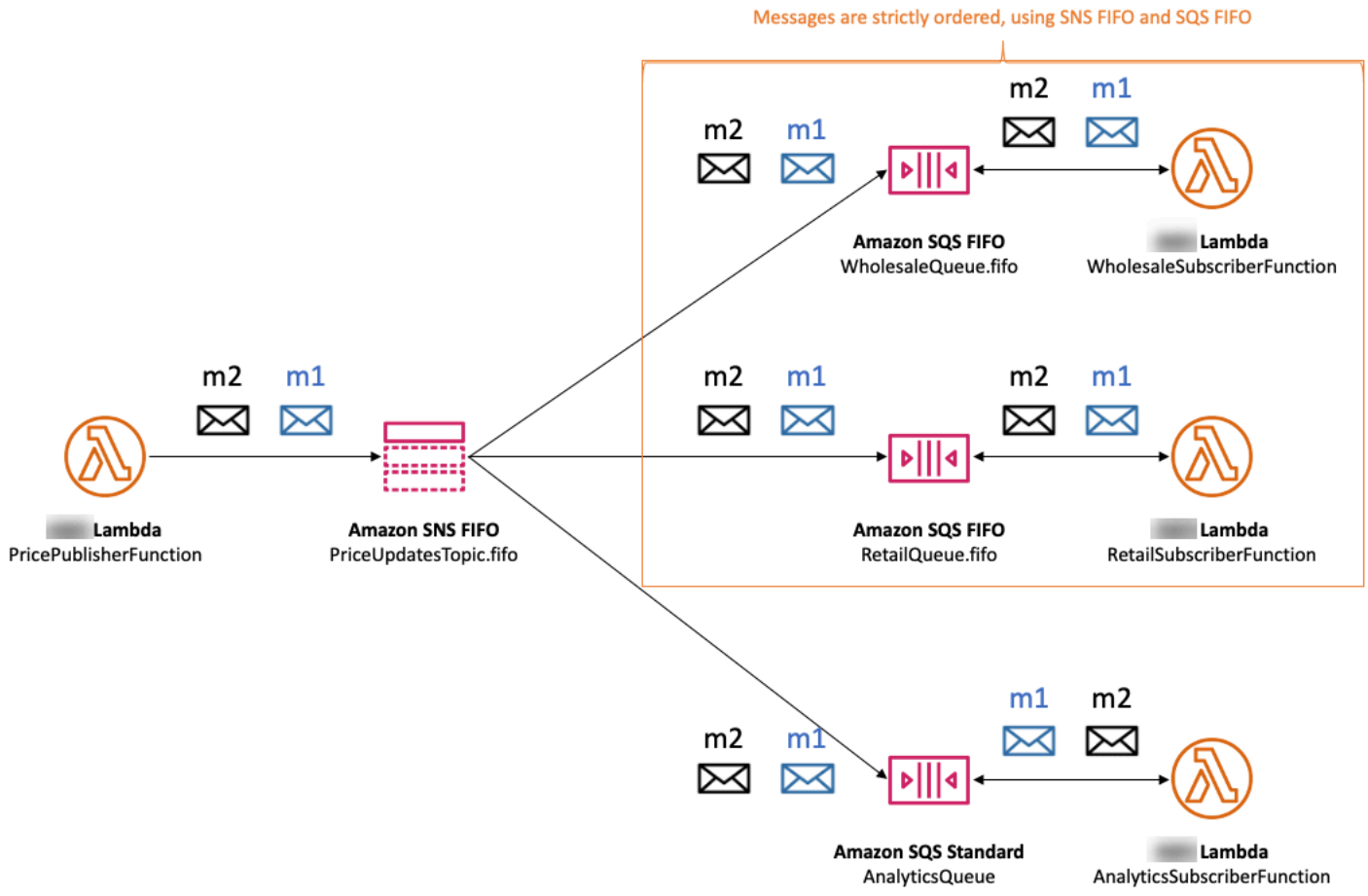
- 库存经理使用价格管理应用程序为每件存货设置价格。在该公司，产品价格可能会因汇率波动、市场需求、销售策略的变化而变化。价格管理应用程序使用 Amazon Lambda 函数，它会在价格变化时将价格更新发布到 Amazon SNS FIFO 主题。
- 批发应用程序为汽车车身修理厂和汽车制造商可以在其中批量购买公司汽车零部件的网站提供后端服务。为了获取价格变化通知，批发应用程序为其 Amazon SQS FIFO 队列订阅价格管理应用程序的 Amazon SNS FIFO 主题。
- 零售应用程序为另一个网站提供后端，车主和汽车改装爱好者可以通过该网站为他们的车辆购买单独的汽车零部件。为了获取价格变化通知，零售应用程序也会为其 Amazon SQS FIFO 队列订阅价格管理应用程序的 Amazon SNS FIFO 主题。
- 一种分析应用程序，可汇总价格更新并将其存储到 Amazon S3 存储桶中，从而使 Amazon Athena 能够出于商业智能 (BI) 目的查询存储桶。为了获取价格变化通知，分析应用程序为其 Amazon SQS 标准队列订阅价格管理应用程序的 Amazon SNS FIFO 主题。与其他应用程序不同，分析应用程序不需要对价格更新进行严格排序。



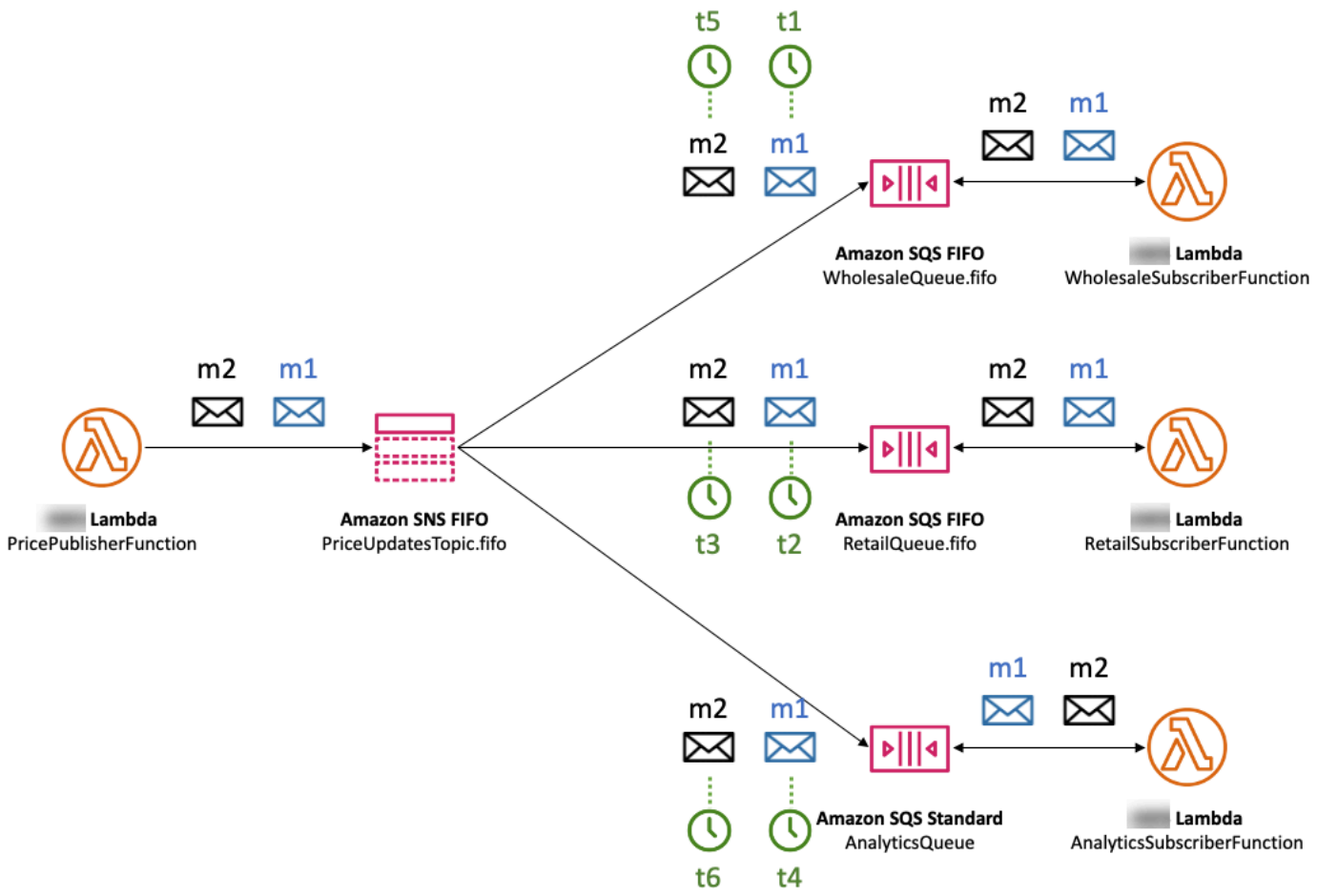
为了使批发和零售应用程序以正确的顺序接收价格更新，价格管理应用程序必须使用严格排序的消息分发系统。使用 Amazon SNS FIFO 主题和 Amazon SQS FIFO 队列可以按顺序处理消息，而不会出现重复。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息排序详细信息。有关实现此使用案例的代码片段，请参阅 适用于 FIFO 主题的 Amazon SNS 代码示例。

FIFO 主题的 Amazon SNS 消息排序详细信息

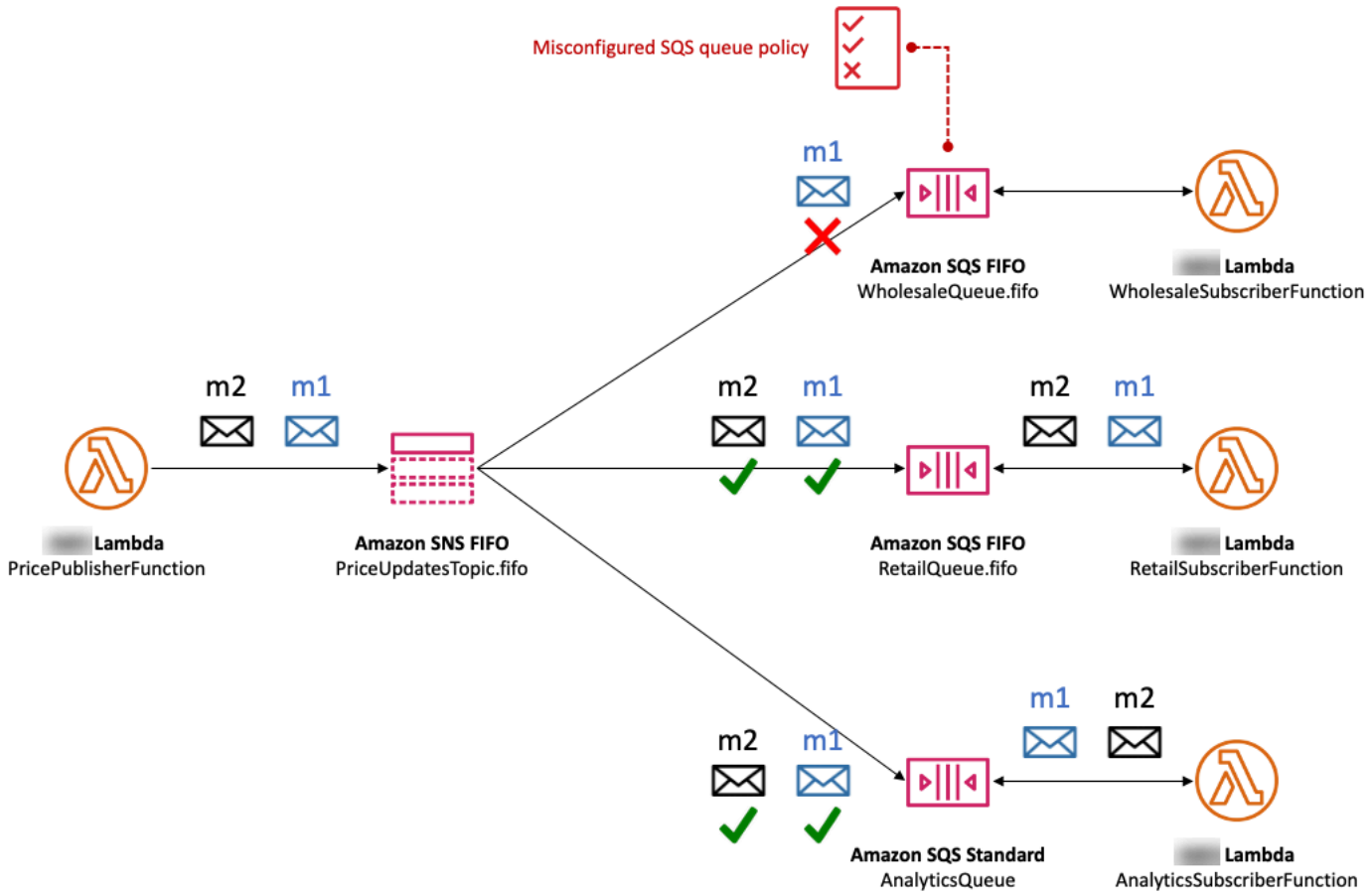
Amazon SNS FIFO 主题始终按照消息发布到主题的确切顺序，向订阅的 Amazon SQS 队列传送消息，并且只传送一次。订阅 Amazon SQS FIFO 队列后，该队列的使用者将按照消息传送到队列的确切顺序接收消息，且不会出现重复消息。但是，订阅了 Amazon SQS 标准队列后，该队列的使用者可能会多次收到乱序消息。这可以进一步将订阅者与发布者分开，从而在消息使用和成本优化方面为订阅者提供更大的灵活性，如基于 Amazon SNS FIFO 主题示例使用案例的下图所示。



请注意，不存在订阅者的隐含排序。以下示例显示消息 m1 首先传输给批发订阅者，然后传输给零售订阅者，再传输给分析订阅者。消息 m2 首先传输给零售订阅者，然后传输给批发订阅者，最后传输给分析订阅者。尽管这两条消息以不同的顺序传递给订阅者，但是每个 Amazon SQS FIFO 订阅者都会保留消息顺序。每个订阅者都与任何其他订阅者隔离感知。

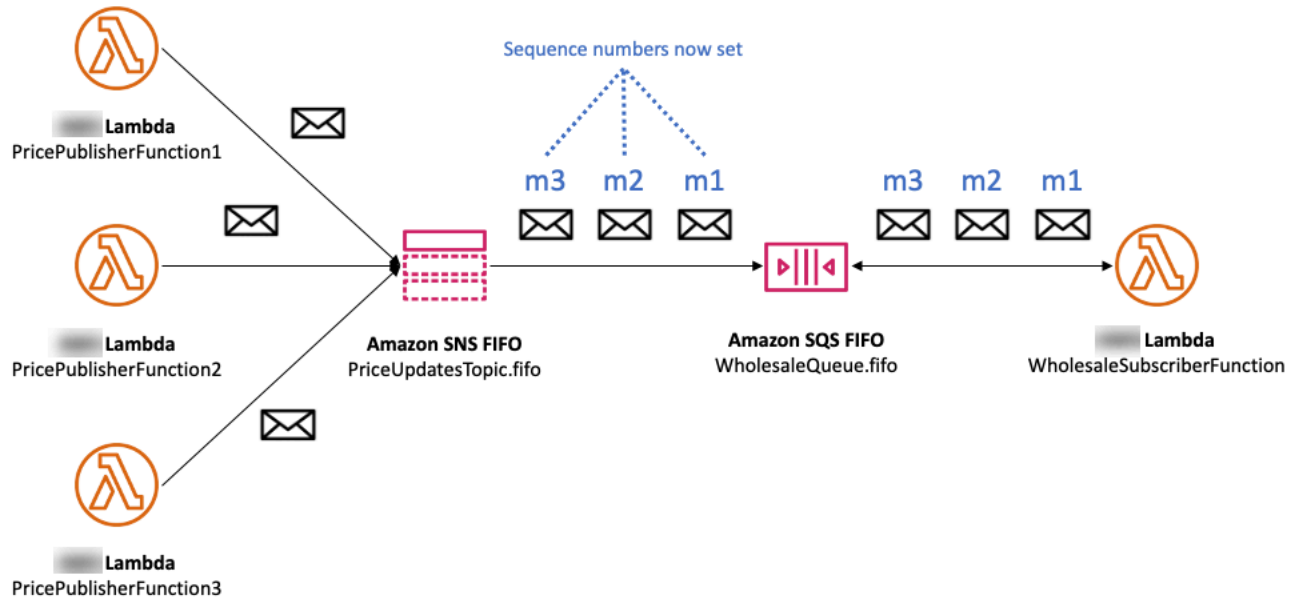


如果 Amazon SQS 队列订阅者无法访问，它可能会不同步。例如，假设批发应用程序队列所有者错误地更改了 Amazon SQS 队列策略，以防止 Amazon SNS 服务委托人将消息传递到队列。在这种情况下，将价格更新传输到批发队列失败，而零售和分析队列的价格更新成功，从而导致订阅者不同步。批发应用程序队列所有者更正队列策略后，Amazon SNS 将继续向订阅队列传递消息。在队列配置不正确时，发布到主题的任何消息都将被删除，除非对应的订阅已配置了死信队列。



您可以让多个应用程序（或同一应用程序中的多个线程）并行向 SNS FIFO 主题发布消息。执行此操作时，您可以有效地将消息排序委托给 Amazon SNS 服务。要确定已建立的消息序列，您可以检查序列号。

序列号是 Amazon SNS 为每条消息分配的大型、非连续的数字。序列号的长度为 128 位，并且每个消息组的序列号会继续增加。序列号作为消息正文的一部分传递给订阅的 Amazon SQS 队列。但是，如果启用原信息传输，则传输到 Amazon SQS 队列的消息不包括序列号或任何其他 Amazon SNS 消息元数据。



Amazon SNS FIFO 主题定义消息组上下文中的排序。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息分组。

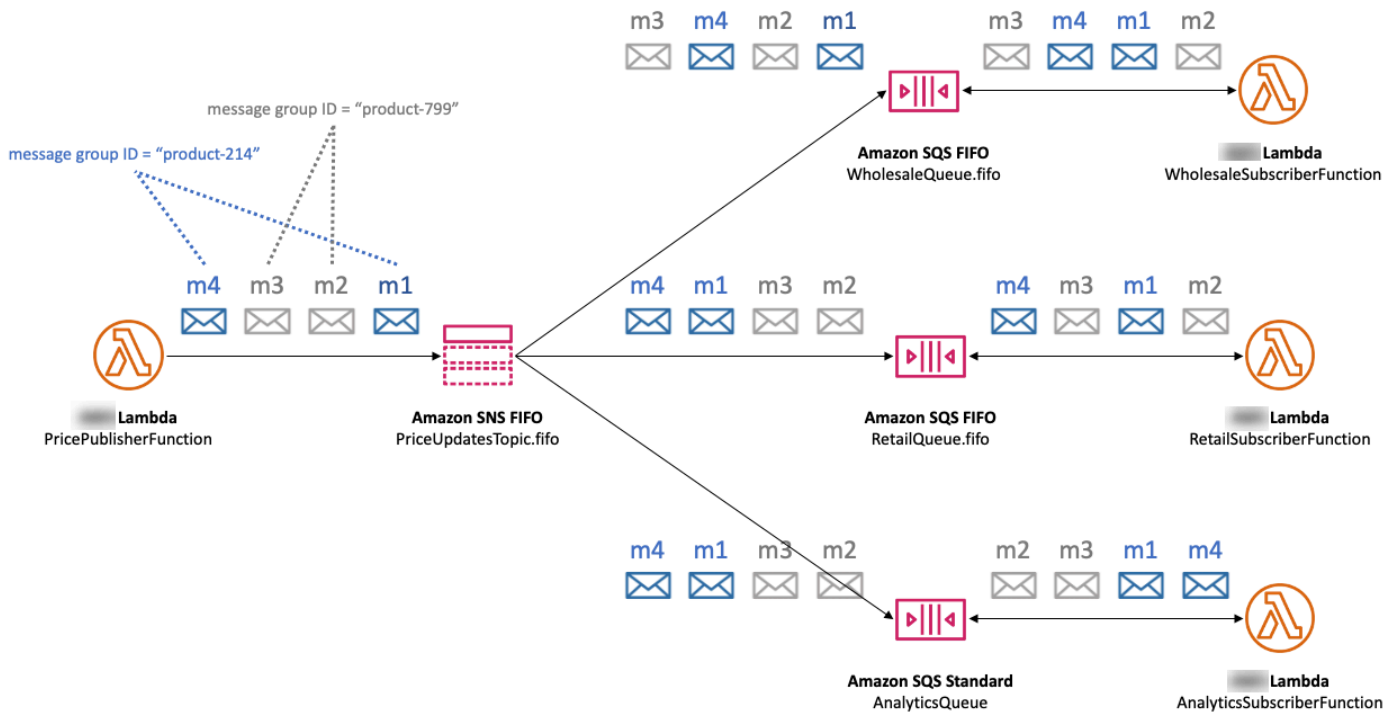
FIFO 主题的 Amazon SNS 消息分组

属于同一组的消息按相对于组的严格顺序逐个处理。

向 Amazon SNS FIFO 主题发布消息时，需设置消息组 ID。组 ID 是指定消息属于特定消息组的强制令牌。SNS FIFO 主题将组 ID 传递给订阅的 Amazon SQS FIFO 队列。SNS FIFO 主题或 SQS FIFO 队列 IDs 中的群组数量没有限制。消息组 ID 不会传递给 Amazon SQS 标准队列。

消息组和订阅之间没有关联性。因此，发布到任何消息组的消息都会传输到所有已订阅队列，但须遵守附加到订阅的任何筛选策略。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息传输 和 FIFO 主题的 Amazon SNS 消息筛选。

在汽车零部件价格管理示例使用案例中，平台中销售的每个产品都有一个专用的消息组。用同一个 Amazon SNS FIFO 主题处理所有价格更新。价格更新的顺序保留在单个汽车零部件产品的上下文中，但不是跨多个产品。下图演示了工作原理。请注意，对于消息组 ID 为 produ ct-214 的产品，消息 m1 在 m 4 之前处理。此顺序将在整个工作流程中保留，这些工作流程使用 Amazon SNS FIFO 和 Amazon SQS FIFO。同样，对于消息组 ID 为 produ ct-799 的产品，消息 m2 在 m3 之前处理。但是，使用 Amazon SQS 标准队列时，将无法保证消息顺序，也不存在消息组。product-214 和 product-799 消息组彼此独立，因此它们的消息排序方式之间没有任何关系。



按消息组分发数据 IDs 以提高性能

为了优化传输吞吐量，Amazon SNS FIFO 主题并行传送来自不同消息组的消息，同时严格维护每个消息组内的消息顺序。每个消息组每秒最多可以传送 300 条消息。因此，要实现单个主题的高吞吐量，请使用大量不同的消息组 IDs。通过利用一组不同的消息组，Amazon SNS FIFO 主题可自动在更多并行分区中分发消息。

> **Note**
>
> Amazon SNS FIFO 主题经过优化，可以跨消息组均匀分发消息 IDs，无论组的数量如何。Amazon 建议您使用大量不同的消息组 IDs 以优化性能。

当以高吞吐量发布到您的 Amazon SNS FIFO 主题并且订阅了一个或多个 Amazon SQS FIFO 队列时，建议您在队列上启用高吞吐量。有关更多信息，请参阅《Amazon Simple Queue Service 开发人员指南》中的 FIFO 队列的高吞吐量。

FIFO 主题的 Amazon SNS 消息传输

Amazon SNS FIFO（先入先出）主题支持向 Amazon SQS 标准队列和 FIFO 队列交付，让客户在集成需要近乎实时数据一致性的分布式应用程序时具有灵活性和控制力。

对于需要保持严格的消息排序或重复数据删除的工作负载，Amazon SNS FIFO 主题与作为传输端点订阅的 Amazon SQS FIFO 队列相结合，可在操作和事件顺序至关重要或不能容忍重复数据时增强应用程序之间的消息传输。

对于允许尽最大努力订购和 at-least-once交付的工作负载，订阅 Amazon SQS 标准队列到 Amaz on SNS FIFO 主题除了可以在不使用 FIFO 的工作负载之间共享队列外，还可以降低成本。

> **Note**
>
> 要将消息从 Amazon SNS FIFO 主题分散到 Amazon Lambda 函数中，需要采取额外的步骤。首先，为 Amazon SQS FIFO 队列或标准队列订阅主题。然后，配置队列以触发函数。有关更多信息，请参阅 Amazon 计算博客上的作为事件源的 SQS FIFO 博文。

SNS FIFO 主题无法将消息传输到客户管理的终端节点，例如电子邮件地址、移动应用程序、用于收发短信 (SMS) 的电话号码或 HTTP(S) 终端节点。这些终端节点类型不能保证保留严格的消息排序。尝试将客户管理的终端节点订阅到 SNS FIFO 主题会导致错误。

SNS FIFO 主题支持与标准主题相同的消息筛选功能。有关更多信息，请参阅 Amazon 计算博客上的 FIFO 主题的 Amazon SNS 消息筛选 和使用 Amazon SNS 消息筛选功能简化您的发布/订阅消息收发。

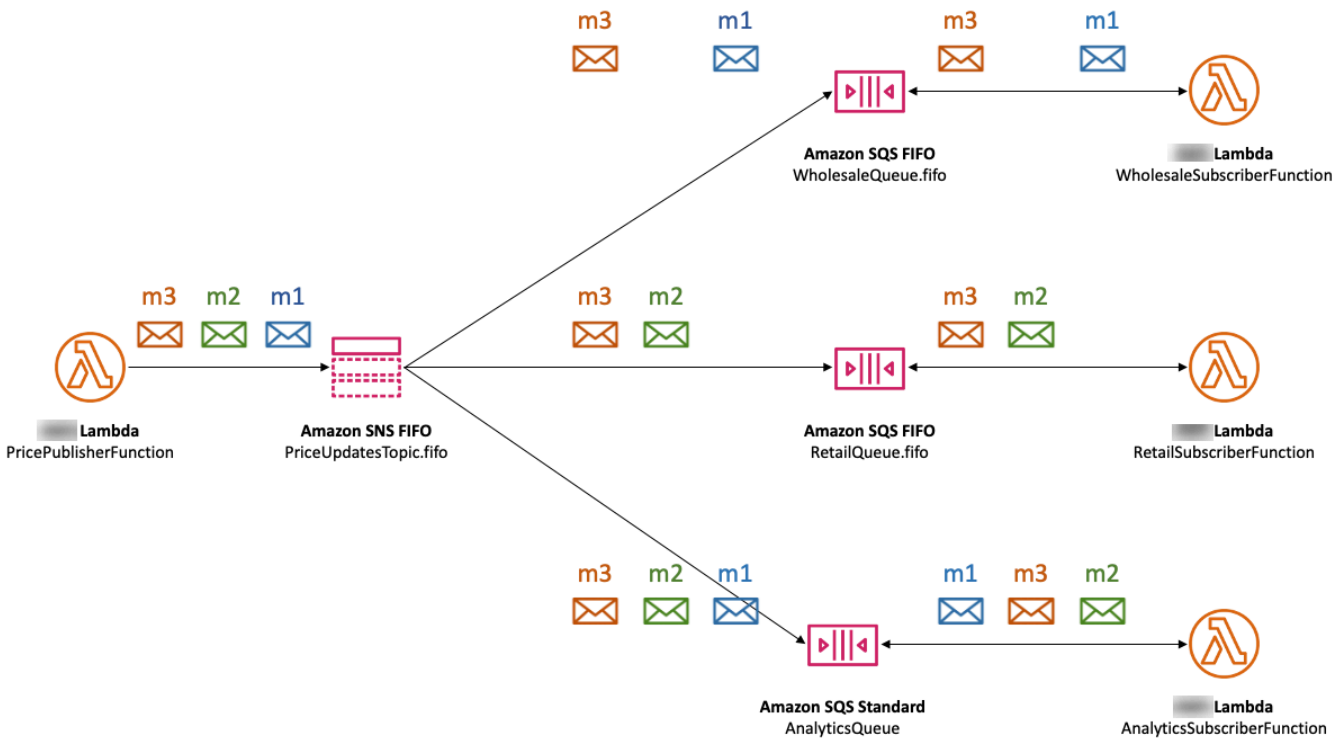
FIFO 主题的 Amazon SNS 消息筛选

Amazon SNS FIFO 主题支持消息筛选。使用消息筛选可通过从发布者系统卸载消息路由逻辑，从订阅者系统卸载消息筛选逻辑来简化您的架构。

当您为 Amazon SQS FIFO 队列或标准队列订阅 SNS FIFO 主题时，您可以使用消息筛选来指定订阅者接收消息的子集，而不是所有消息。每个订阅者都可以将其自己的筛选策略设置为订阅属性。根据筛选策略的范围，筛选策略与入站消息属性或消息正文进行匹配。如果筛选策略匹配，则主题会向订阅者传输消息的副本。如果没有匹配项，则主题不会传输消息的副本。

在汽车零部件价格管理示例使用案例中，假设已设置以下 Amazon SNS 筛选策略且筛选策略范围是 `MessageBody`：

- 对于批发队列，筛选策略 `{"business":["wholesale"]}` 匹配包含名为 `business` 的键且在一组值中具有 `wholesale` 的每条消息。在下图中，消息 m1 中的键之一是值为 `wholesale` 的 `business`。消息 m3 中的键之一是值为 `["wholesale,retail"]` 的 `business`。因此，m1 和 m3 均匹配筛选策略的条件，并且这两条消息都会传输到批发队列中。
- 对于零售队列，筛选策略 `{"business":["retail"]}` 匹配包含名为 `business` 的键且在一组值中具有 `retail` 的每条消息。在图中，消息 m2 中的键之一是值为 `retail` 的 `business`。消息 m3 中的键之一是值为 `["wholesale,retail"]` 的 `business`。因此，m2 和 m3 均匹配筛选策略的条件，并且这两条消息都会传输到零售队列中。
- 对于分析队列，我们希望 Amazon Athena 接收所有记录，因此不应用任何筛选策略。



SNS FIFO 主题支持各种匹配运算符，包括属性字符串值、属性数值和属性键。有关更多信息，请参阅 Amazon SNS 消息筛选。

SNS FIFO 主题不会向订阅的终端节点传输重复的消息。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息重复数据删除。

FIFO 主题的 Amazon SNS 消息重复数据删除

Amazon SNS FIFO 主题和 Amazon SQS FIFO 队列支持消息重复数据删除，只要满足以下条件，即可提供一次性消息传输和处理：

- 订阅的 Amazon SQS FIFO 队列存在，并具有允许 Amazon SNS 服务委托人向队列传输消息的权限。
- Amazon SQS FIFO 队列使用者处理消息，并在可见性超时到期之前将其从队列中删除。
- Amazon SNS 订阅主题没有消息筛选功能。配置邮件筛选时，Amazon SNS FIFO 主题支持 at-most-once传送，因为可以根据您的订阅筛选策略筛选出消息。
- 没有阻止确认邮件传输的网络中断。

> **Note**
>
> 当主题`FifoThroughputScope`属性设置为时，消息重复数据删除适用于整个 Amazon SNS FIFO 主题。`Topic`当主题属性设置`FifoThroughputScope`为时`MessageGroup`，消息重复数据删除将应用于每个单独的消息组。

当您将消息发布到 Amazon SNS FIFO 主题时，该消息必须包含重复数据删除 ID。此 ID 包含在 Amazon SNS FIFO 主题传输给订阅的 Amazon SQS FIFO 队列的消息中。

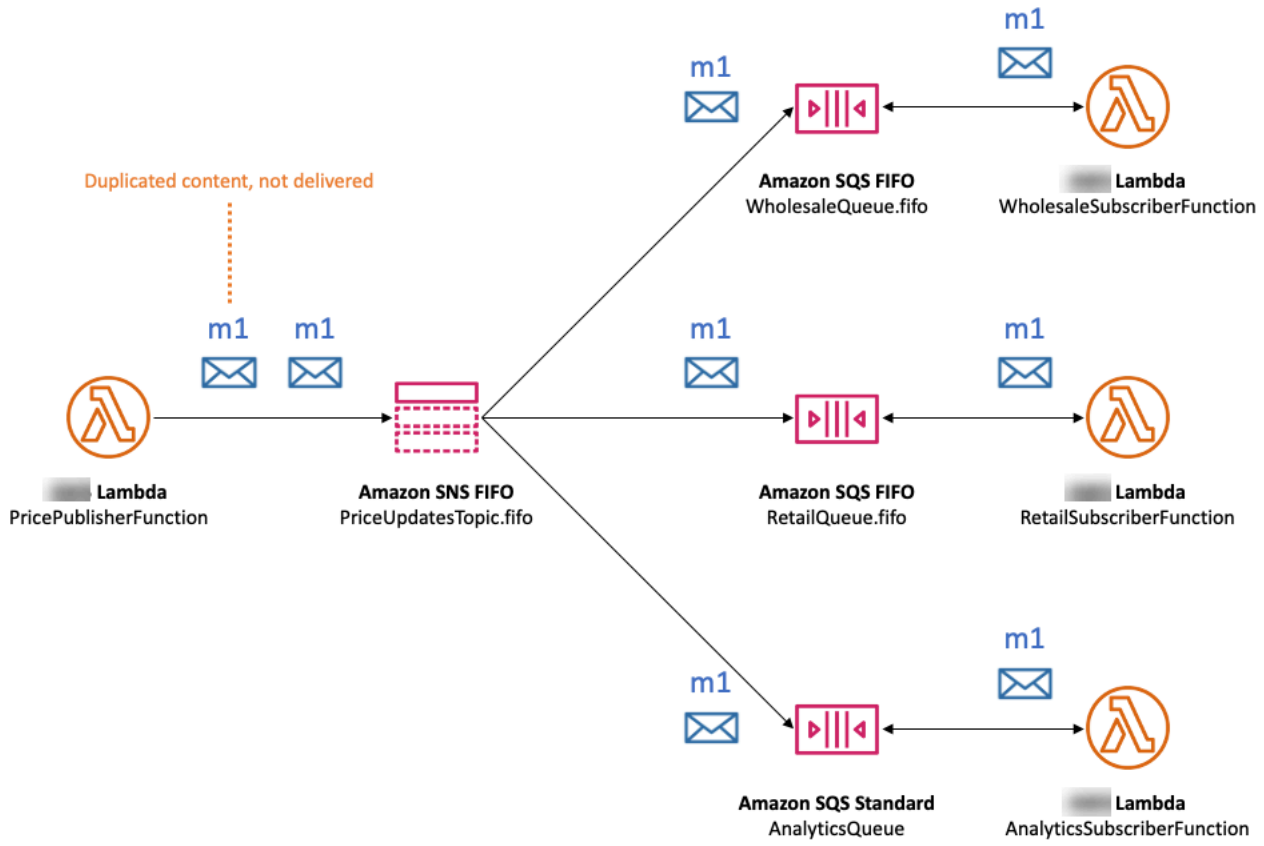
如果成功将具有特定重复数据删除 ID 的消息发布到 Amazon SNS FIFO 主题，则在五分钟重复数据删除间隔内，使用相同重复数据删除 ID 发布的任何消息都将被接受但不会传送。即使在消息传送到订阅的终端节点之后，Amazon SNS FIFO 主题仍在主题属性配置的重复数据删除范围内`FifoThroughputScope`，继续跟踪消息重复数据删除 ID。

如果保证消息正文对于每个已发布的消息都是唯一的，您可以为 Amazon SNS FIFO 主题和订阅的 Amazon SQS FIFO 队列启用基于内容的重复数据删除。Amazon SNS 使用消息正文生成一个唯一的哈希值，以用作每个消息的重复数据删除 ID，因此您无需在发送每条消息时设置重复数据删除 ID。

> **Note**
>
> 消息属性不包括在哈希计算中。

如果为 Amazon SNS FIFO 主题启用基于内容的重复数据删除，并且发布了带有重复数据删除 ID 的消息，则发布的重复数据删除 ID 将覆盖生成的基于内容的重复数据删除 ID。

在汽车零部件价格管理示例使用案例中，公司必须为每次价格更新设置一个通用唯一的重复数据删除 ID。这是因为即使在批发和零售的消息属性不同时，消息正文也可以是相同的。但是，如果公司将业务类型（批发或零售）与产品 ID 和产品价格一起添加到消息正文中，它们可以在 Amazon SNS FIFO 主题和订阅的 Amazon SQS FIFO 队列中启用基于内容的复制。



除了消息排序和重复数据删除外，Amazon SNS FIFO 主题还支持 Amazon KMS 使用密钥进行消息服务器端加密 (SSE)，以及通过 VPC 终端节点进行消息隐私。 Amazon PrivateLink有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息安全性。

FIFO 主题的 Amazon SNS 消息安全性

您可以使用 () 客户主密钥 Amazon Key Management Service (Amazon KMS) 为亚马逊 SNS FIFO 主题和 Amazon SQS FIFO 队列启用加密。CMKs

- 您可以创建新的加密的 FIFO 主题和队列，也可以为现有的 FIFO 主题和队列启用加密。
- 只有邮件正文被加密。消息属性、资源元数据和资源指标仍未加密。

> **Note**
>
> 将加密添加到现有 FIFO 主题或队列中不会加密任何积压的消息，从主题或队列中删除加密会使积压的消息加密。

SNS FIFO 主题会立即解密消息，然后将消息传递到订阅的终端节点。SQS FIFO 队列在将消息返回到使用者应用程序之前解密消息。有关更多信息，请参阅 Amazon 计算博客上的 Amazon SNS 数据加密和使用 Amazon KMS加密发布到 Amazon SNS 的消息。

此外，SNS FIFO 主题和 SQS FIFO 队列支持接口 VPC 终端节点由 Amazon PrivateLink支持的消息隐私。使用接口终端节点，您可以将消息从 Amazon Virtual Private Cloud (Amazon VPC) 子网发送到 FIFO 主题和队列，而无需遍历公有 Internet。这种模式将您的消息传递保存在 Amazon 基础设施和网络中，从而增强了应用程序的整体安全性。使用时 Amazon PrivateLink，您无需设置互联网网关、网络地址转换 (NAT) 或虚拟专用网络 (VPN)。有关更多信息，请参阅 Amazon 安全性博客上的 使用 VPC 端点保护 Amazon SNS 流量 和使用 Amazon PrivateLink保护发布到 Amazon SNS 的消息。

SNS FIFO 主题还支持跨可用区的死信队列和消息存储。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息持久性。

FIFO 主题的 Amazon SNS 消息持久性

Amazon SNS FIFO 主题和 Amazon SQS 队列具有持久性。这两种资源类型都以冗余方式跨多个可用区存储消息，并提供死信队列以处理异常情况。

在 Amazon SNS 中，当 Amazon SNS 主题由于客户端或服务器端错误无法访问订阅的 Amazon SQS 队列时，消息传输将失败。

- 当 Amazon SNS FIFO 主题具有过时的订阅元数据时，会发生客户端错误。两个常见的客户端错误发生于 Amazon SQS 队列所有者执行以下操作之一时：
  - 删除队列。
  - 以防止 Amazon SNS 服务委托人向其传输消息的方式更改队列策略。

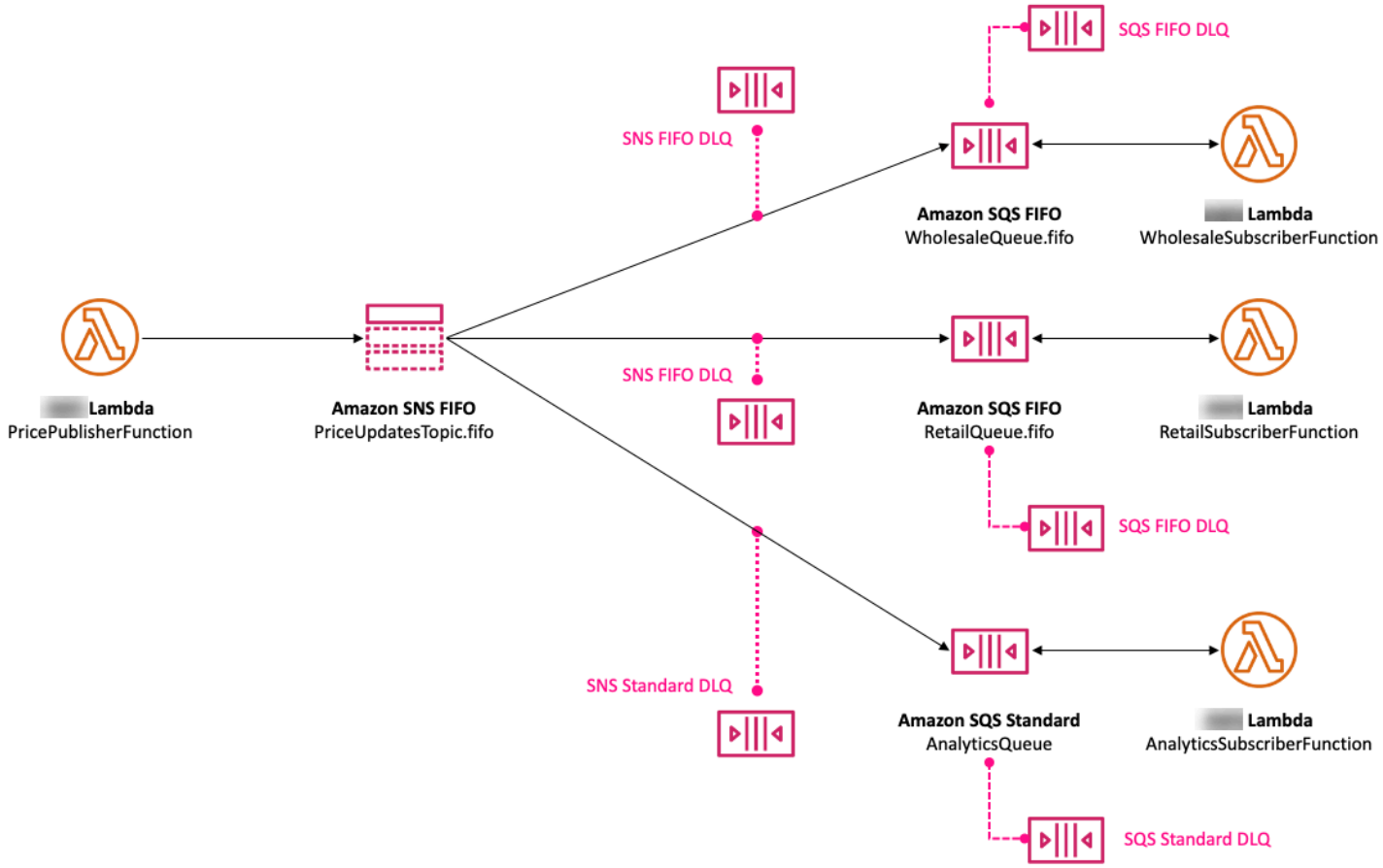
  Amazon SNS 不会重试传输由于客户端错误而失败的消息。
- 在以下情况下可能会发生服务器端错误：
  - Amazon SQS 服务不可用。
  - Amazon SQS 无法处理来自 Amazon SNS 服务的有效请求。

在发生服务器端错误时，Amazon SNS FIFO 主题会最多重试失败的传输 100015 次（时间超过 23 天）。有关更多信息，请参阅 Amazon SNS 消息传输重试。

对于任何类型的错误，Amazon SNS 都可以将消息搁置到 Amazon SQS 死信队列中，以免数据丢失。

在 Amazon SQS 中，当使用者应用程序无法接收消息、处理消息并从队列中删除消息时，消息处理将失败。当接收请求的最大数量失败时，Amazon SQS 可以将消息搁置到死信队列中，以免数据丢失。

在汽车零部件价格管理示例使用案例，公司可以为每个 Amazon SNS FIFO 主题订阅以及每个订阅的 Amazon SQS 队列分配一个 Amazon SQS 死信队列 (DLQ)。这可以保护公司免受任何价格更新损失。



与 Amazon SNS 订阅关联的死信队列必须是与订阅队列类型相同的 Amazon SQS 队列。例如，Amazon SQS FIFO 队列的 Amazon SNS FIFO 订阅必须将 Amazon SQS FIFO 队列用作死信队列。同样，Amazon SQS 标准队列的 Amazon SNS FIFO 订阅必须使用 Amazon SQS 标准队列作为死信队列。有关更多信息，请参阅Amazon SNS 死信队列计算博客上的 “使用 DLQs 适用于 Amazon SNS、Amazon SQS 设计耐用的无服务器应用程序” 一 Amazon Lambda文。Amazon

为了延长持久性以帮助从下游故障中恢复，主题所有者还可以使用 FIFO 主题将消息归档长达 365 天。然后，主题订阅用户可以将这些消息重播到已订阅的端点，以恢复因下游应用程序故障而丢失的消息，或者复制现有应用程序的状态。有关更多信息，请参阅FIFO 主题的 Amazon SNS 消息归档与重播功能。

FIFO 主题的 Amazon SNS 消息归档与重播功能

什么是消息归档与重播功能？

Amazon SNS 提供无代码消息归档与重播功能，专门针对 FIFO（先进先出）主题设计。此功能允许主题所有者将消息直接存储在主题归档中长达 365 天，并在需要时重播给订阅用户。消息归档与重播功能对于恢复丢失的消息以及通过复制状态跨区域或系统同步应用程序至关重要。

可通过 Amazon API Amazon CloudFormation、SDK 和访问此功能 Amazon Web Services Management Console。

关键用例

- 消息恢复 - 通过将消息重播到订阅用户的端点，恢复因下游应用程序故障而丢失的消息。
- 状态复制 - 通过重播从特定时间戳开始的消息，在新环境中复制现有系统的状态。
- 错误更正 - 在中断期间重新发送错过的消息，以确保正确处理所有事件。

消息归档与重播功能的组成部分

管理 Amazon SNS FIFO 主题的消息存档和重播，包括设置保留期、使用监控存档消息 CloudWatch、通过订阅属性启动重播以及了解修改和启动重播所需的权限。

消息归档

- 主题所有者启用归档功能并设置消息保留期，最长可达 365 天。有关更多信息，请参阅适用于 FIFO 主题所有者的 Amazon SNS 消息归档
- CloudWatch 指标有助于监控存档的邮件。

消息重播

- 订阅用户启动重播，选择将消息重新处理到订阅端点的时间窗口。有关更多信息，请参阅FIFO 主题订阅用户的 Amazon SNS 消息重播。
- 您可以使用 `ReplayPolicy` 功能通过订阅属性管理重播。

相关权限

- **SetSubscriptionAttributes** – 需要此权限才能使用订阅上的 ReplayPolicy 属性配置或修改重播设置。
- **Subscribe** – 需要此权限才能附加新的订阅并启动重播。
- **GetTopicAttributes** – 允许查看主题的属性，但重播启动主要围绕订阅管理展开。

适用于 FIFO 主题所有者的 Amazon SNS 消息归档

通过消息归档，您可以归档发布到您的主题的所有消息的单个副本。您可以通过对主题启用消息归档策略将已发布的消息存储在主题中，该策略将为链接到该主题的所有订阅启用消息归档。消息可以归档至少一天，最多 365 天。

设置归档策略时需支付额外费用。有关定价信息，请参阅 Amazon SNS 定价。

使用创建邮件存档策略 Amazon Web Services Management Console

使用此选项，通过 Amazon Web Services Management Console创建新的消息归档策略。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题或创建一个新主题。要了解有关创建主题的更多信息，请参阅创建 Amazon SNS 主题。

   > **Note**
   >
   > Amazon SNS 消息存档和重播仅适用于 application-to-application (A2A) FIFO 主题。

3. 在编辑主题页面上，展开归档策略部分。
4. 启用归档策略特征，然后输入要在主题中归档消息的天数。
5. 选择 Save changes（保存更改）。

查看、编辑和停用消息归档主题策略

- 在主题详细信息页面上，保留策略显示归档策略的状态，包括设置该策略对应的天数。选择归档策略选项卡以查看以下消息归档详细信息：
  - 状态 - 应用归档策略后，归档与重播功能状态显示为活动。当归档策略设置为空的 JSON 对象时，归档与重播功能状态显示为不活动。

  - 消息保留期 - 指定的消息保留天数。
  - 归档开始日期 - 订阅用户可以重播消息的起始日期。
  - JSON 预览 - 归档策略的 JSON 预览。
- （可选）要编辑归档策略，请转到主题摘要页面并选择编辑。
- （可选）要停用归档策略，请转至主题摘要页面并选择编辑。停用归档策略并选择保存更改。
- （可选）要使用归档策略删除主题，必须先按照前面所述停用归档策略。

> **Important**
>
> 为避免意外删除消息，您不能在具有有效消息归档策略的情况下删除主题。必须先停用主题的消息归档策略，然后才能删除该主题。当您停用消息归档策略时，Amazon SNS 会删除所有已归档的消息。删除主题时，订阅将被删除，并且任何传输中的消息都可能无法传送。

使用 API 创建消息归档策略

要使用 API 创建消息归档策略，您需要将属性 `ArchivePolicy` 添加到主题中。您可以使用 API 操作 `CreateTopic` 和 `SetTopicAttributes` 设置 `ArchivePolicy`。`ArchivePolicy` 只有一个值 `MessageRetentionPeriod`，它表示 Amazon SNS 保留消息的天数。要为主题激活消息归档，请将 `MessageRetentionPeriod` 设置为大于零的整数值。例如，要将归档中的消息保留 30 天，请将 `ArchivePolicy` 设置为：

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

要对主题禁用消息归档并清除归档，请取消设置 `ArchivePolicy`，如下所示：

```
{}
```

使用 SDK 创建消息归档策略

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》config和《工具参考指南》中的共享 Amazon SDKs 和*credentials*文件。

以下代码示例展示如何为 Amazon SNS 主题设置 `ArchivePolicy`，以便将发布到该主题的所有消息保留 30 天。

```
// 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);
```

使用创建邮件存档策略 Amazon CloudFormation

要使用创建存档策略， Amazon CloudFormation 请参阅Amazon CloudFormation 用户指南*AWS::SNS::Topic*中的。

授予对加密归档的访问权限

必须先完成以下步骤，然后订阅用户才能开始重播来自加密主题的消息。由于过去的消息会被重播，因此需要为 Amazon SNS 预调配对 KMS 密钥的 `Decrypt` 访问权限，此密钥用于加密归档中的消息。

1. 当您使用 KMS 密钥加密消息并将其存储在主题中时，必须授予 Amazon SNS 通过密钥策略对这些消息解密的能力。有关更多信息，请参阅向 Amazon SNS 授予解密权限。
2. Amazon KMS 为亚马逊 SNS 启用。有关更多信息，请参阅配置 Amazon KMS 权限。

> **Important**
>
> 向 KMS 密钥策略添加新部分时，不要更改策略中任何已存在的部分。如果对主题启用了加密但禁用或删除了 KMS 密钥，或未针对 Amazon SNS 正确地配置 KMS 密钥策略，则 Amazon SNS 无法向您的订阅用户重播消息。

向 Amazon SNS 授予解密权限

要让 Amazon SNS 访问您主题的归档中的加密消息并将其重播到已订阅的端点，您必须启用 Amazon SNS 服务原则来解密这些消息。

以下是允许 Amazon SNS 服务主体在重播主题内的历史消息时解密存储的消息所需的示例策略。

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

使用 Amazon 监控邮件存档指标 CloudWatch

您可以使用以下指标在 Amaz CloudWatch on 上监控存档的邮件。为了收到工作负载异常的通知并帮助避免影响，您可以根据这些指标配置 Amazon CloudWatch 警报。有关更多详细信息，请参阅 Amazon SNS 中的日志记录和监控。

指标	描述
ApproximateNumberOfMessagesArchived	以 60 分钟的分辨率向主题所有者提供在主题归档中归档的消息总数。
ApproximateNumberOfBytesArchived	以 60 分钟的分辨率向主题所有者提供对主题归档中的所有消息归档的总字节数。
NumberOfMessagesArchiveProcessing	以 1 分钟的分辨率向主题所有者提供在间隔期间保存到主题存档的消息数。
NumberOfBytesArchiveProcessing	以 1 分钟的分辨率向主题所有者提供在间隔期间保存到主题存档的总字节数。

GetTopicAttributes API 有一个 BeginningArchiveTime 属性，它表示订阅用户可以开始重播的最早时间戳。以下是此 API 操作的示例响应：

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

FIFO 主题订阅用户的 Amazon SNS 消息重播

Amazon SNS 回放允许主题订阅者从主题数据存储中检索存档的消息并将其重新传送到已订阅的终端节点。

- 订阅创建后可以立即重播消息。
- 重播的留言保留了与原始留言相同的内容MessageId、和Timestamp。
- 该消息包含一个Replayed属性，用于指示这是一条重播的消息。
- 要仅重播特定消息，请对您的订阅应用过滤策略。

有关筛选消息的更多信息，请参阅筛选重播的消息。

使用创建消息重播策略 Amazon Web Services Management Console

使用此选项，通过 Amazon Web Services Management Console创建新的重播策略。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题订阅或创建一个新的主题订阅。要了解有关创建订阅的更多信息，请参阅在 Amazon SNS 主题中创建订阅。
3. 要启动消息重播，请转到重播下拉列表并选择开始重播。
4. 从重播时间范围模式中，进行以下选择：
   a. 选择重播开始日期和时间-选择要开始重播存档邮件的日期（YYYY/MM/DD格式）和时间（24 小时 hh: mm: ss 格式）。开始时间应晚于近似归档时间的开始时间。
   b. （可选）选择重播结束日期和时间-选择要停止重播存档邮件的日期（YYYY/MM/DD格式）和时间（24 小时 hh: mm: ss 格式）。

c. 选择开始重播。

5. （可选）要停止消息重播，请转到订阅详细信息页面，然后从重播下拉列表中选择停止重播。
6. （可选）要使用监控此工作流程中的邮件重播指标 CloudWatch，请参阅使用 Amazon 监控消息重播指标 CloudWatch。

查看和编辑消息重播策略

您可以从订阅详细信息页面执行以下操作：

- 要查看消息重播状态，重播状态字段将显示以下值：
  - 已完成 - 重播已成功重新传送所有消息，现在正在传送新发布的消息。
  - 进行中 – 重播当前正在重播所选消息。
  - 失败 – 重播无法完成。
  - 待处理 - 重播启动时的默认状态。
- （可选）要修改消息重播策略，请转到订阅详细信息页面，然后从重播下拉列表中选择开始重播。开始某个重播将取代现有的重播。

使用 API 向订阅添加重播策略

要重播归档的消息，请使用属性 `ReplayPolicy`。`ReplayPolicy` 可以与 `Subscribe` 和 `SetSubscriptionAttributes` API 操作一起使用。此策略包含以下值：

- **`StartingPoint`**（必需）- 表示从何处开始重播消息。
- **`EndingPoint`**（可选）- 表示何时停止重播消息。如果省略 `EndingPoint`，则重播将继续，直到赶上当前时间。
- **`PointType`**（必需）- 设置起点和终点的类型。目前，`PointType` 唯一支持的值是 `Timestamp`。

例如，要从下游故障中恢复并重新发送 2023 年 10 月 1 日的两小时时段内的所有消息，请使用 `SetSubscriptionAttributes` API 操作设置 `ReplayPolicy`，如下所示：

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

要重播截至 2023 年 10 月 1 日发送到该主题的所有消息，并继续接收与您的主题有关的所有新发布的消息，请使用 `SetSubscriptionAttributes` API 操作对您的订阅设置 `ReplayPolicy`，如下所示：

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

为了验证消息是否已重播，将在每条重播的消息中添加布尔属性 `Replayed`。

使用 SDK 向订阅添加重播策略

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》config和《工具参考指南》中的共享 Amazon SDKs 和*credentials*文件。

以下代码示例显示了如何对订阅设置 `ReplayPolicy`，以便从 Amazon SNS FIFO 主题的归档中重新传送 2023 年 10 月 1 日 2 小时时段内的消息。

```
// 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);
```

理解 EndingPoint

当您向 Amazon SNS 订阅应用 `ReplayPolicy` 时，`EndingPoint` 值是可选的。如果未提供 `EndingPoint`，则重播将从指定的 `StartingPoint` 开始并持续到当前时间，包括处理任何新发布的消息。一旦赶上，订阅将作为常规订阅运行，在新消息发布后接收这些消息。

如果指定了 `EndingPoint`，则该服务将从 `StartingPoint` 一直到 `EndingPoint` 重播消息，然后停止。此操作实际上会暂停订阅。订阅暂停期间，新发布的消息将不会传送到已订阅的端点。

要恢复消息传送，请在不提供 `EndingPoint` 的情况下应用新的 `ReplayPolicy`，并将 `StartingPoint` 设置为继续接收消息的所需时间点。例如，要从之前的重播结束的地方恢复订阅，请将新的 `StartingPoint` 设置为先前提供的 `EndingPoint`。

筛选重播的消息

Amazon SNS 消息筛选可让您控制 Amazon SNS 向您的订阅用户端点重播的消息。当消息筛选和消息归档都已启用时，Amazon SNS 会首先从主题的数据存储中检索消息，然后根据订阅的 `FilterPolicy` 应用消息。当存在匹配项时，消息将传送到订阅的端点，否则消息将被筛选掉。有关更多信息，请参阅 Amazon SNS 订阅筛选策略。

使用 Amazon 监控消息重播指标 CloudWatch

您可以使用以下指标在 Amaz CloudWatch on 上监控重播消息。为了收到工作负载异常的通知并帮助避免影响，您可以根据这些指标配置 Amazon CloudWatch 警报。有关更多详细信息，请参阅 Amazon SNS 中的日志记录和监控。

指标	描述
NumberOfReplayedNotificationsDelivered	以 1 分钟的分辨率向订阅用户提供主题归档中重播的消息总数。
NumberOfReplayedNotificationsFailed	以 1 分钟的分辨率向订阅用户提供主题归档中未能传送的已重播消息的总数。

适用于 FIFO 主题的 Amazon SNS 代码示例

使用以下代码示例将汽车零部件价格管理示例用例与亚马逊 SNS FIFO 主题以及亚马逊 SQS FIFO 队列或标准队列集成。

使用 Amazon 软件开发工具包

使用 Amazon 软件开发工具包，您可以通过将其`FifoTopic`属性设置为来创建 Amazon SNS FIFO 主题。**`true`**您可以通过将 Amazon SQS FIFO 队列的 `FifoQueue` 属性设置为 **`true`** 来创建该队列。此外，您必须将 **`.fifo`** 后缀添加到每个 FIFO 资源的名称。创建 FIFO 主题或队列后，无法将其转换为标准主题或队列。

以下代码示例创建这些 FIFO 和标准队列资源：

- 分发价格更新的 Amazon SNS FIFO 主题
- 为批发和零售应用程序提供这些更新的 Amazon SQS FIFO 队列
- 用于存储记录的分析应用程序的 Amazon SQS 标准队列，可以查询这些记录以获取商业智能 (BI)
- 将三个队列连接到主题的 Amazon SNS FIFO 订阅

本示例将设置订阅中的筛选条件策略。如果通过向主题发布消息来测试示例，请确保您发布的是带 `business` 属性的消息。为属性值指定 `retail` 或 `wholesale`。否则，消息将被筛选掉，且不会传递到订阅的队列中。有关更多信息，请参阅 FIFO 主题的 Amazon SNS 消息筛选。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个 Amazon SQS FIFO 队列和一个标准队列。
- 将队列订阅到主题，发布一条消息到主题。

该测试验证每个队列是否收到消息。完整的示例还显示了添加访问策略，并在最后删除了资源。

```
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);
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。
  - CreateTopic
  - 发布
  - Subscribe

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 Amazon SNS FIFO 主题，将 Amazon SQS FIFO 队列和标准队列订阅到主题，并发布一条消息到主题。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateTopic
  - 发布
  - Subscribe

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 FIFO 主题并为此订阅 Amazon SQS FIFO 队列，然后向 Amazon SNS 主题发布消息。

```
    " 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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的以下主题。

- CreateTopic
- 发布
- Subscribe

接收来自 FIFO 订阅的消息

现在，您可以在三个订阅的应用程序中接收价格更新。如中所示the section called “FIFO 主题使用案例”，每个使用者应用程序的入口点都是 Amazon SQS 队列，其相应 Amazon Lambda 函数可以自动轮询该队列。当 Amazon SQS 队列是 Lambda 函数的事件源时，Lambda 会根据需要扩展其轮询器队列，以高效地使用消息。

有关更多信息，请参阅Amazon Lambda 开发者指南中的Amazon Lambda 与 Amazon SQS 配合使用。有关编写自己的队列轮询器的信息，请参阅《亚马逊简单队列服务开发者指南》ReceiveMessage和《亚马逊简单队列服务 API 参考》中的 Amazon SQS 标准和 FIFO 队列建议。

使用 Amazon CloudFormation

Amazon CloudFormation 允许您使用模板文件将 Amazon 资源集合一起创建和配置为一个单元。本部分提供的模板示例，用于创建以下内容：

- 分发价格更新的 Amazon SNS FIFO 主题
- 为批发和零售应用程序提供这些更新的 Amazon SQS FIFO 队列
- 用于存储记录的分析应用程序的 Amazon SQS 标准队列，可以查询这些记录以获取商业智能 (BI)
- 将三个队列连接到主题的 Amazon SNS FIFO 订阅
- 指定订阅者应用程序的筛选策略只接收他们需要的价格更新

> **Note**
>
> 如果通过向主题发布消息来测试此代码示例，请确保您发布的是带 `business` 属性的消息。为属性值指定 `retail` 或 `wholesale`。否则，消息将被筛选掉，且不会传递到订阅的队列中。

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

```
}
```

有关使用 Amazon CloudFormation 模板部署 Amazon 资源的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 “入门”。

Amazon SNS 消息筛选

默认情况下，Amazon SNS 主题订阅者会收到发布到该主题的每条消息。要仅接收一部分消息，订阅者必须将筛选策略 分配给主题订阅。

筛选策略是一个 JSON 对象，包含定义订阅者接收的消息的属性。根据您为订阅设置的筛选策略范围，Amazon SNS 支持对消息属性或消息正文进行操作的策略。消息正文的筛选策略假设消息有效负载是格式正确的 JSON 对象。

如果某个订阅没有筛选策略，则订阅者将接收发布到其主题的每条消息。当您将一条消息发布到具有筛选策略的某个主题时，Amazon SNS 会为该主题的每个订阅，将消息属性或消息正文与筛选策略中的操作进行比较。如果所有消息属性或消息正文属性满足筛选策略中指定的条件，Amazon SNS 会将消息发送给订阅用户。否则，Amazon SNS 不会将消息发送给该订阅者。

有关更多信息，请参阅筛选发布到主题的消息。

Amazon SNS 订阅筛选策略范围

订`FilterPolicyScope`阅属性允许您通过设置以下值之一来定义过滤范围：

- `MessageAttributes`— 将过滤策略应用于邮件属性（默认设置）。
- `MessageBody`— 将过滤策略应用于邮件正文。

> **Note**
>
> 如果没有为现有筛选策略定义筛选策略范围，则范围默认为 `MessageAttributes`。

Amazon SNS 订阅筛选策略

订阅筛选策略允许您指定属性名称并向每个属性名称分配一个值列表。有关更多信息，请参阅 Amazon SNS 消息筛选。

当 Amazon SNS 根据订阅筛选策略评估消息属性或消息正文属性时，它会忽略未在策略中指定的内容。

> **Important**
>
> Amazon 诸如 IAM 和 Amazon SNS 之类的服务使用一种称为最终一致性的分布式计算模型。对订阅筛选器策略的添加或更改最多需要 15 分钟即可完全生效。

在以下条件下，一个订阅接受一条消息：

- 当筛选策略范围设置为 `MessageAttributes` 时，筛选策略中的每个属性名称都与消息属性名称相匹配。对于筛选策略中匹配的每个属性名称，至少有一个属性值与消息属性值相匹配。
- 当筛选策略范围设置为 `MessageBody` 时，筛选策略中的每个属性名称都与消息正文属性名称相匹配。对于筛选策略中匹配的每个属性名称，至少有一个属性值与消息正文属性值相匹配。

Amazon SNS 目前支持以下筛选运算符：

- AND 逻辑
- OR 逻辑
- OR 运算符
- 键匹配
- 数值精确匹配
- 数值 anything-but 匹配
- 数值范围匹配
- 字符串值精确匹配
- 字符串值 anything-but 匹配
- 使用前缀和 anything-but 运算符的字符串匹配
- 字符串值等于-忽略大小写
- 字符串值 IP 地址匹配
- 字符串值前缀匹配
- 字符串值后缀匹配

Amazon SNS 筛选策略示例

下面的示例演示处理客户事务的 Amazon SNS 主题所发送的消息有效负载。

第一个示例包括 MessageAttributes 字段，其中包含描述事务的属性：

- 客户的兴趣
- 存储名称
- 事件状态
- 购买价格 (USD)

由于此消息包含 MessageAttributes 字段，只要在订阅中将 FilterPolicyScope 设置为 MessageAttributes，任何设置了 FilterPolicy 的主题订阅都可以选择性地接受或拒绝消息。有关将属性应用于消息的信息，请参阅 Amazon SNS 消息属性。

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

以下示例显示了 Message 字段中包含的相同属性，也称为消息有效负载或消息正文。只要在订阅中将 MessageBody 设置为 FilterPolicyScope，任何设置了 FilterPolicy 的主题订阅都可以选择性地接受或拒绝消息。

```
{
"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"
}
```

以下筛选策略基于消息的属性名称和值接受或拒绝消息。

接受示例消息的策略

以下订阅筛选策略中的属性与分配给示例消息的属性匹配。请注意，无论设置为 MessageAttributes 还是 MessageBody，相同的筛选策略都适用于 FilterPolicyScope。每个订阅者根据他们从主题收到的消息的构成来选择其筛选范围。

如果此策略中的任一属性与分配给该消息的属性不匹配，则此策略将拒绝该消息。

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

拒绝示例消息的策略

以下订阅筛选策略在其属性与分配给示例消息的属性之间存在多个不匹配项。例如，由于消息属性中不存在 `encrypted` 属性名称，因此该策略属性会导致消息被拒绝，而不管分配给它的值如何。

如果存在任何不匹配项，则策略将拒绝消息。

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

Amazon SNS 中的筛选策略限制

在 Amazon SNS 中设置筛选策略时，需要记住一些重要规则。这些规则有助于确保过滤策略的有效应用，同时保持系统性能和兼容性。

常见策略限制

在 Amazon SNS 中配置筛选策略时，请遵循以下重要规则，确保它们在保持系统性能和兼容性的同时有效运行：

- 字符串匹配-对于筛选策略中的字符串匹配，比较区分大小写。
- 数值匹配-对于数字匹配，该值的范围可以从 -10^9 到 10^9（-10 亿到 10 亿），小数点后有五位精度。
- 筛选策略复杂度-筛选策略中值的总组合不得超过 150。要计算总的组合数，请将每个数组内所含值的个数相乘。
- 限制密钥数量-过滤策略最多可以有五个密钥。

其它注意事项

- 筛选策略的 JSON 可包含：
  - 用引号引起来的字符串

  - 数字
  - 不带引号的关键字 `true`、`false` 和 `null`

- 使用 Amazon SNS API 时，您必须将筛选策略的 JSON 作为有效的 UTF-8 字符串传递。

- 筛选策略的最大大小为 256 KB。
- 默认情况下，每个主题最多可以有 200 个筛选策略，每个 Amazon 账户最多可以有 10,000 个筛选策略。

  此政策限制不会阻止使用 API 创建 Amazon SQS 队列订阅。`Subscribe`但是，当您在 `Subscribe` API 调用（或 `SetSubscriptionAttributes` API 调用）中附加筛选策略时，它将失败。

  要增加此限额，您可以使用 Amazon 服务限额。

基于属性的筛选的策略限制

基于属性的筛选是默认选项。`FilterPolicyScope` 在订阅中设置为 `MessageAttributes`。

- Amazon SNS 不接受基于属性的筛选的嵌套筛选策略。
- Amazon SNS 仅将策略属性与具有以下数据类型的消息属性进行比较：
  - `String`
  - `String.Array`

> **Important**
>
> 在 Amazon SNS 中使用基于属性的筛选时，必须对某些特殊字符进行双重转义，特别是：
>
> - 双引号 (“)
> - 反斜杠 ()
>
> 未能对这些字符进行双重转义将导致过滤策略与已发布消息的属性不匹配，并且通知将无法传送。

其它注意事项

- 不建议在数组中传递对象，因为嵌套可能会产生意想不到的结果，而基于属性的筛选不支持嵌套。对嵌套策略使用基于有效载荷的筛选。

- Number支持数字属性值。
- Amazon SNS 会忽略数据类型为二进制的消息属性。

复杂性策略示例：

在以下策略示例中，第一个密钥有三个匹配运算符，第二个密钥有一个匹配运算符，第三个密钥有两个匹配运算符。

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

总组合是根据筛选策略中每个密钥的匹配运算符数量的乘积计算得出的：

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

基于有效负载的筛选的策略限制

要从基于属性（默认）的筛选切换到基于有效负载的筛选，您必须在订阅中将 FilterPolicyScope 设置为 MessageBody。

- Amazon SNS 接受基于有效负载的筛选的嵌套筛选策略。
- 对于嵌套策略，只有叶子密钥才计入五个密钥的限制。

密钥限制策略示例：

在以下策略示例中：

- 有两个树叶键：key_c和key_e。
- key_c有四个匹配运算符，嵌套级别为三，key_e有三个匹配运算符，嵌套级别为二。

```
{
```

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

总组合的计算方法是匹配运算符数量和筛选策略中每个密钥的嵌套级别的乘积：

```
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
```

AND/OR 逻辑

在筛选策略中使用和/或逻辑来匹配 Amazon SNS 中的消息属性或消息正文属性。这可以实现更精确、更灵活的邮件过滤。

AND 逻辑

您可使用多个属性名称来应用 AND 逻辑。

考虑以下策略：

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

它匹配任何 `customer_interests` 值设置为 `rugby` 且 `price_usd` 值设置为大于 100 的数字的消息属性或消息正文属性。

> **Note**
>
> 您无法将 AND 逻辑应用于同一属性的值。

OR 逻辑

通过将多个值分配给一个属性名称来应用 OR 逻辑。

考虑以下策略：

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

它匹配任何 `customer_interests` 值设置为 `rugby`、`football`、或 `baseball` 的消息属性或消息正文属性。

OR 运算符

您可以使用 `"$or"` 运算符明确定义筛选策略，以表达策略中多个属性之间的 OR 关系。

只有当策略满足以下所有条件时，Amazon SNS 才会识别 `"$or"` 关系。当其中任一条件未满足时，`"$or"` 会被视为常规属性名称，与策略中的任何其他字符串相同。

- 规则中有一个 `"$or"` 字段属性，后跟一个数组，例如，“`$or” : []`。
- `"$or"` 数组中至少有 2 个对象：`"$or": [{}, {}]`。
- `"$or"` 数组中的所有对象都没有属于保留关键字的字段名。

否则，`"$or"` 将被视为普通属性名称，与策略中的其他字符串相同。

由于数字和前缀是保留关键字，因此以下策略不会被解析为 OR 关系。

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

OR 运算符示例

标准 OR：

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

```
    { "namespace": [ "AWS/EC2" ] }
  ]
}
```

此策略的筛选逻辑是：

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

它匹配以下任一组消息属性：

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

或

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

它还匹配以下任一消息正文：

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

或

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

包括 OR 关系在内的策略限制

考虑以下策略：

```
{
    "source": [ "aws.cloudwatch" ],
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
```

```
      {
        "metricType": [ "MetricType" ] ,
        "$or" : [
          { "metricId": [ 1234, 4321 ] },
          { "spaceId": [ 1000, 2000, 3000 ] }
        ]
      }
    ]
  }
```

该策略的逻辑也可以简化为：

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

具有 OR 关系的策略的复杂度计算可以简化为每个 OR 语句的组合复杂度之和。

总组合数的计算方法如下所示：

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

`source` 有一个值，`metricName` 有两个值，`metricType` 有一个值，`metricId` 有两个值，`spaceId` 有三个值。

考虑以下嵌套筛选策略：

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

```
      ]
    }
  }
```

该策略的逻辑可以简化为：

```
("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")
```

组合总数的计算方法与非嵌套策略相同，只是我们需要考虑键的嵌套级别。

总组合数的计算方法如下所示：

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

metricName 有两个值，namespace 有两个值，scope 是具有一个值的两级嵌套键，source 是具有一个值的两级嵌套键，type 是具有一个值的两级嵌套键。

键匹配

在筛选策略中使用exists运算符根据特定属性是否存在来匹配传入的消息。

- exists仅适用于叶节点（结构中的最终属性）。
- 它不适用于嵌套 JSON 结构中的中间节点。

- 使用 "exists": true 匹配包含指定属性的传入消息。键值必须为非空值。

  例如，以下策略属性使用值为 true 的 exists 运算符：

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

  它匹配包含 store 属性键的任何消息属性列表，如下所示：

  ```
  "store": {"Type": "String", "Value": "fans"}
  ```

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

它还匹配以下任一消息正文：

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

但是，它不匹配不含 store 属性键的任何消息属性列表，如下所示：

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

它与以下消息正文也不匹配：

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

- 使用 "exists": false 匹配不 包含指定属性的传入消息。

> **Note**
>
> "exists": false 仅在存在至少一个属性时才匹配。一组空的属性会导致筛选条件不匹配。

例如，以下策略属性使用值为 false 的 exists 运算符：

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

它不 匹配包含 store 属性键的任何消息属性列表，如下所示：

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

它与以下消息正文也不匹配：

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

但是，它匹配不含 store 属性键的任何消息属性列表，如下所示：

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

它还匹配以下消息正文：

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

数值匹配

通过将数值与消息属性值或消息正文属性值进行匹配来筛选消息。在 JSON 策略中，不会用双引号将字符串值引起来。您可以使用以下数值运算进行筛选。

> **Note**
>
> 前缀仅支持字符串 匹配。

精确匹配

当策略属性值包含 numeric 关键字和 = 运算符时，它将匹配具有相同名称和相同数值的任何消息属性值或消息正文属性值。

请考虑以下策略属性：

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

它匹配以下任一消息属性：

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

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

它还匹配以下任一消息正文：

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

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

Anything-but 匹配

当策略属性值包含关键字 `anything-but` 时，它会匹配不 包含任何策略属性值的任何消息属性值或消息正文属性值。

请考虑以下策略属性：

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

它匹配以下任一消息属性：

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

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

它还匹配以下任一消息正文：

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

```
{
```

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

此外，它还匹配以下消息属性（因为它包含的值不是 100 或 500）：

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

它还匹配以下消息正文（因为它包含的值不是 100 或 500）：

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

但是，它不匹配以下消息属性：

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

它与以下消息正文也不匹配：

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

值范围匹配

除了 = 运算符之外，数值策略属性还可以包含以下运算符：<、<=、> 和 >=。

请考虑以下策略属性：

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

它匹配任何具有负数值的消息属性或消息正文属性。

考虑另一个消息属性：

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

它匹配任何具有正数（最大为 150）的消息属性或消息正文属性。

字符串值匹配

通过将字符串值与消息属性值或消息正文属性值进行匹配来筛选消息。在 JSON 策略中，用双引号将字符串值引起来。您可以使用以下字符串操作来匹配消息属性或消息正文属性：

精确匹配

在策略属性值与一个或多个消息属性值匹配时，会进行精确匹配。对于`String.Array`类型属性，出于匹配目的，数组中的每个元素都被视为一个单独的字符串。

请考虑以下策略属性：

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

它匹配以下消息属性：

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

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

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

它还匹配以下消息正文：

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

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

但是，它与以下消息属性不匹配：

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

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

它与以下消息正文也不匹配：

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

Anything-but 匹配

当策略属性值包含关键字 `anything-but` 时，它会匹配不 包含任何策略属性值的任何消息属性值或消息正文值。`anything-but` 可以与 `"exists": false` 组合。对于`String.Array`类型属性，如果策略属性中没有列出任何数组元素，则匹配该属性。

请考虑以下策略属性：

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

它与以下任何消息属性相匹配：

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

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

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

它还匹配以下任一消息正文：

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

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

此外，它还匹配以下消息属性（因为它包含的值不是 `rugby` 或 `tennis`）：

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

它还匹配以下消息正文（因为它包含的值不是 rugby 或 tennis）：

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

但是，它与以下消息属性不匹配：

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

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

它与以下消息正文也不匹配：

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

将前缀与 anything-but 运算符结合使用

要进行字符串匹配，您也可以将前缀与 anything-but 运算符结合使用。例如，以下策略属性拒绝 order- 前缀：

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

它匹配以下任一属性：

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

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

它还匹配以下任一消息正文：

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

```
{
```

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

但是，它不匹配以下消息属性：

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

它与以下消息正文也不匹配：

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

Equals-ignore-case 匹配

当策略属性包含关键字 `equals-ignore-case` 时，它将与任何信息属性或正文属性值进行不区分大小写的匹配。

请考虑以下策略属性：

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

它匹配以下任一消息属性：

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

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

它还匹配以下任一消息正文：

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

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

IP 地址匹配

您可以使用 `cidr` 运算符来检查传入消息是否源自特定 IP 地址或子网。

请考虑以下策略属性：

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

它匹配以下任一消息属性：

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

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

它还匹配以下任一消息正文：

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

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

但是，它不匹配以下消息属性：

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

它与以下消息正文也不匹配：

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

前缀匹配

当策略属性值包含关键字 `prefix` 时，它匹配以指定字符开头的任何消息属性值或正文属性值。

请考虑以下策略属性：

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

它匹配以下任一消息属性：

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

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

它还匹配以下任一消息正文：

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

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

但是，它不匹配以下消息属性：

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

它与以下消息正文也不匹配：

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

后缀匹配

当策略属性值包含关键字 `suffix` 时，它匹配以指定字符结尾的任何消息属性值或正文属性值。

请考虑以下策略属性：

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

它匹配以下任一消息属性：

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

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

它还匹配以下任一消息正文：

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

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

但是，它不匹配以下消息属性：

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

它与以下消息正文也不匹配：

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

在 Amazon SNS 中应用订阅筛选策略

利用 Amazon SNS 中的消息筛选，您可以根据筛选策略有选择地向订阅用户传送消息。这些策略定义了发送到订阅的消息必须满足的条件。虽然原始消息传送是一个可能影响消息处理的选项，但订阅筛选无需该选项即可运行。

您可以使用 Amazon SNS 控制台将筛选策略应用于 Amazon SNS 订阅。或者，要以编程方式应用策略，您可以使用亚马逊 SNS API、 Amazon Command Line Interface Amazon CLI() 或 Amazon 任何支持 Amazon SNS 的软件开发工具包。你也可以使用 Amazon CloudFormation。

实现原始消息传输

原始消息交付可确保消息有效载荷按原样交付给订阅用户，而无需进行任何额外的编码或转换。当订阅用户需要使用原始消息格式进行处理时，这可能很有用。但是，原始消息的传送与订阅筛选的功能没有直接关系。

应用订阅筛选

要将消息筛选应用于订阅，请使用 JSON 语法定义筛选策略。该策略规定了消息必须满足什么条件才能发送到订阅中。筛选可以基于消息属性，例如消息属性、消息结构甚至消息内容。

原始消息传送和订阅筛选之间的关系

虽然启用原始消息传送会影响订阅用户传递和处理消息的方式，但这并不是使用订阅筛选的先决条件。但是，在订阅用户需要原始消息格式而不做任何修改的情况下，启用原始消息传送可能会与订阅筛选一起发挥作用。

有效筛选的注意事项

在实现消息过滤时，请考虑您的应用程序和订阅用户的特定要求。定义与消息传送标准精确匹配的筛选策略，以确保高效、有针对性的消息分发。

> **Important**
>
> Amazon 诸如 IAM 和 Amazon SNS 之类的服务使用一种称为最终一致性的分布式计算模型。对订阅筛选器策略的添加或更改最多需要 15 分钟即可完全生效。

Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择订阅。
3. 选择订阅，然后选择编辑。
4. 在 Edit（编辑）页面上，展开 Subscription filter policy（订阅筛选策略）部分。
5. 在 attribute-based filtering（基于属性的筛选）或 payload-based filtering（基于有效负载的筛选）之间进行选择。
6. 在 JSON editor（JSON 编辑器）字段中，提供筛选策略的 JSON body（JSON 正文）。
7. 选择 Save changes（保存更改）。

   Amazon SNS 将您的筛选策略应用到订阅。

Amazon CLI

要使用 Amazon Command Line Interface (Amazon CLI) 应用筛选策略，请使用set-subscription-attributes命令，如以下示例所示。对于 `--attribute-name` 选项，请指定 `FilterPolicy`。对于 `--attribute-value`，请指定您的 JSON policy（JSON 策略）。

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

要为您的策略提供有效的 JSON，请用双引号将属性名和值括起来。此外，您必须用引号将整个策略参数括起来。要避免转义引号，您可以使用单引号将策略括起来，并使用双引号将 JSON 名称和值括起来，如以上示例中所示。

如果要从基于属性（默认）的邮件筛选切换到基于负载的邮件过滤，也可以使用该set-subscription-attributes命令。对于 `--attribute-name` 选项，请指定 `FilterPolicyScope`。对于 `--attribute-value`，请指定 `MessageBody`。

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

要验证是否已应用您的筛选策略，请使用 `get-subscription-attributes` 命令。终端输出中的属性应显示 `FilterPolicy` 键的筛选策略，如以下示例中所示：

```
$ 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: . . ."
    }
}
```

Amazon SDKs

以下代码示例演示如何使用 `SetSubscriptionAttributes`。

> **Important**
>
> 如果您使用 SDK for Java 2.x 示例，则类 `SNSMessageFilterPolicy` 并非开箱即用。有关如何安装该类的说明，请参阅 GitHub 网站上的示例。

CLI

Amazon CLI

设置订阅属性

以下 `set-subscription-attributes` 示例将 `RawMessageDelivery` 属性设置为 SQS 订阅。

```
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
```

此命令不生成任何输出。

以下 `set-subscription-attributes` 示例将 `FilterPolicy` 属性设置为 SQS 订阅。

```
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\"] }"
```

此命令不生成任何输出。

以下 `set-subscription-attributes` 示例从 SQS 订阅中移除 `FilterPolicy` 属性。

```
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 "{}"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetSubscriptionAttributes中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SetSubscriptionAttributes中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用SetSubscriptionAttributes于 Python 的Amazon SDK (Boto3) API 参考。

Amazon SNS API

要使用 Amazon SNS API 应用筛选策略，需要请求 `SetSubscriptionAttributes` 操作。将 `AttributeName` 参数设置为 `FilterPolicy`，并将 `AttributeValue` 参数设置为您的筛选策略 JSON。

如果要从基于属性（默认）的消息筛选切换到基于有效负载的消息筛选，您也可以使用 `SetSubscriptionAttributes` 操作。将 `AttributeName` 参数设置为 `FilterPolicyScope`，并将 `AttributeValue` 参数设置为 `MessageBody`。

Amazon CloudFormation

要使用应用筛选策略 Amazon CloudFormation，请使用 JSON 或 YAML 模板创建 Amazon CloudFormation 堆栈。有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的`AWS::SNS::Subscription`资源`FilterPolicy`属性和示例 Amazon CloudFormation 模板。

1. 登录 Amazon CloudFormation 控制台。
2. 选择创建堆栈。
3. 在选择模板页面上，依次选择将模板上传到 Amazon S3、您的文件和下一步。

4. 在指定详细信息页面中，执行以下操作：
   a. 对于堆栈名称，键入 `MyFilterPolicyStack`。
   b. 对于 myHttpEndpoint，键入要订阅您的主题的 HTTP 终端节点。

   > **Tip**
   > 如果没有 HTTP 终端节点，请创建一个。

5. 在选项页面上，选择下一步。
6. 在 Review 页面上，选择 Create 。

在 Amazon SNS 中移除订阅筛选策略

要停止筛选已发送到订阅的消息，请使用空白的 JSON 正文覆盖订阅的筛选策略以删除该策略。在删除该策略后，订阅会接受发布到它的每条消息。

使用 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择订阅。
3. 选择订阅，然后选择编辑。
4. 在 “编辑” ***EXAMPLE1-23bc-4567-d890-ef12g3hij456*** 页面上，展开 “订阅筛选策略” 部分。
5. 在 JSON editor (JSON 编辑器) 字段中，为筛选策略提供空的 JSON 正文：`{}`。
6. 选择 Save changes（保存更改）。

   Amazon SNS 将您的筛选策略应用到订阅。

使用 Amazon CLI

要使用删除筛选器策略 Amazon CLI，请使用`set-subscription-attributes`命令并为`--attribute-value`参数提供一个空的 JSON 正文：

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

使用 Amazon SNS API

要使用 Amazon SNS API 删除筛选策略，需要请求 `SetSubscriptionAttributes` 操作。将 `AttributeName` 参数设置为 `FilterPolicy`，然后为 `AttributeValue` 参数提供一个空白的 JSON 正文。

在 Amazon SNS 中的消息数据保护

什么是消息数据保护？

消息数据保护使用数据保护策略来审计、屏蔽、编辑或屏蔽在应用程序或服务之间移动的敏感信息，从而保护发布到您的 Amazon SNS 主题的数据。 Amazon

消息数据保护功能使用数据标识符，扫描传输中数据内包含的个人身份信息（PII）和受保护健康信息（PHI）。您可以选择使用预定义（或 Amazon SNS 托管式）数据标识符（例如，姓名、地址、信用卡号和处方药代码），也可以创建自己的自定义数据标识符，这些标识符特定于您的业务用例。使用扫描的信息，消息数据保护功能提供详细的审计日志，并允许您采取措施来保护该数据。

消息数据保护功能支持以下操作，以帮助保护敏感的客户信息：

- 审计 – 审计多达 99% 的发布到 Amazon SNS 主题的数据。然后，您可以选择将调查结果发送到亚马逊 CloudWatch、亚马逊 S3 或亚马逊 Data Firehose。
- 去身份识别 – 遮蔽或去除敏感数据而不中断消息发布或传输。
- 拒绝-如果有效负载中存在敏感数据，则阻止应用程序和 Amazon 资源之间的数据传输。

> **Note**
>
> Amazon SNS 仅支持针对 Amazon SNS 标准主题的消息数据保护功能。

为什么应该使用消息数据保护？

通过在监管、风险管理和合规计划中引入消息数据保护功能，您可以实施数据保护策略来帮助识别和防止数据泄露。这为您的团队提供了工具来遵守 HIPAA、GDPR、PCI 和 FedRAMP 等隐私法规，从而帮助降低财务、法律和监管风险。它还可以让您的开发人员摆脱为了保护敏感数据而构建和管理自己的工具的相关运营开销。

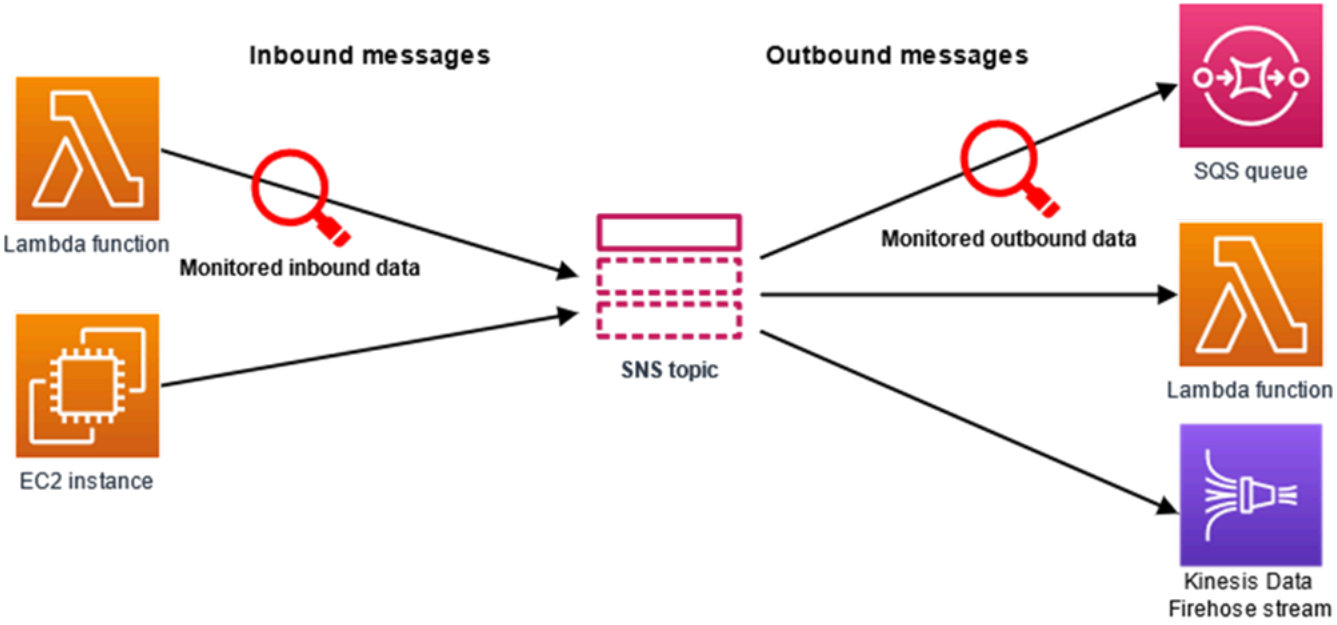
例如，您可以使用消息数据保护功能创建审计策略，确定是否有任何系统无意中发送或接收了敏感数据。如果您的审计结果表明系统将信用卡信息发送到了无需这些信息的系统，您可以使用阻止策略 来阻止传输这些数据。

> **Note**
>
> Amazon SNS 仅支持针对 Amazon SNS 标准主题的消息数据保护功能。

了解 Amazon SNS 数据保护策略

什么是数据保护策略？

Amazon SNS 使用数据保护策略选择要扫描的敏感数据，以及为了保护这些数据不被 Amazon SNS 主题交换而要采取的操作。要选择感兴趣的敏感数据，您可以使用数据标识符。Amazon SNS 消息数据保护功能随后使用机器学习和模式匹配来检测敏感数据。要对找到的数据标识符采取操作，您可以定义审计、去身份识别或拒绝操作。利用这些操作，您可以记录已找到（或未找到）的敏感数据，遮蔽或者去除敏感数据，或者拒绝消息传送。

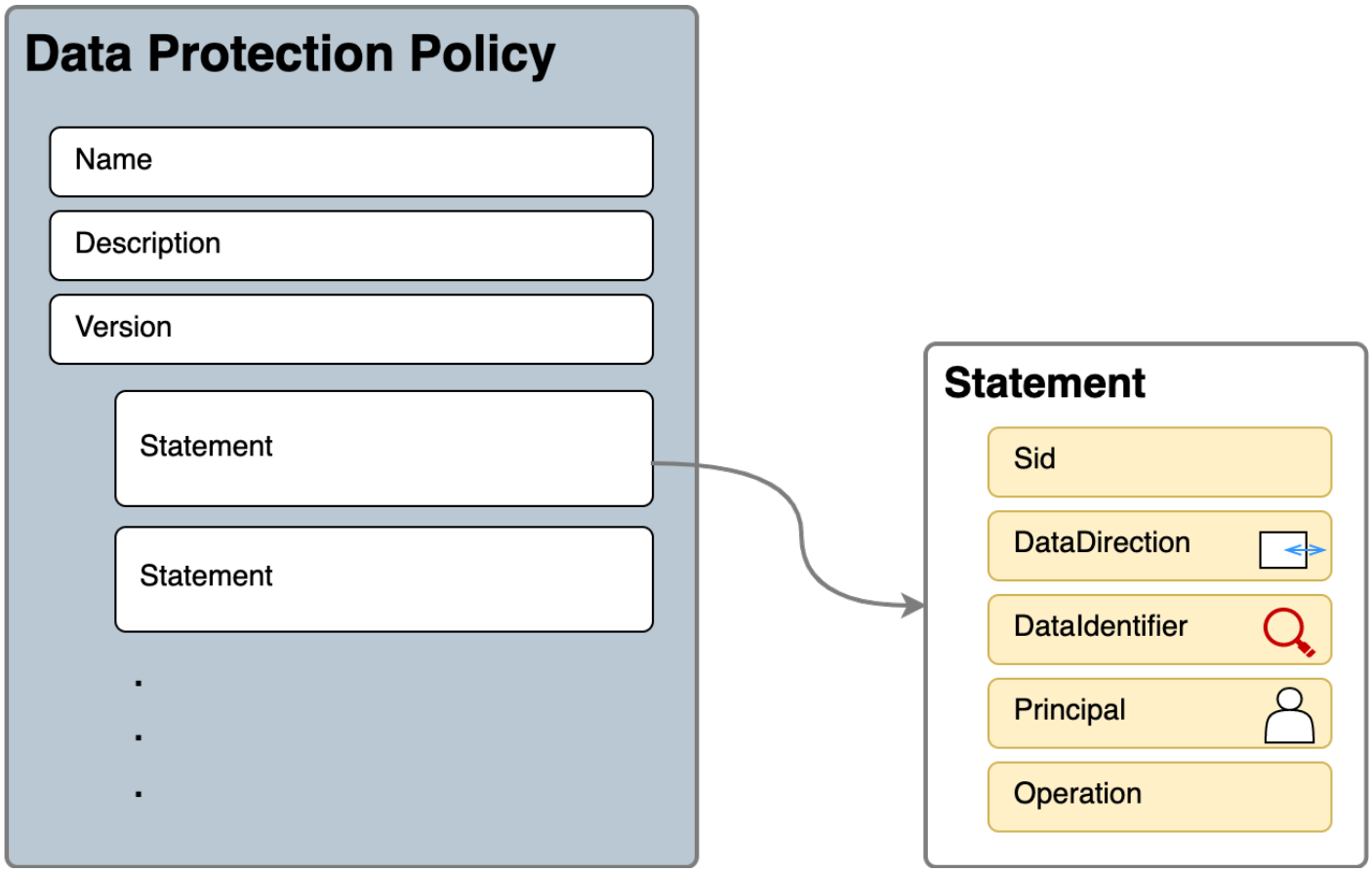


数据保护策略采用什么结构？

如下图所示，数据保护策略文档包含以下元素：

- 文档顶部的可选策略范围信息
- 一个或多个单独语句

每个语句都包含有关单个权限的信息。



每个 Amazon SNS 主题只能定义一个数据保护策略。数据保护策略可以有一个或多个拒绝或去身份识别语句，但只能有一个审计语句。

数据保护策略的 JSON 属性

数据保护策略需要以下基本策略信息用于识别：

- Name（名称）– 策略名称。
- Description（描述）(可选)– 策略描述。
- Version（版本）– 策略语言版本。当前版本为 2021-06-01。
- Statement（语句）– 指定数据保护策略操作的语句列表。

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

```
        ...
  ]
}
```

策略语句的 JSON 属性

策略语句设置数据保护操作的检测上下文。

- Sid（可选）– 语句标识符。
- DataDirection— 与 Amazon SNS 主题相关的入库（用于发布 API 请求）或出库（用于通知传送）。
- DataIdentifier— Amazon SNS 主题应扫描的敏感数据。例如，姓名、地址或电话号码。
- 主体 – 发布到该主题的 IAM 主体或订阅该主题的 IAM 主体。
- Operation（操作）– Amazon SNS 主题在找到敏感数据后执行的后续操作，可以为 Audit（审计）、De-identiyf（去身份识别）（即遮蔽或去除）或者 Deny（拒绝）（即阻止）。

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

策略语句操作的 JSON 属性

策略语句设置以下数据保护操作之一。

- Audit（审计）– 发布指标并发现结果日志，而不中断消息发布或传输。
- De-identify（去身份识别）– 遮蔽或去除敏感数据而不中断消息发布。
- Deny（拒绝）– 阻止 Amazon SNS 发布请求或者使消息传输失败。

如何确定我的数据保护策略的 IAM 主体？

消息数据保护功能使用两个与 Amazon SNS 交互的 IAM 主体。

1. Publish API Principal（发布 API 主体）（入站）– 调用 Amazon SNS `Publish` API 的经过身份验证的 IAM 主体。
2. Subscription Principal（订阅主体）（出站）– 在订阅创建期间调用 `Subscribe` API 的经过身份验证的 IAM 主体。

`SubscriptionPrincipal` 是正式发布的 Amazon SNS 订阅属性，可以从 `GetSubscriptionAttributes` API 检索。

```
{
  "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"
  }
}
```

Amazon SNS 中的数据保护策略操作

以下数据保护策略示例可用于审计和拒绝敏感数据。有关包含示例应用程序的完整教程，请参阅 Introducing message data protection for Amazon SNS（Amazon SNS 消息数据保护简介）博客文章。

审计操作

审计操作对主题入站消息进行采样，并记录 Amazon 目标中发现的敏感数据。采样率可以是 0-99 之间的整数。此操作需要以下类型的日志记录目标之一：

1. FindingsDestination— Amazon SNS 主题在有效负载中发现敏感数据时的记录目标。

2. NoFindingsDestination— 当 Amazon SNS 主题在有效负载中找不到敏感数据时的日志记录目标。

您可以在以下每种日志目标类型 Amazon Web Services 服务 中使用以下内容：

- Amaz CloudWatch on Logs（可选）— `LogGroup` 必须位于主题区域中，并且名称必须以 /aws/vendedlogs/ 开头。
- Amazon Data Firehose（可选）– `DeliveryStream` 必须位于主题区域并且使用 Direct PUT 作为传输流的来源。有关其他详细信息，请参阅《Amazon Data Firehose 开发人员指南》中的来源、目标和名称。
- Amazon S3（可选）– Amazon S3 存储桶名称。要使用启用了 SSE-KMS 加密的 Amazon S3 桶，需要执行额外操作。

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

```
}
```

指定日志目标时所需的权限

在数据保护策略中指定日志记录目标时，对于调用 Amazon SNS `PutDataProtectionPolicy` API 或者带有 `--data-protection-policy` 参数的 `CreateTopic` API 的 IAM 主体，您必须向 IAM 身份策略添加以下权限。

审计目标	IAM 权限
默认	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy 要使用启用了 SSE-KMS 加密的 Amazon S3 桶，需要执行额外操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs:DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:region:account-id:SampleLogGroupName:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}
```

与 SSE-KMS 结合使用时必需的密钥策略

如果您使用 Amazon S3 存储桶作为日志目标，则可以通过使用 Amazon S3 托管密钥启用服务器端加密 (SSE-S3) 或使用 (SSE-KMS) 启用服务器端加密 (SSE-KMS) 来保护存储桶中的数据。 Amazon KMS keys 有关详情，请参阅《Amazon S3 用户指南》中的使用服务器端加密保护数据。

如果选择 SSE-S3，则不需要额外的配置。Amazon S3 处理加密密钥。

如果您选择 SSE-KMS，则必须使用客户托管密钥。您必须更新客户托管密钥的密钥策略，以便日志传输账户可以写入 S3 存储桶。有关与 SSE-KMS 一起使用的所需密钥策略的更多信息，请参阅《亚马逊 CloudWatch 日志用户指南》中的 Amazon S3 存储桶服务器端加密。

审计目标日志示例

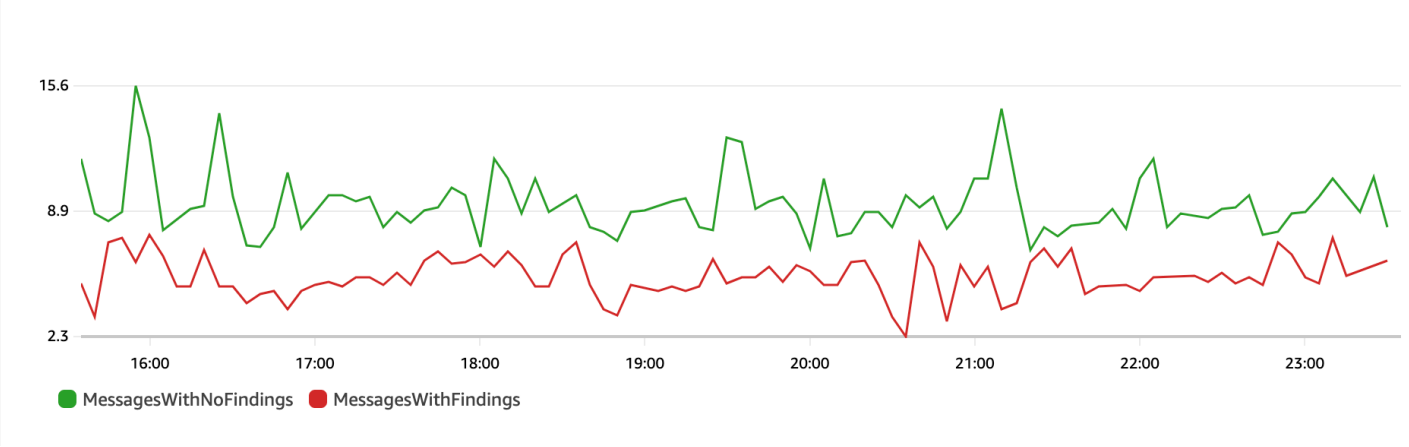
在下例中，使用 `callerPrincipal` 来识别敏感内容的来源，并使用 `messageID` 作为参考来根据 `Publish` API 响应进行检查。

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

审计操作指标

当审计操作指定FindingsDestination或NoFindingsDestination属性时，主题所有者还会收到CloudWatchMessagesWithFindings和MessagesWithNoFindings指标。



去身份识别操作

去身份识别操作会遮蔽或去除所发布或已送达消息中的敏感数据。此操作既适用于入站消息，又适用于出站消息，需要以下类型的配置之一：

- MaskConfig— 使用下表中支持的字符进行掩码。例如，ssn: 123-45-6789 变成 ssn: ###########。

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

支持的遮蔽字符	名称
*	星号
A-Z、a-z 和 0-9	字母数字
	空间
!	感叹号
$	美元符号
%	百分号
&	& 符号
()	括号
+	加号
,	逗号
-	连字符
.	周期
/\	斜杠、反斜杠
#	数字符号
:	冒号
;	分号
=, <>	等于、小于或大于号
@	at 符号
[]	方括号
^	插入符号

支持的遮蔽字符	名称
_	下划线
`	反引号
|	竖线
~	波浪符号

- RedactConfig— 通过完全删除数据进行编辑。例如，ssn: 123-45-6789 变成 ssn: 。

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

对于入站消息，在审计操作之后会对敏感数据进行去身份识别处理，当整条消息全部为敏感数据时，SNS:Publish API 调用方会收到以下无效参数错误。

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

拒绝操作

如果消息包含敏感数据，Deny（拒绝）操作会中断 Publish API 请求，或者中断消息的传输。拒绝操作对象为空，因为它不需要额外配置。

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

在入站消息上，SNS:Publish API 调用方收到授权错误。

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

在出站消息上，Amazon SNS 主题不将消息传输到订阅。要跟踪未经授权的传输，请启用主题的传输状态日志记录。下面是传输状态日志示例：

```
{
```

```
    "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"
}
```

Amazon SNS 数据保护策略示例

以下数据保护策略示例可用于审计和拒绝敏感数据。有关包含示例应用程序的完整教程，请参阅 Introducing message data protection for Amazon SNS（Amazon SNS 消息数据保护简介）博客文章。

审计策略示例

审计策略允许您审计多达 99% 的入站邮件，并将调查结果发送到亚马逊 CloudWatch、Amazon Data Firehose 和亚马逊 S3。

例如，您可以创建审计策略，评估任何系统是否无意中发送或接收了敏感数据。如果您的审计结果表明系统将信用卡信息发送到了无需这些信息的系统，则可以实施数据保护策略来阻止传输此类数据。

以下示例通过查找信用卡号并将发现结果发送到 Lo CloudWatch gs、Firehose 和 Amazon S3 来审核通过该主题的 99% 的消息。

数据保护策略：

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

审计结果格式示例：

```
{
    "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"
}
```

带有入站去身份识别遮蔽语句的策略示例

以下示例通过遮蔽消息内容中的敏感数据，阻止用户将带有 `CreditCardNumber` 的敏感消息发布到主题。

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

入站去身份识别遮蔽结果示例：

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

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

带有入站去身份识别去除语句的策略示例

以下示例通过去除消息内容中的敏感数据，阻止用户将带有 `CreditCardNumber` 的敏感消息发布到主题。

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

入站去身份识别去除结果示例：

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

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

带有出站去身份识别遮蔽语句的策略示例

以下示例通过遮蔽消息内容中的敏感数据，阻止用户使用 `CreditCardNumber` 接收消息。

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

出站去身份识别遮蔽结果示例：

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

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

带有入站去身份识别去除语句的策略示例

以下示例通过去除消息内容中的敏感数据，阻止用户使用 `CreditCardNumber` 接收消息。

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

出站去身份识别去除结果示例：

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

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

入站拒绝语句示例策略

以下示例阻止用户将消息内容中带有 `CreditCardNumber` 的消息发布到主题。API 响应中被拒绝的负载的状态码为”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": {}
      }
    }
  ]
}
```

出站拒绝语句示例策略

以下示例阻止一个 Amazon 账户接收包含以下内容的消息`CreditCardNumber`。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

出站拒绝结果示例，已登录 Amazon CloudWatch：

```
{
  "notification": {
    "messageMD5Sum": "2e8f58ff2eeed723b56b15493fbfb5a5",
    "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
    "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
    "timestamp": "2022-09-08 15:40:57.144"
  },
  "delivery": {
    "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
    "destination": "arn:aws:sqs:us-east-2:664555388960:test",
    "providerResponse": "The topic's data protection policy prohibits this message from
 being delivered to <subscription arn>",
    "dwellTimeMs": 22,
    "attempts": 1,
    "statusCode": 403
  },
  "status": "FAILURE"
```

```
}
```

在 Amazon SNS 中创建数据保护策略

数据保护策略通过审计、去身份识别（遮蔽或去除）和拒绝（阻止）敏感信息在应用程序或 Amazon Web Services 服务之间传输，帮助您保护发布到 Amazon SNS 主题的数据。您可以使用 Amazon API、 Amazon CLI Amazon CloudFormation、或 Amazon Web Services Management Console 在 Amazon SNS 中创建数据保护策略。每个 Amazon SNS 主题只能定义一个策略。每个数据保护策略可以有一个或多个去身份识别和拒绝语句，但只能有一个审计语句。

主题

- 使用 API 在 Amazon SNS 中创建数据保护策略
- 使用 CLI 在 Amazon SNS 中创建数据保护策略
- 使用在 Amazon SNS 中创建数据保护策略 CloudFormation
- 使用控制台在 Amazon SNS 中创建数据保护策略
- 使用 SDK 创建 Amazon SNS 数据保护策略以保护消息数据

使用 API 在 Amazon SNS 中创建数据保护策略

一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service endpoints and quotas。

使用 API 创建数据保护策略

使用 API 创建亚马逊 SNS 数据保护政策。 Amazon

与 Amazon SNS 主题一起创建数据保护策略 (Amazon API)

使用标准 Amazon SNS 主题的 `DataProtectionPolicy` 属性：

- `CreateTopic`

为现有 Amazon SNS 主题检索或创建数据保护策略 (Amazon API)

调用下列一项操作：

- GetDataProtectionPolicy
- PutDataProtectionPolicy

使用 CLI 在 Amazon SNS 中创建数据保护策略

一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service endpoints and quotas。

使用 Amazon CLI创建数据保护策略

使用创建 Amazon SNS 数据保护政策。 Amazon Command Line Interface

与 Amazon SNS 主题一起创建数据保护策略 (Amazon CLI)

使用此选项，与标准 Amazon SNS 主题一起创建新的数据保护策略：

- create-topic

为现有 Amazon SNS 主题创建或检索数据保护策略 (Amazon CLI)

调用下列一项操作：

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

使用在 Amazon SNS 中创建数据保护策略 CloudFormation

一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service endpoints and quotas。

创建数据保护策略 (CloudFormation)

使用创建 Amazon SNS 数据保护政策。 Amazon CloudFormation

与 Amazon SNS 主题一起创建数据保护策略 (CloudFormation)

使用此选项，与标准 Amazon SNS 主题一起创建新的数据保护策略：

- AWS::SNS::Topic

使用控制台在 Amazon SNS 中创建数据保护策略

一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service endpoints and quotas。

与 Amazon SNS 主题一起创建数据保护策略（控制台）

使用此选项，与标准 Amazon SNS 主题一起创建新的数据保护策略。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题或创建一个新主题。有关创建主题的详细信息，请参阅创建 Amazon SNS 主题。
3. 在 Create topic（创建主题）页面的 Details（详细信息）部分，选择 Standard（标准）。
   a. 输入主题的名称。
   b. （可选）输入主题的显示名称。
4. 展开 Data protection policy（数据保护策略）。
5. 选择一个 Configuration mode（配置模式）：
   - Basic（基本）– 使用简单菜单定义数据保护策略。
   - Advanced（高级）– 使用 JSON 定义自定义数据保护策略。
6. （可选）要创建您自己的自定义数据标识符，请展开自定义数据标识符配置部分，执行以下操作：
   a. 为自定义数据标识符输入唯一名称。自定义数据标识符名称支持字母数字、下划线（_）和连字符（-）等字符。最多支持 128 个字符。此名称不能与托管式数据标识符同名。有关自定义数据标识符限制的完整列表，请参阅???。
   b. 输入自定义数据标识符的正则表达式 (RegEx)。 RegEx支持字母数字字符、 RegEx 保留字符和符号。 RegEx 最大长度为 200 个字符。如果太复杂，Amazon SNS 就会失败 API 调用。RegEx 有关 RegEx限制的完整列表，请参阅???。
   c. （可选）选择添加自定义数据标识符以根据需要添加其它数据标识符。每项数据保护策略最多支持 10 个自定义数据标识符。
7. 选择您要添加到数据保护策略中的语句。您可以将审计、去身份识别（遮蔽或去除）和拒绝（阻止）语句类型添加到同一数据保护策略中。
   a. Add audit statement（添加审计语句）– 配置要审计的敏感数据、要为该数据审计的消息百分比以及将审计日志发送到何处。

   > **Note**
   > 每个数据保护策略或主题仅允许使用一个审计语句。

i. 选择 data identifiers（数据标识符）以定义要审计的敏感数据。

ii. 对于 Audit sample rate（审计采样率），输入要在其中审计敏感信息的消息百分比，最大值为 99%。

iii. 在 “审计目标” 中，选择 Amazon Web Services 服务 要发送审计结果的目的地，然后为您 Amazon Web Services 服务 使用的每个目标输入目标名称。您可以从以下 Amazon Web Services 中进行选择：

- Amazon CloudWatch — CloudWatch Logs 是 Amazon 标准的日志解决方案。使用 CloudWatch 日志，您可以使用 Logs Insights（参见此处的示例）执行日志分析，并创建指标和警报。 CloudWatch 日志是许多服务发布日志的地方，这使得使用一个解决方案可以更轻松地聚合所有日志。有关亚马逊的信息 CloudWatch，请参阅亚马逊 CloudWatch 用户指南。
- Amazon Data Firehos e — Firehose 可以满足实时直播到 Splunk 的需求，而 OpenSearch亚马逊 Redshift 可以满足进一步日志分析的需求。有关 Amazon Data Firehose 的信息，请参阅 Amazon Data Firehose 用户指南。
- Amazon Simple Storage Service – Amazon S3 是经济实惠的日志目标，可用于存档目的。您可能需要将日志保留数年。在这种情况下，您可以将日志放入 Amazon S3 中以节省成本。有关 Amazon Simple Storage Service 的信息，请参阅 Amazon Simple Storage Service 用户指南。

b. Add a de-identify statement（添加去身份识别语句）– 配置要在消息中去身份识别的敏感数据（无论是遮蔽还是去除该数据），并且账户将停止传输该数据。

i. 对于 Data Identifiers（数据标识符），选择要去身份识别的敏感数据。

ii. 在 “为其定义此去身份化声明” 中，选择此去身份化声明适用的 Amazon 账户或 IAM 委托人。您可以将其应用于所有 Amazon 账户，也可以应用于使用 Amazon 账户或 IAM 实体的特定账户或 IAM 实体（账户根、角色 IDs 或用户） ARNs。分隔多个 IDs 或 ARNs 使用逗号 (,)。

以下是受支持的 IAM 主体：

- IAM account principals（IAM 账户主体）– 例如 `arn:aws:iam::Amazon-account-ID:root`。
- IAM role principals（IAM 角色主体）– 例如 `arn:aws:iam::Amazon-account-ID:role/role-name`。

- IAM use principals（IAM 用户主体）– 例如 `arn:aws:iam::Amazon-account-ID:user/user-name`。

iii. 对于 De-identify Option（去身份识别选项），请选择要如何对敏感数据去身份识别。支持以下选项：

- Redact（去除）– 完全删除数据。例如，电子邮件 `classified@amazon.com` 将变为电子邮件 。
- Mask（遮蔽）– 使用单个字符替换数据。例如，电子邮件 `classified@amazon.com` 将变为电子邮件 `**********`。

iv. （可选）根据需要继续添加去身份识别语句。

c. Add deny statement（添加拒绝语句）– 配置要在您的主题中阻止传输哪些敏感数据，以及阻止哪些主体传输这些数据。

i. 对于 Data Direction（数据方向），请选择拒绝语句的消息方向：

- Inbound messages（入站消息）– 将此拒绝语句应用于发送到该主题的消息。
- Outbound messages（出站消息）– 将此拒绝语句应用于主题传输到订阅端点的消息。

ii. 选择 Data Identifiers（数据标识符）以定义要拒绝的敏感数据。

iii. 选择应用到此拒绝语句的 IAM principals（IAM 主体）。您可以将其应用于使用 Amazon 账户或 IAM 实体的所有账户 Amazon Web Services 账户、特定或 IAM 实体（例如，账户根、角色 IDs 或用户） ARNs。分隔多个 IDs 或 ARNs 使用逗号 (,)。以下是受支持的 IAM 主体：

- IAM account principals（IAM 账户主体）– 例如 `arn:aws:iam::Amazon-account-ID:root`。
- IAM role principals（IAM 角色主体）– 例如 `arn:aws:iam::Amazon-account-ID:role/role-name`。
- IAM use principals（IAM 用户主体）– 例如 `arn:aws:iam::Amazon-account-ID:user/user-name`。

iv. （可选）根据需要继续添加拒绝语句。

使用 SDK 创建 Amazon SNS 数据保护策略以保护消息数据

一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service endpoints and quotas。

使用 Amazon SDK 创建数据保护策略

使用软件开发工具包创建 Amazon SNS 数据保护政策。 Amazon

与 Amazon SNS 主题一起创建数据保护策略 (Amazon SDK)

使用以下选项，与标准 Amazon SNS 主题一起创建新的数据保护策略：

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();
```

为现有 Amazon SNS 主题创建或检索数据保护策略 (Amazon SDK)

使用以下选项，与标准 Amazon SNS 主题一起创建新的数据保护策略或检索数据保护策略：

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();
```

在 Amazon SNS 中删除数据保护策略

您可以使用 Amazon API、、 Amazon CLI或删除 Amazon SNS 数据保护策略。 Amazon CloudFormation Amazon Web Services Management Console

有关 Amazon SNS 数据保护策略的一般信息，请参阅 了解 Amazon SNS 数据保护策略。

Amazon 账户中 Amazon SNS 数据保护策略资源的数量和大小是有限的。有关更多信息，请参阅《 Amazon Web Services 一般参考》中的 Amazon SNS API 节流。

使用控制台删除数据保护策略

使用控制台删除托管数据保护策略

1. 登录 Amazon SNS 控制台。
2. 选择包含您要删除的数据保护策略的主题。
3. 选择编辑。
4. 展开 Data protection policy（数据保护策略）部分。
5. 在您要删除的数据保护策略语句旁边，选择 Remove（删除）。
6. 选择 Save changes（保存更改）。

使用空 JSON 字符串删除数据保护策略

您可以通过将数据保护策略更新为空 JSON 字符串来删除数据保护策略。

使用删除数据保护策略 Amazon CLI

您可以使用 Amazon CLI删除数据保护策略。

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

Amazon SNS 数据标识符

Amazon SNS 将包括机器学习和模式匹配在内的标准和技术结合使用来检测敏感数据。这些标准和技术统称为数据标识符，可以检测许多国家/地区和区域的大量且不断增长的敏感数据类型。Amazon SNS 托管式数据标识符提供预配置的数据类型，用于保护财务数据、个人健康信息（PHI）和个人身份信息（PII）。

在 Amazon SNS 中使用托管数据标识符

什么是托管数据标识符？

Amazon SNS 托管数据标识符旨在检测特定类型的敏感数据，例如信用卡号、 Amazon 秘密访问密钥或特定国家或地区的护照号码。创建数据保护策略时，您可以将 Amazon SNS 配置为使用这些标识符来分析通过主题传送的消息，并在检测到敏感数据时采取措施。

Amazon SNS 可以使用托管数据标识符检测以下类别的敏感数据：

- 证书，例如私钥或私有访问 Amazon 密钥
- 设备标识符，例如 IP 地址或 MAC 地址
- 财务信息，例如信用卡号
- 健康信息，对于 PHI，包括健康保险或医疗标识号等
- 个人信息，对于 PII，包括驾驶执照或社会安全号码

在每个类别中，Amazon SNS 可以检测多种类型的敏感数据。本节中的主题列出并描述了各种类型以及对其进行检测的相关要求。对于每种类型，它们还说明了托管数据标识符的唯一标识符 (ID)，此标识符设计用于检测数据。创建数据保护策略时，您可以使用此 ID 包含托管数据标识符，供消息数据保护功能进行检测。

关键字要求

为了检测某些类型的敏感数据，Amazon SNS 会扫描数据附近位置的关键字。如果特定类型的数据属于这种情况，此部分中的后续主题将说明该数据的特定关键字要求。

关键字不区分大小写。此外，如果关键字包含空格，Amazon SNS 会自动匹配不包含空格的变体，或包含下划线 (_) 或连字符 (-) 而不是空格的关键字变体。在某些情况下，Amazon SNS 还会扩展或缩写关键字以应对该关键字的常见变体。

敏感数据类型的 Amazon SNS 托管数据标识符

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测的凭证、设备、财务、医疗和个人健康信息 (PHI, Protected Health Information) 类型。除了特定类型的数据之外，这些数据可能也符合个人身份信息 (PII, Personally Identifiable Information) 的条件。

区域相关数据标识符要求使用标识符名称、破折号和两个字母 (ISO 3166-1 alpha-2) 代码。例如，DriversLicense-US。

标识符	类别	国家/地区/语言
BankAccountNumber	财务	DE、ES、FR、GB、IT
CepCode	个人	BR
Cnpj	个人	BR
CpfCode	个人	BR
DriversLicense	个人	AT、AU、BE、BG、CA、CY、CZ、DE、DK、EE、ES、FI、FR、GB、GR、HR、HU、IE、IT、LT、LU、LV、MT、NL、PL、PT、RO、SE、SI、SK、US
DrugEnforcementAgencyNumber	健康	美国

标识符	类别	国家/地区/语言
ElectoralRollNumber	个人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	美国
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	美国
IndividualTaxIdentification Number	个人	美国
InseeCode	个人	FR
MedicareBeneficiaryNumber	健康	美国
NationalDrugCode	健康	美国
NationalIdentificationNumber	个人	DE、ES、IT
NationalInsuranceNumber	个人	GB
NationalProviderId	健康	美国
NhsNumber	健康	GB
NieNumber	个人	ES
NifNumber	个人	ES
PassportNumber	个人	CA、DE、ES、FR、GB、IT、US
PermanentResidenceNumber	个人	CA
PersonalHealthNumber	健康	CA
PhoneNumber	个人	BR、DE、ES、FR、GB、IT、US

标识符	类别	国家/地区/语言
PostalCode	个人	CA
RgNumber	个人	BR
SocialInsuranceNumber	个人	CA
Ssn	个人	ES、US
TaxId	个人	DE、ES、FR、GB
ZipCode	个人	美国

与语言/地区无关的受支持的标识符

标识符	类别
地址	个人
AwsSecretKey	凭证
CreditCardExpiration	财务
CreditCardNumber	财务
CreditCardSecurityCode	财务
EmailAddress	个人
IpAddress	个人
LatLong	个人
名称	个人
OpenSshPrivateKey	凭证
PgpPrivateKey	凭证

标识符	类别
PkcsPrivateKey	凭证
PuttyPrivateKey	凭证
VehicleIdentificationNumber	个人

Amazon SNS 敏感数据类型：凭证

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测的凭证类型。

检测类型	托管数据标识符 ID	所需关键字	国家和地区
Amazon 秘密访问密钥	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	任何
OpenSSH 私有密钥	OpenSshPrivateKey	否	任何
PGP 私有密钥	PgpPrivateKey	否	任何
公有密钥加密标准 (PKCS, Public-Key Cryptography Standard) 私有密钥	PkcsPrivateKey	否	任何
PuTTY 私有密钥	PuttyPrivateKey	否	任何

凭证数据 ARNs 类型的数据标识符

以下列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

凭证数据标识符 ARNs
arn: aws: dataprotection:: aws: 数据标识符/ AwsSecretKey

凭证数据标识符 ARNs
arn: aws: dataprotection:: aws: 数据标识符/ OpenSshPrivateKey
arn: aws: dataprotection:: aws: 数据标识符/ PgpPrivateKey
arn: aws: dataprotection:: aws: 数据标识符/ PkcsPrivateKey
arn: aws: dataprotection:: aws: 数据标识符/ PuttyPrivateKey

Amazon SNS 敏感数据类型：设备

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测的设备标识符类型。

检测类型	托管数据标识符 ID	所需关键字	国家和地区
IP 地址	IpAddress	否	任何

设备数据类型的数据 ARNs 标识符

以下列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

设备数据标识符 ARN
arn: aws: dataprotection:: aws: 数据标识符/ IpAddress

Amazon SNS 敏感数据类型：财务

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测的财务信息类型。

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
银行账号	BankAccountNumber	是，请参阅 银行账号的关键字。	这包括：由最多 34 个字母数字字符组成的	法国、德国、意大利、西班牙、英国

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
	BankAccountNumber-US		国际银行账号 (IBANs)，包括国家代码等元素。	
信用卡到期日期	CreditCardExpiration	exp d、exp m、exp y、expiration、expiry	–	任何
信用卡磁条数据	CreditCardMagneticStripe	是，包括：card data、iso7813、mag、magstripe、stripe、swipe。	这包括轨道 1 和 2。	任何

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
信用卡号	CreditCardNumber	account number、american express、amex、bank card、card、card num、card number、cc #、ccn、check card、credit、credit card#、dankort、debit、debit card、diners club、discover、electron、elo verification code、japanese card bureau、jcb、mastercard、mc、pan、payment account number、payment card number、pcn、union pay、visa	检测要求数据为符合卢恩支票公式的13-19位数序列，并对以下任何类型的信用卡使用标准卡号前缀：美国运通、Dankort、Diner's Club、Discover、Electron、日本信用卡局（JCB）UnionPay、万事达卡和Visa（上标链接下方1）。	任何

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
信用卡验证码	CreditCardSecurityCode	card id、card identification code、card identification number、card security code、card validation code、card validation number、card verification data、card verification value、cvc、cvc2、cvv、cvv2、elo verification code	–	任何

1. Amazon SNS 不会报告出现的以下序列，信用卡发卡机构保留这些序列供公开测试：

   122000000000003、2222405343248877、2222990905257051、2223007648726984、2223577120017 6 和 76009244561。

银行账号的关键字

使用以下关键字检测由最多 34 个字母数字字符组成的国际银行账号 (IBANs)，包括国家/地区代码等元素。

国家或地区	关键字			
法国	account code, account number,			

国家或地区	关键字			
	accountno#, accountnumber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
德国	account code, account number, accountno#, accountnumber#, bankleitzahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			

国家或地区	关键字			
意大利	account code, account number, accountno #, accountnu mber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			
西班牙	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			

国家或地区	关键字			
UK	account code, account number, accountno #, accountnumber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			
美国	bank account、bank acct、checking account、checking acct、deposit account、deposit acct、savings account、savings acct、chequing account、chequing acct			

财务数据类型的数据 ARNs 标识符

以下列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

财务数据标识符 ARNs
arn: aws: 数据保护:: aws: data-identifier/-DE BankAccountNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES BankAccountNumber

财务数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifir BankAccountNumber
arn: aws: dataprotection:: aws: data-identifer BankAccountNumber
arn: aws: dataprotection:: aws: data-it BankAccountNumber
arn: aws: dataprotection:: aws: data-identif BankAccountNumber
arn: aws: dataprotection:: aws: 数据标识符/ CreditCardExpiration
arn: aws: dataprotection:: aws: 数据标识符/ CreditCardNumber
arn: aws: dataprotection:: aws: 数据标识符/ CreditCardSecurityCode

Amazon SNS 敏感数据类型：受保护健康信息（PHI）

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测的受保护健康信息 (PHI, Protected Health Information) 类型。

检测类型	托管数据标识符 ID	所需关键字	国家和地区
毒品管理局 (DEA, Drug Enforcement Agency) 注册号	DrugEnforcementAgencyNumber	dea number, dea registration	美国
健康保险卡号 (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero、carta assicurazione numero、carte d’assurance maladie、carte européenne d'assurance maladie、ceam、ehic、ehic#、finlandehicnumber#、gesundheits	EU

检测类型	托管数据标识符 ID	所需关键字	国家和地区
		karte、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、sairausvakuutus numero、sjukförsäkring nummer、sjukförsäkringskort、suomi ehic-numero、tarjeta de salud、terveyskortti、tessera sanitaria assicurazione	

检测类型	托管数据标识符 ID	所需关键字	国家和地区
		numero、versicherungsnummer	
健康保险索赔编号 (HICN, Health Insurance Claim Number)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	美国
健康保险或医疗识别号	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR
医疗保健通用程序编码系统 (HCPCS, Healthcare Common Procedure Coding System) 代码	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	美国
医疗保险受益人号码 (MBN, Medicare Beneficiary Number)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	美国
国家药品编码 (NDC, National Drug Code)	NationalDrugCode	national drug code, ndc	美国
国家提供商识别码 (NPI, National Provider Identifier)	NationalProviderId	hipaa, n.p.i, national provider, npi	美国
国家健康服务 (NHS, National Health Service) 号码	NhsNumber	national health service, NHS	GB

检测类型	托管数据标识符 ID	所需关键字	国家和地区
个人健康号码 (PHN, Personal Health Number)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

健康保险和医疗识别号的关键字

为了检测各种类型的健康保险和医疗识别号，Amazon SNS 要求关键字的位置必须靠近数字。这包括欧洲健康保险卡号（欧盟、芬兰）、健康保险号码（法国）、医疗保险受益人识别码（美国）、国民保险号码（英国）、NHS 号码（英国）和个人健康号码（加拿大）。

下表列出了 Amazon SNS 识别的特定国家和地区的关键字。

国家或地区	关键词
加拿大	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, sjukförsä

国家或地区	关键词
	kring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicher ungsnummer
芬兰	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskor t, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin , sairausvakuutuskortti, sairausvakuutusnum ero, sjukförsäkring nummer, sjukförsäkringskor t, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
法国	carte d'assuré social, carte vitale, insurance card
UK	national health service、NHS
美国	mbi, medicare beneficiary

受保护 ARNs 的健康信息数据类型 (PHI) 的数据标识符

以下列出了可用于 PHI 数据保护策略的数据标识符 Amazon 资源名称 (ARNs)。

PHI 数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identif DrugEnforcementAgencyNumber
arn: aws: dataprotection:: aws: data-identif HealthcareProcedureCode
arn: aws: dataprotection:: aws: data-identifer HealthInsuranceCardNumber
arn: aws: dataprotection:: aws: data-identif HealthInsuranceClaimNumber
arn: aws: dataprotection:: aws: data-identifir HealthInsuranceNumber

PHI 数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identif MedicareBeneficiaryNumber
arn: aws: dataprotection:: aws: data-identif NationalDrugCode
arn: aws: dataprotection:: aws: data-identifer NationalInsuranceNumber
arn: aws: dataprotection:: aws: data-identif NationalProviderId
arn: aws: dataprotection:: aws: data-identifer NhsNumber
arn: aws: 数据保护:: aws: data-identifier/-CA PersonalHealthNumber

Amazon SNS 敏感数据类型：个人身份信息（PII）

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测的个人身份信息 (PII, Personally Identifiable Information)。

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
出生日期	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	支持包括大多数日期格式，例如所有数字以及数字和月份名称的组合。日期组件可以用空格、斜杠（/）或连字符（-）分隔。	任何
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	巴西

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	巴西
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	巴西

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
驾驶执照识别号	DriversLicense	是，请参阅 驾驶执照识别号的关键字。	–	澳大利亚、奥地利、比利时、保加利亚、加拿大、克罗地亚、塞浦路斯、捷克共和国、丹麦、爱沙尼亚、芬兰、法国、德国、希腊、匈牙利、爱尔兰、意大利、拉脱维亚、立陶宛、卢森堡、马耳他、荷兰、波兰、葡萄牙、罗马尼亚、斯洛伐克、斯洛文尼亚、西班牙、瑞典、英国、美国
选民名册编号	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
个人纳税人识别号	IndividualTaxIdentificationNumber	是，请参阅 纳税人识别号和参考号的关键字。	–	美国
国家统计与经济研究所 (INSEE, National Institute for Statistics and Economic Studies)	InseeCode	是，请参阅 国民身份证号的关键字。	–	法国
身份证号码	NationalIdentificationNumber	是，请参阅 国民身份证号的关键字。	这包括 Documento Nacional de Identidad (DNI) 识别号（西班牙）、Codice fiscale codes（意大利）和国民身份证号（德国）。	德国、意大利、西班牙
国民保险号码 (NINO, National Insurance Number)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	UK

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
Número de identidad de extranjero (NIE)	NieNumber	是，请参阅 纳税人识别号和参考号的关键字。	–	西班牙
Número de Identificación Fiscal (NIF)	NifNumber	是，请参阅 纳税人识别号和参考号的关键字。	–	西班牙
护照编号	PassportNumber	是，请参阅 护照号码的关键字。	–	加拿大、法国、德国、意大利、西班牙、英国、美国

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
永久居留号码	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	–	加拿大

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
电话号码	PhoneNumber	巴西：关键字还包括 cel、celular、fone、móvel、número residencial、numero residencial、telefone 其他：cell、contact、fax、fax number、mobile、phone、phone number、tel、telephone、telephone number	这包括美国的免费电话号码和传真号码。如果关键字靠近数据，则数字不必包含国家/地区代码。如果关键字并不靠近数据，则数字必须包含国家/地区代码。	巴西、加拿大、法国、德国、意大利、西班牙、英国、美国
邮政编码	PostalCode	No	–	加拿大
Registro Geral (RG)	RgNumber	是，请参阅 国民身份证号的关键字。	–	巴西
社会保险号码 (SIN, Social Insurance Number)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	加拿大

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
社会保障号码 (SSN, Social Security number)	Ssn	西班牙 – número de la seguridad social、social security no.、social security no. número de la seguridad social、social security number、socialsecurityno#、ssn、ssn# 美国 – social security、ss#、ssn	–	西班牙、美国
纳税人识别号或参考号	TaxId	是，请参阅 纳税人识别号和参考号的关键字。	这包括 TIN（法国）；Steueridentifikationsnummer（德国）；CIF（西班牙）；以及 TRN、UTR（英国）。	法国、德国、西班牙、英国
美国邮政编码	ZipCode	zip code, zip+4	–	美国

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
邮寄地址	Address	No	尽管关键字并非必需，但检测过程要求地址包含城市或地点的名称以及邮政编码。	澳大利亚、加拿大、法国、德国、意大利、西班牙、英国、美国
电子邮件地址	EmailAddress	电子邮件、电子邮件地址、邮件、邮件地址	–	任何

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
全球定位系统 (GPS, Global Positioning System) 坐标	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS 可以检测纬度和经度坐标成对存储并且采用十进制度数 (DD, Decimal Degrees) 格式的 GPS 坐标，例如 41.948614 ,-87.655311。不支持采用度十进制分 (DDM, Degrees Decimal Minutes) 格式的坐标，例如 41°56.9168'N 87°39.3187'W，也不支持度分秒 (DMS, Degrees, Minutes, Seconds) 格式的坐标，例如 41°56'55.0104"N 87°39'19. 1196"W。	任何
全名	Name	No	Amazon SNS 只能检测全名。只支持拉丁字符集。	任何

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
车辆识别号码 (VIN, Vehicle Identification Number)	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 VINs 可以检测包含 17 个字符的序列并符合 ISO 3779 和 3780 标准的内容。这些标准旨在供全球使用。	任何

驾驶执照识别号的关键字

为了检测各种类型的驾照标识号，Amazon SNS 要求关键字的位置必须靠近数字。下表列出了 Amazon SNS 识别的特定国家和地区的关键字。

国家或地区	关键词
澳大利亚	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

国家或地区	关键词
奥地利	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
比利时	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
保加利亚	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
加拿大	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit, permis de conduire
克罗地亚	vozačka dozvola
塞浦路斯	άδεια οδήγησης
捷克共和国	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
丹麦	kørekort, kørekortnummer

国家或地区	关键词
爱沙尼亚	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
芬兰	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
法国	permis de conduire
德国	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
希腊	δεια οδήγησης, adeia odigisis
匈牙利	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
爱尔兰	ceadúnas tiomána
意大利	patente di guida, patente di guida numero, patente guida, patente guida numero
拉脱维亚	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
立陶宛	vairuotojo pažymėjimas
卢森堡	fahrerlaubnis, führerschäin
马耳他	liċenzja tas-sewqan
荷兰	permis de conduire, rijbewijs, rijbewijsnummer

国家或地区	关键词
波兰	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
葡萄牙	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
罗马尼亚	numărul permisului de conducere, permis de conducere
斯洛伐克	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
斯洛文尼亚	vozniško dovoljenje
西班牙	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
瑞典	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

国家或地区	关键词
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
美国	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

国民身份证号的关键字

为了检测各种类型的国民身份证号，Amazon SNS 要求关键字的位置必须靠近数字。这包括 Documento Nacional de Identidad (DNI) 标识符（西班牙）、French National Institute for Statistics and Economic Studies（INSEE）代码、德国国民身份证号码，以及注册总署（RG, Registro Geral）号码（巴西）。

下表列出了 Amazon SNS 识别的特定国家和地区的关键字。

国家或地区	关键词
巴西	registro geral, rg
法国	assurance sociale, carte nationale d’identité, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#,

国家或地区	关键词
	numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
德国	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
意大利	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
西班牙	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#

护照号码的关键字

为了检测各种类型的护照号码，Amazon SNS 要求关键字的位置必须靠近数字。下表列出了 Amazon SNS 识别的特定国家和地区的关键字。

国家或地区	关键词
加拿大	passeport, passeport#, passport, passport#, passportno, passportno#
法国	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

国家或地区	关键词
德国	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
意大利	italian passport number, numéro passeport , numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
西班牙	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
美国	passport, travel document

纳税人识别号和参考号的关键字

为了检测各种类型的纳税人识别号和参考号，Amazon SNS 要求关键字的位置必须靠近数字。下表列出了 Amazon SNS 识别的特定国家和地区的关键字。

国家或地区	关键词
巴西	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
法国	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#

国家或地区	关键词
德国	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
西班牙	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
美国	个人纳税人识别号（itin, i.t.i.n.）

个人身份信息 (PII) 的数据标识符 ARNs

下表列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

PII 数据标识符 ARNs
arn:aws:dataprotection::aws:data-identifier/Address
arn: aws: dataprotection:: aws: data-identifer CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn: aws: dataprotection:: aws: data-identifer CpfCode
arn: aws: dataprotection:: aws: 数据标识符/ DateOfBirth
arn: aws: dataprotection:: aws: 数据标识符/-AT DriversLicense

PII 数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-CA DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifier/ DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-DE DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-DK DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-EE DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-ES DriversLicense
arn: aws: dataprotection:: aws: data-ident DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-HU DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-IE DriversLicense
arn: aws: dataprotection:: aws: data-it DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-LT DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-LU DriversLicense

PII 数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-MT DriversLicense
arn: aws: dataprotection:: aws: data-identifier/ DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-PT DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-RO DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-SE DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identif DriversLicense
arn: aws: dataprotection:: aws: data-identifer ElectoralRollNumber
arn: aws: dataprotection:: aws: 数据标识符/ EmailAddress
arn: aws: dataprotection:: aws: data-identif IndividualTaxIdentificationNumber
arn: aws: dataprotection:: aws: data-identifir InseeCode
arn: aws: dataprotection:: aws: 数据标识符/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn: aws: 数据保护:: aws: data-identifier/-DE NationalIdentificationNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES NationalIdentificationNumber
arn: aws: dataprotection:: aws: data-it NationalIdentificationNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES NieNumber

PII 数据标识符 ARNs
arn: aws: dataprotection:: aws: 数据标识符/-ES NifNumber
arn: aws: dataprotection:: aws: data-identifir PassportNumber
arn: aws: 数据保护:: aws: data-identifier/-DE PassportNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES PassportNumber
arn: aws: dataprotection:: aws: data-identifir PassportNumber
arn: aws: dataprotection:: aws: data-identifer PassportNumber
arn: aws: dataprotection:: aws: data-it PassportNumber
arn: aws: dataprotection:: aws: data-identif PassportNumber
arn: aws: dataprotection:: aws: data-identifir PermanentResidenceNumber
arn: aws: dataprotection:: aws: data-identifer PhoneNumber
arn: aws: 数据保护:: aws: data-identifier/-DE PhoneNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES PhoneNumber
arn: aws: dataprotection:: aws: data-identifir PhoneNumber
arn: aws: dataprotection:: aws: data-identifer PhoneNumber
arn: aws: dataprotection:: aws: data-it PhoneNumber
arn: aws: dataprotection:: aws: data-identif PhoneNumber
arn: aws: dataprotection:: aws: data-identifir PostalCode
arn: aws: dataprotection:: aws: data-identifer RgNumber
arn: aws: dataprotection:: aws: data-identifir SocialInsuranceNumber
arn:aws:dataprotection::aws:data-identifier/Ssn-ES

PII 数据标识符 ARNs
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn: aws: 数据保护:: aws: data-identifier/-DE TaxId
arn: aws: dataprotection:: aws: 数据标识符/-ES TaxId
arn: aws: dataprotection:: aws: data-identifir TaxId
arn: aws: dataprotection:: aws: data-identifer TaxId
arn: aws: dataprotection:: aws: 数据标识符/ VehicleIdentificationNumber
arn: aws: dataprotection:: aws: data-identif ZipCode

Amazon SNS 消息传输

本主题介绍 Amazon SNS 如何处理各种场景中的消息传送。您将了解原始消息传输，其中 Amazon SNS 以未经修改的原始格式将消息传送到端点。您还将了解如何将消息从 Amazon SNS 主题发送到 Amazon Web Services 账户另一个 Amazon SQS 队列，从而深入了解跨账户消息传送。

本主题提供有关将 Amazon SNS 消息传送到 Amazon Web Services 区域不同的 Amazon SQS 队列或 Lambda 函数的信息、跨区域交付的工作原理以及所涉及的注意事项。

此外，您还将了解如何监控和解释消息传送状态，从而提供有关消息是否成功送达或遇到问题的关键信息。如果消息传送失败，您将了解消息传送重试流程，包括 Amazon SNS 如何自动尝试重新传送消息以确保消息到达预期目标。本主题还讨论了如何使用死信队列来捕获多次尝试后无法传送的消息，从而使您能够有效地分析和排除这些失败。

Amazon SNS 原始消息传输

为避免使 Amazon Data Firehose、Amazon SQS 和 HTTP/S 端点处理 JSON 格式的消息，Amazon SNS 允许原始消息传输：

- 当您为 Amazon Data Firehose 或 Amazon SQS 端点启用原始消息传输时，将从已发布的消息中删除任何 Amazon SNS 元数据，并按原样发送消息。
- 当您为 HTTP/S 终端节点启用原始消息传递时，其值设置为 `true` 的 HTTP 标头 `x-amz-sns-rawdelivery` 将添加到消息中，指示该消息已发布而没有 JSON 格式。
- 当您为 HTTP/S 终端节点启用原始消息传输时，将传输消息正文、客户端 IP 和所需的标头。当您指定消息属性时，将不会发送它。
- 当您为 Firehose 端点启用原始消息传输时，将传输消息正文。当您指定消息属性时，将不会发送它。

要使用 Amazon SDK 启用原始消息传送，必须使用 `SetSubscriptionAttribute` API 操作并将`RawMessageDelivery`属性的值设置为`true`。

利用 Amazon Web Services Management Console实现原始消息传输

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。

3. 在主题页面上，选择一个订阅了 Firehose、Amazon SQS 或 HTTP/S 端点的主题。
4. 在该***MyTopic***页面的订阅部分，选择订阅并选择编辑。
5. 在编辑***EXAMPLE1-23bc-4567-d890-ef12g3hij456***页面的详细信息部分，选择启用原始消息传送。
6. 选择 Save changes（保存更改）。

消息格式示例

在以下示例中，同一消息将发送到同一 Amazon SQS 队列两次。唯一的区别是第一条消息禁用原始消息传输，第二条消息则启用该传输。

- 原始消息传输已禁用

```
{
  "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"
}
```

- 原始消息传输已启用

```
This is a test message.
```

Amazon SQS 订阅的消息属性和原始消息传送

Amazon SNS 支持传输消息属性，从而让您提供与消息相关的结构化元数据项目（如时间戳、地理空间数据、签名和标识符）。对于 Amazon SQS 订阅，在启用原始消息传输时，最多可以发送 10 个消息属性。要发送 10 个以上的消息属性，您必须禁用“原始消息传输”。Amazon SNS 会丢弃所具有的消息属性（定向到启用了原始消息传输的 Amazon SQS 订阅）超过 10 个的消息，将其视为客户端错误。

将 Amazon SNS 消息发送到不同账户中的 Amazon SQS 队列

本文档介绍如何通过另一个账户中的一个或多个 Amazon SQS 队列订阅将通知发布到 Amazon SNS 主题。如果主题和队列在同一账户下，那么您可以采用相同方法设置主题和队列（参阅 将 Amazon SNS 通知扇出到 Amazon SQS 队列进行异步处理）。主要区别在于您处理订阅确认的方式，这取决于您如何为队列订阅主题。

最佳做法是尽可能遵循队列拥有者创建订阅部分中引用的步骤，因为当队列拥有者创建订阅时会自动进行确认。

> **Note**
>
> 如果 Amazon SQS 队列有大量消息，建议队列所有者创建订阅。

队列拥有者创建订阅

创建 Amazon SQS 队列的账户是队列拥有者。如果订阅由队列所有者创建，那么此订阅无需确认。一旦 `Subscribe` 操作完成后，队列即开始接收来自主题的通知。主题所有者必须提供队列所有者的账户权限，允许其对主题调用 `Subscribe` 操作，从而让队列所有者订阅主题所有者的主题。

步骤 1：使用 Amazon Web Services Management Console设置主题策略

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。
3. 选择一个主题，然后选择 Edit（编辑）。
4. 在 “编辑” ***MyTopic*** 页面上，展开 “访问策略” 部分。
5. 输入以下策略：

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

此策略授予账户 111122223333 对账户 123456789012 中的 MyTopic 调用 sns:Subscribe 的权限。

具有账户 111122223333 的凭证的用户可以订阅 MyTopic。此权限允许账户 ID 将权限委派给其 IAM 用户/角色。只有根账户或管理员用户才可以调用 sns:Subscribe。IAM 用户/角色还必须让 sns:subscribe 允许他们的队列订阅。

6. 选择 Save changes（保存更改）。

   拥有账户凭证的用户111122223333可以订阅 MyTopic。

第 2 步：要将 Amazon SQS 队列订阅添加到另一个 Amazon Web Services 账户 主题中的某个主题，请使用 Amazon Web Services Management Console

在开始之前，请确保您拥有主题和队列 ARNs 的权限，并且您已向该主题授予向队列发送消息的权限。

1. 登录 Amazon SQS 控制台。
2. 在导航窗格中，选择 Queues（队列）。
3. 从队列列表中，选择 queue（排队）以订阅 Amazon SNS 主题。
4. 选择 Subscribe to Amazon SNS topic（订阅 Amazon SNS 主题）。
5. 从 Specify an Amazon SNS topic available for this queue menu（指定可用于此队列菜单的 Amazon SNS 主题）中，选择队列的 Amazon SNS topic（Amazon SNS 主题）。
6. 选择 Enter Amazon SNS topic ARN（输入 Amazon SNS 主题 ARN），然后输入主题的 Amazon Resource Name (ARN)。

7. 选择保存。

> **Note**
>
> - 要能够与服务通信，队列必须具有 Amazon SNS 的权限。
> - 由于您是队列的所有者，因此您无需确认订阅。

非队列所有者用户创建订阅

创建订阅但不是队列所有者的任何用户都必须确认订阅。

使用 `Subscribe` 操作时，Amazon SNS 向队列发送订阅确认。订阅将显示在 Amazon SNS 控制台中，其订阅 ID 设置为等待确认。

要确认订阅，具有从队列中读取消息的权限的用户必须检索订阅确认 URL，而且订阅拥有者必须使用订阅确认 URL 以确认订阅。确认订阅前，向主题发布的通知不会发送至队列。要确认订阅，您可以使用 Amazon SQS 控制台或 `ReceiveMessage` 操作。

> **Note**
>
> 在为终端节点订阅主题之前，请通过为队列设置 `sqs:SendMessage` 权限来确保队列可以接收来自主题的消息。有关更多信息，请参阅 步骤 2. 为向 Amazon SQS 队列发送消息的 Amazon SNS 主题授予权限。

步骤 1：要将 Amazon SQS 队列订阅添加到另一个 Amazon Web Services 账户 主题中的某个主题，请使用 Amazon Web Services Management Console

在开始之前，请确保您拥有主题和队列 ARNs 的权限，并且您已向该主题授予向队列发送消息的权限。

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择订阅。
3. 在订阅页面上，选择创建订阅。
4. 在创建订阅页上的详细信息部分，执行以下操作：

   a. 对于 Topic ARN (主题 ARN)，输入主题的 ARN。

b. 对于 Protocol（协议），选择 Amazon SQS。

c. 对于 Endpoint（终端节点），输入队列的 ARN。

d. 选择 Create subscription（创建订阅）。

> **Note**
>
> - 要能够与服务通信，队列必须具有 Amazon SNS 的权限。

下面是允许 Amazon SNS 主题向 Amazon SQS 队列发送消息的示例策略声明。

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

第 2 步：使用确认订阅 Amazon Web Services Management Console

1. 登录 Amazon SQS 控制台。
2. 选择主题处于等待订阅阶段的队列。
3. 选择 Send and receive messages（发送和接收消息），然后选择 Poll for messages（轮询消息）。

   队列中会收到一条带有订阅确认的消息。
4. 在 Body（正文）列中，执行以下操作：

   a. 选择 More Details（更多详情）。

   b. 在 Message Details（消息详细信息）对话框中，找到并记下 SubscribeURL 值。这是您的订阅链接（下面的示例）。有关 API 令牌验证的更多详细信息，请参阅《Amazon SNS API 参考》中的 `ConfirmSubscription`。

```
https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

c. 记下订阅确认链接。必须将 URL 从队列拥有者传递给订阅拥有者。订阅拥有者必须在 Amazon SNS 控制台中输入此 URL。

5. 以订阅拥有者 身份登录 Amazon SNS 控制台。订阅拥有者执行确认。
6. 选择相关主题。
7. 在主题的订阅清单表格中选择相关订阅。它被标记为“Pending confirmation”（等待确认）。
8. 选择 Confirm subscription（确认订阅）。
9. 将出现一个提示订阅确认链接的模态框。粘贴 订阅确认链接。
10. 在模态框中选择 Confirm subscription（确认订阅）。

    将显示 XML 响应，例如：

```
<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>
```

    已订阅队列已准备好接收来自主题的消息。

11. （可选）如果您在 Amazon SNS 控制台中查看主题订阅，则可以看到等待确认消息已被 Subscription ID（订阅 ID）列中的订阅 ARN 取代。

如何强制订阅要求对取消订阅请求进行身份验证？

订阅拥有者必须在订阅确认后将 `AuthenticateOnUnsubscribe` 标志设置为 true。

- 如果订阅由队列拥有者创建，则 `AuthenticateOnUnsubscribe` 自动设置为 true。
- 在没有身份验证的情况下导航到订阅确认链接时，无法将 `AuthenticateOnUnsubscribe` 设置为 true。

将 Amazon SNS 消息发送到不同区域中的 Amazon SQS 队列或 Amazon Lambda 函数

Amazon SNS 支持跨区域传输，无论是默认启用的区域还是可选择加入的区域。有关 Amazon SNS 支持的 Amazon 区域的当前列表（包括选择加入的区域），请参阅《Amazon Web Services 一般参考》中的 Amazon Simple Notification Service 端点和限额。

Amazon SNS 支持跨区域传输通知到 Amazon SQS 队列以及 Amazon Lambda 函数。当其中一个区域是选择加入的区域时，您必须在订阅资源的策略中指定不同的 Amazon SNS 服务委托人。

Amazon SNS 订阅命令必须在相应区域中托管 Amazon SNS 的区域内执行。例如，如果 Amazon SNS 位于 us-east-1 区域的“A”账户中，而 Lambda 函数位于 us-east-2 区域的“B”账户中，则必须在 us-east-1 区域的“A”账户中执行订阅 CLI 命令。

选择加入的区域

Amazon SNS 支持以下选择加入的区域：

区域名称	区域
非洲（开普敦）区域	af-south-1
亚太地区（香港）区域	ap-east-1
亚太地区（海得拉巴）区域	ap-south-2
亚太地区（雅加达）区域	ap-southeast-3
亚太地区（墨尔本）区域	ap-southeast-4
欧洲地区（米兰）	eu-south-1
欧洲地区（西班牙）区域	eu-south-2
欧洲（苏黎世）	eu-central-2
以色列（特拉维夫）区域	il-central-1
中东（巴林）区域	me-south-1

区域名称	区域
中东（阿联酋）区域	me-central-1

有关启用可选区域的信息，请参阅中的Amazon Web Services 一般参考管理 Amazon 区域。

使用 Amazon SNS 将消息从选择加入的区域传输到默认启用的区域时，必须更改为队列创建的资源策略。将委托人 `sns.amazonaws.com` 替换为 `sns.<opt-in-region>.amazonaws.com`。例如：

- 要为美国东部（弗吉尼亚州北部）的 Amazon SQS 队列订阅亚太地区（香港）的 Amazon SNS 主题，请将队列策略中的主体更改为 `sns.ap-east-1.amazonaws.com`。选择加入区域包括 2019 年 3 月 20 日之后推出的任何区域，包括亚太地区（香港）、亚太地区（雅加达）、中东（巴林）、欧盟（米兰）和非洲（开普敦）。2019 年 3 月 20 日之前推出的区域默认情况下处于启用状态。

  对于 Amazon SQS 的跨区域传输支持

跨区域传输类型	支持/不支持
原定设置启用的区域至选择加入区域	在队列的服务主体中使用 `sns.<opt-in-region>.amazonaws.com` 提供支持
选择加入区域至原定设置启用的区域	在队列的服务主体中使用 `sns.<opt-in-region>.amazonaws.com` 提供支持
选择加入区域至选择加入区域	不支持

以下是访问策略声明的示例，该声明允许选择加入区域 (af-south-1) 中的亚马逊 SNS 主题发送到某个区域 (us-east-1) 中的亚马逊 SQS 队列。enabled-by-default 它在路径 `Statement/Principal/Service` 下包含必要的区域化服务主体配置。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.af-south-1.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
        }
      }
    },
    ...
  ]
}
```

- 要为美国东部（弗吉尼亚北部）的 Amazon Lambda 函数订阅亚太地区（香港）的 Amazon SNS 主题，请将 Amazon Lambda 功能策略中的主体更改为。`sns.ap-east-1.amazonaws.com`选择加入区域包括 2019 年 3 月 20 日之后推出的任何区域，包括亚太地区（香港）、亚太地区（雅加达）、中东（巴林）、欧盟（米兰）和非洲（开普敦）。2019 年 3 月 20 日之前推出的区域默认情况下处于启用状态。

  跨区域配送支持 Amazon Lambda

跨区域传输类型	支持/不支持
原定设置启用的区域至选择加入区域	不支持
选择加入区域至原定设置启用的区域	在 Lambda 函数的服务主体中使用 `sns.<opt-in-region>.amazonaws.com` 提供支持
选择加入区域至选择加入区域	不支持

Amazon SNS 消息传输状态

Amazon SNS 支持使用以下 Amazon SNS 终端节点记录发送到主题的通知消息的发送状态：

- Amazon Data Firehose
- Amazon Simple Queue Service
- Amazon Lambda
- HTTPS
- 平台应用程序终端节点

发送状态日志将发送到 Amazon CloudWatch Logs，从而提供对消息传输操作的见解。这些日志可以帮助你：

- 确定消息是否成功传送到端点。
- 识别终端节点对 Amazon SNS 的响应。
- 测量消息停留时间（从发布时间戳到移交到端点之间的时间）。

您可以使用 Amazon Web Services Management Console、 Amazon SDKs、查询 API 或配置传送状态日志记录 Amazon CloudFormation。

记录交付状态的先决条件

本主题概述了允许 Amazon SNS 向其写入传输日志所必需的 IAM 权限， CloudWatch 并解释了默认日志组命名约定。这可确保您拥有正确的设置和访问权限，以监控和分析日志中的消息传送 CloudWatch 日志。

所需的 IAM 权限

为记录交付状态而附加的 IAM 角色必须包含以下权限，才能让 Amazon SNS 写入 CloudWatch 日志。您可以使用具有这些权限的现有角色或在设置过程中创建新角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
```

```
    }
  ]
}
```

日志组命名惯例

默认情况下，Amazon SNS 使用以下命名约定为传输状态 CloudWatch 日志创建日志组。该组中的日志流对应于终端节点协议（例如 Lambda、Amazon SQS）。确保您有权在日志控制台中查看这些 CloudWatch 日志。

```
sns/<region>/<account-id>/<topic-name>
```

使用 Amazon Web Services Management Console配置传输状态日志记录

本主题介绍如何为 Amazon SNS 主题启用消息传输状态日志，包括配置日志设置、分配 IAM 角色以及验证日志是否捕获传输 CloudWatch 日志以进行监控和故障排除。

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。
3. 选择所需的主题，然后选择 “编辑”。
4. 展开 “配送状态记录” 部分。
5. 选择要为其启用日志记录的协议（例如，HTTP、Lambda、Amazon SQS）。
6. 输入成功采样率，即您希望接收 CloudWatch 日志的成功消息的百分比。
7. 在 IAM 角色部分，您必须为成功和失败日志配置角色：
   - 使用现有服务角色-选择具有 Amazon SNS 写入日志所需权限的现有 IAM 角色。 CloudWatch
   - 创建新的服务角色-选择 “创建新角色”，在 IAM 控制台中为成功和失败的交付定义 IAM 角色。有关权限的详细信息，请参阅记录交付状态的先决条件。
8. 选择保存更改。

   启用日志记录后，您可以查看和解析包含消息传送状态的 CloudWatch 日志。有关使用的更多信息 CloudWatch，请参阅CloudWatch文档。

正在验证日志设置

1. 登录 CloudWatch 日志控制台。

2. 找到名为的日志组`sns/<region>/<account-id>/<topic-name>`。
3. 确保已配置的端点协议存在日志流。
4. 向您的主题发送测试消息，并确认日志条目是否出现，指示成功或失败。

使用配置传送状态日志 Amazon SDKs

Amazon SDKs 提供了多种 APIs 语言来设置消息传送状态记录的主题属性。例如，使用 SetTopicAttributesAPI 来配置：

- `LambdaSuccessFeedbackRoleArn`— 用于成功向 Lambda 终端节点传输消息的 IAM 角色。
- `LambdaSuccessFeedbackSampleRate`— 成功发送到 Lambda 终端节点的消息的采样率。
- `LambdaFailureFeedbackRoleArn`— 用于向 Lambda 终端节点传送失败消息的 IAM 角色。

Amazon CLI 命令示例

```
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
```

主题属性

使用以下主题属性名称值表示消息传送状态：

HTTP

- `HTTPSuccessFeedbackRoleArn`— 订阅了 HTTP 终端节点的 Amazon SNS 主题的成功消息传输状态。
- `HTTPSuccessFeedbackSampleRate`— 订阅了 HTTP 终端节点的 Amazon SNS 主题成功采样消息的百分比。
- `HTTPFailureFeedbackRoleArn`— 订阅了 HTTP 终端节点的 Amazon SNS 主题的消息传输失败状态。

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— 订阅了亚马逊 Kinesis Data Firehose 终端节点的亚马逊 SNS 主题的成功消息传输状态。

- `FirehoseSuccessFeedbackSampleRate`— 订阅亚马逊 Kinesis Data Firehose 终端节点的亚马逊 SNS 主题成功采样消息的百分比。
- `FirehoseFailureFeedbackRoleArn`— 订阅了亚马逊 Kinesis Data Firehose 终端节点的亚马逊 SNS 主题的消息传输失败状态。

Amazon Lambda

- `LambdaSuccessFeedbackRoleArn`— 订阅了 Lambda 终端节点的 Amazon SNS 主题的成功消息传输状态。
- `LambdaSuccessFeedbackSampleRate`— 订阅 Lambda 终端节点的 Amazon SNS 主题成功采样消息的百分比。
- `LambdaFailureFeedbackRoleArn`— 订阅了 Lambda 终端节点的 Amazon SNS 主题的消息传输失败状态。

平台应用程序端点

- `ApplicationSuccessFeedbackRoleArn`— 订阅应用程序终端节点的 Amazon SNS 主题的成功消息传输状态。 Amazon
- `ApplicationSuccessFeedbackSampleRate`— 订阅应用程序终端节点的 Amazon SNS 主题成功采样消息的百分比。 Amazon
- `ApplicationFailureFeedbackRoleArn`— 订阅应用程序终端节点的 Amazon SNS 主题的消息传输失败状态。 Amazon

> **Note**
>
> 此外，您可以将应用程序属性配置为将传送状态直接记录到推送通知服务。有关更多信息，请参阅使用用于消息传输状态的 Amazon SNS 应用程序属性。

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— 订阅亚马逊 SQS 终端节点的 Amazon SNS 主题的成功消息传送状态。
- `SQSSuccessFeedbackSampleRate`— 订阅亚马逊 SQS 终端节点的 Amazon SNS 主题成功采样消息的百分比。

- `SQSFailureFeedbackRoleArn`— 订阅亚马逊 SQS 终端节点的 Amazon SNS 主题的消息传送失败状态。

平台应用程序终端节点的日志与其他终端节点一样写入同一个 CloudWatch 日志组。

> **Note**
>
> `<ENDPOINT>SuccessFeedbackRoleArn`和`<ENDPOINT>FailureFeedbackRoleArn`属性用于向 Amazon SNS 授予代表您使用 CloudWatch 日志的写入权限。`<ENDPOINT>SuccessFeedbackSampleRate` 属性用于指定成功传输消息的采样率百分比 (0-100)。配置该`<ENDPOINT>FailureFeedbackRoleArn`属性后，所有失败的消息传送都会生成 CloudWatch 日志。

Amazon 用于配置主题属性的 SDK 示例

以下代码示例演示如何使用 `SetTopicAttributes`。

CLI

Amazon CLI

为主题设置属性

以下 `set-topic-attributes` 示例为指定主题设置 `DisplayName` 属性。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetTopicAttributes中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SetTopicAttributes中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考SetTopicAttributes中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- 有关 API 的详细信息，请参阅适用SetTopicAttributes于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考SetTopicAttributes中的。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考SetTopicAttributes中的。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用SetTopicAttributes于 S AP 的Amazon SDK ABAP API 参考。

使用 Amazon CloudFormation配置传输状态日志记录

要`DeliveryStatusLogging`使用进行配置 Amazon CloudFormation，请使用 JSON 或 YAML 模板创建 Amazon CloudFormation 堆栈。有关更多信息，请参阅《 Amazon CloudFormation 用户指南》中的`AWS::SNS::Topic`资源`DeliveryStatusLogging`属性。以下是 JSON 和

YAML Amazon CloudFormation 模板的示例，这些模板用于创建新主题或使用 Amazon SQS `DeliveryStatusLogging` 协议的所有属性更新现有主题。

确保中引用的 IAM 角色`SuccessFeedbackRoleArn`并`FailureFeedbackRoleArn`具有所需的 CloudWatch 日志权限。

JSON

```
"Resources": {
    "MySNSTopic" : {
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "TestTopic",
            "DisplayName" : "TEST",
            "SignatureVersion" : "2",
            "DeliveryStatusLogging" : [{
                "Protocol": "sqs",
                "SuccessFeedbackSampleRate": "45",
                "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
                "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
            }]
        }
    }
}
```

YAML

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName:TestTopic
      DisplayName:TEST
      SignatureVersion:2
      DeliveryStatusLogging:
       - Protocol: sqs
         SuccessFeedbackSampleRate: 45
         SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
         FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Amazon SNS 消息传输重试

Amazon SNS 为每个传输协议定义了一个传输策略。传输策略定义了在发生服务器端错误时（当承载已订阅终端节点的系统变得不可用时），Amazon SNS 如何重试消息传输。当传输策略用尽时，Amazon SNS 将停止重试传输并丢弃邮件——除非已将死信队列附加到订阅。有关更多信息，请参阅 Amazon SNS 死信队列。

传输协议和策略

> **Note**
>
> - HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S支持自定义策略除外。请参阅创建 HTTP/S 传输策略。
> - Amazon SNS 将抖动应用于传输重试。有关更多信息，请参阅发布在 Amazon 架构博客上的指数回退和抖动博客文章。
> - HTTP/S 端点的策略重试总时间不能超过 3600 秒。这是一项硬性限制，无法增加。

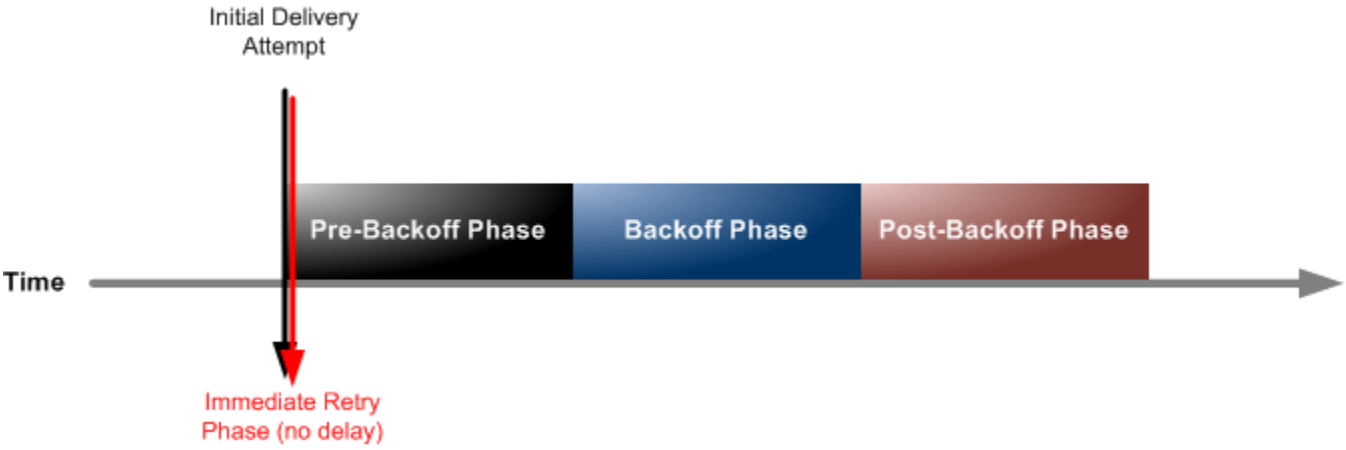
端点类型	传输协议	立即重试（无延迟）阶段	前退避阶段	退避阶段	后退避阶段	总尝试次数
Amazon 托管端点	Amazon Data Firehose[1]	3 次，无延迟	2 次，相隔 1 秒	10 次，带指数退避（1 秒到 20 秒）	100000 次，相隔 20 秒	100015 次，超过 23 天
	Amazon Lambda					
	Amazon SQS					
客户托管的终端节点	SMTP	0 次，无延迟	2 次，相隔 10 秒		38 次，相隔 600 秒	

端点类型	传输协议	立即重试（无延迟）阶段	前退避阶段	退避阶段	后退避阶段	总尝试次数
	短信 移动推送			10 次，带指数退避（10 秒到 600 秒（10 分钟））	（10 分钟）	50 次尝试，超过 6 小时

[1] 对于 Firehose 协议的节流错误，Amazon SNS 使用与客户托管端点相同的传输策略。

传输策略阶段

下图显示了传输策略的各个阶段。



每个传输策略包含四个阶段。

1. 立即重试阶段（无延迟）— 此阶段在首次尝试交付后立即出现。在该阶段中重试之间没有延迟。
2. 退避前阶段 — 此阶段遵循立即重试阶段。Amazon SNS 使用此阶段进行一系列重试，然后再应用退避函数。此阶段指定重试次数以及它们之间的延迟时间量。
3. 退避阶段 — 此阶段使用重试退避功能控制两次重试之间的延迟。此阶段设置了最短延迟时间、最长延迟时间和重试-退避函数，该函数定义了延迟时间从最小值增加到最大值的速度。退避函数可以是算术、指数、几何或线性的。
4. 退避后阶段 — 此阶段遵循退避阶段。此阶段指定重试次数以及它们之间的延迟时间量。它是最后一个阶段。

创建 HTTP/S 传输策略

您可以使用包含四个阶段的传输策略来定义 Amazon SNS 如何重试向 HTTP/S 终端节点传送消息：无延迟、退避和后退缩。此策略允许您覆盖默认的重试设置，并对其进行自定义以匹配您的 HTTP 服务器的容量。

您可以在主题或订阅级别将 HTTP/S 传输策略定义为 JSON 对象：

- 主题级政策-适用于链接到该主题的所有 HTTP/S 订阅。使用`CreateTopic`或`SetTopicAttributes`API 操作来设置此政策。
- 订阅级别政策-仅适用于特定的订阅。使用`Subscribe`或 `SetSubscriptionAttributes`API 操作来设置此政策。

或者，您也可以在 Amazon CloudFormation 模板中使用该AWS::SNS::Subscription资源。

您应该根据 HTTP/S 服务器的容量自定义您的传输策略：

- 所有订阅使用单台服务器-如果主题中的所有 HTTP/S 订阅都使用同一台服务器，请将传输策略设置为主题属性，以确保所有订阅之间的一致性。
- 不同的订阅服务器-如果订阅针对不同的服务器，则为每个订阅创建针对特定服务器容量的唯一交付策略。

您还可以在请求策略中设置`Content-Type`标题以指定通知的媒体类型。默认情况下，Amazon SNS 会将所有通知发送到内容类型设置为 HTTP/S 终端节点。`text/plain; charset=UTF-8`但是，您可以使用请求策略中的`headerContentType`字段来覆盖此默认值。

以下 JSON 对象定义了分为四个阶段的重试次数的交付策略：

1. 无延迟阶段-立即重试 3 次。
2. 退避前阶段-以 1 秒的间隔重试 2 次。
3. 退避阶段 — 重试 10 次，指数延迟范围为 1 到 60 秒。
4. 后退阶段 — 以 60 秒的固定间隔重试 35 次。

在丢弃一条消息之前，Amazon SNS 总共尝试传送该消息 50 次。要保留所有重试后都无法传送的消息，请将您的订阅配置为将无法送达的消息移至死信队列 (DLQ)。有关更多信息，请参阅 Amazon SNS 死信队列。

Amazon SNS 将所有 5XX 错误和 429（发送的请求过多）错误视为可重试。这些错误受配送政策的约束。所有其他错误都被视为永久失败，不会尝试重试。

> **Note**
>
> 此传输策略使用该`maxReceivesPerSecond`属性将每次订阅的传送流量限制为平均每秒 10 条消息。虽然这种机制有助于防止您的 HTTP/S 端点被高流量淹没，但它旨在保持平均传输速率，并且不强制执行严格的上限。偶尔可能会出现超过指定限制的配送流量峰值，尤其是在您的发布速率明显高于限制的情况下。
>
> 当发布（入站）流量超过传送（出站）速率时，可能会导致消息积压和更高的交付延迟。为避免此类问题，请确保该`maxReceivesPerSecond`值与 HTTP/S 服务器的容量和工作负载要求保持一致。

以下传送策略示例覆盖了 HTTP/S 通知的默认内容类型。`application/json`

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

传输策略由重试策略、限制策略和请求策略组成。配送策略中总共有 9 个属性。

策略	描述	限制
`minDelayTarget`	重试的最短延迟时间。	1 至最长延迟时间

策略	描述	限制
	单位：秒	原定设置值：20
maxDelayTarget	重试的最长延迟时间。 单位：秒	最短延迟时间至 3600 原定设置值：20
numRetries	重试总数，包括立即重试、前退避重试、退避重试和后退避重试。	0 至 100 原定设置值：3
numNoDelayRetries	要立即完成的重试次数，各个重试之间无延迟。	0 或更多 原定设置值：0
numMinDelayRetries	前退避阶段的重试次数，各个重试之间有指定的最短延迟时间。	0 或更多 原定设置值：0
numMaxDelayRetries	后退避阶段的重试次数，各个重试之间有最长延迟时间。	0 或更多 原定设置值：0
backoffFunction	各个重试之间退避的模型。	四个选项之一： • 算术 • 指数 • 几何 • 线性 默认：线性
maxReceivesPerSecond	每次订阅每秒的最大平均消息传送数。	1 或更多 默认：无限制（无配送费率限制）

策略	描述	限制
headerContentType	发送到 HTTP/S 端点的通知的内容类型。	如果未定义请求策略，则内容类型原定设置为 text/plain; charset=UTF-8 。 当为订阅禁用原始消息传送时（原定设置），或者在主题级别定义传送策略时，支持的标头内容类型为 application/json 和 text/plain 。 为订阅启用原始消息传送时，支持以下内容类型： • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • 应用程序/ x-www-form-urlencoded • application/xhtml+xml • application/xml

Amazon SNS 使用以下公式计算退避阶段的重试次数：

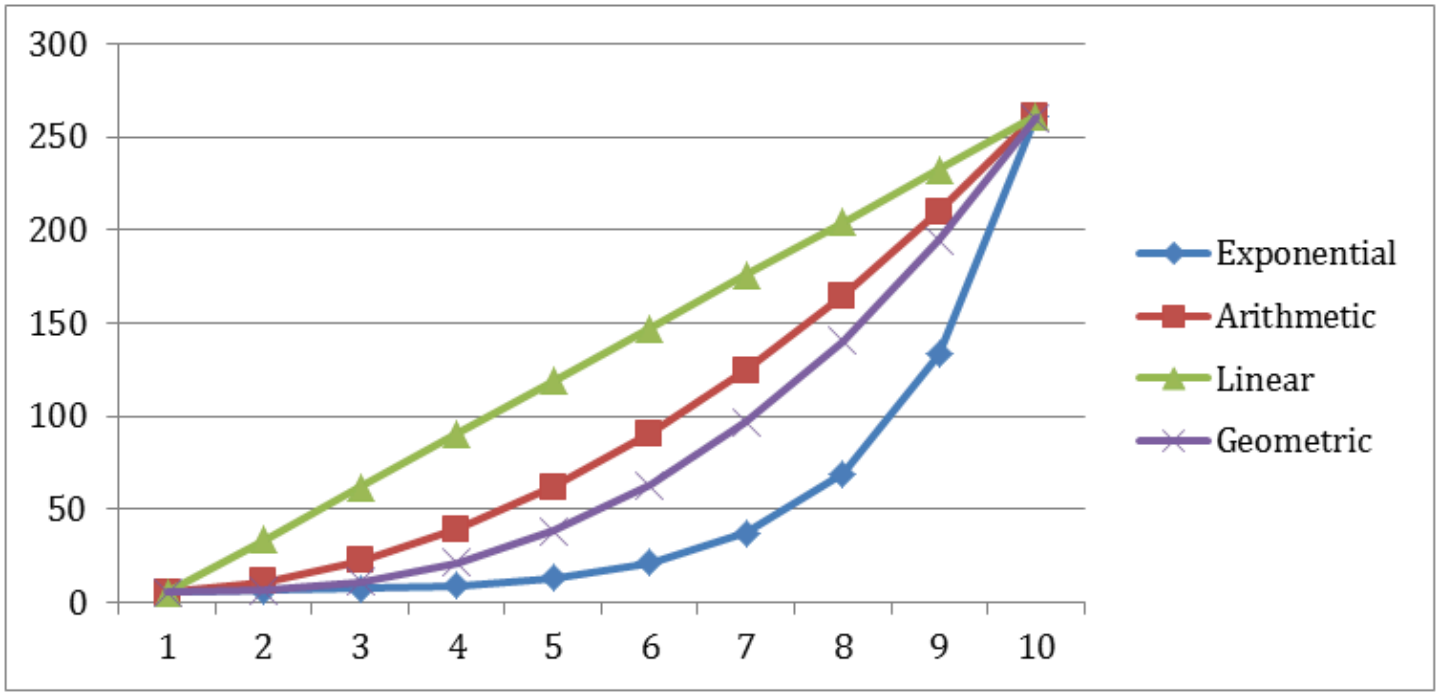
```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

您可以使用三个参数控制退避阶段的重试频率：

- **minDelayTarget**— 设置退避阶段第一次重试尝试的延迟。
- **maxDelayTarget**— 设置退避阶段最后一次重试尝试的延迟。
- **backoffFunction**— 确定 Amazon SNS 用于计算第一次和最后一次重试之间所有重试尝试的延迟时长的算法。您可以从四个可用的重试退避功能中进行选择。

下图说明了在退避阶段，不同的重试退避函数如何影响两次重试之间的延迟。本示例中使用的传输策略包括以下设置：总重试次数 10 次，最小延迟为 5 秒，最大延迟为 260 秒。

- 垂直轴显示每次重试尝试的延迟（以秒为单位）。
- 水平轴代表重试顺序，范围从第一次尝试到第十次尝试。



Amazon SNS 死信队列

死信队列是一个 Amazon SQS 队列，Amazon SNS 订阅可以将其视为目标，用于存放无法成功传输给订阅用户的消息。由于客户端错误或服务器错误而无法传输的消息将保留在死信队列中，以进行进一步分析或重新处理。有关更多信息，请参阅为订阅配置 Amazon SNS 死信队列 和Amazon SNS 消息传输重试。

> **Note**
>
> - 亚马逊 SNS 订阅和亚马逊 SQS 队列必须位于 Amazon 相同的账户和区域下。
> - 对于 FIFO 主题，您可以将 Amazon SQS 队列用作 Amazon SNS 订阅的死信队列。FIFO 主题订阅使用 FIFO 队列，而标准主题订阅使用标准队列。
> - 要将加密的 Amazon SQS 队列用作死信队列，您必须使用自定义 KMS，其密钥策略允许亚马逊 SNS 服务委托人访问 API 操作。 Amazon KMS 有关更多信息，请参阅本指南中的 使用服务器端加密保护 Amazon SNS 数据安全 以及 Amazon Simple Queue Service 开发人员指南中的使用服务器端加密 (SSE) 和 Amazon KMS保护 Amazon SQS 数据。

为什么消息传输会失败？

通常，当 Amazon SNS 由于客户端或服务器端错误无法访问订阅的终端节点时，消息传输将失败。当 Amazon SNS 收到客户端错误，或者继续收到超出相应重试策略指定的重试次数的消息的服务器端错误时，Amazon SNS 会丢弃该消息——除非已将死信队列附加到订阅。失败的传输不会更改您的订阅状态。有关更多信息，请参阅 Amazon SNS 消息传输重试。

客户端错误

当 Amazon SNS 具有过时的订阅元数据时，可能会发生客户端错误。当拥有者删除终端节点（例如，订阅了 Amazon SNS 主题的 Lambda 函数）时，或当拥有者更改附加到已订阅终端节点的策略的方式致使 Amazon SNS 无法将消息传输到终端节点时，通常会发生这些错误。Amazon SNS 不会重试因客户端错误导致失败的消息传输。

服务器端错误

当负责已订阅终端节点的系统变得不可用或返回一个指示该系统无法处理来自 Amazon SNS 的有效请求的异常时，可能会发生服务器端错误。在发生服务器端错误时，Amazon SNS 会使用线性或指数回退函数重试失败的传输。对于由亚马逊 SQS Amazon Lambda支持的 Amazon 托管终端节点导致的服务器端错误，Amazon SNS 在 23 天内重试交付最多 100,015 次。

客户托管的终端节点（如 HTTP、SMTP、SMS 或移动推送）也可能会导致出现服务器端错误。Amazon SNS 也将重试传输到这些类型的终端节点。虽然 HTTP 终端节点支持客户定义的重试策略，但 Amazon SNS 会将 SMTP、SMS 和移动推送终端节点的内部传输重试策略设置为 50 次（时间超过 6 小时）。

死信队列的工作方式

死信队列将附加到 Amazon SNS 订阅（而不是主题），因为消息传输是在订阅级别进行的。这使您能够更轻松地识别每条消息的原始目标终端节点。

与 Amazon SNS 订阅关联的死信队列是普通 Amazon SQS 队列。有关消息保留期的更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中与消息相关的配额。您可以使用 Amazon SQS `SetQueueAttributes` API 操作来更改消息保留策略。为了使您的应用程序更具弹性，我们建议将死信队列的最长保留期设置为 14 天。

如何将消息移至死信队列中？

可以使用重新驱动策略 将您的消息移至死信队列中。重新驱动策略是一个引用死信队列 ARN 的 JSON 对象。`deadLetterTargetArn` 属性指定 ARN。ARN 必须指向与您的亚马逊 SNS 订阅位于 Amazon Web Services 账户 相同和区域的亚马逊 SQS 队列。有关更多信息，请参阅 为订阅配置 Amazon SNS 死信队列。

以下 JSON 对象是附加到 SNS 订阅的示例重新驱动策略。

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

如何将消息移出死信队列？

可以通过两种方式将消息移出死信队列：

- 避免编写 Amazon SQS 使用者逻辑 – 将死信队列设置为 Lambda 函数的事件源以耗尽死信队列。
- 编写 Amazon SQS 使用者逻辑 — 使用 Amazon SQS API Amazon 、SDK Amazon CLI 或编写自定义使用者逻辑，用于轮询、处理和删除死信队列中的消息。

如何监控和记录死信队列？

您可以使用亚马逊 CloudWatch 指标来监控与您的 Amazon SNS 订阅相关的死信队列。所有 Amazon SQS 队列每隔一分钟发布一次 CloudWatch 指标。有关更多信息，请参阅《亚马逊简单队列服务开发者指南》中的 Amazon SQS 可用 CloudWatch 指标。所有带有死信队列的 Amazon SNS 订阅也会发布指标。 CloudWatch 有关更多信息，请参阅 使用监控亚马逊 SNS 主题 CloudWatch。

要收到死信队列中的活动通知，您可以使用 CloudWatch 指标和警报。为 `NumberOfMessagesSent` 指标设置警报是不合适的，因为该指标不会捕获因处理尝试失败而发送到 DLQ 的消息。取而代之的是使用 `ApproximateNumberOfMessagesVisible` 指标，它会捕获 DLQ 中当前可用的所有消息，包括因处理失败而移动的消息。

CloudWatch 警报设置示例

1. 为该**`ApproximateNumberOfMessagesVisible`**指标创建CloudWatch警报。
2. 将警报阈值设置为 1（或基于您的期望值和 DLQ 流量的另一个适当值）。
3. 指定在警报消失时要通知的 Amazon SNS 主题。此 Amazon SNS 主题可以将告警通知传输到任何终端节点类型（例如，电子邮件地址、电话号码或移动寻呼机应用程序）。

您可以使用 CloudWatch 日志来调查导致任何 Amazon SNS 交付失败的异常，以及将消息发送到死信队列。Amazon SNS 可以记录成功和失败的交付。 CloudWatch有关更多信息，请参阅 Amazon SNS 移动应用程序属性。

为订阅配置 Amazon SNS 死信队列

死信队列是一个 Amazon SQS 队列，Amazon SNS 订阅可以将其视为目标，用于存放无法成功传输给订阅用户的消息。由于客户端错误或服务器错误而无法传输的消息将保留在死信队列中，以进行进一步分析或重新处理。有关更多信息，请参阅Amazon SNS 死信队列 和Amazon SNS 消息传输重试。

本页介绍如何使用、 Amazon 软件开发工具包 Amazon Web Services Management Console Amazon CLI、和 Amazon CloudFormation 为 Amazon SNS 订阅配置死信队列。

> **Note**
>
> 对于 FIFO 主题，您可以将 Amazon SQS 队列用作 Amazon SNS 订阅的死信队列。FIFO 主题订阅使用 FIFO 队列，而标准主题订阅使用标准队列。

先决条件

在配置死信队列之前，请完成以下先决条件：

1. 创建名为 `MyTopic` 的 Amazon SNS 主题。
2. 创建名为 `MyEndpoint` 的 Amazon SQS 队列，以用作 Amazon SNS 订阅的终端节点。

3. （跳过 Amazon CloudFormation）在队列中订阅主题。
4. 创建另一个名为 `MyDeadLetterQueue` 的 Amazon SQS 队列，以用作 Amazon SNS 订阅的死信队列。
5. 要向 Amazon SNS 委托人授予对 Amazon SQS API 操作的访问权限，请为 `MyDeadLetterQueue` 设置以下队列策略。

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

要为 Amazon SNS 订阅配置死信队列，请使用 Amazon Web Services Management Console

在开始本教程之前，请确保完成先决条件。

1. 登录 Amazon SQS 控制台。
2. 创建 Amazon SQS 队列或使用现有队列，并在队列的 Details（详细信息）选项卡上记下队列的 ARN，例如：

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

3. 登录 Amazon SNS 控制台。
4. 在导航面板中，选择订阅。
5. 在 Subscriptions (订阅) 页面上，选择现有订阅，然后选择 Edit (编辑)。

6. 在 Edit ***1234a567-bc89-012d-3e45-6fg7h890123i*** (编辑) 页面上，展开 Redrive policy (dead-letter queue) (重新驱动策略（死信队列）) 部分，然后执行以下操作：

   a. 选择 Enabled (已启用)。

   b. 指定 Amazon SQS 队列的 ARN。

7. 选择 Save changes（保存更改）。

   您的订阅将配置为使用死信队列。

使用软件开发工具包为 Amazon SNS 订阅配置死信队列 Amazon

在您运行此示例之前，请确保完成先决条件。

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置Amazon SDKs 和凭据文件。

以下代码示例演示如何使用 `SetSubscriptionAttributesRedrivePolicy`。

Java

适用于 Java 的 SDK 1.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
// 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);
```

要为 Amazon SNS 订阅配置死信队列，请使用 Amazon CLI

在开始本教程之前，请确保完成先决条件。

1. 安装和配置 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南。
2. 使用以下命令。

```
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\"}"
```

使用为 Amazon SNS 订阅配置死信队列 Amazon CloudFormation

在开始本教程之前，请确保完成先决条件。

1. 将以下 JSON 代码复制到名为 `MyDeadLetterQueue.json` 的文件中。

```
{
  "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. 登录 Amazon CloudFormation 控制台。
3. 在 Select Template (选择模板) 页面上，选择 Upload a template to Amazon S3 (将模板上传到 Amazon S3)，再选择您的 `MyDeadLetterQueue.json` 文件，然后选择 Next (下一步)。
4. 在 Specify Details（指定详细信息）页面上，为 Stack Name（堆栈名称）输入 `MyDeadLetterQueue`，然后选择 Next（下一步）。
5. 在选项页面上，选择下一步。
6. 在 Review 页面上，选择 Create 。

   Amazon CloudFormation 开始创建`MyDeadLetterQueue`堆栈并显示 CREATE_IN_ PROGRESS 状态。该过程完成后， Amazon CloudFormation 将显示 “创建_完成” 状态。

Amazon SNS 消息归档、重播和分析

Amazon SNS 标准主题支持通过 Amazon Data Firehose 进行消息归档。您可以扇出通知到 Firehose 传输流，它允许您将通知发送到 Firehose 支持的存储和分析目标，包括 Amazon Simple Storage Service（Amazon S3）、Amazon Redshift 等。

Amazon SNS FIFO 主题支持就地、无代码的消息归档，可让主题所有者存储（或归档）发布到主题的消息长达 365 天。对于具有活动 `ArchivePolicy` 的主题，订阅用户可以创建 `ReplayPolicy`，以将归档的消息检索（或重播）回订阅的端点。要了解有关此特征的更多信息，请参阅FIFO 主题的 Amazon SNS 消息归档与重播功能。

特征	标准主题	FIFO 主题
消息归档	扇出到 Firehose 传输流	适用于 FIFO 主题所有者的 Amazon SNS 消息归档
消息重播	标准主题的重播不是内置特征。许多客户根据自己的消息归档来构建自己的重播。	FIFO 主题订阅用户的 Amazon SNS 消息重播

Amazon SNS 中的资源管理和优化

本主题提供指导，说明如何通过确保最佳性能、减少不必要的成本和维护井井有条的资源来充分利用 Amazon SNS 的潜力。

主题

- Amazon SNS 主题标记

Amazon SNS 主题标记

Amazon SNS 支持对 Amazon SNS 主题进行标记。这可以帮助您跟踪和管理与主题相关的费用，增强 Amazon 身份和访问管理 (IAM) 策略的安全性，并允许您轻松搜索或筛选成千上万个主题。通过添加标签，您可以使用 Resource G Amazon roups 管理您的 Amazon SNS 主题。有关资源组的更多信息，请参阅Amazon 资源组用户指南。

成本分配的标记

要组织并标识您的 Amazon SNS 主题以进行成本分配，您可以添加用于标识主题目标的标签。这在您拥有许多主题时尤其有用。您可以使用成本分配标签来整理 Amazon 账单，以反映您自己的成本结构。为此，请注册以获取包含标签密钥和值的 Amazon 账户账单。有关更多信息，请参阅 Amazon 账单和成本管理用户指南中的设置月度成本分配报告。

例如，您可以添加表示 Amazon SNS 主题的成本中心和用途的标签，如下所示：

资源	键	值
主题 1	成本中心	43289
	应用程序	订单处理
主题 2	成本中心	43289
	应用程序	支付处理
主题 3	成本中心	76585
	应用程序	存档

此标记方案可让您将执行相关任务的两个主题分组到同一成本中心，并使用不同的成本分配标签来标记不相关的活动。

访问控制的标记

Amazon Identity and Access Management 支持根据标签控制对资源的访问权限。在标记资源之后，请在 IAM 策略的条件元素中提供与资源标签相关的信息，以管理基于标签的访问。有关如何使用 Amazon SNS 控制台或 Amazon SDK 标记资源的信息，请参阅配置标签。

您可以限制 IAM 身份的访问。例如，您可以限制对包含键 `environment` 和值 `production` 的标签的所有 Amazon SNS 主题的 `Publish` 和 `PublishBatch` 访问，同时允许访问所有其他 Amazon SNS 主题。在下面的示例中，该策略限制了将消息发布到标记为的主题的能力`production`，同时允许将消息发布到标记为的主题`development`。有关更多信息，请参阅《IAM 用户指南》中的使用标签控制访问。

> **Note**
>
> 为 `Publish` 和 `PublishBatch` 设置 `Publish` 集权限的 IAM 权限。

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

进行标记以便进行资源搜索和筛选

一个 Amazon 账户可以有成千上万个亚马逊 SNS 主题（详情请参阅亚马逊 SNS 配额）。通过标记主题，您可以简化搜索或筛选主题的过程。

例如，您可以拥有数百个与您的生产环境相关的主题。您可以通过给定标签查询所有主题，而不必手动搜索这些主题：

```
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());
    }
```

```
}
```

配置 Amazon SNS 主题标签

本主题介绍如何使用 Amazon Web Services Management Console、软件开发工具包或，为 Amazon Amazon SNS 主题配置标签。 Amazon CLI

> **Important**
>
> 请勿在标签中添加个人身份信息（PII）或其他机密或敏感信息。标签可供许多其他亚马逊云科技访问，包括计费。标签不适合用于私有或敏感数据。

使用 Amazon SNS 主题列出、添加和移除标签 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。
3. 在主题页面上，选择主题，然后选择编辑。
4. 展开标签部分。

   将列出添加到主题的标签。
5. 修改主题标签：
   - 要添加标签，请选择 Add tag（添加标签），然后输入 Key（键）和 Value（值）（可选）。
   - 要删除标签，请选择键值对旁边的删除标签。
6. 选择 Save changes（保存更改）。

使用 Amazon SDK 将标签添加到主题

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅和工具参考指南中的共享配置Amazon SDKs 和凭据文件。

以下代码示例演示如何使用 `TagResource`。

CLI

Amazon CLI

为主题添加标签

以下 `tag-resource` 示例将元数据标签添加到指定 Amazon SNS 主题。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考TagResource中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考TagResource中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

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

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)
```

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

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

- 有关 API 的详细信息，请参阅适用TagResource于 K otlin 的Amazon SDK API 参考。

使用 Amazon SNS API 操作管理标签

要使用 Amazon SNS API 管理标签，请使用以下 API 操作：

- ListTagsForResource
- TagResource
- UntagResource

支持 ABAC 的 API 操作

以下是支持基于属性的访问控制 (ABAC) 的 API 操作列表。有关 ABAC 的更多详细信息，请参阅 ABAC 有什么用？ Amazon在 IAM 用户指南中。

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

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

Amazon SNS 事件来源和目标

Amazon SNS 通过路由事件驱动的通知来连接 Amazon Web Services 服务 和外部系统。Amazon SNS 接收来自各种事件 Amazon Web Services 服务，例如数据管道更新、Amazon EC2 扩展操作或安全警报，并将这些事件发布到 Amazon SNS 主题中。然后，这些主题将通知发送到指定目标。

Amazon SNS 支持两种主要的目的地类型：Application-to-Application (A2A) 和 (A2P)。Application-to-Person 在 A2A 消息收发中，Amazon SNS 可以向 Lambda 发送事件以触发自定义业务逻辑，向 Amazon SQS 发送事件以排队消息，向 Amazon Kinesis Data Firehose 发送事件以将数据流式传输到存储和分析服务。对于 A2P 消息收发，Amazon SNS 可以通过短信、电子邮件和推送通知向移动设备发送通知，确保用户或团队及时收到警报。

通过充当中心枢纽，Amazon SNS 可将通知传送到正确的位置，从而帮助您更有效地自动化和管理 Amazon 基础设施。这种设置允许服务之间的无缝集成以及与用户和系统的可靠通信。

Amazon SNS 事件来源

Amazon SNS 与各种类别的 Amazon Web Services 服务 广泛内容集成，允许这些服务向亚马逊 SNS 主题发布事件。这种集成提供了关键事件（例如基础架构、应用程序性能和成本管理的变化）的实时通知。

> **Note**
>
> Amazon SNS 于 2020 年 10 月推出了 FIFO 主题。目前，大多数 Amazon 服务仅支持向标准主题发送事件。

分析服务

下表介绍了 Amazon SNS 如何与 Athena 和 A Amazon Data Pipeline mazon Redshift 等 Amazon 分析服务集成，为关键事件提供实时通知，包括违反控制限制、管道状态更新和数据仓库活动。

您可以利用这些集成来自动响应并保持对数据操作的有效监督。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Athena – 允许您在使用标准 SQL 的 Amazon S3 中分析数据。	在超出控制限制时接收通知。有关更多信息，请参阅 Amazon Athena 用户指南中的设置数据使用控制限制。
Amazon Data Pipeline – 帮助自动执行数据的移动与转换。	接收有关管道组件状态的通知。有关更多信息，请参阅 Amazon Data Pipeline 开发人员指南中的 SnsAlarm。
Amazon Redshift – 管理数据仓库的所有设置、操作和扩展工作。	接收 Amazon Redshift 事件的通知。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 事件通知。

应用程序集成服务

下表介绍了 Amazon SNS 如何与应用程序集成服务（例如 EventBridge 和）集成 Amazon Step Functions，从而为关键业务应用程序启用实时数据路由和通知。

您可以利用这些集成来接收来自 EventBridge 事件的警报，并使用 Step Functions 编排工作流程，从而增强应用程序的自动化和响应能力。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
亚马逊 EventBridge — 提供来自您自己的应用程序、 software-as-a-service (SaaS) 应用程序和 Amazon 服务的实时数据流，并将这些数据传送到目标，包括 Amazon SNS。 EventBridge 以前叫做 “ CloudWatch 活动”。	接收 EventBridge 事件通知。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge目标。
Amazon Step Functions— 允许您将 Amazon Lambda 功能和其他 Amazon 服务结合起来构建关键业务应用程序。	接收 Step Functions 事件的通知。有关更多信息，请参阅 Amazon Step Functions 开发人员指南中的使用 Step Functions 调用 Amazon SNS。

账单和成本管理服务

下表描述了如何 Amazon 账单与成本管理 与 Amazon SNS 集成，以提供预算、价格变动和成本异常通知。

您可以利用此集成来设置 Amazon SNS 主题，以接收有关您的 Amazon 支出的实时提醒，从而帮助您监控成本并有效地应对意外费用。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon 账单与成本管理 – 提供帮助您监控成本并支付账单的特征。	接收预算通知、价格变动通知和异常提示。有关更多信息，请参阅 Amazon Billing 用户指南中的以下页面： • 针对预算通知创建 Amazon SNS 主题 • 设置通知 • 检测异常支出 Amazon Cost Anomaly Detection

业务应用程序服务

下表介绍了 Amazon Chime 如何与 Amazon SNS 集成以发送重要会议事件的通知，从而使您能够随时了解自己的通信和日程安排。

借助此集成，您可以利用 Amazon Chime SDK 事件通知来改进组织内外的协作工具。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Chime – 允许您在组织内外开会、聊天和拨打业务电话。	接收重要的会议事件通知。有关更多信息，请参阅 Amazon Chime 开发人员指南中的 Amazon Chime 开发工具包事件通知。

计算服务

下表描述了 Amazon SNS 如何与各种 Amazon 计算服务集成，使您能够接收有关关键事件的通知，例如 Auto Scaling 操作、 EC2 Image Builder 完成、Elastic Beanstalk 环境变化、Lambda 函数输出和 Lightsail 指标阈值。

您可以利用这些集成，通过随时了解 Amazon Web Services 服务中的重要更新和操作来高效管理您的应用程序和资源。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon A EC2 uto Scaling — 帮助您获得正确数量的亚马逊弹性计算云 (Amazon EC2) 实例，用于处理应用程序的负载。	当 Auto Scaling 启动或终止您的 Auto Scaling 组中的亚马逊 EC2实例时，会收到通知。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的在您的 Auto Scaling 群组扩展时获取 Amazon EC2 SNS 通知。
EC2 Image Builder — 帮助自动创建、管理和部署自定义、安全的 up-to-date服务器映像，这些映像已预先安装并预先配置了软件和设置，以满足特定 IT 标准。	在构建完成时接收通知。有关更多信息，请参阅 C Amazon ompute 博客上的 “在 EC2 Image Builder 管道中跟踪最新服务器映像”。
Amazon Elastic Beanstalk – 处理有关容量预配置、负载均衡、扩展您的应用程序和提供应用程序运行状况监控的部署细节。	接收影响您的应用程序的重要事件的通知。有关更多信息，请参阅 Amazon Elastic Beanstalk 开发人员指南中的使用 Amazon SNS 进行 Elastic Beanstalk 环境通知。
Amazon Lambda – 您可以运行代码，而无需预置或管理服务器。	通过将 SNS 主题设置为 Lambda 死信队列或 Lambda 目标来接收函数输出数据。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的异步调用。
Amazon Lightsail — 帮助开发人员开始使用 Amazon 来构建网站或网络应用程序。	在您的实例、数据库或负载均衡器的指标超过指定阈值时接收通知。有关更多信息，请参阅 Amazon Lightsail 开发人员指南中的在 Amazon Lightsail 中添加通知联系人。

容器服务

下表描述了 Amazon SNS 如何与 Amazon EKS Distro 和 Amazon ECS 等 Amazon 容器服务集成，允许您跟踪 Amazon EKS 集群的更新和安全补丁，并接收有关新的 ECS 优化的 AMI 版本的通知。

您可以随时了解重要的更新和更改，从而利用这些集成来维护容器部署的安全性和效率。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon EKS Distro – 允许您在部署应用程序的任何位置创建可靠且安全的集群。	跟踪使用 Amazon EKS Distro 创建的集群的更新和安全补丁。有关更多信息，请参阅介绍 Amazon EKS Distro - Amazon EKS 使用的开源 Kubernetes 发行版。
Amazon Elastic Container Service (Amazon ECS) – 使您能够运行、停止和管理集群上的容器。	在新的Amazon ECS 优化 AMI 可用时接收通知。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的订阅经 Amazon ECS 优化的 AMI 更新通知。

客户参与服务

下表介绍了 Amazon SNS 如何通过与 Amazon Connect 和 Amazon Simple Email Service (SES) 集成来增强客户参与服务，使您能够接收警报和验证，配置双向短信，以及监控退件、投诉和送达的电子邮件通知。 Amazon End User Messaging SMS

这些集成可帮助您管理跨多个渠道的客户通信。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Connect – 允许您设置全渠道云联络中心以与客户互动。	接收提示和验证。有关更多信息，请参阅《Amazon Connect 管理员指南》中的 Amazon Connect 的力量。 Amazon
Amazon End User Messaging SMS – 通过向客户发送电子邮件、短信和语音消息以及推送通知，帮助客户吸引他们。	配置双向短信，让您可以接收来自客户的消息。有关更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Two-way SMS messaging。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Simple Email Service (Amazon SES) – 为您提供使用您自己的电子邮件地址和域发送和接收电子邮件的经济高效的方法。	接收退回邮件、投诉和送达通知。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的为 Amazon SES 配置 Amazon SNS 通知。

数据库服务

下表描述了 Amazon SNS 如何与 Amazon Amazon Database Migration Service (DMS)、亚马逊 DynamoDB、亚马逊、亚马逊 Neptune、Ama ElastiCache zon Redshift 和亚马逊关系数据库服务 (RDS) 等数据库服务集成，以发送有关数据迁移、维护活动、缓存更新和数据库更改等重要事件的通知。

这些集成通过提供有关关键运营事件的及时警报，帮助您更有效地监控和管理数据库环境。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Database Migration Service – 将数据从本地数据库迁移到 Amazon Web Services 云。	Amazon DMS 事件发生时接收通知；例如，创建或删除复制实例时。有关更多信息，请参阅 Amazon Database Migration Service 用户指南中的在 Amazon Database Migration Service 中使用事件和通知。
Amazon DynamoDB – 一种完全托管的 NoSQL 数据库服务，提供快速而可预测的性能，能够实现无缝扩展。	在发生维护事件时接收通知。有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的自定义 DAX 集群设置。
Amazon ElastiCache — 提供高性能、可调整大小且经济实惠的内存缓存，同时消除与部署和管理分布式缓存环境相关的复杂性。	在发生重要事件时接收通知。有关更多信息，请参阅亚马逊 ElastiCache (Memcached) 用户指南中的事件通知和亚马逊 SNS。
Amazon Neptune – 允许您构建和运行与高度连接的数据集配合使用的应用程序。	在 Neptune 事件发生时接收通知。有关更多信息，请参阅 Neptune 用户指南中的使用 Neptune 事件通知。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Redshift – 管理数据仓库的所有设置、操作和扩展工作。	接收 Amazon Redshift 事件的通知。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 事件通知。
Amazon Relational Database Servic e — 使在 Amazon 云中设置、操作和扩展关系数据库变得更加容易。	接收有关 Amazon RDS 事件的通知。有关更多信息，请参阅 Amazon RDS 用户指南中的使用 Amazon RDS 事件通知。

开发人员工具服务

下表介绍了 Amazon SNS 如何与 Amazon 开发者工具服务（例如、、 Amazon CodeBuild、 Amazon CodeCommit、 Amazon CodeDeploy Amazon）集成 CodeGuru，以及 Amazon CodePipeline 如何针对关键事件（例如构建状态更改、存储库更新、部署进度、性能异常和管道操作）提供通知。

这些集成通过及时接收有关重要事件的警报，帮助您有效地监控和管理软件开发工作流程。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon CodeBuild – 可编译源代码，运行单元测试，并生成可供部署的项目。	在构建成功、失败或从一个构建阶段迁移到另一个构建阶段时接收通知。有关更多信息，请参阅《Amazon CodeBuild 用户指南》 CodeBuild中的生成通知示例。
Amazon CodeCommit – 提供版本控制，以在云中私有存储和管理资产。	接收有关 CodeCommit 仓库事件的通知。有关更多信息，请参阅Amazon CodeCommit 用户指南中的示例：为 Amazon SNS 主题创建 Amazon CodeCommit 触发器。
Amazon CodeDeploy— 自动将应用程序部署到亚马逊 EC2 实例、本地实例、无服务器 Lambda 函数或 Amazon ECS 服务。	接收有关 CodeDeploy 部署或实例事件的通知。有关更多信息，请参阅《Amazon CodeDeploy 用户指南》中的为 CodeDeploy 事件创建触发器。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon CodeGuru — 从您的实时应用程序收集运行时性能数据，并提供建议，以帮助您微调应用程序性能。	在发生异常时接收通知。有关更多信息，请参阅 A mazon CodeGuru 用户指南中的处理异常和建议报告。
Amazon CodePipeline – 自动执行持续发布软件更改所需的步骤。	接收有关批准操作的通知。有关更多信息，请参阅《Amazon CodePipeline 用户指南》CodePipeline中的 “管理批准操作”。

前端 Web 和移动服务

下表介绍了 Amazon SNS Amazon End User Messaging SMS 如何与之集成，通过发送电子邮件、短信、语音消息和推送通知来增强客户参与度，包括配置双向短信以接收客户消息的功能。

借助这种集成，您能够通过各种沟通渠道更有效地与客户互动。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon End User Messaging SMS – 通过向客户发送电子邮件、短信和语音消息以及推送通知，帮助客户吸引他们。	配置双向短信，让您可以接收来自客户的消息。有关更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Two-way SMS messaging。

游戏开发服务

下表介绍了 Amazon SNS 如何与 Amazon GameLift 服务器集成，为基于会话的多人游戏服务器中的配对和队列事件提供通知。

这种集成可帮助游戏开发者自动执行和监控其游戏服务器的部署、操作和扩展，从而确保无缝的游戏体验。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon S GameLift er vers — 为在云端托管基于会话的多人游戏服务器提供解决方案，包括	接收匹配和队列事件通知。有关更多信息，请参阅以下页面：

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
用于部署、操作和扩展游戏服务器的完全托管服务。	• 有关配对通知，请参阅《Amazon Serv GameLift ers FlexMatch 开发者指南》中的设置 FlexMatch 事件通知。 • 有关队列通知，请参阅 Amazon Serv GameLift ers 开发者指南中的为游戏会话放置设置事件通知。

物联网服务

下表介绍了 Amazon SNS 如何 Amazon IoT 与、、和 Amazon IoT Core等服务集成 Amazon IoT Device Defender Amazon IoT Events，为物联网事件 Amazon IoT Greengrass和警报提供通知。

借助这些集成，您可以有效地监控设备行为，接收异常活动的警报，以及通过实时更新和操作来管理物联网设备。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon IoT Core— 提供将您的物联网设备连接到其他设备和服务的云 Amazon Web Services 云 服务。	接收 Amazon IoT Core 事件通知。有关更多信息，请参阅 Amazon IoT 开发人员指南中的创建 Amazon SNS 规则。
Amazon IoT Device Defender – 审核设备的配置，监控互联设备以检测异常行为，并降低安全风险。	在设备违反行为时接收告警。有关更多信息，请参阅《Amazon IoT 开发者指南》中的如何使用 d Amazon IoT Device Defender etec t。
Amazon IoT Events – 让您了解如何监控您的设备和设备机群中的故障情况或操作中的更改，并在发生此类事件时触发措施。	接收 Amazon IoT Events 事件通知。有关更多信息，请参阅 Amazon IoT Events 开发人员指南中的 Amazon Simple Notification Service。
Amazon IoT Greengrass— 扩展 Amazon 到物理设备，这样它们就可以根据自己生成的数据在本地采取行动，同时仍然使用云进行管理、分析和持久存储。	接收 Amazon IoT Greengrass 事件通知。有关更多信息，请参阅 Amazon IoT Greengrass Version 1 开发人员指南中的 SNS 连接器。

机器学习服务

下表描述了亚马逊 SNS 如何与 Amazon 机器学习服务（例如亚马逊、Amazon DevOps Guru、A CodeGuru mazon Lookout for Metrics、Amazon Rekognition 和 Amazon AI）集成，以提供异常通知、运营见解和 SageMaker 数据标签活动。

借助这些集成，您可以监控应用程序性能，接收数据异常警报，并通过实时更新简化机器学习模型的部署。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon CodeGuru — 从您的实时应用程序收集运行时性能数据，并提供建议，以帮助您微调应用程序性能。	在发生异常时接收通知。有关更多信息，请参阅 A mazon CodeGuru 用户指南中的处理异常和建议报告。
Amazon DevOps Guru — 使用机器学习生成运营见解，以帮助您提高运营应用程序的性能。	转发洞察和确认。有关更多信息，请参阅 Amazon 管理与治理博客上的 “与 PagerDuty Amazon DevOps Guru 一起向待命团队提供基于机器学习的运营见解”。
Amazon Lookout for Metrics – 查找数据中的异常情况，确定其根本原因，并使您能够快速采取措施。	接收异常通知。有关更多信息，请参阅 Amazon Lookout for Metrics 开发人员指南中的结合使用 Amazon SNS 与 Lookout for Metrics。
Amazon Rekognition – 让您能够将图像和视频分析添加到您的应用程序	接收请求结果通知。有关更多信息，请参阅 Amazon Rekognition 开发人员指南中的参考：视频分析结果通知。
Amazon SageMaker AI — 使数据科学家和开发人员能够构建和训练机器学习模型，然后将其直接部署到可用于生产的托管环境中。	在标记数据对象时接收通知。有关更多信息，请参阅 Amazon A SageMaker I 开发者指南中的创建流式标签任务。

管理和治理服务

下表介绍了 Amazon SNS 如何与 Amazon 管理和治理服务（例如聊天应用程序中的 Amazon Q Developer）集成 CloudTrail， Amazon CloudFormation CloudWatch、、 Amazon Config、Amazon Control Tower、、 Amazon License Manager、 Amazon Service Catalog Amazon Systems Manager、和，以及为基础设施变更、合规性警报和运营见解等关键事件提供通知。

这些集成通过向相关团队和系统提供及时的警报和更新，帮助您高效地监控和管理 Amazon 环境。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
聊天应用程序中的 Amazon Q Developer — 使 DevOps 软件开发团队能够使用 Amazon Chime 和 Slack 聊天室来监控和响应中的操作事件。Amazon Web Services 云	将通知发送到聊天室。有关更多信息，请参阅《聊天应用程序中的 Amazon Q 开发者管理员指南》中的 “在聊天应用程序中设置 Amazon Q Devel oper”。
Amazon CloudFormation— 使您能够以可预测的方式重复创建和配置 Amazon 基础架构部署。	在创建和更新堆栈时接收通知。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的设置 Amazon CloudFormation 堆栈选项。
Amazon CloudTrail – 提供您的 Amazon Web Services 账户 活动的事件历史记录。	将新的日志文件 CloudTrail 发布到您的 Amazon S3 存储桶时会收到通知。有关更多信息，请参阅Amazon CloudTrail 用户指南 CloudTrail中的配置 Amazon SNS 通知。
Amazon CloudWatch — 实时监控您的 Amazon 资源和您运行 Amazon 的应用程序。	在告警状态更改时接收通知。有关更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。
Amazon Config— 提供中 Amazon 资源配置的详细视图 Amazon Web Services 账户。	在更新资源时，或者在 Amazon Config 针对您的资源评估自定义规则或托管规则时接收通知。有关更多信息，请参阅《Amazon Config 开发人员指南》中的Amazon Config 发送到 SNS 主题的通知和示例配置项目变更通知。
Amazon Control Tower— 使您能够设置和管理安全、合规的多账户 Amazon 环境。	使用提示可帮助您防止登录区内的漂移，并接收合规性通知。有关更多信息，请参阅 Amazon Control Tower 用户指南中的通过 Amazon Simple Notification Service 跟踪提示。
Amazon License Manager— 帮助您在本地环境中 Amazon 集中管理软件供应商提供的软件许可证。	接收 License Manager 通知和提示。有关更多信息，请参阅《L icense Manager 用户指南》中的 License Manager 设置和 “ Amazon 管理和治理” 博客上的 “创建 Amazon License Manager 通知 ServiceNow 事件”。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Service Catalog – 使 IT 管理员可以创建、管理和向最终用户分发已批准的产品组合，然后，最终用户可以在个性化的门户中访问他们所需的产品。	接收有关堆栈事件的通知。有关更多信息，请参阅《Service Catalog 管理员指南》中的 Amazon Service Catalog 通知约束。
Amazon Systems Manager— 允许您在上查看和控制您的基础架构 Amazon。	接收有关命令状态的通知。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的使用 Amazon SNS 通知监控 Systems Manager 状态变更。

媒体服务

下表介绍了 Amazon SNS 如何与 Amazon Elastic Transcoder 集成，以便在媒体转码任务状态发生变化时发送通知，从而使您能够有效地监控和管理将存储在 Amazon S3 中的媒体文件转换为适用于消费类播放设备的格式的过程。

这种集成通过提供有关作业状态的实时警报，帮助您简化媒体处理工作流程。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Elastic Transcoder – 让您可以将 Amazon S3 中存储的媒体文件转换为消费者播放设备所要求的媒体文件格式。	在作业状态更改时接收通知。有关更多信息，请参阅 Amazon Elastic Transcoder 开发人员指南中的任务状态通知。

迁移和传输服务

下表介绍了 Amazon SNS 如何与 Amazon 迁移和传输服务（例如 Amazon Application Discovery Service， Amazon Database Migration Service (DMS) 和）集成 Amazon Snowball Edge，以提供服务器数据收集、数据库迁移活动和数据传输任务等事件的通知。

这些集成通过提供有关关键迁移任务的实时警报和更新，帮助您有效地管理和监控云迁移流程。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Application Discovery Service— Amazon Web Services 云 通过收集有关本地服务器的使用情况和配置数据，帮助您规划向的迁移。	通过接收事件通知 Amazon CloudTrail。有关更多信息，请参阅 Application Discovery Service 用户指南中的使用 Amazon CloudTrail记录 Application Discovery Service API 调用。
Amazon Database Migration Service – 将数据从本地数据库迁移到 Amazon Web Services 云。	Amazon DMS 事件发生时接收通知；例如，创建或删除复制实例时。有关更多信息，请参阅 Amazon Database Migration Service 用户指南中的在 Amazon Database Migration Service 中使用事件和通知。
Amazon Snowball Edge— 使用物理存储设备在 Amazon S3 和您的现场数据存储位置之间 faster-than-internet快速传输大量数据。	接收 Snowball Edge 任务的通知。有关更多信息，请参阅《Amazon Snowball Edge User Guide》中的 Notifications for Snow Family devices。

网络和内容分发服务

下表描述了 Amazon SNS 如何与 Amazon 联网和内容分发服务（例如亚马逊 API Gateway、Amazon、Elastic Load Balancing CloudFront Amazon Direct Connect、Amazon Load Balancing、Amazon Route 53 和亚马逊 VPC）集成，以发送有关 API 消息、 CloudFront 指标警报、连接状态更改、负载均衡器事件、运行状况检查状态和 VPC 终端节点活动等事件的通知。

这些集成通过提供及时的警报和更新，帮助您监控和管理网络和内容传送操作。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon API Gatew ay — 使您能够以任何规模创建和 WebSocket APIs 部署自己的 REST。	接收发布到 API Gateway 终端节点的消息。有关更多信息，请参阅《API Gateway 开发者指南》中的教程：通过 Amazon 集成构建 API Gateway RES T API。
亚马逊 CloudFront — 加快静态和动态网页内容的分发，例如.html、.css、.php、图像和媒体文件。	当发生基于指定 CloudFront 指标的警报时，会收到通知。有关更多信息，请参阅《Amazon

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
	CloudFront 开发者指南》中的设置警报以接收通知。
Amazon Direct Connect— 通过标准以太网光纤电缆将您的内部网络链接到某个 Amazon Direct Connect 位置。	在监控 Amazon Direct Connect 连接状态的告警状态更改时接收通知。有关更多信息，请参阅《Amazon Direct Connect 用户指南》中的创建 CloudWatch 警报以监控 Amazon Direct Connect 连接。
Elastic Load Balancing — 自动将您的传入流量分配到多个或多个可用区域中的多个目标，例如亚马逊 EC2 实例、容器和 IP 地址。	接收您为负载均衡器事件创建的告警的通知。有关更多信息，请参阅《传统负载均衡器用户指南》中的为负载均衡器创建 CloudWatch 警报。
Amazon Route 53 – 提供域注册、DNS 路由和运行状况检查。	在运行状况检查状态为运行不佳时接收通知。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的在运行状况检查状态为运行不佳时接收 Amazon SNS 通知（控制台）。
Amazon Virtual Private Cloud（亚马逊 VPC）— 使您能够将 Amazon 资源启动到您定义的虚拟网络中。	接收接口终端节点上发生的特定事件的提醒。有关更多信息，请参阅 Amazon VPC 用户指南中的为终端节点服务创建和管理通知。

安全性、身份与合规性服务

下表描述了 Amazon SNS 如何与 Amazon 安全、身份和合规服务（例如 Amazon Directory Service亚马 GuardDuty逊、Amazon Inspector）集成，并提供有关目录状态更改 Amazon Security Hub、安全发现、Inspector 事件和安全中心公告的通知。

这些集成通过提供有关安全和合规事件的及时警报和更新，帮助您保持稳健的安全实践。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Directory Service— 提供多种与其他方式一起使用 Microsoft 活动目录 (AD) 的方法 Amazon Web Services 服务。	在目录状态发生变化时接收电子邮件或文本 (SMS) 消息。有关更多信息，请参阅 Amazon Directory Service 管理指南中的配置目录状态通知。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon GuardDuty — 提供持续的安全监控，以帮助识别 Amazon 环境中意外的、可能未经授权的或恶意的活动。	接收有关新发布的调查结果类型、现有调查结果类型的更新以及其他功能更改的通知的信息。有关更多信息，请参阅 A mazon GuardDuty 用户指南中的订阅 GuardDuty 公告 SNS 主题。
Amazon Inspector — 测试您的亚马逊 EC2 实例的网络可访问性以及在这些实例上运行的应用程序的安全状态。	接收有关 Amazon Inspector 事件的通知。有关更多信息，请参阅 Amazon Inspector 用户指南中的为 Amazon Inspector 通知设置 SNS 主题。
Amazon Security Hub – 自动执行 Amazon 安全检查并集中管理安全提示。	接收有关 Amazon Security Hub 公告的通知，包括有关已添加、编辑或停用的 Amazon Security Hub 控件或标准的通知。有关更多信息，请参阅通过 Amazon SNS 订阅 Amazon Security Hub 公告。

无服务器服务

下表描述了 Amazon SNS 如何与亚马逊 DynamoDB、Amazon 和 Lambda 等服务集成 EventBridge，以发送有关关键事件（例如维护更新、实时数据流和 Lambda 函数输出）的通知。

借助这些集成，您可以及时接收有关关键操作的警报并自动响应这些事件，从而高效地监控和管理应用程序。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon DynamoDB – 一种完全托管的 NoSQL 数据库服务，提供快速而可预测的性能，能够实现无缝扩展。	在发生维护事件时接收通知。有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的自定义 DAX 集群设置。
亚马逊 EventBridge — 提供来自您自己的应用程序、 software-as-a-service (SaaS) 应用程序的实时数据流， Amazon Web Services 服务 并将这些数据路由到目标，包括 Amazon SNS。EventBridge 以前叫做 “ CloudWatch 活动”。	接收 EventBridge 事件通知。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge目标。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Lambda – 您可以运行代码，而无需预置或管理服务器。	通过将 SNS 主题设置为 Lambda 死信队列或 Lambda 目标来接收函数输出数据。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的异步调用。

存储服务

下表介绍了 Amazon SNS 如何与诸如亚马逊弹性文件系统 (EFS) Amazon Backup、Amazon S3 Glacier、Amazon S3 等 Amazon 存储服务集成，以及 Amazon Snowball Edge 如何针对备份活动、文件系统警报、数据检索任务、存储桶更改和数据传输操作等各种事件提供通知。

这些集成通过及时接收有关关键存储事件的警报，帮助您高效地监控和管理存储解决方案。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Backup – 帮助您在云中以及本地集中管理和自动执行跨 Amazon Web Services 服务的数据备份	接收 Amazon Backup 事件通知。有关更多信息，请参阅《Amazon Backup 开发者指南》中的 “使用 Amazon SNS 跟踪 Amazon Backup 事件”。
Amazon Elastic File System — 为您的亚马逊 EC2 实例提供文件存储。	接收您为 Amazon EFS 事件创建的告警的通知。有关更多信息，请参阅 Amazon Elastic File System 用户指南中的自动监控工具。
Amazon S3 Glacier – 为不经常使用的数据提供存储。	在文件库上设置通知配置，以便在作业完成时向 SNS 主题发送消息。有关更多信息，请参阅 Amazon S3 Glacier 开发人员指南中的在 Amazon S3 Glacier 中配置文件库通知。
Amazon Simple Storage Service (Amazon S3) – 提供对象存储服务。	在 Amazon S3 存储桶发生更改时或在对象未复制到目标区域的罕见情况下接收通知。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的演练：为存储桶配置通知（SNS 主题或 SQS 队列）和使用复制指标和 Amazon S3 事件通知监控进度。

Amazon Web Services 服务	与 Amazon SNS 一起使用的益处
Amazon Snowball Edge— 使用物理存储设备在 Amazon S3 和您的现场数据存储位置之间 faster-than-internet快速传输大量数据。	接收 Snowball Edge 任务的通知。有关更多信息，请参阅《Amazon Snowball Edge User Guide》中的 Notifications for Snow Family devices。

其他事件来源

下表描述了如何使用 Amazon SNS 及时接收有关 Amazon 每日功能更新和 Amazon IP 地址范围变化的通知，确保您了解最新的 Amazon 服务版本、实例类型、VPC 终端节点和公有 IP 地址的更改。

这些集成可帮助您顺应基础 Amazon 架构 up-to-date的变化并有效地管理您的资源。

来源	与 Amazon SNS 一起使用的益处
Amazon 每日功能更新	通过 Amazon SNS 主题及时接收有关 Amazon 的版本和更新的详细信息。这些版本包括 Amazon Web Services 区域与 Amazon 服务配额 Amazon Web Services 服务 集成的亚马逊 VPC 终端节点、亚马逊实例类型、亚马逊 A SageMaker I EC2 实例类型、亚马逊 Nimble Studio 实例类型、亚马逊 RDS 数据库引擎版本和亚马逊 MSK Apache Kafka 版本。Amazon Web Services 服务有关更多信息，请参阅Amazon 新闻博客中的通过 Amazon SNS 订阅 Amazon 每日专题更新。
Amazon IP 地址范围	通过 Amazon SNS 主题接收 Amazon IP 范围变更通知。有关更多信息，请参阅中的 Amazon IP 地址范围通知和Amazon 新闻博客中的 “Amazon Web Services 一般参考通过 Amazon SNS 订阅 Amazon 公有 IP 地址变更”。

有关事件驱动型计算的更多信息，请参阅以下资源：

- 什么是事件驱动型架构？
- 使用 Amazon SNS 以及计算、存储、数据库 Amazon 和网络服务进行事件驱动型计算，请访问计算博客Amazon
- 在 Compute 博客上使用 Amazon 事件分叉管道丰富事件驱动架构Amazon

Amazon SNS 事件目标

本主题按照 application-to-application (A2A) 消息和 application-to-person (A2P) 通知列出了所有事件目的地。

> **Note**
>
> Amazon SNS 于 2020 年 10 月推出了 FIFO 主题。目前，大多数 Amazon 服务仅支持接收来自 SNS 标准主题的事件。Amazon SQS 支持接收来自 SNS 标准主题和 FIFO 主题的事件。

A2A 目标

下表描述了亚马逊 SNS 如何将事件传送到各个 application-to-application (A2A) 目的地，例如亚马逊数据 Firehose、Lambda、亚马逊 SQS、事件分叉管道和 HTTP/S 终端节点。 Amazon

这些集成允许您存档和分析数据、触发自定义业务逻辑、促进应用程序集成以及将事件路由到外部 Webhook，从而提高事件驱动型架构的效率和灵活性。

事件目标	与 Amazon SNS 一起使用的益处
Amazon Data Firehose	将事件传输到传输流以进行存档和分析。通过传输流，您可以将事件传送到亚马逊简单存储服务 (Amazon S3)、Amazon Redshift 和 OpenSearch 亚马逊服务（OpenSearch 服务）等 Amazon 目的地，或第三方目的地，例如Datadog、New Relic、MongoDB和Splunk。有关更多信息，请参阅 扇出到 Firehose 传输流。
Amazon Lambda	将事件传输到函数，用于触发自定义业务逻辑的执行。有关更多信息，请参阅 将 Amazon SNS 通知扇出到 Lambda 函数以进行自动处理。

事件目标	与 Amazon SNS 一起使用的益处
Amazon SQS	将事件传输到队列，以进行应用程序集成。有关更多信息，请参阅 将 Amazon SNS 通知扇出到 Amazon SQS 队列进行异步处理。
Amazon 事件分叉管道	将事件传输到事件备份和存储、事件搜索和分析或事件重放管道。有关更多信息，请参阅 将 Amazon SNS 事件扇出到 Amazon Event Fork Pipelines。
HTTP/S	向外部 Webhook 传输事件。有关更多信息，请参阅 将 Amazon SNS 通知扇出到 HTTPS 端点。

A2P 目标

下表描述了 Amazon SNS 如何向各种目的地发送 application-to-person (A2P) 通知，包括通过短信和本地推送通知的手机、电子邮件收件箱、Amazon Chime 聊天室、Slack 频道以及通过以下方式向待命团队提供运营见解。 PagerDuty

这些集成通过支持跨多个平台和通信渠道的实时警报和更新，提高了通信和运营效率。

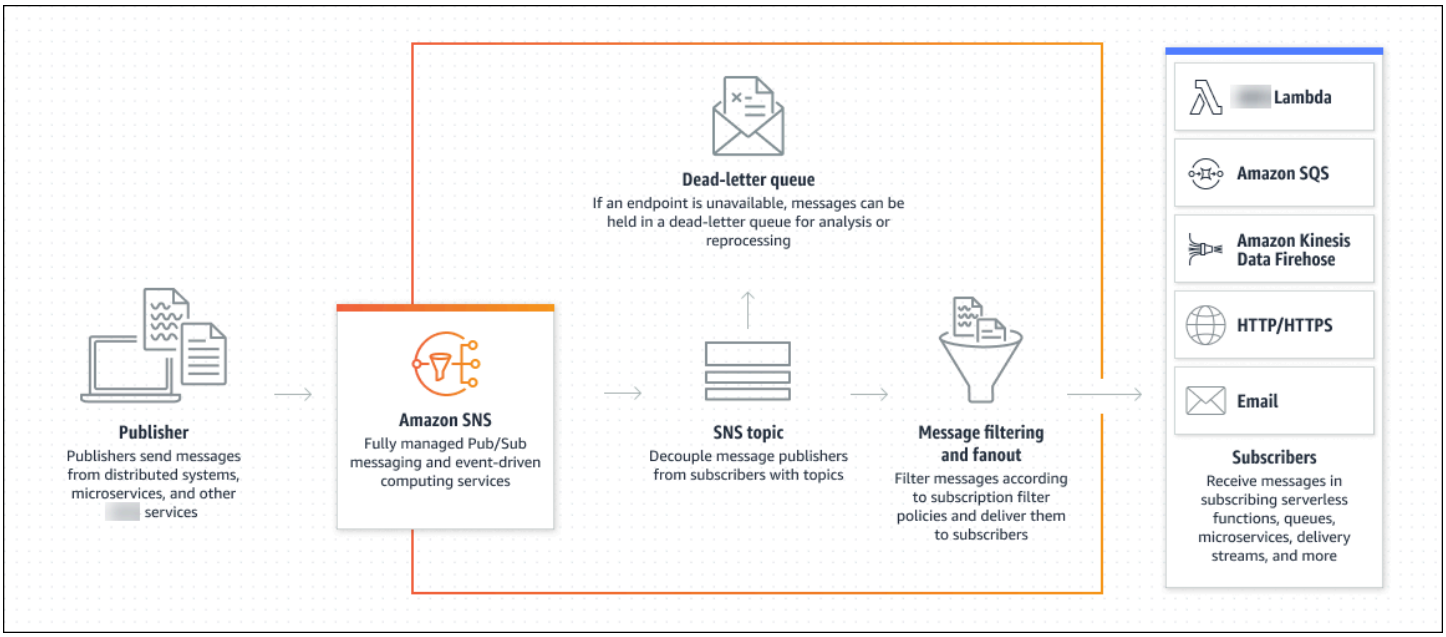
事件目标	与 Amazon SNS 一起使用的益处
短信	以短信形式将事件传输到移动电话。有关更多信息，请参阅 移动文本消息与 Amazon SNS。
电子邮件	将事件作为电子邮件发送到收件箱。有关更多信息，请参阅 Amazon SNS 电子邮件订阅设置和管理。
平台终端节点	将事件作为本机推送通知发送到移动电话。有关更多信息，请参阅 使用 Amazon SNS 发送移动推送通知。
聊天应用程序中的 Amazon Q 开发者版	将事件传输到 Amazon Chime 聊天室或 Slack 频道。有关更多信息，请参阅《聊天应用程序中

事件目标	与 Amazon SNS 一起使用的益处
	的 Amazon Q 开发者管理员指南》中的以下页面： • 使用 Amazon Chime 在聊天应用程序中设置 Amazon Q Developer • 使用 Slack 在聊天应用程序中设置 Amazon Q Developer • 在与其他 Amazon 服务的聊天应用程序中使用 Amazon Q Developer
PagerDuty	向待命团队传输运营洞察。有关更多信息，请参阅Amazon 管理与治理博客上的 “通过 PagerDuty Amazon DevOps Guru 向待命团队提供基于机器学习的运营见解”。

> **Note**
>
> 您可以将原生 Amazon 事件和自定义事件传送到聊天应用程序：
>
> - 原生 Amazon 事件 — 你可以在聊天应用程序中使用 Amazon Q Developer 通过亚马逊 SNS 主题向 Amazon Chime 和 Slack 发送原生 Amazon 事件。支持的原生 Amazon 事件集包括来自 Amazon 账单与成本管理、、 Amazon Health Amazon CloudFormation CloudWatch、Amazon 等的事件。有关更多信息，请参阅《Amazon Q 开发者聊天应用程序管理员指南》中的 “在聊天应用程序中使用与其他服务的聊天应用程序中使用 Amazon Q Dev eloper”。
> - 自定义事件 – 您还可以通过 Amazon SNS 主题将自定义事件发送到 Amazon Chime、Slack 和 Microsoft 团队。为此，请将自定义事件发布到 SNS 主题，该主题将事件传输到订阅的 Lambda 函数。然后，Lambda 函数使用聊天应用程序的 Webhook 将事件传输给收件人。有关更多信息，请参阅 如何使用 Webhook 将 Amazon SNS 消息发布到 Amazon Chime、Slack 或 Microsoft 团队？

使用 Amazon SNS 发送消息 application-to-application

Amazon SNS 通过将发布者与订阅者分开来简化 application-to-application (A2A) 消息传递，后者支持微服务、分布式系统和无服务器应用程序。消息会发送到 Amazon SNS 主题，在那里可以对其进行筛选并发送给 Lambda、Amazon SQS 或 HTTP 端点等订阅用户。如果传输失败，消息将存储在死信队列中，以供进一步分析或重新处理。



扇出到 Firehose 传输流

您可以将 Amazon Data Firehose 传送流订阅亚马逊 SNS 主题，这样您就可以向其他存储和分析终端节点发送通知。发布到 Amazon SNS 主题的消息将发送到订阅的 Firehose 传输流，并按照 Firehose 中配置的方式传输到目标。订阅所有者最多可以为 Amazon SNS 主题订阅五个 Firehose 传输流。每个 Firehose 传输流对于每秒请求量和吞吐量都有一个原定设置限额。此限制可能会导致发布的消息（入站流量）多于传输的消息（出站流量）。当入站流量多于出站流量时，您的订阅可能会累积大量的消息积压，从而可能会导致较长的消息传输延迟。您可以根据发布率请求增加限额，以避免对您的工作负载产生不利影响。

通过Firehose传送流，您可以将亚马逊SNS通知分发给亚马逊简单存储服务（Amazon S3）、亚马逊Redshift、亚马逊服务（服务）以及第三方OpenSearch 服务提供商，例如Datadog、 OpenSearch New Relic、MongoDB和Splunk。

例如，您可以使用此功能将发送到 Amazon S3 存储桶中的主题的消息永久存储以用于合规性、存档或其他目的。为此，请创建一个带有 Amazon S3 存储桶目标的 Firehose 传输流，然后将该传输流

订阅亚马逊 SNS 主题。再举一个例子，要对发送到 Amazon SNS 主题的消息进行分析，请创建带有 OpenSearch 服务索引目标的传输流。然后，您可以为 Amazon SNS 主题订阅 Firehose 传输流。

Amazon SNS 还支持针对发送到 Firehose 端点的通知进行消息传输状态日志记录。有关更多信息，请参阅 Amazon SNS 消息传输状态。

订阅 Firehose 传输流到 Amazon SNS 主题的先决条件

要将 Amazon Data Firehose 传送流订阅到 SNS 主题，您 Amazon Web Services 账户 必须具备以下条件：

- 标准 SNS 主题。有关更多信息，请参阅 创建 Amazon SNS 主题。
- Firehose 传输流。有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的创建 Amazon Data Firehose 传输流和授予您的应用程序对 Firehose 资源的访问权限。
- 信任 Amazon SNS 服务委托人并有权写入传输流的 Amazon Identity and Access Management (IAM) 角色。创建订阅时，您将输入此角色的 Amazon Resource Name (ARN) 作为 `SubscriptionRoleARN`。Amazon SNS 担任此角色，它允许 Amazon SNS 将记录放入 Firehose 传输流中。

  下面的示例策略显示了建议的权限：

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

要提供使用 Firehose 的完全权限，您还可以使用 Amazon 托管策略。`AmazonKinesisFirehoseFullAccess`或者，要为使用 Firehose 提供更严格的权限，您可以创建自己的策略。至少，策略必须提供在特定传输流上运行 `PutRecord` 操作的权限。

在所有情况下，您都还必须编辑信任关系以包括 Amazon SNS 服务委托人。例如：

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

有关创建角色的更多信息，请参阅 IAM 用户指南中的创建角色以向 Amazon 服务委派权限。

完成这些要求后，您可以将传输流订阅到 SNS 主题。

将 Firehose 传输流订阅到 Amazon SNS 主题

要将 Amazon SNS 通知传输至 Amazon Data Firehose 传输流，首先确保您已经解决了所有先决条件。有关支持的端点的列表，请参阅《Amazon Web Services 一般参考》中的 Amazon Data Firehose 端点和限额。

将 Firehose 传输流订阅到主题

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择订阅。
3. 在订阅页面上，选择创建订阅。
4. 在创建订阅页上的详细信息部分，执行以下操作：
   a. 对于 Topic ARN（主题 ARN），选择标准主题的 Amazon Resource Name (ARN)。
   b. 对于协议，选择 Firehose。

c. 对于端点，选择可从 Amazon SNS 中接收通知的 Firehose 传输流的 ARN。

d. 对于订阅角色 ARN，指定您为写入到 Firehose 传输流创建的 Amazon Identity and Access Management（IAM）角色的 ARN。有关更多信息，请参阅 订阅 Firehose 传输流到 Amazon SNS 主题的先决条件。

e. （可选）要从已发布消息中删除任何 Amazon SNS 元数据，请选择 Enable raw message delivery（启用原始消息传输）。有关更多信息，请参阅 Amazon SNS 原始消息传输。

5. （可选）要配置筛选策略，请展开 Subscription filter policy（订阅筛选策略）部分。有关更多信息，请参阅 Amazon SNS 订阅筛选策略。

6. （可选）要为订阅配置死信队列，请展开 Redrive policy (dead-letter queue)（重新驱动策略（死信队列））部分。有关更多信息，请参阅 Amazon SNS 死信队列。

7. 选择创建订阅。

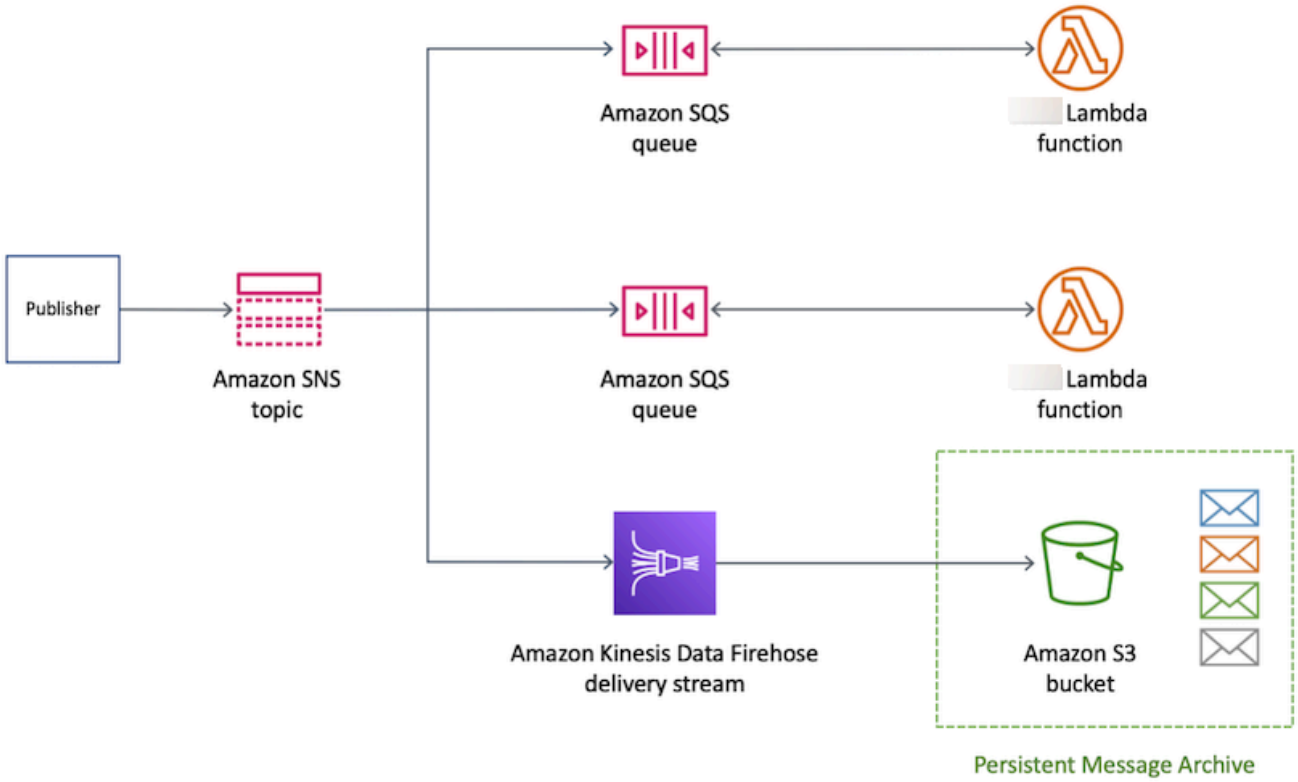
控制台将创建订阅并打开订阅的 Details（详细信息）页面。

管理跨多个传送流目标的 Amazon SNS 消息

Amazon Data Firehose 传输流允许您管理跨多个目的地的 Amazon SNS 消息，从而实现与亚马逊 S3、亚马逊服务、 OpenSearch 亚马逊 Redshift 和 HTTP 终端节点的集成，用于存储、索引和分析。通过正确配置消息格式和传送，您可以将 Amazon SNS 通知存储在 Amazon S3 中以供日后处理，使用 Amazon Athena 分析结构化消息数据，为消息编制索引以进行实时搜索和可视化， OpenSearch 在 Amazon Redshift 中结构化存档以进行高级查询。

分析 Amazon SNS 消息并将其存储在 Amazon S3 目标中

本主题介绍了 Amazon Data Firehose 交付流如何将数据发布到亚马逊简单存储服务 (Amazon S3) Simple S3。



主题

- 格式化 Amazon SNS 通知以存储在 Amazon S3 目标中
- 使用 Athena 分析在 Amazon S3 中存储的 Amazon SNS 消息

格式化 Amazon SNS 通知以存储在 Amazon S3 目标中

以下示例显示了发送到亚马逊简单存储服务 (Amazon S3) Storage Service 存储桶的 Amazon SNS 通知，为了便于阅读，该通知带有缩进。

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。禁用原始邮件传输时，Amazon SNS 会将 JSON 元数据添加到消息中，其中包括以下属性：
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`

- Timestamp
- UnsubscribeURL
- MessageAttributes

有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传输。

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

以下示例显示通过 Amazon Data Firehose 传输流发送到同一 Amazon S3 存储桶的三条 SNS 消息。应用缓冲，并用换行符分隔每条消息。

```
{"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"}
```

使用 Athena 分析在 Amazon S3 中存储的 Amazon SNS 消息

本页介绍如何分析通过亚马逊数据 Firehose 传送流发送到亚马逊简单存储服务 (Amazon S3) 目的地的亚马逊 SNS 消息。

要分析通过 Firehose 传输流发送到 Amazon S3 目标的 SNS 消息

1. 配置 Amazon S3 资源。有关说明，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶和 Amazon Simple Storage Service 用户指南中的使用 Amazon S3 存储桶。
2. 配置传输流。有关说明，请参阅《Amazon Data Firehose 开发人员指南》中的选择 Amazon S3 作为目标。
3. 使用 Amazon Athena 通过标准的 SQL 查询 Amazon S3 对象。有关更多信息，请参阅 Amazon Athena 用户指南中的入门。

示例查询

在本示例查询中，我们假设满足以下条件：

- 消息存储在 `default` schema 的 `notifications` 表中。
- `notifications` 表包含一个类型为 `string` 的 `timestamp` 列。

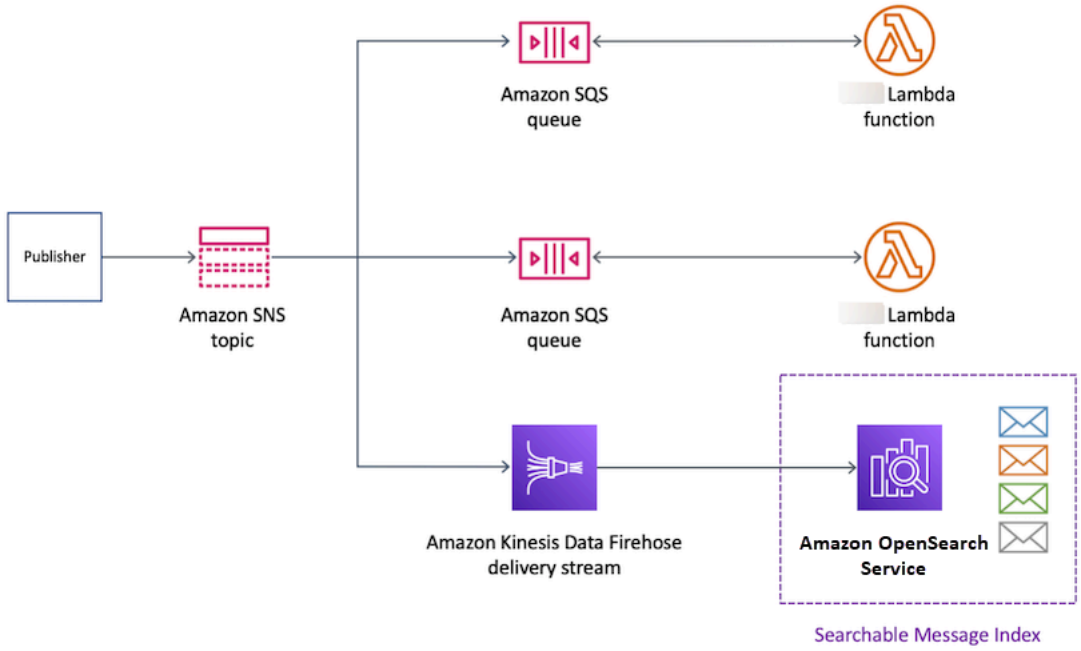
以下查询返回在指定日期范围内收到的所有 SNS 消息：

```
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';
```

将 Amazon SNS 消息与亚马逊 OpenSearch 服务目标集成

本节介绍亚马逊数据 Firehose 交付流如何将数据发布到亚马逊 OpenSearch 服务（OpenSearch 服务）。



主题

- 在服务索引中 OpenSearch 存储和格式化 Amazon SNS 通知
- 分析服务目标的 Amazon SNS OpenSearch 消息

在服务索引中 OpenSearch 存储和格式化 Amazon SNS 通知

以下示例演示了向名为的亚马逊 OpenSearch 服务（OpenSearch 服务）索引发送的 Amazon SNS 通知。`my-index`此索引在 `Timestamp` 字段中具有时间筛选字段。SNS 通知放置在负载的 `_source` 属性中。

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。禁用原始邮件传输时，Amazon SNS 会将 JSON 元数据添加到消息中，其中包括以下属性：
>
> - `Type`

- MessageId
- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传输。


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

```
  ]
}
```

分析服务目标的 Amazon SNS OpenSearch 消息

本主题介绍如何分析通过 Amazon Data Firehose 传送流发送到亚马逊 OpenSearch 服务（服务）OpenSearch 目的地的 Amazon SNS 消息。

分析通过 Firehose 发送到服务目标的传送流 OpenSearch 的 SNS 消息

1. 配置您的 OpenSearch 服务资源。有关说明，请参阅《亚马逊 OpenSearch 服务开发者指南》中的 “亚马逊 OpenSearch 服务入门”。
2. 配置传输流。有关说明，请参阅 Amazon Data Firehose 开发者指南中的为您的目的地选择 OpenSearch 服务。
3. 使用 OpenSearch 服务查询和 Kibana 运行查询。有关更多信息，请参阅《亚马逊 OpenSearch 服务开发者指南》中的步骤 3：在 OpenSearch 服务域中搜索文档和 Kibana。

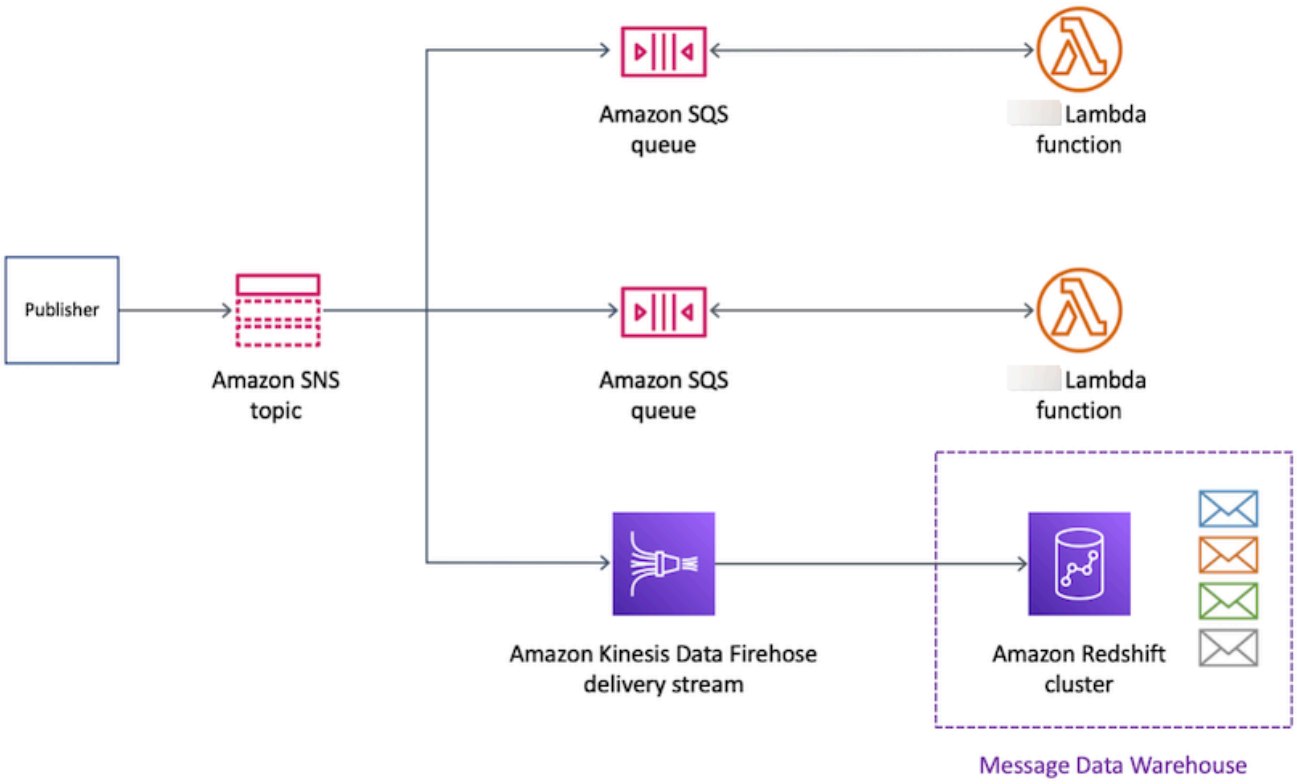
示例查询

以下示例查询指定日期范围内接收的所有 SNS 消息的 `my-index` 索引：

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

在 Amazon Redshift 目标中配置 Amazon SNS 消息传输和分析

本主题介绍如何将亚马逊 SNS 通知分散到亚马逊 Data Firehose 传送流，然后该流将数据发布到亚马逊 Redshift。通过此设置，您可以连接到 Amazon Redshift 数据库，并使用 SQL 查询工具检索符合特定条件的 Amazon SNS 消息。



主题

- 在 Amazon Redshift 表中构造 Amazon SNS 消息归档
- 分析存储在 Amazon Redshift 目标中的 Amazon SNS 消息

在 Amazon Redshift 表中构造 Amazon SNS 消息归档

对于 Amazon Redshift 终端节点，亚马逊 SNS 消息以行形式存档在表中。以下是如何存储数据的示例：

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。禁用原始邮件传输时，Amazon SNS 会将 JSON 元数据添加到消息中，其中包括以下属性：

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

有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传输。
尽管 Amazon SNS 使用此列表中显示的大写向邮件添加了属性，但 Amazon Redshift 表中的列名称以所有小写字符显示。要转换 Amazon Redshift 终端节点的 JSON 元数据，您可以使用 SQL `COPY` 命令。有关更多信息，请参阅 Amazon Redshift 数据库开发人员指南中的从 JSON 中复制示例和使用“auto ignorecase”选项从 JSON 数据中加载。

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
通知	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	示例主题	示例消息	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=unsubscribe&=arnSubscriptionArn: aws: sns: us-east-1:11111111	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						1111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	
通知	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	示例主题2	示例消息2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=unsubscribe&=arnSubscriptionArn: aws: sns: us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
通知	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	示例主题 3	示例消息 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=unsubscribe&=arnSubscriptionArn: aws: sns: us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

有关向 Amazon Redshift 终端节点扇出通知的更多信息，请参阅 在 Amazon Redshift 目标中配置 Amazon SNS 消息传输和分析。

分析存储在 Amazon Redshift 目标中的 Amazon SNS 消息

本主题介绍如何分析通过亚马逊 Data Firehose 传送流发送到亚马逊 Redshift 目的地的 Amazon SNS 消息。

要分析通过 Firehose 传输流发送到 Amazon Redshift 目标的 SNS 消息

1. 配置您的 Amazon Redshift 资源。有关说明，请参阅 Amazon Redshift 入门指南中的Amazon Redshift 入门。
2. 配置传输流。有关说明，请参阅《Amazon Data Firehose 开发人员指南》中的选择 Amazon Redshift 作为目标。
3. 运行查询。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 使用查询编辑器查询数据库。

示例查询

在本示例查询中，我们假设满足以下条件：

- 消息存储在默认 `public` schema 的 `notifications` 表中。
- SNS 消息的 `Timestamp` 属性存储在表的 `timestamp` 列中，其列数据类型为 `timestamptz`。

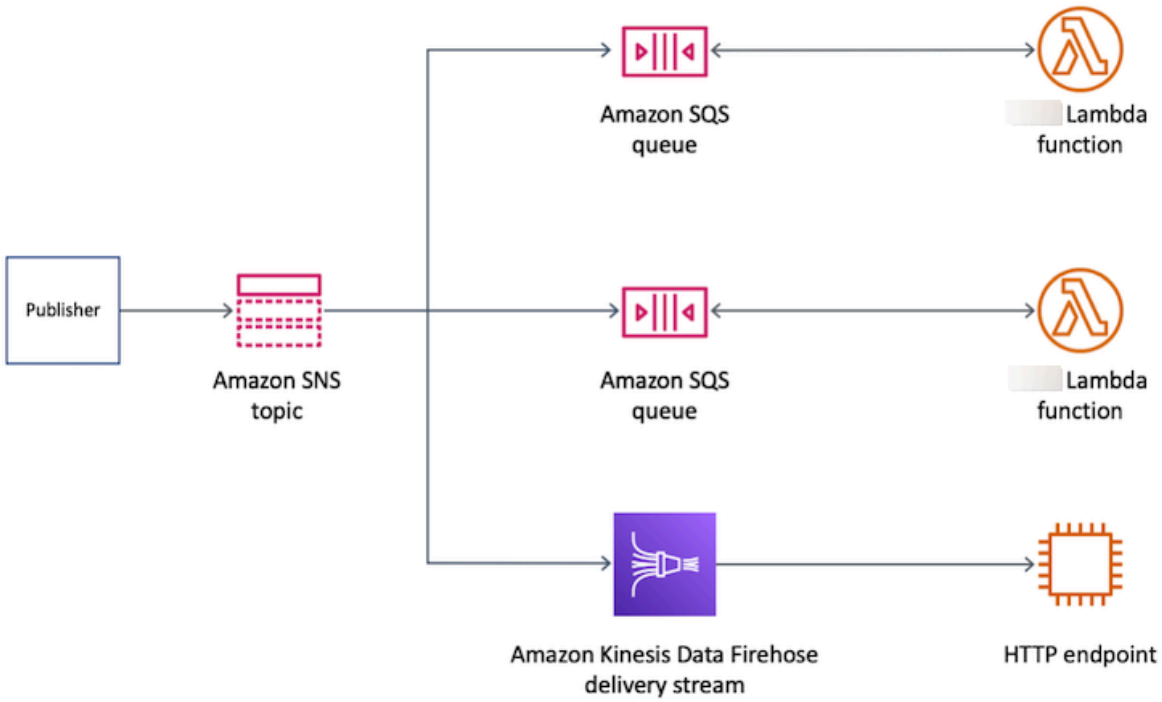
> **Note**
>
> 要转换 Amazon Redshift 终端节点的 JSON 元数据，您可以使用 SQL `COPY` 命令。有关更多信息，请参阅 Amazon Redshift 数据库开发人员指南中的从 JSON 中复制示例和使用“auto ignorecase”选项从 JSON 数据中加载。

以下查询返回在指定日期范围内收到的所有 SNS 消息：

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

使用 Amazon Data Firehose 配置向 HTTP 目标发送 Amazon SNS 消息

本主题介绍了 Amazon Data Firehose 传输流如何将数据发布到 HTTP 终端节点。



主题

- 传送至 HTTP 目标的 Amazon SNS 通知格式

传送至 HTTP 目标的 Amazon SNS 通知格式

以下是来自亚马逊 SNS 的 HTTP POST 请求正文的示例，该请求正文通过亚马逊数据 Firehose 传输流发送到 HTTP 终端节点。Amazon SNS 通知在 records 属性中被编码为 base64 有效负载。

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传输。

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

```
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```

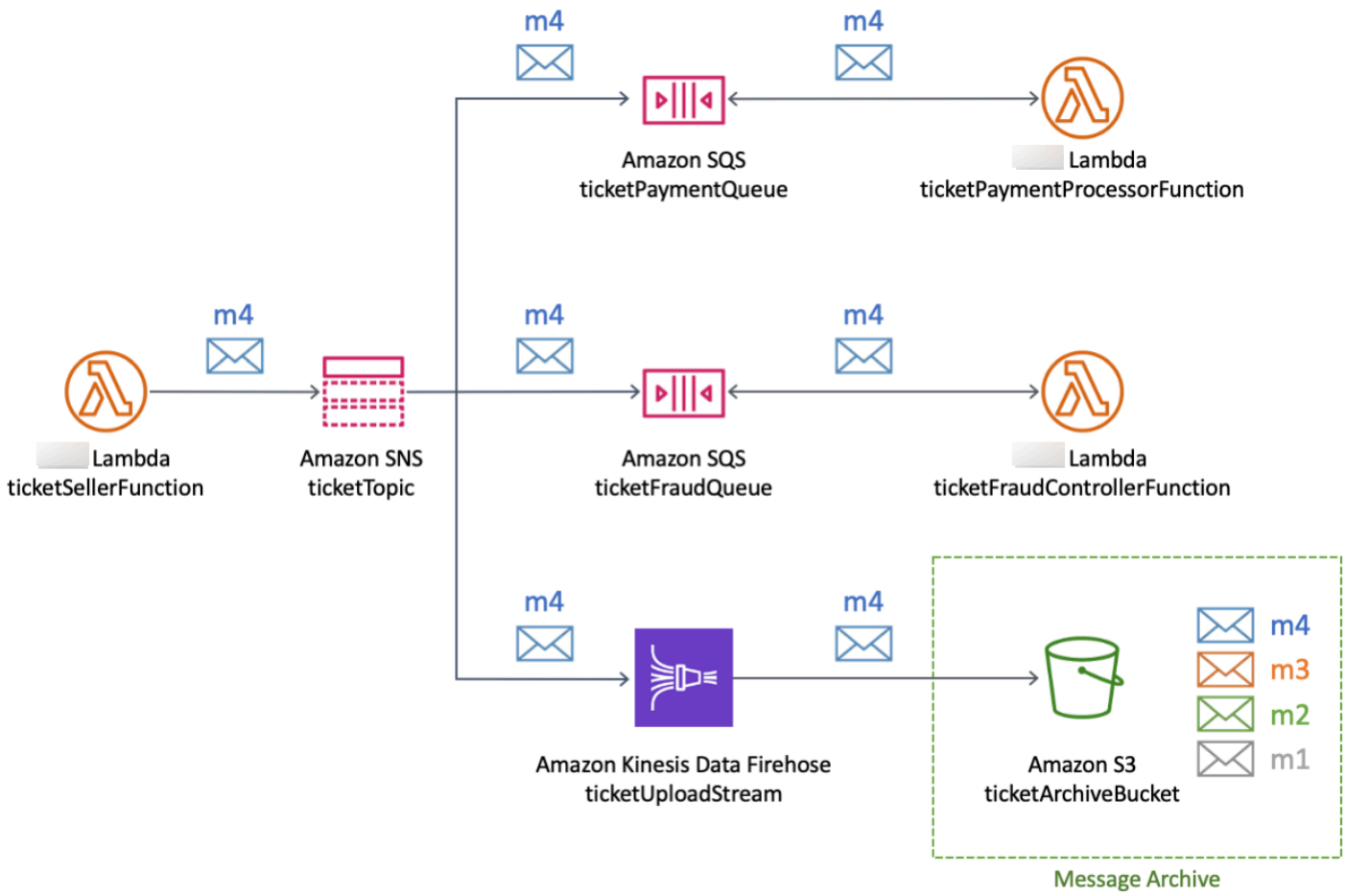
Amazon SNS 消息归档和分析：航空票务平台示例应用场景

本主题提供了存档和分析 Amazon SNS 消息的常见用例的教程。

此使用案例的设置是一个在受监管环境中运行的航空公司票务平台。

1. 该平台受合规性框架的约束，要求公司将所有售票记录存档至少五年。
2. 为了实现数据保留方面的合规目标，该公司订阅了现有亚马逊 SNS 主题的 Amazon Data Firehose 传送流。
3. 传输流的目标是 Amazon Simple Storage Service (Amazon S3) 存储桶。通过此配置，发布到 SNS 主题的所有事件都将存档到 Amazon S3 存储桶中。

下图显示了此配置的架构：



为了运行分析并了解门票销售的情况，该公司使用 Amazon Athena 运行 SQL 查询。例如，公司可以通过查询来了解最受欢迎的目的地和最频繁的旅客。

要为此用例创建 Amazon 资源，您可以使用 Amazon Web Services Management Console 或 Amazon CloudFormation 模板。

主题

- 为 Amazon SNS 消息归档和分析设置初始 Amazon 资源
- 为 Amazon SNS 消息归档设置 Firehose 传输流
- 为 Firehose 传输流订阅 Amazon SNS 主题
- 测试和查询 Amazon SNS 配置以实现有效的数据管理
- 使用 Amazon CloudFormation 模板自动归档 Amazon SNS 消息

为 Amazon SNS 消息归档和分析设置初始 Amazon 资源

本主题介绍如何创建消息存档和分析示例用例所需的资源：

- 一个 Amazon Simple Storage Service (Amazon S3) 存储桶
- 两个 Amazon Simple Queue Service (Amazon SQS) 队列
- 一个 Amazon SNS 主题
- 对 Amazon SNS 主题的两个 Amazon SQS 订阅

要创建起始资源

1. 创建 Amazon S3 存储桶：
   a. 打开 Amazon S3 控制台。
   b. 选择创建存储桶。
   c. 对于 Bucket name（存储桶名称），请输入全局唯一名称。保留其他字段作为默认值。
   d. 选择 创建存储桶 。

   有关 Amazon S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶和 Amazon Simple Storage Service 用户指南中的使用 Amazon S3 存储桶。
2. 创建两个 Amazon SQS 队列：
   a. 打开 Amazon SQS 控制台。
   b. 选择创建队列。
   c. 对于 Type（类型），选择 Standard（标准）。
   d. 对于名称，请输入 **ticketPaymentQueue**。
   e. 在 Access policy（访问策略）下，对于 Choose method（选择方法），选择 Advanced（高级）。
   f. 在 JSON 策略框中，粘贴以下策略：

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

在此访问策略中，将 Amazon Web Services 账户 数字 (*123456789012*) 替换为您自己的数字，然后相应地更改 Amazon 区域 (*us-east-1*)。

g. 选择创建队列。

h. 重复这些步骤以创建第二个名为 **ticketFraudQueue** 的 SQS 队列。

有关创建 SQS 队列的更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的创建 Amazon SQS 队列（控制台）。

3. 创建 SNS 主题：

   a. 打开 Amazon SNS 控制台中的主题页面。

   b. 选择创建主题。

   c. 在 Details（详细信息）下，对于 Type（类型），选择 Standard（标准）。

   d. 对于名称，请输入 **ticketTopic**。

   e. 选择创建主题。

   有关创建 SNS 主题的更多信息，请参阅 创建 Amazon SNS 主题。

4. 同时为两个 SQS 队列订阅 SNS 主题。

   a. 在 Amazon SNS 控制台上的在 ticketTopic 主题的详细信息页面，选择 Create subscription（创建订阅）。

   b. 在 Details（详细信息）下，对于 Protocol（方案），选择 Amazon SQS。

   c. 对于终端节点，选择队列的亚马逊资源名称 (ARN)。ticketPaymentQueue

   d. 选择创建订阅。

e. 重复这些步骤，使用队列的 ARN 创建第二个订阅。ticketFraudQueue

有关订阅 SNS 主题的更多信息，请参阅 在 Amazon SNS 主题中创建订阅。您还可以从 Amazon SQS 控制台为 SQS 队列订阅 SNS 主题。有关更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的将 Amazon SQS 队列订阅到 Amazon SNS 主题（控制台）。

您已为此示例使用案例创建了初始资源。要继续，请参阅 为 Amazon SNS 消息归档设置 Firehose 传输流。

为 Amazon SNS 消息归档设置 Firehose 传输流

本主题介绍如何为消息存档和分析示例用例创建 Amazon Data Firehose 传输流。

创建 Firehose 传输流

1. 打开 Amazon Kinesis 服务控制台。
2. 选择 Firehose，然后选择创建传输流。
3. 在 New delivery stream（新传输流）页面上，对于 Delivery stream name（传输流名称），输入 **`ticketUploadStream`**，然后选择 Next（下一步）。
4. 在 Process records（处理记录）页面上，选择 Next（下一步）。
5. 在 Choose a destination（选择目标）页面上，执行以下操作：

   a. 对于 Destination（目标），选择 Amazon S3。

   b. 在 S3 destination（S3 目标）下，对于 S3 bucket（S3 存储桶），选择您最初创建的 S3 存储桶。

   c. 选择下一步。
6. 在 Configure settings（配置设置）页面上，对于 S3 缓冲区条件，执行以下操作：

   - 对于 Buffer size（缓冲区大小），输入 **1**。
   - 对于 Buffer interval（缓冲区时间间隔），输入 **60**。

   通过将这些值用于 Amazon S3 缓冲区，您可以快速测试配置。满足的第一个缓冲条件将触发至 S3 存储桶的数据传输。
7. 在配置设置页面上，对于权限，选择创建自动分配所需权限的 Amazon Identity and Access Management (IAM) 角色。然后选择下一步。

8. 在 Review（审核）页面上，选择 Create delivery stream（创建传输流）。
9. 从 Kinesis Data Firehose 交付流页面中，选择你刚刚ticketUploadStream创建的传输流（ ）。在 Details（详细信息）选项卡上，记下流的 Amazon Resource Name (ARN) 以供日后使用。

有关创建传输流的更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的创建 Amazon Data Firehose 传输流。有关创建 IAM 角色的更多信息，请参阅 I A M 用户指南中的创建角色以向 Amazon 服务委派权限。

您已创建具有所需权限的 Firehose 传输流。要继续，请参阅 为 Firehose 传输流订阅 Amazon SNS 主题。

为 Firehose 传输流订阅 Amazon SNS 主题

本主题介绍如何为消息存档和分析示例用例创建以下资源：

- 允许亚马逊 SNS 订阅在 Amazon Data Firehose 传输流中保存记录的 Amazon Identity and Access Management (IAM) 角色。
- Firehose 直播订阅亚马逊 SNS 主题。

要为 Amazon SNS 订阅创建 IAM 角色

1. 打开 IAM 控制台的角色页面。
2. 选择 Create role（创建角色）。
3. 对于Select type of trusted entity（选择受信任实体的类型），选择 Amazon service（服务）。
4. 对于 Choose a use case（选择使用案例），选择 SNS。然后选择下一步：权限。
5. 选择下一步：标签。
6. 选择下一步：审核。
7. 在审核页面上，对于角色名称，输入 **ticketUploadStreamSubscriptionRole**。然后选择创建角色。
8. 创建角色后，选择其名称 (ticketUploadStreamSubscriptionRole)。
9. 在角色的 Summary（摘要）页面上，选择 Add inline policy（添加内联策略）。
10. 在 Create policy（创建策略）页面上，选择 JSON 选项卡，然后将以下策略粘贴到文本框中：

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

在本政策中，将 Amazon Web Services 账户 数字 (*123456789012*) 替换为自己的数字，然后相应地更改 Amazon 区域 (*us-east-1*)。

11. 选择查看策略。
12. 在 Create policy（创建策略）页面上，对于 Name（名称），输入 **FirehoseSnsPolicy**。然后选择创建策略。
13. 在角色的 Summary（摘要）页面上，记下角色 ARN供稍后使用。

有关创建 IAM 角色的更多信息，请参阅 I A M 用户指南中的创建角色以向 Amazon 服务委派权限。

为 Firehose 传输流订阅 SNS 主题

1. 打开 Amazon SNS 控制台中的主题页面。
2. 在 Subscriptions（订阅）选项卡上，选择 Create subscription（创建订阅）。
3. 在详细信息下，对于协议，选择 Amazon Data Firehose。
4. 在 En d point 中，输入您之前创建的ticketUploadStream传输流的亚马逊资源名称 (ARN)。例如，输入 **arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream**。
5. 对于订阅角色 ARN，请输入您之前创建的 ticketUploadStreamSubscriptionRoleIAM 角色的 ARN。例如，输入 **arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole**。

6. 选择 Enable raw message delivery（启用原始消息传输）复选框。
7. 选择创建订阅。

您已创建 IAM 角色和 SNS 主题订阅。要继续，请参阅 测试和查询 Amazon SNS 配置以实现有效的数据管理。

测试和查询 Amazon SNS 配置以实现有效的数据管理

本主题介绍如何通过向 Amazon SNS 主题发布消息来测试消息存档和分析示例用例。这些说明包括一个示例查询，您可以运行并根据自己的需求对其进行调整。

测试配置

1. 打开 Amazon SNS 控制台中的主题页面。
2. 选择 **ticketTopic** 主题。
3. 选择发布消息。
4. 在 Publish message to topic（发布消息到主题）页面上，为消息正文输入以下内容。在消息结尾处添加换行符。

   ```
   {"BookingDate":"2020-12-15","BookingTime":"2020-12-15
    04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
   ```

   保留所有其他选项作为默认值。
5. 选择发布消息。

   有关发布消息的更多信息，请参阅 发布 Amazon SNS 消息。
6. 在 60 秒的传输流间隔后，打开 Amazon Simple Storage Service (Amazon S3) 控制台并选择您最初创建的 Amazon S3 存储桶。

   存储桶中会显示已发布的消息。

要查询数据

1. 打开 Amazon Athena 控制台。
2. 运行查询。

   例如，假设 `default` schema 中的 `notifications` 表包含下列数据：

```
{"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"}
```

要查找最佳目标，请运行以下查询：

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

要查询在特定日期和时间范围内售出的票证，请运行如下所示的查询：

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

您可以根据自己的需求调整两个示例查询。有关使用 Athena 运行查询的更多信息，请参阅 Amazon Athena 用户指南中的入门。

清理

为避免在完成测试后产生使用费用，请删除您在本教程中创建的以下资源：

- Amazon SNS 订阅
- Amazon SNS 主题
- Amazon Simple Queue Service（Amazon SQS）队列
- Amazon S3 存储桶
- Amazon Data Firehose 传输流
- Amazon Identity and Access Management (IAM) 角色和策略

使用 Amazon CloudFormation 模板自动归档 Amazon SNS 消息

要自动部署 Amazon SNS 消息存档和分析示例使用案例，您可以使用以下 YAML 模板：

```
---
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
```

将 Amazon SNS 通知扇出到 Lambda 函数以进行自动处理

亚马逊 SNS 与集成 Amazon Lambda，允许您触发 Lambda 函数以响应亚马逊 SNS 通知。将消息发布到 函数订阅的 SNS 主题时，将使用已发布消息的负载调用 Lambda 函数。Lambda 函数接收消息负载作为输入参数，可以操作消息中的信息、将消息发布到其他 SNS 主题或将消息发送到其他服务。Amazon

此外，Amazon SNS 还支持针对发送到 Lambda 终端节点的消息通知的消息传输状态属性。有关更多信息，请参阅 Amazon SNS 消息传输状态。

主题

- 跨区域将 Amazon SNS 与 Lambda 函数集成的先决条件
- 将 Lambda 函数订阅到 Amazon SNS 主题

跨区域将 Amazon SNS 与 Lambda 函数集成的先决条件

要使用 Amazon SNS 通知调用 Lambda 函数，您需要以下信息：

- Lambda 函数
- 一个 Amazon SNS 主题

有关创建 Lambda 函数以与 Amazon SNS 结合使用的信息，请参阅将 Lambda 与 Amazon SNS 结合使用。有关创建 Amazon SNS 主题的信息，请参阅创建主题。

当您使用 Amazon SNS 将消息从选择加入区域传送到默认启用的区域时，您必须通过将委托人 `sns.amazonaws.com` 替换为 `sns.<opt-in-region>.amazonaws.com` 来更改在 Amazon Lambda 函数中创建的策略。

例如，如果您希望为美国东部（弗吉尼亚北部）的 Lambda 函数订阅亚太地区（香港）的 SNS 主题，请将 AWS Lambda 函数策略中的委托人更改为 `sns.ap-east-1.amazonaws.com`。选择加入的区域包括 2019 年 3 月 20 日之后推出的任何区域，包括亚太地区（香港）、中东（巴林）、欧盟（米兰）和非洲（开普敦）。2019 年 3 月 20 日之前推出的区域默认情况下处于启用状态。

> **Note**
>
> Amazon 不支持从默认启用的区域向 Lambda 跨区域交付到可选区域。此外，也不支持将 SNS 消息从选择加入区域到其他选择加入区域的跨区域转发。

将 Lambda 函数订阅到 Amazon SNS 主题

本主题介绍如何将 Lambda 函数订阅 Amazon SNS 主题，从而使该函数能够由已发布的消息触发。

1. 登录 Amazon SNS 控制台。

2. 在导航面板上，选择主题。
3. 在主题 页上，选择一个主题。
4. 在订阅部分中，选择创建订阅。
5. 在创建订阅页上的详细信息部分，执行以下操作：
   a. 验证所选主题 ARN。
   b. 对于协议，请选择 Amazon Lambda。
   c. 对于终端节点，输入函数的 ARN。
   d. 选择创建订阅。

将消息发布到 函数订阅的 SNS 主题时，将使用已发布消息的负载调用 Lambda 函数。有关如何使用 Amazon Lambda 亚马逊 SNS 的信息（包括教程），请参阅使用 Amazon SN Amazon Lambda S。

将 Amazon SNS 通知扇出到 Amazon SQS 队列进行异步处理

Amazon SNS 与 Amazon Simple Queue Service (Amazon SQS) 密切配合。这些服务为开发人员提供了不同的益处。Amazon SNS 允许应用程序通过“推送”机制将时间要求严格的消息发送到多个订阅者，无需定期检查或“轮询”更新。Amazon SQS 是分布式应用程序通过轮询模型交换消息的消息队列服务，它可用于解耦发送和接收组件，而无需每个组件同时可用。通过将 Amazon SNS 和 Amazon SQS 配合使用，可以将消息发送到要求立即通知事件的应用程序，也可以在 Amazon SQS 队列中存留消息以供其他应用程序稍后进行处理。

为 Amazon SQS 队列订阅 Amazon SNS 主题时，您可以向该主题发布消息，Amazon SNS 会向已订阅队列发送 Amazon SQS 消息。Amazon SQS 消息包括已向主题发布的相关主题和消息，包括有关 JSON 文档中消息的元数据。Amazon SQS 消息与以下 JSON 文档相似。

```
{
   "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"
}
```

为 Amazon SQS 队列订阅 Amazon SNS 主题

要允许 Amazon SNS 主题向亚马逊 SQS 队列发送消息，请选择以下选项之一：

- 使用 Amazon SQS 控制台，这简化了过程。有关更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的将 Amazon SQS 队列订阅到 Amazon SNS 主题。
- 使用以下步骤：
  1. 获取您要发送消息的目标队列的 Amazon 资源名称（ARN）以及要为队列订阅的主题。
  2. 向 Amazon SNS 主题授予 `sqs:SendMessage` 权限，以便该主题向队列发送消息。
  3. 为队列订阅 Amazon SNS 主题。
  4. 向 IAM 用户或 Amazon Web Services 账户 授予适当权限，使之能够发布到 Amazon SNS 主题和阅读来自 Amazon SQS 队列的消息。
  5. 通过向主题发布消息，并读取来自队列的消息，对其进行测试。

要了解如何设置主题以向位于不同的 Amazon账户中的队列发送消息，请参阅将 Amazon SNS 消息发送到不同账户中的 Amazon SQS 队列。

要查看用于创建向两个队列发送消息的主题的 Amazon CloudFormation 模板，请参阅通过 Amazon CloudFormation自动实现从 Amazon SNS 到 Amazon SQS 的消息收发。

步骤 1：获取队列的 ARN 和主题

为队列订阅主题时，需要队列 ARN 的副本。同样，为主题授予向队列发送消息的权限时，需要主题 ARN 的副本。

要获取队列 ARN，您可以使用亚马逊 SQS 控制台或 GetQueueAttributesAPI 操作。

从 Amazon SQS 控制台获取队列 ARN

1. 登录 Amazon Web Services Management Console 并打开 Amazon SQS 控制台，网址为。https://console.aws.amazon.com/sqs/
2. 选定您要获取 ARN 的队列框。
3. 从详细信息部分中，复制 ARN 值，然后使用该值订阅 Amazon SNS 主题。

要获取主题 ARN，您可以使用 Amazon SNS 控制台、`sns-get-topic-attributes` 命令或 `GetQueueAttributes` API 操作。

从 Amazon SNS 控制台获取主题 ARN

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择要获取其 ARN 的主题。
3. 从详细信息部分中，复制 ARN 值，这样就可使用该值为 Amazon SNS 主题授予向队列发送消息的权限。

步骤 2. 为向 Amazon SQS 队列发送消息的 Amazon SNS 主题授予权限

针对能够向队列发送消息的 Amazon SNS 主题，您必须对队列设置策略，以允许 Amazon SNS 主题执行 `sqs:SendMessage` 操作。

获取主题和队列之后，方可为队列订阅主题。如果您尚未创建主题或队列，请您立刻创建。有关更多信息，请参阅创建主题，并参阅《Amazon Simple Queue Service 开发人员指南》中的创建队列。

要对队列设置策略，您可以使用 Amazon SQS 控制台或 SetQueueAttributesAPI 操作。开始前，请确保针对您想要允许向队列发送消息的主题，您已拥有其 ARN。如果要为队列订阅多个主题，策略必须对于每个主题包含一个 `Statement` 元素。

使用 Amazon SQS 控制台对队列设置 SendMessage 策略

1. 登录 Amazon Web Services Management Console 并打开 Amazon SQS 控制台，网址为。https://console.aws.amazon.com/sqs/
2. 选择要设置其策略的队列的框，然后依次选择 Access policy（访问策略）选项卡、编辑。
3. 在 Access policy（访问策略）部分中，定义谁可以访问您的队列。

   - 添加允许主题操作的一项条件。
   - 将 `Principal` 设置为 Amazon SNS 服务，如下例所示。
   - 使用 `aws:SourceArn` 或 `aws:SourceAccount` 全局条件键来防止混淆代理场景。要使用这些条件键，请将值设置为主题的 ARN。如果您的队列订阅了多个主题，则可以改用 `aws:SourceAccount`。

   例如，以下策略允许 MyTopic 向发送消息 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"
        }
      }
    }
  ]
}
```

步骤 3. 为队列订阅 Amazon SNS 主题

必须为队列订阅 Amazon SNS 主题后，方可通过主题向队列发送消息。您可以按队列 ARN 指定队列。要订阅主题，您可以使用 Amazon SNS 控制台、`sns-subscribe` CLI 命令或 `Subscribe` API 操作。开始前，必须确保您拥有要订阅队列的 ARN。

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。
3. 在主题 页上，选择一个主题。
4. 在该***MyTopic***页面的订阅页面中，选择创建订阅。
5. 在创建订阅页上的详细信息部分，执行以下操作：

   a. 验证主题 ARN。

   b. 对于 Protocol（协议），选择 Amazon SQS。

   c. 对于 Endpoint（终端节点），输入 Amazon SQS 队列的 ARN。

   d. 选择创建订阅。

   确认订阅后，您新建订阅的“Subscription ID”将显示其订阅 ID。如果订阅由队列所有者创建，则订阅将自动确认，且订阅立刻可用。

一般情况下，您可以在您自己的账户中为您的队列订阅您自己的主题。但是，您还可以通过另一账户为队列订阅主题。如果创建订阅的用户并非队列所有者（例如，如果账户 A 的用户为账户 B 中的队列订阅账户 A 中的主题），则必须对订阅进行确认。有关通过不同账户订阅队列和确认订阅的更多信息，请参阅 将 Amazon SNS 消息发送到不同账户中的 Amazon SQS 队列。

步骤 4：向用户授予对适当主题和队列操作的权限

您应使用 Amazon Identity and Access Management (IAM) 仅允许相应的用户向 Amazon SNS 主题发布消息以及从 Amazon SQS 队列中读取/删除消息。有关针对 IAM 用户控制主题和队列操作的更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的 将基于身份的策略用于 Amazon SNS 和 Amazon SQS 中的 Identity and Access Management。

可以采取两种方式控制对主题或队列的访问：

- 添加策略至 IAM 用户或群组。为用户授予主题或队列权限的最简单方式就是创建群组，并为该群组添加适当策略，然后向此群组添加用户。相比较而言，向群组添加或删除用户，比追踪您为单独用户而设定的各项策略要简单得多。
- 添加策略至主题或队列。如果您想将某个主题或队列的权限授予其他 Amazon 账户，那么唯一的方法就是添加一个以 Amazon Web Services 账户 您想要授予权限的委托人为其委托人的策略。

绝大多数情况下，您应使用第一种方法（通过向群组添加或删除适当用户的方式，向群组添加策略，管理用户权限）。如果您需要向另一账户的用户授予权限，那么应使用第二种方法。

添加策略至 IAM 用户或群组

如果您向 IAM 用户或群组添加了以下策略，则需要向该用户或群组中的成员授予对该主题执行`sns:Publish`操作的权限 MyTopic。

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

如果您向 IAM 用户或群组添加了以下策略，则需要向该用户或群组中的成员授予对队列 MyQueue 1 `sqs:ReceiveMessage` 和 MyQueue 2 执行和`sqs:DeleteMessage`操作的权限。

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

添加策略至主题或队列

以下策略示例显示如何为主题和队列授予另一账户授权。

> **Note**
>
> 当您授予其他人 Amazon Web Services 账户 访问您账户中某项资源的权限时，也就是向对该资源拥有管理员级访问权限（通配符访问权限）的 IAM 用户授予权限。此操作将自动拒绝其他账户中的所有其他 IAM 用户访问您的资源。如果您要向该 Amazon Web Services 账户 中的特定 IAM 用户授予访问您的资源的权限，那么拥有管理员级访问权限的账户或 IAM 用户必须将此项资源的访问权限委派给这些 IAM 用户。有关跨账户委派的更多信息，请参阅使用 IAM 指南中的启用跨账户访问。

如果您向账户 123456789012 MyTopic 中的主题添加了以下策略，则需要向账户 111122223333 授予对该主题执行操作的权限。`sns:Publish`

```
{
  "Statement": [
    {
      "Effect": "Allow",
```

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

如果您向账户 123456789012 MyQueue 中的队列添加了以下策略，则需要授予账户 111122223333 对该队列执行和操作的权限。`sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

步骤 5：测试主题的队列订阅

通过发布主题，查看主题向队列发送的消息，可以测试主题的队列订阅情况。

利用 Amazon SNS 控制台发布主题

1. 使用有权发布主题的 Amazon Web Services 账户 或 IAM 用户的证书，登录 Amazon Web Services Management Console 并打开 Amazon SNS 控制台，网址为。https://console.aws.amazon.com/sns/
2. 在导航面板上，选择主题，然后选择发布到主题。

3. 在主题框中，输入主题（例如 **Testing publish to queue**），在消息框中，输入一些文字（例如 **Hello world!**），然后选择发布消息。界面将显示如下消息：Your message has been successfully published。

利用 Amazon SQS 控制台查看来自主题的消息

1. 使用有权查看队列中消息的 Amazon Web Services 账户 或 IAM 用户的证书，登录 Amazon Web Services Management Console 并打开 Amazon SQS 控制台，网址为。https://console.aws.amazon.com/sqs/
2. 选择已订阅该主题的 queue（队列）。
3. 选择 Send and receive messages（发送和接收消息），然后选择 Poll for messages（轮询消息）。界面将显示“Notification”类型消息。
4. 在正文列中，选择更多详细信息。“Message Details”框包括 JSON 文档，此文档包含您发布主题的主题和消息。消息与以下 JSON 文档相似。

```
{
  "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. 选择关闭。您已经成功发布到一个主题，该主题向队列发送通知消息。

通过 Amazon CloudFormation自动实现从 Amazon SNS 到 Amazon SQS 的消息收发

Amazon CloudFormation 允许您使用模板文件将 Amazon 资源集合作为一个单元一起创建和配置。通过本部分提供的模板示例，您可以轻松部署向队列发布的主题。模板通过以下操作协助您处理设置步

骤，即创建两个队列、创建订阅队列的主题、添加策略至队列，以便主题能够向队列发送消息，以及创建 IAM 用户和群组，控制对这些资源的访问。

有关使用 Amazon CloudFormation 模板部署 Amazon 资源的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 “入门”。

使用 Amazon CloudFormation 模板在中设置主题和队列 Amazon Web Services 账户

该模板示例创建一个 Amazon SNS 主题，该主题能够向具备相应权限的两个 Amazon SQS 队列发送信息，以便一个 IAM 组的成员能够向该主题发布消息，而另一个组则从该队列读取消息。模板还用于创建可添加至各个群组的 IAM 用户。

您可以将模板内容复制到文件中，您也可以从 “模板” 页面下载Amazon CloudFormation 模板。在模板页面上，选择按 Amazon 服务浏览示例模板，然后选择亚马逊简单队列服务。

我的设置SNSTopic 为发布到两个已订阅的终端节点，即两个 Amazon SQS 队列MyQueue（1 MyQueue 和 2）。 MyPublishTopicGroup 是一个 IAM 群组，其成员有权SNSTopic 使用 Publish API 操作或 sns-p ub lish 命令向 My 发布内容。该模板创建 IAM 用户 MyPublishUser ， MyQueueUser 并为他们提供登录配置文件和访问密钥。通过该模板创建堆栈的用户将指定登录界面密码作为输入参数。该模板使用 MyPublishUserKey 和为两个 IAM 用户创建访问密钥 MyQueueUserKey。AddUserToMyPublishTopicGroup 添加 MyPublishUser 到， MyPublishTopicGroup 这样用户就可以拥有分配给该组的权限。

我RDMessageQueueGroup 是一个 IAM 群组，其成员有权使用和 DeleteMessageAPI 操作从两个 Amazon SQS 队列中读取ReceiveMessage和删除消息。 AddUserToMyQueueGroup 添加 MyQueueUser 到 “我的”，RDMessageQueueGroup 这样用户就可以拥有分配给该组的权限。MyQueuePolicy 为 My 分配SNSTopic 向这两个队列发布通知的权限。

以下清单显示了 Amazon CloudFormation 模板的内容。

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

```
    }
  }
}
```

将 Amazon SNS 通知扇出到 HTTPS 端点

您可以使用 Amazon SNS 向一个或多个 HTTP 或 HTTPS 端点发送通知消息。为终端节点订阅主题时，您可以向主题发布通知，Amazon SNS 将发送 HTTP POST 请求，向已订阅终端节点传递通知内容。订阅终端节点时，您可以选择 Amazon SNS 是否使用 HTTP 或 HTTPS 向终端节点发送 POST 请求。如果您使用 HTTPS，则可以利用 Amazon SNS 对以下功能的支持：

- 服务器名称指示 (SNI) - 这使 Amazon SNS 可以支持需要 SNI 的 HTTPS 终端节点，如需要多个证书来承载多个域的服务器。有关 SNI 的更多信息，请参阅服务器名称指示。

- 基本和摘要式访问身份验证 - 这使您可以在 HTTPS URL 中为 HTTP POST 请求指定用户名和密码，如 `https://user:password@domain.com` 或 `https://user@domain.com`。在使用 HTTPS 建立的 SSL 连接上，会对该用户名和密码进行加密。只有域名以明文形式发送。有关基本和摘要式访问身份验证的更多信息，请参阅 RFC-2617。

> **Important**
>
> Amazon SNS 目前不支持私有 HTTP(S) 端点。
> 对于您 URLs 已授予 API 访问权限的委托人，只能从 Amazon `GetSubscriptionAttributes` SNS API 操作中检索 HTTPS。

> **Note**
>
> 客户端服务必须能够支持 `HTTP/1.1 401 Unauthorized` 标头响应

此项请求包含已向主题发布的相关主题和消息，包括 JSON 文档中通知的元数据。此项请求与以下 HTTP POST 请求相似。有关 HTTP 标头和请求正文 JSON 格式的详细信息，请参阅 HTTP/HTTPS 标题 和 HTTP/HTTPS 通知 JSON 格式。

> **Note**
>
> Amazon SNS 将所有 5XX 错误和 429（发送的请求过多）错误视为可重试。这些错误受配送政策的约束。所有其他错误都被视为永久失败，不会尝试重试。

```
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"
}
```

为 HTTPS 端点订阅 Amazon SNS 主题

本主题介绍如何将 HTTP/S 终端节点订阅到亚马逊 SNS 主题。

主题

- 步骤 1：确保您的终端节点已准备好处理 Amazon SNS 消息

- 步骤 2：订阅 Amazon SNS 主题 HTTP/HTTPS 终端节点
- 第 3 步：确认您的 Amazon SNS 订阅
- 步骤 4：可选 - 设置 Amazon SNS 订阅的传送策略
- 步骤 5：可选 - 授予用户发布 Amazon SNS 主题的权限
- 步骤 6：向 HTTP/HTTPS 端点发送 Amazon SNS 消息

步骤 1：确保您的终端节点已准备好处理 Amazon SNS 消息

确保供 Amazon SNS 使用发送订阅确认和通知消息的 HTTP 或 HTTPS 终端节点能够处理 HTTP POST 请求之后，方可订阅相关主题的 HTTP 或 HTTPS 终端节点。一般情况下，这要求创建和部署 Web 应用程序（例如，若您的终端主机正在通过 Apache 和 Tomcat 运行 Linux，则为 Java servlet），用于处理来自 Amazon SNS 的 HTTP 请求。当您订阅 HTTP 终端节点时，Amazon SNS 会向其发送一条订阅确认请求。当您创建订阅时，终端节点必须已经准备好接收和处理此请求，因为 Amazon SNS 会同时发送此请求。在您确认订阅前，Amazon SNS 不会向终端节点发送消息。订阅确认后，在已订阅主题上执行发布操作时，Amazon SNS 会向终端节点发送通知。

设置您的终端节点，处理订阅确认和通知消息

1. 您的代码将读取 Amazon SNS 向您的终端节点发送的 HTTP POST 请求的 HTTP 标头。您的代码将查找标头字段 `x-amz-sns-message-type`，此标头字段将显示 Amazon SNS 向您发送的消息类型。查看标头后，您可以确定消息类型，而无需分析 HTTP 请求正文。您需要处理如下两种类型消息：`SubscriptionConfirmation` 和 `Notification`。仅当从主题中删除订阅时，方使用 `UnsubscribeConfirmation` 消息。

   有关 HTTP 标头的详细信息，请参阅 HTTP/HTTPS 标题。以下 HTTP POST 请求为订阅确认消息的一个示例。

```
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. 您的代码将分析 HTTP POST 请求正文中的 JSON 文档和 content-type text/plain，以读取构成 Amazon SNS 消息的名称/值对。使用 JSON 分析器将控制字符的转义字符转换回 ASCII 字符值（例如，将 \n 转换成换行符）。您可以使用现有 JSON 分析器（例如 Jackson JSON 处理器）或者由您自己写入。要将主题和消息字段中的文本作为有效 JSON 发送，Amazon SNS 必须将部分控制字符转换成可包含在 JSON 文档中的转义字符。向您的终端节点发送的 POST 请求正文中包括 JSON 文档，当您接收到该文档时，若您想要获取发布到主题上的原始主题和消息的精确字符，则必须将转义字符转换回其原始字符值。由于签名采用了原始形式的消息和主题作为待签字符串的一部分，因此如果您想要验证通知签名，则上述操作非常重要。
3. 您的代码应对 Amazon SNS 发送的通知、订阅确认或取消订阅确认消息进行验证。使用 Amazon SNS 消息所含信息，终端节点可以重新创建签名，以便您可以通过将自己的签名与 Amazon SNS 随消息发送的签名进行匹配，来验证消息的内容。有关验证消息签名的更多信息，请参阅 验证 Amazon SNS 消息签名。
4. 根据标头字段 `x-amz-sns-message-type` 指定的类型，您的代码将读取 HTTP 请求正文所含的 JSON 文档，并处理该消息。这里是处理两大主要消息类型的指导原则：

   SubscriptionConfirmation

   读取 `SubscribeURL` 值，访问此 URL。要确认订阅并通过此终端节点接收通知，必须访问 `SubscribeURL` URL（例如，向此 URL 发送 HTTP GET 请求）。参阅上一步中 HTTP 请求示例，查看 `SubscribeURL` 相关情况。有关 `SubscriptionConfirmation` 消息格式的更多信息，请参阅 HTTP/HTTPS 订阅确认 JSON 格式。访问 URL 时，您将获取与以下 XML 文档相似的响应。文档会在 `ConfirmSubscriptionResult` 元素内返回终端节点的订阅 ARN。

```
<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>
```

除了访问之外SubscribeURL，您还可以使用在SubscriptionConfirmation消息中Token设置相应值的ConfirmSubscription操作来确认订阅。如果您仅允许主题所有者和订阅所有者拥有取消订阅终端节点的权限，那么您可以通过 Amazon 签名调用 ConfirmSubscription 操作。

通知

读取 Subject 和 Message 值，获取已向主题发布的通知信息。

有关 Notification 消息格式的详细信息，请参阅 HTTP/HTTPS 标题。以下 HTTP POST 请求为向终端节点 example.com.发送的通知消息的示例。

```
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. 确保您的终端节点已通过适当的状态代码对来自 Amazon SNS 的 HTTP POST 消息作出响应。此项连接将在大约 15 秒内超时。在连接超时前，如果您的端点不响应，或者您的端点返回的状态代码超出 200–4xx 范围，那么 Amazon SNS 会认为消息发送尝试失败。
6. 确保您的代码能够处理 Amazon SNS 的消息发送重试。如果 Amazon SNS 未能接收到从终端节点发出的发送成功响应，它将会尝试再次发送消息。这适用于包括订阅确认消息在内的所有消息。默认情况下，如果消息初次发送失败，那么 Amazon SNS 会通过失败尝试期间设定为 20 秒的延时进行多达 3 次的尝试。

   > **Note**
   >
   > 消息请求大约 15 秒后超时。这表示，如果因超时引起消息发送失败，那么 Amazon SNS 将在前一次发送尝试后 35 秒左右重新发送。您可以为终端节点设置不同的发送策略。

   Amazon SNS 使用 `x-amz-sns-message-id` 标头字段来唯一标识发布到 Amazon SNS 主题的每条消息。通过将已处理 IDs 的消息与传入的消息进行比较，可以确定该消息是否为重试尝试。
7. 如果您要订阅 HTTPS 终端节点，请确保终端节点具备由可信赖证书颁发机构 (CA) 颁发的服务器证书。 将仅向具有 所信任 CA 签署的服务器证书的 HTTPS 终端节点发送消息。Amazon SNS 将仅向具有 Amazon SNS 所信任 CA 签署的服务器证书的 HTTPS 终端节点发送消息。
8. 对您已创建的代码进行部署，以便接收 Amazon SNS 消息。当您订阅终端节点时，该终端节点必须准备好至少接收订阅确认消息。

步骤 2：订阅 Amazon SNS 主题 HTTP/HTTPS 终端节点

要通过主题向 HTTP 或 HTTPS 终端节点发送消息，必须为终端节点订阅 Amazon SNS 主题。您可以通过终端节点的 URL 指定终端节点。要订阅主题，您可以使用 Amazon SNS 控制台、sns-subscribe 命令或 Subscribe API 操作。开始操作前，应确保拥有想要订阅终端节点的 URL，并且该终端节点按照步骤 1 所述已准备好接收确认和通知消息。

利用 Amazon SNS 控制台为 HTTP 或 HTTPS 终端节点订阅主题

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择订阅。
3. 选择 Create subscription（创建订阅）。
4. 在 Protocol 下拉列表中，选择 HTTP 或 HTTPS。
5. 在终端节点框中，粘贴您希望主题向其发送消息的终端节点的 URL，然后选择创建订阅。
6. 将显示确认消息。选择关闭。

   将显示您的新订阅的订阅 ID PendingConfirmation。当您确认订阅时，Subscription ID (订阅 ID) 将显示订阅 ID。

第 3 步：确认您的 Amazon SNS 订阅

要确认您的 Amazon SNS 订阅，请按照以下步骤操作，确保您的终端节点能够成功接收消息。此过程包括设置您的终端节点以处理传入的确认消息、检索确认 URL 以及确认订阅。根据您的设置，您可以自动或手动确认订阅。

1. 订阅亚马逊 SNS 主题后，Amazon SNS 会向您的终端节点发送确认消息。此消息`SubscribeURL`包含必须用于确认订阅的。
2. 您的终端节点必须设置为监听来自 Amazon SNS 的传入消息。当确认消息到达时，**`SubscribeURL`**从消息中提取该消息。
3. 获得后`SubscribeURL`，您可以通过以下两种方式之一确认订阅：

   - 自动确认-您的终端节点可以通过向发送 HTTP GET 请求来自动确认订阅`SubscribeURL`。

     此方法不需要手动干预。
   - 手动确认-如果未设置自动确认，请复制确认消息**`SubscribeURL`**中的内容，然后将其粘贴到浏览器的地址栏中。

     这将手动确认订阅。
4. 您也可以使用 Amazon SNS 控制台验证订阅状态：

   a. 登录 Amazon SNS 控制台。

   b. 在导航窗格中，选择订阅。

   c. 在列表中找到您的订阅。

- 如果已确认，`SubscriptionArn`将显示。
- 如果仍未确认，它将显示为。`PendingConfirmation`

步骤 4：可选 - 设置 Amazon SNS 订阅的传送策略

默认情况下，如果消息初次发送失败，那么 Amazon SNS 会通过失败尝试期间设定为 20 秒的延时进行多达 3 次的尝试。按照步骤 1 所述，您的终端节点应包括能够处理已重试消息的代码。通过设置主题或订阅的发送策略，您可以控制 Amazon SNS 即将重试失败消息的频率和间隔。您还可以在 `DeliveryPolicy` 中指定 HTTP/S 通知的内容类型。有关更多信息，请参阅 创建 HTTP/S 传输策略。

步骤 5：可选 - 授予用户发布 Amazon SNS 主题的权限

默认情况下，只有主题所有者才拥有发布主题的权限。要允许其他用户或应用程序向该主题发布内容，您应使用 Amazon Identity and Access Management (IAM) 授予该主题的发布权限。有关授予 IAM 用户 Amazon SNS 操作权限的更多信息，请参阅 将基于身份的策略用于 Amazon SNS。

可以采取两种方式控制对主题的访问：

- 添加策略至 IAM 用户或群组。向用户授予主题权限的最简单方式就是创建群组，向该群组添加适当策略，再向此群组添加用户。相比较而言，向群组添加或删除用户，比追踪您为单独用户而设定的各项策略要简单得多。
- 添加策略至主题。如果您想向另一 Amazon 账户授予主题权限，那么唯一的方法是添加策略，并且该策略必须具备您想向其授予权限的主要 Amazon Web Services 账户 。

绝大多数情况下，您应使用第一种方法（通过向群组添加或删除适当用户的方式，向群组添加策略，管理用户权限）。如果您需要向另一账户的用户授予权限，请使用第二种方法。

如果您向 IAM 用户或群组添加了以下策略，则需要向该用户或群组中的成员授予对该主题执行`sns:Publish`操作的权限 MyTopic。

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

```
}
```

以下策略示例显示如何向主题授予另一账户权限。

> **Note**
>
> 当您授予其他人 Amazon Web Services 账户 访问您账户中某项资源的权限时，也就是向对该资源拥有管理员级访问权限（通配符访问权限）的 IAM 用户授予权限。此操作将自动拒绝其他账户中的所有其他 IAM 用户访问您的资源。如果您想向该中的特定 IAM 用户授予 Amazon Web Services 账户 访问您的资源的权限，则该账户或具有管理员级别访问权限的 IAM 用户必须将资源权限委托给这些 IAM 用户。有关跨账户委派的更多信息，请参阅使用 IAM 指南中的启用跨账户访问。

如果您向账户 123456789012 MyTopic 中的主题添加了以下策略，则需要向账户 111122223333 授予对该主题执行操作的权限。`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"
  }]
}
```

步骤 6：向 HTTP/HTTPS 端点发送 Amazon SNS 消息

您可以通过发布到主题来向主题订阅发送消息。要订阅主题，您可以使用 Amazon SNS 控制台、`sns-publish` CLI 命令或 `Publish` API。

如果您遵循 步骤 1，则在您的终端节点上部署的代码将处理通知。

利用 Amazon SNS 控制台发布主题

1. 使用有权发布主题的 Amazon Web Services 账户 或 IAM 用户的证书，登录 Amazon Web Services Management Console 并打开 Amazon SNS 控制台，网址为。https://console.aws.amazon.com/sns/

2. 在导航面板上，选择主题，然后选择一个主题。
3. 选择发布消息按钮。
4. 在主题框中，输入主题（例如，**Testing publish to my endpoint**）。
5. 在消息框中，输入一些文本（例如 **Hello world!**），然后选择 发布消息。

   界面将显示如下消息：Your message has been successfully published。

验证 Amazon SNS 消息签名

Amazon SNS 使用消息签名来确认发送到您的 HTTP 终端节点的消息的真实性。为确保消息完整性并防止欺骗，您必须在处理任何 Amazon SNS 消息之前验证签名。

什么时候应该验证 Amazon SNS 签名？

在以下情况下，您应该验证 Amazon SNS 消息签名：

- 当 Amazon SNS 向您的 HTTP (S) 终端节点发送通知消息时。
- 当 Amazon SNS 在`Subscribe`或 `Unsubscribe`API 调用后向您的终端节点发送确认消息时。

Amazon SNS 支持两个签名版本：

- SignatureVersion1 — 使用 SHA1 消息的哈希值。
- SignatureVersion2 — 使用 SHA256 消息的哈希值。这提供了更强的安全性，是推荐的选项。

要正确验证 SNS 消息签名，请遵循以下最佳实践：

- 请务必使用 HTTPS 检索签名证书，以防止未经授权的拦截攻击。
- 检查证书是否由 Amazon SNS 颁发。
- 确认证书的信任链有效。
- 证书应来自 SNS 签名的 URL。
- 未经验证，请勿信任消息中提供的任何证书。
- 拒绝任何带有意外消息的消息`TopicArn`，以防止欺骗。
- Amazon SDKs 适用于 Amazon SNS 的提供了内置的验证逻辑，从而降低了错误实施的风险。

在 Amazon SNS 主题上配置消息签名版本

在 Amazon SNS 主题上配置消息签名版本可以增强消息验证过程的安全性和兼容性。

在 `SignatureVersion` 1 (SHA1) 和 `SignatureVersion` 2 (SHA256) 之间进行选择，以控制用于签名消息的哈希算法。亚马逊 SNS 主题默认为 1。**`SignatureVersion`**您可以使用 `SetTopicAttributes`API 操作配置此设置。

使用以下示例通过以下`SignatureVersion`方式设置主题属性 Amazon CLI：

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

使用基于 HTTP 查询的请求时验证 Amazon SNS 消息的签名

在使用基于 HTTP 查询的请求时验证 Amazon SNS 消息的签名可确保消息的真实性和完整性。此过程确认消息源自 Amazon SNS，并且在传输过程中未被篡改。通过解析消息、构造正确的签名字符串以及根据可信的公钥验证签名，可以保护您的系统免受欺骗和未经授权的消息更改。

1. 从 Amazon SNS 发送的 HTTP POST 请求正文中的 JSON 文档中提取键值对。这些字段是构造待签字符串所必需的。

   - `Message`
   - `Subject`（如果存在）
   - `MessageId`
   - `Timestamp`
   - `TopicArn`
   - `Type`

   例如：

   ```
   MESSAGE_FILE="message.json"
   FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
   ```

> **Note**
>
> 如果任何字段包含转义字符（例如\n），请将其转换为其原始形式以确保完全匹配。

2. 在 Amazon SNS 消息中找到该SigningCertURL字段。该证书包含验证消息签名所需的公钥。例如：

```
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
```

3. 确保来自SigningCertURL可信 Amazon 域（例如 https://sns.us-east-1.amazonaws.com)。 出于安全考虑，拒绝任何 URLs 外部 Amazon 域名。
4. 从提供的 URL 下载 X.509 证书。例如：

```
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
```

5. 从下载的 X.509 证书中提取公钥。公钥允许您解密消息的签名并验证其完整性。例如：

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. 不同的消息类型需要在待签字符串中使用不同的键值对。确定消息类型（Amazon SNS 消息中的Type字段）以确定要包含哪些键值对：
   - 通知消息-包括Message、MessageId、Subject（如果存在）Timestamp、TopicArn、和Type。
   - SubscriptionConfirmation或UnsubscribeConfirmation 消息-包括Message、MessageIdSubscribeURL、Timestamp、Token、TopicArn、和Type。
7. Amazon SNS 要求签名字符串遵循严格的固定字段顺序进行验证。必须仅包含明确必填的字段，不能添加额外的字段。可选字段（例如Subject）只有在消息中存在时才必须包含且必须出现在必填字段顺序所定义的确切位置。例如：

```
KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
```

> **Important**
>
> 不要在字符串的末尾添加换行符。

8. 按字节排序顺序排列键值对（按键名称的字母顺序排列）。

9. 使用以下格式示例构造待签字符串：

```
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
```

通知消息示例：

```
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 示例：

```
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. 消息中的`Signature`字段采用 Base64 编码。你需要对其进行解码，将其原始二进制形式与派生的哈希值进行比较。例如：

```
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
echo "$SIGNATURE" | base64 -d > signature.bin
```

11. 使用该`SignatureVersion`字段选择哈希算法：
    - 对于 `SignatureVersion` 1，使用 SHA1（例如，`-sha1`）。
    - 对于 `SignatureVersion` 2，使用 SHA256（例如，`-sha256`）。
12. 要确认 Amazon SNS 消息的真实性，请生成构造字符串的哈希值，并使用公钥验证签名。

```
openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
 "$STRING_TO_SIGN"
```

    如果签名有效，则输出为`Verified OK`。否则，输出为`Verification Failure`。

带有错误处理功能的脚本示例

以下示例脚本可自动执行验证过程：

```
#!/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)
```

解析 Amazon SNS 消息格式

当 Amazon SNS 向 HTTP/HTTPS 终端节点发送消息时，它们同时包含 HTTP 标头和 JSON 消息正文。这些消息采用结构化格式，包括消息类型、主题 ARN、时间戳和数字签名等元数据。通过正确解析 Amazon SNS 消息，您可以确定消息是订阅确认、通知还是取消订阅确认，提取相关数据，并使用签名验证来验证真实性。

HTTP/HTTPS 标题

当 Amazon SNS 向 HTTP/HTTPS 终端节点发送订阅确认消息、通知、或者取消订阅确认消息时，它将发出一个带有多个 Amazon SNS 标头值的 POST 消息。可以使用标头值执行以下任务，例如识别消息类型而无需解析 JSON 消息主体来读取 `Type` 值。原定设置情况下，Amazon SNS 会将所有通知发送到 HTTP/S 端点，`Content-Type` 设置为 `text/plain; charset=UTF-8`。要选择除文本/纯文本（原定设置）以外的 `Content-Type`，请参阅创建 HTTP/S 传输策略中的 `headerContentType`。

`x-amz-sns-message-type`

消息类型。可能的值为 `SubscriptionConfirmation`、`Notification` 和 `UnsubscribeConfirmation`。

`x-amz-sns-message-id`

通用唯一标识符（UUID），它对于每条发布的消息是唯一的。对于 Amazon SNS 在重试期间重新发送的通知，使用原始消息的消息 ID。

x-amz-sns-topic-arn

此消息所发布到的主题的 Amazon 资源名称（ARN）。

x-amz-sns-subscription-arn

用于订阅终端节点的 ARN。

下面的 HTTP POST 标头是一条发送至 HTTP 端点的 `Notification` 消息的标头示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

HTTP/HTTPS 订阅确认 JSON 格式

在您订阅HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS终端节点之后。此条消息包含您必须访问的 `SubscribeURL` 值，以确认订阅（或者，您可以将 `Token` 值与 `ConfirmSubscription` 结合使用）。

> **Note**
>
> 直到订阅被确认后，Amazon SNS 才会向终端节点发送通知

订阅确认消息是一条 POST 消息，消息正文包含一个带以下名称/值对的 JSON 格式文档。

Type

消息类型。为订阅确认，消息类型为：`SubscriptionConfirmation`。

MessageId

通用唯一标识符（UUID），它对于每条发布的消息是唯一的。对于 Amazon SNS 在重试期间重新发送的消息，原始消息的消息 ID 被使用。

Token

您可以使用 ConfirmSubscription 操作确认订阅的一个值。或者，您只需访问SubscribeURL。

TopicArn

终端节点已经订阅该主题的 Amazon Resource Name。

Message

一个描述消息的字符串。为订阅确认，字符串看上去像这样：

```
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

为了确认订阅而必须访问的 URL。或者，您可以改为将 Token 与 ConfirmSubscription 操作结合使用以确认订阅。

Timestamp

订阅确认发出的时间 (GMT)。

SignatureVersion

所用 Amazon SNS 签名的版本。

- 如果 SignatureVersion 为 1，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 签名。
- 如果 SignatureVersion 为 2，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA256withRSA 签名。

Signature

Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 或 SHA256withRSA 签名。

SigningCertURL

用于签署消息的证书的 URL。

以下 HTTP POST 消息是发送至 HTTP 端点的一条 `SubscriptionConfirmation` 消息的示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

HTTP/HTTPS 通知 JSON 格式

当 Amazon SNS 向已订阅的 HTTP 或 HTTPS 终端节点发送一条通知时，发送到终端节点的 POST 消息具有包含一个带下列名称/值对的 JSON 格式文档的消息正文。

`Type`

消息类型。用于通知，这种类型属于`Notification`。

`MessageId`

通用唯一标识符（UUID），它对于每条发布的消息是唯一的。对于 Amazon SNS 在重试期间重新发送的通知，使用原始消息的消息 ID。

TopicArn

此消息所发布到的主题的 Amazon 资源名称（ARN）。

Subject

在将通知发布至主题时指定的 Subject 参数。

> **Note**
>
> 此参数为可选参数。如果未指定 Subject，则此 JSON 格式文档中不会显示该名称/值对。

Message

当通知发布至主题时指定的 Message 值。

Timestamp

通知发布的时间 (GMT)。

SignatureVersion

所用 Amazon SNS 签名的版本。

- 如果 SignatureVersion 为 1，则 Signature 是 Message、MessageId、Subject（如果存在）、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 签名。
- 如果 SignatureVersion 为 2，则 Signature 是 Message、MessageId、Subject（如果存在）、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA256withRSA 签名。

Signature

Message、MessageId、Subject（如果存在）、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 或 SHA256withRSA 签名。

SigningCertURL

用于签署消息的证书的 URL。

UnsubscribeURL

可以用作从主题取消订阅终端节点的 URL。如果您访问此 URL，那么 Amazon SNS 将取消订阅终端节点并不发送通知至此终端节点。

以下 HTTP POST 消息是发送至 HTTP 端点的一条 Notification 消息的示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
```

HTTP/HTTPS 取消订阅确认 JSON 格式

HTTP/HTTPS 终端节点从一个主题取消订阅之后，Amazon SNS 将向终端节点发送一条取消订阅的消息。

取消订阅确认消息是一条 POST 消息，消息正文包含一个带以下名称/值对的 JSON 格式文档。

Type

消息类型。为取消订阅确认，消息类型为`UnsubscribeConfirmation`。

MessageId

通用唯一标识符（UUID），它对于每条发布的消息是唯一的。对于 Amazon SNS 在重试期间重新发送的消息，原始消息的消息 ID 被使用。

Token

您可以使用 ConfirmSubscription 操作重新确认订阅的一个值。或者，您只需访问SubscribeURL。

TopicArn

此终端节点已经从主题取消订阅的 Amazon Resource Name (ARN)。

Message

一个描述消息的字符串。为了取消订阅确认，字符串应看起来像这样：

```
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

为了重新确认订阅，您必须访问的 URL。或者，您可以改为将 Token 与 ConfirmSubscription 操作结合使用以重新确认订阅。

Timestamp

取消订阅确认发送的时间(GMT)。

SignatureVersion

所用 Amazon SNS 签名的版本。

- 如果 SignatureVersion 为 1，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 签名。
- 如果 SignatureVersion 为 2，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA256withRSA 签名。

Signature

Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 或 SHA256withRSA 签名。

SigningCertURL

用于签署消息的证书的 URL。

以下 HTTP POST 消息是发送至 HTTP 端点的一条 `UnsubscribeConfirmation` 消息的示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37fb6...",
  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJm...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

SetSubscriptionAttributes配送策略 JSON 格式

如果您向 `SetSubscriptionAttributes` 操作发送一个请求并将 `AttributeName` 参数设置为 `DeliveryPolicy` 值，那么 `AttributeValue` 参数的值必须是一个有效的 JSON 对象。例如，以下例子将传输策略设置为 5 次重试。

```
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}}
...
```

为 AttributeValue 参数的值使用下列 JSON 格式。

```
{
    "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"
    }
}
```

有关该SetSubscriptionAttribute操作的更多信息，请SetSubscriptionAttributes访问《亚马逊简单通知服务 API 参考》。有关支持的 HTTP content-type 标头的更多信息，请参阅创建 HTTP/S 传输策略。

SetTopicAttributes 配送策略 JSON 格式

如果您向 SetTopicAttributes 操作发送一个请求并将 AttributeName 参数设置为 DeliveryPolicy 值，那么 AttributeValue 参数的值必须是一个有效的 JSON 对象。例如，以下例子将传输策略设置为 5 次重试。

```
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}}}
...
```

为 AttributeValue 参数的值使用下列 JSON 格式。

```
{
```

```
    "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"
        }
    }
}
```

有关该`SetTopicAttribute`操作的更多信息，请SetTopicAttributes访问《亚马逊简单通知服务 API 参考》。有关支持的 HTTP content-type 标头的更多信息，请参阅创建 HTTP/S 传输策略。

将 Amazon SNS 事件扇出到 Amazon Event Fork Pipelines

对于事件归档和分析，Amazon SNS 现在建议使用其与 Amazon Data Firehose 的本机集成。您可以将 Firehose 传输流订阅 SNS 主题，这样您就可以向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、亚马逊 Redshift 表、亚马逊 OpenSearch 服务（服务）等。OpenSearch 将 Amazon SNS 与 Firehose 传输流配合使用是一种完全托管且无需代码的解决方案，您无需使用任何功能。 Amazon Lambda 有关更多信息，请参阅 扇出到 Firehose 传输流。

您可以使用 Amazon SNS 构建事件驱动的应用程序，这些应用程序使用订阅者服务自动执行工作以响应发布者服务所触发的事件。此架构模式可提高服务的可重用性、可互操作性和可扩展性。但是，将事件处理分解为可满足常见事件处理要求的管道（例如，事件存储、备份、搜索、分析和重放）可能会非常耗费人力。

为了加快事件驱动型应用程序的开发，您可以订阅 Amazon SNS 主题的事件处理管道（由事件 Amazon 分叉管道提供支持）。 Amazon Event Fork Pipelines 是一套基于Amazon 无服务器应用程序模型 (SA Amazon M) 的开源嵌套应用程序，您可以直接从 Ev Amazon ent Fork Pipelines 套件（选择显示创建自定义 IAM 角色或资源策略的应用程序）将其部署到您的 Amazon 账户中。

有关 E Amazon vent Fork Pipelines 用例，请参阅部署和测试 Amazon SNS Event Fork Pipelines 示例应用程序。

主题

- Amazon 事件分叉管道的工作原理
- 部署 Amazon 事件分叉管道
- 部署和测试 Amazon SNS Event Fork Pipelines 示例应用程序
- 将 Amazon 事件分叉管道订阅到 Amazon SNS 主题

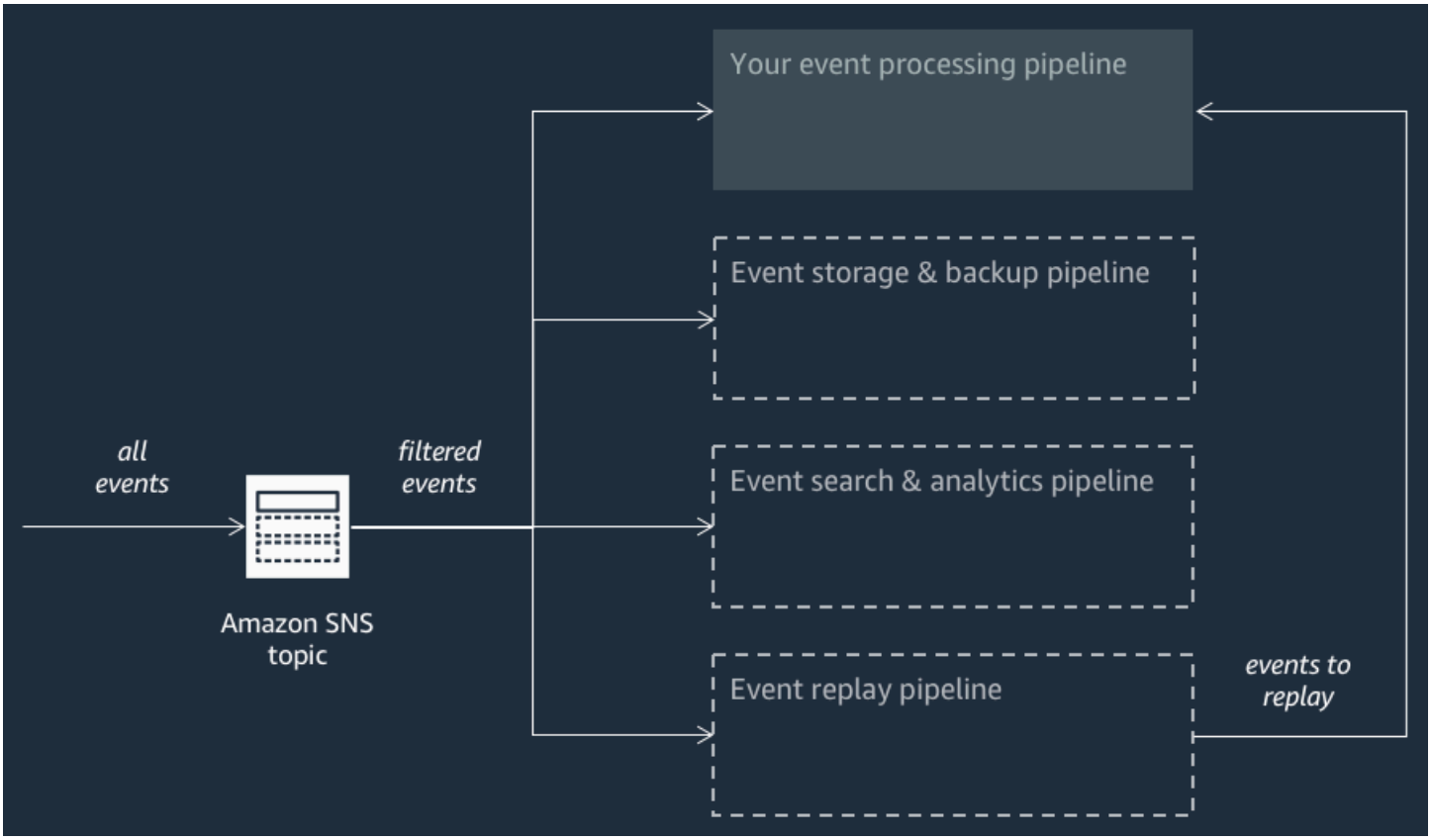
Amazon 事件分叉管道的工作原理

Amazon Event Fork Pipelines 是一种无服务器设计模式。但是，它也是一套基于 S Amazon AM 的嵌套无服务器应用程序（您可以将其直接从 Amazon Serverless Application Repository (S Amazon AR) 部署到您的， Amazon Web Services 账户 以丰富您的事件驱动平台）。您可以根据架构的需要单独部署这些嵌套的应用程序。

主题

- 事件存储与备份管线
- 事件搜索与分析管线
- 事件重播管线

下图显示了一个由三个嵌套应用程序补充的 Amazon Event Fork Pipelines 应用程序。根据您的架构要求，您可以在 Amazon SAR 上独立部署 Amazon Event Fork Pipelines 套件中的任何管道。



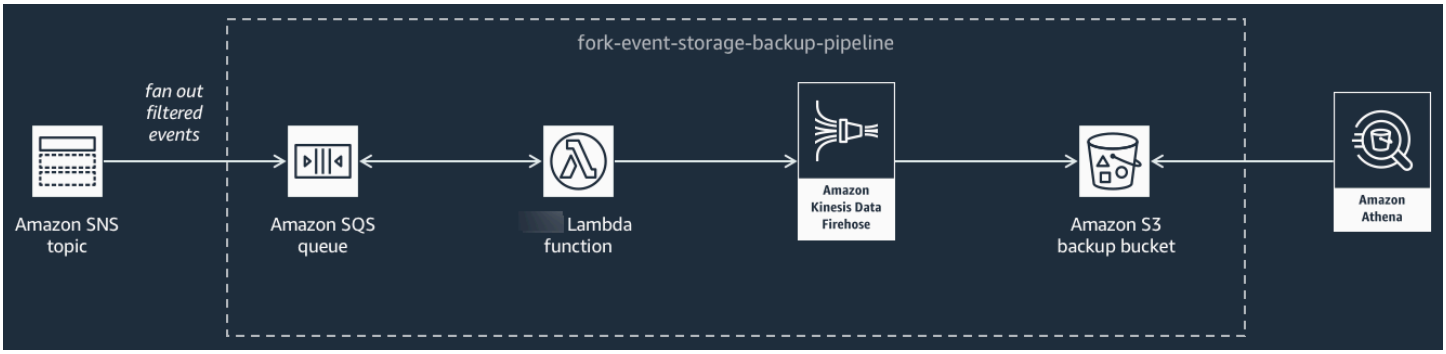
为每个管线订阅了相同的 Amazon SNS 主题，并允许管线在事件发布到主题时并行处理这些事件。每个管线都是独立的，并且可以设置其自己的订阅筛选策略。这允许管线仅处理它感兴趣的部分事件（而不是发布到主题的所有事件）。

> **Note**
>
> 由于您将三个 Amazon 事件分叉管道放置在常规事件处理管道旁边（可能已经订阅了您的 Amazon SNS 主题），因此您无需更改当前消息发布者的任何部分即可在现有 Amazon 工作负载中利用事件分叉管道。

事件存储与备份管线

下图显示了事件存储与备份管线。您可以为此管线订阅 Amazon SNS 主题来自动备份流经系统的事件。

该管道由一个用于缓冲由 Amazon SNS 主题传送的事件的 Amazon SQS 队列、一个自动轮询队列中这些事件并将其推送到 Amazon Data Firehose 流的 Amazon Lambda 函数，以及一个持久备份流加载的事件的 Amazon S3 存储桶。

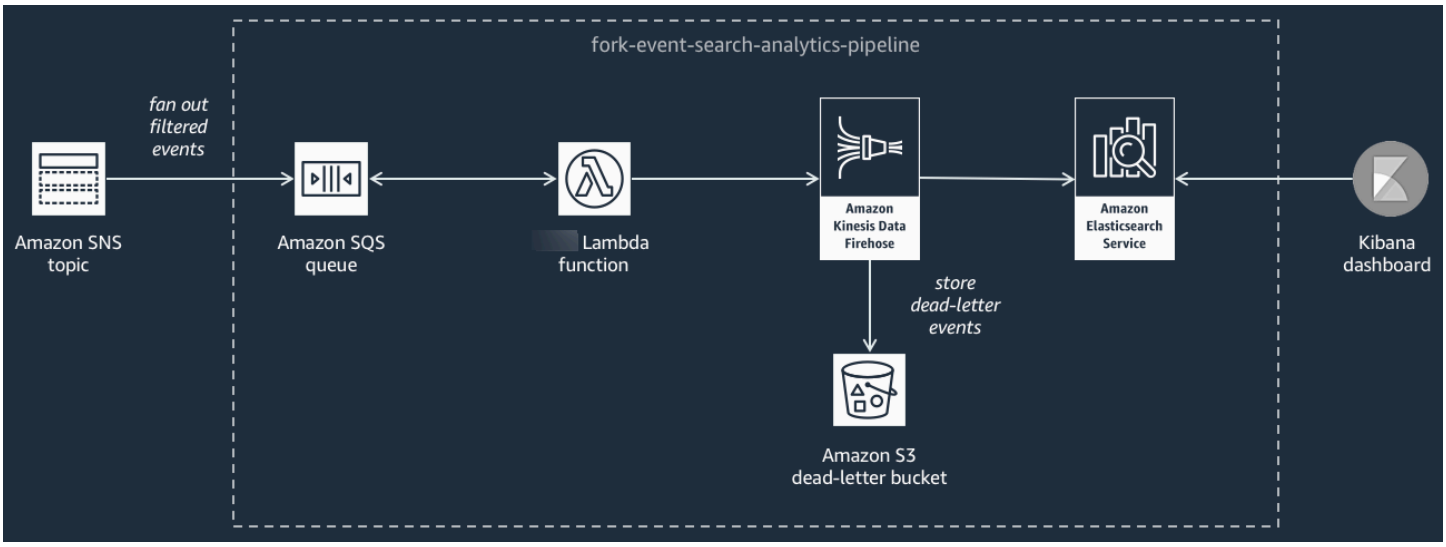


要微调 Firehose 流的行为，可将其配置为在将事件加载到存储桶之前对事件进行缓冲、转换和压缩。在加载事件时，可以使用 Amazon Athena 通过标准 SQL 查询来查询存储桶。您也可以将管道配置为重用现有 Amazon S3 存储桶或创建一个新的存储桶。

事件搜索与分析管线

下图显示了事件搜索与分析管线。您可以为此管线订阅 Amazon SNS 主题以便在搜索域中为流经系统的事件编制索引，然后对这些事件进行分析。

该管道由一个用于缓冲亚马逊 SNS 主题传送的事件的 Amazon SQS 队列、一个轮询队列中的事件并将其推送到 Amazon Data Firehose 流中的 Amazon Lambda 函数、一个为 Firehose 流加载的事件编制索引的亚马逊 OpenSearch 服务域以及一个存储无法在搜索域中编制索引的死信事件的 Amazon S3 存储桶组成。



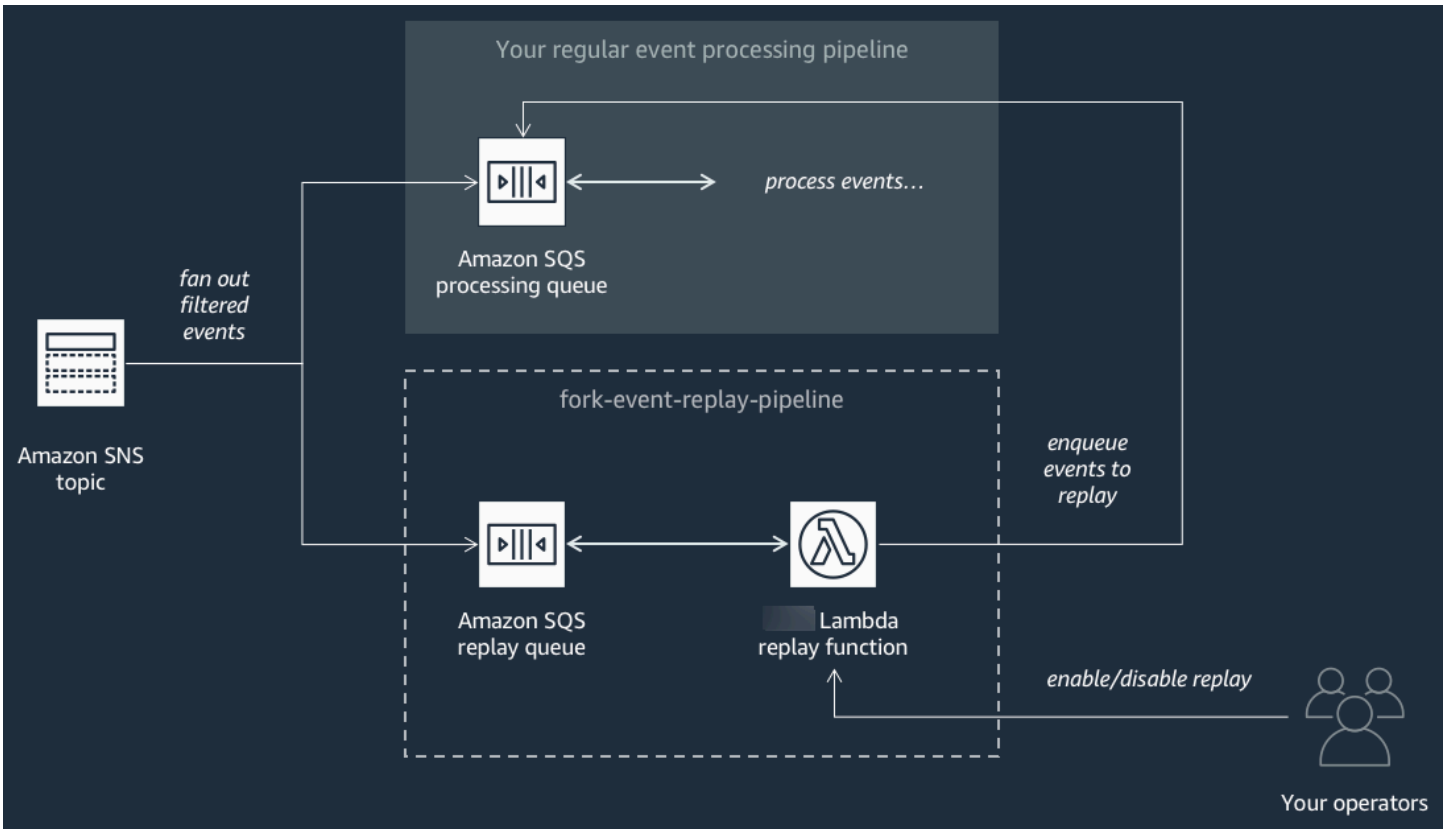
要在事件缓冲、转换和压缩方面微调 Firehose 流，您可以配置此管线。

您还可以配置管道是应重复使用您的现有 OpenSearch 域 Amazon Web Services 账户 还是为您创建一个新域。在搜索域中为事件编制索引时，您可以使用 Kibana 对事件运行分析并实时更新可视化控制面板。

事件重播管线

下图显示了事件重播管线。要记录系统在过去 14 天内处理过的事件（例如，当您的平台需要从故障中恢复时），您可以为此管道订阅 Amazon SNS 主题，然后重新处理事件。

该管道由一个 Amazon SQS 队列组成，该队列用于缓冲由 Amazon SNS 主题传送的事件，以及一个用于轮询队列中的事件并将其重新驱动到您的常规事件处理管道的 Amazon Lambda 函数，该管道也订阅了您的主题。



> **Note**
>
> 默认情况下，重播功能已禁用，而不会重新导入您的事件。如果您需要重新处理事件，则必须启用 Amazon SQS 重播队列作为 Amazon Lambda 重播函数的事件源。

部署 Amazon 事件分叉管道

Amazon Event Fork Pipelines 套件（选择 “显示创建自定义 IAM 角色或资源策略的应用程序”）在中作为一组公共应用程序提供 Amazon Serverless Application Repository，您可以从中使用Amazon

Lambda 控制台手动部署和测试它们。有关使用 Amazon Lambda 控制台部署管道的信息，请参阅将 Amazon 事件分叉管道订阅到 Amazon SNS 主题。

在生产场景中，我们建议在整个应用程序的 Amazon SAM 模板中嵌入 Amazon 事件分支管道。嵌套应用程序功能允许您通过将资源添加到 Amazon SAM 模板、引用嵌套应用程序的 S Amazon AR `ApplicationId` 和`AWS::Serverless::Application`来实现此`SemanticVersion`目的。

例如，您可以将以下 YAML 代码段添加到 SA Amazon M 模板的`Resources`部分，从而将事件存储和备份管道用作嵌套应用程序。

```
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
```

指定参数值时，您可以使用 Amazon CloudFormation 内部函数来引用模板中的其他资源。例如，在上面的 YAML 片段中，`TopicArn`参数引用了模板中其他地方定义的`AWS::SNS::Topic` Amazon SAM 资源`MySNSTopic`。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的内置函数参考。

> **Note**
>
> 您的 Amazon SAR 应用程序的 Amazon Lambda 控制台页面包括复制为 SAM 资源按钮，该按钮可将嵌套 SA Amazon R 应用程序所需的 YAML 复制到剪贴板。

部署和测试 Amazon SNS Event Fork Pipelines 示例应用程序

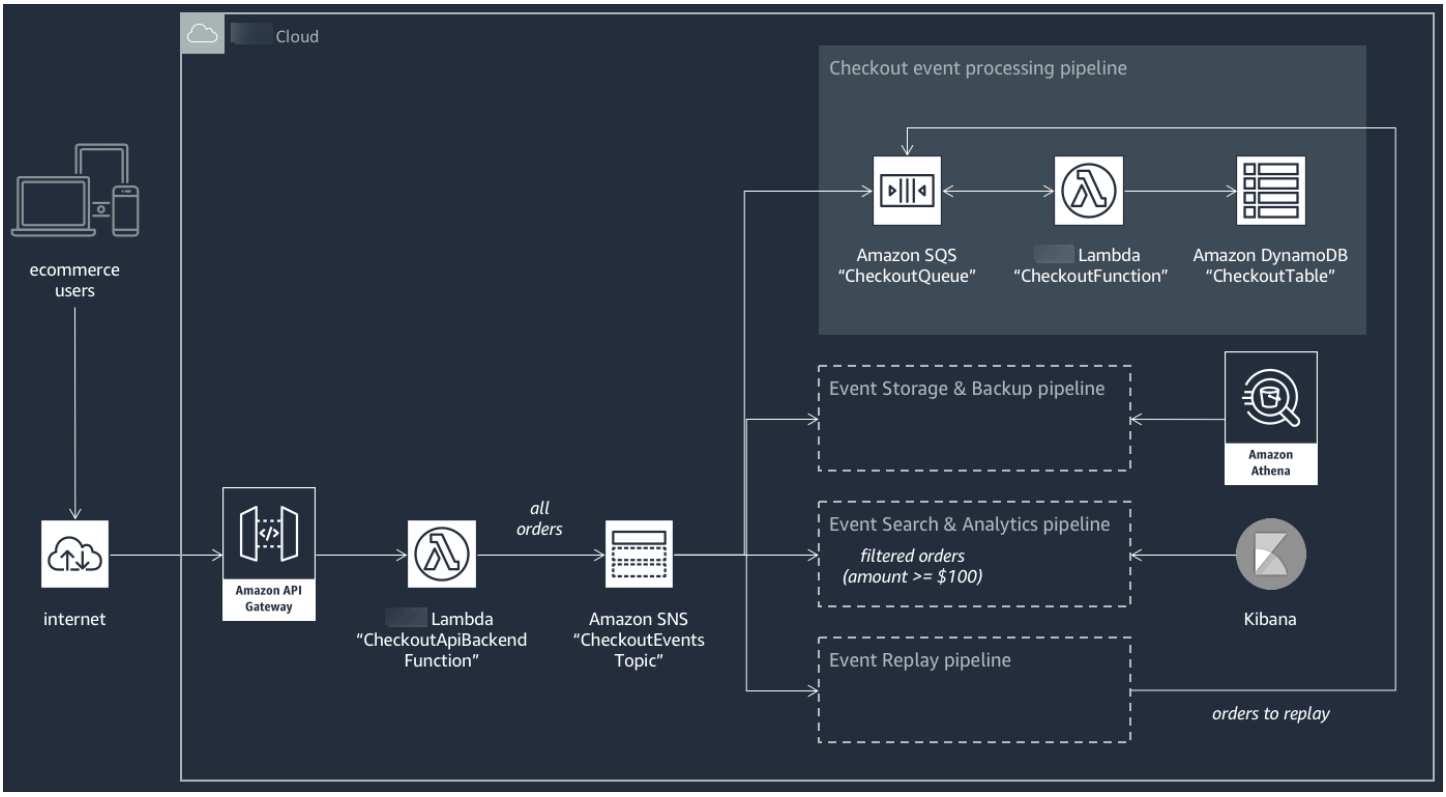
为了加快事件驱动型应用程序的开发，您可以订阅 Amazon SNS 主题的事件处理管道（由事件 Amazon 分叉管道提供支持）。 Amazon Event Fork Pipelines 是一套基于Amazon 无服务器应用程序模型 (SA Amazon M) 的开源嵌套应用程序，您可以直接从 Ev Amazon ent Fork Pipelines 套件（选择显示创建自定义 IAM 角色或资源策略的应用程序）将其部署到您的 Amazon 账户中。有关更多信息，请参阅 Amazon 事件分叉管道的工作原理。

本页介绍如何使用部署和测试 E Amazon vent Fork Pipelines 示例应用程序。 Amazon Web Services Management Console

> **Important**
>
> 为避免在部署完 Amazon 事件分支管道示例应用程序后产生不必要的费用，请删除其 Amazon CloudFormation 堆栈。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的在 Amazon CloudFormation 控制台上删除堆栈。

Amazon 事件分叉管道用例示例

以下场景描述了一个使用 Event Fork Pipelines Amazon 的事件驱动型无服务器电子商务应用程序。您可以在中使用此示例电子商务应用程序， Amazon Serverless Application Repository 然后 Amazon Web Services 账户 使用控制台将其部署到您的 Amazon Lambda 控制台中，您可以在其中对其进行测试并检查其源代码 GitHub。



该电子商务应用程序通过 RESTful API Gateway托管并由该 Amazon Lambda 功能支持的API接受买家的订单`CheckoutApiBackendFunction`。此函数将收到的所有订单发布到名为`CheckoutEventsTopic` 的 Amazon SNS 主题，该主题转而将订单分散到四个不同的管道。

第一个管道是由电子商务应用程序的拥有者设计和实现的常规结算处理管道。该管道具有用于缓冲所有已收到订单的 Amazon SQS 队列`CheckoutQueue`、一个名为`CheckoutFunction`的 Amazon Lambda 函数，用于轮询队列以处理这些订单，还有一个用于安全保存所有已下订单的 DynamoDB 表`CheckoutTable`。

应用 Amazon 事件分叉管道

电子商务应用程序的组件处理核心业务逻辑。但是，电子商务应用程序拥有者还需满足：

- 合规性 - 安全的、压缩的静态加密备份，清理敏感信息
- 弹性 - 在执行过程中断的情况下重播最近的订单
- 可搜索性 - 对已下订单运行分析并生成指标

应用程序所有者无需实现此事件处理逻辑，而是可以订阅 `CheckoutEventsTopic` Amazon SNS 主题的 Amazon Event Fork Pipelines

- 事件存储与备份管线配置为转换数据以删除信用卡详细信息，缓冲数据 60 秒，使用 GZIP 压缩数据，并使用 Amazon S3 的原定设置客户自主管理型密钥来加密数据。此密钥由 () 管理 Amazon 并由 Amazon Key Management Service (Amazon KMS) 提供支持。

  有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的为您的目的地选择 Amazon S3、Amazon Data Firehose 数据转换和配置设置。
- 为事件搜索与分析管线配置了一个 30 秒的索引重试持续时间、一个用于存储无法在搜索域中编制索引的订单的存储桶和一个用来限制已编制索引的订单集的筛选策略。

  有关更多信息，请参阅 Amazon Data Firehose 开发者指南中的为您的目的地选择 OpenSearch 服务。
- 事件重播管线 为配置了常规订单处理管道（由电子商务应用程序拥有者设计和实施）的 Amazon SQS 队列部分。

  有关更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的队列名称和 URL。

在事件搜索与分析管道的配置中设置以下 JSON 筛选策略。它仅匹配总金额为 100 美元或更多的传入订单。有关更多信息，请参阅 Amazon SNS 消息筛选。

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

使用 E Amazon vent Fork Pipelines 模式，电子商务应用程序所有者可以避免开发开销，这种开销通常是在为事件处理编写非微分逻辑之后出现的。取而代之的是，她可以将 Amazon 事件分叉管道直接从部署 Amazon Serverless Application Repository 到她身上 Amazon Web Services 账户。

第 1 步：部署 Amazon SNS 示例应用程序

1. 登录 Amazon Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 依次选择 Browse serverless app repository（浏览无服务器应用程序存储库）、Public applications（公共应用程序）、Show apps that create custom roles or resource policies（显示创建 IAM 角色或资源策略的应用程序）。
   b. 搜索 `fork-example-ecommerce-checkout-api`，然后选择该应用程序。
4. 在 fork-example-ecommerce-checkout-api 页面上，执行以下操作：
   a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`fork-example-ecommerce-my-app`）。

      > **Note**
      > - 要稍后轻松找到您的资源，请保留前缀 `fork-example-ecommerce`。
      > - 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 Amazon CloudFormation 堆栈（而不是创建新的堆栈）。

   b. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：
      - `DEBUG`
      - `ERROR`
      - `INFO`（默认值）
      - `WARNING`
5. 选择 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (我确认此应用程序创建自定义 IAM 角色和资源策略并部署嵌套应用程序)，然后在页面底部选择 Deploy (部署)。

在 fork-example-ecommerce-的部署状态*my-app*页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， Amazon CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_PROGRESS 状态。该过程完成后， Amazon CloudFormation 将显示 “创建_完成” 状态。

> **Note**
>
> 部署所有资源可能需要 20-30 分钟。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

步骤 2：执行与 SNS 关联的示例应用程序

1. 在 Amazon Lambda 控制台的导航面板上，选择应用程序。
2. 在 Applications (应用程序) 页面上的搜索字段中，搜索 `serverlessrepo-fork-example-ecommerce-`*`my-app`*，然后选择该应用程序。
3. 在 Resources (资源) 部分中，执行以下操作：
   a. 例如，要查找类型为的资源 ApiGatewayRestApi，请按类型对资源进行排序`ServerlessRestApi`，然后展开该资源。
   b. 将显示两个嵌套资源，分别是 “ApiGateway部署” 和 “ApiGateway阶段”。
   c. 复制链接 Prod API endpoint (Prod API 终端节点) 并为其附加 `/checkout`，例如：

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

4. 将以下 JSON 复制到名为 `test_event.json` 的文件中。

   ```
   {
      "id": 15311,
      "date": "2019-03-25T23:41:11-08:00",
      "status": "confirmed",
      "customer": {
         "id": 65144,
     "quantity": 2,
         "price": 25.00,
         "subtotal": 50.00
   ```

```
    }]
}
```

5. 要将 HTTPS 请求发送到您的 API 端点，请通过执行 `curl` 命令来将示例事件负载作为输入传递，例如：

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API 将返回以下空响应，并指示已成功执行：

```
{ }
```

第 3 步：验证 Amazon SNS 应用程序和管道性能

步骤 1：验证示例签出管道的执行

1. 登录 Amazon DynamoDB 控制台。
2. 在导航面板上，选择表。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutTable`。
4. 在表详细信息页面上，选择项目，然后选择已创建的项目。

   将显示存储的属性。

步骤 2：验证事件存储与备份管道的执行

1. 登录 Amazon S3 控制台。
2. 在导航面板上，选择 Buckets (存储桶)。
3. 搜索 `serverlessrepo-fork-example`，然后选择 `CheckoutBucket`。
4. 导航目录层次结构，直到找到扩展名为 `.gz` 的文件。
5. 要下载该文件，请依次选择 Actions (操作) 和 Open (打开)。
6. 为管道配置了一个 Lambda 函数，此函数将清理信用卡信息以实现合规性。

   要验证存储的 JSON 负载不包含任何信用卡信息，请解压缩该文件。

步骤 3：验证事件搜索与分析管道的执行

1. 登录到OpenSearch 服务控制台。
2. 在导航面板上的 My domains (我的域) 下，选择前缀为 `serverl-analyt` 的域。
3. 为管道配置了一个 Amazon SNS 订阅筛选策略，该策略设置一个数值匹配条件。

   要验证该事件是否因为指的是价值高于100美元的订单而被索引，请在服务器分析***abcdefgh1ijk***页面上选择指数，checkout_even ts。

步骤 4：验证事件重播管道的执行

1. 登录 Amazon SQS 控制台。
2. 在队列列表中，搜索 `serverlessrepo-fork-example` 并选择 `ReplayQueue`。
3. 选择发送和接收消息。
4. 在 fork-example-ecommerce-***my-app***... ReplayP-ReplayQueue-中发送和接收消息***123ABCD4E5F6***对话框中，选择轮询留言。
5. 要验证事件是否已入队，请选择队列中显示的消息旁边的 More Details (更多详细信息)。

步骤 4：模拟问题并重播事件以进行恢复

步骤 1：启用模拟的问题并发送第二个 API 请求

1. 登录 Amazon Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutFunction`。
4. 在 fork-example-ecommerce-***my-app***-CheckoutFunction-***ABCDEF***... 页面的环境变量部分中，将 BUG_ ENABLED 变量设置为 true，然后选择保存。
5. 将以下 JSON 复制到名为 `test_event_2.json` 的文件中。

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
"quantity": 1,
       "price": 75.00,
```

```
        "subtotal": 75.00
    }]
}
```

6. 要将 HTTPS 请求发送到您的 API 端点，请通过执行 `curl` 命令来将示例事件负载作为输入传递，例如：

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API 将返回以下空响应，并指示已成功执行：

```
{ }
```

步骤 2：验证模拟数据损坏

1. 登录 Amazon DynamoDB 控制台。
2. 在导航面板上，选择表。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutTable`。
4. 在表详细信息页面上，选择项目，然后选择已创建的项目。

   将显示存储的属性，其中一些标记为 CORRUPTED! (已损坏!)

步骤 3：禁用模拟的问题

1. 登录 Amazon Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutFunction`。
4. 在 fork-example-ecommerce-***my-app***-CheckoutFunction-***ABCDEF***... 页面的环境变量部分中，将 BUG_ ENABLED 变量设置为 false，然后选择保存。

步骤 4：启用重播以从问题中恢复

1. 在 Amazon Lambda 控制台的导航面板上，选择功能。
2. 搜索 `serverlessrepo-fork-example` 并选择 `ReplayFunction`。
3. 展开 Designer 部分，选择 SQS 磁贴，然后在 SQS 部分中，选择 Enabled (启用)。

> **Note**
>
> 启用 Amazon SQS 事件源触发器大约需要 1 分钟。

4. 选择保存。
5. 要查看已恢复的属性，请返回到 Amazon DynamoDB 控制台。
6. 要禁用重播，请返回 Amazon Lambda 控制台并禁用 Amazon SQS 事件源触发器。`ReplayFunction`

将 Amazon 事件分叉管道订阅到 Amazon SNS 主题

为了加快事件驱动型应用程序的开发，您可以订阅 Amazon SNS 主题的事件处理管道（由事件 Amazon 分叉管道提供支持）。 Amazon Event Fork Pipelines 是一套基于Amazon 无服务器应用程序模型 (SA Amazon M) 的开源嵌套应用程序，您可以直接从 Ev Amazon ent Fork Pipelines 套件（选择显示创建自定义 IAM 角色或资源策略的应用程序）将其部署到您的 Amazon 账户中。有关更多信息，请参阅 Amazon 事件分叉管道的工作原理。

本节介绍如何使用部署管道，然后 Amazon Web Services Management Console 将 Amazon 事件分叉管道订阅 Amazon SNS 主题。在开始之前，请创建 Amazon SNS 主题。

要删除构成管道的资源，请在 Amazon Lambda 控制台的应用程序页面上找到管道，展开 SAM 模板部分，选择CloudFormation堆栈，然后选择其他操作，删除堆栈。

订阅事件存储与备份管道并将其部署到 Amazon SNS

> 对于事件归档和分析，Amazon SNS 现在建议使用其与 Amazon Data Firehose 的本机集成。您可以将 Firehose 传输流订阅 SNS 主题，这样您就可以向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、亚马逊 Redshift 表、亚马逊 OpenSearch 服务（服务）等。OpenSearch 将 Amazon SNS 与 Firehose 传输流配合使用是一种完全托管且无需代码的解决方案，您无需使用任何功能。 Amazon Lambda 有关更多信息，请参阅 扇出到 Firehose 传输流。

本教程说明如何部署事件存储与备份管道并为该管道订阅 Amazon SNS 主题。此过程会自动将与管道关联的 Amazon SAM 模板转换为 Amazon CloudFormation 堆栈，然后将该堆栈部署到您的 Amazon Web Services 账户。此过程还会创建和配置构成事件存储与备份管道的资源集，包括以下内容：

- Amazon SQS 队列

- Lambda 函数
- Firehose 传输流
- Amazon S3 备份存储桶

有关配置以 Amazon S3 存储桶作为目标的流的更多信息，请参阅`S3DestinationConfiguration`《亚马逊数据 Firehose API 参考》。

有关转换事件以及配置事件缓冲、事件压缩和事件加密的详细信息，请参阅《Amazon Data Firehose 开发人员指南》中的创建 Amazon Data Firehose 传输流。

有关筛选事件的更多信息，请参阅本指南中的 Amazon SNS 订阅筛选策略。

1. 登录 Amazon Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 依次选择 Browse serverless app repository（浏览无服务器应用程序存储库）、Public applications（公共应用程序）、Show apps that create custom roles or resource policies（显示创建 IAM 角色或资源策略的应用程序）。
   b. 搜索 `fork-event-storage-backup-pipeline`，然后选择该应用程序。
4. 在 fork-event-storage-backup-pipelin e 页面上，执行以下操作：
   a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`my-app-backup`）。

      > **Note**
      >
      > - 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 Amazon CloudFormation 堆栈（而不是创建新的堆栈）。

   b. （可选）对于 BucketArn，请输入加载传入事件的 Amazon S3 存储桶的 ARN。如果您未输入值，则会在您的 Amazon 账户中创建一个新的 Amazon S3 存储桶。
   c. （可选）对于 DataTransformationFunctionArn，输入用于转换传入事件的 Lambda 函数的 ARN。如果您不输入值，则将禁用数据转换。
   d. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：
      - `DEBUG`

- `ERROR`
- `INFO`（默认值）
- `WARNING`

e. 对于 TopicArn，输入要订阅此分叉管道实例的 Amazon SNS 主题的 ARN。

f. （可选）对于StreamBufferingIntervalInSeconds和 StreamBufferingSizeInMBs，输入用于配置传入事件缓冲的值。如果您不输入任何值，则使用 300 秒和 5 MB。

g. （可选）输入以下StreamCompressionFormat设置之一以压缩传入的事件：

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED`（默认值）
- `ZIP`

h. （可选）对于 StreamPrefix，输入字符串前缀以命名存储在 Amazon S3 备份存储桶中的文件。如果您不输入值，则不使用任何前缀。

i. （可选）对于 SubscriptionFilterPolicy，输入 JSON 格式的 Amazon SNS 订阅筛选策略，用于筛选传入的事件。筛选策略决定在 OpenSearch 服务索引中对哪些事件进行索引。如果您不输入值，则不使用筛选（为所有事件编制索引）。

j. （可选）对于 SubscriptionFilterPolicyScope，输入字符串`MessageBody`或`MessageAttributes`以启用基于负载或基于属性的邮件筛选。

k. 选择 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (我确认此应用程序创建自定义 IAM 角色和资源策略并部署嵌套应用程序)，然后选择 Deploy (部署)。

在 “部署状态 ***my-app***” 页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， Amazon CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_PROGRESS 状态。该过程完成后， Amazon CloudFormation 将显示 “创建_完成” 状态。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

发布到您的 Amazon SNS 主题的消息存储在事件存储和备份管道自动配置的 Amazon S3 备份存储桶中。

订阅事件搜索与分析管道并将其部署到 Amazon SNS

对于事件归档和分析，Amazon SNS 现在建议使用其与 Amazon Data Firehose 的本机集成。您可以将 Firehose 传输流订阅 SNS 主题，这样您就可以向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、亚马逊 Redshift 表、亚马逊 OpenSearch 服务（服务）等。OpenSearch 将 Amazon SNS 与 Firehose 传输流配合使用是一种完全托管且无需代码的解决方案，您无需使用任何功能。 Amazon Lambda 有关更多信息，请参阅 扇出到 Firehose 传输流。

本教程说明如何部署事件搜索与分析管道并为该管道订阅 Amazon SNS 主题。此过程会自动将与管道关联的 Amazon SAM 模板转换为 Amazon CloudFormation 堆栈，然后将该堆栈部署到您的 Amazon Web Services 账户。此过程还会创建和配置构成事件搜索与分析管道的资源集，包括以下内容：

- Amazon SQS 队列
- Lambda 函数
- Firehose 传输流
- 亚马逊 OpenSearch 服务域名
- Amazon S3 死信存储桶

有关将流配置为以索引作为目标的更多信息，请参阅《Amazon Data Firehose API Reference》中的 `ElasticsearchDestinationConfiguration`。

有关转换事件以及配置事件缓冲、事件压缩和事件加密的详细信息，请参阅《Amazon Data Firehose 开发人员指南》中的创建 Amazon Data Firehose 传输流。

有关筛选事件的更多信息，请参阅本指南中的 Amazon SNS 订阅筛选策略。

1. 登录 Amazon Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 依次选择 Browse serverless app repository（浏览无服务器应用程序存储库）、Public applications（公共应用程序）、Show apps that create custom roles or resource policies（显示创建 IAM 角色或资源策略的应用程序）。
   b. 搜索 `fork-event-search-analytics-pipeline`，然后选择该应用程序。
4. 在 fork-event-search-analytics-pipelin e 页面上，执行以下操作：

a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`my-app-search`）。

> **Note**
>
> 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 Amazon CloudFormation 堆栈（而不是创建新的堆栈）。

b. （可选）对于 DataTransformationFunctionArn，输入用于转换传入事件的 Lambda 函数的 ARN。如果您不输入值，则将禁用数据转换。

c. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：

- `DEBUG`
- `ERROR`
- `INFO`（默认值）
- `WARNING`

d. （可选）对于 SearchDomainArn，输入 OpenSearch 服务域的 ARN，即配置所需计算和存储功能的集群。如果您不输入值，则使用默认配置创建新域。

e. 对于 TopicArn，输入要订阅此分叉管道实例的 Amazon SNS 主题的 ARN。

f. 对于 SearchIndexName，输入用于事件搜索和分析的 OpenSearch 服务索引的名称。

> **Note**
>
> 以下配额适用于索引名称：
>
> - 不能包含大写字母
> - 不能包含以下字符：`\ / * ? " < > | ` , #`
> - 不能以下列字符开头：`- + _`
> - 不能为以下内容：`. ..`
> - 长度不能超过 80 个字符
> - 长度不能超过 255 个字节
> - 不能包含冒号（来自 OpenSearch 服务 7.0）

g. （可选）为 OpenSearch 服务索引的轮换周期输入以下SearchIndexRotationPeriod设置之一：

- NoRotation（默认值）
- OneDay
- OneHour
- OneMonth
- OneWeek

索引轮换将时间戳附加到索引名称，从而促进旧数据的到期。

h. 对于 SearchTypeName，输入用于在索引中组织事件的 OpenSearch 服务类型的名称。

> **Note**
>
> - OpenSearch 服务类型名称可以包含任何字符（空字节除外），但不能以开头_。
> - 对于 S OpenSearch ervice 6.x，每个索引只能有一个类型。如果您为具有其他类型的现有索引指定新类型，Firehose 会返回运行时错误。

i. （可选）对于StreamBufferingIntervalInSeconds和 StreamBufferingSizeInMBs，输入用于配置传入事件缓冲的值。如果您不输入任何值，则使用 300 秒和 5 MB。

j. （可选）输入以下StreamCompressionFormat设置之一以压缩传入的事件：

- GZIP
- SNAPPY
- UNCOMPRESSED（默认值）
- ZIP

k. （可选）对于 StreamPrefix，输入字符串前缀以命名存储在 Amazon S3 死信存储桶中的文件。如果您不输入值，则不使用任何前缀。

l. （可选）对于 StreamRetryDurationInSecons，输入 Firehose 无法索引服务索引中的 OpenSearch 事件时的重试持续时间。如果您不输入值，则使用 300 秒。

m. （可选）对于 SubscriptionFilterPolicy，输入 JSON 格式的 Amazon SNS 订阅筛选策略，用于筛选传入的事件。筛选策略决定在 OpenSearch 服务索引中对哪些事件进行索引。如果您不输入值，则不使用筛选（为所有事件编制索引）。

n. 选择 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (我确认此应用程序创建自定义 IAM 角色和资源策略并部署嵌套应用程序)，然后选择 Deploy (部署)。

在 “部署状态 ***my-app-search***” 页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， Amazon CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_PROGRESS 状态。该过程完成后， Amazon CloudFormation 将显示 “创建_完成” 状态。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

发布到您的 Amazon SNS 主题的消息将在事件搜索和分析管 OpenSearch 道自动配置的服务索引中编制索引。如果管道无法为事件编制索引，则会将其存储在 Amazon S3 死信存储桶中。

使用 Amazon SNS 集成，部署事件重播管道

本教程说明如何部署事件重播管道并为该管道订阅 Amazon SNS 主题。此过程会自动将与管道关联的 Amazon SAM 模板转换为 Amazon CloudFormation 堆栈，然后将该堆栈部署到您的 Amazon Web Services 账户。此过程还会创建和配置构成事件重播管道的资源集，包括一个 Amazon SQS 队列和一个 Lambda 函数。

有关筛选事件的更多信息，请参阅本指南中的 Amazon SNS 订阅筛选策略。

1. 登录 Amazon Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 依次选择 Browse serverless app repository（浏览无服务器应用程序存储库）、Public applications（公共应用程序）、Show apps that create custom roles or resource policies（显示创建 IAM 角色或资源策略的应用程序）。
   b. 搜索 `fork-event-replay-pipeline`，然后选择该应用程序。
4. 在 fork-event-replay-pipeline 页面中，执行以下操作：
   a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`my-app-replay`）。

      > **Note**
      > 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 Amazon CloudFormation 堆栈（而不是创建新的堆栈）。

   b. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：

- DEBUG
- ERROR
- INFO（默认值）
- WARNING

c. （可选 ReplayQueueRetentionPeriodInSeconds）输入 Amazon SQS 重播队列保留消息的时间长度（以秒为单位）。如果您不输入值，则使用 1209600 秒（14 天）。

d. 对于 TopicArn，输入要订阅此分叉管道实例的 Amazon SNS 主题的 ARN。

e. 对于 DestinationQueueName，输入 Lambda 重播函数向其转发消息的 Amazon SQS 队列的名称。

f. （可选）对于 SubscriptionFilterPolicy，输入 JSON 格式的 Amazon SNS 订阅筛选策略，用于筛选传入的事件。筛选策略决定缓冲哪些事件以进行重播。如果您不输入值，则不使用筛选（缓冲所有事件以进行重播）。

g. 选择 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (我确认此应用程序创建自定义 IAM 角色和资源策略并部署嵌套应用程序)，然后选择 Deploy (部署)。

在 “部署状态 ***my-app-replay***” 页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， Amazon CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_PROGRESS 状态。该过程完成后， Amazon CloudFormation 将显示 “创建_完成” 状态。

部署完成后，Lambda 将显示 Your application has been deployed (您的应用程序已部署完成)状态。

将在由事件重播管道自动预置的 Amazon SQS 索引中缓冲发布到您的 Amazon SNS 主题的消息以进行重播。

> **Note**
>
> 默认情况下，禁用重播。要启用重播，请导航到 Lambda 控制台上的函数页面，展开 Designer 部分，选择 SQS 磁贴，然后在 SQS 部分中，选择 Enabled（启用）。

将 Amazon S EventBridge cheduler 与亚马逊 SNS 配合使用

Amazon Sched EventBridge uler 是一种无服务器计划程序，允许您通过一个中央托管服务创建、运行和管理任务。使用 EventBridge Scheduler，您可以使用 Cron 和费率表达式为重复模式创建计划，也

可以配置一次性调用。您可以设置灵活的交付时间窗口、定义重试限制，并为失败的 API 调用设置最大保留时间。

本页介绍如何使用 EventBridge 计划程序按计划发布来自 Amazon SNS 主题的消息。

设置执行角色

创建新计划时， EventBridge 调度程序必须有权代表您调用其目标 API 操作。您可以使用执行角色向 EventBridge 调度器授予这些权限。您附加到计划执行角色的权限策略定义了所需权限。这些权限取决于您希望 EventBridge 调度程序调用的目标 API。

当您使用 EventBridge 调度器控制台创建计划时，如以下过程所示， EventBridge 调度器会根据您选择的目标自动设置执行角色。如果要使用 EventBridge 调度器 SDKs、或中的一个来创建计划 Amazon CLI Amazon CloudFormation，则必须有一个现有的执行角色来授予 EventBridge 调度器调用目标所需的权限。有关为计划手动设置执行角色的更多信息，请参阅《日程EventBridge 安排器用户指南》中的设置执行角色。

创建计划

使用控制台创建计划

1. 在https://console.aws.amazon.com/scheduler/家中打开 Amazon EventBridge 日程安排器控制台。
2. 在计划页面，选择创建计划。
3. 在指定计划详细信息页面，在计划名称和描述部分中，执行以下操作：

   a. 对于计划名称，输入计划的名称。例如，**`MyTestSchedule`**。

   b. （可选）对于描述，输入对计划的描述。例如，**`My first schedule`**。

   c. 对于计划组，从下拉列表中选择一个计划组。如果您没有计划组，选择默认。要创建计划组，选择创建自己的计划。

   您可以使用计划组将标签添加到计划组。

4. • 选择计划选项。

出现	请执行此操作...
一次性计划	对于日期和时间，请执行以下操作：

出现	请执行此操作...
一次性计划仅在您指定的日期和时间调用一次目标。	• 输入 YYYY/MM/DD 格式的有效日期。 • 输入 24 小时 hh:mm 格式的时间戳。 • 对于时区，选择时区。
定期计划 定期计划按照您使用 cron 表达式或 rate 表达式指定的速率调用目标。	a. 对于计划类型，执行以下操作之一： • 要使用 cron 表达式定义计划，请选择基于 cron 的计划并输入 cron 表达式。 • 要使用 rate 表达式定义计划，请选择基于 rate 的计划并输入 rate 表达式。 有关 cron 和费率表达式的更多信息，请参阅 A mazon EventBridge 计划程序用户指南中的计划 EventBridge 程序中的计划类型。 b. 对于灵活的时间窗口，选择关闭以关闭该选项，或者选择一个预定义的时间窗口。例如，如果您选择 15 分钟并且将定期计划设置为每小时调用一次其目标，则该计划将在每小时开始后的 15 分钟内运行。

5. （可选）如果您在上一步中选择定期计划，在时间范围部分，请执行以下操作：

a. 对于时区，请选择时区。

b. 对于开始日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。

c. 对于结束日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。

6. 选择下一步。

7. 在选择目标页面上，选择 EventBridge 调度器调用 Amazon 的 API 操作：

   a. 选择 Amazon SNS 发布。

   b. 在 “发布” 部分中，选择一个 SNS 主题或选择 “新建” SNS 话题。

   c. （可选）输入 JSON 有效载荷。如果您未输入有效负载，则 EventBridge 调度器会使用空事件来调用该函数。

8. 选择下一步。

9. 在设置页面上，执行以下操作：

   a. 要打开计划，在计划状态下，切换启用计划。

   b. 要为计划配置重试策略，在重试策略和死信队列（DLQ）下，请执行以下操作：

      - 切换重试。
      - 对于事件的最大持续时间，输入 EventBridge 调度器必须保留未处理事件的最大小时数和最小值。
      - 最长时间为 24 小时。
      - 在 “最大重试次数” 中，输入目标返回错误时 EventBridge 调度器重试计划的最大次数。

        最大值为 185 次重试。

      使用重试策略，如果计划未能调用其目标，则调 EventBridge 度程序会重新运行该计划。如果已配置，则必须为计划设置最长保留时间和最大重试次数。

   c. 选择 EventBridge 日程安排器存储未传送事件的位置。

死信队列（DLQ）选项	请执行此操作...	
请勿存储	选择 None。	

死信队列（DLQ）选项	请执行此操作...
将活动存储 Amazon Web Services 账户 在您创建日程安排的同一位置	a. 在 “我的 Amazon Web Services 账户 作为 DLQ” 中选择 “选择 Amazon SQS 队列”。 b. 选择 Amazon SQS 队列的 Amazon 资源名称（ARN）。
将活动存储在与您创建日程表不同的 Amazon Web Services 账户 位置	a. 选择在其他队列中指定一个 Amazon SQS 队列 Amazon Web Services 账户 作为 DLQ。 b. 输入 Amazon SQS 队列的 Amazon 资源名称（ARN）。

d. 要使用客户托管密钥加密目标输入，在加密下，选择自定义加密设置（高级）。

   如果选择此选项，请输入现有的 KMS 密钥 ARN 或选择创建一个 Amazon KMS key 以导航到 Amazon KMS 控制台。有关 EventBridge 计划程序如何加密静态数据的更多信息，请参阅 A mazon EventBridge 计划程序用户指南中的静态加密。

e. 要让 EventBridge Scheduler 为您创建新的执行角色，请选择为此计划创建新角色。然后，在角色名称中输入名称。如果您选择此选项，S EventBridge cheduler 会将模板化目标所需的权限附加到该角色。

10. 选择下一步。
11. 在查看并创建计划页面上，查看计划的详细信息。在每个部分中，选择编辑返回到该步骤并编辑其详细信息。
12. 选择创建计划。

    您可以在计划页面上查看新的和现有的计划列表。在状态列下，验证新计划是否已启用。

相关资源

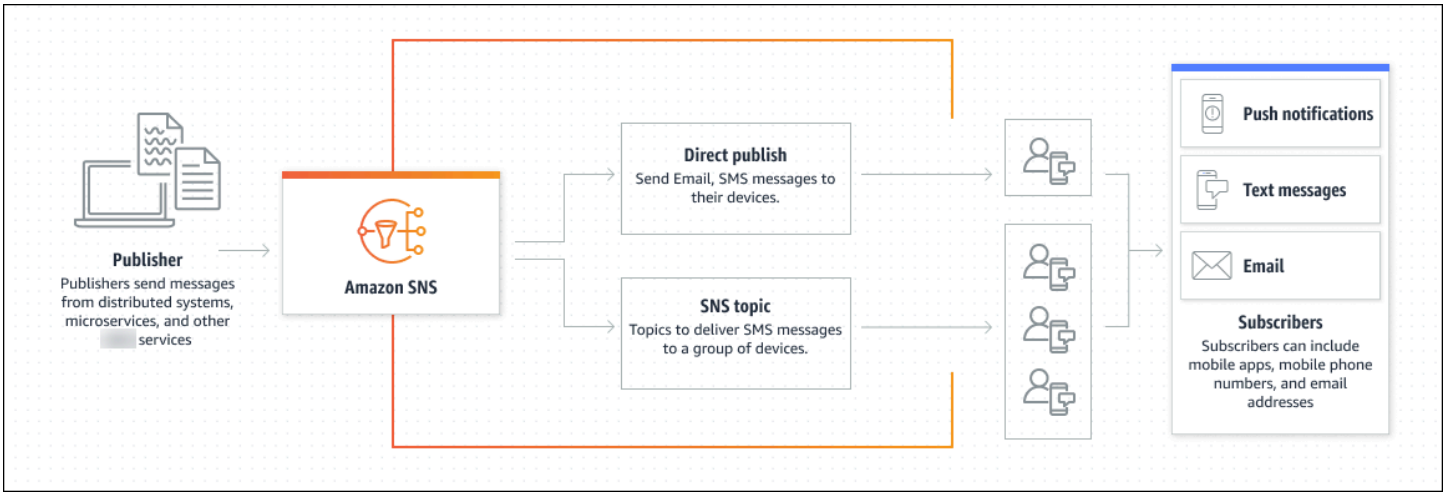
有关 EventBridge 调度程序的更多信息，请参阅以下内容：

- EventBridge 日程安排器用户指南
- EventBridge 调度器 API 参考
- EventBridge 调度程序定价

使用 Amazon SNS 发送消息 application-to-person

Amazon SNS application-to-person (A2P) 消息允许您通过 SMS（短信服务）将通知和提醒直接发送到客户的移动设备。使用此功能，您可以向移动应用程序发送推送通知，向手机号码发送短信，向电子邮件地址发送纯文本电子邮件。此外，您可以灵活地使用主题向多个收件人分发消息，或者将消息直接发布到各个移动端点以进行个性化通信。

本主题介绍如何使用 Amazon SNS 向订阅者发送用户通知，例如移动应用程序、手机号码和电子邮件地址。



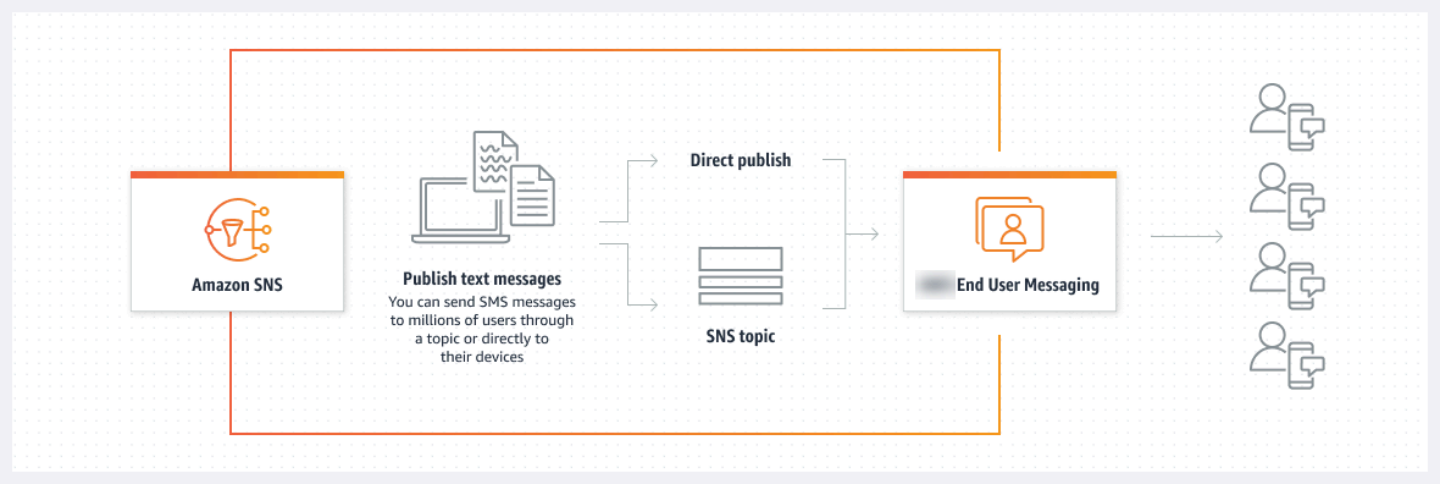
移动文本消息与 Amazon SNS

> **Important**
>
> 《Amazon SNS 短信开发人员指南》已更新。Amazon SNS 已与 Amazon End User Messaging SMS 集成，用于传输短信。此指南包含有关如何创建、配置和管理 Amazon SNS 短信的最新信息。

Amazon SNS 移动文本消息（SMS）旨在协助将消息传输到各种平台，例如支持 SMS 的 Web、移动设备和业务应用程序。用户可以通过订阅主题向一个或多个电话号码发送消息，从而简化分发过程。

Amazon SNS 消息由传送 Amazon End User Messaging SMS，这可确保可靠的消息传输。在 Amazon SNS 中 APIs，您可以设置各种属性，例如消息类型（促销或交易）、每月支出限额、退出列表和消息传输优化。

Amazon End User Messaging SMS 处理通过其全球短信供应网络向目标电话号码传输消息。它管理路线、传送状态以及对地区法规的遵守情况。要访问其他短信功能，例如精细权限、电话池、配置集、短信模拟器和国家/地区规则，请参阅《Amazon End User Messaging SMS User Guide》。



以下关键特征可帮助您发送可扩展且易于扩展的 Amazon SNS 短信：

自定义消息首选项

根据您的预算和用例设置短信偏好设置， Amazon Web Services 账户 从而为您自定义 SMS 发送方式。例如，您可以选择消息是优先考虑成本效率还是可靠传输。

设置支出配额

通过以下方法定制短信传输：为各个消息传输指定支出配额以及为您的 Amazon Web Services 账户指定每月支出配额。在当地法律和法规（例如美国和加拿大）要求的情况下，短信收件人可以选择退出，这意味着他们可以选择停止接收来自您的短信。 Amazon Web Services 账户收件人选择不接收消息后，您可以在某些限制条件下重新加入该电话号码，以便继续发送消息。

在全球范围内发送短信

Amazon SNS 在多个区域内均支持短信收发功能，让您可以向 240 多个国家和地区发送消息。

Amazon SNS 如何传送我的短信？

当您请求 Amazon SNS 代表您发送短信时，将使用 Amazon End User Messaging SMS分派消息。Amazon SNS 和 Amazon SNS 之间的集成 Amazon End User Messaging SMS 具有以下好处：

IAM 政策

您可以利用 IAM 和资源策略在其他 Amazon 服务和区域之间控制和分配对您的 SMS 资源的访问权限。

Amazon End User Messaging SMS 配置

所有与起源 ID 相关的配置（创建、配置更新、配置新来源 IDs、更改注册模板）都使用。 Amazon End User Messaging SMS

Amazon End User Messaging SMS 计费

不过，所有短信计费都已完成 Amazon End User Messaging SMS。您可以整合用于短信工作负载的 Amazon 支出，同时在一个中心位置采购和管理短信资源。

开始使用 Amazon SNS 短信

> **Important**
>
> 《Amazon SNS 短信开发人员指南》已更新。Amazon SNS 已与 Amazon End User Messaging SMS 集成，用于传输短信。此指南包含有关如何创建、配置和管理 Amazon SNS 短信的最新信息。

本主题将指导您管理您的 SMS 沙箱，以及配置 IAM 和基于资源的策略，以授予 Amazon SNS 访问和使用所需的权限。 Amazon End User Messaging SMS APIs

先决条件

Amazon SNS 建议更新您的 IAM 策略，使其包含以下操作，以确保对您的 Amazon SNS 资源的全面控制和可见性：

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

使用 Amazon SNS 短信沙盒

新创建的 Amazon SNS 短信账户会自动放入短信沙箱，通过降低欺诈和滥用风险来确保 Amazon 客户和收件人的安全。此环境是用于测试和开发目的的安全空间。在短信沙盒中操作时，您可以访问所有 Amazon SNS 特征，但要遵守某些限制：

- 您只能将短信发送到已验证的目的地电话号码。
- 您最多可以有 10 个已验证的目标电话号码。
- 您只能在验证或上次验证尝试后至少 24 小时删除目标电话号码。

当您的账户移出沙盒时，这些限制将被删除，您可以向任何收件人发送短信。

初始步骤

新的 Amazon SNS 短信账户被置于短信沙盒中。使用以下步骤在沙箱中创建和管理电话号码、创建发件号码和发件人 IDs以及注册您的公司。

1. 将目标电话号码添加到短信沙盒。有关添加、管理电话号码和将其移出 Amazon SNS 短信沙盒的详细信息，请参阅 在 Amazon SNS 短信沙盒中添加和验证电话号码。
2. 创建发起身份，该身份是您向收件人发送短信后，他们在其设备上看到的身份。要了解有关发起身份的更多信息，包括您可以使用的不同类型，请参阅Amazon SNS 短信的发起身份文档。
3. 注册您的公司。某些国家/地区要求您注册公司的身份，以便能够购买电话号码或发件人 IDs 并查看您发送给所在国家/地区的收件人的消息。有关哪些国家/地区需要注册的信息，请参阅《Amazon End User Messaging SMS User Guide》中 Supported countries and regions for SMS messaging with Amazon End User Messaging SMS。
4. 将您的消息发送到主题或手机。有关更多信息，请参阅 使用 Amazon SNS 发送短信。

在 Amazon SNS 短信沙盒中添加和验证电话号码

必须先完成以下设置步骤，然后才能开始 Amazon Web Services 账户 在短信沙箱中发送 SMS 消息。这样可以确保您的帐户已准备好发送短信，并且您的目标电话号码已得到正确验证。

1. 创建来源 ID。与 SMS 沙箱之外的账户类似，在向某些国家或地区的收件人发送 SMS 消息之前，必须先提供发端 ID。
2. 在 SMS 沙箱中添加要向其发送消息的目标电话号码。
3. 验证电话号码，确保目标电话号码可用于您的 SMS 消息。

添加并验证目标电话号码

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，选择支持短信收发的区域。
3. 在导航窗格中，选择文本消息(SMS)。

4. 在沙盒目标电话号码部分，选择添加电话号码。
5. 在 “目的地详情” 下，提供以下信息，然后选择 “添加电话号码”：
   - 目的地@@ 的国家代码和电话号码。
   - 您希望发送验证消息的语言。
6. 添加电话号码后，Amazon SNS 将向提供的目标电话号码发送 OTP。此OTP是验证所必需的。
7. 您将在您提供的目标电话号码上以标准短信形式收到 OTP。
   - 如果您在 15 分钟内没有收到 OTP，请在 Amazon SNS 控制台中选择 “重新发送验证码”。
   - 在 24 小时内，您最多可以重发 OTP 五次。
8. 收到 OTP 后，将其输入到验证码框中，然后选择验证电话号码。
9. 检查验证状态。
   - 成功验证电话号码后，电话号码及其验证状态将显示在沙盒目标电话号码部分。
   - 如果状态为 “待处理”，则表示验证不成功。例如，如果您没有正确输入国家/地区代码，则可能会发生这种情况。
   - 只有在上次尝试验证后 24 小时或更长时间后，您才能删除待处理或已验证的电话号码。
10. 如果您希望在其他地区使用相同的目标电话号码，请对要使用该号码的每个地区重复上述步骤。

未收到 OTP 短信疑难解答

排查可能导致电话号码无法接收 OTP 短信的常见问题。

- Amazon SNS 短信支出限额：如果您的 Amazon Web Services 账户 已超过发送短信的支出限额，则在提高限额或解决账单问题之前，可能无法发送更多消息，包括 OTP 短信。
- 未选择接收短信通知的电话号码：在某些国家或地区，收件人必须选择接收来自短代码的短信，短代码通常用于 OTP 短信。如果收件人的电话号码未被选中，他们将不会收到 OTP 短信。
- 运营商限制或过滤：某些移动运营商可能设置了限制或过滤机制，阻止发送某些类型的短信，包括 OTP 短信。这可能是由于运营商实施的安全策略或反垃圾邮件措施所致。
- 电话号码无效或不正确：如果收件人提供的电话号码不正确或无效，OTP 短信将无法送达。
- 网络问题：临时的网络问题或中断可能会导致无法向收件人的手机发送短信（包括 OTP 短信）。
- 延迟传送：在某些情况下，由于网络拥塞或其他因素，短信可能会延迟传送。OTP 短信最终可能会送达，但可能会延迟到预期的时间范围之外。
- 账户暂停或终止：如果您的账户存在问题 Amazon Web Services 账户，例如未付款或违反服务 Amazon 条款，则可能会暂停或终止 Amazon SNS 消息功能，包括 OTP 短信。

从 Amazon SNS 短信沙盒中删除电话号码

您可以从短信沙盒中删除待处理和已验证的目标电话号码。

> **Important**
>
> 您只能在验证电话号码后等待 24 小时，或者在您最后一次验证尝试后等待 24 小时再删除电话号码。

要从 SMS 沙盒中删除目标电话号码

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，选择您在其中添加了目标电话号码的支持短信收发的区域。
3. 在导航窗格中，选择文本消息（SMS）。
4. 在移动文本消息（SMS）页面上，前往沙盒目标电话号码部分。
5. 选择要删除的特定电话号码，然后选择删除电话号码。
6. 要确认您要删除电话号码，请输入 **`delete me`**，然后选择 Delete（删除）。

   请确保自您验证或尝试验证目标电话号码以来已过去 24 小时或更长时间，然后再继续删除。
7. 在添加目标电话号码但不再计划使用它的每个区域中重复这些步骤。

移出 Amazon SNS 短信沙盒

要 Amazon Web Services 账户 退出 SMS 沙箱，必须先添加、验证和测试目标电话号码。完成此操作后，使用创建案例 Amazon Web Services 支持。

请求将您的 Amazon 账户移出短信沙箱

1. 验证电话号码
   a. 当您在短信沙箱中 Amazon Web Services 账户 时，打开 Amazon SN S 控制台。
   b. 在导航窗格中的“移动”下方，选择文本消息（短信）。
   c. 在沙盒目标电话号码部分，添加和验证一个或多个目标电话号码。此验证可确保您可以成功发送和接收消息。
2. 测试短信发布

- 确认您可以将消息发送到至少一个已验证的目标电话号码并接收消息。有关如何发布短信的更多详细说明，请参阅使用 Amazon SNS 向手机发布短信。

3. 启动沙盒编辑

   - 在 Amazon SNS 控制台的 Mobile text messaging (SMS)（移动文本消息 (SMS)）页面上，在 Account information（账户信息）下，选择 Exit SMS sandbox（退出 SMS 沙盒）。此操作会将您重定向到 Amazon Support Center，并在选择增加服务配额选项的情况下自动创建支持案例。

4. 填写表单

   - 在支持表单中的增加服务配额下，执行以下操作：

     i. 选择 SNS 文本消息作为服务。

     ii. 提供您打算从中发送短信的网站 URL 或应用程序名称。

     iii. 指定您将发送的消息的类型：一次性密码、促销消息或事务性消息。

     iv. 选择您要从中发送短信的 Amazon Web Services 区域。

     v. 列出您计划向其发送短信的国家或地区。

     vi. 描述您的客户如何选择接收消息。

     vii. 包含您打算使用的所有消息模板。

5. 指定配额和区域

   - 在 Requests (请求) 下，执行以下操作：

     i. 选择你要移动Amazon Web Services 区域的地方 Amazon Web Services 账户。

     ii. 对于资源类型，选择一般限制。

     iii. 对于配额，选择退出短信沙盒。

     iv. （可选）要申请额外增加配额或其他调整，请选择添加其他请求并指定必要的详细信息。

     v. 对于新限额值，输入您要申请的限制（以 USD 为单位）。

6. 其他详细信息

   a. 在案例描述中，提供与您的请求相关的所有其他详细信息。

   b. 在联系选项下，然后选择您的首选联系语言。

7. 提交请求

   - 选择提交将请求发送到 Amazon Web Services 支持。

Amazon Web Services 支持 团队会在 24 小时内对您的请求做出初步回应。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们要仔细考虑每个请求。如果我们可以，我们将在 24 小时内准予您的请求。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。

如果您的使用案例与我们的策略不符，我们可能无法准予您的请求。

Amazon SNS 短信的发起身份

> **Important**
>
> 《Amazon SNS 短信开发人员指南》已更新。Amazon SNS 已与 Amazon End User Messaging SMS 集成，用于传输短信。此指南包含有关如何创建、配置和管理 Amazon SNS 短信的最新信息。

短信的发起身份是用于表示短信发件人的标识符。您可以使用以下类型的发起身份向您的收件人标识自己的身份：

起源号码

标识 SMS 消息发件人电话号码的数字字符串。发起号码有几种类型，包括长代码（通常包含 10 位或更多位数字的标准电话号码）、10 位数字的长代码（10DLC）、免费电话号码（TFN）和短代码（包含四到七位数字的电话号码）。

在当地法律要求使用发件人 IDs 代替发件人号码的国家/地区，不提供对发件人号码的支持。当您使用源号码发送 SMS 消息时，收件人的设备将源号码显示为发件人的电话号码。您可以按使用案例指定不同的源号码。

有关更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Phone numbers。

> **Tip**
>
> 要查看您 Amazon 账户中所有现有来源编号的列表，请在 Amazon SNS 控制台的导航窗格中，选择来源编号。

发件人 IDs

用于标识 SMS 消息发件人的字母名称。当您使用发件人 ID 发送短信，并且收件人位于支持发件人 ID 身份验证的区域时，在收件人的设备上会显示您的发件人 ID 而不是电话号码。发件人 ID 可以向 SMS 收件人提供比电话号码、长代码或短代码所能提供的更多的发送人信息。

全球多个国家和地区 IDs 都支持发件人。在有些地方，如果您是一个商家，若要向个人客户发送短信，则必须使用事先在监管机构或行业组注册的发件人 ID。有关支持或要求发件人的国家和地区的完整列表 IDs，请参阅Amazon End User Messaging SMS 用户指南 Amazon End User Messaging SMS中的支持短信发送的国家和地区。

使用发件人不收取额外费用 IDs。但是，对于发件人 ID 身份验证的支持和要求因国家/地区而异。一些主要市场（包括加拿大、中国和美国）不支持使用 IDs发件人。有些区域要求向个人客户发送 SMS 消息的公司必须使用事先在监管机构或行业组注册的发件人 ID。

有关更多信息，请参阅《Amazon End User Messaging SMS 用户指南》 IDs中的 “发件人”。

在 Amazon SNS 中配置短信收发

> **Important**
>
> 《Amazon SNS 短信开发人员指南》已更新。Amazon SNS 已与 Amazon End User Messaging SMS 集成，用于传输短信。此指南包含有关如何创建、配置和管理 Amazon SNS 短信的最新信息。

您可以使用 Amazon SNS 短信中的配置来设置 SMS 首选项以满足您的要求，例如调整支出配额和设置传输状态日志记录。本主题还详细介绍了如何使用 Amazon SNS 控制台和 S Amazon DK 向主题发布短信、如何高效处理配额以及如何检索短信活动的详细统计数据。

使用 Amazon SNS 发送短信

本节介绍如何使用 Amazon SNS 发送短信，包括发布到主题、订阅话题的电话号码、设置消息的属性以及直接发布到手机。

将短信发布到 Amazon SNS 主题

您可以通过使用多个电话号码订阅 Amazon SNS 主题，一次将一条 SMS 消息发布至这些手机号码。SNS 主题是您可以添加订阅者并随后向所有订阅者发布消息的通信渠道。订阅者会收到发布到该主题的所有消息，直到您取消订阅或订阅者选择不接收来自您的 Amazon 账户的 SMS 消息。

使用 Amazon 管理控制台向主题发送消息

要创建主题，请执行以下操作

如果您还没有要发送 SMS 消息的主题，请完成以下步骤。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，选择支持短信收发的区域。
3. 在导航窗格中，选择 Topics（主题）。
4. 在 Topics（主页）页面上，选择 Create topic（创建主题）。
5. 在 Create topic（创建主题）页面上的 Details（详细信息）下，执行以下操作：
   a. 对于 Type（类型），选择 Standard（标准）。
   b. 对于 Name（名称），输入一个主题名称。
   c. （可选）对于 Display name（显示名称），请为您的 SMS 消息输入自定义前缀。在您向主题发送消息时，Amazon SNS 会在显示名称之前加上右尖括号 (>) 和空格。显示名称不区分大小写，Amazon SNS 会将显示名称转换为大写字符。例如，如果主题的显示名称是 `MyTopic`，而消息是 `Hello World!`，则该消息会显示为：

      ```
      MYTOPIC> Hello World!
      ```
6. 选择创建主题。主题的名称和 Amazon Resource Name (ARN) 显示在 Topics（主题）页面。

创建 SMS 订阅

您可以使用订阅通过仅向主题发布一次消息，将 SMS 消息发送给多个收件人。

> **Note**
> 当您开始使用 Amazon SNS 发送短信时，您的 Amazon 账户处于短信沙箱中。SMS 沙盒为您提供了一个安全的环境，让您可以尝试 Amazon SNS 功能，而不会拿您作为 SMS 发件人的声誉冒险。在您的账户位于 SMS 沙盒时，您可以使用 Amazon SNS 的所有功能，但您只能向已验证的目标电话号码发送 SMS 消息。有关更多信息，请参阅 使用 Amazon SNS 短信沙盒。

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择订阅。
3. 在订阅页面上，选择创建订阅。

4. 在 Create subscription（创建订阅）页面上的 Details（详细信息）下，执行以下操作：
   a. 对于 Topic ARN（主题 ARN），输入或选择要向其发送 SMS 消息的主题的 Amazon Resource Name (ARN)。
   b. 对于 Protocol（协议），选择 SMS。
   c. 对于 Endpoint（终端节点），输入要订阅主题的电话号码。
5. 选择创建订阅。订阅信息显示在 Subscriptions（订阅）页面。

   要添加更多电话号码，请重复以下步骤。您还可以添加其他类型的订阅，例如电子邮件。

发送邮件

当您向一个主题发布消息时，Amazon SNS 会尝试将该消息传输至订阅该主题的每个电话号码。

1. 在 Amazon SNS 控制台中的 Topics（主题）页面上，选择要向其发送 SMS 消息的主题的名称。
2. 在主题详细信息页面上，选择发布消息。
3. 在 Publish message to topic（将消息发布到主题）页面上的 Message details（消息详细信息）下，执行以下操作：
   a. 对于 Subject（主题），请将该字段留空，除非您的主题包含电子邮件订阅，并且您想要同时发布到电子邮件和 SMS 订阅。Amazon SNS 使用您输入的 Subject（主题）作为电子邮件主题行。
   b. （可选）对于 Time to Live (TTL)（存活时间 (TTL)），输入 Amazon SNS 向任何移动应用程序终端节点订阅者发送 SMS 消息所需的秒数。
4. 在 Message body（消息正文）下，执行以下操作：
   a. 对于 Message structure（消息结构），选择 Identical payload for all delivery protocols（所有传输协议的负载相同）向订阅了您的主题的所有协议类型发送相同消息。或者，选择 Custom payload for each delivery protocol（针对每个传输协议的自定义负载）为不同协议类型的订阅者自定义消息。例如，您可以输入电话号码订阅者的默认消息，以及电子邮件订阅者的自定义消息。
   b. 对于 Message body to send to the endpoint（要发送到终端节点的消息正文），输入您的消息或每个传输协议的自定义消息。

      如果您的主题包含显示名称，Amazon SNS 会将其添加到消息，这会增加消息的长度。显示名称的长度是名称的字符数加上两个字符，也就是 Amazon SNS 添加的右尖括号 (>) 和空格。

有关 SMS 消息大小配额的信息，请参阅 使用 Amazon SNS 向手机发布短信。

5. （可选）在消息属性中，添加消息元数据，例如时间戳、签名和。 IDs
6. 选择发布消息。Amazon SNS 会发送 SMS 消息并显示成功消息。

使用向主题发送消息 Amazon SDKs

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置Amazon SDKs 和凭据文件。

以下代码示例显示了如何：

- 创建 Amazon SNS 主题。
- 使用手机号码订阅主题。
- 向主题发布 SMS 消息，以使所有订阅的电话号码一次接收消息。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个主题并返回其 ARN。

```
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 "";
    }
}
```

为终端节点订阅主题。

```
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);
        }
    }
}
```

设置消息的属性，例如发件人的 ID、最高价格及其类型。消息属性是可选的。

```
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);
        }
    }
}
```

向主题发布消息。消息将会发送到每个订阅用户。

```
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);
        }
    }
}
```

使用 Amazon SNS 向手机发布短信

您可以使用 Amazon SNS 直接向移动电话发送 SMS 消息，而无需订阅 Amazon SNS 主题的电话号码。

> **Note**
>
> 如果您想要将一条消息一次发布至多个电话号码，使用电话号码订阅主题会很有用。有关向主题发布 SMS 消息的说明，请参阅 将短信发布到 Amazon SNS 主题。

在发送消息时，您可以控制是否对消息进行成本或可靠性传输优化。您还可以指定发件人 ID 或源号码。如果您使用 Amazon SNS API 或 Amazon SDKs，以编程方式发送消息，则可以为消息传输指定最高价格。

每条 SMS 消息最多能包含 140 字节，字符配额具体取决于编码方案。例如，SMS 消息可以包含：

- 160 个 GSM 字符
- 140 个 ASCII 字符
- 70 个 UCS-2 字符

如果您发布的消息超出大小配额，Amazon SNS 会将其作为多条消息发送，保证每条消息都符合大小配额。消息以整个词为边界，而不会在一个词的中间截断。单个 SMS 发布操作的总大小配额为 1600 字节。

在发送 SMS 消息时，您可以使用 E.164 格式指定电话号码（用于国际电信的标准电话编号结构）。遵循此格式的电话号码最多可包含 15 位数字，并以加号 (+) 和国家/地区代码作为前缀。例如，E.164 格式的美国电话号码显示为 +1 XXX555 0100。

发送消息（控制台）

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，选择支持短信收发的区域。
3. 在导航窗格中，选择文本消息(SMS)。
4. 在 Mobile text messaging (SMS)（移动文本消息 (SMS)）页面上，选择 Publish text message（发布文本消息）。
5. 在 Publish SMS message（发布 SMS 消息）页面上，对于 Message type（消息类型），选择下列选项之一：

   - Promotional（促销）– 不重要的消息，例如营销消息。
   - Transactional（事务性）– 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。

   > **Note**
   >
   > 此消息级别的设置会覆盖您的账户级别的默认消息类型。您可以从 Mobile text messaging (SMS)（移动文本消息 (SMS)）页面的 Text messaging preferences（文本消息首选项）部分中设置账户级别的默认消息类型。

   有关促销和事务性消息的定价信息，请参阅全球 SMS 定价。
6. 对于 Destination phone number（目标电话号码），请输入您想要向其发送消息的电话号码。
7. 对于 Message（消息），请输入要发送的消息。
8. （可选）在 Origination identities（源身份）下，指定如何向收件人识别自己：

- 要指定 Sender ID（发件人 ID），请键入包含 3-11 个字母数字字符的自定义 ID，其中包括至少一个字母，并且不能包含空格。该发件人 ID 在接收设备上显示为消息发件人。例如，您可以使用自己的企业品牌让消息来源更易于识别。

  对发件人的支持因国家和/或地区 IDs 而异。例如，发送至美国电话号码的消息不显示发件人 ID。有关支持发件人的国家和地区 IDs，请参阅《Amazon End User Messaging SMS 用户指南》 Amazon End User Messaging SMS中支持的 SMS 消息发送的国家和地区。

  如果您未指定发件人 ID，以下标识之一将显示为源身份：

  - 在支持长代码的国家/地区，显示长代码。
  - 在仅支持发 IDs 件人的国家/地区，会显示通知。

  此消息级别的发件人 ID 会覆盖您在 Text messaging preferences 页面中设置的默认发件人 ID。

- 要指定 Origination number（远啊号码），输入一个由 5-14 个数字组成的字符串，以显示为收件人设备上的发件人电话号码。此字符串必须与您在 Amazon Web Services 账户 中为目的地国家/地区配置的起始号码相匹配。起始号码可以是 10DLC 号码、免费电话号码、 person-to-person长码或短码。有关更多信息，请参阅 Amazon SNS 短信的发起身份。

  如果您未指定源号码，Amazon SNS 会根据您的 Amazon Web Services 账户 配置选择用于 SMS 文本消息的源号码。

9. 如果您要向位于印度的收件人发送 SMS 消息，请展开 Country-specific attributes（特定于国家/地区的属性），然后指定以下属性：

   - Entity ID（实体 ID）– 向位于印度的收件人发送 SMS 消息的实体 ID 或委托人实体 (PE) ID。此 ID 是印度电信监管局 (TRAI) 提供的 1–50 个字符组成的唯一字符串，用于识别您在 TRAI 中注册的实体。
   - Template ID（模板 ID）– 向位于印度的收件人发送 SMS 消息的模板 ID。此 ID 是 TRAI 提供的唯一字符串，该字符串包含 1-50 个字符，用于标识您在 TRAI 中注册的模板。模板 ID 必须与您为邮件指定的发件人 ID 相关联。

   有关向位于印度的收件人发送短信的更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 India sender ID registration process。

10. 选择发布消息。

> **Tip**
> 要从源号码发送 SMS 消息，您还可以在 Amazon SNS 控制台导航面板中选择 Origination numbers（源号码）。选择在 Capabilities（功能）列中包含 SMS 的源号码，然后选择 Publish text message（发布文本消息）。

发送消息 (Amazon SDKs)

要使用其中一个发送短信 Amazon SDKs，请使用该软件开发工具包中与 Amazon SNS API 中的`Publish`请求相对应的 API 操作。通过此请求，您可以直接向电话号码发送 SMS 消息。您也可使用 `MessageAttributes` 参数设置以下属性名称的值：

`AWS.SNS.SMS.SenderID`

包含 3-11 个字母数字字符或连字符 (-) 的自定义 ID，其中包含至少一个字母，并且不能包含空格。该发件人 ID 在接收设备上显示为消息发件人。例如，您可以使用自己的企业品牌让消息来源更易于识别。

对发件人的支持因国家或地区 IDs 而异。例如，发送至美国电话号码的消息不显示发件人 ID。有关支持发件人的国家或地区的列表 IDs，请参阅《Amazon End User Messaging SMS 用户指南》Amazon End User Messaging SMS中支持使用发送短信的国家和地区。

如果您未指定发件人 ID，则在支持的国家或地区中长代码会显示为发件人 ID。对于要求使用按字母顺序排列发件人 ID 的国家和地区，NOTICE 显示为发件人 ID。

此消息级别的属性会覆盖您可以通过 `SetSMSAttributes` 请求设置的账户级别属性 `DefaultSenderID`。

`AWS.MM.SMS.OriginationNumber`

一个由 5—14 个数字组成的自定义字符串，其中可以包括一个可选的前导加号 (+)。此数字字串在接收设备上显示为发件人的电话号码。该字符串必须与您的 Amazon 账户中为目的地国家/地区配置的发件人号码相匹配。发起号码可以是 10DLC 号码、免费电话号码、 person-to-person (P2P) 长码或短码。有关更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Phone numbers。

如果您未指定发货号，Amazon SNS 会根据 Amazon 您的账户配置选择发货号。

AWS.SNS.SMS.MaxPrice

您愿意用于发送 SMS 消息的最高价格（以美元为单位）。如果 Amazon SNS 确定发送消息会产生超出您最高价格的成本，则它不会发送消息。

如果您的 month-to-date SMS 费用已超过为该属性设置的配额，则此`MonthlySpendLimit`属性无效。您可以使用 `SetSMSAttributes` 请求设置 `MonthlySpendLimit` 属性。

如果您发送消息至 Amazon SNS 主题，则该最高价格适用于传输给订阅该主题的每个电话号码的每条消息。

AWS.SNS.SMS.SMSType

要发送的消息类型：

- **Promotional**（默认）– 不重要的消息，例如营销消息。
- **Transactional** – 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。

此消息级别的属性会覆盖您可以通过 `SetSMSAttributes` 请求设置的账户级别属性 `DefaultSMSType`。

AWS.MM.SMS.EntityId

只有向位于印度的收件人发送 SMS 消息时，才需要此属性。

这是您用于向位于印度的收件人发送 SMS 消息的实体 ID 或委托人实体 (PE) ID。此 ID 是印度电信监管局 (TRAI) 提供的 1–50 个字符组成的唯一字符串，用于识别您在 TRAI 中注册的实体。

AWS.MM.SMS.TemplateId

只有向位于印度的收件人发送 SMS 消息时，才需要此属性。

这是您向位于印度的收件人发送 SMS 消息的模板。此 ID 是 TRAI 提供的唯一字符串，该字符串包含 1-50 个字符，用于标识您在 TRAI 中注册的模板。模板 ID 必须与您为邮件指定的发件人 ID 相关联。

发送消息

以下代码示例演示了如何使用 Amazon SNS 发布 SMS 消息。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 Publish。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 Publish。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 Publish。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 Publish。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 Publish。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 Publish。

设置 Amazon SNS 中的短信收发首选项

使用 Amazon SNS 指定 SMS 消息的首选项。例如，您可以指定是否要优化传输以确保成本或可靠性、您的每月支出限额、如何记录传输以及是否要订阅每日 SMS 使用情况报告。

这些首选项对从您的账户发送的每个 SMS 消息有效，但在您发送各条消息时可覆盖部分设置。有关更多信息，请参阅 使用 Amazon SNS 向手机发布短信。

使用 Amazon Web Services Management Console设置发送 SMS 消息的首选项

1. 登录 Amazon SNS 控制台。
2. 选择支持 SMS 消息收发的区域。
3. 在导航面板上，选择移动，然后选择文本消息(SMS)。
4. 在移动文本消息(SMS) 页上，在文本消息发送首选项部分中，选择编辑。
5. 在 Edit text messaging preferences (编辑文本消息发送首选项) 页上，在 Details (详细信息) 部分中，执行以下操作：

   a. 对于默认消息类型，选择下列选项之一：

      - 促销（默认）– 非重要消息（例如营销消息）。Amazon SNS 以产生最低成本为基准来优化消息传输。
      - 事务性（默认）– 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。Amazon SNS 以实现最高可靠性为基准来优化消息传输。

有关促销和事务处理消息的定价信息，请参阅全球 SMS 定价。

b. （可选）对于 Account spend limit (账户花费限额)，请输入您在每个日历月想要为 SMS 消息支付的金额，以 USD 为单位。

> **Important**
>
> - 默认情况下，支出配额设为 1.00 USD。如果要提高服务配额，请提交请求。
> - 如果在控制台中设置的金额超过您的服务配额，Amazon SNS 会停止发布 SMS 消息。
> - 由于 Amazon SNS 是分布式系统，它会在超过支出配额的几分钟内停止发送 SMS 消息。在该间隔内，如果您继续发送 SMS 消息，可能会产生超出配额的成本。

6. （可选）对于默认发件人 ID，请输入一个自定义 ID（如您的企业品牌），它显示为接收设备的发送者。

> **Note**
>
> 对发件人的支持因国家/地区 IDs 而异。

7. （可选）输入 Amazon S3 bucket name for usage reports（使用情况报告的 Amazon S3 存储桶名称）的名称。

> **Note**
>
> 亚马逊 S3 存储桶策略必须授予对亚马逊 SNS 的写入权限。

8. 选择 Save changes（保存更改）。

设置首选项 (Amazon SDKs)

要使用其中一个来设置您的短信偏好 Amazon SDKs，请使用该软件开发工具包中与 Amazon SNS `SetSMSAttributes` API 中的请求相对应的操作。通过此请求，您可以将值分配给不同的 SMS 属性，例如您的每月支出配额和默认 SMS 类型（促销或事务）。有关所有短信属性，请参阅《亚马逊简单通知服务 API 参考》SMSAttributes中的 “设置”。

以下代码示例演示如何使用 `SetSMSAttributes`。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

如何使用 Amazon SNS 设置默认SMSType 属性。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考SMSAttributes中的设置。

CLI

Amazon CLI

设置 SMS 消息属性

以下 `set-sms-attributes` 示例将 SMS 消息的默认发件人 ID 设置为 `MyName`。

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅SMSAttributes《Amazon CLI 命令参考》中的 “设置”。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SMSAttributes中的设置。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing
 messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考SMSAttributes中的设置。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

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

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考SMSAttributes中的设置。

为特定于国家/地区的传送设置短信收发首选项

您可以通过仅向特定目标国家/地区发送消息来管理和控制短信流量。这样可以确保您的消息仅发送到经批准的国家/地区，从而避免不必要的短信费用。以下说明使用 Amazon Pinpoint 的保护配置来指定您想要允许或阻止的国家/地区。

1. 打开 Amazon SMS 控制台，网址为https://console.aws.amazon.com/sms-voice/。
2. 在导航窗格的概述下的快速入门部分，选择创建保护配置。
3. 在 “保护配置详细信息” 下，为您的保护配置输入适合企业使用的名称（例如， Allow-Only-AU）。
4. 在短信国家/地区规则下，选中地区/国家复选框以阻止向所有支持的国家/地区发送消息。
5. 取消选中要将消息发送到的国家/地区的复选框。例如，要仅允许向澳大利亚发送消息，请取消选中澳大利亚对应的复选框。
6. 在保护配置关联部分的关联类型下，选择账户默认。这将确保 Amazon End User Messaging SMS 保护配置影响通过亚马逊 SNS、Amazon Cogn ito 和 Amazon Pinpoint API 调用发送的所有消息。`SendMessages`
7. 选择创建保护配置，保存您的设置。

   将显示以下确认消息：

   ```
   Success Protect configuration protect-abc0123456789 has been created.
   ```

8. 登录 Amazon SNS 控制台。
9. 向其中一个被阻止的国家/地区（例如印度）发布消息。

   该消息将不会被传送。您可以使用在传送失败日志中对此进行验证CloudWatch。在日志组sns/region/AccountID/DirectPublishToPhoneNumber/Failure中搜索类似于以下示例的响应：

   ```
   {
   ```

```
"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"
}
```

管理 Amazon SNS 电话号码和订阅

Amazon SNS 提供了多个用于管理账户 SMS 消息接收者的选项。在限定的频率内，您可以将已选择不从您的账户接收 SMS 消息的电话号码重新加入。若要停止向 SMS 订阅发送消息，您可以删除订阅或发布至订阅的主题。

退出接收 SMS 消息

如果当地法律和法规有要求（例如美国和加拿大），SMS 收件人可以使用自己的设备，通过向该消息回复以下内容来表示退出：

- ARRET（法语）
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

要退出，收件人必须回复 Amazon SNS 用于传输消息的相同发起号码。选择退出后， Amazon Web Services 账户 除非您选择使用电话号码，否则收件人将不再收到您发送的 SMS 消息。

如果电话号码订阅了 Amazon SNS 主题，退出不会删除订阅，而短信将无法传输至该订阅，除非您重新加入其电话号码。

使用 Amazon SNS 控制台管理手机号码和订阅

您可以使用 Amazon SNS 控制台控制哪些电话号码从您的账户接收 SMS 消息。

加入已选择退出 Amazon SNS 控制台的电话号码

您可以查看哪些电话号码已退出来自您的账户的短信，并重新加入这些电话号码，以便继续向其发送消息。

对于每个电话号码，您只能每隔 30 天重新加入一次。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单上，将区域选择器设置为支持 SMS 消息收发的区域。
3. 在导航面板上，选择文本消息(SMS)。
4. 在移动文本消息(SMS) 页面的已退出的电话号码部分中，将显示已退出的电话号码。
5. 选中您想要加入的电话号码的复选框，然后选择选择加入。电话号码将不再处于退出状态，并将接收您发送的 SMS 消息。

在 Amazon SNS 控制台中删除短信订阅

删除 SMS 订阅可停止在您发布至主题时向该电话号码发送 SMS 消息。

1. 在导航面板中，选择订阅。
2. 选中要删除的订阅的复选框。然后选择 Actions，再选择 Delete Subscriptions。
3. 在 Delete（删除）窗口中，选择 Delete（删除）。Amazon SNS 将删除订阅并显示成功消息。

在 Amazon SNS 控制台中删除主题

当您不想再向其订阅终端节点发布消息时，可删除主题。

1. 在导航面板上，选择主题。
2. 选中要删除的主题的复选框。然后选择 Actions，再选择 Delete Topics。
3. 在 Delete（删除）窗口中，选择 Delete（删除）。Amazon SNS 删除主题并显示成功消息。

使用 Amazon SDK 管理手机号码和订阅

您可以使用向 Amazon SNS 发出编程请求，并管理哪些电话号码可以从您的账户接收短信。 Amazon SDKs

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置Amazon SDKs 和凭据文件。

使用 SDK 查看所有已选择退出的电话号码 Amazon

要查看所有已退出的电话号码，请使用 Amazon SNS API 提交 `ListPhoneNumbersOptedOut` 请求。

以下代码示例演示如何使用 `ListPhoneNumbersOptedOut`。

CLI

Amazon CLI

列出 SMS 消息退出

以下 `list-phone-numbers-opted-out` 示例将列出退出 SMS 消息接收的电话号码。

```
aws sns list-phone-numbers-opted-out
```

输出：

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListPhoneNumbersOptedOut中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListPhoneNumbersOptedOut中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考ListPhoneNumbersOptedOut中的。

使用 SDK 检查电话号码是否已选择退出 Amazon

要检查电话号码是否退出，请使用 Amazon SNS API 提交 `CheckIfPhoneNumberIsOptedOut` 请求。

以下代码示例演示如何使用 `CheckIfPhoneNumberIsOptedOut`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考CheckIfPhoneNumberIsOptedOut中的。

CLI

Amazon CLI

检查电话号码的 SMS 消息退出

以下`check-if-phone-number-is-opted-out`示例检查指定的电话号码是否已选择不接收来自当前 Amazon 账户的 SMS 消息。

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

输出：

```
{
    "isOptedOut": false
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CheckIfPhoneNumberIsOptedOut中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CheckIfPhoneNumberIsOptedOut中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CheckIfPhoneNumberIsOptedOut中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考CheckIfPhoneNumberIsOptedOut中的。

加入已选择退出 Amazon SNS API 的电话号码

要选择加入电话号码，请使用 Amazon SNS API 提交 `OptInPhoneNumber` 请求。

对于每个电话号码，您只能每隔 30 天重新加入一次。

使用 S Amazon DK 删除短信订阅

要从 Amazon SNS 主题删除 SMS 订阅，请使用 Amazon SNS API 提交 `ListSubscriptions` 请求来获取订阅 ARN，然后将该 ARN 传递给 `Unsubscribe` 请求。

以下代码示例演示如何使用 `Unsubscribe`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

通过订阅 ARN 取消订阅某个主题。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 Unsubscribe。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的 Unsubscribe。

CLI

Amazon CLI

从主题取消订阅

以下 `unsubscribe` 示例将从主题删除指定的订阅。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 Unsubscribe。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 Unsubscribe。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 Unsubscribe。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 Unsubscribe。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考 中的 Unsubscribe。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 Unsubscribe。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的 Unsubscribe。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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.")
```

- 有关 API 的详细信息，请参阅适用于 S wift 的Amazon SDK API 参考中的取消订阅。

使用 Amazon SDK 删除主题

要删除主题及其所有订阅，请使用 Amazon SNS API 提交 `ListTopics` 请求来获取主题 ARN，然后将该 ARN 传递给 `DeleteTopic` 请求。

以下代码示例演示如何使用 `DeleteTopic`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按主题 ARN 删除主题。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DeleteTopic中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DeleteTopic中的。

CLI

Amazon CLI

删除 SNS 主题

以下 `delete-topic` 示例将删除指定的 SNS 主题。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteTopic中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考DeleteTopic中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteTopic中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

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

```
  // }
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteTopic中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

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

- 有关 API 的详细信息，请参阅适用DeleteTopic于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DeleteTopic中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 Python 的Amazon SDK (Boto3) API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 S AP 的Amazon SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSSNS

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

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

- 有关 API 的详细信息，请参阅适用于 S wift 的Amazon SDK API 参考DeleteTopic中。

Amazon SNS 短信活动监控

通过监控您的 SMS 活动，您可以跟踪目标电话号码、成功或失败的传输、失败的原因、成本及其他信息。Amazon SNS 通过在控制台中汇总统计数据、向亚马逊 CloudWatch发送信息以及将每日短信使用报告发送到您指定的亚马逊 S3 存储桶来提供帮助。

查看 Amazon SNS 短信传输统计数据

您可以使用 Amazon SNS 控制台查看您近期 SMS 传输的统计数据。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单上，将区域选择器设置为支持 SMS 消息收发的区域。
3. 在导航面板上，选择文本消息(SMS)。
4. 在 Text messaging (SMS) (文本消息(SMS)) 页面上的 Account stats (账户统计数据) 部分中，查看关于您的事务处理和促销 SMS 信息传输的图表。每个图表显示之前 15 天内的以下数据：
   - 传输率（成功传输的百分比）
   - 发送（传输尝试的数量）
   - 失败（传输失败的数量）

在此页面，您还可以选择 Usage（使用量）按钮转到存储您的每日使用情况报告的 Amazon S3 存储桶。有关更多信息，请参阅 订阅 Amazon SNS 每日短信使用量报告。

使用亚马逊 CloudWatch 指标和日志监控 Amazon SNS 短信发送情况

您可以使用 Amazon CloudWatch 和 Amaz CloudWatch on Logs 来监控您的短信发送情况。

查看亚马逊 CloudWatch 指标

Amazon SNS 会自动收集有关您发送短信的指标并将其推送到亚马逊。 CloudWatch您可以使用 CloudWatch 监控这些指标并创建警报，以便在指标超过阈值时提醒您。例如，您可以监控 CloudWatch 指标以了解您的短信发送率和 month-to-date短信费用。

有关监控 CloudWatch 指标、设置 CloudWatch 警报和可用指标类型的信息，请参阅使用监控亚马逊 SNS 主题 CloudWatch。

查看 CloudWatch 日志

通过启用 Amazon SNS 写入亚马逊 CloudWatch 日志，您可以收集有关成功和不成功发送短信的信息。对于您发送的每条 SMS 消息，Amazon SNS 会写入日志，其中包括消息价格、成功或失败状态、失败原因（如果消息发送失败）、消息停留时间以及其他信息。

启用和查看 SMS 消息的 CloudWatch 日志

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单上，将区域选择器设置为支持 SMS 消息收发的区域。
3. 在导航面板上，选择文本消息(SMS)。
4. 在移动文本消息(SMS) 页上，在文本消息发送首选项部分中，选择编辑。
5. 在下一页上，展开 Delivery status logging（传输状态日志记录）部分。
6. 对于成功采样率，请指定 Amazon SNS 将在日志中 CloudWatch 写入日志的成功发送短信的百分比。例如：

   - 要仅将失败传输写入日志，请将此值设为 0。
   - 要将 10% 的成功传输写入日志，请将其设为 10。

   如果您不指定百分比，Amazon SNS 会将所有成功传输写入日志。
7. 为提供所需的权限，请执行以下操作之一：

   - 要创建新的服务角色，请选择 Create new service role（创建新的服务角色），然后选择 Create new roles（创建新角色）。在下一页上，选择 Allow（允许）以授予 Amazon SNS 对您账户资源的写入访问权限。
   - 要使用现有服务角色，请选择 Use existing service role（使用现有服务角色），然后将 ARN 名称粘贴到 IAM role for successful and failed deliveries（成功和失败传输的 IAM 角色）框中。

     您指定的服务角色必须允许对账户资源进行写入访问。有关创建 IAM 角色的更多信息，请参阅 IAM 用户指南中的为 Amazon 服务创建角色。
8. 选择 Save changes（保存更改）。
9. 回到 Mobile text messaging (SMS)（移动文本消息 (SMS)）页面上，转至 Delivery status logs（传输状态日志）部分查看任何可用的日志。

> **Note**
>
> 根据目标电话号码的运营商，Amazon SNS 控制台中显示传输日志最长可能需要 72 小时。

成功 SMS 传输的示例日志

成功 SMS 传输的传输状态日志与下面的示例类似：

```
{
    "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"
}
```

失败 SMS 传输的示例日志

失败 SMS 传输的传输状态日志与下面的示例类似：

```
{
    "notification": {
        "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
        "timestamp": "2016-06-28 00:40:34.559"
    },
    "delivery": {
        "mnc": 0,
```

```
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 0,
        "providerResponse": "Unknown error attempting to reach phone",
        "dwellTimeMs": 1420,
        "dwellTimeMsUntilDeviceAck": 1692
    },
    "status": "FAILURE"
}
```

SMS 传输失败的原因

`providerResponse` 属性中提供失败的原因。SMS 消息传输失败可能是因为以下原因：

- 被电话运营商作为垃圾消息屏蔽
- 目的地位于阻止列表中
- 电话号码无效
- 消息正文无效
- 电话运营商已屏蔽此消息
- 电话运营商目前无法访问/不可用
- 电话已屏蔽 SMS
- 电话位于阻止列表中
- 电话当前无法访问/可用
- 电话号码已退出
- 此传输会超过最高价格
- 尝试联系电话时发生未知错误

订阅 Amazon SNS 每日短信使用量报告

您可以通过从 Amazon SNS 订阅每日使用量报告来监控您的 SMS 消息传输。在每个您至少发送了一条 SMS 消息的日子，Amazon SNS 都将向您指定的 Amazon S3 存储桶发送 CSV 文件格式的使用情况报告。在 Amazon S3 存储桶中提供 SMS 使用情况报告需要 24 小时。

每日使用量报告信息

该使用情况报告包括通过您的账户发送的每条 SMS 消息的以下信息。

请注意，此报告不包含发送到已选择退出的收件人的消息。

- 消息发布时间（UTC 时间）
- 消息 ID
- 目标电话号码
- 消息类型
- 传输状态
- 消息价格 (USD)
- 分段编号（如果一条消息过长，则会拆分为多个分段）
- 分段总数

> **Note**
>
> 如果 Amazon SNS 没有收到部分编号，我们将其值设置为零。

订阅每日使用量报告

要订阅每日使用情况报告，您必须通过适当的权限创建 Amazon S3 存储桶。

为您的每日使用情况报告创建 Amazon S3 存储桶

1. 从发送短信 Amazon Web Services 账户 的，登录 Amazon S3 控制台。
2. 选择创建存储桶。
3. 对于存储桶名称，我们建议您输入对账户和组织唯一的名称。例如，使用模式 `<my-bucket-prefix>-<account_id>-<org-id>`。

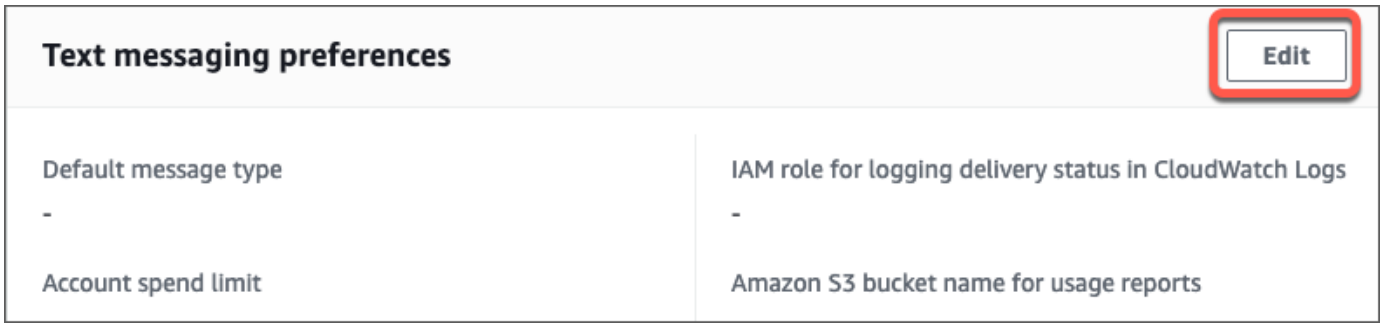
   有关存储桶名称约定和限制的信息，请参阅 Amazon Simple Storage Service 用户指南中的存储桶命名规则。
4. 选择创建。
5. 在 All Buckets（所有存储桶）表中，选择存储桶。
6. 在 Permissions（权限）部分中，选择 Bucket policy（存储桶策略）。
7. 在 Bucket Policy Editor（存储桶策略编辑器窗口中，提供允许 Amazon SNS 服务委托人写入您的存储桶的策略。有关示例，请参阅存储桶策略的示例。

如果您使用示例策略，请记得*my-s3-bucket*使用您在步骤 3 中选择的存储桶名称进行替换。

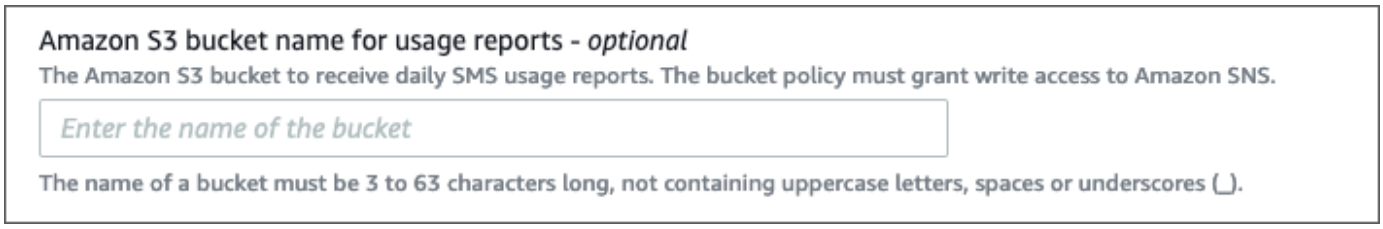
8. 选择保存。

订阅每日使用情况报告

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择文本消息(SMS)。
3. 在 Text messaging (SMS) (文本消息收发(SMS)) 页上，在 Text messaging preferences (文本消息收发首选项) 部分中，选择 Edit (编辑)。



4. 在 Edit text messaging preferences (编辑文本消息收发首选项) 页上，在 Details (详细信息) 部分中，指定 Amazon S3 bucket name for usage reports (使用率报告的 Amazon S3 存储桶名称)。



5. 选择保存更改。

存储桶策略的示例

以下策略允许 Amazon SNS 服务委托人执行 `s3:PutObject`、`s3:GetBucketLocation` 和 `s3:ListBucket` 操作。

Amazon 为所有服务提供工具，其服务委托人已被授予访问您账户中资源的权限。当 Amazon S3 存储桶策略声明中的主体是混淆代理问题时。要限制哪些存储桶中的哪些区域和账户可以接收每日使用情况报告，请使用 `aws:SourceArn`，如下面的示例所示。如果您不想限制可生成这些报告的区域，请使用 `aws:SourceAccount` 限制生成报告的账户。如果您不知道源的 ARN，请参阅 `aws:SourceAccount`。

在您创建 Amazon S3 存储桶以便从 Amazon SNS 接收每日 SMS 使用情况报告时，请使用以下包含混淆代理人保护的示例。

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
                }
            }
        },
        {
            "Sid": "AllowGetBucketLocation",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:GetBucketLocation",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
                }
            }
        },
        {
            "Sid": "AllowListBucket",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
                }
            }
        }
    ]
}
```

> **Note**
>
> 您可以将使用情况报告发布到 Amazon S3 存储桶，这些存储桶由 Amazon S3 策略中 `Condition` 元素指定的 Amazon Web Services 账户 拥有。要将使用情况报告发布到其他人 Amazon Web Services 账户 拥有的 Amazon S3 存储桶，请参阅如何从另一个存储桶复制 Amazon S3 对象 Amazon Web Services 账户？。

每日使用量报告的示例

在您订阅每日使用量报告后，Amazon SNS 会每天将包含使用量数据的 CSV 文件放在以下位置：

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

每个文件可包含最多 50000 条记录。如果一天内的记录超出此配额，则 Amazon SNS 会添加多个文件。下面显示了一个示例报告：

```
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
```

为 Amazon SNS 短信收发请求支持

> **Important**
>
> 《Amazon SNS 短信开发人员指南》已更新。Amazon SNS 已与 Amazon End User Messaging SMS 集成，用于传输短信。此指南包含有关如何创建、配置和管理 Amazon SNS 短信的最新信息。

在您联系之前，Amazon SNS 的某些短信选项不适用于您的 Amazon 账户。 Amazon Web Services 支持在 Amazon Web Services 支持 中心中创建一个案例来请求以下任意项目：

- 提高您的每月短信支持阈值

  默认情况下，每月支出阈值设为 1.00 美元 (USD)。您的支出阈值决定您可以使用 Amazon SNS 发送的消息量。您可以为您的 SMS 使用案例请求符合预计的每月消息量的支出阈值。
- 从 SMS 沙盒迁移，以便您可以不受限制地发送 SMS 消息。有关更多信息，请参阅 移出 Amazon SNS 短信沙盒。
- 专用源号码
- 专用发件人 ID。发送人 ID 是一个自定义 ID，它在接收人的设备上显示为发送人。例如，您可以使用自己的企业品牌让消息来源更易于识别。对发件人的支持因国家或地区 IDs 而异。有关更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Supported countries and regions for SMS messaging with Amazon End User Messaging SMS。

请求提升您的每月 Amazon SNS 短信支出配额

Amazon SNS 提供支出配额，以帮助您管理使用账户发送 SMS 所产生的每月最高成本。支出配额限制了您在遭受恶意攻击时的风险，并防止上游应用程序发送超出预期的消息。如果 Amazon SNS 确定发送 SMS 消息会产生超出当月支出配额的费用，则您可以将其配置为停止发布 SMS 消息。

为确保您的运营不受影响，我们建议申请足够高的支出配额，以支持您的生产工作负载。有关更多信息，请参阅步骤 1：开立 Amazon SNS SMS 案例。收到配额后，您可以通过应用完整配额或较小的值来管理风险，如步骤 2：更新 SMS 设置中所述。通过应用一个较低的值，您可以控制您的每月支出，并在必要时选择进行纵向扩展。

> **Important**
>
> 由于 Amazon SNS 是分布式系统，它会在超过支出配额的几分钟内停止发送 SMS 消息。在此期间内，如果您继续发送短信，可能会产生超出限额的费用。

我们将所有新账户的支出限额设定为每月 1.00 美元 (USD)。此配额旨在让您测试 Amazon SNS 的消息发送功能。要申请增加您账户的短信支出配额，请在 Support Center 中提交增加配额的 Amazon 案例。

主题

- 步骤 1：开立 Amazon SNS SMS 案例
- 步骤 2：在 Amazon SNS 控制台中更新您的 SMS 设置

步骤 1：开立 Amazon SNS SMS 案例

您可以在 Support Center 中提交配额增加案例，申请增加每月支出 Amazon 配额。

> **Note**
>
> 请求表中的某些字段将标记“可选”。但是， Amazon Web Services 支持 需要以下步骤中提到的所有信息，才能处理您的请求。如果您没有提供所有必需的信息，可能会在处理请求期间遇到延迟。

1. 登录 Amazon Web Services Management Console 到 https://console.aws.amazon.com/。
2. 在 Support (支持) 菜单上，选择 Support Center (支持中心)。
3. 在您的支持案例窗格上，选择创建案例。
4. 选择想要提高服务限制？链接，然后完成以下操作：
   - 对于限制类型，选择 SNS 短信。
   - （可选）对于提供指向将发送 SMS 消息的网站或应用程序的链接，提供有关将发送 SMS 消息的网站、应用程序或服务的信息。
   - （可选）对于您计划发送什么类型的消息，选择您计划使用长代码发送的消息类型：
     - 一次性密码 – 提供您的客户用于向您的网站或应用程序进行身份验证的密码的消息。
     - 促销 – 宣传您的业务或服务的非关键性消息，如特别优惠或公告。

  - 事务性 – 为客户事务提供支持的重要信息性消息，如订单确认或账户提醒。事务性消息不得包含促销或营销内容。
- （可选）对于您要从哪个 Amazon 区域发送消息，请选择您要从哪个区域发送消息。
- （可选）对于您计划将消息发送到的国家/地区，输入您要在其中购买短代码的国家或地区。
- （可选）在您的客户如何选择接收您的消息中，提供有关您的选择加入流程的详细信息。
- （可选）在请提供您计划用于向客户发送消息的消息模板字段中，包括您将要使用的模板。

5. 在请求下，填写以下部分：

   - 对于区域，选择您要从中发送消息的区域。

     > **Note**
     >
     > 请求部分中必须填写“区域”。即使您在案例详情部分中提供了这些信息，也必须在此处包含这些信息。

   - 对于资源类型，选择一般限制。
   - 对于限制，选择提高账户支出阈值。
6. 对于“新限额值”，请输入您可在每个日历月为短信支付的最高金额（以 USD 为单位）。
7. 在案例描述下，对于使用情形描述，提供以下详细信息：

   - 发送短信的公司或服务的网站或应用程序。
   - 您的网站或应用程序提供的服务以及您的短信有助于该服务的方式。
   - 用户注册以自愿接收您的网站、应用程序或其他位置上的短信的方式。

   如果您请求的支出限额（您为新限额值指定的值）超出 10000 美元 (USD)，请针对您要向其发送消息的每个国家/地区提供以下其他详细信息：

   - 您使用的是发件人 ID 还是短代码。如果使用的是发件人 ID，请提供：
     - 发件人 ID。
     - 此发件人 ID 是否已向该国家/地区的无线运营商注册。
   - 您的消息传递的最大预期值 transactions-per-second (TPS)。
   - 平均消息大小。
   - 您将发送到该国家/地区的消息的模板。

- （可选）字符编码需求（如果有）。

8. （可选）如果您想提交其他任何请求，请选择添加其他请求。如果包含多个请求，请提供每个请求所需的信息。有关所需信息，请参阅为 Amazon SNS 短信收发请求支持内的其他部分。
9. 在联系选项下，对于首选联系语言，请选择您希望接收有关此案例的通信时使用的语言。
10. 完成后，选择提交。

Amazon Web Services 支持 团队会在 24 小时内对您的请求做出初步回应。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们要仔细考虑每个请求。如果我们可以，我们将在 24 小时内准予您的请求。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。

如果您的使用案例与我们的策略不符，我们可能无法准予您的请求。

步骤 2：在 Amazon SNS 控制台中更新您的 SMS 设置

在我们通知您的每月支出配额已提升后，您必须在 Amazon SNS 控制台中调整您账户的支出配额。

> **Important**
>
> 您必须完成以下步骤，否则您的 SMS 支出限额将不会增加。

在控制台中调整您的支出限额

1. 登录 Amazon SNS 控制台。
2. 打开左侧导航菜单，展开移动，然后选择文本消息 (SMS)。
3. 在移动文本消息(SMS) 页上，在文本消息发送首选项部分中，选择编辑。
4. 在编辑文本消息首选项页面上的详细信息部分中，对于账户支出限额字段，输入新的 SMS 支出限额。

   > **Note**
   >
   > 您可能会看到一条警告，指出输入的值大于默认支出限额。您可以忽略此警告。

5. 选择 Save changes（保存更改）。

> **Note**
>
> 如果您收到“Invalid Parameter”（参数无效）错误，请检查 Amazon Support 的联系人并确认您输入的新 SMS 支出限额正确无误。如果您仍然遇到问题，请在 Support Center 中 Amazon 提出问题。

当您在 Amazon Web Services 支持 中心创建案例时，请务必包含所提交的请求类型的所有必填信息。否则，在继续操作之前， Amazon Web Services 支持 必须与您联系以获取此信息。通过提交详细案例，您可帮助确保案例即时完成。有关特定类型的 SMS 请求所需的详细信息，请参阅以下主题。

有关发件人的更多信息 IDs，请参阅《Amazon End User Messaging SMS 用户指南》中的以下文档：

Amazon End User Messaging SMS 话题	描述
请求提升支出配额	您的支出配额决定了您 Amazon End User Messaging SMS 每月可以花多少钱发送短信。
在支持中心开立案例，获取发件人 ID	如果您计划向需要发件人的国家/地区的收件人 IDs 发送邮件，则可以通过在 Amazon Web Services 支持 中心创建新问题来申请发件人 ID。

Amazon SNS 短信最佳实践

> **Important**
>
> 《Amazon SNS 短信开发人员指南》已更新。Amazon SNS 已与 Amazon End User Messaging SMS 集成，用于传输短信。此指南包含有关如何创建、配置和管理 Amazon SNS 短信的最新信息。

手机用户对于未经请求的短信的容忍度往往非常低。未经请求的短信活动的响应率将几乎始终较低，因此，您的相应投资回报将会较差。

此外，手机运营商会持续审核批量短信发件人。他们会从其确定发送未经请求的消息的数量中限制或阻止消息。

发送未经请求的内容也是一种违反 Amazon 可接受使用策略的行为。Amazon SNS 团队会定期审核 SMS 活动，而且会在出现您发送未经请求的消息的情况下限制或阻止您发送消息的能力。

最后，在许多国家、地区和司法管辖区中，对于发送未经请求的短信设有严重处罚。例如，在美国，电话消费者保护法案 (TCPA) 规定，消费者对于其收到的每条未经请求的消息都有权享有 500-1,500 美元的赔偿费用 (由发件人支付)。

本部分介绍了几个可帮助您提升客户参与度并避免代价高昂的处罚的最佳实践。但请注意，本节不包含法律建议。务必咨询律师来获取法律建议。

遵守法律、法规和运营商要求

如果您违反客户所在地的法律和法规，您可能面对重大罚款和处罚。因此，务必了解您开展业务的每个国家/地区内与短信收发相关的法律。

以下列表包含一些链接，这些链接指向适用于全球主要市场内的短信通信的关键法律。

- 美国：1991 年《电话消费者保护法案》（简称 TCPA）适用于特定类型的短信。有关更多信息，请访问美国联邦通信委员会 (Federal Communications Commission) 网站上的规则和法规。
- 英国：2003 年《隐私与电子通信（EC 指令）条例》（简称 PECR）适用于特定类型的短信。有关更多信息，请访问英国信息委员会办公室的网站，查看什么是 PECR？。
- 欧盟：2002 年《隐私与电子通信条例》（有时候称为 ePrivacy 指令）适用于特定类型的短信。有关更多信息，请访问 Europa.eu 网站，查看该法律的完整文本。
- 加拿大：《打击 Internet 和无线垃圾邮件法案》（通常称为加拿大反垃圾邮件法律或 CASL）适用于特定类型的短信。有关更多信息，请访问加拿大国会网站，查看该法律的完整文本。
- 日本：《有关特定电子邮件传输规定的法案》适用于特定类型的短信。有关更多信息，请访问日本内务与通信省 (Japanese Ministry of Internal Affairs and Communications) 网站，查看日本的垃圾邮件应对措施。

作为发件人，即使您的公司或部门并非位于这些国家/地区之一，这些法律也可能适用于您。此列表中的某些法律最初是为解决未经请求的电子邮件或电话呼叫而制定的，但它们已被解释或扩展为同样适用于短信。其他国家/地区可能有自己的与短信传输相关的法律。请咨询您的客户所在的每个国家/地区的律师以获得法律建议。

在许多国家/地区，本地运营商具有确定哪些类型的流量可通过其网络传输的最终权力。这意味着运营商可能会对 SMS 内容施加比当地法律最低要求更严格的限制。

获取权限

在您计划发送特定类型的消息时，切勿向未明确要求接收此类消息的收件人发送消息。不要共享选择加入名单，即使在同一家公司内的部门之间也是如此。

如果收件人可以使用在线表格进行注册以接收您的消息，请添加预防系统，防止自动化脚本在人员不知道的情况下进行订阅。您还应该限制用户在单个会话中可以提交某个电话号码的次数。

当您收到短信选择加入请求时，请向收件人发送消息，要求他们确认希望接收来自您的消息。请勿在收件人确认订阅之前向其发送任何其他消息。订阅确认消息可能类似于以下示例：

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

维护包含每个选择加入请求和确认的日期、时间和来源的记录。这在运营商或监管机构请求它的情况下可能会有用，并且还可以帮助您执行客户列表的例行审核。

选择加入工作流程

在某些情况下（例如美国免费电话或短代码注册），移动运营商要求您提供整个选择加入工作流程的模型或屏幕截图。模型或屏幕截图必须与收件人将要完成的选择加入工作流程非常接近。

您的模型或屏幕截图应包括下面列出的所有必要披露信息，以保持最高的合规水平。

必要的披露信息

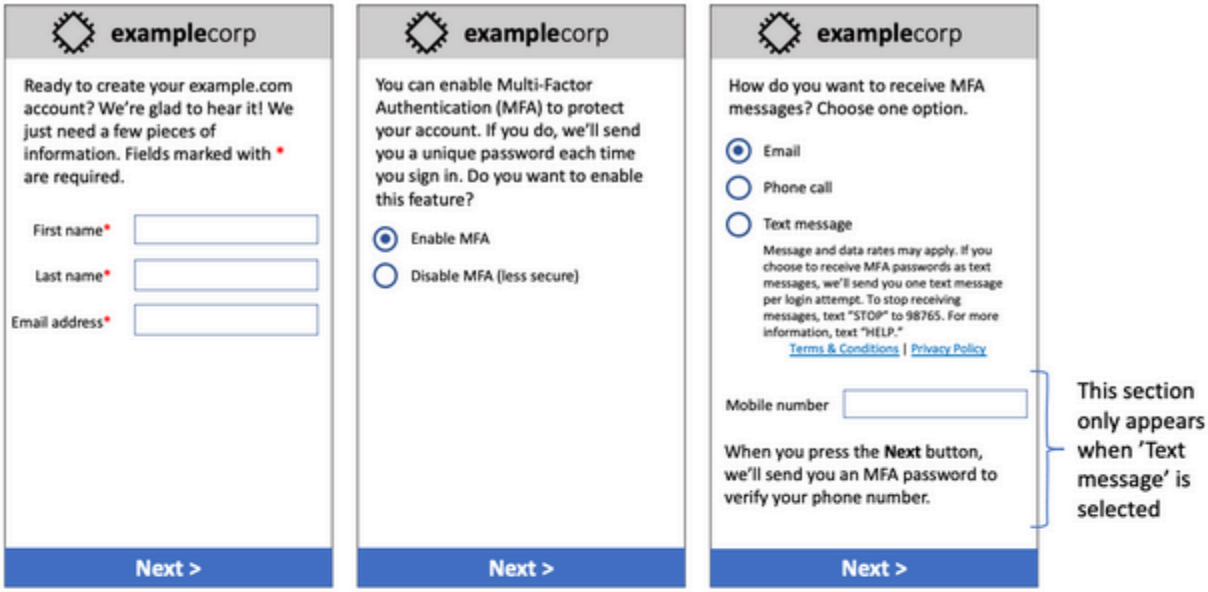
- 对您将通过程序发送的消息使用场景的描述。
- 陈述“可能会收取消息和数据费用”。
- 说明收件人收到您的消息的频率。例如，定期消息发送程序可以说明“每周一条消息”。一次性密码或多重验证使用场景可以说明“消息频率会有变化”或“每次登录尝试一条消息”。
- 指向您的条款和条件以及隐私策略文档的链接。

不合规的选择加入方法的常见拒绝原因

- 提供的公司名称与模型或屏幕截图中提供的名称不相符。在选择加入工作流程描述中需要解释任何不明确的关系。
- 似乎会向收件人发送消息，但事先并未征得明确同意。所有消息发送都必须征得明确同意。
- 似乎需要接收短信才能注册服务。如果工作流程没有提供其他形式（例如，电子邮件或语音呼叫）作为选择加入消息的替代方法，则这不合规。

- 关于选择加入的说明完全包含在服务条款中。披露内容应始终在选择加入时提交给接收人，而不是包含在关联的策略文档中。
- 客户同意接收您发送的一种类型的消息，而您向他们发送了其他类型的文本消息。例如，他们同意接收一次性密码，但还会向其发送投票和调查消息。
- 没有向收件人提供所需的披露信息（如上所列）。

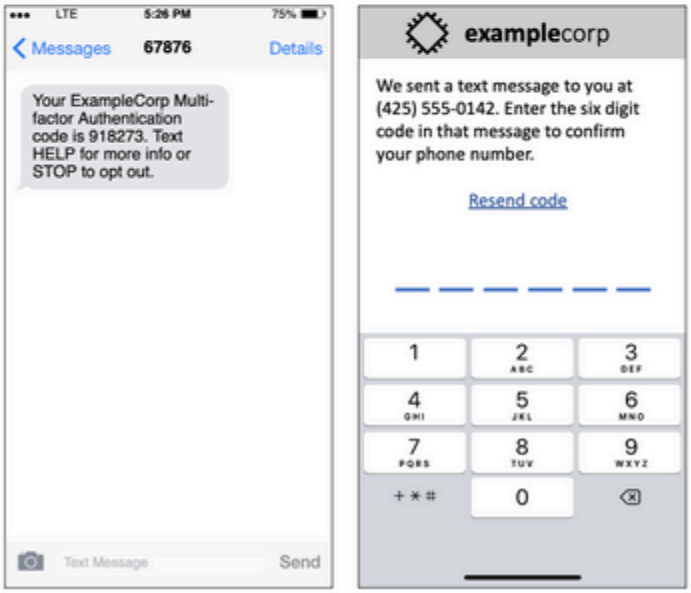
以下示例符合移动运营商对多重验证使用场景的要求。



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.

多重身份验证应用场景的模型

它包含最终的文本和图像，展示了完整的选择加入流程，并附有注释。在选择加入流程中，客户必须采取意图明确的操作来表示同意接收文本消息，且流程中包含所有必需的披露信息。

其他选择加入工作流程类型

在符合此处所述要求的前提下，移动运营商还会接受应用程序和网站之外的选择加入工作流程，例如口头或书面选择加入。合规的选择加入工作流程以及口头或书面脚本需要征求收件人的明确同意，允许接收特定的消息类型。这方面的例子包括客户支持座席使用的语音脚本，用来在记录到服务数据库之前征得同意，或者在宣传单上列出的电话号码。要提供这些选择加入工作流程类型的模型，您可以提供收集数字的选择加入脚本、营销材料或数据库的屏幕截图。如果选择加入的场景不明确或使用场景超过一定数量，则移动运营商可能会对这些使用场景有其他疑问。

不要发送到旧名单

人们经常更换电话号码。两年前已获取联系同意书的电话号码今天可能是其他人在使用。不要在新的消息发送程序中使用旧的电话号码列表；如果这样做，则可能会因为号码不再使用而导致一些消息失败，而另外一些人可能会因为不记得一开始同意过会选择退出。

审核客户列表

如果要发送周期性短信活动，请定期审核您的客户列表。审核您的客户列表可确保仅接收您的消息的客户是有兴趣接收这些消息的人员。

审核您的列表时，向每个选择加入的客户发送提醒他们已订阅的消息，并为他们提供有关取消订阅的信息。提醒消息可能类似于以下示例：

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

保留记录

保留显示每位客户何时请求从您那里接收短信以及您向每位客户发送哪个消息的记录。全球许多国家和地区要求短信发件人以易于检索的方式来维护这些记录。移动运营商还可能会随时向您请求提供此信息。您所必须提供的确切信息因国家或地区而异。有关记录保留要求的更多信息，请查看您的客户所在的每个国家或地区有关商业短信收发的法规。

有时，运营商或监管机构会要求我们提供客户选择接收您的消息的证据。在这种情况下，请与您 Amazon Web Services 支持 联系并提供承运人或代理机构所需的信息清单。如果您无法提供所需信息，则我们可能会暂停您发送更多短信的功能。

提供清晰、诚实、简洁的信息

SMS 是一种独特的媒介。160-的character-per-message 限制意味着您的消息必须简洁。您在其他通信渠道（例如电子邮件）中会使用的技巧可能不适用于短信渠道，甚至在与短信一起使用时可能显得不

诚实或具有欺骗性。如果消息中的内容与最佳实践不一致，收件人可能会忽略您的消息；在最糟糕的情况下，移动运营商可能会将您的消息标识为垃圾消息，以后会屏蔽来自您电话号码的消息。

此部分为创建有效的短信正文提供了一些技巧和观点。

将自己标识为发件人

收件人应该能够立即分辨出是您发布的消息。遵循此最佳实践的发件人会在每条消息的开头添加一个识别名称（“计划名称”）。

请勿执行以下操作：

```
Your account has been accessed from a new device. Reply Y to confirm.
```

试试这个：

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

不要试图让你的消息看起来像一条 person-to-person消息

一些营销人员倾向于让他们的短信看起来像来自个人，从而为他们的短信增添个性化风格。但是，这种技巧可能会让您的消息看起来像是网络钓鱼尝试。

请勿执行以下操作：

```
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.
```

试试这个：

```
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.
```

在涉及到金钱时请谨慎

诈骗者经常会利用人们省钱和获利的渴望。不要提供好得难以置信的优惠。不要利用金钱的诱惑来骗人。不要使用货币符号来指示金钱。

请勿执行以下操作：

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

试试这个：

```
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.
```

仅使用必要的字符

品牌通常倾向于在消息中使用 ™ 或 ® 等商标符号来保护自己的商标。但是，在 160 个字符的短信的标准字符集（称为 GSM 字母）中，不包括这些符号。当您发送某条包含这些字符的消息时，会使用不同的字符编码系统自动发送您的消息，而这一系统的每段消息仅支持 70 个字符。因此，您的消息可能会分为几段。由于您需要为发送的每段消息付费，因此发送整条消息的费用可能会超出您的预期。此外，收件人可能会收到您发来的多条连续消息，而不是一条消息。有关短信字符编码的更多信息，请参阅Amazon SNS 中的 SMS 字符限制。

请勿执行以下操作：

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

试试这个：

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> 前面两个示例几乎相同，但第一个示例包含注册商标符号 (®)，它不是 GSM 字母表的一部分。因此，第一个示例作为两段消息发送，而第二个示例作为一段消息发送。

使用有效且安全的链接

如果您的消息包含链接，请仔细检查链接以确保链接可以正常工作。在公司网络之外的设备上测试您的链接，确保能够正确解析链接。由于 SMS 消息有 160 个字符的限制，因此 URLs 可以将很长时间分成多条消息。您应该使用重定向域名来提供简短的域名 URLs。但是，您不应使用免费的链接缩短服务（如 tinyurl.com 或 bitly.com），因为运营商倾向于过滤掉包含这些域中链接的消息。不过，只要链接指向专供贵公司或机构使用的域，您就可以使用付费的链接缩短服务。

请勿执行以下操作：

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

试试这个：

```
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.
```

限制使用的缩略语数量

SMS 渠道的 160 个字符限制使一些发件人认为，他们需要在消息中大量使用缩写。但是，对许多读者来说，过度使用缩写会显得不专业，并可能导致一些用户将您的消息举报为垃圾消息。您完全可以编写流畅的消息而不过多地使用缩写。

请勿执行以下操作：

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

试试这个：

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

适当地响应

当收件人回复您的消息时，请确保您使用有用的信息进行响应。例如，当客户响应包含关键字“HELP”的其中一个消息时，请向他们发送有关其订阅的程序、您每月将发送的消息数量以及他们可与您取得联系以获取更多信息的方式的信息。HELP 响应可能类似于以下示例：

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

当客户使用关键字“STOP”回复时，让他们了解到他们将不会接收任何消息。STOP 响应可能类似于以下示例：

```
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.
```

基于参与度调整您的发送

您客户的优先级可能随着时间推移而发生变化。如果客户发现您的消息不再有用，则他们可能会选择完全不再使用您的消息，或者甚至将您的消息报告为未经请求的消息。出于这些原因，您必须基于客户参与度调整您的发送活动。

对于与您的消息互动很少的客户，您应调整相应的消息发送频率。例如，如果向参与的客户每周发送消息，您可以为参与度较低的客户创建单独的每月摘要文件。

最后，从您的客户列表中删除完全未参与的客户。此步骤可防止客户对您的消息感到沮丧。这还可为您节省资金并且帮助保护您作为发件人的声誉。

在适当时间发送

仅在正常白天工作时间内发送消息。如果您在晚餐时间或午夜发送消息，则很可能您的客户将从您的列表中取消订阅以避免被打扰。此外，在您的客户无法立即响应短信时进行发送是没有意义的。

如果您将活动或旅程发送给非常多的受众，请仔细检查您的发送号码的吞吐速率。将收件人数量除以您的吞吐速率，确定向所有收件人发送消息需要多长时间。

避免跨通道疲劳

在您的活动中，如果使用多个信道（如电子邮件、SMS 和推送消息），请勿在每个渠道中发送相同消息。当时在多个渠道中同时发送相同消息时，您的客户可能会认为您的发送行为很烦人而不是有用。

使用专用短代码

如果使用短代码，请为每个品牌和每种类型的消息维护单独的短代码。例如，如果您的公司有两个品牌，请为每个品牌使用单独的短代码。同样，如果您发送事务和促销消息，请为每种类型的消息使用单独的短代码。要了解有关请求短代码的更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Requesting short codes for SMS messaging with Amazon End User Messaging SMS。

验证您的目标电话号码

当您通过 Amazon SNS 发送 SMS 消息时，您需要为发送的每段消息付费。您为每段消息支付的价格因收件人所在的国家或地区而异。有关短信定价的信息，请参阅Amazon 全球短信定价。

当 Amazon SNS 接受发送 SMS 消息的请求（调用 SendMessagesAPI 的结果，或者活动或旅程启动的结果）时，您需要支付发送该消息的费用。即使预期的收件人实际上没有收到消息，您也需要支付费用。例如，如果收件人的电话号码已停用，或者您向其发送消息的号码不是有效的手机号码，但仍会向您收取发送消息的费用。

Amazon SNS 接受发送 SMS 消息的有效请求并尝试发送这些消息。因此，您应该验证向其发送消息的电话号码是否为有效的手机号码。您可以使用 Amazon End User Messaging SMS 发送测试消息，以确定电话号码是否有效以及该号码的类型（例如手机、座机或 VoIP）。有关更多信息，请参阅《Amazon End User Messaging SMS User Guide》中的 Send a test message with the SMS simulator。

设计时要考虑冗余

对于任务关键型消息传送程序，我们建议您在多个 Amazon Web Services 区域中配置 Amazon SNS。亚马逊 SNS 有多种版本可供选择。 Amazon Web Services 区域有关已推出 Amazon SNS 的区域的完整列表，请参阅《Amazon Web Services 一般参考》https://docs.amazonaws.cn/general/latest/gr/sns.html。

您用于发送 SMS 的电话号码（包括短代码、长代码、免费电话号码和 10DLC 号码）无法跨 Amazon Web Services 区域进行复制。因此，要在多个区域中使用 Amazon SNS，您必须在要使用 Amazon SNS 的每个区域中申请单独的电话号码。例如，如果您使用短代码向美国的收件人发送短信，则需要在计划使用的每个短代码中申请单独 Amazon Web Services 区域 的短代码。

在某些国家/地区，您还可以使用多种类型的电话号码来增加冗余。例如，在美国，您可以请求短代码、10DLC 号码和免费电话号码。这些电话号码中的每一种都采用不同的途径向收件人发消息。提供多种电话号码类型（相同 Amazon Web Services 区域 或分布在多个电话号码中 Amazon Web Services 区域）可提供额外的冗余层，这有助于提高弹性。

SMS 限额和限制

有关短信限额和限制，请参阅《Amazon End User Messaging SMS User Guide》中的 SMS and MMS limits and restrictions。

管理退出关键词

SMS 收件人可以使用他们的设备通过回复某关键字来选择不接收消息。有关更多信息，请参阅 退出接收 SMS 消息。

CreatePool

使用 `CreatePool` API 操作创建新池并将指定的来源身份关联到该池。有关更多信息，请参阅 Amazon End User Messaging SMS API 参考CreatePool中的。

PutKeyword

使用 `PutKeyword` API 操作创建或更新发起电话号码或池的关键字配置。有关更多信息，请参阅 Amazon End User Messaging SMS API 参考PutKeyword中的。

管理号码设置

要管理您向 Su Amazon pport 请求并分配给帐户的专用短代码和长代码的设置，请参阅使用 Amazon CLI中的更改电话号码的功能Amazon End User Messaging SMS。

Amazon SNS 中的 SMS 字符限制

一条短信最多能包含 140 字节的信息。您在一条短信中可以包含的字符数取决于消息中所包含字符的类型。

如果您的消息仅使用 GSM 03.38 字符集中的字符（也称为 GSM 7 位字母），则它最多能包含 160 个字符。如果您的消息包含 GSM 03.38 字符集以外的任何字符，则它最多可以有 70 个字符。在发送 SMS 消息时，Amazon SNS 会自动确定要使用的最有效编码。

当消息包含的字符数超过最大字符数时，消息将拆分为多个部分。将消息拆分为多个部分时，每个部分都包含有关其前面的消息部分的其他信息。当接收人的设备接收以这种方式分隔的消息部分时，它使用此附加信息来确保所有消息部分都以正确的顺序显示。根据接收人的移动运营商和设备，多条消息可能会显示为单条消息或由单独消息组成的序列。因此，每个消息部分中的字符数减少至 153 个（对于只包含 GSM 03.38 字符的消息）或 67 个（对于包含其他字符的消息）。您可以通过使用短信长度计算器工具来估算消息包含的消息部分数量，其中一些工具是在线提供的。任何消息支持的最大大小为 1,600 个 GSM 字符或 630 个非 GSM 字符。有关吞吐量和消息大小的更多信息，请参阅《Amazon Pinpoint 用户指南》中的 Amazon Pinpoint 中的 SMS 字符数限制。

要查看您发送的每条消息的消息部分的数量，您应首先启用事件流设置。执行此操作时，Amazon SNS 将在消息传递到接收人的移动提供商时生成一个 `_SMS.SUCCESS` 事件。`_SMS.SUCCESS` 事件记录包含名为 `attributes.number_of_message_parts` 的属性。此属性指定消息包含的消息部分的数量。

> **Important**
>
> 当您发送包含多个消息部分的消息时，您需要针对消息中包含的这些数量的消息部分付费。

GSM 03.38 字符集

下表列出了 GSM 03.38 字符集中所存在的所有字符。如果您所发送的消息只包含下表中显示的字符，那么该消息最多可以包含 160 个字符。

GSM 03.38 标准字符												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X 形	Y	Z
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
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						

除上表中所示符号以外，GSM 03.38 字符集还包含其他几个符号。但是，这些字符中的每个字符都会算作两个字符，因为这些字符中还包含一个看不见的转义字符：

- ^
- {
- }
- \
- [
-]
- ~
- |

- €

最后，GSM 03.38 字符集还包含以下非打印字符：

- 空格字符。
- 换行控制，它表示一行文本的结束和另一行文本的开始。
- 回车控制，它会移动到一行文本的开头（通常跟在换行符后面）。
- 转义控制，它会自动添加到前一列表中的字符中。

示例消息

本部分包含几个示例短信。对于每个示例，此部分显示消息的字符总数以及消息部分的数量。

示例 1：只包含 GSM 03.38 字母表中的字符的长消息

以下消息仅包含 GSM 03.38 字母表中的字符。

```
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.
```

上述消息包含 180 个字符，因此必须将其拆分为多个消息部分。将消息拆分为多个消息部分时，每个部分可以包含 153 个 GSM 03.38 字符。因此，此消息作为 2 个消息部分发送。

示例 2：包含多字节字符的消息

以下消息包含多个中文字符，所有这些字符都在 GSM 03.38 字母表之外。

```
##################################################·####1994#7#########
```

上述消息包含 71 个字符。但是，由于消息中的几乎所有字符都在 GSM 03.38 字母表之外，因此它作为两个消息部分发送。每个消息部分最多可包含 67 个字符。

示例 3：包含单个非 GSM 字符的消息

以下消息包含不属于 GSM 03.38 字母表的单个字符。在此示例中，该字符是一个右单引号 (’)，它是与常规撇号 (') 不同的字符。字处理应用程序（如 Microsoft Word）通常会自动用右单引号替换撇号。如果您在 Microsoft Word 中草拟 SMS 消息并将其粘贴到 Amazon SNS 中，则应删除这些特殊字符并用撇号替换它们。

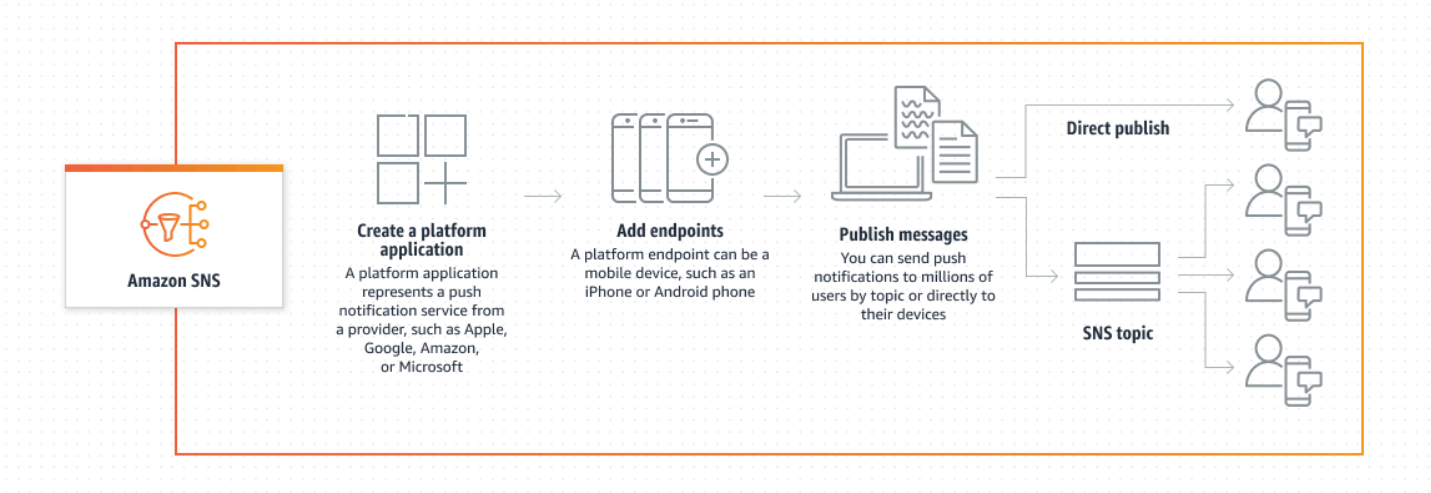
```
John: Your appointment with Dr. Salazar's office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

上述消息包含 130 个字符。但是，由于它包含右单引号字符，而此字符不是 GSM 03.38 字母表的一部分，因此它作为两个消息部分发送。

如果您将此消息中的右单引号字符替换为撇号（它是 GSM 03.38 字母表的一部分），则该消息将作为单个消息部分发送。

使用 Amazon SNS 发送移动推送通知

您可以使用 Amazon SNS 将推送通知消息直接发送至移动设备上的应用程序。发送到移动端点的推送通知消息可在移动应用程序中显示为消息提醒、徽章更新或声音警报。



主题

- Amazon SNS 用户通知的工作原理
- 使用 Amazon SNS 设置推送通知
- 在 Amazon SNS 中设置移动应用程序
- 使用 Amazon SNS 发送移动推送通知
- Amazon SNS 移动应用程序属性
- 移动应用程序的 Amazon SNS 应用程序事件通知
- 移动推送 API 操作
- Amazon SNS 移动推送 API 常见错误
- 使用移动推送通知的 Amazon SNS 生存时间消息属性

- Amazon SNS 移动应用程序支持的区域
- 管理 Amazon SNS 移动推送通知的最佳实践

Amazon SNS 用户通知的工作原理

使用以下受支持的推送通知服务之一将推送通知消息发送到移动设备和桌面：

- Amazon Device Messaging（ADM）
- 适用于 iOS 和 Mac OS X 的 Apple 推送通知服务 (APNs)
- 百度云推送（百度）
- Firebase Cloud Messaging (FCM)
- 适用于 Windows Phone 的 Microsoft 推送通知服务 (MPNS)
- Windows 推送通知服务（WNS）

推送通知服务（例如 APNs 和 FCM）与注册使用其服务的每个应用程序和关联的移动设备保持连接。在应用程序和移动设备注册时，推送通知服务会返回设备令牌。Amazon SNS 使用该设备令牌创建它能够将直接推送通知消息发送到的移动终端节点。为使 Amazon SNS 与不同推送通知服务通信，您需要将推送通知服务凭证提交给用于代表您的 Amazon SNS。有关更多信息，请参阅 使用 Amazon SNS 设置推送通知。

除了发送直接推送通知消息，还可以使用 Amazon SNS 将消息发送到订阅某个主题的移动终端节点。其概念与订阅其他终端节点类型（如 Amazon SQS、HTTP/S、电子邮件和 SMS）相同，如 什么是 Amazon SNS？ 中所述。不同之处在于 Amazon SNS 使用推送通知服务通信，使订阅的移动终端节点接收发送给相应主题的推送通知消息。

使用 Amazon SNS 设置推送通知

1. 为要支持的移动平台获取凭证和设备令牌。
2. 通过 Amazon SNS，使用凭证创建平台应用程序对象 (`PlatformApplicationArn`)。有关更多信息，请参阅 创建 Amazon SNS 平台应用程序。
3. 使用返回的凭证从推送通知服务请求您的移动应用程序和设备的设备令牌。收到的令牌表示您的移动应用程序和设备。
4. 通过 Amazon SNS，使用设备令牌和 `PlatformApplicationArn` 创建平台终端节点对象 (`EndpointArn`)。有关更多信息，请参阅 为移动通知设置 Amazon SNS 平台端点。

5. 使用 `EndpointArn` 向移动设备上的应用发布消息。有关更多信息，请参阅 直接 Amazon SNS 移动设备消息传递 和 Amazon Simple Notification Service API 参考 中的发布 API。

在 Amazon SNS 中设置移动应用程序

本主题介绍如何 Amazon Web Services Management Console 使用中所述的信息在中设置移动应用程序Amazon SNS 用户通知的先决条件。

Amazon SNS 用户通知的先决条件

要开始使用 Amazon SNS 移动推送通知，您需要：

- 用于连接支持的推送通知服务之一的一组凭证：ADM、Baidu、FCM APNs、MPNS 或 WNS。
- 移动应用程序和设备的设备令牌或注册 ID。
- 配置 Amazon SNS 以将推送通知消息发送到移动终端节点。
- 注册并配置移动应用程序来使用支持的推送通知服务之一。

使用推送通知服务注册您的应用程序需要几个步骤。Amazon SNS 需要您提供给推送通知服务的一些信息才能将直接推送通知消息发送到移动终端节点。通常而言，您需要连接推送通知服务所需的凭证、从推送通知服务获得的设备令牌或注册 ID（表示移动设备和移动应用程序），以及已注册推送通知服务的移动应用程序。

凭证的准确格式因移动平台而异，但在所有情况下，这些凭证必须在与平台建立连接时提交。为每个移动应用程序发布一组凭证，并且必须将其用于将消息发送到该应用程序的所有实例。

具体名称根据使用的推送通知服务而不同。例如，当 APNs 用作推送通知服务时，您需要设备令牌。或者，如果使用 FCM，对应于设备令牌的是注册 ID。设备令牌或注册 ID 是由移动设备的操作系统发送到应用程序的字符串。它唯一标识运行在特定移动设备上的移动应用程序的实例，可以视为此应用程序/设备对的唯一标识符。

Amazon SNS 将凭证（以及其他几个设置）存储为平台应用程序资源。设备令牌（以及一些其他设置）以被称为平台端点的对象来表示。每个平台终端节点属于一个特定平台应用程序，可以使用存储在其对应平台应用程序中的凭证与每个平台终端节点进行通信。

下面几节包括每个受支持推送通知服务的先决条件。获得必备信息后，您可以使用 Amazon Web Services Management Console 或 Amazon SNS 移动推送发送推送通知消息。 APIs有关更多信息，请参阅 使用 Amazon SNS 设置推送通知。

创建 Amazon SNS 平台应用程序

要从 Amazon SNS 向移动端点发送通知（无论是直接发送还是通过订阅主题），您必须先创建平台应用程序。向注册应用程序后 Amazon，您需要为应用程序和移动设备创建终端节点。此终端节点允许 Amazon SNS 向设备发送消息。

要创建平台应用程序

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择推送通知。
3. 在 Platform applications（平台应用程序）部分，选择 Create platform application（创建平台应用程序）。
4. 选择您的 Amazon Web Services 区域。有关可以在其中创建移动应用程序的 Amazon 区域的列表，请参阅Amazon SNS 移动应用程序支持的区域。
5. 输入以下应用程序详细信息：
   - 应用程序名称-为您的平台应用程序提供一个名称。名称必须介于 1 到 256 个字符之间，可以包含大写和小写字母、数字、下划线、连字符和句点。
   - 推送通知平台-选择应用程序注册的相应通知服务（例如，Apple 推送通知服务 (APNs)、Firebase 云消息 (FCM)）。
6. 根据您选择的平台，您需要提供特定的凭证：
   - 对于 APNs（Apple 推送通知服务）-在基于令牌的身份验证或基于证书的身份验证之间进行选择。
     - 对于基于令牌的身份验证，请上传 .p8 文件（通过 Keychain Access 生成）。
     - 对于基于证书的身份验证，请上传 .p12 文件（也从 Keychain Access 导出）。
   - 对于 FCM（Firebase 云消息传递），请输入 Firebase 控制台中的服务器密钥。
   - 对于其他平台（例如 ADM 或 GCM），请输入相应的 API 密钥或凭证。
7. 输入必要的详细信息后，选择创建平台应用程序。此操作将该应用程序注册到 Amazon SNS 并创建相应的平台应用程序对象。
8. 创建后，Amazon SNS 会生成并返回一个（`PlatformApplicationArn`亚马逊资源名称）。此 ARN 可唯一标识您的平台应用程序，用于为移动设备创建终端节点。

为移动通知设置 Amazon SNS 平台端点

当应用程序和移动设备注册推送通知服务（例如 APNs 或 Firebase Cloud Messaging）时，推送通知服务会返回设备令牌。Amazon SNS 使用此设备令牌创建平台终端节点，该终端节点充当向设备上的应用程序发送直接推送通知消息的目标。平台终端节点充当桥梁，将 Amazon SNS 发送的消息路由到推送通知服务，然后传送到相应的移动设备。有关更多信息，请参阅Amazon SNS 用户通知的先决条件 和使用 Amazon SNS 设置推送通知。

了解设备令牌和平台端点

设备令牌唯一标识在推送通知服务（例如，Firebase 云消息）中注册的移动设备。 APNs当应用程序向推送通知服务注册时，它会生成特定于该应用程序和设备的设备令牌。Amazon SNS 使用此设备令牌在相应的平台应用程序中创建平台终端节点。

平台终端节点允许 Amazon SNS 通过推送通知服务向设备发送推送通知消息，从而保持您的应用程序与用户设备之间的连接。

创建平台终端节点

要使用 Amazon SNS 将通知推送到应用程序，必须先通过调用创建平台终端节点操作，将该应用程序的设备令牌注册到 Amazon SNS。此操作以参数的形式获取平台应用程序的 Amazon Resource Name (ARN) 以及设备令牌，并返回所创建平台终端节点的 ARN。

`CreatePlatformEndpoint` 操作会执行以下操作：

- 如果平台端点已存在，则不重新创建。向调用方返回现有平台终端节点的 ARN。
- 如果存在具有相同设备令牌但不同设置的平台端点，则不重新创建。向调用方引发异常。
- 如果平台端点不存在，则创建它。向调用方返回新创建的平台终端节点的 ARN。

您不应在每次应用程序启动时立即调用创建平台终端节点操作，因为此方法并不总是提供正常工作的终端节点。例如，在相同设备上卸载并重新安装了应用程序，并且其终端节点已存在但被禁用时，会出现这种情况。成功的注册过程应该完成以下任务：

1. 确保此应用程序/设备组合存在平台终端节点。
2. 确保平台终端节点中的设备令牌是最新的有效设备令牌。
3. 确保平台终端节点已启用并且已准备好使用。

伪代码

以下伪代码介绍了在各种各样的开始条件中创建正常工作的、当前启用的平台终端节点的推荐做法。不论应用程序是否首次注册、此应用程序的平台终端节点是否已存在、平台终端节点是否已启用、是否具有正确的设备令牌等等，此方法均适用。连续多次调用该方法是安全的，因为它不会创建重复的平台终端节点；如果现有平台终端节点已是最新的并且已启用，也不会更改它。

```
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
```

在应用程序希望注册或重新注册自身时，可以随时使用此方法。它还可在向 Amazon SNS 通知设备令牌更改时使用。在这种情况下，只需使用最新的设备令牌值调用操作即可。有关此方法需要说明的几点是：

- 在两种情况下可能会调用创建平台终端节点操作。在刚开始时，应用程序不知道自己的平台终端节点 ARN 时可能会调用该方法；这种情况出现在首次注册期间。在初始 `GetEndpointAttributes` 操作失败并返回“未找到”异常时，也会调用该方法；这种情况发生在应用程序知道其端点 ARN 但该 ARN 已被删除时。
- 调用 `GetEndpointAttributes` 操作来验证平台端点的状态，即使刚刚创建了平台端点。平台终端节点已存在但被禁用时会出现这种情况。在这种情况下，创建平台终端节点操作成功，但不启用平台终端节点，因此您必须在返回成功之前仔细检查平台终端节点的状态。

Amazon SDK 示例

以下代码显示了如何使用提供的 Amazon SNS 客户端来实现之前的伪代码。 Amazon SDKs

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置Amazon SDKs 和凭据文件。

CLI

Amazon CLI

创建平台应用程序端点

以下 `create-platform-endpoint` 示例使用指定令牌为指定平台应用程序创建端点。

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

输出：

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

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

有关更多信息，请参阅 移动推送 API 操作。

故障排除

使用过期设备令牌重复调用创建平台终端节点操作

特别是对于 FCM 端点，您可能会认为最好是存储应用程序发布的第一个设备令牌，然后在每次应用程序启动时，使用该设备令牌调用创建平台端点操作。这似乎正确，因为它会使得应用程序无需管理设备令牌的状态，Amazon SNS 自动将设备令牌更新为其最新值。但是，此解决方案存在多个严重问题：

- Amazon SNS 依靠 FCM 的反馈将失效的设备令牌更新为新的设备令牌。FCM 会将旧设备令牌的相关信息保留一段时间，但并非无限期保留。FCM 忘掉旧设备令牌与新设备令牌之间的联系之

后，Amazon SNS 无法再将存储在平台终端节点中的设备令牌更新为其正确值，而只是改为禁用平台终端节点。

- 平台应用程序将包含与同一个设备令牌对应的多个平台终端节点。
- Amazon SNS 对使用相同设备令牌创建的平台终端节点数量施加了配额。最终，新终端节点的创建将会失败，引发无效参数异常并显示以下错误消息："此终端节点已注册到其他令牌。"

有关管理 FCM 端点的更多信息，请参阅 Amazon SNS 管理 Firebase Cloud Messaging 端点。

重新启用与无效设备令牌关联的平台终端节点

当移动平台（例如 APNs 或 FCM）通知 Amazon SNS 发布请求中使用的设备令牌无效时，Amazon SNS 将禁用与该设备令牌关联的平台终端节点。然后，Amazon SNS 将拒绝向该设备令牌进行的后续发布操作。虽然您可能会认为最好的方法是简单地重新启用平台终端节点并保持发布，但大多数情况下这样做不会有成效：发布的消息不会被传输，平台终端节点随后很快会再次被禁用。

这是因为与平台终端节点关联的设备令牌已真正无效。由于它不再对应于任何已安装的应用程序，因此以它为目标的传输操作不会成功。下次发布到该平台终端节点时，移动平台将再次通知 Amazon SNS 设备令牌无效，Amazon SNS 将再次禁用该平台终端节点。

要重新启用已禁用的平台终端节点，该终端节点需要关联到有效的设备令牌（使用设置终端节点属性操作调用），然后再启用。只有这样，以该平台终端节点为目标的传输操作才会成功。要在不更新设备令牌的情况下重新启用平台终端节点，这种方法只有当与终端节点关联的设备令牌曾经无效但重新变得有效时才有起作用。例如，在相同移动设备上卸载、然后重新安装了某个应用程序并收到了相同的设备令牌时，会出现这种情况。上述方法会执行此操作，确保只有在验证与某个平台终端节点关联的设备令牌是最新可用的设备令牌时，才重新启用该平台终端节点。

将设备令牌与 Amazon SNS 集成以发送移动通知

当您首次使用通知服务（例如 Apple 推送通知服务 (APNs) 和 Firebase Cloud Messaging (FCM)）注册应用程序和移动设备时，该服务会返回设备令牌或注册 IDs 信息。这些令牌/ IDs 将添加到 Amazon SNS 中，以便使用 API 为应用程序和设备创建终端节点。`PlatformApplicationArn`创建终端节点后，将返回一个 `EndpointArn`，Amazon SNS 使用该终端节点将通知定向到正确的应用程序/设备。

您可以通过以下方式向 Amazon SNS 添加设备令牌或注册 IDs ：

- 通过手动添加单个令牌 Amazon Web Services Management Console
- 使用 `CreatePlatformEndpoint` API 上载多个令牌
- 为 future 设备注册代币

手动添加设备令牌或注册 ID

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择 “推送通知”。
3. 在平台应用程序部分，选择您的应用程序，然后选择编辑。如果您尚未创建平台应用程序，请立即按照创建 Amazon SNS 平台应用程序指南进行创建。
4. 选择 Create Endpoint（创建端点）。
5. 在终端节点令牌框中，根据您使用的通知服务（例如，FCM 注册 ID），输入令牌或注册 ID。
6. （可选）在 “用户数据” 字段中输入其他数据。此数据必须采用 UTF-8 编码且小于 2 KB。
7. 选择 Create Endpoint（创建端点）。

创建终端节点后，您可以直接向移动设备或订阅 Amazon SNS 主题的移动设备发送消息。

使用 **`CreatePlatformEndpoint`** API 上传多个令牌

以下步骤说明如何使用提供的示例 Java 应用程序（`bulkupload`软件包）将多个令牌（设备令牌或注册 IDs）上传 Amazon 到 Amazon SNS。您可以使用本示例应用帮助您开始上传现有令牌。

> **Note**
>
> 下面的步骤使用 Eclipse Java IDE。这些步骤假设您已安装 适用于 Java 的 Amazon SDK 并且拥有自己的 Amazon 安全证书 Amazon Web Services 账户。有关更多信息，请参阅 适用于 Java 的 Amazon SDK。有关证书的更多信息，请参阅 IAM 用户指南中的Amazon 安全证书。

1. 下载并解压缩 snsmobilepush.zip 文件。
2. 在 Eclipse 中创建一个新的 Java 项目，然后将该`SNSSamples`文件夹导入到该项目中。
3. 下载 OpenCSV 库并将其添加到构建路径中。
4. 在`BulkUpload.properties`文件中，指定以下内容：
   - 您的`ApplicationArn`（平台应用程序 ARN）。
   - 包含令牌的 CSV 文件的绝对路径。
   - 记录成功和失败令牌的文件名。例如，`goodTokens.csv` 和 `badTokens.csv`。
   - （可选）分隔符、引号字符和要使用的线程数的配置。

完成后的 `BulkUpload.properties` 与下文类似：

```
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. 运行 BatchCreatePlatformEndpointSample.java 应用程序将令牌上传到亚马逊 SNS。成功上传的令牌将被登录`goodTokens.csv`，而格式错误的令牌将被登录。`badTokens.csv`

从设备上注册令牌以备将来安装应用程序

对于此过程，您有两种选择：

使用亚马逊 Cognito 服务

您的移动应用程序可以使用临时安全证书来创建终端节点。建议使用 Amazon Cognito 来生成临时证书。有关更多信息，请参阅 Amazon Cognito 开发者指南

要跟踪应用程序注册，请在创建新的 ARNs 终端节点时使用 Amazon SNS 事件接收通知。

或者，您可以使用 `ListEndpointByPlatformApplication`API 检索已注册的终端节点列表。

使用代理服务器

如果您的应用程序基础架构已支持在安装时注册设备，则可以将服务器用作代理。它将通过 API 将设备令牌转发给亚马逊 SNS。`CreatePlatformEndpoint`

Amazon SNS 创建的终端节点 ARN 将被返回，并可由您的服务器存储以备将来发布消息。

Amazon SNS Apple 推送通知身份验证方法

您可以通过提供将您识别为应用程序开发人员的信息，授权 Amazon SNS 将推送通知发送到您的 iOS 或 macOS 应用程序。要进行身份验证，请在创建平台应用程序时提供密钥或者证书，您可以通过 Apple 开发人员账户获得这两项。

令牌签名密钥

Amazon SNS 用于签署 Apple 推送通知服务 (APNs) 身份验证令牌的私有签名密钥。

如果您提供签名密钥，Amazon SNS 将使用令牌对您发送 APNs的每条推送通知进行身份验证。使用您的签名密钥，您可以向 APNs 生产环境和沙盒环境发送推送通知。

您的签名密钥不会过期，您可以将相同的签名密钥用于多个应用程序。有关更多信息，请参阅 Apple 网站 “开发者账户帮助” 部分中的 APNs 使用身份验证令牌进行通信。

证书

在您发送推送通知 APNs 时，Amazon SNS 使用它进行身份验证的 TLS 证书。您可以从 Apple 开发人员账户获取该证书。

证书将在一年后过期。出现这种情况时，您必须创建新证书并将其提供给 Amazon SNS。有关更多信息，请参阅 Apple 开发者网站 APNs上的与建立基于证书的连接。

使用管理控制台 Amazon 管理 APNs 设置

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择推送通知。
3. 在 “平台应用程序” 部分，选择要编辑其 APNs 设置的应用程序，然后选择 “编辑”。如果您尚未创建平台应用程序，请立即按照创建 Amazon SNS 平台应用程序指南进行创建。
4. 选择 “编辑” 以修改平台应用程序的设置。
5. 在 “身份验证类型” 部分，选择以下选项之一：
   - 基于令牌的身份验证（建议用于现代 APNs 集成）
   - 基于证书的身份验证（较旧的方法）
6. 根据身份验证类型配置您的证书：
   - 对于基于令牌的身份验证：
     - 上传 .p8 文件，这是您从 Apple 开发者帐户下载的身份验证令牌签名密钥。
     - 输入你在 Apple 开发者账户中找到的签名密钥 ID。导航到 “证书” IDs、“配置文件”、“密钥”，然后选择要使用的密钥。
     - 提供您的 Apple 开发者账户中的团队标识符。你可以在 “会员资格” 页面上找到这个。
     - 输入分配给您的应用程序的捆绑包标识符。您可以在 “证书” IDs 和 “配置文件”、“应用程序” 下找到此信息 IDs。

- 对于基于证书的身份验证：
  - 上传您的 TLS 证书的 .p12 文件。从 Apple 开发者账户下载证书后，可以从 macOS 上的 Keychain Access 中导出此文件。
  - 如果您为.p12 证书分配了密码，请在此处输入。

7. 输入必要的凭据后，选择保存更改以更新设置。

Amazon SNS 与 Firebase Cloud Messaging 身份验证设置集成

本主题介绍如何从 Google 获取用于该 API 所需的 FCM API (HTTP v1) 凭据， Amazon CLI 以及。Amazon Amazon Web Services Management Console

> **Important**
>
> 2024 年 3 月 26 日 – Amazon SNS 支持适用于 Apple 设备和 Webpush 目的地的 FCM HTTP v1 API。我们建议您在 2024 年 6 月 1 日或之前将现有的移动推送应用程序迁移到最新的 FCM HTTP v1 API，以避免应用程序中断。
> 2024 年 1 月 18 日 – Amazon SNS 推出了对 FCM HTTP v1 API 的支持，用于向 Android 设备发送移动推送通知。
> 2023 年 6 月 20 日 - Google 弃用了其 Firebase Cloud Messaging（FCM）旧版 HTTP API。Amazon SNS 现在支持使用 FCM HTTP v1 API 向所有设备类型传送。我们建议您在 2024 年 6 月 1 日或之前将现有的移动推送应用程序迁移到最新的 FCM HTTP v1 API，以避免中断。

您可以通过提供将您识别为应用程序开发人员的信息，授权 Amazon SNS 将推送通知发送到您的应用程序。要进行身份验证，请在创建平台应用程序时提供 API 密钥或令牌。您可以从 Firebase 应用程序控制台获取以下信息：

API 密钥

API 密钥是调用 Firebase 的旧版 API 时使用的凭证。谷歌 APIs 将于2024年6月20日移除FCM Legacy。如果您当前使用 API 密钥作为平台凭证，则可以通过选择令牌作为选项并上传您的 Firebase 应用程序的关联 JSON 文件来更新平台凭证。

令牌

在调用 HTTP v1 API 时，会使用有效期很短的访问令牌。这是 Firebase 的建议用于发送推送通知的 API。为了生成访问令牌，Firebase 以私有密钥文件（也称为 service.json 文件）的形式为开发人员提供了一组凭证。

先决条件

您必须先获取 FCM service.json 凭证，然后才能开始在 Amazon SNS 中管理 FCM 设置。要获取您的 service.json 凭据，请参阅谷歌 Firebase 文档中的从旧版 FCM 迁移 APIs 到 HTTP v1。

使用 CLI 管理 FCM 设置

您可以使用 Amazon API 创建 FCM 推送通知。一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅《Amazon Web Services 一般参考 Guide》中的 Amazon Simple Notification Service endpoints and quotas。

创建 FCM 推送通知以及亚马逊 SNS 主题Amazon (API)

使用密钥凭证时，`PlatformCredential` 为 `API key`。使用令牌凭证时，`PlatformCredential` 为一个采用 JSON 格式的私有密钥文件：

- `CreatePlatformApplication`

检索现有亚马逊 SNS 主题 (API) 的 FCM 凭证类型Amazon

检索凭证类型 `"AuthenticationMethod": "Token"` 或 `"AuthenticationMethod": "Key"`：

- GetPlatformApplicationAttributes

为现有 Amazon SNS 主题设置 FCM 属性（Amazon API）

设置 FCM 属性：

- SetPlatformApplicationAttributes

使用控制台管理 FCM 设置

您可以使用 Amazon Command Line Interface (CLI) 创建 FCM 推送通知。一个 Amazon 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service endpoints and quotas。

与 Amazon SNS 主题一起创建 FCM 推送通知（Amazon CLI）

使用密钥凭证时，`PlatformCredential` 为 `API key`。使用令牌凭证时，`PlatformCredential` 为一个采用 JSON 格式的私有密钥文件。使用 Amazon CLI 时，文件必须为字符串格式，并且必须忽略特殊字符。为了正确格式化文件，Amazon SNS 建议使用以下命令：`SERVICE_JSON=`jq @json <<< cat service.json``：

- create-platform-application

检索现有 Amazon SNS 主题的 FCM 凭证类型（Amazon CLI）

检索凭证类型 `"AuthenticationMethod": "Token"` 或 `"AuthenticationMethod": "Key"`：

- get-platform-application-attributes

为现有 Amazon SNS 主题设置 FCM 属性（Amazon CLI）

设置 FCM 属性：

- set-platform-application-attributes

管理 FCM 设置（控制台）

使用以下步骤在亚马逊 SNS 中输入和管理您的 Firebase 云消息 (FCM) 凭证。

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择推送通知。
3. 在平台应用程序部分，选择要编辑其凭据的 FCM 平台应用程序，然后选择编辑。
4. 在 Firebase 云端消息传递凭据部分，选择以下选项之一：
    - 基于令牌的身份验证（推荐方法）-上传您从 Firebase 控制台下载的私钥文件 (JSON)。此文件包含为 FCM 通知生成短期访问令牌所需的凭据。要获取此文件，请执行以下操作：

1. 前往您的 Firebase 应用程序控制台。
2. 在 “项目设置” 中，选择 “云消息”。
3. 下载私钥 JSON 文件（用于基于令牌的身份验证方法）。

- API 密钥身份验证 — 如果您更喜欢使用较旧的 API 密钥身份验证方法，请在提供的字段中输入 Google API 密钥。要获取此文件，请执行以下操作：
  1. 前往您的 Firebase 应用程序控制台。
  2. 在 “项目设置” 中，选择 “云消息”。
  3. 复制用于发送通知的服务器密钥（API 密钥）。

5. 完成后，选择保存更改。

相关主题

- 在 Amazon SNS 中使用 Google Firebase Cloud Messaging v1 有效载荷

Amazon SNS 管理 Firebase Cloud Messaging 端点

管理和维护设备令牌

您可以按照以下步骤确保移动应用程序的推送通知的可送达性：

1. 将所有设备令牌、相应的 Amazon SNS 终端节点 ARNs和时间戳存储在您的应用程序服务器上。
2. 移除所有陈旧的令牌并删除相应的 Amazon SNS ARNs 终端节点。

应用程序首次启动后，您将收到该设备的设备令牌（也称为注册令牌）。此设备令牌由设备的操作系统创建，并与您的 FCM 应用程序相关联。收到此设备令牌后，您可以将其作为平台端点注册到 Amazon SNS。我们建议您存储设备令牌、Amazon SNS 平台端点 ARN 和时间戳，方法是将它们保存到您的应用程序服务器或其他永久存储中。要设置 FCM 应用以检索和存储设备令牌，请参阅 Google 的 Firebase 文档中的检索和存储注册令牌。

维护 up-to-date代币很重要。如果满足以下条件，用户的设备令牌可以更改：

1. 移动应用程序将在新设备上恢复。
2. 用户卸载或更新应用程序。
3. 用户清除应用程序数据。

当您的设备令牌发生变化时，我们建议您使用新令牌更新相应的 Amazon SNS 端点。这样，Amazon SNS 就可以继续与注册的设备进行通信。您可以通过在移动应用程序中实现以下伪代码来做到这一点。它描述了创建和维护已启用的平台端点的推荐做法。这种方法可以在每次移动应用程序启动时执行，也可以在后台作为计划任务执行。

伪代码

使用以下 FCM 伪代码管理和维护设备令牌。

```
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
```

要详细了解令牌更新要求，请参阅 Google 的 Firebase 文档中的定期更新令牌。

检测令牌无效

当向带有无效设备令牌的 FCM v1 端点发送消息时，Amazon SNS 将收到以下例外情况之一：

- `UNREGISTERED`（HTTP 404）– 当 Amazon SNS 收到此异常时，您将收到一个传输失败事件，该事件的 `FailureType` 为 `InvalidPlatformToken`，且 `FailureMessage` 为与端点关联的平台令牌无效。当传送失败时，Amazon SNS 将禁用您的平台端点，但有此例外情况。
- `INVALID_ARGUMENT`（HTTP 400）– 当 Amazon SNS 收到此异常时，这意味着设备令牌或消息有效载荷无效。有关更多信息，请参阅ErrorCode谷歌的 Firebase 文档。

由于在这两种情况下都可以返回 `INVALID_ARGUMENT`，因此 Amazon SNS 将返回 `InvalidNotification` `FailureType` 以及通知正文无效 `FailureMessage`。当您收到此错误时，请验证您的有效载荷是否正确。如果正确，请验证设备令牌是否正确 up-to-date。当传送失败时，Amazon SNS 不会禁用您的平台端点，但有此例外情况。

您会遇到 `InvalidPlatformToken` 传送失败事件的另一种情况是，注册的设备令牌不属于尝试发送该消息的应用程序。在这种情况下，Google 将返回 SENDER_ID_MISMATCH 错误。当传送失败时，Amazon SNS 将禁用您的平台端点，但有此例外情况。

当您为应用程序设置交付状态日志 CloudWatch时，可以查看从 FCM v1 API 收到的所有观察到的错误代码。

要接收应用程序的传送事件，请参阅可用应用程序事件。

移除过时令牌

一旦向端点设备传送消息开始失败，令牌就会被视为过时。Amazon SNS 将这些过时令牌设置为您的平台应用程序的禁用端点。当您向已禁用的端点发布内容时，Amazon SNS 将返回一个 `EventDeliveryFailure` 事件，该事件的 `FailureType` 为 `EndpointDisabled`，`FailureMessage` 为端点处于禁用状态。要接收应用程序的传送事件，请参阅可用应用程序事件。

当您收到来自 Amazon SNS 的此错误时，您需要移除或更新平台应用程序中的过时令牌。

使用 Amazon SNS 发送移动推送通知

本部分描述如何发送移动推送通知消息。

向主题发布

您还可以使用 Amazon SNS 向订阅了某一主题的移动终端节点发送消息。其概念与订阅其他终端节点类型（如 Amazon SQS、HTTP/S、电子邮件和 SMS）相同，如 什么是 Amazon SNS？ 中所述。不同之处在于，Amazon SNS 通过 Apple Push Notification Service (APNS) 和 Google Firebase Cloud Messaging (FCM) 等通知服务进行通信。通过通知服务通信，订阅的移动端点可以接收发送给相应主题的通知。

直接 Amazon SNS 移动设备消息传递

您可以将 Amazon SNS 推送通知消息直接发送到代表移动设备上的应用程序的端点。

发送直送消息

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择 Push notifications（推送通知）。
3. 例如，在移动推送通知页面的平台应用程序部分，选择应用程序的名称***MyApp***。
4. 在该***MyApp***页面的终端节点部分，选择终端节点，然后选择发布消息。
5. 在 Publish message to endpoint (向终端节点发布消息) 页面上，输入将显示在移动设备上的应用程序中的消息，然后选择发布消息。

   Amazon SNS 将通知消息发送到平台通知服务，而平台通知服务会将该消息发送到应用程序。

使用特定于平台的有效载荷发布 Amazon SNS 通知

您可以使用 Amazon Web Services Management Console 或 Amazon SNS APIs 向移动设备发送包含特定平台负载的自定义消息。有关使用 Amazon SNS 的信息 APIs，请参阅移动推送 API 操作和中的`SNSMobilePush.java`文件。`snsmobilepush.zip`

发送 JSON 格式化的消息

在发送平台特定的负载时，数据必须格式化为 JSON 键-值对字符串（用引号进行转义）。

下面的示例显示 FCM 平台的一条自定义消息。

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

发送平台特定的消息

除了以键-值对形式发送自定义数据之外，您还可以发送平台特定的键-值对。

以下示例显示如何在 FCM `data` 参数中的自定义数据键-值对之后加入 FCM 参数 `time_to_live` 和 `collapse_key`。

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

有关 Amazon SNS 中支持的每个推送通知服务所支持的键-值对列表，请参阅以下内容：

> **Important**
>
> Amazon SNS 现在支持 Firebase Cloud Messaging（FCM）HTTP v1 API，用于向 Android 设备发送移动推送通知。
> 2024 年 3 月 26 日 – Amazon SNS 支持适用于 Apple 设备和 Webpush 目的地的 FCM HTTP v1 API。我们建议您在 2024 年 6 月 1 日或之前将现有的移动推送应用程序迁移到最新的 FCM HTTP v1 API，以避免应用程序中断。

- APNs 文档中的@@ 有效载荷密钥参考
- FCM 文档中的 Firebase 云消息收发 HTTP 协议
- ADM 文档中的发送消息

在多个平台上向应用程序发送消息

要向安装在多个平台（例如 FCM 和）设备上的应用程序发送消息 APNs，您必须先在移动终端节点上订阅 Amazon SNS 中的主题，然后将消息发布到该主题。

以下示例显示了向 FCM 和 ADM 上 APNs订阅的移动终端节点发送的消息：

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

将消息 APNs 作为警报或后台通知发送到

Amazon SNS 可以向广告 APNs 发送消息`alert`或`background`通知（有关更多信息，请参阅 APNs 文档中的将后台更新推送到您的应用程序）。

- `alert` APNs 通知通过显示警报消息、播放声音或在应用程序的图标上添加徽章来通知用户。
- `background` APNs 通知会被唤醒或指示您的应用程序根据通知的内容采取行动，而不通知用户。

指定自定义 APNs 标题值

我们建议使用 Amazon SNS `Publish` API 操作为`AWS.SNS.MOBILE.APNS.PUSH_TYPE`保留消息属性指定自定义值 Amazon SDKs，或。 Amazon CLI以下 CLI 示例为指定的主题将 `content-available` 设置为 1 并将 `apns-push-type` 设置为 `background`。

```
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**
>
> 确保 JSON 结构有效。在每个键/值对（最后一个除外）后面添加一个逗号。

根据有效载荷 APNs 推断推送类型标头

如果您未设置`apns-push-type` APNs 标头，Amazon SNS 会`background`根据您的 JSON 格式的负载 APNs 配置`aps`字典中的`content-available`密钥将标头设置为`alert`或依据。

> **Note**
>
> Amazon SNS 只能推断 `background` 或 `alert` 标头，尽管 `apns-push-type` 标头可以设置为其他值。

- `apns-push-type` 设置为 `alert`
  - 如果 `aps` 字典包含设置为 1 的 `content-available` 和一个或多个触发用户交互的键。

- 如果 aps 字典包含设置为 0 的 content-available 或如果 content-available 密钥不存在。
- 如果 content-available 键的值不是整数或布尔值。

- apns-push-type 设置为 background
  - 如果 aps 字典仅包含设置为 1 的 content-available 且不包含触发用户交互的其他键。

> **Important**
>
> 如果 Amazon SNS 将原始配置对象 APNs 作为仅限后台的通知发送，则必须在字典中包含content-available设置为。1 aps尽管您可以包含自定义键，但 aps 字典不得包含触发用户交互的任何键（例如，警报、徽章或声音）。

下面是一个示例原始配置对象。

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

在本示例中，Amazon SNS 将消息的apns-push-type APNs 标头设置为。background当 Amazon SNS 检测到 apn 字典包含设置为 1 的 content-available 键—并且不包含任何其他可触发用户交互的键时—它将标题设置为 background。

在 Amazon SNS 中使用 Google Firebase Cloud Messaging v1 有效载荷

Amazon SNS 支持使用 FCM HTTP v1 API 向 Android、iOS 和 Webpush 目标发送通知。本主题提供了使用 CLI 或 Amazon SNS API 发布移动推送通知时的有效载荷结构示例。

发送 FCM 通知时，您可以在有效载荷中包含以下消息类型：

- 数据消息 - 数据消息由您的客户端应用程序处理，并包含自定义键值对。在构造数据消息时，必须包含以 JSON 对象作为值的 data 键，然后输入您的自定义键值对。
- 通知消息或显示消息 - 通知消息包含由 FCM SDK 处理的一组预定义键。这些键因您要传送到的设备类型而异。有关特定于平台的通知键的更多信息，请参阅以下内容：
  - Android 通知键
  - APNS 通知键
  - Webpush 通知键

有关 FCM 消息类型的更多信息，请参阅 Google 的 Firebase 文档中的消息类型。

使用 FCM v1 有效载荷结构发送消息

如果您是首次创建 FCM 应用程序，或者希望利用 FCM v1 的功能，则可以选择发送 FCM v1 格式的有效载荷。为此，必须包含顶层键 `fcmV1Message`。有关构建 FCM v1 有效负载的更多信息，请参阅 Google 的 Firebase 文档中的从旧版 FCM 迁移 APIs 到 HTTP v1 和跨平台自定义消息。

发送到 Amazon SNS 的 FCM v1 有效载荷示例：

> **Note**
>
> 使用 Amazon SNS 发布通知时，以下示例中使用的 `GCM` 键值必须编码为字符串。

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

发送 JSON 有效载荷时，请务必在请求中包含该 `message-structure` 属性，并将其设置为 `json`。

CLI 示例：

```
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
```

有关发送 FCM v1 格式的有效载荷的更多信息，请参阅 Google 的 Firebase 文档中的以下内容：

- 从旧版 FCM 迁移 APIs 到 HTTP v1
- 关于 FCM 消息
- REST 资源：projects.messages

使用旧版有效载荷结构向 FCM v1 API 发送消息

迁移到 FCM v1 时，您无需更改用于旧版凭证的有效载荷结构。Amazon SNS 会将您的有效载荷转换为新的 FCM v1 有效载荷结构，然后发送给 Google。

输入消息有效载荷格式：

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

发送给 Google 的消息：

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

潜在风险

- 旧版到 v1 的映射不支持 Apple 推送通知服务（APNS）`headers` 或 `fcm_options` 键。如果希望使用这些字段，请发送 FCM v1 有效载荷。
- 在某些情况下，FCM v1 要求消息标头才能向您的 APNs 设备发送静默通知。如果您当前正在向 APNs 设备发送静默通知，则它们不适用于传统方法。相反，我们建议使用 FCM v1 有效载荷以避免意外问题。要查找 APNs 标题列表及其用途，请参阅《Apple 开发者指南》 APNs中的 “与之通信”。
- 如果您在发送通知时使用的是 TTL Amazon SNS 属性，则它只会在 `android` 字段中进行更新。如果您想设置 TTL APNS 属性，请使用 FCM v1 有效载荷。
- `android`、`apns` 和 `webpush` 键将被映射并用所提供的所有相关键填充。例如，如果您提供 `title`（这是所有三个平台之间共享的键），FCM v1 映射将使用您提供的标题填充所有三个平台。
- 一些平台间的共享键需要不同的值类型。例如，传递给 `apns` 的 `badge` 键需要一个整数值，而传递给 `webpush` 的 `badge` 键需要一个字符串值。如果您提供了 `badge` 键，FCM v1 映射将仅填充您为其提供了有效值的键。

FCM 传输失败事件

下表提供了 Amazon SNS 失败类型，该类型与从 Google 收到的 FCM v1 通知请求的错误/状态代码相对应。当您为应用程序设置交付状态日志 CloudWatch 时，可以查看从 FCM v1 API 收到的所有观察到的错误代码。

FCM 错误/状态代码	Amazon SNS 失败类型	失败消息	原因和缓解措施
`UNREGISTERED`	`InvalidPlatformToken`	与端点关联的平台令牌无效。	连接到您的端点的设备令牌已过时或无效。Amazon SNS 禁

FCM 错误/状态代码	Amazon SNS 失败类型	失败消息	原因和缓解措施
			用了您的端点。将 Amazon SNS 端点更新为最新的设备令牌。
INVALID_ARGUMENT	InvalidNotification	通知正文无效。	设备令牌或消息有效载荷可能无效。验证您的消息有效载荷是否有效。如果消息有效载荷有效，请将 Amazon SNS 端点更新为最新的设备令牌。
SENDER_ID_MISMATCH	InvalidPlatformToken	与端点关联的平台令牌无效。	与设备令牌关联的平台应用程序无权向设备令牌发送消息。确认您在 Amazon SNS 平台应用程序中使用了正确的 FCM 凭证。
UNAVAILABLE	DependencyUnavailable	依赖关系不可用。	FCM 无法实时处理请求。Amazon SNS 执行的所有重试都失败。您可以将这些消息存储在死信队列（DLQ）中，以后再重新发送它们。

FCM 错误/状态代码	Amazon SNS 失败类型	失败消息	原因和缓解措施
INTERNAL	UnexpectedFailure	意外失败；请联系 Amazon。失败短语 [内部错误]。	FCM 服务器在处理您的请求时遇到错误。Amazon SNS 执行的所有重试都失败。您可以将这些消息存储在死信队列（DLQ）中，以后再重新发送它们。
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	平台应用程序凭证无效。	无法发送针对 iOS 设备或 Webpush 设备的消息。验证您的开发和生产凭证是否有效。
QUOTA_EXCEEDED	Throttled	请求受 [gcm] 限制。	已超过消息速率配额、设备消息速率配额或主题消息速率配额。有关如何解决此问题的信息，请参阅 ErrorCodeGoogle 的 Firebase 文档中的。
PERMISSION_DENIED	InvalidNotification	通知正文无效。	如果出现 PERMISSION_DENIED 异常，则调用方（您的 FCM 应用程序）无权在有效载荷中执行指定操作。导航到您的 FCM 控制台，并验证您的凭证是否启用了所需的 API 操作。

Amazon SNS 移动应用程序属性

Amazon Simple Notification Service (Amazon SNS) 支持记录推送通知消息的传输状态。配置应用程序属性后，从 Amazon SNS 发送到移动终端节点的消息的 CloudWatch 日志条目将发送到日志。记录消息传输状态有助于提供更好的业务洞察力，例如以下方面：

- 了解推送通知消息是否已从 Amazon SNS 传输到推送通知服务。
- 识别从推送通知服务发送到 Amazon SNS 的响应。
- 确定消息停留时间（发布时间戳与将消息转交给推送通知服务之间的时间差）。

要为消息传送状态配置应用程序属性，您可以使用 Amazon 软件开发套件 (SDKs) 或查询 API。
Amazon Web Services Management Console

使用配置邮件传送状态属性 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，指向移动，并选择推送通知。
3. 从平台应用程序部分，选择包含要接收 CloudWatch 日志的终端节点的应用程序。
4. 选择 Application Actions (应用程序操作)，然后选择 Delivery Status (传输状态)。
5. 在传输状态对话框中，选择创建 IAM 角色。

   随后您将被重定向至 IAM 控制台。
6. 选择 “允许”，授予 Amazon SNS 写入权限，让您可以代表您使用 CloudWatch 日志。
7. 现在，返回 “传送状态” 对话框中，在 “成功样本百分比 (0-100)” 字段中输入一个数字，表示要接收 CloudWatch 日志的成功发送邮件的百分比。

   > **Note**
   >
   > 为消息传送状态配置应用程序属性后，所有失败的消息传送都会生成 CloudWatch 日志。

8. 最后，选择保存配置。现在，您将能够查看和解析包含消息传送状态的 CloudWatch 日志。有关使用的更多信息 CloudWatch，请参阅CloudWatch文档。

Amazon SNS 消息传输状态日志示例 CloudWatch

为应用程序终端节点配置消息传送状态属性后，将生成 CloudWatch 日志。示例日志采用 JSON 格式，如下所示：

SUCCESS

```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
    "token": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\":[{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```

FAILURE

```
{
  "status": "FAILURE",
  "notification": {
    "timestamp": "2015-01-26 23:29:35.678",
    "messageId": "c3ad79b0-8996-550a-8bfa-24f05989898f"
  },
  "delivery": {
    "statusCode": 8,
    "dwellTimeMs": 1451,
    "token": "examp1e29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken,
 id=1, cause=null)",
```

```
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/
APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

有关推送通知服务响应代码的列表，请参阅平台响应代码。

使用配置消息传送状态属性 Amazon SDKs

Amazon SDKs提供了多种语言版本，用于 APIs 在 Amazon SNS 中使用消息传送状态属性。

下面的 Java 示例显示了如何使用 `SetPlatformApplicationAttributes` API 为推送通知消息的消息传输状态配置应用程序属性。您可以对消息传输状态使用以下属性：`SuccessFeedbackRoleArn`、`FailureFeedbackRoleArn` 和 `SuccessFeedbackSampleRate`。`SuccessFeedbackRoleArn`和`FailureFeedbackRoleArn`属性用于向 Amazon SNS 授予代表您使用 CloudWatch 日志的写入权限。`SuccessFeedbackSampleRate` 属性用于指定成功传输消息的采样率百分比 (0-100)。配置该`FailureFeedbackRoleArn`属性后，所有失败的消息传送都会生成 CloudWatch 日志。

```
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);
```

有关适用于 Java 的开发工具包的更多信息，请参阅 适用于 Java 的 Amazon SDK入门指南。

平台响应代码

下面是推送通知服务响应代码链接列表：

推送通知服务	响应代码
Amazon Device Messaging（ADM）	请参阅 ADM 文档中的响应格式。

推送通知服务	响应代码
苹果推送通知服务 (APNs)	请参阅《本地和远程通知编程指南》 APNs中的“与之通信” APNs 中的 HTTP/2 响应。
Firebase Cloud Messaging (FCM)	请参阅 Firebase Cloud Messaging 文档中的下游消息错误响应代码
适用于 Windows Phone 的 Microsoft 推送通知服务 (MPNS)	请参阅 Windows 8 开发文档中的 Windows Phone 8 的推送通知服务响应代码。
Windows 推送通知服务 (WNS)	请参阅 Windows 8 开发文档中推送通知服务请求和响应标头（Windows 运行时应用程序）中的“响应代码”。

移动应用程序的 Amazon SNS 应用程序事件通知

Amazon SNS 提供在发生特定应用程序事件时触发通知的支持。然后，您可以对该事件采取一些编程操作。您的应用程序必须支持推送通知服务，例如 Apple 推送通知服务 (APNs)、Firebase 云消息 (FCM) 和 Windows 推送通知服务 (WNS)。您可以使用 Amazon SNS 控制台设置应用程序事件通知 Amazon CLI，或者。 Amazon SDKs

可用应用程序事件

应用程序事件通知跟踪各个平台端点何时创建、删除、更新以及出现传输故障。以下是应用程序事件的属性名称。

属性名称	通知触发器
`EventEndpointCreated`	向应用程序添加新的平台端点。
`EventEndpointDeleted`	删除与应用程序关联的任何平台端点。
`EventEndpointUpdated`	与应用程序关联的平台端点的任何属性发生更改。

属性名称	通知触发器
EventDeliveryFailure	向与应用程序关联的任何平台端点的传输操作发生永久性故障。 **Note** 要跟踪平台应用程序端的传输故障，需要为应用程序订阅消息传输状态事件。有关更多信息，请参阅使用用于消息传输状态的 Amazon SNS 应用程序属性。

您可以将任何属性与应用程序关联，然后应用程序就可以接收这些事件通知。

发送移动推送通知

要发送应用程序事件通知，您需要为每种事件类型指定用于接收通知的主题。Amazon SNS 发送通知时，主题可以将它们路由至将采取编程操作的终端节点。

> **Important**
>
> 高容量应用程序将创建大量的应用程序事件通知（例如，数万条），这会“淹没”供人们使用的端点，例如电子邮件、电话号码和移动应用程序。在向主题发送应用程序事件通知时，需要考虑以下指导原则：
>
> - 每个接收通知的主题都应仅包含对编程终端节点（例如 HTTP 或 HTTPS 终端节点、Amazon SQS 队列或 Amazon Lambda 函数）的订阅。
> - 要减少通知触发的处理量，请将每个主题的订阅数限制在很小的数目（例如，五个或更少）。

您可以使用 Amazon SNS 控制台、 Amazon Command Line Interface (Amazon CLI) 或，发送应用程序事件通知。 Amazon SDKs

Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择移动，推送通知。
3. 在移动推送通知页面上的平台应用程序部分中，选择应用程序，然后选择编辑。

4. 展开事件通知部分。
5. 依次选择操作和配置事件。
6. 输入 ARNs 要用于以下事件的主题：
   - 已创建端点
   - 已删除端点
   - 已更新端点
   - 传输失败
7. 选择 Save changes（保存更改）。

Amazon CLI

运行 set-platform-application-attributes 命令。

以下示例为全部四个应用程序事件设置相同的 Amazon SNS 主题：

```
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"
```

Amazon SDKs

通过使用软件开发工具包向 Amazon SNS API 提交`SetPlatformApplicationAttributes`请求来设置应用程序事件通知。 Amazon

有关 Amazon SDK 开发者指南和代码示例的完整列表，包括入门帮助和有关先前版本的信息，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。

移动推送 API 操作

要使用 Amazon SNS 移动推送 APIs，您必须首先满足推送通知服务的先决条件，例如 Apple 推送通知服务 (APNs) 和 Firebase 云消息 (FCM)。有关这些先决条件的更多信息，请参阅Amazon SNS 用户通知的先决条件。

要使用向移动应用程序和设备发送推送通知消息 APIs，必须先使用返回`PlatformApplicationArn`属性的`CreatePlatformApplication`操作。然后 `PlatformApplicationArn` 使用 `CreatePlatformEndpoint` 属性，返回 `EndpointArn` 属性。之后，可以在 `EndpointArn` 操作中使用 `Publish` 属性将通知消息发送到移动应用程序和设备，也可以在 `EndpointArn` 操作中使用 `Subscribe` 属性订阅主题。有关更多信息，请参阅 使用 Amazon SNS 设置推送通知。

Amazon SNS 的移动推送 APIs 如下：

`CreatePlatformApplication`

为其中一个支持的推送通知服务（例如 APNs 和 FCM）创建平台应用程序对象，设备和移动应用程序可以注册到该服务。返回 `PlatformApplicationArn` 操作所使用的 `CreatePlatformEndpoint` 属性。

`CreatePlatformEndpoint`

为受支持推送通知服务上的设备和移动应用程序创建终端节点。`CreatePlatformEndpoint` 使用从 `PlatformApplicationArn` 操作返回的 `CreatePlatformApplication` 属性。`EndpointArn` 属性是使用 `CreatePlatformEndpoint` 时返回的，它用在 `Publish` 操作中将通知消息发送到移动应用程序和设备。

`CreateTopic`

创建可以发布消息的主题。

`DeleteEndpoint`

删除一个受支持推送通知服务上的设备和移动应用程序的终端节点。

`DeletePlatformApplication`

删除平台应用程序数据元。

`DeleteTopic`

删除主题及其所有订阅。

GetEndpointAttributes

检索设备和移动应用程序的终端节点属性。

GetPlatformApplicationAttributes

检索平台应用程序数据元的属性。

ListEndpointsByPlatformApplication

列出受支持推送通知服务中的设备和移动应用程序的终端节点和终端节点属性。

ListPlatformApplications

列出受支持推送通知服务的平台应用程序数据元。

Publish

向主题的所有订阅终端节点发送通知消息。

SetEndpointAttributes

设置设备和移动应用程序的终端节点属性。

SetPlatformApplicationAttributes

设置平台应用程序数据元的属性。

Subscribe

准备通过向终端节点发送确认消息来订阅终端节点。要实际创建订阅，端点所有者必须使用确认消息中的令牌调用 ConfirmSubscription 操作。

Unsubscribe

删除订阅。

Amazon SNS 移动推送 API 常见错误

下表列出了 Amazon SNS APIs 为移动推送返回的错误。有关用于移动推送的 Amazon SNS APIs 的更多信息，请参阅。移动推送 API 操作

错误	描述	HTTPS 状态代码	API 操作
Application Name is null string	必需的应用程序名称设置为 Null。	400	CreatePlatformApplication

错误	描述	HTTPS 状态代码	API 操作
Platform Name is null string	必需的平台名称设置为 Null。	400	`CreatePlatformApplication`
Platform Name is invalid	为平台名称提供的或 out-of-range值无效。	400	`CreatePlatformApplication`
APNs — 校长不是有效的证书	为 APNs 主体提供的证书无效，即 SSL 证书。有关更多信息，请参阅CreatePlatformApplication《亚马逊简单通知服务 API 参考》。	400	`CreatePlatformApplication`
APNs — Principal 是有效的证书，但不是.pem 格式	为 APNs 主体提供了非.pem 格式的有效证书，即 SSL 证书。	400	`CreatePlatformApplication`
APNs — 主体是已过期的证书	为 APNs 主体提供了已过期的证书，即 SSL 证书。	400	`CreatePlatformApplication`
APNs — 委托人不是 Apple 颁发的证书	为 APNs 委托人（即 SSL 证书）提供了非 Apple 颁发的证书。	400	`CreatePlatformApplication`
APNs — 未提供本金	未提供 APNs 主体，即 SSL 证书。	400	`CreatePlatformApplication`

错误	描述	HTTPS 状态代码	API 操作
APNs — 未提供凭证	未提供作为私钥的 APNs 凭证。有关更多信息，请参阅CreatePlatformApplication《亚马逊简单通知服务 API 参考》。	400	`CreatePlatformApplication`
APNs — 凭证不是有效的.pem 格式	作为私钥的 APNs 凭证不是有效的.pem 格式。	400	`CreatePlatformApplication`
FCM — APIKey 未提供服务器	未提供 FCM 凭证（即 API 密钥）。有关更多信息，请参阅CreatePlatformApplication《亚马逊简单通知服务 API 参考》。	400	`CreatePlatformApplication`
FCM — 服务器 APIKey 是空的	FCM 凭证（即 API 密钥）为空。	400	`CreatePlatformApplication`
FCM — 服务器 APIKey 为空字符串	FCM 凭证（即 API 密钥）为 Null。	400	`CreatePlatformApplication`
FCM — 服务器 APIKey 无效	FCM 凭证（即 API 密钥）无效。	400	`CreatePlatformApplication`
ADM — clientsecret is not provided	未提供必需的客户端密钥。	400	`CreatePlatformApplication`

错误	描述	HTTPS 状态代码	API 操作
ADM — clientsecret is a null string	客户端密钥所需的字符串为 Null。	400	`CreatePlatformApplication`
ADM — client_secret is empty string	客户端密钥所需的字符串为空。	400	`CreatePlatformApplication`
ADM — client_secret is not valid	客户端密钥所需的字符串无效。	400	`CreatePlatformApplication`
ADM — client_id is empty string	客户端 ID 所需的字符串为空。	400	`CreatePlatformApplication`
ADM — clientId is not provided	未提供客户端 ID 所需的字符串。	400	`CreatePlatformApplication`
ADM — clientid is a null string	客户端 ID 所需的字符串为 Null。	400	`CreatePlatformApplication`
ADM — client_id is not valid	客户端 ID 所需的字符串无效。	400	`CreatePlatformApplication`
EventEndpointCreated ARN 格式无效	EventEndpointCreated ARN 格式无效。	400	`CreatePlatformApplication`
EventEndpointDeleted ARN 格式无效	EventEndpointDeleted ARN 格式无效。	400	`CreatePlatformApplication`

错误	描述	HTTPS 状态代码	API 操作
EventEndpointUpdated ARN 格式无效	EventEndpointUpdated ARN 格式无效。	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure ARN 格式无效	EventDeliveryAttemptFailure ARN 格式无效。	400	`CreatePlatformApplication`
EventDeliveryFailure ARN 格式无效	EventDeliveryFailure ARN 格式无效。	400	`CreatePlatformApplication`
EventEndpointCreated 不是一个现有的话题	EventEndpointCreated 不是现有的主题。	400	`CreatePlatformApplication`
EventEndpointDeleted 不是一个现有的话题	EventEndpointDeleted 不是现有的主题。	400	`CreatePlatformApplication`
EventEndpointUpdated 不是一个现有的话题	EventEndpointUpdated 不是现有的主题。	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure 不是一个现有的话题	EventDeliveryAttemptFailure 不是现有的主题。	400	`CreatePlatformApplication`
EventDeliveryFailure 不是一个现有的话题	EventDeliveryFailure 不是现有的主题。	400	`CreatePlatformApplication`
Platform ARN is invalid	平台 ARN 无效。	400	`SetPlatformAttributes`

错误	描述	HTTPS 状态代码	API 操作
Platform ARN is valid but does not belong to the user	平台 ARN 有效，但不属于该用户。	400	`SetPlatformAttributes`
APNs — 校长不是有效的证书	为 APNs 主体提供的证书无效，即 SSL 证书。有关更多信息，请参阅CreatePlatformApplication《亚马逊简单通知服务 API 参考》。	400	`SetPlatformAttributes`
APNs — Principal 是有效的证书，但不是.pem 格式	为 APNs 主体提供了非.pem 格式的有效证书，即 SSL 证书。	400	`SetPlatformAttributes`
APNs — 主体是已过期的证书	为 APNs 主体提供了已过期的证书，即 SSL 证书。	400	`SetPlatformAttributes`
APNs — 委托人不是 Apple 颁发的证书	为 APNs 委托人（即 SSL 证书）提供了非 Apple 颁发的证书。	400	`SetPlatformAttributes`
APNs — 未提供本金	未提供 APNs 主体，即 SSL 证书。	400	`SetPlatformAttributes`
APNs — 未提供凭证	未提供作为私钥的 APNs 凭证。有关更多信息，请参阅CreatePlatformApplication《亚马逊简单通知服务 API 参考》。	400	`SetPlatformAttributes`

错误	描述	HTTPS 状态代码	API 操作
APNs — 凭证不是有效的.pem 格式	作为私钥的 APNs 凭证不是有效的.pem 格式。	400	`SetPlatformAttributes`
FCM — APIKey 未提供服务器	未提供 FCM 凭证（即 API 密钥）。有关更多信息，请参阅CreatePlatformApplication《亚马逊简单通知服务 API 参考》。	400	`SetPlatformAttributes`
FCM — 服务器 APIKey 为空字符串	FCM 凭证（即 API 密钥）为 Null。	400	`SetPlatformAttributes`
ADM — clientId is not provided	未提供客户端 ID 所需的字符串。	400	`SetPlatformAttributes`
ADM — clientid is a null string	客户端 ID 所需的字符串为 Null。	400	`SetPlatformAttributes`
ADM — clientsecret is not provided	未提供必需的客户端密钥。	400	`SetPlatformAttributes`
ADM — clientsecret is a null string	客户端密钥所需的字符串为 Null。	400	`SetPlatformAttributes`
EventEndpointUpdated ARN 格式无效	EventEndpointUpdated ARN 格式无效。	400	`SetPlatformAttributes`
EventEndpointDeleted ARN 格式无效	EventEndpointDeleted ARN 格式无效。	400	`SetPlatformAttributes`
EventEndpointUpdated ARN 格式无效	EventEndpointUpdated ARN 格式无效。	400	`SetPlatformAttributes`

错误	描述	HTTPS 状态代码	API 操作
EventDeliveryAttemptFailure ARN 格式无效	EventDeliveryAttemptFailure ARN 格式无效。	400	`SetPlatformAttributes`
EventDeliveryFailure ARN 格式无效	EventDeliveryFailure ARN 格式无效。	400	`SetPlatformAttributes`
EventEndpointCreated 不是一个现有的话题	EventEndpointCreated 不是现有的主题。	400	`SetPlatformAttributes`
EventEndpointDeleted 不是一个现有的话题	EventEndpointDeleted 不是现有的主题。	400	`SetPlatformAttributes`
EventEndpointUpdated 不是一个现有的话题	EventEndpointUpdated 不是现有的主题。	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure 不是一个现有的话题	EventDeliveryAttemptFailure 不是现有的主题。	400	`SetPlatformAttributes`
EventDeliveryFailure 不是一个现有的话题	EventDeliveryFailure 不是现有的主题。	400	`SetPlatformAttributes`
Platform ARN is invalid	平台 ARN 无效。	400	`GetPlatformApplicationAttributes`
Platform ARN is valid but does not belong to the user	平台 ARN 有效，但不属于该用户。	403	`GetPlatformApplicationAttributes`
Token specified is invalid	指定的令牌无效。	400	`ListPlatformApplications`

错误	描述	HTTPS 状态代码	API 操作
Platform ARN is invalid	平台 ARN 无效。	400	`ListEndpointsByPlatformApplication`
Platform ARN is valid but does not belong to the user	平台 ARN 有效，但不属于该用户。	404	`ListEndpointsByPlatformApplication`
Token specified is invalid	指定的令牌无效。	400	`ListEndpointsByPlatformApplication`
Platform ARN is invalid	平台 ARN 无效。	400	`DeletePlatformApplication`
Platform ARN is valid but does not belong to the user	平台 ARN 有效，但不属于该用户。	403	`DeletePlatformApplication`
Platform ARN is invalid	平台 ARN 无效。	400	`CreatePlatformEndpoint`
Platform ARN is valid but does not belong to the user	平台 ARN 有效，但不属于该用户。	404	`CreatePlatformEndpoint`
Token is not specified	未指定令牌。	400	`CreatePlatformEndpoint`
Token is not of correct length	令牌长度不正确。	400	`CreatePlatformEndpoint`

错误	描述	HTTPS 状态代码	API 操作
Customer User data is too large	客户用户数据的长度不能超过 2048 个字节（UTF-8 编码）。	400	`CreatePlatformEndpoint`
Endpoint ARN is invalid	端点 ARN 无效。	400	`DeleteEndpoint`
Endpoint ARN is valid but does not belong to the user	端点 ARN 有效，但不属于该用户。	403	`DeleteEndpoint`
Endpoint ARN is invalid	端点 ARN 无效。	400	`SetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	端点 ARN 有效，但不属于该用户。	403	`SetEndpointAttributes`
Token is not specified	未指定令牌。	400	`SetEndpointAttributes`
Token is not of correct length	令牌长度不正确。	400	`SetEndpointAttributes`
Customer User data is too large	客户用户数据的长度不能超过 2048 个字节（UTF-8 编码）。	400	`SetEndpointAttributes`
Endpoint ARN is invalid	端点 ARN 无效。	400	`GetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	端点 ARN 有效，但不属于该用户。	403	`GetEndpointAttributes`
Target ARN is invalid	目标 ARN 无效。	400	`Publish`

错误	描述	HTTPS 状态代码	API 操作
Target ARN is valid but does not belong to the user	目标 ARN 有效，但不属于该用户。	403	`Publish`
Message format is invalid	消息格式无效。	400	`Publish`
Message size is larger than supported by protocol/end-service	消息大小超过协议/端点服务支持的范围。	400	`Publish`

使用移动推送通知的 Amazon SNS 生存时间消息属性

Amazon Simple Notification Service (Amazon SNS) 支持设置移动推送通知消息的生存时间 (TTL) 消息属性。该功能不同于在移动推送通知服务的 Amazon SNS 消息正文中设置 TTL（如果该推送通知服务支持的话）的功能，如发送到 Android 时的 Amazon Device Messaging（ADM）和 Firebase Cloud Messaging（FCM）。

TTL 消息属性用于指定有关消息的过期元数据。这允许您指定推送通知服务（例如 Apple 推送通知服务 (APNs) 或 FCM）将消息传送到端点所需的时间。如果因为某种原因（如移动设备已关闭），消息无法在指定的 TTL 内传达，则系统将丢弃该消息，且不再尝试传送它。要在消息属性中指定 TTL，可以使用 Amazon Web Services Management Console、 Amazon 软件开发套件 (SDKs) 或查询 API。

推送通知服务的 TTL 消息属性

以下是推送通知服务的 TTL 消息属性列表，您可以使用这些属性在使用 Amazon SDKs 或查询 API 时进行设置：

推送通知服务	TTL 消息属性
Amazon Device Messaging（ADM）	`AWS.SNS.MOBILE.ADM.TTL`
苹果推送通知服务 (APNs)	`AWS.SNS.MOBILE.APNS.TTL`
Apple 推送通知服务沙盒 (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`

推送通知服务	TTL 消息属性
百度云推送（百度）	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging（FCM，发送到 Android 时）	`AWS.SNS.MOBILE.FCM.TTL`
Windows 推送通知服务（WNS）	`AWS.SNS.MOBILE.WNS.TTL`

每个推送通知服务以不同的方式处理 TTL。Amazon SNS 提供了涵盖所有推送通知服务的 TTL 抽象视图，使您能够更方便地指定 TTL。当您使用指定 TTL（ Amazon Web Services Management Console 以秒为单位）时，您只需输入一次 TTL 值，然后 Amazon SNS 将在发布消息时计算每种选定推送通知服务的 TTL。

TTL 是相对于发布时间的。在将推送通知消息转交给特定的推送通知服务之前，Amazon SNS 会计算该推送通知的停留时间（发布时间戳与将消息转交给推送通知服务之间的时间差），并将剩余的 TTL 传递给特定的推送通知服务。如果 TTL 短于停留时间，Amazon SNS 不会尝试发布。

如果您为推送通知消息指定 TTL，则 TTL 值必须为正整数，除非的值对推送通知服务`0`有特定的含义，例如 with APNs 和 FCM（发送到 Android 时）。如果 TTL 值设为 `0`，但该推送通知服务对 `0` 无具体意义，则 Amazon SNS 将丢弃该消息。有关使用`0`时设置为的 TTL 参数的更多信息 APNs，请参阅 Binary Provider API 文档中远程通知的表 A-3 项目标识符。

决定 TTL 的优先顺序

Amazon SNS 根据以下顺序来决定推送通知消息的 TTL，数字越小，优先级越高：

1. 消息属性 TTL
2. 消息正文 TTL
3. 推送通知服务默认 TTL（随服务而变）
4. Amazon SNS 默认 TTL（4 周）

如果您为同一条消息设置了不同的 TTL 值（分别是消息属性和消息正文的 TTL），则 Amazon SNS 会修改消息正文中的 TTL，以匹配消息属性中指定的 TTL。

使用指定 TTL Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。

2. 在导航面板中，选择移动，推送通知。
3. 在 Mobile push notifications (移动推送通知) 页面上的平台应用程序部分中，选择应用程序。
4. 在该***MyApplication***页面的终端节点部分，选择应用程序终端节点，然后选择发布消息。
5. 在消息详细信息部分中，输入 TTL（推送通知服务必须向终端节点发送消息的秒数）。
6. 选择发布消息。

Amazon SNS 移动应用程序支持的区域

目前，您可以在以下区域内创建移动应用程序：

- 美国东部（俄亥俄州）
- 美国东部（弗吉尼亚州北部）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 非洲（开普敦）
- 亚太地区（香港）
- 亚太地区（雅加达）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（东京）
- 加拿大（中部）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲地区（巴黎）
- 欧洲地区（斯德哥尔摩）
- 中东（巴林）

- 中东（阿联酋）
- 南美洲（圣保罗）
- Amazon GovCloud （美国西部）

管理 Amazon SNS 移动推送通知的最佳实践

本部分介绍可帮助您提升客户参与度的最佳实践。

终端节点管理

如果由于用户在设备上进行操作（例如，在设备上重新安装应用程序）导致设备令牌发生变化，或者证书更新影响了在特定 iOS 版本上运行的设备，则可能导致传送过程出现问题。Apple 推荐的最佳做法是在 APNs 每次启动应用程序时进行注册。

由于在用户每次打开应用时设备令牌不会发生变化，因此可以使用幂等 `CreatePlatformEndpoint` API。但是，如果令牌本身无效，或者端点有效但已禁用（例如，生产环境和沙盒环境不匹配），则上述方法可能会为同一设备引入重复项。

可以使用设备令牌管理机制，例如伪代码中的一种此类机制。

有关管理和维护 FCM v1 设备令牌的信息，请参阅 Amazon SNS 管理 Firebase Cloud Messaging 端点。

传送状态日志记录

要监控推送通知传送状态，我们建议您为 Amazon SNS 平台应用程序启用传送状态日志记录。这有助于您排查传送失败问题，因为日志包含从推送平台服务返回的提供商响应代码。有关启用传送状态日志记录的详细信息，请参阅如何访问 Amazon SNS 主题的推送通知传送日志记录？

事件通知

要以事件驱动的方式管理终端节点，您可以利用事件通知功能。这样，已配置的 Amazon SNS 主题就可以针对终端节点创建、删除、更新和传送失败等平台应用程序事件，向订阅者（例如 Lambda 函数）发送事件。

Amazon SNS 电子邮件订阅设置和管理

您可以使用、或为电子邮件地址订阅 Amazon SNS 主题。 Amazon Web Services Management Console 适用于 Java 的 Amazon SDK 适用于 .NET 的 Amazon SDK

> **备注**
>
> - 不支持自定义电子邮件正文。电子邮件传输功能旨在提供内部系统提示，而不是营销消息。
> - 仅标准主题支持直接订阅电子邮件端点。
> - 电子邮件传送吞吐量受到限制。有关更多信息，请参阅 Amazon SNS 配额。

> **Important**
>
> 要防止邮件列表接收人取消订阅来自 Amazon SNS 主题电子邮件的所有接收人，请参阅设置需要身份验证才能从 Amazon 支持取消订阅的电子邮件订阅。

使用电子邮件地址订阅 Amazon SNS 主题 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择订阅。
3. 在订阅页面上，选择创建订阅。
4. 在创建订阅页上的详细信息部分，执行以下操作：
   a. 对于 Topic ARN（主题 ARN），选择主题的 Amazon Resource Name (ARN)。
   b. 对于协议，选择电子邮件。
   c. 对于 Endpoint（终端节点），输入电子邮件地址。
   d. （可选）要配置筛选策略，请展开 Subscription filter policy（订阅筛选策略）部分。有关更多信息，请参阅 Amazon SNS 订阅筛选策略。
   e. （可选）要启用基于有效负载的筛选，请将 `Filter Policy Scope` 配置为 `MessageBody`。有关更多信息，请参阅 Amazon SNS 订阅筛选策略范围。
   f. （可选）要为订阅配置死信队列，请展开 Redrive policy (dead-letter queue)（重新驱动策略（死信队列））部分。有关更多信息，请参阅 Amazon SNS 死信队列。
   g. 选择创建订阅。

      控制台将创建订阅并打开订阅的 Details（详细信息）页面。

您必须先确认订阅，然后才能开始接收消息。

要确认订阅

1. 检查您的电子邮件收件箱，然后从 Amazon SNS 中的电子邮件中选择 Confirm subscription（确认订阅）。
2. Amazon SNS 会打开您的 Web 浏览器，并显示带有您的订阅 ID 的订阅确认信息。

使用软件开发工具包订阅 Amazon SNS 主题的电子邮件地址 Amazon

要使用 S Amazon DK，必须使用您的凭据对其进行配置。有关更多信息，请参阅和工具参考指南中的共享配置Amazon SDKs 和凭据文件。

以下代码示例演示如何使用 `Subscribe`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

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

使用可选筛选条件为队列订阅主题。

```
    /// <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;
```

```
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 Subscribe。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

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

将移动应用程序订阅到主题。

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

将 Lambda 函数订阅到主题。

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

将 SQS 队列订阅到主题。

```
        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;
            }
```

使用筛选器订阅主题。

```
        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;
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 Subscribe。

CLI

Amazon CLI

订阅主题

以下 `subscribe` 命令将电子邮件地址订阅到指定主题。

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

输出：

```
{
    "SubscriptionArn": "pending confirmation"
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 Subscribe。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用可选筛选条件为队列订阅主题。

```
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
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 Subscribe。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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);
        }
    }
}
```

将 HTTP 端点订阅到主题。

```
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);
        }
    }
}
```

将 Lambda 函数订阅到主题。

```
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 "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 Subscribe。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

将移动应用程序订阅到主题。

```
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;
};
```

将 Lambda 函数订阅到主题。

```
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;
};
```

将 SQS 队列订阅到主题。

```
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;
};
```

使用筛选器订阅主题。

```
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;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 Subscribe。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

将 Lambda 函数订阅到主题。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Kotlin API 参考中的 Unsubscribe。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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());
}
```

将 HTTP 端点订阅到主题。

```
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());
}
```

- 有关 API 详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 Subscribe。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 Subscribe。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 Subscribe。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 Subscribe。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
   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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的订阅。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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.")
```

通过电话号码订阅主题即可通过短信接收通知。

```
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.")
```

- 如需详细了解 API，请参阅在 Amazon Swift 开发工具包中订阅 API 参考。

使用 Amazon SNS 的代码示例 Amazon SDKs

以下代码示例展示了如何将 Amazon SNS 与 Amazon 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务服务结合来完成特定任务的代码示例。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

开始使用 Amazon SNS

以下代码示例展示了如何开始使用 Amazon SNS。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ListTopics中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_sns.cpp 源文件的代码。

```
#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;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ListTopics中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ListTopics中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListTopics中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

初始化 SNS 客户端，并在您的账户中列出主题。

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

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListTopics中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListTopics于 K otlin 的Amazon SDK API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

Package.swift 文件。

```
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")

    ]
)
```

主要 Swift 程序。

```
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)
        }
    }
}
```

- 如需了解 API 的详细信息，请参阅适用ListTopics于 S wift 的Amazon SDK API 参考。

代码示例

- 使用 Amazon SNS 的基本示例 Amazon SDKs
  - 开始使用 Amazon SNS
  - 使用 Amazon SNS 执行的操作 Amazon SDKs
    - CheckIfPhoneNumberIsOptedOut与 Amazon SDK 或 CLI 配合使用
    - ConfirmSubscription与 Amazon SDK 或 CLI 配合使用
    - CreateTopic与 Amazon SDK 或 CLI 配合使用
    - DeleteTopic与 Amazon SDK 或 CLI 配合使用
    - GetSMSAttributes与 Amazon SDK 或 CLI 配合使用
    - GetTopicAttributes与 Amazon SDK 或 CLI 配合使用
    - ListPhoneNumbersOptedOut与 Amazon SDK 或 CLI 配合使用
    - ListSubscriptions与 Amazon SDK 或 CLI 配合使用
    - ListTopics与 Amazon SDK 或 CLI 配合使用
    - Publish与 Amazon SDK 或 CLI 配合使用
    - SetSMSAttributes与 Amazon SDK 或 CLI 配合使用
    - SetSubscriptionAttributes与 Amazon SDK 或 CLI 配合使用
    - 与 Amazon SDK SetSubscriptionAttributesRedrivePolicy 配合使用
    - SetTopicAttributes与 Amazon SDK 或 CLI 配合使用
    - Subscribe与 Amazon SDK 或 CLI 配合使用
    - TagResource与 Amazon SDK 或 CLI 配合使用
    - Unsubscribe与 Amazon SDK 或 CLI 配合使用
- 使用 Amazon SNS 的场景 Amazon SDKs
  - 构建应用程序以将数据提交到 DynamoDB 表
  - 构建转换消息的发布和订阅应用程序

- 使用软件开发工具包为 Amazon SNS 推送通知创建平台终端节点 Amazon
- 创建照片资产管理应用程序，让用户能够使用标签管理照片
- 创建 Amazon Textract 浏览器应用程序
- 使用软件开发工具包创建并发布到 FIFO Amazon SNS 主题 Amazon
- 使用 Amazon Rekognition 使用软件开发工具包检测视频中的人物和物体 Amazon
- 使用软件开发工具包向 Amazon SNS 主题发布短信 Amazon
- 使用软件开发工具包通过 Amazon S3 向亚马逊 SNS 发布一条大消息 Amazon
- 使用软件开发工具包发布 Amazon SNS 短信 Amazon
- 使用软件开发工具包将 Amazon SNS 消息发布到亚马逊 SQS 队列 Amazon
- 使用 API Gateway 调用 Lambda 函数
- 使用计划的事件调用 Lambda 函数

- 亚马逊 SNS 的无服务器示例
  - 通过 Amazon SNS 触发器调用 Lambda 函数

使用 Amazon SNS 的基本示例 Amazon SDKs

以下代码示例展示了如何将 Amazon 简单通知服务的基础知识与配合使用 Amazon SDKs。

示例

- 开始使用 Amazon SNS
- 使用 Amazon SNS 执行的操作 Amazon SDKs
  - CheckIfPhoneNumberIsOptedOut与 Amazon SDK 或 CLI 配合使用
  - ConfirmSubscription与 Amazon SDK 或 CLI 配合使用
  - CreateTopic与 Amazon SDK 或 CLI 配合使用
  - DeleteTopic与 Amazon SDK 或 CLI 配合使用
  - GetSMSAttributes与 Amazon SDK 或 CLI 配合使用
  - GetTopicAttributes与 Amazon SDK 或 CLI 配合使用
  - ListPhoneNumbersOptedOut与 Amazon SDK 或 CLI 配合使用
  - ListSubscriptions与 Amazon SDK 或 CLI 配合使用
  - ListTopics与 Amazon SDK 或 CLI 配合使用
  - Publish与 Amazon SDK 或 CLI 配合使用

- SetSMSAttributes与 Amazon SDK 或 CLI 配合使用
- SetSubscriptionAttributes与 Amazon SDK 或 CLI 配合使用
- 与 Amazon SDK SetSubscriptionAttributesRedrivePolicy 配合使用
- SetTopicAttributes与 Amazon SDK 或 CLI 配合使用
- Subscribe与 Amazon SDK 或 CLI 配合使用
- TagResource与 Amazon SDK 或 CLI 配合使用
- Unsubscribe与 Amazon SDK 或 CLI 配合使用

开始使用 Amazon SNS

以下代码示例展示了如何开始使用 Amazon SNS。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ListTopics中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})
```

```
if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_sns.cpp 源文件的代码。

```
#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;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ListTopics中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ListTopics中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListTopics中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

初始化 SNS 客户端，并在您的账户中列出主题。

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

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListTopics中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
```

```
            }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListTopics于 K otlin 的Amazon SDK API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

Package.swift 文件。

```
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")

    ]
)
```

主要 Swift 程序。

```
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)
        }
    }
}
```

- 如需了解 API 的详细信息，请参阅适用ListTopics于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SNS 执行的操作 Amazon SDKs

以下代码示例演示了如何使用执行单个 Amazon SNS 操作。 Amazon SDKs每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon SNS API，是必须在上下文中运行的较大型程序的代码节选。您可以在使用 Amazon SNS 的场景 Amazon SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Simple Notification Service API 参考。

示例

- CheckIfPhoneNumberIsOptedOut与 Amazon SDK 或 CLI 配合使用
- ConfirmSubscription与 Amazon SDK 或 CLI 配合使用
- CreateTopic与 Amazon SDK 或 CLI 配合使用
- DeleteTopic与 Amazon SDK 或 CLI 配合使用
- GetSMSAttributes与 Amazon SDK 或 CLI 配合使用
- GetTopicAttributes与 Amazon SDK 或 CLI 配合使用
- ListPhoneNumbersOptedOut与 Amazon SDK 或 CLI 配合使用
- ListSubscriptions与 Amazon SDK 或 CLI 配合使用
- ListTopics与 Amazon SDK 或 CLI 配合使用
- Publish与 Amazon SDK 或 CLI 配合使用
- SetSMSAttributes与 Amazon SDK 或 CLI 配合使用
- SetSubscriptionAttributes与 Amazon SDK 或 CLI 配合使用
- 与 Amazon SDK SetSubscriptionAttributesRedrivePolicy 配合使用
- SetTopicAttributes与 Amazon SDK 或 CLI 配合使用
- Subscribe与 Amazon SDK 或 CLI 配合使用
- TagResource与 Amazon SDK 或 CLI 配合使用
- Unsubscribe与 Amazon SDK 或 CLI 配合使用

`CheckIfPhoneNumberIsOptedOut`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CheckIfPhoneNumberIsOptedOut`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考CheckIfPhoneNumberIsOptedOut中的。

CLI

Amazon CLI

检查电话号码的 SMS 消息退出

以下`check-if-phone-number-is-opted-out`示例检查指定的电话号码是否已选择不接收来自当前 Amazon 账户的 SMS 消息。

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

输出：

```
{
    "isOptedOut": false
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CheckIfPhoneNumberIsOptedOut中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
```

```
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CheckIfPhoneNumberIsOptedOut中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
```

```
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CheckIfPhoneNumberIsOptedOut中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考CheckIfPhoneNumberIsOptedOut中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmSubscription`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmSubscription`。

CLI

Amazon CLI

确认订阅

以下 `confirm-subscription` 命令将完成订阅名为 `my-topic` 的 SNS 主题时启动的确认过程。--token 参数来自发送到订阅调用中指定的通知端点的确认消息。

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

输出：

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ConfirmSubscription中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ConfirmSubscription中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
```

```
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ConfirmSubscription中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考ConfirmSubscription中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateTopic`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateTopic`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并发布到 FIFO 主题
- 将消息发布到队列

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用特定的名称创建主题。

```
    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;
        }
    }
```

创建一个包含名称以及特定 FIFO 和重复数据消除属性的新主题。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考CreateTopic中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考CreateTopic中的。

CLI

Amazon CLI

创建 SNS 主题

以下 `create-topic` 示例将创建名为 `my-topic` 的 SNS 主题。

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

输出：

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

有关更多信息，请参阅《Amazon 命令行界面用户指南》中的在 Amazon SQS 和 Amazon SNS 中使用命令Amazon 行界面。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateTopic中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考CreateTopic中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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 "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateTopic中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

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

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

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CreateTopic中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

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

- 有关 API 的详细信息，请参阅适用CreateTopic于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考CreateTopic中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用CreateTopic于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考CreateTopic中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用CreateTopic于 Rust 的Amazon SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   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.
```

- 有关 API 的详细信息，请参阅适用CreateTopic于 S AP 的Amazon SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
        }
```

- 如需了解 API 的详细信息，请参阅适用CreateTopic于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteTopic`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteTopic`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 将消息发布到队列

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按主题 ARN 删除主题。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DeleteTopic中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DeleteTopic中的。

CLI

Amazon CLI

删除 SNS 主题

以下 `delete-topic` 示例将删除指定的 SNS 主题。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteTopic中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考DeleteTopic中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteTopic中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
```

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

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteTopic中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

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

```
    }
}
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DeleteTopic中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 Python 的Amazon SDK (Boto3) API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DeleteTopic于 S AP 的Amazon SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSSNS

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

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

- 如需了解 API 的详细信息，请参阅适用DeleteTopic于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetSMSAttributes`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetSMSAttributes`。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 Get SMSAttributes in 适用于 C++ 的 Amazon SDK API 参考。

CLI

Amazon CLI

列出默认 SMS 消息属性

以下 `get-sms-attributes` 示例将列出发送 SMS 消息的默认属性。

```
aws sns get-sms-attributes
```

输出：

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetSMSAttributes。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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");
    }
}
```

- 有关 API 的详细信息，请参阅 Get SMSAttributes in Amazon SDK for Java 2.x API 参考。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );
```

```
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 Get SMSAttributes in 适用于 JavaScript 的 Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 Get SMSAttributes in 适用于 PHP 的 Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetTopicAttributes`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetTopicAttributes`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考GetTopicAttributes中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考GetTopicAttributes中的。

CLI

Amazon CLI

检索主题的属性

以下 `get-topic-attributes` 示例将显示指定主题的属性。

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

输出：

```
{
    "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"
    }
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考GetTopicAttributes中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考GetTopicAttributes中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
```

```
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考GetTopicAttributes中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

导入 SDK 和客户端模块，然后调用 API。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考GetTopicAttributes中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
```

```
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- 有关 API 的详细信息，请参阅适用GetTopicAttributes于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考GetTopicAttributes中的。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   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.
```

- 有关 API 的详细信息，请参阅适用GetTopicAttributes于 S AP 的Amazon SDK ABAP API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListPhoneNumbersOptedOut`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListPhoneNumbersOptedOut`。

CLI

Amazon CLI

列出 SMS 消息退出

以下 `list-phone-numbers-opted-out` 示例将列出退出 SMS 消息接收的电话号码。

```
aws sns list-phone-numbers-opted-out
```

输出：

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListPhoneNumbersOptedOut中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListPhoneNumbersOptedOut中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考ListPhoneNumbersOptedOut中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListSubscriptions`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListSubscriptions`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ListSubscriptions中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
//! 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;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ListSubscriptions中的。

CLI

Amazon CLI

列出 SNS 订阅

以下`list-subscriptions`示例显示了您 Amazon 账户中的 SNS 订阅列表。

```
aws sns list-subscriptions
```

输出：

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

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListSubscriptions中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListSubscriptions中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
```

```
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListSubscriptions中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun listSNSSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListSubscriptions于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考ListSubscriptions中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用ListSubscriptions于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考ListSubscriptions中的。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   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.
```

- 有关 API 的详细信息，请参阅适用ListSubscriptions于 S AP 的Amazon SDK ABAP API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListTopics`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListTopics`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ListTopics中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
//! 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;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ListTopics中的。

CLI

Amazon CLI

列出 SNS 主题

以下`list-topics`示例列出了您 Amazon 账户中的所有 SNS 主题。

```
aws sns list-topics
```

输出：

```
{
```

```
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListTopics中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ListTopics中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListTopics中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListTopics中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListTopics于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考ListTopics中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用ListTopics于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考ListTopics中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用ListTopics于 Rust 的Amazon SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用ListTopics于 S AP 的Amazon SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)
            }
        }
```

- 如需了解 API 的详细信息，请参阅适用ListTopics于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`Publish`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `Publish`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并发布到 FIFO 主题
- 发布 SMS 文本消息
- 将消息发布到队列

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

向主题发布消息。

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

使用组、复制和属性选项向主题发布消息。

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

将用户的选择应用于发布操作。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 Publish。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

发布带有属性的消息。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 Publish。

CLI

Amazon CLI

示例 1：向主题发布消息

以下 `publish` 示例将指定消息发布到指定 SNS 主题。该消息来自一个文本文件，您可以在该文件中包含换行符。

```
aws sns publish \
```

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

message.txt 的内容：

```
Hello World
Second Line
```

输出：

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

示例 2：向电话号码发布 SMS 消息

以下 publish 示例将消息 Hello world! 发布到电话号码 +1-555-555-0100。

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

输出：

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

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 Publish。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 Publish。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 Publish。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                   if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
```

```
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

使用组、复制和属性选项向主题发布消息。

```
  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();
    }
  }
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考中的 Publish。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

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

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 Publish。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
```

```
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 Publish。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例显示发布一条 MessageAttribute 声明为内联的消息。

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

示例 2：此示例显示发布一条事先 MessageAttributes 声明了多个消息的情况。

```
$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
```

- 有关 API 的详细信息，请参阅在 Amazon Tools for PowerShell Cmdlet 参考中发布 (V4)。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

发布包含属性的消息，以便订阅可以根据属性进行筛选。

```
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
```

发布基于订阅用户的协议采取不同形式的消息。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 Publish。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 Publish。

Rust

SDK for Rust

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 Publish。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的发布。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)")
```

- 有关 API 的详细信息，请参阅在 Amazon Swift 开发工具包中发布 API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SetSMSAttributes`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SetSMSAttributes`。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

如何使用 Amazon SNS 设置默认SMSType 属性。

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考SMSAttributes中的设置。

CLI

Amazon CLI

设置 SMS 消息属性

以下 `set-sms-attributes` 示例将 SMS 消息的默认发件人 ID 设置为 `MyName`。

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅SMSAttributes《Amazon CLI 命令参考》中的 “设置”。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SMSAttributes中的设置。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing
 messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考SMSAttributes中的设置。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考SMSAttributes中的设置。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SetSubscriptionAttributes`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SetSubscriptionAttributes`。

CLI

Amazon CLI

设置订阅属性

以下 `set-subscription-attributes` 示例将 `RawMessageDelivery` 属性设置为 SQS 订阅。

```
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
```

此命令不生成任何输出。

以下 `set-subscription-attributes` 示例将 `FilterPolicy` 属性设置为 SQS 订阅。

```
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\"] }"
```

此命令不生成任何输出。

以下 `set-subscription-attributes` 示例从 SQS 订阅中移除 `FilterPolicy` 属性。

```
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 "{}"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetSubscriptionAttributes中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SetSubscriptionAttributes中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用SetSubscriptionAttributes于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 Amazon SDK **SetSubscriptionAttributesRedrivePolicy** 配合使用

以下代码示例演示了如何使用 `SetSubscriptionAttributesRedrivePolicy`。

Java

适用于 Java 的 SDK 1.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// 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);
```

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SetTopicAttributes`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SetTopicAttributes`。

CLI

Amazon CLI

为主题设置属性

以下 `set-topic-attributes` 示例为指定主题设置 `DisplayName` 属性。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetTopicAttributes中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SetTopicAttributes中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考SetTopicAttributes中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- 有关 API 的详细信息，请参阅适用SetTopicAttributes于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考SetTopicAttributes中的。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考SetTopicAttributes中的。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 的详细信息，请参阅适用SetTopicAttributes于 S AP 的Amazon SDK ABAP API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`Subscribe`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `Subscribe`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并发布到 FIFO 主题
- 将消息发布到队列

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

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

使用可选筛选条件为队列订阅主题。

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

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 Subscribe。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

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

将移动应用程序订阅到主题。

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

将 Lambda 函数订阅到主题。

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

```
}
```

将 SQS 队列订阅到主题。

```
        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;
            }
```

使用筛选器订阅主题。

```
        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;
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 Subscribe。

CLI

Amazon CLI

订阅主题

以下 `subscribe` 命令将电子邮件地址订阅到指定主题。

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

输出：

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 Subscribe。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用可选筛选条件为队列订阅主题。

```
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
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 Subscribe。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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);
        }
    }
}
```

将 HTTP 端点订阅到主题。

```
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);
        }
    }
}
```

将 Lambda 函数订阅到主题。

```
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 "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 Subscribe。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

将移动应用程序订阅到主题。

```
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;
};
```

将 Lambda 函数订阅到主题。

```
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;
};
```

将 SQS 队列订阅到主题。

```
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;
};
```

使用筛选器订阅主题。

```
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;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 Subscribe。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
```

```
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

将 Lambda 函数订阅到主题。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Kotlin API 参考中的 Unsubscribe。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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());
}
```

将 HTTP 端点订阅到主题。

```
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());
}
```

- 有关 API 详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 Subscribe。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 Subscribe。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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
```

- 有关更多信息，请参阅《适用于 Ruby 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 Subscribe。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 Subscribe。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
   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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的订阅。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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.")
```

通过电话号码订阅主题即可通过短信接收通知。

```
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.")
```

- 如需详细了解 API，请参阅在 Amazon Swift 开发工具包中订阅 API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`TagResource`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `TagResource`。

CLI

Amazon CLI

为主题添加标签

以下 `tag-resource` 示例将元数据标签添加到指定 Amazon SNS 主题。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考TagResource中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考TagResource中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

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

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

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

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

- 有关 API 的详细信息，请参阅适用TagResource于 K otlin 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`Unsubscribe`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `Unsubscribe`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 将消息发布到队列

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

通过订阅 ARN 取消订阅某个主题。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 Unsubscribe。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的 Unsubscribe。

CLI

Amazon CLI

从主题取消订阅

以下 `unsubscribe` 示例将从主题删除指定的订阅。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 Unsubscribe。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 Unsubscribe。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
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({});
```

导入 SDK 和客户端模块，然后调用 API。

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

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
```

```
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 Unsubscribe。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
```

```
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 Unsubscribe。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考 中的 Unsubscribe。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 Unsubscribe。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的 Unsubscribe。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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.")
```

- 有关 API 的详细信息，请参阅 S wift Amazon 开发工具包中的 “取消订阅 API 参考”。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SNS 的场景 Amazon SDKs

以下代码示例向您展示了如何使用在 Amazon SNS 中实现常见场景。 Amazon SDKs这些场景向您展示了如何通过调用 Amazon SNS 中的多个函数或与其他 Amazon Web Services 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 构建应用程序以将数据提交到 DynamoDB 表

- 构建转换消息的发布和订阅应用程序
- 使用软件开发工具包为 Amazon SNS 推送通知创建平台终端节点 Amazon
- 创建照片资产管理应用程序，让用户能够使用标签管理照片
- 创建 Amazon Textract 浏览器应用程序
- 使用软件开发工具包创建并发布到 FIFO Amazon SNS 主题 Amazon
- 使用 Amazon Rekognition 使用软件开发工具包检测视频中的人物和物体 Amazon
- 使用软件开发工具包向 Amazon SNS 主题发布短信 Amazon
- 使用软件开发工具包通过 Amazon S3 向亚马逊 SNS 发布一条大消息 Amazon
- 使用软件开发工具包发布 Amazon SNS 短信 Amazon
- 使用软件开发工具包将 Amazon SNS 消息发布到亚马逊 SQS 队列 Amazon
- 使用 API Gateway 调用 Lambda 函数
- 使用计划的事件调用 Lambda 函数

构建应用程序以将数据提交到 DynamoDB 表

以下代码示例展示如何构建将数据提交到 Amazon DynamoDB 表并在用户更新该表时通知您的应用程序。

Java

适用于 Java 的 SDK 2.x

展示如何创建动态 Web 应用程序，该应用程序使用 Amazon DynamoDB Java API 提交数据并使用 Amazon Simple Notification Service Java API 发送文本消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SNS

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

此示例展示了如何构建一个应用程序，使用户能够向 Amazon DynamoDB 表提交数据，并使用 Amazon Simple Notification Service (Amazon SNS) 向管理员发送文本消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 适用于 JavaScript 的 Amazon SDK v3 开发人员指南中找到。

本示例中使用的服务

- DynamoDB
- Amazon SNS

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

构建转换消息的发布和订阅应用程序

以下代码示例展示如何创建具有订阅和发布功能并能转换消息的应用程序。

.NET

适用于 .NET 的 Amazon SDK

展示如何使用 Amazon Simple Notification Service .NET API 创建具有订阅和发布功能的 Web 应用程序。此外，此示例应用程序还会转换消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Simple Notification Service Java API 创建具有订阅和发布功能的 Web 应用程序。此外，此示例应用程序还会转换消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

有关如何设置和运行使用 Java Async API 的示例的完整源代码和说明，请参阅上的GitHub完整示例。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

Kotlin

适用于 Kotlin 的 SDK

展示如何使用 Amazon SNS Kotlin API 创建具有订阅和发布功能的应用程序。此外，此示例应用程序还会转换消息。

有关如何创建 Web 应用程序的完整源代码和说明，请参阅上的完整示例GitHub。

有关如何创建原生 Android 应用程序的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包为 Amazon SNS 推送通知创建平台终端节点 Amazon

以下代码示例展示如何为 Amazon SNS 推送通知创建平台端点。

CLI

Amazon CLI

创建平台应用程序端点

以下 `create-platform-endpoint` 示例使用指定令牌为指定平台应用程序创建端点。

```
aws sns create-platform-endpoint \
```

```
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

输出：

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建照片资产管理应用程序，让用户能够使用标签管理照片

以下代码示例演示了如何创建无服务器应用程序，让用户能够使用标签管理照片。

.NET

适用于 .NET 的 Amazon SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

适用于 Java 的 SDK 2.x

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

适用于 Kotlin 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

适用于 PHP 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

适用于 Rust 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 Amazon 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何使用 适用于 JavaScript 的 Amazon SDK 来构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并将其显示在交互式网页中。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用 Amazon Simple Storage Service（Amazon S3）进行存储；对于通知，它将轮询订阅 Amazon Simple Notification Service（Amazon SNS）主题的 Amazon Simple Queue Service（Amazon SQS）队列。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

适用于 Python 的 SDK（Boto3）

演示如何 适用于 Python (Boto3) 的 Amazon SDK 与 Amazon Textract 配合使用来检测文档图像中的文本、表单和表格元素。输入图像和 Amazon Textract 输出在 Tkinter 应用程序中显示，该应用程序可让您探索检测到的元素。

- 将文档图像提交到 Amazon Textract 并探索检测到的元素的输出。

- 将图像直接提交到 Amazon Textract，或通过 Amazon Simple Storage Service（Amazon S3）桶提交图像。
- 使用异步 APIs 启动任务，该任务在任务完成时向亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 主题发布通知。
- 轮询 Amazon Simple Queue Service (Amazon SQS) 队列，以获取任务完成消息并显示结果。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包创建并发布到 FIFO Amazon SNS 主题 Amazon

以下代码示例展示如何创建并发布到 FIFO Amazon SNS 主题。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个 Amazon SQS FIFO 队列和一个标准队列。
- 将队列订阅到主题，发布一条消息到主题。

该测试验证每个队列是否收到消息。完整的示例还显示了添加访问策略，并在最后删除了资源。

```
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);
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。
  - CreateTopic
  - 发布
  - Subscribe

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 Amazon SNS FIFO 主题，将 Amazon SQS FIFO 队列和标准队列订阅到主题，并发布一条消息到主题。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateTopic
  - 发布
  - Subscribe

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 FIFO 主题并为此订阅 Amazon SQS FIFO 队列，然后向 Amazon SNS 主题发布消息。

```
    " 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.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的Amazon SDK 的 API 参考中的以下主题。
  - CreateTopic
  - 发布
  - Subscribe

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Rekognition 使用软件开发工具包检测视频中的人物和物体 Amazon

以下代码示例展示如何使用 Amazon Rekognition 检测视频中的人物和对象。

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3

- Amazon SES
- Amazon SNS
- Amazon SQS

Python

适用于 Python 的 SDK（Boto3）

通过启动异步检测任务，使用 Amazon Rekognition 来检测视频中的人脸、对象和人物。此示例还将 Amazon Rekognition 配置为在任务完成时通知 Amazon Simple Notification Service (Amazon SNS) 主题，并订阅该主题的 Amazon Simple Queue Service (Amazon SQS) 队列。当队列收到有关任务的消息时，将检索该任务并输出结果。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包向 Amazon SNS 主题发布短信 Amazon

以下代码示例显示了如何：

- 创建 Amazon SNS 主题。
- 使用手机号码订阅主题。
- 向主题发布 SMS 消息，以使所有订阅的电话号码一次接收消息。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个主题并返回其 ARN。

```
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 "";
    }
}
```

为终端节点订阅主题。

```
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);
        }
    }
}
```

设置消息的属性，例如发件人的 ID、最高价格及其类型。消息属性是可选的。

```
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);
        }
    }
}
```

向主题发布消息。消息将会发送到每个订阅用户。

```
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);
        }
    }
}
```

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包通过 Amazon S3 向亚马逊 SNS 发布一条大消息 Amazon

以下代码示例演示了如何使用 Amazon S3 向 Amazon SNS 发布大型消息来存储消息负载。

Java

适用于 Java 的 SDK 1.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

要发布大型消息，请使用适用于 Java 的 Amazon SNS 扩展客户端库。您发送的消息将引用包含实际消息内容的 Amazon S3 对象。

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

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包发布 Amazon SNS 短信 Amazon

以下代码示例演示了如何使用 Amazon SNS 发布 SMS 消息。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 Publish。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 Publish。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 Publish。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 Publish。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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());
}
```

- 有关更多信息，请参阅《适用于 PHP 的 Amazon SDK 开发人员指南》。
- 有关 API 详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 Publish。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 Publish。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包将 Amazon SNS 消息发布到亚马逊 SQS 队列 Amazon

以下代码示例演示了如何：

- 创建主题（FIFO 或非 FIFO）。
- 针对主题订阅多个队列，并提供应用筛选条件的选项。
- 将消息发布到主题。
- 轮询队列中是否有收到的消息。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

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

创建一个包装 Amazon SQS 操作的类。

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

创建一个包装 Amazon SNS 操作的类。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- 发布
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage
  - SetQueueAttributes

- Subscribe
- Unsubscribe

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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))
}
```

定义一个结构来包装此示例中使用的 Amazon SNS 操作。

```
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
}
```

定义一个结构来包装此示例中使用的 Amazon SQS 操作。

```
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
}
```

清理资源。

```
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)
  }
 }
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API 参考》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage

- SetQueueAttributes
- Subscribe
- Unsubscribe

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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 "";
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

这是此场景的切入点。

```
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();
};
```

前面的代码提供了必要的依赖关系并启动了场景。下一节包含示例的大部分内容。

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的以下主题。

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- 发布
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage
  - SetQueueAttributes

- Subscribe
- Unsubscribe

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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)
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 API Gateway 调用 Lambda 函数

以下代码示例展示了如何创建由 Amazon API Gateway 调用的 Amazon Lambda 函数。

Java

适用于 Java 的 SDK 2.x

演示如何使用 Lambda Java 运行时 API 创建 Amazon Lambda 函数。此示例调用不同的 Amazon 服务来执行特定的用例。此示例展示了如何创建通过 Amazon API Gateway 调用的 Lambda 函数，该函数扫描 Amazon DynamoDB 表获取工作周年纪念日，并使用 Amazon Simple Notification Service (Amazon SNS)向员工发送文本消息，祝贺他们的周年纪念日。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何使用 Lambda JavaScript 运行时 API 创建 Amazon Lambda 函数。此示例调用不同的 Amazon 服务来执行特定的用例。此示例展示了如何创建通过 Amazon API Gateway 调用的 Lambda 函数，该函数扫描 Amazon DynamoDB 表获取工作周年纪念日，并使用 Amazon Simple Notification Service (Amazon SNS)向员工发送文本消息，祝贺他们的周年纪念日。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 适用于 JavaScript 的 Amazon SDK v3 开发人员指南中找到。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

适用于 Python 的 SDK（Boto3）

此示例显示如何创建和使用以 Amazon Lambda 函数为目标的 Amazon API Gateway REST API。Lambda 处理程序演示了如何基于 HTTP 方法进行路由；如何从查询字符串、标头和正文中获取数据；以及如何返回 JSON 响应。

- 部署 Lambda 函数。
- 使用 API Gateway 创建 REST API
- 创建以 Lambda 函数为目标的 REST 资源。
- 授予允许 API Gateway 调用 Lambda 函数的权限。

- 使用请求软件包向 REST API 发送请求。
- 清理演示期间创建的所有资源。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用计划的事件调用 Lambda 函数

以下代码示例演示如何创建由 Amazon EventBridge 计划事件调用的 Amazon Lambda 函数。

Java

适用于 Java 的 SDK 2.x

演示如何创建调用函数的 Amazon EventBridge 计划事件。 Amazon Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。此示例调用不同的 Amazon 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- CloudWatch 日志
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何创建调用函数的 Amazon EventBridge 计划事件。 Amazon Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在此示例中，您将使用 Lambda 运行时 API 创建一个 Lambda 函数。 JavaScript 此示例调用不同的 Amazon 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 适用于 JavaScript 的 Amazon SDK v3 开发人员指南中找到。

本示例中使用的服务

- CloudWatch 日志
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

适用于 Python 的 SDK（Boto3）

此示例说明如何将 Amazon Lambda 函数注册为计划的 Amazon EventBridge 事件的目标。Lambda 处理程序将友好的消息和完整的事件数据写入 Amazon CloudWatch 日志，以供日后检索。

- 部署 Lambda 函数。
- 创建 EventBridge 计划事件并将 Lambda 函数设为目标。
- 授予允许 EventBridge 调用 Lambda 函数的权限。
- 打印来自 CloudWatch Logs 的最新数据以显示计划调用的结果。
- 清理演示期间创建的所有资源。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- CloudWatch 日志

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

亚马逊 SNS 的无服务器示例

以下代码示例展示了如何将 Amazon SNS 与配合使用。 Amazon SDKs

示例

- 通过 Amazon SNS 触发器调用 Lambda 函数

通过 Amazon SNS 触发器调用 Lambda 函数

以下代码示例演示了如何实现一个 Lambda 函数，该函数接收通过接收来自 SNS 主题的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 .NET 将 SNS 事件与 Lambda 结合使用。

```
// 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

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Go 将 SNS 事件与 Lambda 结合使用。

```
// 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

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

通过 Java 将 SNS 事件与 Lambda 结合使用。

```
// 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

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Lambda JavaScript 消费 SNS 事件。

```
// 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;
  }
}
```

使用 Lambda TypeScript 消费 SNS 事件。

```
// 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

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

通过 PHP 将 SNS 事件与 Lambda 结合使用。

```
// 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

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Python 将 SNS 事件与 Lambda 结合使用。

```
# 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

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 SNS 事件与 Lambda 结合使用。

```
# 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

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

通过 Rust 将 SNS 事件与 Lambda 结合使用。

```
// 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
}
```

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与软件开发工具包配合使用 Amazon。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon SNS 安全性

分 Amazon 分担责任模型适用于亚马逊简单通知服务中的数据保护。如本模型所述 Amazon ，负责保护运行所有内容的全球基础架构 Amazon Web Services 云。您负责维护对托管在此基础结构上的内容的控制。此内容包括您使用的 Amazon 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon Identity and Access Management (IAM) 设置个人用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 Amazon 建议使用 TLS 1.2 或更高版本。
- 使用设置 API 和用户活动日志 Amazon CloudTrail。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。
- 如果您在 Amazon 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。
- 消息数据保护
  - 消息数据保护是 Amazon SNS 的一项新的重要功能
  - 使用 MDP 扫描消息中的机密或敏感信息
  - 对流经主题的所有内容提供消息审计
  - 对发布到主题的消息和主题传输的消息提供内容访问控制

> **Important**
>
> 我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括您使用控制台、API 或使用亚马逊 SNS 或其他亚马逊 Web Services 时。 Amazon CLI Amazon SDKs您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。当您向外部服务器提供 URL 时，强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

Amazon SNS 数据加密

数据保护指在数据传输（发往和离开 Amazon SNS 时）和处于静态（存储在 Amazon SNS 数据中心的磁盘上时）期间保护数据。您可以使用安全套接字层 (SSL) 或客户端加密保护传输中的数据。默认情况下，Amazon SNS 使用磁盘加密来存储消息和文件。您可以在将消息保存到数据中心的加密文件系统之前请求 Amazon SNS 对其进行加密，从而保护静态数据。Amazon SNS 建议使用 SSE 来优化数据加密。

使用服务器端加密保护 Amazon SNS 数据安全

服务器端加密 (SSE) 允许您使用在 () 中管理的密钥保护 Amazon SNS 主题中的消息内容，从而将敏感数据存储在加密主题 Amazon Key Management Service 中Amazon KMS。

一旦 Amazon SNS 收到消息，SSE 就会对消息进行加密。这些消息以加密形式存储，并且仅当消息发送时，才会对消息进行解密。

- 有关使用 Amazon Web Services Management Console 或 适用于 Java 的 Amazon SDK （通过使用`CreateTopic`和 `SetTopicAttributes` API 操作设置`KmsMasterKeyId`属性）管理 SSE 的信息，请参阅使用服务器端加密设置 Amazon SNS 主题加密。
- 有关使用 Amazon CloudFormation （通过使用`AWS::SNS::Topic`资源设置`KmsMasterKeyId`属性）创建加密主题的信息，请参阅《Amazon CloudFormation 用户指南》。

> **Important**
>
> 针对启用了 SSE 的主题的所有请求都必须使用 HTTPS 和签名版本 4。
> 有关其他服务与加密主题的兼容性的信息，请参阅服务文档。
> Amazon SNS 仅支持对称加密 KMS 密钥。您不能使用任何其他类型的 KMS 密钥来加密您的服务资源。有关确定 KMS 密钥是否为对称加密密钥的帮助，请参阅识别非对称 KMS 密钥。

Amazon KMS 将安全、高度可用的硬件和软件相结合，提供可扩展到云端的密钥管理系统。当您将 Amazon SNS 与一起使用时 Amazon KMS，加密您的消息数据的数据密钥也会被加密并与它们保护的数据一起存储。

使用 Amazon KMS具有以下好处：

- 您可以自行创建和管理 Amazon KMS key。

- 您也可以将 Amazon托管的 KMS 密钥用于 Amazon SNS，这些密钥对于每个账户和地区都是唯一的。
- Amazon KMS 安全标准可以帮助您满足与加密相关的合规性要求。

有关更多信息，请参阅什么是 Amazon Key Management Service？ 在《Amazon Key Management Service 开发人员指南》中。

加密范围

SSE 将对 Amazon SNS 主题中的消息正文进行加密。

SSE 不对以下各项进行加密：

- 主题元数据（主题名称和属性）
- 消息元数据（主题、消息 ID、时间戳和属性）
- 数据保护策略
- 每个主题的指标数

> **Note**
>
> - 仅在启用主题加密后发送消息时对其进行加密。Amazon SNS 不会加密积压的消息。
> - 任何加密的消息将保持加密状态，即使已禁用其主题的加密。

关键术语

以下关键术语有助于您更好地了解 SSE 的功能。有关详细说明，请参阅 Amazon Simple Notification Service API 参考。

数据密钥

数据加密密钥 (DEK) 负责加密 Amazon SNS 消息的内容。

有关更多信息，请参阅 Amazon Key Management Service 开发人员指南中的数据密钥和 Amazon Encryption SDK 开发人员指南中的信封加密。

Amazon KMS key ID

您的账户或其他账户中的别名、别名 ARN、密钥 ID 或密钥 ARN Amazon KMS key，或者自定义 Amazon KMS的 ARN。虽然 Amazon KMS 适用于 Amazon SNS 的 Amazon 托管别名始终是`alias/aws/sns`，但自定义的别名 Amazon KMS 可以是。`alias/MyAlias`您可以利用这些 Amazon KMS 密钥保护 Amazon SNS 主题中的消息。

> **Note**
>
> 记住以下内容：
>
> - 首次使用为主题指定适用于 Amazon SNS 的 Amazon 托管 KMS 时， Amazon KMS 会为亚马逊 SNS 创建 Amazon 托管 KMS。 Amazon Web Services Management Console
> - 或者，在启用 SSE 的情况下首次对主题使用`Publish`操作时， Amazon KMS 会为 Amazon SNS 创建 Amazon 托管 KMS。

您可以创建 Amazon KMS 密钥，定义控制 Amazon KMS 密钥使用方式的策略，并 Amazon KMS 使用 Amazon KMS 控制台的Amazon KMS keys部分或`CreateKey` Amazon KMS 操作来审计使用情况。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的 Amazon KMS keys和创建密钥。有关 Amazon KMS 标识符的更多示例，请参阅 Amazon Key Management Service API 参考KeyId中的。有关查找 Amazon KMS 标识符的信息，请参阅《Amazon Key Management Service 开发者指南》中的 “查找密钥 ID 和 ARN”。

> **Important**
>
> 使用需要支付额外费用 Amazon KMS。有关更多信息，请参阅 估算成本 Amazon KMS 和 Amazon Key Management Service 定价。

管理 Amazon SNS 加密密钥和成本

以下部分提供了有关使用 Amazon Key Management Service (Amazon KMS) 中托管的密钥的信息。

> **Note**
>
> Amazon SNS 仅支持对称加密 KMS 密钥。您不能使用任何其他类型的 KMS 密钥来加密您的服务资源。有关确定 KMS 密钥是否为对称加密密钥的帮助，请参阅识别非对称 KMS 密钥。

估算成本 Amazon KMS

为了预测成本并更好地了解您的 Amazon 账单，您可能需要知道 Amazon SNS 使用您的账单的频率。Amazon KMS key

> **Note**
>
> 尽管以下公式可让您很好地了解预计成本，但由于 Amazon SNS 的分布式特性，实际成本可能更高。

要计算每个主题的 API 请求数 (R) 数，请使用以下公式：

```
R = B / D * (2 * P)
```

B 是账单周期（以秒为单位）。

D 是数据密钥重用周期（以秒为单位，Amazon SNS 重用数据密钥长达 5 分钟）。

P 是发送到 Amazon SNS 主题的发布委托人的数量。

以下是一些示例计算。有关准确的定价信息，请参阅 Amazon Key Management Service 定价。

示例 1：计算 1 个发布者和 1 个主题的 Amazon KMS API 调用次数

此示例假定：

- 账单周期为 1 月 1 日 - 31 日（2678400 秒）。
- 数据密钥重用周期为 5 分钟（300 秒）。
- 提供了 1 个主题。
- 提供了 1 个发布委托人。

```
2,678,400 / 300 * (2 * 1) = 17,856
```

示例 2：计算多名发布者和 2 个主题的 Amazon KMS API 调用数

此示例假定：

- 账单周期为 2 月 1 日 - 28 日（2419200 秒）。

- 数据密钥重用周期为 5 分钟（300 秒）。
- 提供了 2 个主题。
- 第一个主题具有 3 个发布委托人。
- 第二个主题具有 5 个发布委托人。

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

配置 Amazon KMS 权限

在使用 SSE 之前，必须将 Amazon KMS key 策略配置为允许对主题进行加密以及对消息进行加密和解密。有关 Amazon KMS 权限的示例和更多信息，请参阅 Amazon Key Management Service 开发人员指南中的 Amazon KMS API 权限：操作和资源参考。有关如何使用服务器端加密设置 Amazon SNS 主题的详细信息，请参阅其他信息。

> **Note**
>
> 您也可以使用 IAM policy 来管理对称加密 KMS 密钥的权限。有关更多信息，请参阅将 IAM 策略与一起使用 Amazon KMS。
>
> 虽然您可以配置向 Amazon SNS 发送和接收的全局权限，但 Amazon KMS 需要在 IAM 策略`Resource`部分 KMSs 中明确命名特定区域的完整 ARN。

您还必须确保的密钥策略 Amazon KMS key 允许必要的权限。为此，请将在 Amazon SNS 中创建和使用加密消息的委托人指定为 CMK 密钥策略中的用户。

或者，您可以在分配给在 Amazon SNS 中发布和订阅以接收加密消息的委托人的 IAM 策略中指定所需的 Amazon KMS 操作和 KMS ARN。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的管理对 Amazon KMS的访问。

如果为您的 Amazon SNS 主题选择了客户自主管理型密钥，并且您正在使用别名通过 IAM policy 或带有条件密钥 `kms:ResourceAliases` 的 KMS 密钥策略来控制 KMS 密钥的访问权限，请确保所选的客户自主管理型密钥也关联了别名。有关使用别名控制 KMS 密钥的访问权限的更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的使用别名控制 KMS 密钥的访问权限。

允许用户使用 SSE 将消息发送到主题

发布者必须具有 Amazon KMS key的 `kms:GenerateDataKey*` 和 `kms:Decrypt` 权限。

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

启用来自 Amazon 服务的事件源和加密主题之间的兼容性

一些 Amazon 服务会向 Amazon SNS 主题发布事件。要允许这些事件源使用加密主题，您必须执行以下步骤。

1. 使用客户自主管理型密钥。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的创建密钥。
2. 要允许 Amazon 服务拥有`kms:GenerateDataKey*`和`kms:Decrypt`权限，请在 KMS 策略中添加以下语句。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

事件源	服务主体
Amazon CloudWatch	`cloudwatch.amazonaws.com`
亚马逊 CloudWatch 活动	`events.amazonaws.com`
Amazon CodeCommit	`codecommit.amazonaws.com`
Amazon Database Migration Service	`dms.amazonaws.com`
Amazon Directory Service	`ds.amazonaws.com`
Amazon DynamoDB	`dynamodb.amazonaws.com`
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`
Amazon Simple Email Service	`ses.amazonaws.com`
Amazon Simple Storage Service	`s3.amazonaws.com`
Amazon Snowball Edge	`importexport.amazonaws.com`
Amazon Systems Manager 事件管理器	Amazon Systems Manager 事件管理器包含两个服务原则：`ssm-incidents.amazonaws.com` ；`ssm-contacts.amazonaws.com`

> **Note**
>
> 某些 Amazon SNS 事件源要求您在策略中提供 IAM 角色（而不是服务委托人）：Amazon KMS key
>
> - Amazon A EC2 uto Scaling

- Amazon Elastic Transcoder
- Amazon CodePipeline
- Amazon Config
- Amazon Elastic Beanstalk
- Amazon IoT
- EC2 Image Builder

3. 将 `aws:SourceAccount` 和 `aws:SourceArn` 条件键添加到 KMS 资源策略，以进一步保护 KMS 密钥免受混淆代理攻击。有关每种案例的具体详细信息，请参阅上述特定于服务的文档列表。

> **Important**
>
> EventBridge-to-encrypted主题不支持在`aws:SourceAccount` Amazon KMS 策略中添加`aws:SourceArn`、和`aws:SourceOrgID`。

```
{
  "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. 使用 KMS 为您的主题启用 SSE。

5. 向事件源提供加密主题的 ARN。

Amazon KMS 错误

当您使用 Amazon SNS 和时 Amazon KMS，可能会遇到错误。以下列表描述了这些错误和可能的故障排除解决方案。

KMSAccessDeniedException

密文引用了不存在的或您无权访问的密钥。

HTTP 状态代码：400

KMSDisabled例外

由于指定的 KMS 未启用，请求被拒绝。

HTTP 状态代码：400

KMSInvalidStateException

由于指定资源的状态对此请求无效，请求被拒绝。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的 Amazon KMS keys的密钥状态。

HTTP 状态代码：400

KMSNotFoundException

由于找不到指定的实体或资源，请求被拒绝。

HTTP 状态代码：400

KMSOptInRequired

Amazon 访问密钥 ID 需要订阅该服务。

HTTP 状态代码：403

KMSThrottling例外

由于请求限制而导致请求被拒绝。有关节流的更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的限额。

HTTP 状态代码：400

使用服务器端加密设置 Amazon SNS 主题加密

Amazon SNS 支持服务器端加密 (SSE)，以使用 Amazon Key Management Service () 保护消息内容。Amazon KMS按照以下说明使用亚马逊 SNS 控制台或 CDK 启用 SSE。

选项 1：使用启用加密 Amazon Web Services Management Console

1. 登录 Amazon SNS 控制台。
2. 导航到主题页面，选择您的主题，然后选择编辑。
3. 展开加密部分并执行以下操作：
   - 将加密切换为启用。
   - 选择Amazon 托管的 SNS 钥匙 (alias/aws/sns) 作为加密密钥。此选项默认处于选中状态。
4. 选择 Save changes（保存更改）。

> **Note**
>
> - 如果 Amazon 托管式密钥 尚不存在，则会自动创建。
> - 如果您看不到密钥或权限不足，请向管理员询问`kms:ListAliases`和`kms:DescribeKey`。

选项 2：使用启用加密 Amazon CDK

要使用 Amazon 托管 SNS 键入你的 CDK 应用程序，添加以下片段：

```
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();
    }
}
```

其他信息

- 自定义 KMS 密钥-如果需要，您可以指定自定义密钥。在 Amazon SNS 控制台中，从列表中选择您的自定义 KMS 密钥或输入 ARN。
- 自定义 KMS 密钥的权限-如果使用自定义 KMS 密钥，请在密钥策略中包含以下内容，以允许 Amazon SNS 加密和解密消息：

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

对使用者的影响

启用 SSE 不会改变订阅者使用消息的方式。 Amazon 透明地管理加密和解密。消息在静态状态下保持加密状态，并在发送给订阅者之前自动解密。为了获得最佳安全性， Amazon 建议为所有端点启用 HTTPS，以确保消息的安全传输。

使用加密的 Amazon SQS 队列订阅设置 Amazon SNS 主题加密

您可以为主题启用服务器端加密 (SSE) 以保护其数据。要允许 Amazon SNS 将消息发送到加密的 Amazon SQS 队列，与 Amazon SQS 队列关联的客户自主管理型密钥必须具有一个策略语句，该语句向 Amazon SNS 服务主体授予对 Amazon KMS API 操作 `GenerateDataKey` 和 `Decrypt` 的访问权限。有关使用 SSE 的更多信息，请参阅使用服务器端加密保护 Amazon SNS 数据安全。

本主题介绍如何使用加密的 Amazon SQS 队列订阅为亚马逊 SNS 主题启用 SSE。 Amazon Web Services Management Console

步骤 1：创建自定义 KMS 密钥

1. 通过至少具有 `AWSKeyManagementServicePowerUser` 策略的用户登录 Amazon KMS 控制台。
2. 选择 Create a key (创建密钥)。
3. 要创建对称加密 KMS 密钥，请为 Key type（密钥类型）选择 Symmetric（对称）。

   有关如何在 Amazon KMS 控制台中创建非对称 KMS 密钥的信息，请参阅创建非对称 KMS 密钥（控制台）。
4. 在 Key usage（密钥用法）中，已为您选择了 Encrypt and decrypt（加密和解密）选项。

   有关如何创建用于生成和验证 MAC 代码的 KMS 密钥的信息，请参阅创建 HMAC KMS 密钥。

   有关高级选项的更多信息，请参阅专用密钥。
5. 选择下一步。
6. 键入 KMS 密钥的别名。别名名称不能以 **`aws/`** 开头。该**`aws/`**前缀由 Amazon Web Services 保留，用于 Amazon 托管式密钥 在您的账户中表示。

   > **Note**
   > 添加、删除或更新别名可以允许或拒绝对 KMS 密钥的权限。有关详细信息，请参阅适用于 Amazon KMS的 ABAC 和使用别名控制 KMS 密钥的访问权限。

   别名是一个显示名称，您可以使用它来标识 KMS 密钥。我们建议您选择一个别名，用来指示您计划保护的数据类型或计划与 KMS 密钥搭配使用的应用程序。

在 Amazon Web Services Management Console中创建 KMS 密钥时需要别名。当您使用CreateKey操作时，它们是可选的。

7. （可选）为 KMS 密钥键入描述。

   现在，除非密钥状态为 `Pending Deletion` 或 `Pending Replica Deletion`，否则您可以随时添加描述或更新描述。要添加、更改或删除现有客户托管密钥的描述，请在中编辑描述 Amazon Web Services Management Console 或使用UpdateKeyDescription操作。

8. (可选) 键入标签键和一个可选标签值。要向 KMS 密钥添加多个标签，请选择 Add tag（添加标签）。

   > **Note**
   >
   > 标记或取消标记 KMS 密钥可以允许或拒绝对 KMS 密钥的权限。有关详细信息，请参阅适用于 Amazon KMS的 ABAC 和使用标签控制 KMS 密钥的访问权限。

   向 Amazon 资源添加标签时， Amazon 会生成一份成本分配报告，其中包含按标签汇总的使用量和成本。标签还可以用来控制对 KMS 密钥的访问。有关标记 KMS 密钥的信息，请参阅标记密钥和适用于 Amazon KMS的 ABAC。

9. 选择下一步。

10. 选择可管理 KMS 密钥的 IAM 用户和角色。

    > **Note**
    >
    > 此密钥策略赋予了对此 KMS 密钥的 Amazon Web Services 账户 完全控制权。此控制权允许账户管理员使用 IAM policy 授予其他主体管理 KMS 密钥的权限。有关详细信息，请参阅原定设置密钥策略。
    >
    > IAM 最佳实践不鼓励使用具有长期凭证的 IAM 用户。而应尽可能使用提供临时凭证的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

11. （可选）要阻止选定 IAM 用户和角色删除此 KMS 密钥，请在页面底部的 Key deletion（密钥删除）部分中，清除 Allow key administrators to delete this key（允许密钥管理员删除此密钥）复选框。

12. 选择下一步。

13. 选择可在加密操作中使用密钥的 IAM 用户和角色。选择下一步。

14. 在 Review and edit key policy (审核和编辑密钥策略) 页面上，向密钥策略添加以下语句，然后选择 Finish (完成)。

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

新的客户自主管理型密钥将显示在密钥列表中。

步骤 2：创建加密的 Amazon SNS 主题

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。
3. 选择创建主题。
4. 在 Create new topic (创建新主题) 页面上，对于 Name (名称)，输入主题名称（例如，`MyEncryptedTopic`），然后选择 Create topic (创建主题)。
5. 展开加密部分并执行以下操作：

   a. 选择 Enable server-side encryption (启用服务器端加密)。

   b. 指定客户自主管理型密钥。有关更多信息，请参阅 关键术语。

   对于每种客户自主管理型密钥类型，将显示描述、账户和客户自主管理型密钥 ARN。

   > **Important**
   > 如果您不是客户自主管理型密钥的拥有者，或者您登录的账户没有 `kms:ListAliases` 和 `kms:DescribeKey` 权限，则无法在 Amazon SNS 控制台上查看有关客户自主管理型密钥的信息。

要求客户自主管理型密钥的拥有者授予您这些权限。有关更多信息，请参阅 Amazon Key Management Service 开发人员指南中的 Amazon KMS API 权限：操作和资源参考。

c. 对于客户托管密钥 MyCustomKey，请选择您之前创建的密钥，然后选择启用服务器端加密。

6. 选择 Save changes（保存更改）。

   您的主题已启用 SSE 并显示MyTopic页面。

   主题的加密状态、 Amazon 账户、客户自主管理型密钥、客户自主管理型密钥 ARN 和描述将显示在加密选项卡上。

您的新的加密主题将显示在主题列表中。

步骤 3：创建并订阅加密的 Amazon SQS 队列

1. 登录 Amazon SQS 控制台。
2. 选择 Create New Queue（创建队列）。
3. 在 Create New Queue (创建新队列) 页面上，执行以下操作：

   a. 输入 Queue Name (队列名称)（例如，`MyEncryptedQueue1`）。

   b. 选择 Standard Queue (标准队列)，然后选择 Configure Queue (配置队列)。

   c. 选择 Use SSE (使用 SSE)。

   d. 对于 Amazon KMS keyMyCustomKey，选择您之前创建的队列，然后选择创建队列。

4. 重复该过程以创建第二个队列（例如，名为 `MyEncryptedQueue2`）。

   您的新的加密队列将显示在队列列表中。

5. 在 Amazon SQS 控制台上，选择 `MyEncryptedQueue1` 和 `MyEncryptedQueue2`，然后选择 Queue Actions（队列操作）、Subscribe Queues to SNS Topic（订阅队列至 SNS 主题）。
6. 在 “订阅主题” 对话框中，选择 “选择主题” MyEncryptedTopic，然后选择 “订阅”。

   您的加密队列对加密主题的订阅将显示在 Topic Subscription Result (主题订阅结果) 对话框中。

7. 选择确定。

步骤 4：向加密的主题发布消息

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择主题。
3. 从主题列表中选择，MyEncryptedTopic然后选择发布消息。
4. 在 Publish a message (发布消息) 页面上，执行以下操作：
   a. （可选）在 Message details (消息详细信息) 部分中，输入 Subject (主题)（例如：`Testing message publishing`）。
   b. 在 Message body (消息正文) 部分中，输入消息正文（例如，`My message body is encrypted at rest.`）。
   c. 选择发布消息。

您的消息将发布到订阅的加密队列。

步骤 5：验证消息传输

1. 登录 Amazon SQS 控制台。
2. 从队列列表中选择 MyEncryptedQueue1，然后选择发送和接收消息。
3. 在 “在 MyEncryptedQueue1 页中发送和接收消息” 上，选择 “轮询留言”。

   此时将显示您之前发送的消息。
4. 选择 More Details (更多详细信息) 以查看您的消息。
5. 完成后，选择 Close (关闭)。
6. 对 MyEncryptedQueue2 重复该过程。

使用 VPC 端点保护 Amazon SNS 流量

Amazon SNS 的 Amazon Virtual Private Cloud (Amazon VPC) 终端节点是 VPC 内的逻辑实体，仅允许连接到 Amazon SNS。VPC 将请求路由到 Amazon SNS 并将响应路由回 VPC。以下部分提供有关使用 VPC 端点和创建 VPC 端点策略的信息。

如果您使用亚马逊虚拟私有云（亚马逊 VPC）托管 Amazon 资源，则可以在您的 VPC 和亚马逊 SNS 之间建立私有连接。利用此连接，您可以将消息发布到 Amazon SNS 主题而无需通过公共 Internet 发送消息。

Amazon VPC 是一项 Amazon 服务，可用于在您定义的虚拟网络中启动 Amazon 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将您的 VPC 连接到 Amazon SNS，请定义一个接口 VPC 终端节点。这种类型的终端节点使您能够将 VPC 连接到 Amazon 服务。该终端节点提供了到 Amazon SNS 的可靠、可扩展的连接，无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 Amazon VPC 用户指南中的 Amazon Web Services 服务 使用接口 VPC 终端节点访问和。

此部分中的信息适用于 Amazon VPC 的用户。要了解更多信息并开始创建 VPC，请参阅 Amazon VPC 用户指南中的规划您的 VPC。

> **Note**
>
> VPC 终端节点不允许您将 Amazon SNS 主题订阅到私有 IP 地址。

为 Amazon SNS 创建 Amazon VPC 终端节点

要将消息从 Amazon VPC 发布到您的 Amazon SNS 主题，请创建一个接口 VPC 终端节点。然后，您可以将消息发布到主题，同时保留您利用该 VPC 管理的网络内的流量。

使用以下信息创建终端节点并测试 VPC 与 Amazon SNS 之间的连接。或者，有关可帮助您从头开始的演练，请参阅从 Amazon VPC 发布 Amazon SNS 消息。

创建终端节点

您可以使用 Amazon Web Services Management Console、、、软件开发工具包、亚马逊 SNS API 或在您的 VPC 中创建亚马逊 Amazon SNS 终端节点。 Amazon CLI Amazon CloudFormation

有关使用 Amazon VPC 控制台或 Amazon CLI创建和配置端点的信息，请参阅 Amazon VPC 用户指南中的创建接口端点。

> **Important**
>
> 您只能将 Amazon Virtual Private Cloud 与 HTTPS Amazon SNS 端点一起使用。
> 在创建终端节点时，请将 Amazon SNS 指定为您希望 VPC 连接到的服务。在 Amazon VPC 控制台中，服务名称因区域而异。例如，如果您选择美国东部（弗吉尼亚北部），则服务名称为 com.amazonaws.us-east-1.sns。
> 当您将 Amazon SNS 配置为从 Amazon VPC 发送消息时，必须启用私有 DNS 并按格式 `sns.us-east-2.amazonaws.com` 指定端点。

私有 DNS 不支持传统端点，例如 `queue.amazonaws.com` 或 `us-east-2.queue.amazonaws.com`。

有关使用创建和配置终端节点的信息 Amazon CloudFormation，请参阅《Amazon CloudFormation 用户指南》中的`AWS::EC2::VPCEndpoint`资源。

测试您的 VPC 和 Amazon SNS 之间的连接

在为 Amazon SNS 创建终端节点后，您可以将消息从您的 VPC 发布到 Amazon SNS 主题。要测试此连接，请执行以下操作：

1. 连接到驻留在您的 VPC 中的亚马逊 EC2 实例。有关连接的信息，请参阅亚马逊 EC2 文档中的连接到您的 Linux 实例或连接到您的 Windows 实例。

   例如，要使用 SSH 客户端连接到 Linux 实例，请从终端运行以下命令：

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   其中：

   - ec2-key-pair.pem 是包含您在创建实例时亚马逊 EC2 提供的密钥对的文件。
   - instance-hostname 是实例的公有主机名。要在 Amazon EC2 控制台中获取主机名，请执行以下操作：选择实例，选择您的实例，然后找到公有 DNS 的值。
2. 在您的实例中，将 Amazon SNS `publish` 命令与 Amazon CLI结合使用。您可以使用以下命令将简单的消息发送到主题：

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   其中：

   - aw s-region 是 Amazon 该主题所在的区域。
   - sns-topic-arn是该主题的亚马逊资源名称 (ARN)。从 Amazon SNS 控制台获取该 ARN：选择 Topics（主题），查找您的主题，然后找到 ARN 列中对应的值。

   如果 Amazon SNS 已成功收到消息，终端会打印一个消息 ID，如下所示：

   ```
   {
   ```

```
    "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

为 Amazon SNS 创建 Amazon VPC 终端节点策略

您可以为 Amazon SNS 的 Amazon VPC 终端节点创建一个策略，在该策略中指定以下内容：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问权限。

以下示例 VPC 终端节点策略指定允许 IAM 用户 `MyUser` 将内容发布到 Amazon SNS 主题 `MyTopic`。

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

以下各项将被拒绝：

- 其他 Amazon SNS API 操作，例如 `sns:Subscribe` 和 `sns:Unsubscribe`。
- 其他尝试使用该 VPC 终端节点的 IAM 用户和规则。
- 发布到其他 Amazon SNS 主题的 `MyUser`。

> **Note**
>
> IAM 用户仍然可以从 VPC 外部使用其他 Amazon SNS API 操作。

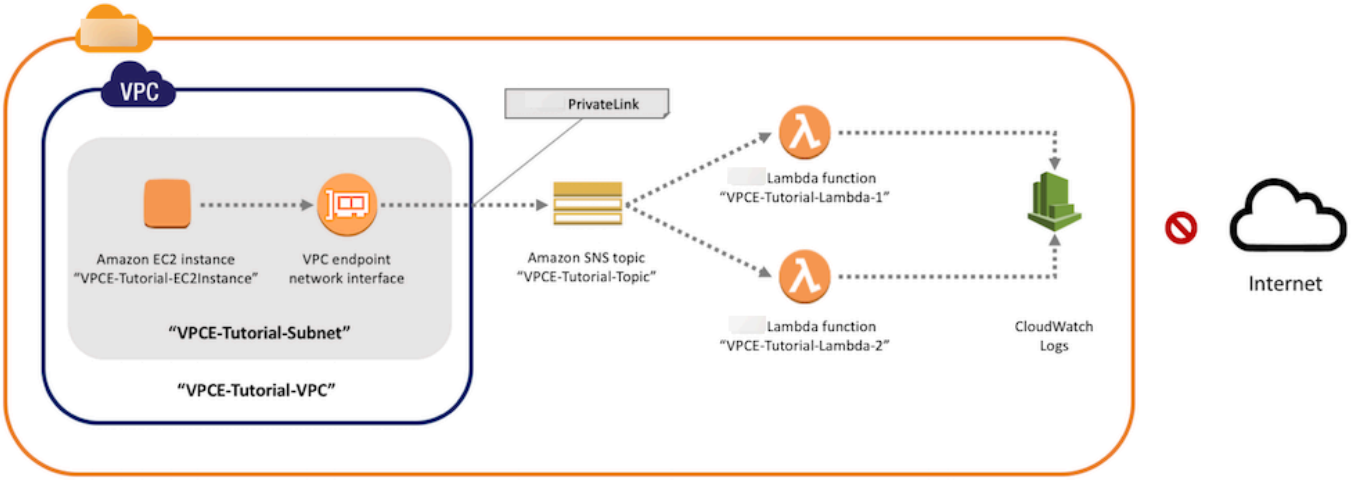
从 Amazon VPC 发布 Amazon SNS 消息

本节介绍如何发布到 Amazon SNS 主题，同时保护消息在私有网络中的安全性。您发布来自托管在亚马逊虚拟私有云 (Amazon VPC) 中的亚马逊 EC2 实例的消息。消息保留在 Amazon 网络中，无需通过公共互联网传输。通过从 VPC 私下发布消息，您可以提高应用程序与 Amazon SNS 之间的流量的安全性。这种安全性在您发布有关客户的个人身份信息（PII）或当您的应用程序受市场规定约束时至关重要。例如，如果您有一个必须符合健康保险可携性与责任法 (HIPAA) 的医疗保健系统或有一个必须符合支付卡行业数据安全标准 (PCI DSS) 的金融系统，则私下发布会很有用。

常见步骤如下：

- 使用 Amazon CloudFormation 模板在您的中自动创建临时专用网络 Amazon Web Services 账户。
- 创建一个用于连接 VPC 与 Amazon SNS 的 VPC 终端节点。
- 登录亚马逊 EC2 实例，私下向亚马逊 SNS 主题发布消息。
- 验证消息是否已成功发送。
- 删除您在此过程中创建的资源，这样它们就不会保留在您的资源中 Amazon Web Services 账户。

下图描述了您在完成这些步骤时在 Amazon 账户中创建的专用网络：



该网络由包含一个 Amazon EC2 实例的 VPC 组成。该实例通过接口 VPC 终端节点 连接到 Amazon SNS。这种类型的端点连接到由提供支持的服务 Amazon PrivateLink。建立此连接后，即使网络已与公共互联网断开连接，您也可以登录到 Amazon EC2 实例并向 Amazon SNS 主题发布消息。该主题将其收到的消息分散到两个订阅 Amazon Lambda 函数。这些函数将它们收到的消息记录在 Amazon CloudWatch 日志中。

完成这些步骤大约需要 20 分钟。

主题

- 开始前的准备工作
- 第 1 步：创建亚马逊 EC2 密钥对
- 步骤 2：创建 Amazon 资源
- 步骤 3：确认您的 Amazon EC2 实例无法访问互联网
- 步骤 4：为 Amazon SNS 创建 Amazon VPC 终端节点
- 步骤 5：向 Amazon SNS 主题发布消息
- 步骤 6：验证您的消息传输
- 步骤 7：清除
- 相关资源

开始前的准备工作

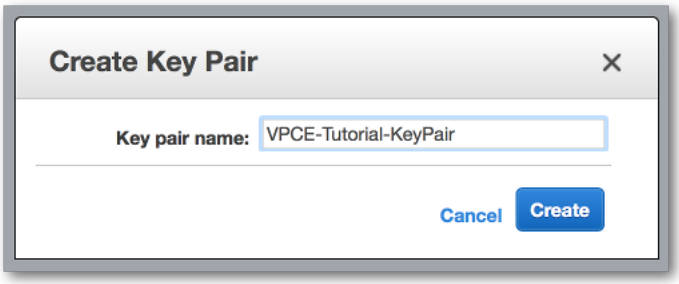
在开始之前，您需要 Amazon Web Services (Amazon) 账户。当您注册时，您的账户会自动注册使用中的所有服务 Amazon，包括亚马逊 SNS 和 Amazon VPC。如果您尚未创建账户，请前往 https://www.amazonaws.cn/，然后选择 Create a Free Account（创建免费账户）。

第 1 步：创建亚马逊 EC2 密钥对

密钥对用于登录亚马逊 EC2 实例。它包含一个用于加密您的登录信息的公有密钥和一个用于解密该信息的私有密钥。创建密钥对时，下载该私有密钥的副本。稍后，您可以使用 key pair 登录到 Amazon EC2 实例。要登录，请指定密钥对的名称，然后提供密钥对。

创建密钥对

1. 登录 Amazon Web Services Management Console 并打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航菜单中，找到 Network & Security (网络与安全) 部分。然后，选择 Key Pairs (密钥对)。
3. 选择创建密钥对。
4. 在创建密钥对窗口中，在密钥对名称中，键入 **`VPCE-Tutorial-KeyPair`**。然后选择 Create。



5. 您的浏览器会自动下载私有密钥文件。将它保存至安全位置。Amazon EC2 为该文件提供的扩展名为 `.pem`。
6. (可选) 如果您在 Mac 或 Linux 计算机上使用 SSH 客户端连接到您的实例，请使用 `chmod` 命令设置您的私有密钥文件的权限，以确保只有您可以读取该文件。

   a. 打开终端并导航到包含该私有密钥的目录：

   ```
   $ cd /filepath_to_private_key/
   ```

   b. 使用以下命令设置权限：

   ```
   $ chmod 400 VPCE-Tutorial-KeyPair.pem
   ```

步骤 2：创建 Amazon 资源

要设置基础架构，请使用 Amazon CloudFormation 模板。模板是用作构建 Amazon 资源（例如 Amazon EC2 实例和 Amazon SNS 主题）的蓝图的文件。此过程的模板已提供 GitHub 给您下载。

您可以向提供模板 Amazon CloudFormation Amazon CloudFormation ，并将所需的资源配置为堆栈 Amazon Web Services 账户。堆栈是可作为单个单元管理的一系列资源。完成这些步骤后，您可以使用 Amazon CloudFormation 一次性删除堆栈中的所有资源。除非您愿意 Amazon Web Services 账户，否则这些资源不会保留在您的中。

此过程的堆栈包括以下资源：

- VPC 和关联的网络资源，包括子网、安全组、Internet 网关和路由表。
- 在 VPC 的子网中启动的 Amazon EC2 实例。
- Amazon SNS 主题。
- 两个 Amazon Lambda 功能。这些函数接收发布到 Amazon SNS 主题的消息，并在日志中 CloudWatch 记录事件。

- Amazon CloudWatch 指标和日志。
- 一个允许亚马逊 EC2 实例使用亚马逊 SNS 的 IAM 角色和一个允许 Lambda 函数写入日志的 IAM 角色。 CloudWatch

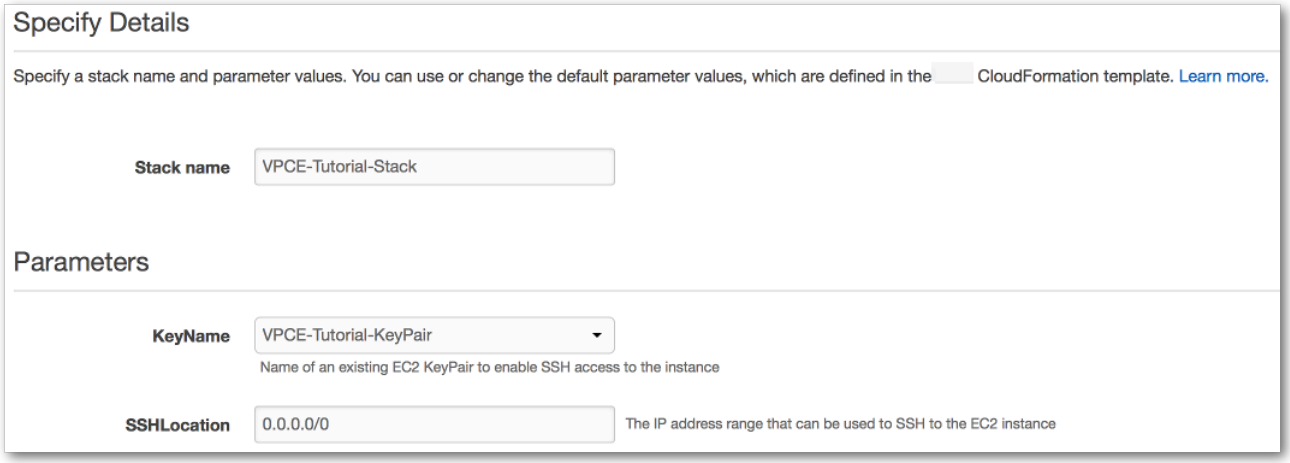
创建 Amazon 资源

1. 从 GitHub 网站下载模板文件。
2. 登录 Amazon CloudFormation 控制台。
3. 选择创建堆栈。
4. 在选择模板页面上，依次选择将模板上传到 Amazon S3、您的文件和下一步。
5. 在 Specify Details (指定详细信息) 页面上，指定堆栈和密钥名称：

   a. 对于 Stack name，键入 **`VPCE-Tutorial-Stack`**。

   b. 对于 KeyName，选择 VPCE 教程-。KeyPair

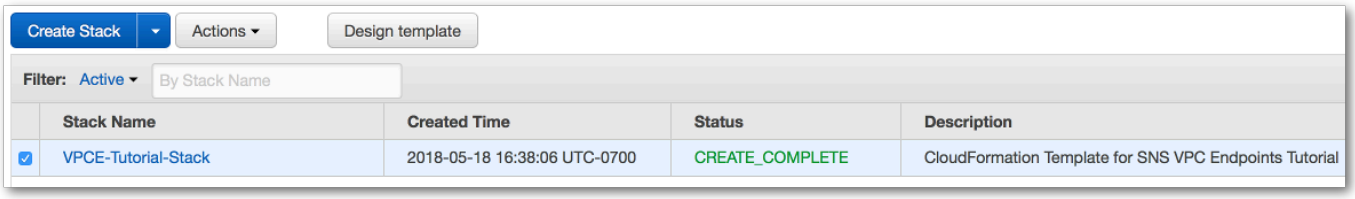
   c. 对于 SSHLocation，保留默认值**`0.0.0.0/0`**。



   d. 选择下一步。

6. 在 Options (选项) 页面上，保留所有默认值，然后选择 Next (下一步)。
7. 在 Review (审核) 页面上，验证堆栈详细信息。
8. 在 “权能” 下，确认 Amazon CloudFormation 可能会使用自定义名称创建 IAM 资源。
9. 选择创建。

   Amazon CloudFormation 控制台打开堆栈页面。的 VPCE-Tutorial-Stack状态为 CREATE_IN_PROGRESS。在创建过程完成后的几分钟内，该状态将变为 CREATE_COMPLETE。



> **Tip**
>
> 选择刷新按钮以查看最新堆栈状态。

步骤 3：确认您的 Amazon EC2 实例无法访问互联网

上一步中在您的 VPC 中启动的 Amazon EC2 实例无法访问互联网。它不允许出站流量，而且无法将发布到 Amazon SNS。通过登录该实例验证此项。然后，尝试连接到公共终端节点，并尝试向 Amazon SNS 发布消息。

此时，发布操作尝试失败。在后面的步骤中，在您为 Amazon SNS 创建 VPC 终端节点后，您的发布尝试成功。

连接到您的 Amazon EC2 实例

1. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航菜单中，找到 Instances (实例) 部分。然后，选择 Instances (实例)。
3. 在实例列表中，选择 VPCE-。Tutorial-EC2Instance
4. 复制 “公用 DNS” 列中提供的主机名。

5. 打开终端。在包含密钥对的目录中，使用以下命令连接到实例，其中*instance-hostname*是您从 Amazon EC2 控制台复制的主机名：

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

验证实例是否缺少 Internet 连接

- 在您的终端中，尝试连接到任何公共终端节点，如 amazon.com：

```
$ ping amazon.com
```

由于连接尝试失败，您可以随时进行取消（Windows 上按 Ctrl + C 或 macOS 上按 Command + C）。

验证实例是否缺少到 Amazon SNS 的连接

1. 登录 Amazon SNS 控制台。
2. 在左侧导航菜单中，选择 Topics (主题)。
3. 在 Topics (主题) 页面上，复制主题 VPCE-Tutorial-Topic 的 Amazon Resource Name (ARN)。
4. 在您的终端中，尝试向该主题发布消息：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

由于发布尝试失败，您可以随时进行取消。

步骤 4：为 Amazon SNS 创建 Amazon VPC 终端节点

要将您的 VPC 连接到 Amazon SNS，请定义一个接口 VPC 终端节点。添加终端节点后，您可以登录您的 VPC 中的亚马逊 EC2 实例，然后您可以从那里使用 Amazon SNS API。您可以将消息发布到该主题，而且私下发布消息。它们停留在 Amazon 网络中，并且不会在公共互联网上旅行。

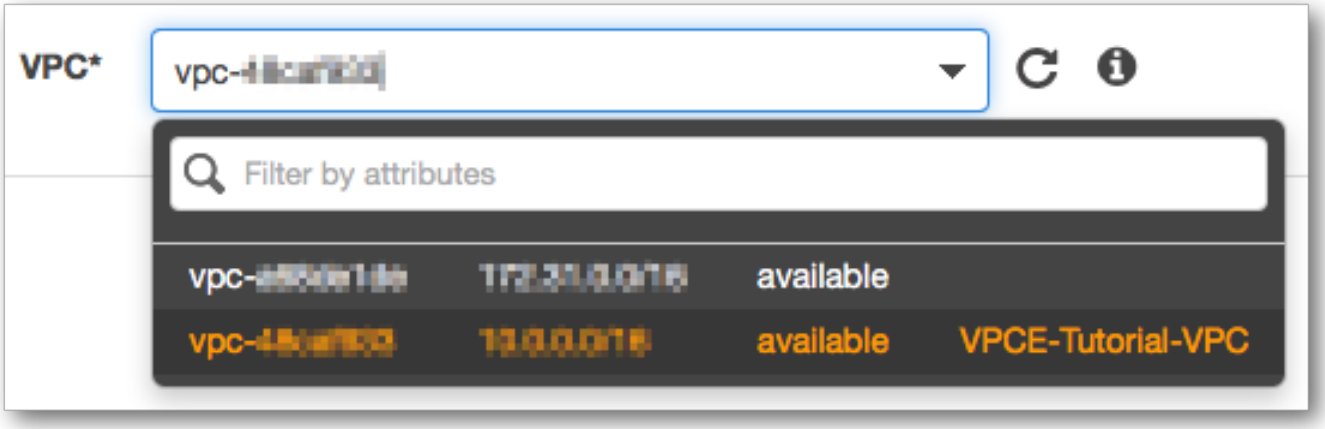
> **Note**
> 该实例仍然无法访问互联网上的其他 Amazon 服务和终端节点。

创建终端节点

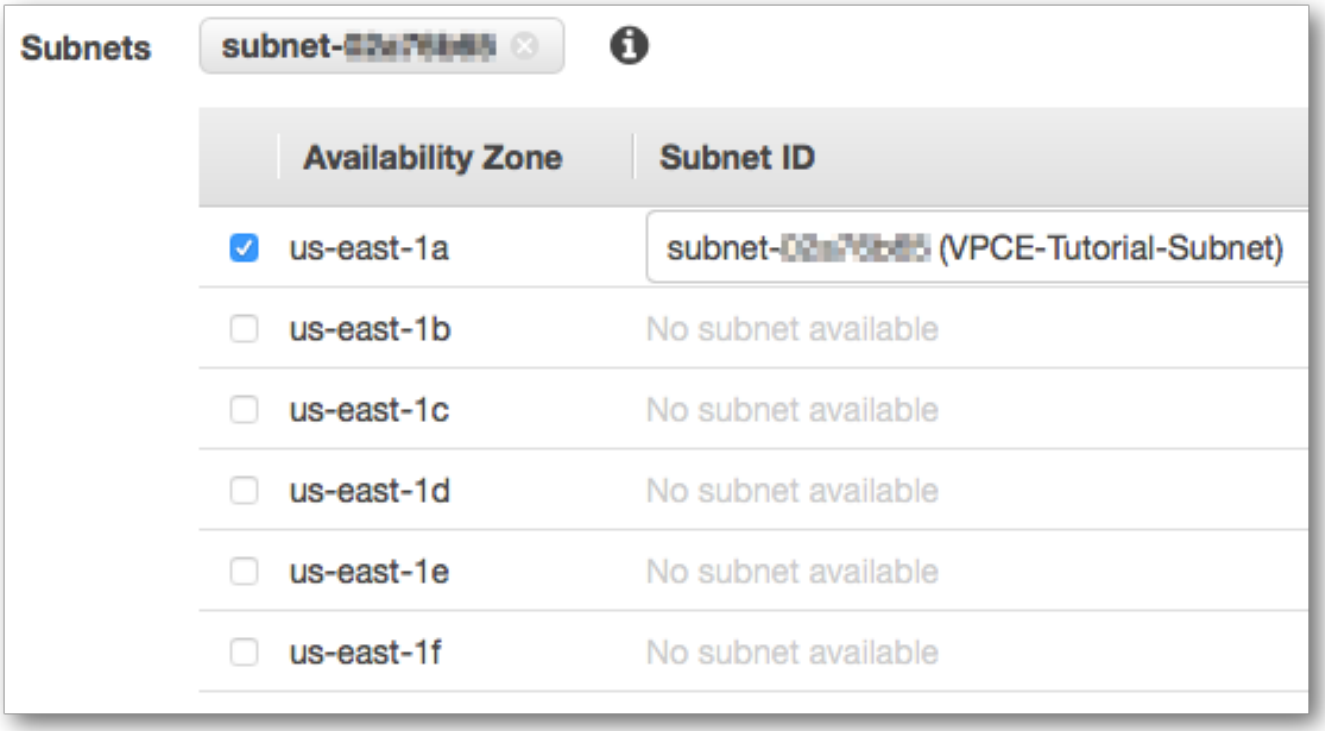
1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在左侧导航菜单中，选择端点。
3. 选择 Create Endpoint（创建端点）。

4. 在 Create Endpoint（创建终端节点）页面上，对于 Service category（服务类别），保留默认选择 Amazon 服务。
5. 对于 Service Name（服务名称），选择 Amazon SNS 的服务名称。

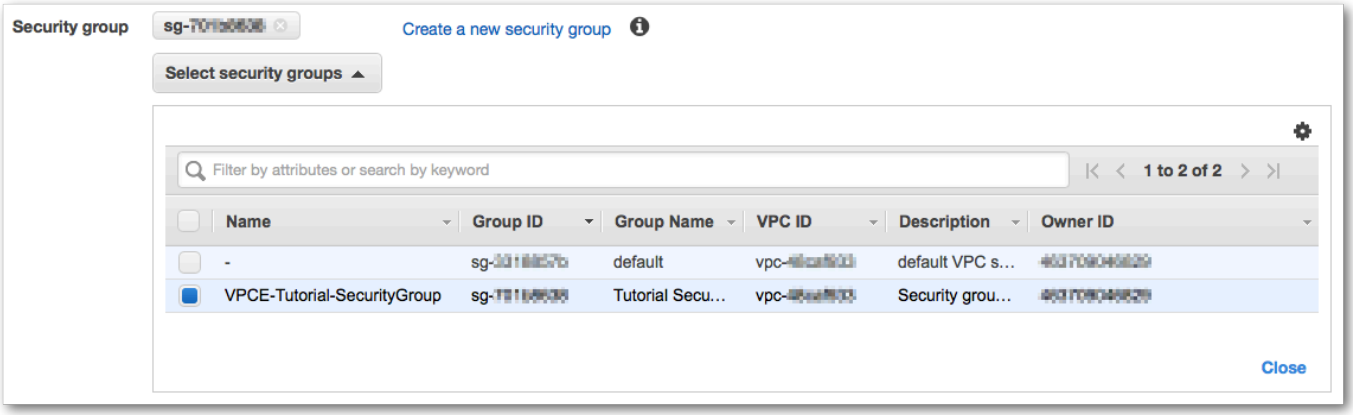
   服务名称因所选区域而异。例如，如果您选择美国东部（弗吉尼亚北部），则服务名称为 com.amazonaws。*us-east-1*.sns。
6. 对于 VPC，选择名为 VPCE-Tutorial-VPC 的 VPC。



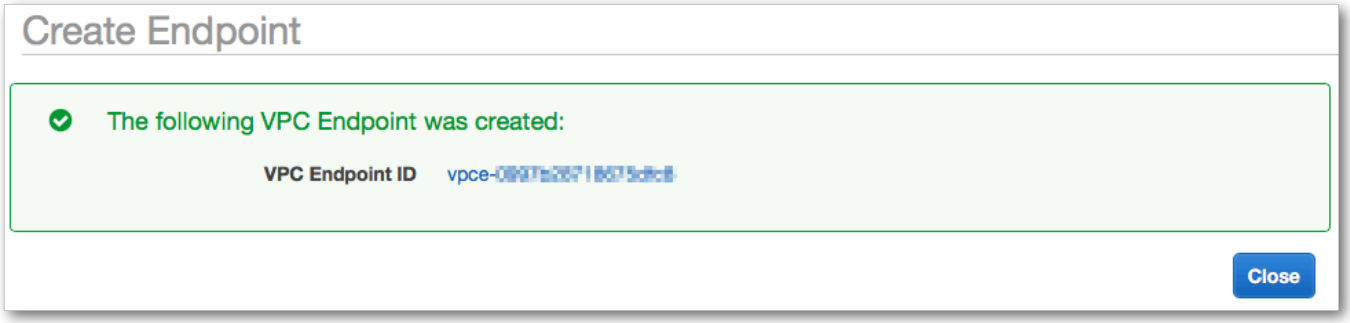
7. 对于 Subnets (子网)，选择在子网 ID 中具有 VPCE-Tutorial-Subnet 的子网。



8. 对于 Enable Private DNS Name (启用私有 DNS 名称)，选择 Enable for this endpoint (为此终端节点启用)。
9. 对于安全组，选择选择安全组，然后选择 vpce-T utorial-。SecurityGroup

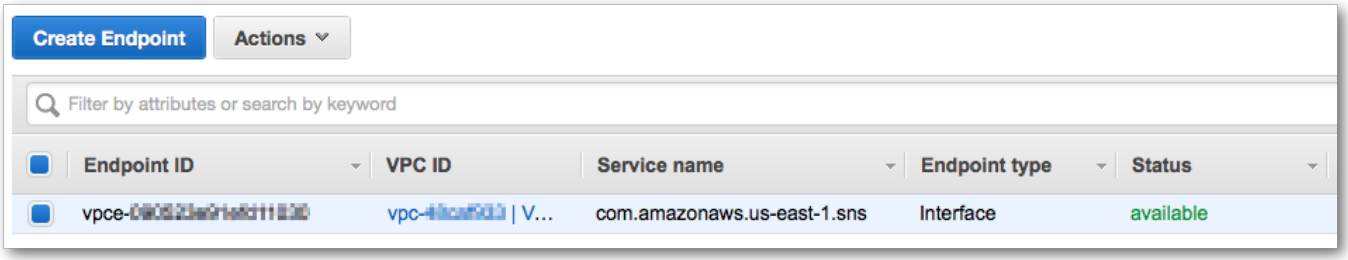


10. 选择创建端点。Amazon VPC 控制台确认已创建 VPC 终端节点。



11. 选择关闭。

    Amazon VPC 控制台会打开 Endpoints（终端节点）页面。新终端节点的状态为 pending (待处理)。在创建过程完成后的几分钟内，该状态将变为 available (可用)。



步骤 5：向 Amazon SNS 主题发布消息

现在，您的 VPC 包含了 Amazon SNS 的终端节点，您可以登录该亚马逊 EC2 实例并向该主题发布消息。

发布消息

1. 如果您的终端不再连接到您的 Amazon EC2 实例，请重新连接：

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. 运行您在之前执行过的相同命令，以向您的 Amazon SNS 主题发布消息。此时，发布尝试成功，并且 Amazon SNS 会返回一条消息 ID：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"


{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

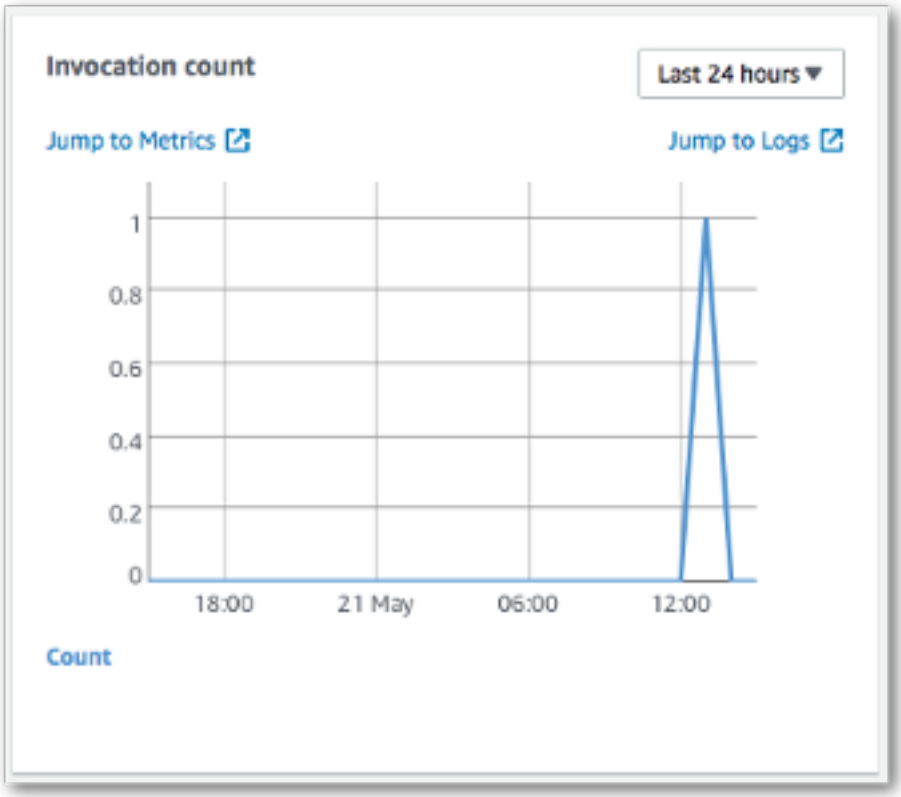
步骤 6：验证您的消息传输

当 Amazon SNS 主题收到消息时，它会通过将消息发送到两个订阅 Lambda 函数来扇出该消息。当这些函数收到消息时，它们会将事件记录到 CloudWatch 日志中。要验证您的消息传送是否成功，请检查函数是否已调用，并检查 CloudWatch 日志是否已更新。

验证是否已调用 Lambda 函数

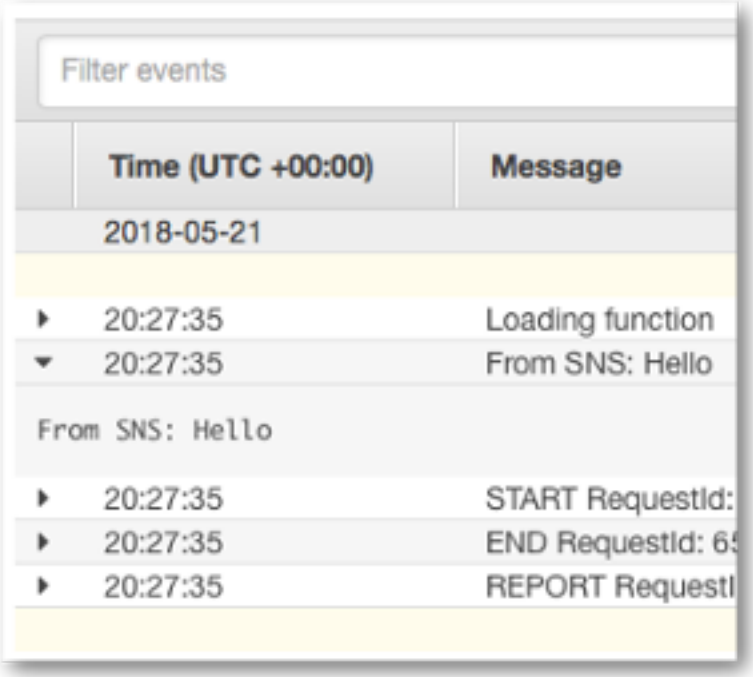
1. 打开 Amazon Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 在 Functions (函数) 页面上，选择 VPCE-Tutorial-Lambda-1。
3. 选择监控。
4. 检查 Invocation count (调用计数) 图表。此图显示了已运行 Lambda 函数的次数。

   调用计数与您向主题发布消息的次数匹配。



验证 CloudWatch 日志是否已更新

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航菜单中，选择 Logs (日志)。
3. 检查由 Lambda 函数写入的日志：
   a. 选择 /-Tutorial aws/lambda/VPCE-Lambda-1/ 日志组。
   b. 选择日志流。
   c. 检查日志是否包含条目 `From SNS: Hello`。



d. 选择控制台顶部的 Log Groups (日志组) 以返回 Log Groups (日志组) 页面。然后，对-Tutorial-the /aws/lambda/VPCE Lambda-2/ 日志组重复上述步骤。

恭喜您！通过将 Amazon SNS 的终端节点添加到 VPC，您能够从由 VPC 管理的网络内将消息发布到主题。消息是私下发布的，不会对公共 Internet 公开。

步骤 7：清除

除非您想要保留您创建的资源，否则可立即将其删除。通过删除不再使用的 Amazon 资源，可以防止对您的资源产生不必要的费用 Amazon Web Services 账户。

首先，使用 Amazon VPC 控制台删除您的 VPC 终端节点。然后，通过在 Amazon CloudFormation 控制台中删除堆栈来删除您创建的其他资源。当您删除堆栈时， Amazon CloudFormation 会将该堆栈的资源从中移除 Amazon Web Services 账户。

删除您的 VPC 终端节点

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在左侧导航菜单中，选择端点。
3. 选择您创建的终端节点。
4. 选择 Actions (操作)，然后选择 Delete Endpoint (终端节点)。

5. 在 Delete Endpoint (删除终端节点) 窗口中，选择 Yes, Delete (是，删除)。

   终端节点状态将变为 deleting (正在删除)。删除操作完成后，终端节点将从页面中删除。

删除您的 Amazon CloudFormation 堆栈

1. 在 https://console.aws.amazon.com/cloudformation 上打开 Amazon CloudFormation 控制台。
2. 选择堆栈 VPCE-Tutorial-Stack。
3. 选择 Actions) (操作) ，然后选择 Delete Stack (删除堆栈) 。
4. 在 Delete Stack (删除堆栈) 窗口中，选择 Yes, Delete (是，删除)。

   堆栈状态将变为 DELETE_IN_PROGRESS。删除操作完成后，堆栈将从页面中删除。

相关资源

有关详细信息，请参阅以下资源：

- Amazon 安全博客：使用保护发布到 Amazon SNS 的消息 Amazon PrivateLink
- 什么是 Amazon VPC？
- VPC 端点
- 什么是亚马逊 EC2？
- Amazon CloudFormation 概念

通过消息数据保护增强 Amazon SNS 的安全性

- 消息数据保护是 Amazon SNS 中的一项功能，用于定义您自己的规则和策略，以审计和控制动态数据而不是静态数据的内容。
- 消息数据保护面向以消息为中心的企业应用程序，提供监管、合规和审计服务，因此，Amazon SNS 主题所有者可以控制数据的传入和传出，并且可以跟踪和记录内容流。
- 您可以编写基于负载的监管规则，以阻止未经授权的负载内容进入您的消息流。
- 您可以向单独订阅者授予不同的内容访问权限，并审计整个内容流流程。

Amazon SNS 中的 Identity and Access Management

访问 Amazon SNS 需要有 Amazon 可用于验证您的请求的证书。这些证书必须具有访问 Amazon 资源（例如 Amazon SNS 主题和消息）的权限。下面几部分提供详细信息来说明如何使用 Amazon Identity and Access Management (IAM) 和 Amazon SNS 控制谁能访问您的资源，从而对这些资源进行保护。

Amazon Identity and Access Management (IAM) Amazon Web Services 服务 可帮助管理员安全地控制对 Amazon 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和获得授权（具有权限）来使用 Amazon SNS 资源。您可以使用 IAM Amazon Web Services 服务 ，无需支付额外费用。

受众

您的使用方式 Amazon Identity and Access Management (IAM) 会有所不同，具体取决于您在 Amazon SNS 中所做的工作。

服务用户 – 如果您使用 Amazon SNS 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon SNS 特征来完成工作，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon SNS 中的功能，请参阅 Amazon Simple Notification Service 身份和访问故障排查。

服务管理员 – 如果您在公司负责管理 Amazon SNS 资源，您可能对 Amazon SNS 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon SNS 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon SNS 搭配使用的更多信息，请参阅Amazon SNS 如何与 IAM 配合使用。

IAM 管理员 – 如果您是 IAM 管理员，您可能需要了解如何编写策略以管理对 Amazon SNS 的访问的详细信息。要查看您可在 IAM 中使用的 Amazon SNS 基于身份的策略示例，请参阅适用于 Amazon Simple Notification Service 的基于身份的策略示例。

使用身份进行身份验证

身份验证是您 Amazon 使用身份凭证登录的方式。您必须以 Amazon Web Services 账户 根用户身份、IAM 用户身份或通过担任 IAM 角色进行身份验证（登录 Amazon）。

如果您 Amazon 以编程方式访问，则会 Amazon 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 Amazon 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的用于签署 API 请求的Amazon 签名版本 4。

无论使用何种身份验证方法，您都可能需要提供其他安全信息。例如， Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 中的Amazon 多重身份验证。

Amazon Web Services 账户 root 用户

创建时 Amazon Web Services 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 Amazon Web Services 服务 和资源。此身份被称为 Amazon Web Services 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 IAM 用户指南中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 Amazon Web Services 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C 或者任何使用 Amazon Web Services 服务 通过身份源提供的凭据进行访问的用户。 Amazon Directory Service当联合身份访问时 Amazon Web Services 账户，他们将扮演角色，角色提供临时证书。

IAM 用户和群组

I AM 用户是您 Amazon Web Services 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的用例，应在需要时更新访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并向该群组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

I AM 角色是您内部具有特定权限 Amazon Web Services 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。要在中临时担任 IAM 角色 Amazon Web Services Management Console，您可以从用

户切换到 IAM 角色（控制台）。您可以通过调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的代入角色的方法。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问：要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM 用户指南》中的针对第三方身份提供商创建角色（联合身份验证）。
- 临时 IAM 用户权限：IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取：您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 Amazon Web Services 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 中的跨账户资源访问。
- 跨服务访问 — 有些 Amazon Web Services 服务 使用其他 Amazon Web Services 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序 EC2 或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 Amazon，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 Amazon Web Services 服务 向下游服务发出请求的请求。 Amazon Web Services 服务只有当服务收到需要与其他 Amazon Web Services 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 Amazon Web Services 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 Amazon Web Services 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon 上运行的应用程序 EC2 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时证书。这比在 EC2 实例中存储访问密钥更可取。要为 EC2 实例分配 Amazon 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在 EC2 实例上运行的程序获得临时证书。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色向在 A mazon EC2 实例上运行的应用程序授予权限。

使用策略管理访问

您可以 Amazon 通过创建策略并将其附加到 Amazon 身份或资源来控制中的访问权限。策略是其中的一个对象 Amazon ，当与身份或资源关联时，它会定义其权限。 Amazon 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 Amazon 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 Amazon Web Services Management Console Amazon CLI、或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户托管策略定义自定义 IAM 权限。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 Amazon Web Services 账户。托管策略包括 Amazon 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 Amazon Web Services 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 Amazon 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。 Amazon WAF要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

Amazon 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)- SCPs 是指定组织或组织单位 (OU) 的最大权限的 JSON 策略 Amazon Organizations。 Amazon Organizations 是一项用于对您的企业拥有的多 Amazon Web Services 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP 限制成员账户中实体的权限，包括每个 Amazon Web Services 账户 root 用户。有关 Organization SCPs s 和的更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs) — RCPs 是 JSON 策略，您可以使用它来设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制成员账户中资源的权限，并可能影响身份（包括 Amazon Web Services 账户 根用户）的有效权限，无论他们是否属于您的组织。有关 Organizations 的更多信息 RCPs，包括 Amazon Web Services 服务 该支持的列表 RCPs，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 Amazon 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

访问控制

Amazon SNS 拥有自己的基于资源的权限系统，该系统使用以与 Amazon Identity and Access Management (IAM) 策略相同的语言编写的策略。这意味着 Amazon SNS 策略与 IAM 策略的作用类似。

> **Note**
>
> 重要的是要明白，所有人 Amazon Web Services 账户 都可以将其权限委托给其账户下的用户。跨账户访问允许您共享对 Amazon 资源的访问，而不需要管理其他用户。有关使用跨账户访问的信息，请参阅 IAM 用户指南中的启用跨账户访问。

Amazon SNS 访问控制应用场景

对于如何授权或拒绝资源访问，您有很大的灵活性。然而，普通的使用案例都相当简单。

- 您想向其他人授予 Amazon Web Services 账户 特定类型的主题操作（例如 “发布”）。有关更多信息，请参阅 授予对主题的 Amazon Web Services 账户 访问权限。
- 您希望仅对 HTTPS 协议限制主题订阅。有关更多信息，请参阅 限制 HTTPS 订阅。
- 您希望允许 Amazon SNS 向 Amazon SQS 队列发布消息。有关更多信息，请参阅 发布消息到 Amazon SQS 队列。

Amazon SNS 访问策略关键概念

以下章节介绍了您需要了解的概念，以便使用访问策略语言。它们都按逻辑顺序介绍，您需要了解的第一项术语在清单最前面。

权限

权限是指允许或不允许访问某种特殊资源的概念。权限基本遵循这种形式：“如果 D 适用时，那么 A 被允许或未被允许向 C 执行 B”。例如：只要 Jane 使用 HTTP 协议 (D)，那么 Jane (A) 就可以向 TopicA (C) 发布 (B) 信息。每当 Jane 向 TopicA 发布信息时，服务器会检查她是否有权限或该请求是否满足权限中所规定的条件。

语句

语句是指对用访问策略语言编写的单个权限的正式说明。您通常编写的语句是一个更大的容器式文档，被称为策略（见下一概念），的一部分。

策略

策略是一个文档（使用访问策略语言编写），充当存放一个或多个语句的容器。例如，策略中可能包含两个语句：一个语句规定 Jane 可以使用电子邮件协议进行订阅，另一个语句规定 Bob 不能向主题 A 发布消息。如下图所示，等效的场景将具有两个策略，一个策略规定 Jane 可以使用电子邮件协议进行订阅，另一个策略规定 Bob 不能向主题 A 发布消息。

策略文档中只允许使用 ASCII 字符。你可以利用`aws:SourceAccount`和解决`aws:SourceOwner`需要插入其他包含非 ASCII 字符 Amazon ARNs 的服务的场景。请参阅 `aws:SourceAccount` 与 `aws:SourceOwner` 之间的差异。

Issuer

发布者是指编写策略以授予资源权限的人。发行者（顾名思义）始终是资源所有者。 Amazon 不允许 Amazon 服务用户为他们不拥有的资源创建策略。如果 John 是资源所有者，则在 John 提交他为授予该资源权限而编写的策略时，将 Amazon 验证他的身份。

主体

委托人是指在策略中获取权限的个人和多个人。委托人是“如果 D 适用的情况下，那么 A 可以对 C 执行 B ”语句中的 A。在策略中，您可将委托人设置为“任何人”（即，您可指定一个通配符代表所有人）。您这样操作，例如，如果您不想根据请求者的实际身份限制访问，那么您可以根据其他的识别特征，例如请求者的 IP 地址。

操作

操作是委托人可以执行的活动。操作是“如果 D 适用的情况下，那么 A 可以对 C 执行 B”语句中的 B。通常，操作只是请求中的操作 Amazon。例如，Jane 使用 `Action=Subscribe` 向 Amazon SNS 发送请求。在一个策略中您可指定一个或多个操作。

资源

资源是委托人请求访问的数据元。在表述“在满足 D 的情况下，A 拥有对 C 执行 B 的许可”中，C 即指资源。

条件与密钥

条件是任何有关权限的限制条件和具体内容。条件是“如果 D 适用的情况下，那么 A 可以对 C 执行 B”语句中的 D。说明条件的策略部分可能是整个部分最详细且最复杂的内容。普通条件与以下项目相关：

- 日期和时间（例如，请求必须在指定日期前到达）
- IP 地址（例如，请求者的 IP 地址必须是某个特定 CIDR 范围的一部分）

一个密钥是设置访问限制指定的特性。例如，访问日期和时间。

您需使用条件和密钥一起明确说明限制。下列示例可帮助您以最简单的方式了解如何实际实施限制：若您要在 2010 年 5 月 30 日之前限制访问，则使用名为 `DateLessThan` 的条件。您可以使用名为 `aws:CurrentTime` 的密钥并将其设置为 `2010-05-30T00:00:00Z`。 Amazon 定义您可以使用的条件和键。 Amazon 服务本身（例如，Amazon SQS 或 Amazon SNS）也可能定义特定于服务的密钥。有关更多信息，请参阅 Amazon SNS API 权限：操作和资源参考。

请求者

请求者是向 Amazon 服务发送请求并请求访问特定资源的人。请求者向其发送的请求本质上 Amazon 是：“在 D 适用的情况下，你允许我对 C 执行 B 吗？”

评估

评估是 Amazon 服务根据适用的策略确定是否应拒绝或允许传入请求的流程。有关评估逻辑的信息，请查见评估逻辑。

效果

效果是指在评估期间您希望一个策略语句返回的结果。当您在一个策略中编写语句时，您需指定该值，可能值为拒绝和允许。

例如，您可以编写一个策略，并声明拒绝所有来自南极洲地区的请求（效果相当于：如果请求是使用为南极洲配置的 IP 地址发出的，则拒绝该请求）。同样地，您可以编写一个策略，其中声明允许所有并

非来自南极洲地区的请求（效果=如果请求不是来自南极洲地区，则允许）。虽然这两个语句看似执行相同的操作，但是在访问策略语言逻辑上，它们是不同的。有关更多信息，请参阅 评估逻辑。

尽管仅可指定两个效果值（“允许”或“拒绝”），但在执行策略评估时，却可能产生三种不同的结果，即：默认拒绝、允许或显式拒绝。有关更多详细，参见以下概念和 评估逻辑。

默认拒绝

默认拒绝是从一个策略中没有允许或显式拒绝的默认结果。

允许

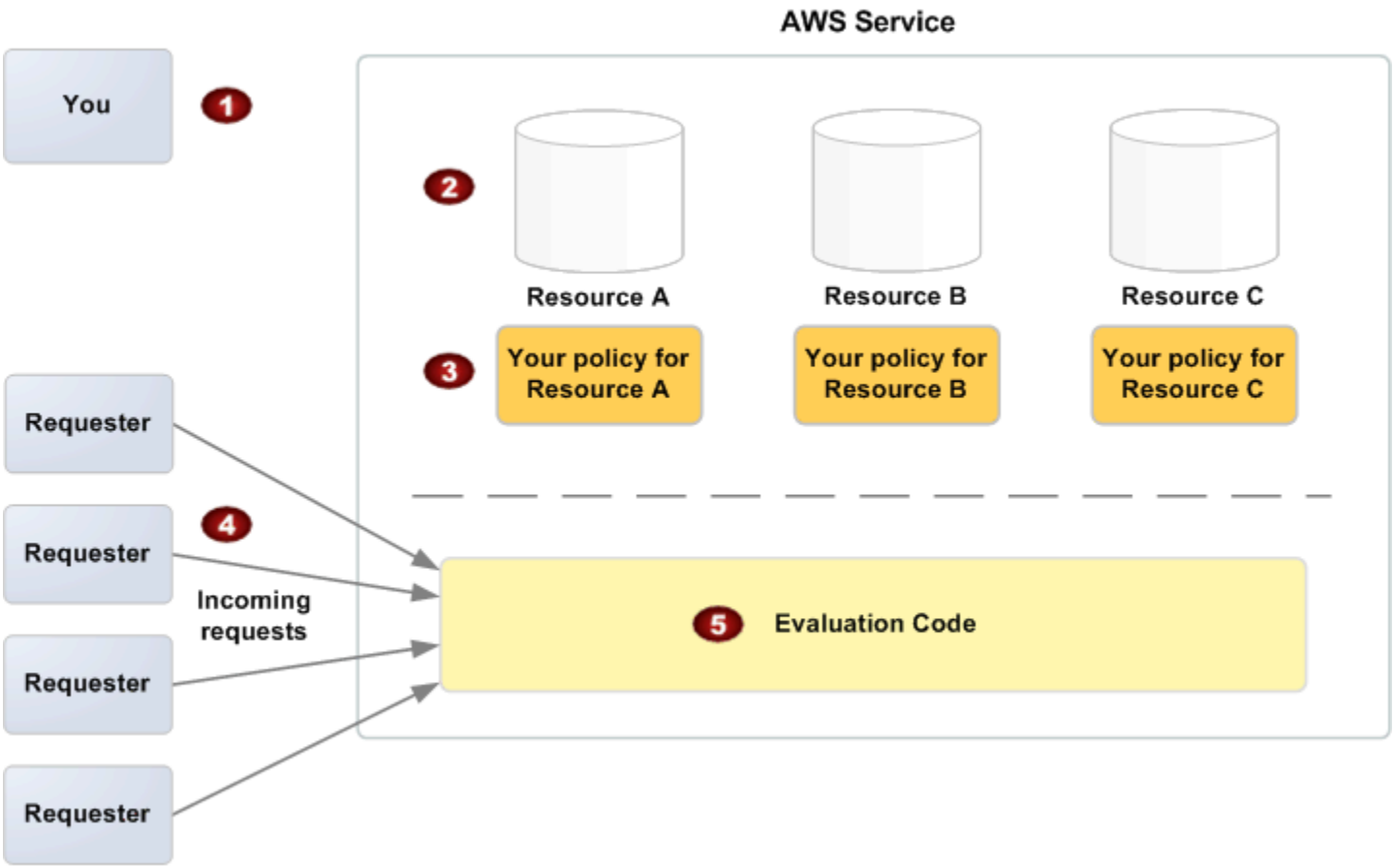
假设任何声明的条件已满足，效果=允许的语句会产生允许。例如：若在 2010 年 4 月 30 日下午 1:00 之前收到请求，则请求将获得允许。“允许”可置换所有“默认拒绝”，但无法置换“显式拒绝”。

显式拒绝

假设任何声明的条件已满足，效果=拒绝的语句会产生显式拒绝。例如：拒绝来自南极洲的所有请求。不管其他什么策略会允许，所有来自南极洲地区的请求都会被拒绝。

Amazon SNS 访问控制架构概述

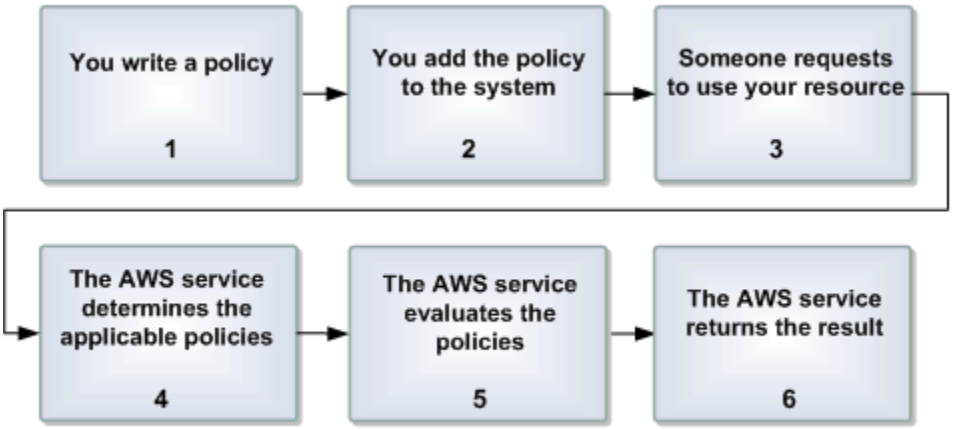
下列图和表格介绍了为您提供资源访问控制的相互作用的主要组成部分。



1	您，资源所有者。
2	您的资源（包含在 Amazon 服务中；例如，Amazon SQS 队列）。
3	您的策略。 通常情况下，每个资源拥有一个策略，虽然您可以有多个。该 Amazon 服务本身提供了一个用于上传和管理政策的 API。
4	请求者及其对 Amazon 服务的传入请求。
5	访问策略语言评估代码。 这是 Amazon 服务中的一组代码，用于根据适用的策略评估传入的请求，并确定是否允许请求者访问资源。有关产品如何做出决定的信息，请参见 评估逻辑。

在 Amazon SNS 中使用访问策略语言

下面的图表介绍了访问控制与访问策略语言协作的一般过程。



将访问控制与访问策略语言一起使用的过程

1	为您的资源编写一个策略。 例如，您编写策略为 Amazon SNS 主题指定权限。
2	您将您的保单上传到 Amazon。 该 Amazon 服务本身提供了一个用于上传政策的 API。例如，您使用 Amazon SNS `SetTopicAttributes` 操作为特定的 Amazon SNS 主题上载策略。
3	某人向您发出使用您的资源的请求。 例如，某用户向 Amazon SNS 发送使用您的一个主题的请求。
4	该 Amazon 服务决定哪些策略适用于该请求。 例如，Amazon SNS 将查看所有可用的 Amazon SNS 策略，并确定哪些策略适用（基于资源是什么、请求者是谁等）。
5	该 Amazon 服务对策略进行评估。 例如，Amazon SNS 将对策略进行评估，确定是否允许请求者使用您的主题。有关决策逻辑的信息，请参见评估逻辑。
6	该 Amazon 服务要么拒绝请求，要么继续处理请求。

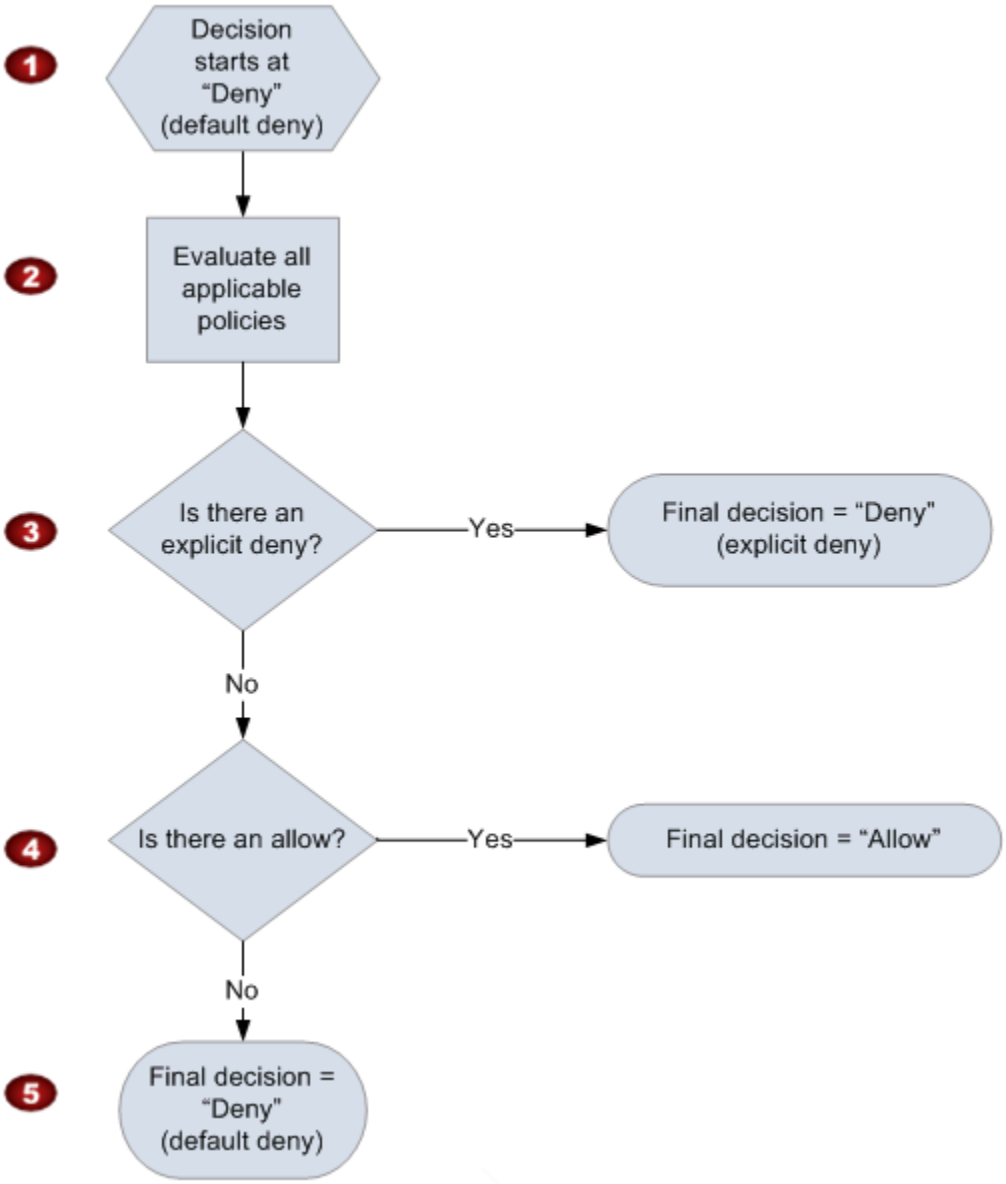
	例如，根据策略评估结果，服务将返回一个“访问被拒绝”的错误信息给请求者，或继续处理该请求。

评估逻辑

评价期间的目的是为了确定应该允许还是拒绝授予请求。评估逻辑遵循多个基本规则：

- 在默认情况下，除了您，任何人提出使用您资源的请求均会被拒绝。
- 一个允许可以超控任何其他默认拒绝
- 一个显式拒绝可以超控任何允许
- 策略评估的顺序不重要

以下流程图和讨论更加详细地描述了如何做出决定。



1	决定开始是一个默认拒绝。
2	然后执行代码将评估适用于请求的所有策略（根据资源、委托人、操作和条件）。 执行代码评估策略的顺序不重要。
3	在所有这些策略中，执行代码将寻找一个能适用于请求的显式拒绝指令。 即使仅找到一处，执行代码也会发回“拒绝”的决定，并结束处理流程（此为“显式拒绝”；有关更多信息，请参见 显式拒绝）。

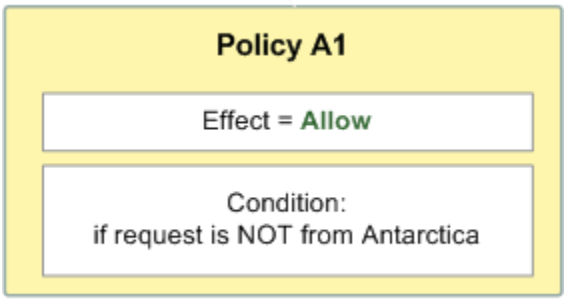
4	如果没有找到显式拒绝，那么执行代码将寻找适用于请求的任何“允许”指令。 如果它还是找到了一个，那么执行代码将返回一个“允许”决定，且整个过程完成（服务将继续处理该请求）。
5	如果没有找到允许，那么最终的决定将是“拒绝”，因为没有显式拒绝或允许，所以这将被视为是一个默认拒绝（有关更多信息，请参见默认拒绝）。

显式拒绝和默认拒绝的相互作用

如果策略不直接适用于请求，那么策略将产生一个默认拒绝。例如，如果用户请求使用 Amazon SNS，但该主题的策略根本没有提及该用户的 Amazon Web Services 账户 政策，则该策略会导致默认拒绝。

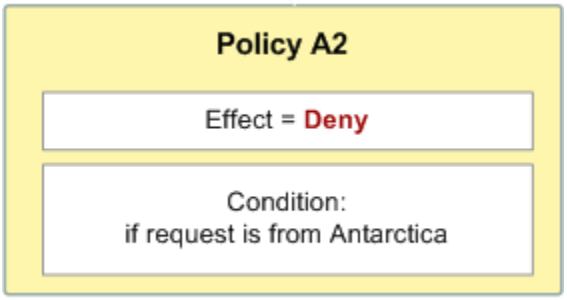
如果一个语句中的某个条件未被满足，那么策略将产生一个默认拒绝。如果语句中的所有条件都满足，那么根据策略中的效果元素的值，策略或许会产生允许，或许会产生显式拒绝。如果一个条件未被满足，策略没有指定如何处理，那么在那种情况下默认值将产生一个默认拒绝。

例如，假设您想要阻止来自南极洲地区的请求进入。只要请求不是来自于南极洲地区，您编写的策略（称作策略 A1）将允许接受请求。下列示意图说明了该策略。



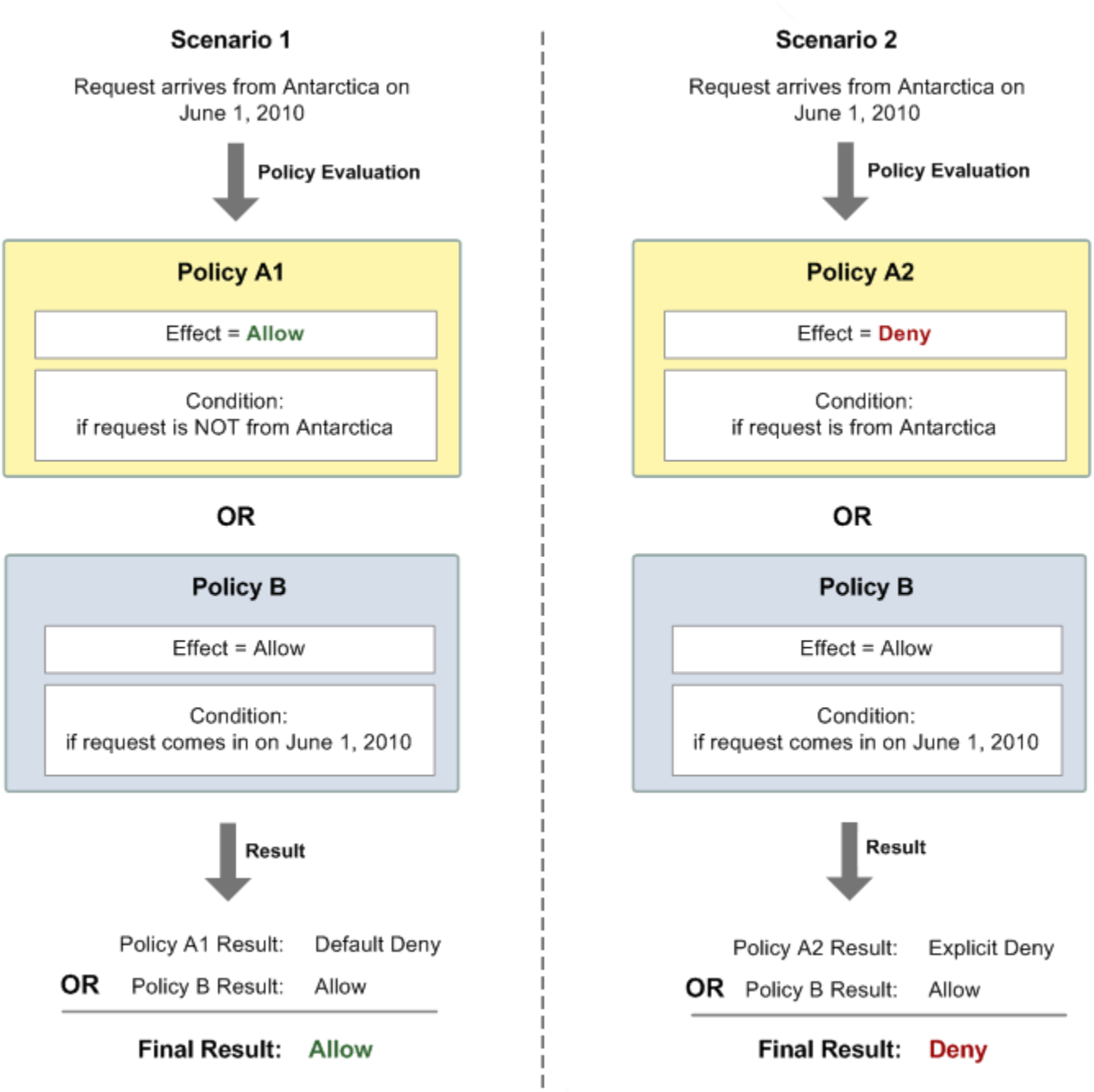
如果某人从美国发出请求，那么条件已经满足 (该请求不是来自南极洲)。因此，该请求将被允许。但是，如果某人从南极洲地区发出请求，那么条件未满足，因此策略结果将是默认拒绝。

您可通过按照下列示意图重新编写策略（称作策略 A2）将结果转变为一个显式拒绝。此时，如果请求是来自南极洲地区，那么策略将明确拒绝该请求。



如果某人从南极洲发出请求，那么条件已经满足，策略的结果将是一个显式拒绝。

默认拒绝和显式拒绝的区别很重要，因为默认拒绝可以被允许覆盖，但显式拒绝就不能。例如，假设有另一个策略，允许在 2010 年 6 月 1 日到达的请求。那么，与限制从南极洲访问的策略相结合，该策略将如何对总体结果产生影响？ 当将按日期要求设置的策略与上述策略 A1 和 A2 相结合时，我们将对比综合结果。方案 1 是将策略 A1 与策略 B 相结合，方案 2 是将策略 A2 与策略 B 相结合。以下图表和讨论显示了如果于 2010 年 6 月 1 日从南极洲区域发出请求输入时的结果。



在方案 1 中，策略 A1 将返回一个默认拒绝，如本节之前所描述的那样。Policy B 返回“允许”结果，因为该策略（依照定义）允许在 2010 年 6 月 1 日发送请求。Policy B 返回的“允许”结果将置换 Policy A1 的“默认拒绝”结果，因此，请求获得允许。

在方案 2 中，策略 A2 返回了一个显式拒绝，如本节之前所描述的那样。此外，策略 B 返回了一个允许。从策略 A2 发出的显式拒绝将超控从策略 B 发出的允许，因此该请求会被拒绝。

用于 Amazon SNS 访问控制的示例案例

本节描述了针对访问控制的几个典型使用案例示例。

授予对主题的 Amazon Web Services 账户 访问权限

假设您在 Amazon SNS 中有一个主题，并且您想允许一个或多个 Amazon Web Services 账户 主题对该主题执行特定操作，例如发布消息。使用 Amazon SNS API 操作 `AddPermission` 即可做到这一点。

该`AddPermission`操作允许您指定主题、列表 Amazon Web Services 账户 IDs、操作列表和标签。然后，Amazon SNS 会自动生成新的策略声明，并将其添加到该主题的访问控制策略中。您无需自己编写策略声明，Amazon SNS 将为您处理。如果您以后需要移除该策略，则可以通过调用 `RemovePermission` 并提供您在添加权限时使用的标签来实现。

例如，如果您调用 `AddPermission` arn: aws: sns: us-east-2:444455556666:，指定 MyTopic ID 1111-2222-3333 Amazon Web Services 账户 、操作`Publish`和标签，Amazon `grant-1234-publish` SNS 将生成以下策略声明并将其插入到该主题的访问控制策略中：

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

添加此声明后， Amazon Web Services 账户 1111-2222-3333 将有权向该主题发布消息。

其他信息：

- 自定义策略管理：虽然 `AddPermission` 可以方便地授予权限，但对于更复杂的场景，例如添加条件或向特定 IAM 角色或服务授予权限，手动管理主题的访问控制策略通常很有用。为此，您可以使用 `SetTopicAttributes` API 直接更新策略属性。
- 安全最佳实践：在授予权限时要谨慎，确保只有受信任的实体 Amazon Web Services 账户 或实体才能访问您的 Amazon SNS 主题。定期审查和审核您的主题所附的策略，以维护安全性。

- 策略限制：请记住，Amazon SNS 策略的规模和复杂性有限制。如果您需要添加许多权限或复杂条件，请确保您的策略保持在这些限制范围内。

限制 HTTPS 订阅

要将您的 Amazon SNS 主题的通知传送协议限制为 HTTPS，您必须创建自定义策略。Amazon SNS 中的 `AddPermission` 操作不允许您在授予主题访问权限时指定协议限制。因此，您需要手动编写强制执行此限制的策略，然后使用 `SetTopicAttributes` 操作将该策略应用于您的主题。

您可以通过以下方式创建限制订阅 HTTPS 的策略：

1. 撰写策略。该策略必须指定您要授予访问权限的 Amazon Web Services 账户 ID，并强制执行仅允许 HTTPS 订阅的条件。以下是一个策略示例，该策略授予 Amazon Web Services 账户 ID 1111-2222-3333 订阅该主题的权限，但前提是使用的协议是 HTTPS。

```
{
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
      }
    }
  }]
}
```

2. 应用策略。使用 Amazon SNS API 中的 `SetTopicAttributes` 操作将此策略应用于您的主题。将主题的 `Policy` 属性设置为您创建的 JSON 策略。

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

其他信息：

- 自定义访问控制。这种方法允许您实施更精细的访问控制，例如限制订阅协议，而光靠 `AddPermission` 操作是无法实现的。自定义策略为需要特定条件（例如协议强制执行或 IP 地址限制）的场景提供了灵活性。
- 安全最佳实践。将订阅限制为 HTTPS 可确保传输中的数据经过加密，从而增强通知的安全性。定期查看您的主题策略，确保它们符合您的安全和合规性要求。
- 策略测试。在生产环境中应用策略之前，请在开发环境中对其进行测试，以确保其行为符合预期。这有助于防止意外访问问题或意外限制。

发布消息到 Amazon SQS 队列

要将来自您的 Amazon SNS 主题的消息发布到 Amazon SQS 队列，您需要在 Amazon SQS 队列上配置正确的权限。虽然 Amazon SNS 和 Amazon SQS 都 Amazon使用访问控制策略语言，但您必须在 Amazon SQS 队列上明确设置策略，以允许从亚马逊 SNS 主题发送消息。

您可以使用 `SetQueueAttributes` 操作将自定义策略应用于 Amazon SQS 队列，从而实现此目的。与 Amazon SNS 不同，Amazon SQS 不支持创建带有条件的策略声明的 `AddPermission` 操作。因此，您必须手动编写策略。

以下是 Amazon SQS 策略的示例，该策略授予对 Amazon SNS 向队列发送消息的权限。请注意，此策略与 Amazon SQS 队列有关，与 Amazon SNS 主题无关。这些指定操作都是 Amazon SQS 操作，并且资源是队列的 Amazon 资源名称（ARN）。您可以使用 `GetQueueAttributes` 操作检索队列的 ARN。

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

```
}
```

这种策略，根据所发送信息的来源，利用 `aws:SourceArn` 条件限制对 SQS 队列的访问。这样可以确保只允许将来自指定 SNS 主题（在本例中为 arn: aws: sns: us-east-2:444455556666: ）的消息传送到队列。MyTopic

其他信息：

- 队列 ARN。确保您使用 `GetQueueAttributes` 操作检索 Amazon SQS 队列的正确 ARN。此 ARN 对于设置正确的权限至关重要。
- 安全最佳实践。设置策略时，请始终遵循最低权限原则。仅向 Amazon SNS 主题授予与 Amazon SQS 队列进行交互的必要权限，并定期查看您的政策，确保其有效且安全 up-to-date
- Amazon SNS 中的默认策略。Amazon SNS 不会自动授予允许其他 Amazon Web Services 服务 或账户访问新创建主题的默认策略。默认情况下，Amazon SNS 主题是在没有权限的情况下创建的，这意味着它们是私有的，只有创建它们的账户才能访问。要为其他 Amazon Web Services 服务、账户或委托人启用访问权限，您必须明确定义访问策略并将其附加到该主题。这符合最低权限原则，确保默认情况下不会授予任何意外访问权限。
- 测试和验证。设置策略后，通过将消息发布到 Amazon SNS 主题并验证消息是否已成功传送到 Amazon SQS 队列来测试集成。这有助于确认策略配置是否正确。

允许 Amazon S3 事件通知发布到主题

要允许另一个 Amazon S3 存储桶 Amazon Web Services 账户 向您的 Amazon SNS 主题发布事件通知，您需要相应地配置该主题的访问策略。这包括编写自定义策略，向特定 Amazon Web Services 账户 的 Amazon S3 服务授予权限，然后将此策略应用于您的 Amazon SNS 主题。

您可以通过以下方式进行设置：

1. 撰写策略。该政策应授予 Amazon S3 服务 (s3.amazonaws.com) 发布到您的 Amazon SNS 主题的必要权限。您将使用`SourceAccount`条件来确保只有拥有 Amazon S3 存储桶的指定 Amazon Web Services 账户人员才能向您的主题发布通知。

   下面是一个策略示例：

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

- 话题所有者 — 111122223333 是 Amazon Web Services 账户 拥有亚马逊 SNS 话题的 ID。
- 亚马逊 S3 存储桶所有者 — 444455556666 是 Amazon Web Services 账户 拥有发送通知的亚马逊 S3 存储桶的 ID。

2. 应用策略。使用 `SetTopicAttributes` 操作在您的 Amazon SNS 主题上设置此策略。这将更新主题的访问控制，使其包含您的自定义策略中指定的权限。

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

其他信息：

- 使用 **`SourceAccount`** 条件。该`SourceAccount`条件可确保只有源自指定 Amazon Web Services 账户 （在本例中为 444455556666）的事件才能触发 Amazon SNS 主题。这是一项安全措施，可防止未经授权的账户向您的主题发送通知。
- 其他服务支持 **`SourceAccount`**。以下服务支持 `SourceAccount` 条件。当您想要根据原始账户限制访问您的 Amazon SNS 主题时，使用此条件至关重要。
  - Amazon API Gateway
  - Amazon CloudWatch
  - Amazon DevOps Guru
  - Amazon EventBridge
  - 亚马逊 GameLift 服务器
  - Amazon Pinpoint SMS 和 Voice API

- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- Amazon CodeCommit
- Amazon Directory Service
- Amazon Lambda
- Amazon Systems Manager Incident Manager

- 测试和验证。应用策略后，通过在 Amazon S3 存储桶中触发事件并确认该事件已成功发布到您的 Amazon SNS 主题来测试设置。这将有助于确保您的策略配置正确。
- 安全最佳实践。定期审查和审核您的 Amazon SNS 主题策略，确保它们符合您的安全要求。仅限可信账户和服务访问对于维护安全运营至关重要。

允许 Amazon SES 向其他账户拥有的主题发布

您可以 Amazon Web Services 服务 允许其他人发布他人拥有的主题 Amazon Web Services 账户。假设您登录了 111122223333 账户，打开了 Amazon SES 并创建了一封电子邮件。要将有关此电子邮件的通知发布到 444455556666 账户拥有的 Amazon SNS 主题，您需要创建一个如下所示的策略。为此，您需要提供有关委托人（其他服务）和每个资源的所有权的信息。`Resource` 语句提供主题 ARN，其中包括主题所有者的账户 ID 444455556666。`"aws:SourceOwner": "111122223333"` 语句指定您的账户拥有该电子邮件。

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

向 Amazon SNS 发布事件时，以下服务支持 `aws:SourceOwner`：

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- 亚马逊 GameLift 服务器
- Amazon Pinpoint SMS 和 Voice API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- Amazon CodeCommit
- Amazon Directory Service
- Amazon Lambda
- Amazon Systems Manager Incident Manager

`aws:SourceAccount` 与 `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner` 已弃用，新服务只能通过 `aws:SourceArn` 和 `aws:SourceAccount` 与 Amazon SNS 集成。对于目前支持 `aws:SourceOwner` 的现有服务，Amazon SNS 仍保持向后兼容性。

`aws:SourceAccount` 和 `aws:SourceOwner` 条件键由一些 Amazon Web Services 服务 在它们发布到 Amazon SNS 主题时设置。如果支持，则该值将是服务代表其发布数据的 12 位 Amazon 账户 ID。一些服务支持其中一项，一些服务支持另一项。

- 允许 Amazon S3 事件通知发布到主题请参阅，了解 Amazon S3 通知的使用方式`aws:SourceAccount`以及支持该条件的 Amazon 服务列表。
- 允许 Amazon SES 向其他账户拥有的主题发布有关 Amazon SES 的使用方式`aws:SourceOwner`以及支持该条件的 Amazon 服务列表，请参阅。

允许组织中的账户 Amazon Organizations 向其他账户中的主题发布内容

该 Amazon Organizations 服务可帮助您集中管理账单，控制访问权限和安全性，并在整个系统中共享资源 Amazon Web Services 账户。

您可以在 Organizations 控制台中找到您的组织 ID。有关更多信息，请参阅从管理账户查看组织的详细信息。

在此示例中，任何组织成员都`myOrgId`可以在账户 Amazon Web Services 账户 中向 Amazon SNS 主题发布内容`MyTopic`。`444455556666`该策略使用 `aws:PrincipalOrgID` 全局条件键检查组织 ID 值。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

允许将任何 CloudWatch 警报发布到其他账户中的某个主题

使用以下步骤调用带有不同 CloudWatch 警报的 Amazon SNS 主题。 Amazon Web Services 账户此示例使用两个账户：

- 账户 A 用于创建 CloudWatch 警报。

- 账户 B 用于创建 SNS 主题。

在账户 B 中创建 SNS 主题

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择主题，然后选择创建主题。
3. 为主题类型选择标准，然后为该主题创建名称。
4. 选择创建主题，然后复制该主题的 ARN。
5. 在导航窗格中，选择订阅，然后选择创建订阅。
6. 在主题 ARN 部分添加主题的 ARN，选择电子邮件作为协议，然后输入电子邮件地址。
7. 选择 “创建订阅”，然后查看您的电子邮件以确认订阅。

在账户 A 中创建 CloudWatch 警报

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择警报，然后选择创建警报。
3. 如果您尚未创建警报，请立即创建警报。否则，请选择您的指标，然后提供阈值和比较参数的详细信息。
4. 在配置操作的通知下，选择使用主题 ARN 通知其他账户，然后输入账户 B 中的主题 ARN。
5. 为警报创建名称，然后选择创建警报。

更新账户 B 中 SNS 主题的访问策略

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择主题，然后选择主题。
3. 选择 “编辑”，然后将以下内容添加到策略中：

> **Note**
>
> 将以下政策中的示例值替换为您自己的值。

```
{
```

```
  "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:example-region:111122223333:alarm:"
        }
      }
    }
  ]
}
```

测试警报

要测试警报，要么根据指标数据点更改警报阈值，要么手动更改警报状态。当您更改警报阈值或警报状态时，您会收到一封电子邮件通知。

使用本地 Amazon SNS 主题并转发消息的解决方法

使用以下步骤启用跨账户 Amazon SNS 警报通知 CloudWatch：

1. 使用与CloudWatch警报相同的账户 (111122223333) 创建亚马逊 SNS 主题。
2. 为该亚马逊 SNS 主题订阅 Lambda 函数或亚马逊 EventBridge规则。
3. 然后，Lambda 函数或 EventBridge 规则可以将消息发布到目标账户 (444455556666) 中的 Amazon SNS 主题。

仅限从特定 VPC 终端节点发布到 Amazon SNS 主题

在这种情况下，账户 444455556666 中的主题只允许从具有 ID `vpce-1ab2c34d` 的 VPC 终端节点发布。

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

Amazon SNS 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon SNS 的访问权限之前，了解哪些 IAM 功能可用于 Amazon SNS。

可与 Amazon Simple Notification Service 结合使用的 IAM 功能

IAM 特征	Amazon SNS 支持
基于身份的策略	是
基于资源的策略	是
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	否
ABAC（策略中的标签）	部分

IAM 特征	Amazon SNS 支持
临时凭证	是
主体权限	是
服务角色	是
服务相关角色	否

要全面了解 Amazon SNS 和其他 Amazon 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的Amazon 服务。

Amazon 的托管策略 Amazon Simple Notification Service

Amazon 托管策略是由创建和管理的独立策略 Amazon。 Amazon 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， Amazon 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 Amazon 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 Amazon 托管策略中定义的权限。如果 Amazon 更新 Amazon 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 Amazon 最有可能在启动新的 API 或现有服务可以使用新 Amazon Web Services 服务 的 API 操作时更新 Amazon 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略。

Amazon 托管策略：Amazon A SNSFull ccess

`AmazonSNSFullAccess`使用提供对 Amazon SNS 的完全访问权限。 Amazon Web Services Management Console该策略还包括以下针对使用 Amazon SNS 调用 Amazon End User Messaging SMS 时的读取和写入操作。您可以将此策略附加到您的用户、组或角色。

权限详细信息

以下权限仅在使用 Amazon SNS APIs 时适用：

- `sns:*` – 允许执行与 Amazon SNS 相关的任何操作的完全权限。此通配符（*）表示用户可以执行所有可能的 Amazon SNS 操作。

- `sms-voice:DescribeVerifiedDestinationNumbers` – 允许您检索已通过验证可在 Amazon Web Services 账户中发送短信的电话号码列表。
- `sms-voice:CreateVerifiedDestinationNumber`— 允许您验证新的电话号码，以便在其中使用短信服务 Amazon。
- `sms-voice:SendDestinationNumberVerificationCode` – 允许您向正在 Amazon中进行短信收发验证的电话号码发送验证码。
- `sms-voice:SendTextMessage` - 允许您创建新的短信并将其发送到接收者的电话号码。每次调用时，`SendTextMessage` 仅向一个接收者发送短信。
- `sms-voice:DeleteVerifiedDestinationNumber`— 允许您从已验证的号码列表中删除电话号码 Amazon Web Services 账户
- `sms-voice:VerifyDestinationNumber` – 允许您启动和完成要用于 Amazon中的短信收发服务的电话号码的验证过程。
- `sms-voice:DescribeAccountAttributes` - 允许您检索有关与 Amazon中的短信收发服务相关的账户级别属性的详细信息。
- `sms-voice:DescribeSpendLimits` - 允许您检索有关与 Amazon Web Services 账户中的短信收发服务相关的支出限额的信息
- `sms-voice:DescribePhoneNumbers` - 允许您检索与 Amazon Web Services 账户 中的短信收发服务关联的电话号码的详细信息
- `sms-voice:SetTextMessageSpendLimitOverride`— 允许您在其中设置或改写短信的支出限额 Amazon Web Services 账户
- `sms-voice:DescribeOptedOutNumbers` - 允许您检索已选择不接收来自您的 Amazon 账户的短信的电话号码列表。
- `sms-voice:DeleteOptedOutNumber`— 允许您从已选择退出的号码列表中删除电话号码 Amazon Web Services 账户

`AmazonSNSFullAccess` 示例策略

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

要查看此策略的权限，请参阅《Amazon 托管策略参考》中的 Amazon A SNSFull cces s。

Amazon 托管策略：Amazon SNSRead OnlyAccess

`AmazonSNSReadOnlyAccess` 通过 Amazon Web Services Management Console提供对 Amazon SNS 的只读访问权限。该策略还包括以下使用 Amazon SNS 调用 Amazon End User Messaging SMS 时的只读操作。您可以将此策略附加到用户、组和角色。

权限详细信息

以下权限仅在使用 Amazon SNS APIs 时适用：

- `sns:GetTopicAttributes` - 允许您检索 Amazon SNS 主题的属性。这包括主题的 ARN（Amazon 资源名称）、订阅用户列表、交付策略、访问控制策略以及与该主题关联的任何其他元数据等信息。

- `sns:List*` - 允许您对 Amazon SNS 资源执行任何以 `List` 开头的操作。这包括列出与 Amazon SNS 相关的各种元素的权限，例如：
  - `sns:ListTopics` - 允许您在 Amazon Web Services 账户中检索所有 Amazon SNS 主题的列表。
  - `sns:ListSubscriptions` - 允许您检索 Amazon SNS 主题的所有订阅列表。
  - `sns:ListSubscriptionsByTopic` - 允许您列出对 Amazon SNS 主题的所有订阅。
  - `sns:ListPlatformApplications` - 允许您列出为移动推送通知创建的所有平台应用程序。
  - `sns:ListEndpointsByPlatformApplication` - 允许您列出与平台应用程序关联的所有端点。
- `sns:CheckIfPhoneNumberIsOptedOut` - 允许您检查特定的电话号码是否已选择不通过 Amazon SNS 接收短信。
- `sns:GetEndpointAttributes` - 允许您检索与 Amazon SNS 平台应用程序关联的端点的属性。这可能包括端点的启用状态、自定义用户数据以及与端点关联的任何其他元数据等属性。
- `sns:GetDataProtectionPolicy` - 允许您检索与 Amazon SNS 主题相关的数据保护策略。
- `sns:GetPlatformApplicationAttributes` - 允许您检索 Amazon SNS 平台应用程序的属性。Amazon SNS 中使用平台应用程序，通过 Apple Push Notification Service（APNS）或 Firebase Cloud Messaging（FCM）等服务向移动设备发送推送通知。
- `sns:GetSMSAttributes` - 允许您检索 Amazon Web Services 账户的默认短信设置。
- `sns:GetSMSSandboxAccountStatus` - 允许您检索 Amazon Web Services 账户的短信沙盒的当前状态。
- `sns:GetSubscriptionAttributes` - 允许您检索 Amazon SNS 主题的特定订阅的属性。
- `sms-voice:DescribeVerifiedDestinationNumbers`— 允许您查看或检索已通过验证可在其中发送 SMS 消息的电话号码列表 Amazon Web Services 账户
- `sms-voice:DescribeAccountAttributes` - 允许您查看或检索有关与 Amazon中的短信收发服务相关的账户级别属性的信息。
- `sms-voice:DescribeSpendLimits` - 允许您查看或检索有关与您的 Amazon Web Services 账户中的短信收发服务相关的支出限额的信息
- `sms-voice:DescribePhoneNumbers` – 允许您查看或检索有关用于 Amazon Web Services 账户中的短信收发服务的电话号码的信息
- `sms-voice:DescribeOptedOutNumbers` - 允许您查看或检索已选择不接收来自您的 Amazon Web Services 账户的短信的电话号码列表

`AmazonSNSReadOnlyAccess` 示例策略

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

要查看此策略的权限，请参阅《Amazon 托管策略参考》中的 Amazon A SNSFull cces s。

亚马逊 SNS 更新了托管政策 Amazon

查看自该服务开始跟踪这些更改以来对 Amazon SNS Amazon 托管政策的更新的详细信息。有关此页面更改的自动提示，请订阅 Amazon SNS 文档历史记录页面上的 RSS 源。

更改	描述	日期
AmazonSNSFullAccess – 对现有策略的更新	Amazon SNS 添加了新的权限，允许使用 Amazon Web Services Managemen t Console完全访问 Amazon SNS。	09/24/2024
亚马逊 SNSRead OnlyAccess-对现有政策的更新	Amazon SNS 添加了新的权限，允许使用 Amazon Web Services Management Console对 Amazon SNS 进行只读访问。	09/24/2024
Amazon SNS 开启了跟踪更改	Amazon SNS 开始跟踪其 Amazon 托管政策的变更。	08/27/2024

Amazon SNS 的策略操作

支持策略操作：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 Amazon SNS 操作的列表，请参阅《服务授权参考》中的 Amazon Simple Notification Service 定义的资源。

Amazon SNS 中的策略操作在操作前面使用以下前缀：

```
sns
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

要查看 Amazon SNS 基于身份的策略的示例，请参阅适用于 Amazon Simple Notification Service 的基于身份的策略示例。

Amazon SNS 的策略资源

支持策略资源：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 Amazon SNS 资源类型及其列表 ARNs，请参阅服务授权参考中的亚马逊简单通知服务定义的操作。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon Simple Notification Service 定义的资源。

要查看 Amazon SNS 基于身份的策略的示例，请参阅适用于 Amazon Simple Notification Service 的基于身份的策略示例。

Amazon SNS 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 Amazon 使用逻辑 `AND` 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑`OR`运算来 Amazon 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

Amazon 支持全局条件密钥和特定于服务的条件键。要查看所有 Amazon 全局条件键，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥。

要查看 Amazon SNS 条件键的列表，请参阅《服务授权参考》中的 Amazon Simple Notification Service 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Simple Notification Service 定义的资源。

要查看 Amazon SNS 基于身份的策略的示例，请参阅适用于 Amazon Simple Notification Service 的基于身份的策略示例。

ACLs 在 Amazon SNS 中

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

ABAC 与 Amazon SNS

支持 ABAC（策略中的标签）：部分支持

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 Amazon，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 Amazon 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 Amazon SNS

支持临时凭证：是

当你使用临时证书登录时，有些 Amazon Web Services 服务 不起作用。有关更多信息，包括哪些 Amazon Web Services 服务 适用于临时证书，请参阅 IAM 用户指南中的Amazon Web Services 服务与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 Amazon Web Services Management Console 使用的是临时证书。例如，当您 Amazon 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的从用户切换到 IAM 角色（控制台）。

您可以使用 Amazon CLI 或 Amazon API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 Amazon。 Amazon 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon SNS 的跨服务主体权限

支持转发访问会话（FAS）：是

当您使用 IAM 用户或角色在中执行操作时 Amazon，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 Amazon Web Services 服务 向下游服务发出请求的请求。 Amazon Web Services 服务只有当服务收到需要与其他 Amazon Web Services 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amazon SNS 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon SNS 的功能。仅当 Amazon SNS 提供相关指导时才编辑服务角色。

Amazon SNS 的服务相关角色

支持服务相关角色：否

服务相关角色是一种链接到的服务角色。 Amazon Web Services 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 Amazon Web Services 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的Amazon 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 Amazon Simple Notification Service 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Amazon SNS 资源的权限。他们也无法使用 Amazon Web Services Management Console、 Amazon Command Line Interface (Amazon CLI) 或 Amazon API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 Amazon SNS 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅服务授权参考中的亚马逊简单通知服务的操作、资源和条件密钥。 ARNs

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon SNS 资源。这些操作可能会使 Amazon Web Services 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 Amazon 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的Amazon 托管策略。它们在你的版本中可用 Amazon Web Services 账户。我

们建议您通过定义针对您的用例的 Amazon 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略或工作职能的Amazon 托管式策略。

- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 Amazon Web Services 服务，例如 Amazon CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 Amazon Web Services 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon SNS 控制台

要访问 Amazon Simple Notification Service 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon SNS 资源的详细信息。 Amazon Web Services 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 Amazon CLI 或 Amazon API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 Amazon SNS 控制台，还要将 Amazon `ConsoleAccess` SNS `ReadOnly` Amazon 或托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

其他策略类型

Amazon 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)- SCPs 是指定组织或组织单位 (OU) 的最大权限的 JSON 策略 Amazon Organizations。 Amazon Organizations 是一项用于对您的企业拥有的多 Amazon Web Services 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP 限制成员账户中实体的权限，包括每个 Amazon Web Services 账户 root 用户。有关 Organization SCPs s 和的更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs) — RCPs 是 JSON 策略，您可以使用它来设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制成员账户中资源的权限，并可能影响身份（包括 Amazon Web Services 账户 根用户）的有效权限，无论他们是否属于您的组织。有关 Organizations 的更多信息 RCPs，包括 Amazon Web Services 服务 该支持的列表 RCPs，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 Amazon 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 Amazon CLI 或 Amazon API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
```

```
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon SNS 基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 Amazon SNS 的基于身份的策略示例

要查看 Amazon SNS 基于身份的策略的示例，请参阅适用于 Amazon Simple Notification Service 的基于身份的策略示例。

Amazon SNS 内基于资源的策略

支持基于资源的策略	是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 Amazon Web Services 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 Amazon Web Services 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

将基于身份的策略用于 Amazon SNS

亚马逊简单通知服务与 Amazon Identity and Access Management (IAM) 集成，因此您可以指定您的用户 Amazon Web Services 账户 可以使用亚马逊 SNS 资源执行哪些亚马逊 SNS 操作。您可以指定策略中的特定主题。例如，您可以使用变量创建一项 IAM 策略，此策略向您组织中的特定用户授予权限，允许其通过您的 Amazon Web Services 账户中的特定主题执行 `Publish` 操作。有关更多信息，请参阅 Using IAM 指南中的 Policy Variables。

> **Important**
>
> 结合使用 Amazon SNS 和 IAM 并不会改变您使用 Amazon SNS 的方式。Amazon SNS 操作没有发生变化，且没有关于用户和访问控制的新 Amazon SNS 操作。

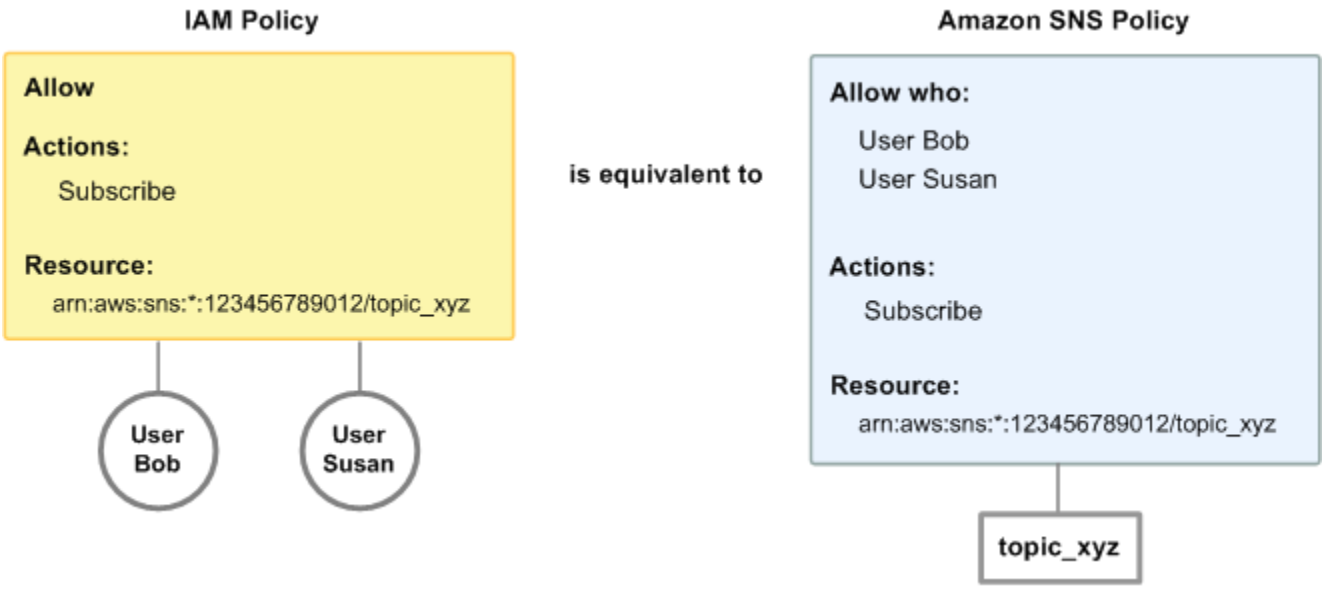
有关涉及 Amazon SNS 操作和资源的策略示例，请参阅 Amazon SNS 的策略示例。

IAM 和 Amazon SNS 策略一起使用

您可以利用 IAM 策略限制您的用户对 Amazon SNS 操作和主题的访问权限。IAM 策略只能限制您 Amazon 账户内的用户访问权限，而不能限制其他用户的访问权限 Amazon Web Services 账户。

您可对特定主题使用 Amazon SNS 策略，以限制谁能使用该主题（例如，谁能向主题发布消息，谁能订阅该主题等）。Amazon SNS 策略可以向其他 Amazon Web Services 账户人或您自己的用户授予访问权限。 Amazon Web Services 账户

您可以使用 IAM 策略、Amazon SNS 策略或此两者向您的用户授予访问 Amazon SNS 主题的权限。在绝大部分情况下，无论采用上述哪种方式，都可以得到同样的结果。例如，下图显示了等效的 IAM 策略和 Amazon SNS 策略。IAM 策略允许对您的 Amazon Web Services 账户 名为 topic_xyz 的主题进行 Amazon SNS `Subscribe` 操作。IAM 策略已附加到用户 Bob 和 Susan（这意味着 Bob 和 Susan 拥有策略中规定的权限）。 策略Amazon SNS同样也会向 Bob 和 Susan 授予对 topic_xyz 访问 `Subscribe` 的权限。



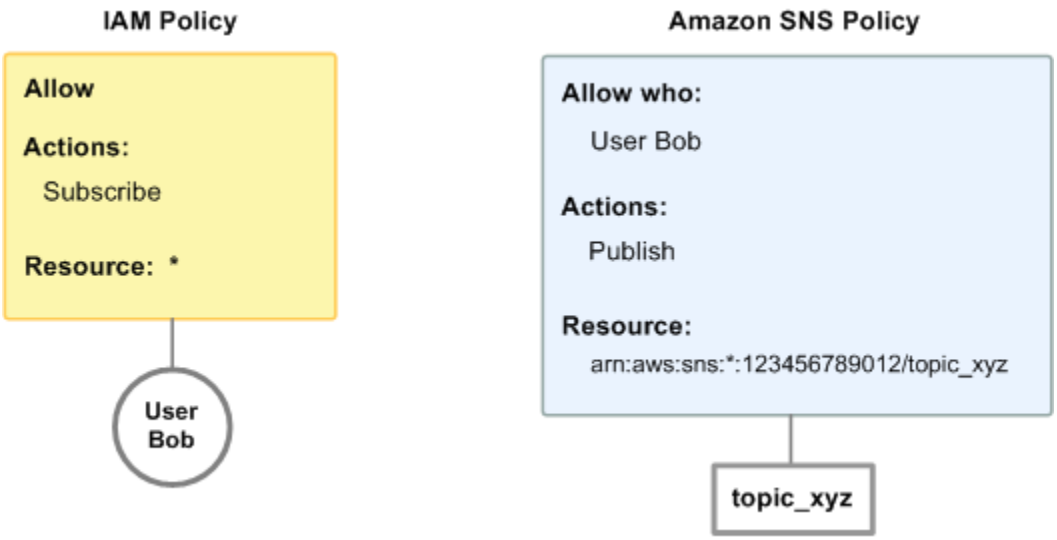
> **Note**
>
> 上述示例显示了不带任何条件的简单策略。您可以在上述任一策略中指定特定条件，并获得同样的结果。

Amazon IAM 和 Amazon SNS 策略之间有一个区别：亚马逊 SNS 策略系统允许您向 Amazon Web Services 账户其他策略授予权限，而 IAM 策略不允许。

将由您自己决定如何是否上述两种系统管理您的权限，您可以根据自身需求做出决定。以下示例展示这两种策略系统是如何共同运行的。

Example 1

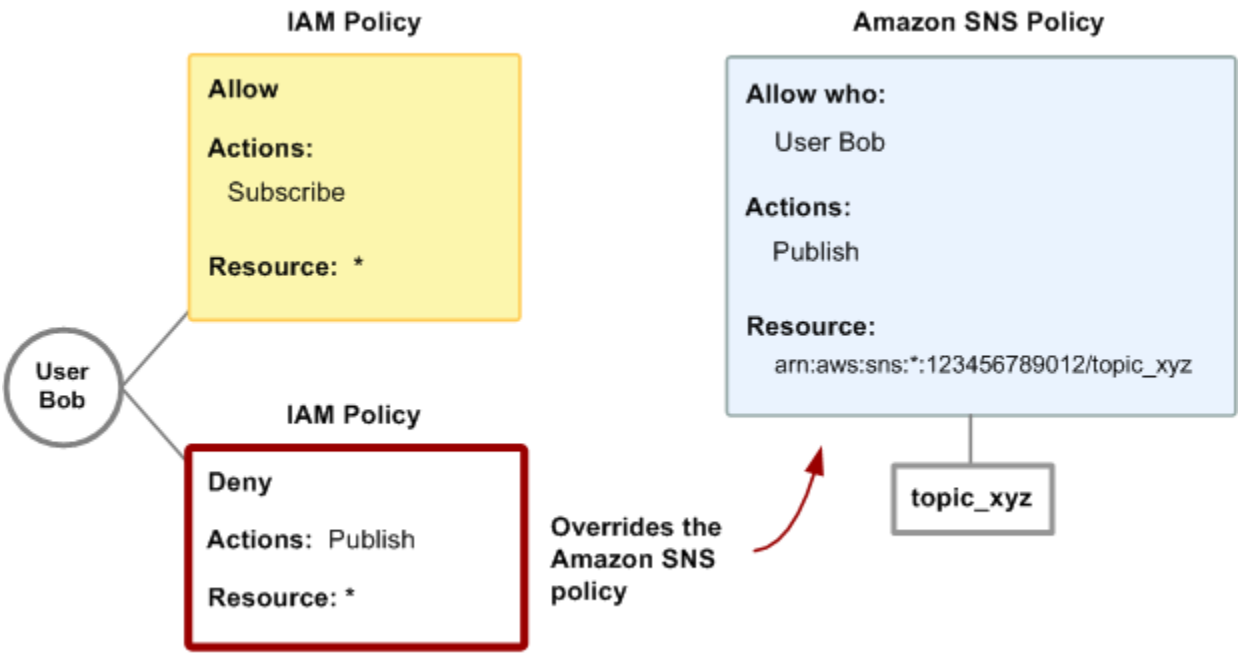
在本示例中，IAM 策略和 Amazon SNS 策略均会应用于 Bob。IAM 政策授予他处理任何主题 Amazon Web Services 账户的`Subscribe`权限，而 Amazon SNS 政策则授予他使用特定主题的权限 (top `Publish` ic_xyz)。下图阐明了这一概念。



如果 Bob 发送订阅 Amazon 账户中任何主题的请求，IAM 策略将允许该操作。如果 Bob 要发送向 topic_xyz 发布消息的请求，那么 Amazon SNS 策略将允许此操作。

Example 2

在本示例中，我们基于示例 1（其中，Bob 拥有两个适用于他的策略）来进行描述。让我们看看 Bob 无法向 topic_xyz 发布消息的情况，因此您要将其发布主题的功能全部删除。完成上述操作最简单的方法就是在所有主题中添加拒绝他访问 `Publish` 操作的 IAM 策略。第三项策略将覆盖 Amazon SNS 策略，该策略最初为其授予向 topic_xyz 发布的权限，覆盖的原因是显式拒绝始终覆盖允许策略（有关策略评估逻辑的更多信息，请参阅 评估逻辑）。下图阐明了这一概念。



有关涉及 Amazon SNS 操作和资源的策略示例，请参阅 Amazon SNS 的策略示例。

Amazon SNS 资源 ARN 格式

对于 Amazon SNS，主题是您可以在策略中指定的唯一资源类型。以下是主题的 Amazon Resource Name (ARN) 格式。

```
arn:aws:sns:region:account_ID:topic_name
```

有关的更多信息 ARNs，请访问 IAM 用户指南ARNs中的。

Example

以下是在 us-east-2 区域命名的 MyTopic 主题的 ARN，属于 123456789012。 Amazon Web Services 账户

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

Example

如果您 MyTopic 在 Amazon SNS 支持的每个不同区域中都有一个名为的主题，则可以使用以下 ARN 来指定主题。

```
arn:aws:sns:*:123456789012:MyTopic
```

您可以在主题名称中使用 * 和 ? 通配符。例如，以下可以参考由 Bob 创建、前缀为 bob_ 的所有主题。

```
arn:aws:sns:*:123456789012:bob_*
```

为方便起见，您创建主题时，Amazon SNS 将在响应中返回主题 ARN。

Amazon SNS API 操作

在 IAM 策略中，您可以指定任何由 Amazon SNS 提供的操作。但是，`ConfirmSubscription` 和 `Unsubscribe` 操作无需验证，这表示即使您在策略中指定了上述操作，IAM 也不会限制用户访问这些操作。

您在策略中指定的所有操作必须加上小写的字符串 `sns:` 为前缀。例如，您可以使用 `sns:*` 指定所有 Amazon SNS 操作。欲了解操作列表，请前往 Amazon Simple Notification Service API 参考。

Amazon SNS 策略密钥

Amazon SNS 实施了以下 Amazon 广泛的策略密钥，以及一些特定于服务的密钥。

有关各自支持的条件键的列表 Amazon Web Services 服务，请参阅 IAM 用户指南 Amazon Web Services 服务中的操作、资源和条件键。有关可以用于多个条件键的列表 Amazon Web Services 服务，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥。

Amazon SNS 使用下列服务特定密钥。使用策略中限制访问 `Subscribe` 请求的这些密钥。

- sns:endpoint—来自 `Subscribe` 请求或之前已确认的订阅的 URL、电子邮件地址或 ARN。通过字符串条件（请参阅 Amazon SNS 的策略示例）限制访问特定终端节点（例如 *@yourcompany.com）。
- sns:protocol—来自 `Subscribe` 请求或之前已确认的订阅的 `protocol` 值。与字符串条件一起使用（请参阅 Amazon SNS 的策略示例），以限制向特定传输协议发布消息（例如，https）。

Amazon SNS 的策略示例

本节演示几个控制用户访问 Amazon SNS 的单一策略。

> **Note**
>
> 未来，根据策略陈述的目标，Amazon SNS 可能会添加在逻辑上包含以下策略之一的新操作。

Example 1：允许群组创建和管理主题

在本示例中，我们创建授权访问 `CreateTopic`、`ListTopics`、`SetTopicAttributes` 和 `DeleteTopic` 的策略。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2：允许 IT 群组向特定主题发布消息

在本示例中，我们为 IT 创建群组，并将授权访问 `Publish` 的策略分配至相关特定主题上。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3：让用户 Amazon Web Services 账户 能够订阅主题

在示例中，我们创建授予访问 `Subscribe` 操作的权限的策略，以及针对 `sns:Protocol` 和 `sns:Endpoint` 策略密钥的字符串匹配条件。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
```

```
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "sns:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4：允许合作伙伴向特定主题发布消息

您可以利用 Amazon SNS 策略或 IAM 策略允许一位合作伙伴发布到特定主题。如果您的合作伙伴有 Amazon Web Services 账户，那么使用 Amazon SNS 政策可能会更方便。但是，合作伙伴公司中任何拥有 Amazon 安全凭证的人都可以发布有关该主题的消息。本示例假设您想要对特定人员进行访问限制（或应用程序限制）。要实现上述目的，您需要像对待您公司内的用户一样对待合作伙伴，并使用 IAM 策略而非 Amazon SNS 策略。

在此示例中，我们创建了一个名 WidgetCo 为代表合作伙伴公司的群组；我们为合作伙伴公司中需要访问权限的特定人员（或应用程序）创建一个用户；然后将该用户放入该群组。

然后，我们附加一个策略，授予该群组对名为的特定主题的`Publish`访问权限WidgetPartnerTopic。

我们还想阻止该 WidgetCo 小组对主题进行任何其他操作，因此我们添加了一份声明，拒绝允许除以外的任何主题以外`Publish`的任何其他主题进行任何 Amazon SNS 操作。 WidgetPartnerTopic只有在系统中的其他地方存在广泛策略（该策略向用户授予广泛访问 Amazon SNS 的权限）时，才需要执行此操作。

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
```

```
}
```

管理自定义 Amazon SNS IAM 策略

自定义 IAM 策略允许您为单个 IAM 用户、群组或角色指定权限，授予或限制对特定 Amazon 资源和操作的访问权限。在管理 Amazon SNS 资源时，自定义 IAM 策略允许您根据组织的安全和运营要求定制访问权限。

按照以下步骤管理 Amazon SNS 的自定义 IAM 策略：

1. 登录 Amazon Web Services Management Console 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 要创建新的自定义 IAM 策略，请选择创建策略并选择 SNS。 要编辑现有策略，请从列表中选择该策略，然后选择编辑策略。
4. 在策略编辑器中，定义访问 Amazon SNS 资源的权限。您可以根据自己的特定要求指定操作、资源和条件。
5. 要授予 Amazon SNS 操作的权限，请在您的 IAM 策略中加入相关的 Amazon SNS 操作，例如 `sns:Publish`、`sns:Subscribe` 和 `sns:DeleteTopic`。定义权限适用的 Amazon SNS 主题的 ARN（Amazon 资源名称）。
6. 指定应将策略附加到的 IAM 用户、群组或角色。您可以将策略直接附加到 IAM 用户或群组，也可以将其与 Amazon Web Services 服务 或应用程序使用的 IAM 角色相关联。
7. 查看 IAM 策略配置，确保其符合您的访问控制要求。验证后，保存策略更改。
8. 将自定义 IAM 策略附加到您 Amazon Web Services 账户中的相关 IAM 用户、群组或角色。这会向他们授予策略中定义的用于管理 Amazon SNS 资源的权限。

将临时安全凭证用于 Amazon SNS

Amazon Identity and Access Management (IAM) 允许您向需要访问您的 Amazon 资源的用户和应用程序授予临时安全证书。这些临时安全凭证主要用于 IAM 角色和通过 SAML 和 OpenID Connect（OIDC）等行业标准协议进行联合访问。

要有效地管理对 Amazon 资源的访问权限，必须了解以下关键概念：

- IAM 角色 — 角色用于委派对 Amazon 资源的访问权限。角色可以由诸如 Amazon 实 EC2 例、Lambda 函数之类的实体或其他用户担任。 Amazon Web Services 账户

- 联合用户-这些用户是通过外部身份提供商 (IdPs) 使用 SAML 或 OIDC 进行身份验证的用户。建议人类用户使用联合访问权限，而软件应用程序应使用 IAM 角色。
- 无@@ 处不在的角色 — 对于需要 Amazon 访问权限的外部应用程序，您可以使用 IAM Roles Anywhere 来安全地管理访问权限，而无需创建长期证书。

您可以使用临时安全凭证对 Amazon SNS 发出请求。 SDKs 和 API 库使用这些凭据计算必要的签名来验证您的请求。凭证过期的请求将被 Amazon SNS 拒绝。

有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色和向经过外部身份验证的用户提供访问权限（身份联合验证）。

Example HTTPS 请求示例

以下示例演示如何使用从 Amazon Security Token Service (STS) 获得的临时安全证书对 Amazon SNS 请求进行身份验证。

```
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>
```

对请求进行身份验证的步骤

1. 获取临时安全凭证 - 使用 Amazon STS 担任角色或获取联合用户凭证。这将为您提供访问密钥 ID、秘密访问密钥和安全令牌。
2. 构造请求 — 包括您的 Amazon SNS 操作所需的参数（例如 CreateTopic），并确保使用 HTTPS 进行安全通信。
3. 签署请求 - 使用 Amazon 签名版本 4 流程签署您的请求。这包括创建规范请求 string-to-sign，然后计算签名。有关 Amazon 签名版本 4 的更多信息，请参阅 Amazon EBS 用户指南中的使用签名版本 4 签名。
4. 发送请求 — 包括 X-Amz-Security-Token 在您的请求标头中将临时安全证书传递给 Amazon SNS。

Amazon SNS API 权限：操作和资源参考

以下列表提供了特定于访问控制的 Amazon SNS 实施的信息：

- 每个策略都必须仅覆盖单一主题（在编写一个策略时，请勿包括覆盖不同主题的语句）
- 每个策略必须有一个独立的策略`Id`
- 策略中的每个语句必须是一个独立的语句`sid`

策略配额

以下表格列举了策略语句的最大配额。

名称	最大配额
字节	30 kb
语句	100
委托人	1 到 200（0 无效。）
资源	1（0 无效。它的值应与策略主题的 ARN 相匹配。）

有效的 Amazon SNS 策略操作

Amazon SNS 支持以下表格所示的操作。

操作	描述
sns: AddPermission	授予向主题策略添加权限的权限。
sns: DeleteTopic	授予删除一个主题的权限。
sns: GetDataProtectionPolicy	授予权限以检索主题的数据保护策略。
sns: GetTopicAttributes	授予获取所有主题属性的权限。
sns: ListSubscriptionsByTopic	授予检索对特定主题的所有订阅的权限。

操作	描述
sns: ListTagsForResource	授予列出添加到特定主题的所有标签的权限。
sns:Publish	授予向主题或终端节点发布和批量发布内容的权限。有关更多信息，请参阅发布和PublishBatch《亚马逊简单通知服务 API 参考》。
sns: PutDataProtectionPolicy	授予权限以设置主题的数据保护策略。
sns: RemovePermission	授予在主题策略中取消任何权限的权限。
sns: SetTopicAttributes	授予设置一个主题的属性的权限。
sns:Subscribe	授予订阅一个主题的权限。

特定于服务的密钥

Amazon SNS 使用下列服务特定密钥。您可以在限制访问 `Subscribe` 请求的策略中使用它们。

- sns:endpoint—来自 `Subscribe` 请求或之前已确认的订阅的 URL、电子邮件地址或 ARN。与字符串条件一起使用（请参阅 Amazon SNS 的策略示例），以限制对某个特定终端节点的访问（例如，*@example.com）。
- sns:protocol—来自 `Subscribe` 请求或之前已确认的订阅的 `protocol` 值。与字符串条件一起使用（请参阅 Amazon SNS 的策略示例），以限制向特定传输协议发布消息（例如，https）。

> **Important**
>
> 当您通过 sns:Endpoint 使用一个策略控制访问时，注意 DNS 问题可能将来会影响终端节点的域名解析。

Amazon Simple Notification Service 身份和访问故障排查

您可以使用以下信息，帮助诊断和修复在使用 Amazon SNS 和 IAM 时可能遇到的常见问题。

我无权在 Amazon SNS 中执行操作

如果您收到一个错误，指明您无权执行某个操作，则必须更新策略以允许您执行该操作。

当 `mateojackson` 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `sns:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

在此情况下，Mateo 的策略必须更新以允许其使用 `sns:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 Amazon 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，指明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amazon SNS。

有些 Amazon Web Services 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon SNS 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 Amazon 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人访问我 Amazon Web Services 账户 的 Amazon SNS 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon SNS 是否支持这些功能，请参阅Amazon SNS 如何与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 Amazon Web Services 账户 ，请参阅 IAM 用户指南中的向您拥有 Amazon Web Services 账户 的另一个 IAM 用户提供访问权限。

- 要了解如何向第三方提供对您的资源的访问权限 Amazon Web Services 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 Amazon Web Services 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amazon SNS 中的日志记录和监控

Amazon SNS 允许您通过记录 API 调用和监控主题来跟踪 CloudTrail和监控消息活动。 CloudWatch这些工具可帮助您深入了解消息传送，排查问题，并确保消息收发工作流程的正常运行。本主题包含以下内容：

- 使用记录 Amazon SNS API 调用 Amazon CloudTrail。通过此日志记录，您可以跟踪对您的 Amazon SNS 主题执行的操作，例如主题创建、订阅管理和消息发布。通过分析 CloudTrail 日志，您可以确定谁提出了特定 API 请求以及这些请求是何时发出的，从而帮助您审计 Amazon SNS 的使用情况并对其进行故障排除。
- 使用监控亚马逊 SNS 主题 CloudWatch。 CloudWatch 提供的指标使您可以实时观察 Amazon SNS 主题的性能和运行状况。根据这些指标设置警报，使您能够及时响应任何异常情况，例如传送失败或消息延迟过长。此监控功能可确保您能通过主动解决潜在问题来保持基于 SNS 的消息系统的可靠性。

使用记录 Amazon SNS API 调用 Amazon CloudTrail

Amazon SNS 与Amazon CloudTrail一项服务集成，该服务提供用户、角色或角色所执行操作的 Amazon Web Services 服务记录。 CloudTrail 将 SNS 的所有 API 调用捕获为事件。捕获的调用包括来自 SNS 控制台的调用和对 SNS API 操作的代码调用。使用收集的信息 CloudTrail，您可以确定向 SNS 发出的请求、发出请求的 IP 地址、发出请求的时间以及其他详细信息。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon Web Services 服务发出。

CloudTrail 在您创建账户 Amazon Web Services 账户 时在您的账户中处于活动状态，并且您自动可以访问 CloudTrail 活动历史记录。 CloudTrail 事件历史记录提供了过去 90 天中记录的管理事件的可查看、可搜索、可下载且不可变的记录。 Amazon Web Services 区域有关更多信息，请参阅《Amazon CloudTrail 用户指南》中的 “使用 CloudTrail 事件历史记录”。查看活动历史记录不 CloudTrail收取任何费用。

要持续记录 Amazon Web Services 账户 过去 90 天内的事件，请创建跟踪或 CloudTrailLake 事件数据存储。

CloudTrail 步道

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。使用创建的所有跟踪 Amazon Web Services Management Console 都是多区域的。您可以通过使用 Amazon CLI创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可以捕获账户 Amazon Web Services 区域 中的所有活动。如果您创建单区域跟踪，则只能查看跟踪的 Amazon Web Services 区域中记录的事件。有关跟踪的更多信息，请参阅《Amazon CloudTrail 用户指南》中的为您的 Amazon Web Services 账户创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以免费将正在进行的管理事件的一份副本传送到您的 Amazon S3 存储桶，但会收取 Amazon S3 存储费用。 CloudTrail 有关 CloudTrail 定价的更多信息，请参阅Amazon CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

CloudTrail 湖泊事件数据存储

CloudTrail L@@ ak e 允许您对事件运行基于 SQL 的查询。 CloudTrail Lake 将基于行的 JSON 格式的现有事件转换为 Apache ORC 格式。ORC 是一种针对快速检索数据进行优化的列式存储格式。事件将被聚合到事件数据存储中，它是基于您通过应用高级事件选择器选择的条件的不可变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅Amazon CloudTrail 用户指南中的使用 Amazon CloudTrail Lake。

CloudTrail 湖泊事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的定价选项。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅Amazon CloudTrail 定价。

中的 SNS 数据事件 CloudTrail

数据事件可提供对资源或在资源中所执行资源操作（例如，读取或写入 Amazon S3 对象）的相关信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下， CloudTrail 不记录数据事件。 CloudTrail 事件历史记录不记录数据事件。

记录数据事件将收取额外费用。有关 CloudTrail 定价的更多信息，请参阅Amazon CloudTrail 定价。

您可以使用 CloudTrail 控制台、 Amazon CLI或 CloudTrail API 操作记录 SNS 资源类型的数据事件。有关如何记录数据事件的更多信息，请参阅《Amazon CloudTrail 用户指南》中的使用 Amazon Web Services Management Console记录数据事件和使用 Amazon Command Line Interface记录数据事件。

下表列出了您可以记录数据事件的 SNS 资源类型。数据事件类型（控制台）列显示要从控制 CloudTrail 台上的数据事件类型列表中选择的值。res ources.type 值列显示该`resources.type`值，您将在使用或配置高级事件选择器时指定该值。 Amazon CLI CloudTrail APIs“ APIs 记录到的数据 CloudTrail” 列显示了 CloudTrail 针对该资源类型记录的 API 调用。

数据事件类型（控制台）	resources.type 值	数据 APIs 已记录到 CloudTrail
SNS 主题	`AWS::SNS::Topic`	• `Publish` • `PublishBatch`
SNS 平台端点	`AWS::SNS::PlatformEndpoint`	• `Publish` 有关更多详细信息，请参阅 Amazon CloudTrail API 参考`AdvancedEventSelector` 中的。

> **Note**
>
> 未记录 SNS 资源类型`AWS::SNS::PhoneNumber`。 CloudTrail

您可以将高级事件选择器配置为在 `eventName`、`readOnly` 和 `resources.ARN` 字段上进行筛选，从而仅记录那些对您很重要的事件。有关这些字段的更多信息，请参阅 AdvancedFieldSelector《Amazon CloudTrail API 参考》中的 。

有关记录数据事件的信息，请参阅 CloudTrail 用户指南 Amazon CLI 中的使用记录数据事件 Amazon Web Services Management Console 和使用记录数据事件。

中的 SNS 管理事件 CloudTrail

管理事件提供有关对中的资源执行的管理操作的信息 Amazon Web Services 账户。这些也称为控制面板操作。默认情况下， CloudTrail 记录管理事件。

Amazon SNS 将以下 SNS 控制平面操作记录 CloudTrail 为管理事件。

- 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**
>
> 如果您未登录到 Amazon Web Services（未经身份验证的模式），并且调用了ConfirmSubscription或Unsubscribe操作，则这些操作将不会被登录到 CloudTrail。例如，当您选择电子邮件通知中提供的链接确认对某一主题的待确认订阅时，会在未验证模式下调用 ConfirmSubscription 操作。在此示例中，ConfirmSubscription操作不会记录到 CloudTrail。

SNS 活动示例

事件代表来自任何来源的单个请求，包括有关所请求的 API 操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此事件不会按任何特定顺序出现。

以下示例显示了一个演示**ListTopicsCreateTopic**、和**DeleteTopic**操作 CloudTrail 的事件。

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

以下示例显示了一个演示**Publish**操作 CloudTrail 的事件。

```
{
  "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"
  }
}
```

以下示例显示了一个演示**PublishBatch**操作 CloudTrail 的事件。

```
{
  "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"
```

```
  }
}
```

有关 CloudTrail 录音内容的信息，请参阅《Amazon CloudTrail 用户指南》中的CloudTrail录制内容。

使用监控亚马逊 SNS 主题 CloudWatch

Amazon SNS 和亚马逊 CloudWatch 集成在一起，因此您可以收集、查看和分析每条有效的亚马逊 SNS 通知的指标。配置 CloudWatch Amazon SNS 后，您可以更好地了解您的 Amazon SNS 主题、推送通知和短信发送的表现。例如，您可以设置一个告警，以便在某个 Amazon SNS 指标（如 `NumberOfNotificationsFailed`）达到指定阈值时向您发送电子邮件通知。有关 Amazon SNS 发送到的所有指标的列表 CloudWatch，请参阅。Amazon SNS 指标有关 Amazon SNS 推送通知的更多信息，请参阅 使用 Amazon SNS 发送移动推送通知。

> **Note**
>
> 系统会自动收集您 CloudWatch 为 Amazon SNS 主题配置的指标，并每隔 1 分钟推送 CloudWatch 一次。这些指标是针对所有符合活跃 CloudWatch 指南的话题收集的。自该主题的上次活动（即任何 API 调用）起，该主题最多在六小时内被视为处于活动状态。CloudWatch
> 中报告的亚马逊 SNS 指标不收取任何费用 CloudWatch；这些指标是作为亚马逊 SNS 服务的一部分提供的。

查看 Amazon SNS 的 CloudWatch 指标

您可以使用 CloudWatch 控制台、 CloudWatch自己的命令行界面 (CLI) 或使用 API 以编程方式监控 Amazon SNS 的指标。 CloudWatch 以下过程展示如何使用 Amazon Web Services Management Console访问指标。

使用 CloudWatch 控制台查看指标

1. 登录 CloudWatch 控制台。
2. 在导航面板上，选择 Metrics。
3. 在 All metrics (全部指标) 选项卡上，选择 SNS，然后选择以下维度之一：
   - Country, SMS Type (国家/地区，短信类型)
   - PhoneNumber

- Topic Metrics (主题指标)
- Metrics with no dimensions (无维度指标)

4. 要查看详细信息，请选择特定项目。例如，如果您选择 “主题指标”，然后选择 NumberOfMessagesPublished，则会显示 6 小时时间范围内在 1 分钟内发布的 Amazon SNS 消息的平均数量。
5. 要查看 Amazon SNS 使用情况指标，请在 All metrics（所有指标）选项卡上选择 Usage（使用情况），然后选择 Target Amazon SNS usage metric（目标 Amazon SNS 使用情况指标），例如 `NumberOfMessagesPublishedPerAccount`。

为 Amazon SNS 指标设置 CloudWatch 警报

CloudWatch 还允许您在指标达到阈值时设置警报。例如，您可以为指标设置警报 NumberOfNotificationsFailed，以便在采样周期内达到指定的阈值时，系统会发送一封电子邮件通知来通知您该事件。

使用 CloudWatch 控制台设置警报

1. 登录 Amazon Web Services Management Console 并打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 选择警报，然后选择创建警报按钮。这样会启动“Create Alarm”向导。
3. 滚动 Amazon SNS 指标找到您想要设置告警的指标的位置。选择该指标创建一个警报并选择继续。
4. 填写指标的名称、描述、阈值、时间值，然后选择继续。
5. 按照报警器说明选择“Alarm”。如果您 CloudWatch 想在达到警报状态时向您发送电子邮件，请选择现有的 Amazon SNS 主题或选择创建新电子邮件主题。如果您选择 新建电子邮件主题，则可以为新主题设置名称和电子邮件地址。此清单将会被保存下来并在将来报警器的下列框显示。选择继续。

   > **Note**
   >
   > 如果您使用新建电子邮件主题来创建新的 Amazon SNS 主题，那么电子邮件地址必须通过验证才能接收通知。当报警器进入报警状态时，才发送电子邮件。如果在电子邮件地址验证之前报警状态发生变化，那么他们不会收到通知。

6. 此时，“Create Alarm”向导会给您一次机会检查您准备创建的报警器。如果你想做一些变动，那么您可使用右边的“Edit”链接。如果您满意，请选择创建警报。

有关使用 CloudWatch 和警报的更多信息，请参阅CloudWatch文档。

Amazon SNS 指标

Amazon SNS 向发送以下指标。 CloudWatch

命名空间	指标	描述
Amazon/SNS	`NumberOfMessagesPublished`	发布到您的 Amazon SNS 主题的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum
Amazon/SNS	`NumberOfNotificationsDelivered`	从您的 Amazon SNS 主题成功传输到订阅终端节点的消息数量。 要想成功传输，终端节点的订阅必须接受消息。在以下两种情况下订阅可接受消息：a.) 它缺少筛选策略或 b.) 其筛选策略中包含的属性与分配给消息的属性相匹配。如果订阅拒绝消息，则传输尝试不会计入此指标。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum
Amazon/SNS	`NumberOfNotificationsFailed`	Amazon SNS 传输失败的消息数量。 对于 Amazon SQS、电子邮件、SMS 或移动推送终端节点，当 Amazon SNS 停止尝试传送消息时

命名空间	指标	描述
		，此指标会递增 1。对于 HTTP 或 HTTPS 终端节点，此指标包括每次失败的传输尝试（含初始尝试之后的重试）。对于所有其他终端节点，消息传输失败时计数增加 1（不考虑尝试次数）。 此指标不包括被订阅筛选策略拒绝的消息。 您可以控制 HTTP 终端节点的重试次数。有关更多信息，请参阅 Amazon SNS 消息传输重试。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average
Amazon/SNS	`NumberOfNotificationsFilteredOut`	被订阅筛选策略拒绝的消息数量。如果消息属性与策略属性不匹配，筛选策略会拒绝消息。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average

命名空间	指标	描述
Amazon/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	被基于属性的筛选的订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average
Amazon/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	被基于有效负载的筛选的订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average
Amazon/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	由于消息属性无效（例如属性 JSON 格式不正确）而被订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average

命名空间	指标	描述
Amazon/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	由于消息没有属性而被订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average
Amazon/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	由于消息正文对于筛选条件无效（例如无效的 JSON 消息正文）而被订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average
Amazon/SNS	`NumberOfNotificationsRedrivenToDlq`	已移动到死信队列的消息量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average
Amazon/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	无法移动到死信队列中的消息量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Sum、Average

命名空间	指标	描述
Amazon/SNS	`PublishSize`	已发布消息的大小。 单位：字节 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：Minimum、Maximum、Average 和 Count
Amazon/SNS	`SMSMonthToDateSpentUSD`	当前日历月开始以来您因发送 SMS 消息累积产生的费用。 您可以为此指标设置警报，以了解您的 month-to-date费用何时接近账户的每月短信支出配额。如果 Amazon SNS 确定发送 SMS 消息产生的费用会超过此配额，它会在几分钟内停止发布 SMS 消息。 有关设置您的每月 SMS 支出配额的信息，或有关向 Amazon请求提高支出配额的信息，请参阅设置 Amazon SNS 中的短信收发首选项。 单位：美元 有效维度：无 有效统计数据：Sum

命名空间	指标	描述
Amazon/SNS	`SMSSuccessRate`	SMS 消息传输的成功率。 单位：计数 有效尺寸： PhoneNumber 有效统计数据：Sum、Average、Data Samples

Amazon SNS 指标的维度

Amazon 简单通知服务会将以下维度发送至 CloudWatch。

维度	描述
`Application`	对应用程序对象进行筛选，这些对象表示在支持的推送通知服务之一（例如和 FCM）中注册的应用程序 APNs 和设备。
`Application,Platform`	对应用程序和平台对象进行筛选，其中平台对象用于支持的推送通知服务，例如 APNs 和 FCM。
`Country`	用于 SMS 消息的目标国家或地区的筛选条件。代表该国家或地区的是其 ISO 3166-1 α-2 代码。
`PhoneNumber`	当您将 SMS 直接发布到电话号码（无主题）时，筛选电话号码。
`Platform`	筛选推送通知服务的平台对象，例如 APNs 和 FCM。
`TopicName`	筛选 Amazon SNS 主题名称。
`SMSType`	SMS 消息的消息类型的筛选条件。可以为 promotional 或 transactional。

Amazon SNS 使用情况指标

Amazon 简单通知服务向发送以下使用量指标 CloudWatch。

命名空间	服务	指标	资源	类型	描述
Amazon/Usage	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	资源	• 在您的账户中发布到您的 Amazon SNS 主题的消息数量。Amazon • 单位：无 • 有效统计数据：Sum
Amazon/Usage	SNS	ResourceCount	ApproximateNumberOfTopics	资源	• 您 Amazon 账号中话题的大致数量。 • 单位：无 • 有效统计数据：Average、Minimum、Maximum、Sum
Amazon/Usage	SNS	ResourceCount	ApproximateNumberOfFilterPolicies	资源	• 您的 Amazon 账户中的大致筛选条件策略数量。 • 单位：无 • 有效统计数据：Average、Minimum、Maximum、Sum

命名空间	服务	指标	资源	类型	描述
Amazon/Usage	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	资源	• 您 Amazon 账户中待处理订阅的大致数量。 • 单位：无 • 有效统计数据：Average、Minimum、Maximum、Sum

命名空间	服务	指标	资源	类型	描述
Amazon/Usage	SNS	CallCount	• AddPermission • CheckIfPhoneNumberIsOptedOut • CreatePlatformApplication • CreatePlatformEndpoint • ConfirmSubscription • CreateSMSSandboxPhoneNumber • CreateTopic • DeleteEndpoint • DeletePlatformApplication • DeleteSMSSandboxPhoneNumber • DeleteTopic	API	• 您 Amazon 账户中选定的亚马逊 SNS API 的 API 调用次数。 • 尾部不允许有内容。 单位：无 • 有效统计数据：Sum

命名空间	服务	指标	资源	类型	描述
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

命名空间	服务	指标	资源	类型	描述
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

命名空间	服务	指标	资源	类型	描述
			• Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Amazon SNS 的合规性验证

要了解是否属于特定合规计划的范围，请参阅Amazon Web Services 服务 “” Amazon Web Services 服务 中的 “按合规计划划分的范围”，然后选择您感兴趣的合规计划。 Amazon Web Services 服务 有关一般信息，请参阅合规计划。

您可以使用下载第三方审计报告 Amazon Artifact。有关更多信息，请参阅中的 “下载报告” Amazon Artifact。

您在使用 Amazon Web Services 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 Amazon 提供了以下资源来帮助实现合规性：

- Security & Compliance：这些解决方案实施指南讨论了架构考虑因素，并提供了部署安全性和合规性功能的步骤。
- 合规资源 — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 使用Amazon Config 开发人员指南中的规则评估资源 — 该 Amazon Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- Amazon Security Hub— 这 Amazon Web Services 服务 提供了您内部安全状态的全面视图 Amazon。Security Hub 通过安全控制措施评估您的 Amazon 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控制措施的列表，请参阅 Security Hub 控制措施参考。
- Amazon GuardDuty — 它通过监控您的 Amazon Web Services 账户环境中是否存在可疑和恶意活动，来 Amazon Web Services 服务 检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。

Amazon SNS 中的恢复能力

通过利用围绕可用区的 Amazon 全球基础设施，确保 Amazon SNS 的弹性。 Amazon Web Services 区域 Amazon Web Services 区域 提供物理分离和隔离的可用区，通过低延迟、高吞吐量和高度冗余的网络连接。这种架构允许在可用区之间进行无缝失效转移，而不会中断，与传统的数据中心基础架构相比，应用程序和数据库本质上更具容错性和可扩展性。通过使用可用区，Amazon SNS 订阅用户可以从增强的可用性和可靠性中受益，即使存在潜在中断，也能保证消息传送。有关 Amazon Web Services 区域 和可用区的更多信息，请参阅Amazon 全球基础设施。

此外，对 Amazon SNS 主题的订阅可配置传输重试和死信队列，从而自动处理瞬时故障，确保消息可靠地到达预定目标。

Amazon SNS 还支持消息筛选和消息属性，这使您可以根据其特定应用场景定制韧性策略，从而增强应用程序的整体稳定性。

Amazon SNS 中的基础设施安全性

作为一项托管服务，Amazon SNS 受《安全、身份与合规最佳实践》文档中的 Amazon 全球网络安全程序的保护。

使用 Amazon API 操作通过网络访问亚马逊 SNS。客户端必须支持传输层安全性（TLS）1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。

您必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service (Amazon STS) 生成用于签名请求的临时安全凭证。

您可以从任何网络位置调用这些 API 操作，但 Amazon SNS 支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 Amazon SNS 策略来控制来自特定亚马逊 VPC 终端节点或特定终端节点的访问。 VPCs这将有效隔离在 Amazon SNS 主题中仅从一个特定 VPC 到给定 Amazon 队列的网络访问。有关更多信息，请参阅 仅限从特定 VPC 终端节点发布到 Amazon SNS 主题。

Amazon SNS 的安全最佳实践

Amazon 为 Amazon SNS 提供了许多安全功能。在您自己的安全策略的上下文中查看这些安全功能。

> **Note**
>
> 有关这些安全功能的指南适用于常见的应用场景和实施。我们建议您在特定应用场景、架构和威胁模型的上下文中查看这些最佳实践。

预防性最佳实践

以下是针对 Amazon SNS 的预防性安全最佳实践。

主题

- 确保主题不可公开访问
- 实施最低权限访问
- 对需要 Amazon SNS 访问权限的应用程序和 Amazon 服务使用 IAM 角色
- 实施服务器端加密
- 实施传输中数据加密
- 考虑使用 VPC 终端节点来访问 Amazon SNS
- 确保订阅未配置为提供原始 http 终端节点

确保主题不可公开访问

除非您明确要求互联网上的任何人能够读取或写入您的 Amazon SNS 主题，否则应确保您的主题不可公开访问（世界上所有人或任何经过身份验证的 Amazon 用户都可以访问）。

- 避免创建 `Principal` 设置为 `""` 的策略。
- 避免使用通配符 (*)。相反，对一个特定用户或多个用户命名。

实施最低权限访问

授予权限后，您可以决定这些权限的接收者、权限所针对的主题以及要允许这些主题使用的特定 API 操作。实施最低权限原则对于降低安全风险至关重要。它还有助于减少错误或恶意意图的负面影响。

遵循授予最低权限的标准安全建议。也就是说，只授予执行特定任务所需的权限。您可以通过使用与用户访问相关的安全策略组合来实施最低权限。

Amazon SNS 使用发布者-订阅者模型，需要三种类型的用户账户访问：

- 管理员 – 用于创建、修改和删除主题的访问权。管理员还控制主题策略。
- 发布者 – 用于向主题发送消息的访问权。
- 订阅者 – 用于订阅主题的访问权。

有关详细信息，请参阅以下章节：

- Amazon SNS 中的 Identity and Access Management
- Amazon SNS API 权限：操作和资源参考

对需要 Amazon SNS 访问权限的应用程序和 Amazon 服务使用 IAM 角色

要使应用程序或 Amazon 服务（例如亚马逊 EC2）访问亚马逊 SNS 主题，它们必须在 Amazon API 请求中使用有效的 Amazon 凭证。由于这些证书不会自动轮换，因此您不应将 Amazon 凭证直接存储在应用程序或 EC2 实例中。

您应该使用 IAM 角色来管理需要访问 Amazon SNS 的应用程序或服务的临时凭证。使用角色时，您无需向 EC2 实例或 Amazon 服务分配长期证书（例如用户名、密码和访问密钥），例如 Amazon Lambda。相反，该角色提供临时权限，供应用程序在调用其他 Amazon 资源时使用。

有关更多信息，请参阅 IAM 用户指南中的 IAM 角色和角色的常见场景：用户、应用程序和服务。

实施服务器端加密

要减少数据泄漏问题，可以通过静态加密，使用存储在与消息存储位置不同的位置的密钥来对消息进行加密。服务器端加密 (SSE) 提供静态数据加密。Amazon SNS 在存储消息时将在消息级别对数据进行加密，并在您访问消息时为您解密消息。SSE 使用中管理的密钥 Amazon Key Management Service。当您对请求进行身份验证并具有访问权限时，访问加密主题和未加密主题之间没有区别。

有关更多信息，请参阅 使用服务器端加密保护 Amazon SNS 数据安全 和 管理 Amazon SNS 加密密钥和成本。

实施传输中数据加密

可以但不建议使用 HTTP 发布在传输过程中未加密的消息。但是，当使用对主题进行静态加密时 Amazon KMS，需要使用 HTTPS 来发布消息，以确保静态和传输中的消息都要加密。虽然该主题不会自动拒绝 HTTP 消息，但必须使用 HTTPS 来维持安全标准。

Amazon 建议您使用 HTTPS 而不是 HTTP。使用 HTTPS 时，消息将在传输过程中自动加密，即使 SNS 主题本身没有加密。如果没有 HTTPS，基于网络的攻击者就可以窃听网络流量或使用诸如之类的攻击来操纵网络流量。 man-in-the-middle

要仅通过 HTTPS 强制实施加密连接，请将 `aws:SecureTransport` 条件添加到已附加到未加密 SNS 主题的 IAM 策略中。这会强制消息发布者使用 HTTPS 而不是 HTTP。您可以使用以下示例策略作为指导：

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:1234567890:test-topic"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

考虑使用 VPC 终端节点来访问 Amazon SNS

如果您的主题必须能够与您交互，但这些主题绝对不能暴露在 Internet 上，则可使用 VPC 终端节点将主题访问限制为仅访问特定 VPC 中的主机。您可以使用主题策略来控制从特定 Amazon VPC 终端节点或特定终端节点访问主题的权限 VPCs。

Amazon SNS VPC 终端节点提供两种方式来控制对消息的访问：

- 您可以控制允许通过特定 VPC 端点访问的请求、用户或组。
- 您可以使用主题策略控制哪些 VPCs 或 VPC 终端节点可以访问您的主题。

有关更多信息，请参阅创建终端节点 和为 Amazon SNS 创建 Amazon VPC 终端节点策略。

确保订阅未配置为提供原始 http 终端节点

避免将订阅配置为传输到原始 http 终端节点。始终将订阅传输到终端节点域名。例如，配置为向终端节点 `http://1.2.3.4/my-path` 传输的订阅应该更改为 `http://my.domain.name/my-path`。

使用 Amazon SNS 主题疑难解答 Amazon X-Ray

Amazon X-Ray 收集有关您的应用程序所处理的请求的数据，并允许您查看和筛选数据，以确定潜在的问题和优化机会。对于对应用程序的任何跟踪请求，您可以查看有关请求、响应以及应用程序对下游 Amazon 资源、微服务、数据库和 HTTP Web APIs 的调用的详细信息。

您可以将 X-Ray 与 Amazon SNS 结合使用来跟踪和分析经由您应用程序的消息。您可以使用 Amazon Web Services Management Console 来查看 Amazon SNS 与您的应用程序使用的其他服务之间的连接图。您还可以使用控制台查看指标，例如平均延迟和故障率。有关更多信息，请参阅 Amazon X-Ray 开发人员指南中的 Amazon SNS 和 Amazon X-Ray。

Amazon SNS 中的主动跟踪

用于跟踪和分析用户通过您的亚马逊 SNS 主题传递 Amazon X-Ray 到亚马逊 Data Firehose、Amazon S Amazon LambdaQS 和 HT TP/S 终端节点订阅的请求。

使用 X-Ray，您可以 end-to-end查看每个请求，从而使您能够：

- 确定什么叫你的 Amazon SNS 话题以及订阅的下游有哪些服务。
- 分析延迟，例如：
  - 处理之前在 Amazon SNS 主题上花费的时间。
  - 每个订阅终端节点的交付时间。

> **Important**
>
> 订阅数很多的 Amazon SNS 主题可能达到大小限制，无法完全跟踪。有关跟踪文档大小限制的信息，请参阅《 Amazon 一般参考》中的 X 射线服务配额。

如果您从正在跟踪的服务中调用 Amazon SNS API，则 Amazon SNS 会传递跟踪，即使在 API 上未启用 X-Ray 跟踪也是如此。

Amazon SNS 对于标准主题和 FIFO 主题都支持 X-Ray 跟踪。您可以使用 Amazon SNS 控制台、Amazon SNS `SetTopicAttributes` API、Amazon Simple Notification Service CLI 参考或 Amazon CloudFormation 为 Amazon SNS 主题启用 X-Ray。

要了解有关将 Amazon SNS 与 X-Ray 结合使用的更多信息，请参阅《 Amazon X-Ray 开发人员指南》中的 Amazon SNS 和 Amazon X-Ray。

主动跟踪权限

使用 Amazon SNS 控制台时，Amazon SNS 会尝试为 Amazon SNS 主题创建调用 X-Ray 所需的权限。如果您没有足够的权限使用 Amazon SNS 控制台，则尝试可能会被拒绝。有关更多信息，请参阅Amazon SNS 中的 Identity and Access Management 和用于 Amazon SNS 访问控制的示例案例。

使用 CLI 时，必须手动配置权限。这些权限是使用资源策略配置的。有关在 X-Ray 中使用所需权限的更多信息，请参阅 Amazon SNS 和 Amazon X-Ray。

使用控制台对 Amazon SNS 主题启用主动跟踪 Amazon

在 Amazon SNS 主题上启用主动跟踪后，它会读取跟踪 ID，根据跟踪 ID 向客户发送数据，并将跟踪 ID 传播到下游服务。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题或创建一个新主题。有关创建主题的详细信息，请参阅创建 Amazon SNS 主题。
3. 在创建主题页面的详细信息部分，选择主题类型：FIFO或标准。
   a. 输入主题的名称。
   b. （可选）输入主题的显示名称。
4. 展开 Active tracing（主动跟踪），然后选择 Use active tracing（使用主动跟踪）。

为亚马逊 SNS 主题启用 X-Ray 后，您可以使用 X-Ray 服务地图查看该主题的 end-to-end跟踪和服务地图。

使用软件开发工具包对 Amazon SNS 主题启用主动跟踪 Amazon

以下代码示例展示了如何使用适用于 Java 的 S Amazon DK 在 Amazon SNS 主题上启用主动跟踪。

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

使用 CLI 在 Amazon SNS 主题上启用主动跟踪 Amazon

以下代码示例显示了如何使用 CLI Amazon 在 Amazon SNS 主题上启用主动跟踪。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

使用在 Amazon SNS 主题上启用主动跟踪 Amazon CloudFormation

以下 Amazon CloudFormation 堆栈显示了如何对 Amazon SNS 主题启用主动跟踪。

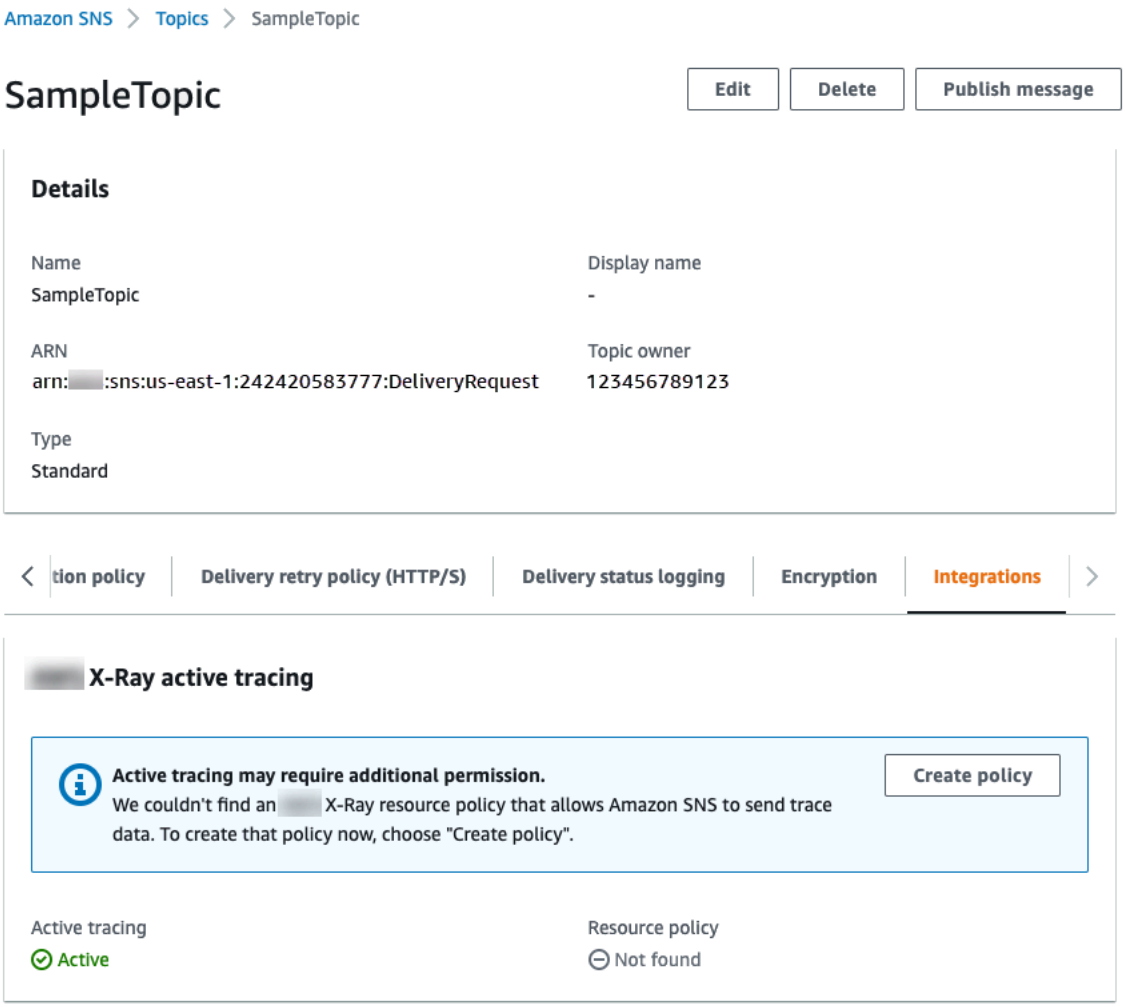
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

验证为您的主题启用了主动跟踪

您可以使用 Amazon SNS 控制台来验证是否对您的主题启用了主动跟踪，或者资源策略是否添加失败。

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页上，选择一个主题。
4. 请选择 Integrations（集成）选项卡。

   启用主动跟踪后，会显示绿色的 Active（活动）图标。
5. 如果您启用了主动跟踪但没有看到资源策略已添加，请选择 Create policy（创建策略）以添加所需的额外权限。



测试主动跟踪

1. 登录 Amazon SNS 控制台。

2. 创建 Amazon SNS 主题。有关如何执行该操作的详细信息，请参阅 要使用创建主题 Amazon Web Services Management Console。
3. 展开 Active tracing（主动跟踪），然后选择 Use active tracing（使用主动跟踪）。
4. 向 Amazon SNS 主题发布消息。有关如何执行该操作的详细信息，请参阅 要使用 Amazon Web Services Management Console将消息发布到 Amazon SNS 主题。
5. 使用 X-Ray 服务地图查看该主题的 end-to-end跟踪和服务地图。

Amazon SNS 文档历史记录

下表介绍了对 Amazon Simple Notification Service 开发人员指南的最近更改。

服务功能有时会逐步推广到提供服务的 Amazon 区域。我们仅在第一次发布时更新了此文档。我们不提供有关区域可用性的信息，也不会宣布后续区域支持情况。有关服务功能的区域可用性以及订阅更新通知的信息，请参阅新增内容 Amazon？ 。

变更	说明	日期
增加了对双堆栈（IPv4 和 IPv6）端点的支持	Amazon SNS 现在支持 IPv6 API 请求，允许客户端使用 IPv4 IPv6、或双堆栈连接到公共终端节点。	2025 年 4 月 3 日
亚马逊 SNS 为亚马逊 SNS FIFO 主题添加了该`FifoThroughputScope` 属性	Amazon SNS 支持该 FifoThroughputScope 属性，该属性指定适用于 FIFO 主题的吞吐量配额和重复数据删除行为。有效值为 `Topic` 或 `MessageGroup` 。	2025 年 1 月 21 日
`AmazonSNSFullAccess` 和 `AmazonSNSReadOnlyAccess` 托管式策略更新	Amazon SNS 添加了新的权限 AmazonSNSFullAccess 以及 AmazonSNSReadOnlyAccess 托管策略，允许通过其他方式访问 Amazon SNS。 Amazon Web Services Management Console	2024 年 9 月 24 日
Amazon SNS 与 Amazon End User Messaging SMS 用于发送短信的集成	Amazon SNS 支持新功能，例如短信资源管理、双向消息、精细资源权限、国家/地区封禁规则以及所有 Amazon 短信的集中计费，而无需对 Amazon SNS 使用的配置或全	2024 年 9 月 24 日

	球 Amazon 短信网络进行任何更改。	
加拿大西部（卡尔加里）支持 FIFO 主题	Amazon SNS 支持加拿大西部（卡尔加里）的 FIFO 主题。	2024 年 3 月 28 日
五个新区域支持 Amazon SNS 短信	Amazon SNS 增加了对以下区域的短信支持：亚太地区（海得拉巴）、亚太地区（墨尔本）、中东（阿联酋）、欧洲（苏黎世）和欧洲（西班牙）。	2024 年 2 月 8 日
Firebase Cloud Messaging（FCM）HTTP v1 支持	Amazon SNS 支持 FCM v1 凭证。	2024 年 1 月 18 日
Amazon SNS 在亚太地区（雅加达）支持 SMS	Amazon SNS 在亚太地区（雅加达）支持 SMS 消息收发	2023 年 12 月 14 日
Amazon CloudFormation 支持为 Amazon SNS 主题`DeliveryStatusLogging` 进行配置	Amazon CloudFormation 支持在创建或更新 Amazon SNS 主题`DeliveryStatusLogging` 时进行配置。	2023 年 12 月 7 日
增加了新的消息筛选运算符	现在，在筛选 Amazon SNS 消息时，您可以使用后缀匹配、等于-忽略大小写和 OR 运算符。	2023 年 11 月 16 日
增加了对消息归档与重播功能的支持	主题所有者可以将主题的消息归档最多 365 天。主题订阅用户可以将归档的消息重播回订阅的端点，以恢复由于下游应用程序出现故障而受影响的消息，或者复制现有应用程序的状态。	2023 年 10 月 26 日

增加了对标准队列订阅 FIFO 主题的支持	您可以将 Amazon SQS FIFO 队列或标准队列订阅到 Amazon SNS FIFO 主题。只有 Amazon SQS FIFO 队列才能保证消息按顺序接收，并且没有重复消息。	2023 年 9 月 14 日
增加了对以色列（特拉维夫）的短信支持	以色列（特拉维夫）区域现在支持 Amazon SNS 短信。	2023 年 8 月 28 日
支持添加到 FIFO 主题的 X-Ray 活动跟踪	以前仅支持亚马逊 SNS 标准主题， Amazon X-Ray 现在可以跟踪和分析用户通过您的 FIFO 主题传输到您的亚马逊数据 Firehose、Amazon Amazon Lambda SQS 和 HTTP/S 终端节点订阅的请求。	2023 年 5 月 31 日
增强的 Content-Type 标头支持	您可以在请求策略中设置 Content-Type 标头，以指定通知的媒体类型。	2023 年 3 月 23 日
添加了主动跟踪支持	Amazon X-Ray 跟踪和分析用户通过您的亚马逊 SNS 标准主题传输到您的亚马逊 Data Firehose、Amazon S Amazon Lambda QS 和 HTTP/S 终端节点订阅的请求。	2023 年 2 月 8 日
新加坡发件人 ID 注册	新增了 IDs 在新加坡注册发件人的说明。	2023 年 1 月 10 日
基于有效负载的消息筛选	您可以利用基于有效负载的筛选来根据消息有效负载筛选消息，避免处理不需要的数据而产生相关成本。	2022 年 11 月 22 日

SHA256 为 Amazon SNS 消息签名添加了哈希算法	使用 Amazon SNS 消息签名时增加了对 SHA256 哈希算法的支持。	2022 年 9 月 15 日
增加了更多支持 SMS 消息发送的区域	Amazon SNS 在以下地区支持短信：非洲（开普敦）、亚太地区（大阪）、欧洲（米兰）和 Amazon GovCloud （美国东部）。	2022 年 9 月 9 日
增加了消息数据保护支持	消息数据保护使用数据保护策略来审计和屏蔽在应用程序或 Amazon 服务之间移动的敏感信息，从而保护发布到您的 Amazon SNS 主题的数据。	2022 年 9 月 8 日
免费电话号码的新注册流程	添加了对使用免费电话号码 (TFN) 向美国接收人发送 Amazon SNS 消息的支持。	2022 年 8 月 1 日
支持基于属性的访问控制 (ABAC)	添加了对 API 操作的基于属性的访问控制 (ABAC) 的支持，包括 `Publish` 和 `PublishBatch` 。ABAC 是一种授权策略，它根据标签定义访问权限，这些标签可以附加到 IAM 资源（例如 IAM 用户和角色）和 Amazon 资源（如 Amazon SNS 主题），以简化权限管理。	2022 年 1 月 10 日
支持推送通知的基于 Apple 令牌的身份验证	您可以通过提供将您识别为应用程序开发人员的信息，授权 Amazon SNS 将推送通知发送到您的 iOS 或 macOS 应用程序。	2021 年 10 月 28 日

SMS 消息的新发件人放置在 SMS 沙盒中	SMS 沙盒用于防止出现欺诈和滥用以及保护您作为发件人的声誉。当您的 Amazon 账户处于短信沙箱中时，您只能向经过验证的目标电话号码发送 SMS 消息。	2021 年 6 月 1 日
SMS 消息的新发件人放置在 SMS 沙盒中	SMS 沙盒用于防止出现欺诈和滥用以及保护您作为发件人的声誉。当您的 Amazon 账户处于短信沙箱中时，您只能向经过验证的目标电话号码发送 SMS 消息。	2021 年 6 月 1 日
向位于印度的收件人发送 SMS 消息的新属性	要想位于印度的收件人发送 SMS 消息，现在需要两项新的属性 Entity ID（实体 ID）和 Template ID（模板 ID）。	2021 年 4 月 22 日
消息筛选运算符的更新	新的运算符 `cidr` 可用于匹配消息源 IP 地址和子网。您现在还可以检查是否存在属性键，并使用前缀与 `anything-but` 运算符进行属性字符串匹配。	2021 年 4 月 7 日
终止对美国目的地的 P2P 长代码的支持	自 2021 年 6 月 1 日起，美国电信提供商不再支持使用 person-to-person (P2P) 长码与美国目的地通信 application-to-person (A2P)。相反，您可以使用短代码、免费电话号码或名为 10DLC 的新型源号码。	2021 年 2 月 16 日
支持 1 分钟 Amazon 指标 CloudWatch	Amazon SNS 的 1 分钟 CloudWatch 指标现已在所有地区推出。 Amazon	2021 年 1 月 28 日

支持 Amazon Data Firehose 端点	您可以将 Firehose 传输流订阅到 SNS 主题。这允许您向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、Amazon Redshift 表、 OpenSearch 亚马逊服务OpenSearch（服务）等。	2021 年 1 月 12 日
源号码可用	您可以在发送短信 (SMS) 时使用源号码。	2020 年 10 月 23 日
支持 Amazon SNS FIFO 主题	要集成几乎实时地需要数据一致性的分布式应用程序，您可以将 Amazon SNS 先进先出 (FIFO) 主题与 Amazon SQS FIFO 队列结合使用。	2020 年 10 月 22 日
适用于 Java 的 Amazon SNS 扩展客户端库可用	您可以使用此库发布大型 Amazon SNS 消息。	2020 年 8 月 25 日
SSE 在中国区域可用	Amazon SNS 的服务器端加密 (SSE) 在中国区域可用。	2020 年 1 月 20 日
Support 支持 DLQs 使用捕获无法投递的邮件	要捕获无法送达的消息，您可以将 Amazon SQS 死信队列与 Amazon SNS 订阅结合使用。	2019 年 11 月 14 日
Support 支持指定自定义 APNs 标题值	您可以指定自定义标 APNs 题值。	2019 年 10 月 18 日
Support 支持 `apns-push-type` " 标题字段 APNs	您可以将标`apns-push-type`题字段用于通过发送的移动通知 APNs。	2019 年 9 月 10 日
Support 支持使用主题疑难解答 Amazon X-Ray	您可以使用 X-Ray 对通过 SNS 主题传递的消息进行问题排查。	2019 年 7 月 24 日

支持使用“exists”运算符进行属性键匹配	要检查传入消息中是否存在其键与筛选策略中列出的键匹配的属性，您可以使用 exists 运算符。	2019 年 7 月 5 日
支持多个数值的 anything-but 匹配	除了多个字符串之外，Amazon SNS 还允许多个数值的 anything-but 匹配。	2019 年 7 月 5 日
Amazon SNS 发行说明可用作 RSS 源	在此页面上的标题（文档历史记录）后，请选择 RSS。	2019 年 6 月 22 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。