aws

Guia do Desenvolvedor

Amazon Simple Notification Service

Amazon Simple Notification Service: Guia do Desenvolvedor

Table of Contents

O que é o Amazon SNS?

O Amazon Simple Notification Service (Amazon SNS) é um serviço totalmente gerenciado que fornece entrega de mensagens de editores (produtores) para assinantes (consumidores). Os publicadores se comunicam de maneira assíncrona com os assinantes produzindo e enviando mensagens para um tópico, que é um canal de comunicação e um ponto de acesso lógico.

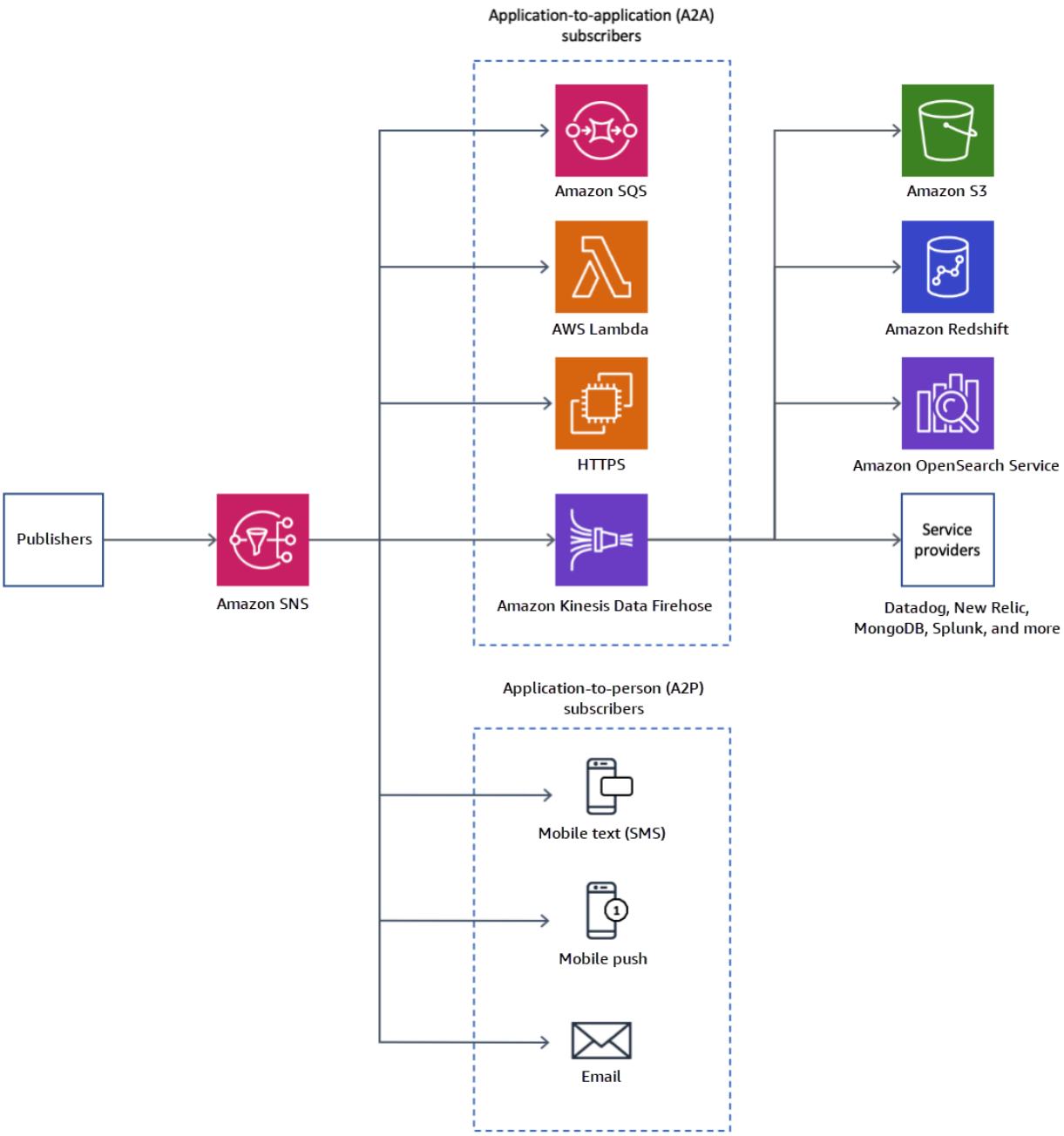
Como funciona

No SNS, os editores enviam mensagens para um tópico, que funciona como um canal de comunicação. O tópico atua como um ponto de acesso lógico, garantindo que as mensagens sejam entregues a vários assinantes em diferentes plataformas.

Os assinantes de um tópico do SNS podem receber mensagens por meio de diferentes endpoints, dependendo do caso de uso, como:

- Amazon SQS
- Lambda
- Pontos de extremidade HTTP (S)
- E-mail
- Notificações por push para dispositivos móveis
- Mensagens de texto móveis (SMS)
- Amazon Data Firehose
- Provedores de serviços (por exemplo, Datadog, MongoDB, Splunk)

O SNS suporta mensagens Application-to-Application (A2A) e Application-to-Person (A2P), oferecendo flexibilidade para enviar mensagens entre diferentes aplicativos ou diretamente para telefones celulares, endereços de e-mail e muito mais.



Acessando o Amazon SNS

Você pode acessar e gerenciar o Amazon SNS por meio do console ou AWS CLI AWS SDKs, dependendo do seu método preferido de interação. O console oferece uma interface gráfica para

tarefas básicas, enquanto o AWS CLI e SDKs fornece recursos avançados de configuração e automação para casos de uso mais complexos.

- OConsole do Amazon SNSfornece uma interface de usuário conveniente para criar tópicos e assinaturas, enviar e receber mensagens e monitorar eventos e logs.
- O AWS Command Line Interface (AWS CLI) fornece acesso direto à API do Amazon SNS para casos de uso avançados de configuração e automação. Para obter mais informações, consulte Usar o Amazon SNS com o AWS CLI.
- AWS fornece SDKs em vários idiomas. Para obter mais informações, consulte Kits SDKs de ferramentas.

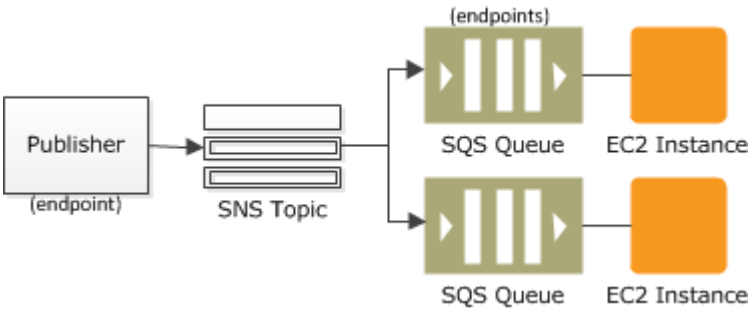
Cenários comuns do Amazon SNS

Use esses cenários comuns do Amazon SNS para implementar arquiteturas escaláveis e orientadas por eventos e garantir a comunicação confiável e em tempo real entre aplicações e usuários.

Integração de aplicações

O cenário de Fanout é quando uma mensagem publicada em um tópico do SNS é replicada e enviada para vários endpoints, como fluxos de entrega do Firehose, filas do Amazon SQS, endpoints HTTP (S) e funções do Lambda. Isso permite o processamento paralelo assíncrono.

Por exemplo, você pode desenvolver uma aplicação que publica uma mensagem em um tópico do SNS sempre que um pedido de um produto é feito. Assim, as filas do SQS inscritas no tópico do SNS receberão notificações idênticas para o novo pedido. Uma instância de servidor Amazon Elastic Compute Cloud (Amazon EC2) conectada a uma das filas do SQS pode lidar com o processamento ou o atendimento do pedido. E você pode anexar outra instância EC2 do servidor Amazon a um data warehouse para análise de todos os pedidos recebidos.



Também é possível usar “fanout” para replicar os dados enviados ao seu ambiente e produção com seu ambiente de teste. Expandindo o exemplo anterior, você pode inscrever mais uma fila do SQS

para o mesmo tópico do SNS para novos pedidos. Ao anexar essa nova fila do SQS ao seu ambiente de teste, você pode continuar a melhorar e testar seu aplicativo usando os dados recebidos do seu ambiente de produção.

> **Important**
>
> Certifique-se de considerar a privacidade e a segurança dos dados antes de enviar quaisquer dados de produção para o seu ambiente de teste.

Para obter mais informações, consulte os seguintes recursos:

- Fanout de fluxos de entrega do Firehose
- Fanout notificações do Amazon SNS para funções do Lambda para processamento automatizado
- Fanout de notificações do Amazon SNS para filas do Amazon SQS para processamento assíncrono
- Fanout de notificações do Amazon SNS para endpoints HTTPS
- Computação orientada a eventos com o Amazon SNS AWS e serviços de computação, armazenamento, banco de dados e rede

Alertas do

Alertas de aplicações e do sistema são notificações que são acionadas por limites predefinidos. O Amazon SNS pode enviar essas notificações para usuários especificados via SMS e e-mail. Por exemplo, você pode receber uma notificação imediata quando ocorrer um evento, como uma alteração específica no seu grupo do Amazon EC2 Auto Scaling, um novo arquivo carregado em um bucket do Amazon S3 ou um limite métrico violado na Amazon. CloudWatch Para obter mais informações, consulte Configuração de notificações do Amazon SNS no Guia CloudWatch do usuário da Amazon.

Notificações ao usuário

O Amazon SNS pode enviar mensagens de e-mail de push e mensagens de texto (mensagens SMS) para indivíduos ou grupos. Por exemplo, você pode enviar confirmações de pedidos de comércio eletrônico como notificações do usuário. Para obter mais informações sobre o uso do Amazon SNS para enviar mensagens SMS, consulte Mensagens de texto em dispositivos móveis com o Amazon SNS.

Notificações por push para dispositivos móveis

Notificações por push para dispositivos móveis permitem que você envie mensagens diretamente para aplicativos móveis. Por exemplo, você pode usar o Amazon SNS para enviar notificações de atualização para um aplicativo. A mensagem de notificação pode incluir um link para fazer download e instalar a atualização. Para obter mais informações sobre como usar o Amazon SNS para enviar mensagens de notificação por push, consulteEnviar notificações por push para dispositivos móveis com o Amazon SNS.

Preços do Amazon SNS

O Amazon SNS não tem custos iniciais. Você paga com base no número de mensagens publicadas, no número de notificações que você entrega e em quaisquer chamadas adicionais à API para gerenciar tópicos e assinaturas. Os preços de entrega variam de acordo com o tipo de endpoint. Você pode começar gratuitamente com o nível gratuito do Amazon SNS. Para obter informações, consulte Definição global de preço de SMS.

Recursos e funcionalidades do Amazon SNS

O Amazon SNS oferece um conjunto abrangente de recursos projetados para aprimorar as mensagens entre aplicações e usuários. Esses recursos permitem comunicação perfeita, entrega segura de mensagens e gerenciamento robusto de mensagens, garantindo alta disponibilidade, durabilidade e flexibilidade para uma ampla variedade de casos de uso de mensagens.

Application-to-application mensagens

Um pplication-to-application sistema de mensagens oferece suporte a assinantes, como fluxos de entrega do Amazon Data Firehose, funções Lambda, filas do Amazon SQS, endpoints e pipelines do Event Fork. HTTP/S AWS Isso permite a entrega eficiente de mensagens em arquiteturas orientadas por eventos.

Application-to-person notifications

pplication-to-personAs notificações fornecem notificações de usuários aos assinantes, como aplicativos móveis, números de telefone celular e endereços de e-mail.

Tópicos padrão e FIFO

Os tópicos do FIFO garantem a ordenação, o agrupamento e a desduplicação rigorosos de mensagens, permitindo que o FIFO e as filas padrão se inscrevam para processamento de

mensagens. Os tópicos padrão são usados quando a ordenação de mensagens e a possível duplicação não são essenciais, oferecendo suporte a todos os protocolos de entrega para casos de uso mais amplos.

Durabilidade das mensagens

O Amazon SNS usa várias estratégias que funcionam juntas para fornecer durabilidade das mensagens:

- As mensagens publicadas são armazenadas em vários servidores e datacenters geograficamente separados.
- Se um endpoint inscrito não estiver disponível, o Amazon SNS executará umPolítica de novas tentativas de entrega.
- Para preservar todas as mensagens que não são entregues antes do término da política de repetição de entrega, você pode criar uma fila de mensagens não entregues.

Arquivamento, reprodução e análise de mensagens

É possível arquivar mensagens com o Amazon SNS de várias formas, inclusive inscrever fluxos de entrega do Firehose em tópicos do SNS, o que permite enviar notificações a outros endpoints de análise, como buckets do Amazon Simple Storage Service (Amazon S3), tabelas do Amazon Redshift e muito mais. Além disso, os tópicos FIFO do Amazon SNS comportam arquivamento e reprodução de mensagens como um arquivamento de mensagens local, sem código, que permite aos proprietários de tópicos armazenar (ou arquivar) mensagens no respectivo tópico. Os assinantes de tópicos podem então recuperar (ou reproduzir) as mensagens arquivadas de volta em um endpoint inscrito. Para saber mais, consulte Arquivamento e reprodução de mensagens do Amazon SNS de tópicos FIFO.

Atributos de mensagens

Os Atributos de mensagem do Amazon SNS permitem fornecer quaisquer metadados arbitrários sobre a mensagem.

Filtragem de mensagens

Por padrão, cada assinante recebe todas as mensagens publicadas no tópico. Para receber um subconjunto de mensagens, um assinante deve atribuir uma política de filtro à assinatura do tópico. Um assinante também pode definir o escopo da política de filtro para habilitar a filtragem baseada em carga útil ou baseada em atributo. O valor padrão para o escopo da política de filtro é `MessageAttributes`. Quando os atributos de mensagem de entrada correspondem aos atributos de política de filtro, a mensagem é entregue ao endpoint inscrito. Caso contrário,

a mensagem será filtrada. Quando o escopo da política de filtro é `MessageBody`, os atributos da política de filtro são comparados com a carga útil. Para obter mais informações, consulte Filtragem de mensagens.

Segurança de mensagens

A criptografia do lado do servidor protege o conteúdo das mensagens armazenadas nos tópicos do Amazon SNS, usando chaves de criptografia fornecidas pelo. AWS KMS Para obter mais informações, consulte the section called “Segurança dos dados com a criptografia do lado do servidor” Você também pode estabelecer uma conexão privada entre o Amazon SNS e sua nuvem privada virtual (VPC). Para obter mais informações, consulte. the section called “Proteger o tráfego com endpoints da VPC”

AWS serviços comumente usados com o Amazon SNS

Integre o Amazon SNS a vários Nuvem AWS serviços para impulsionar o tratamento de mensagens, melhorar o controle de acesso, permitir o processamento orientado por eventos e automatizar recursos. Essa integração otimiza o desempenho, fortalece a segurança e simplifica as operações.

Amazon CloudWatch

CloudWatch A Amazon fornece monitoramento e observabilidade para o Amazon SNS, ajudando você a rastrear a entrega de mensagens, detectar anomalias e solucionar problemas. Com o CloudWatch, você pode:

- Monitore as métricas do Amazon SNS, como o número de mensagens publicadas, entregues ou falhadas em todos os tópicos e assinaturas.
- Configure CloudWatch alarmes para acionar ações automatizadas quando as métricas do Amazon SNS excederem limites predefinidos, como altas falhas de entrega ou limitação.
- Use CloudWatch Logs para capturar o status de entrega do Amazon SNS para mensagens enviadas para endpoints HTTP/S, Lambda e Amazon SQS para depuração e auditoria.

Para obter mais informações, consulte Monitorando tópicos do Amazon SNS usando CloudWatch.

Amazon SQS

O Amazon SQS é um serviço de enfileiramento de mensagens totalmente gerenciado que permite a comunicação segura, durável e escalável entre componentes de software distribuídos. Ele ajuda a desacoplar a arquitetura do aplicativo armazenando mensagens em buffer,

garantindo uma entrega confiável e evitando falhas no sistema devido à perda de mensagens. O Amazon SQS se integra ao Amazon SNS das seguintes formas:

- Filas de mensagens mortas — O Amazon SNS pode rotear mensagens não entregues para uma fila de mensagens mortas do Amazon SQS para solução de problemas e reprocessamento.
- Assinaturas de tópicos — Você pode inscrever uma fila do Amazon SQS em um tópico do Amazon SNS, permitindo que o Amazon SNS distribua mensagens para vários consumidores usando o Amazon SQS.
- Suporte para filas FIFO — As filas FIFO do Amazon SQS podem ser inscritas nos tópicos FIFO do Amazon SNS, garantindo uma ordenação rigorosa de mensagens e um processamento exato. As filas padrão do Amazon SQS também podem se inscrever nos tópicos do Amazon SNS, mas não garantem a entrega ou a desduplicação de mensagens solicitadas.

AWS CloudFormation

AWS CloudFormation automatiza o provisionamento e o gerenciamento de AWS recursos, incluindo tópicos e assinaturas do Amazon SNS, usando infraestrutura como código (IaC). Com AWS CloudFormation, você pode:

- Defina tópicos, assinaturas e permissões do Amazon SNS em um modelo reutilizável e controlado por versão.
- Garanta a implantação consistente dos recursos do Amazon SNS em várias Contas da AWS regiões.
- Atualize ou modifique as configurações do Amazon SNS usando conjuntos de alterações sem intervenção manual.

Para obter mais informações, consulte o Guia do usuário do AWS CloudFormation.

AWS CloudTrail

CloudTrail fornece visibilidade da atividade da API para o Amazon SNS, ajudando você a monitorar e auditar o acesso aos tópicos, assinaturas e mensagens do Amazon SNS. Com CloudTrail, você pode:

- Acompanhe as chamadas de API feitas para o Amazon SNS, incluindo quem acessou ou modificou tópicos, assinaturas e permissões.
- Detecte atividades não autorizadas ou inesperadas analisando registros para fins de segurança e conformidade.
- Integre-se com AWS Security Hub a Amazon CloudWatch ou crie alertas com base em ações incomuns do Amazon SNS.

Para obter mais informações, consulte o Registrando chamadas AWS de API do SNS usando AWS CloudTrail.

AWS Lambda

AWS Lambda é um serviço de computação sem servidor que executa automaticamente seu código em resposta a eventos, eliminando a necessidade de provisionar ou gerenciar servidores. Ele permite que você crie aplicativos orientados por eventos que escalam automaticamente e são executados em um ambiente computacional altamente disponível.

O Amazon SNS se integra ao Lambda, permitindo que você inscreva uma função do Lambda em um tópico do Amazon SNS. Quando um tópico do Amazon SNS recebe uma mensagem, ele pode acionar a função Lambda, permitindo o processamento em tempo real, a automação e a execução da lógica do aplicativo. Essa integração é comumente usada para:

- Processamento orientado por eventos — Acione automaticamente funções em resposta às mensagens do Amazon SNS.
- Transformação de dados — Modifique ou filtre mensagens do Amazon SNS antes de encaminhá-las para outros serviços.
- Fluxos de trabalho automatizados — Processe notificações para alertas de aplicativos, monitoramento do sistema ou orquestração de eventos.

AWS Identity and Access Management (IAM)

O IAM fornece controle de acesso seguro para AWS recursos, permitindo que você gerencie quem pode acessar seus tópicos do Amazon SNS, quais ações eles podem realizar e sob quais condições. Com o IAM, você pode:

- Autentique usuários e serviços antes que eles possam interagir com os tópicos do Amazon SNS.
- Defina permissões refinadas para especificar quais tópicos ou funções do Amazon SNS os usuários ou funções podem publicar, assinar ou gerenciar.
- Use políticas baseadas em identidade para aplicar as melhores práticas de segurança, como restringir o acesso a endereços IP ou condições específicas Contas da AWS.

Para obter mais informações, consulte Usar políticas baseadas em identidade com o Amazon SNS.

AWS Key Management Service (AWS KMS)

AWS KMS aprimora a segurança do Amazon SNS ao habilitar a criptografia do lado do servidor (SSE) para confidencialidade das mensagens. Com AWS KMS, você pode:

- Criptografe mensagens do Amazon SNS em repouso AWS usando chaves de criptografia gerenciadas ou gerenciadas pelo cliente (). CMKs
- Controle o acesso aos tópicos do Amazon SNS definindo políticas chave refinadas que restringem quem pode publicar ou assinar.
- Garanta a conformidade com os requisitos regulatórios e de segurança auditando o uso da chave por meio AWS CloudTrail de.

Para obter mais informações, consulte Gerenciar chaves e custos de criptografia do Amazon SNS.

AWS X-Ray

O X-Ray fornece rastreamento para o Amazon SNS, ajudando você a analisar e depurar o fluxo de mensagens por meio de sua arquitetura orientada por eventos. Com o X-Ray, você pode:

- Rastreie a entrega de mensagens do Amazon SNS em vários Serviços da AWS endpoints, como Lambda, Amazon SQS e HTTP/S.
- Identifique gargalos de latência visualizando quanto tempo as mensagens levam para serem publicadas, entregues e processadas.
- Detecte erros e novas tentativas nos fluxos de mensagens do Amazon SNS para solucionar problemas de entrega com falha ou tempos de processamento lentos.

Para obter mais informações, consulte Rastreamento ativo no Amazon SNS.

Usando o Amazon SNS com um SDK AWS

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK para Go	AWS SDK para Go exemplos de código
AWS SDK para Java	AWS SDK para Java exemplos de código

Documentação do SDK	Exemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK para .NET	AWS SDK para .NET exemplos de código
AWS SDK para PHP	AWS SDK para PHP exemplos de código
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell exemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
SDK da AWS para SAP ABAP	SDK da AWS para SAP ABAP exemplos de código
AWS SDK for Swift	AWS SDK for Swift exemplos de código

Para obter exemplos específicos do Amazon SNS, consulteExemplos de código para o Amazon SNS usando AWS SDKs.

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

Criar um tópico do Amazon SNS e publique mensagens

Este tópico fornece as etapas fundamentais para gerenciar os recursos do Amazon SNS, focando especificamente em tópicos, assinaturas e publicação de mensagens. Primeiro, você configurará as permissões de acesso necessárias para o Amazon SNS, garantindo que você tenha as permissões corretas para criar e gerenciar recursos do Amazon SNS. Em seguida, você criará um novo tópico do Amazon SNS, que serve como o hub central para gerenciar e entregar mensagens aos assinantes. Depois de criar o tópico, você continuará criando uma assinatura para esse tópico, permitindo que endpoints específicos recebam as mensagens publicadas nele.

Quando o tópico e a assinatura estiverem prontos, você publicará uma mensagem sobre o tópico, observando como o Amazon SNS entrega a mensagem de forma eficiente a todos os endpoints inscritos. Por fim, você aprenderá a excluir a assinatura e o tópico, completando o ciclo de vida dos recursos do Amazon SNS que você gerenciou. Essa abordagem fornece uma compreensão clara das operações fundamentais no Amazon SNS, equipando você com as habilidades práticas necessárias para gerenciar fluxos de trabalho de mensagens usando o console do Amazon SNS.

Configurar o acesso para o Amazon SNS

Antes de usar o Amazon SNS pela primeira vez, conclua as etapas abaixo.

Crie um Conta da AWS e um usuário do IAM

Para acessar qualquer AWS serviço, você deve primeiro criar um Conta da AWS. Você pode usar o seu Conta da AWS para visualizar seus relatórios de atividade e uso e para gerenciar a autenticação e o acesso.

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

Quando você se inscreve em um Conta da AWS, um usuário Conta da AWS root é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu usuário Conta da AWS raiz AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu usuário Conta da AWS root

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Fazer login como usuário-raiz no Guia do usuário do Início de Sessão da AWS .
2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

   Para obter instruções, consulte Habilitar o AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Criar um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .
2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Adicionar grupos no Guia do usuário do AWS IAM Identity Center .

Próximas etapas

Agora que você está preparado para trabalhar com o Amazon SNS, comece com:

1. Criar um tópico do Amazon SNS
2. Criação de uma assinatura em um tópico do Amazon SNS
3. Publicar uma mensagem do Amazon SNS
4. Excluir uma assinatura e um tópico do Amazon SNS

Criar um tópico do Amazon SNS

Um tópico do Amazon SNS é um ponto de acesso lógico que atua como um canal de comunicação. Um tópico permite agrupar vários endpoints (como Amazon SQS AWS Lambda, HTTP/S ou um endereço de e-mail).

Para transmitir as mensagens de um sistema produtor de mensagem (por exemplo, um site de comércio eletrônico) que trabalha com vários outros serviços que exigem suas mensagens (por exemplo, sistemas de cumprimento e checkout), crie um tópico para o sistema produtor.

A tarefa inicial e mais comum do Amazon SNS é a criação de um tópico. Esta página mostra como você pode usar o AWS Management Console AWS SDK para Java, o e o AWS SDK para .NET para criar um tópico.

Durante a criação, você escolhe um tipo de tópico (padrão ou FIFO) e nomeia o tópico. Depois de criado um tópico, você não pode alterar o tipo ou o nome do tópico. Todas as outras opções de configuração são opcionais durante a criação do tópico e você pode editá-las posteriormente.

> **Important**
>
> Não inclua informações de identificação pessoal (PII) nem outras informações confidenciais ou sigilosas em nomes de tópicos. Os nomes dos tópicos podem ser acessados por outros Amazon Web Services, incluindo CloudWatch Logs. Os nomes de tópicos não devem ser usados para dados privados ou sigilosos.

Para criar um tópico usando o AWS Management Console

A criação de um tópico no Amazon SNS estabelece a base para a distribuição de mensagens, permitindo que você publique mensagens que podem ser distribuídas para vários assinantes. Essa etapa é essencial para definir o tipo, as configurações de criptografia e as políticas de acesso do tópico, garantindo que o tópico atenda aos requisitos operacionais, de segurança e de conformidade da organização.

1. Faça login no console do Amazon SNS.
2. Execute um destes procedimentos:
   - Se nenhum tópico já tiver sido criado Conta da AWS antes, leia a descrição do Amazon SNS na página inicial.
   - Se os tópicos já tiverem sido criados abaixo Conta da AWS do seu, no painel de navegação, escolha Tópicos.
3. Na página Tópicos, escolha Criar tópico.

4. Na página Criar tópico, na seção Detalhes, faça o seguinte:

   a. Em Tipo, escolha um tipo de tópico (Standard ou FIFO).

   b. Insira um Nome para o tópico. Para um tópico FIFO, adicione .fifo ao final do nome.

   c. (Opcional) Insira um Nome de exibição para o tópico.

   > **Important**
   >
   > Ao assinar um endpoint de e-mail, a contagem combinada de caracteres para o nome de exibição do tópico do Amazon SNS e o endereço de e-mail do remetente (por exemplo, no-reply@sns.amazonaws.com) não deve exceder 320 caracteres UTF-8. Você pode usar uma ferramenta de codificação de terceiros para verificar o tamanho do endereço de remetente antes de configurar um nome de exibição para seu tópico do Amazon SNS.

   d. (Opcional) Para um tópico FIFO, você pode escolher Desduplicação de mensagens baseada em conteúdo para habilitar a desduplicação de mensagens padrão. Para obter mais informações, consulte Desduplicação de mensagens do Amazon SNS para tópicos FIFO.

5. (Opcional) Expanda a seção Criptografia e faça o seguinte. Para obter mais informações, consulte Segurança dos dados do Amazon SNS com a criptografia do lado do servidor.

   a. Selecione Habilitar criptografia.

   b. Especifique a AWS KMS chave. Para obter mais informações, consulte Principais termos.

   Para cada tipo de KMS, são exibidos Description (Descrição), Account (Conta) e KMS ARN (ARN do KMS).

   > **Important**
   >
   > Se você não for o proprietário do KMS ou se fizer login com uma conta que não tenha as permissões `kms:ListAliases` e `kms:DescribeKey`, não será possível visualizar as informações sobre o KMS no console do Amazon SNS.
   > Peça ao proprietário do KMS para conceder essas permissões a você. Para obter mais informações, consulte Permissões da API do KMS: referência de ações e recursos do AWS KMS no Guia do desenvolvedor do AWS Key Management Service .

- O KMS AWS gerenciado para Amazon SNS (padrão alias/aws/sns) é selecionado por padrão.

> **Note**
>
> Lembre-se do seguinte:
>
> - A primeira vez que você usa o AWS Management Console para especificar o KMS AWS gerenciado para o Amazon SNS para um tópico AWS KMS , cria AWS o KMS gerenciado para o Amazon SNS.
> - Como alternativa, na primeira vez que você usa a `Publish` ação em um tópico com o SSE ativado, AWS KMS cria o KMS AWS gerenciado para o Amazon SNS.

- Para usar um KMS personalizado da sua AWS conta, escolha o campo Chave KMS e, em seguida, escolha o KMS personalizado na lista.

> **Note**
>
> Para obter instruções sobre como criar chaves personalizadas KMSs, consulte Criação de chaves no Guia do AWS Key Management Service desenvolvedor

- Para usar um ARN KMS personalizado da AWS sua conta ou de AWS outra conta, insira-o no campo Chave do KMS.

6. (Opcional) Por padrão, somente o proprietário do tópico pode publicar ou assinar o tópico. Para configurar permissões de acesso adicionais, expanda a seção Política de acesso. Para obter mais informações, consulte Gerenciamento de identidade e acesso no Amazon SNS e Casos de exemplo para controle de acesso do Amazon SNS.

> **Note**
>
> Quando você cria um tópico usando o console, a política padrão usa a chave de condição `aws:SourceOwner`. Essa chave é semelhante a `aws:SourceAccount`.

7. (Opcional) Para configurar como o Amazon SNS repete tentativas de entrega de mensagem com falha, expanda a seção Delivery retry policy (HTTP/S) (Política de repetição de entrega

(HTTP/S)). Para obter mais informações, consulte Novas tentativas de entrega de mensagens do Amazon SNS.

8. (Opcional) Para configurar como o Amazon SNS registra a entrega de mensagens CloudWatch, expanda a seção Registro de status de entrega. Para obter mais informações, consulte Status de entrega de mensagens do Amazon SNS.
9. (Opcional) Para adicionar tags de metadados ao tópico, expanda a seção Tags, insira uma Chave e um Valor (opcional) e escolha Adicionar tag. Para obter mais informações, consulte Marcação de tópicos do Amazon SNS.
10. Escolha Criar tópico.

    O tópico é criado e a ***MyTopic***página é exibida.

    O nome do tópico, o ARN, o nome de exibição (opcional) e o ID da AWS conta do proprietário do tópico são exibidos na seção Detalhes.
11. Copie o ARN do tópico para a área de transferência, por exemplo:

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

Para criar um tópico usando um AWS SDK

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

Os exemplos de código a seguir mostram como usar o `CreateTopic`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico com um nome específico.

```
    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;
        }
    }
```

Crie um tópico com um nome e atributos específicos de FIFO e desduplicação.

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

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um tópico do SNS

O exemplo `create-topic` a seguir cria um tópico do SNS chamado `my-topic`.

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

Saída:

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

Para obter mais informações, consulte Usando a interface de linha de AWS comando com o Amazon SQS e o Amazon SNS no Guia do usuário AWS da interface de linha de comando.

- Para obter detalhes da API, consulte CreateTopicna Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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 "";
    }
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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()
    }
```

```
}
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


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

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

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a CreateTopicReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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

```
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

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

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
```

```
}
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   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.
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
        }
```

- Para obter detalhes da API, consulte CreateTopica referência da API AWS SDK for Swift.

Criação de uma assinatura em um tópico do Amazon SNS

Para receber mensagens publicadas em um tópico, você precisa inscrever um endpoint nesse tópico. Depois de inscrito, o endpoint começa a receber todas as mensagens publicadas no tópico associado.

> **Note**
>
> Endpoints HTTP (S), endereços de e-mail e outros AWS recursos Contas da AWS exigem a confirmação da assinatura antes de poderem receber mensagens.

Como inscrever um endpoint em um tópico do Amazon SNS

Inscrever um endpoint em um tópico do Amazon SNS permite a entrega de mensagens para o endpoint especificado, garantindo que os sistemas ou usuários certos recebam notificações quando uma mensagem é publicada no tópico. Essa etapa é essencial para vincular o tópico aos consumidores, sejam eles aplicativos, destinatários de e-mail ou outros serviços, permitindo uma comunicação perfeita entre sistemas.

1. Faça login no console do Amazon SNS.
2. No painel de navegação à esquerda, selecione Assinaturas.
3. Na página Assinaturas, escolha Criar assinatura.

4. Na página Criar assinatura, na seção Detalhes, faça o seguinte:

   a. Em Topic ARN (ARN do tópico), escolha o nome do recurso da Amazon (ARN) de um tópico. Esse valor é o AWS ARN que foi gerado quando você criou o tópico do Amazon SNS, por exemplo. `arn:aws:sns:us-east-2:123456789012:your_topic`

   b. Em Protocolo, escolha um tipo de endpoint. Os tipos de endpoint disponíveis são:

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

      > **Note**
      >
      > Para se inscrever em um tópico FIFO do SNS, escolha esta opção.

      - AWS Lambda
      - Endpoint da aplicação da plataforma
      - SMS

   c. Em Endpoint, insira o valor do endpoint, como um endereço de e-mail ou o ARN de uma fila do Amazon SQS.

   d. Somente endpoints do Firehose: em ARN da função de assinatura, especifique o ARN do perfil do IAM que você criou para gravar em fluxos de entrega do Firehose. Para obter mais informações, consulte Pré-requisitos para assinatura de fluxos de entrega do Firehose para tópicos do Amazon SNS.

   e. (Opcional) Para endpoints HTTP/S, do Firehose ou do Amazon SQS, você também pode habilitar a entrega de mensagens brutas. Para obter mais informações, consulte Entrega de mensagens brutas do Amazon SNS.

   f. (Opcional) Para configurar uma política de filtros, expanda a seção Subscription filter policy (Política de filtro de assinatura). Para obter mais informações, consulte Políticas de filtro de assinatura do Amazon SNS.

   g. (Opcional) Para habilitar a filtragem baseada em carga útil, configure `Filter Policy Scope` como `MessageBody`. Para obter mais informações, consulte Escopo de política de filtro de assinaturas do Amazon SNS.

h. (Opcional) Para configurar uma fila de mensagens não entregues para a assinatura, expanda a seção Redrive policy (dead-letter queue) (Política de redirecionamento (fila de mensagens não entregues)). Para obter mais informações, consulte Filas de mensagens não entregues do Amazon SNS.

i. Selecione Create subscription.

   O console cria a assinatura e abre a página Details (Detalhes) da assinatura.

Publicar uma mensagem do Amazon SNS

Depois de criar um tópico do Amazon SNS e inscrever um endpoint nele, é possível publicar mensagens no tópico. Quando uma mensagem é publicada, o Amazon SNS tenta entregar a mensagem aos endpoints inscritos.

Para publicar mensagens nos tópicos do Amazon SNS usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação à esquerda, selecione Tópicos.
3. Na página Topics (Tópicos), selecione um tópico e escolha Publish message (Publicar em um tópico).

   O console abre a página Publish message to topic (Publicar mensagem no tópico).
4. Na seção Basic details (Detalhes básicos) faça o seguinte:

   a. (Opcional) Insira o Subject (Assunto) de uma mensagem.

   b. Para um tópico FIFO, insira um Message group ID (ID do grupo de mensagens). As mensagens no mesmo grupo de mensagens são entregues na ordem em que são publicadas.

   c. Para um tópico FIFO, insira um Message deduplication ID (ID de desduplicação de mensagem). Este ID é opcional se você habilitou a configuração de Desduplicação de mensagens baseada em conteúdo para o tópico.

   d. (Opcional) Para notificações por push para dispositivos móveis, insira um valor para Time to Live (TTL) (Vida útil (TTL)) em segundos. Esse é o tempo que um serviço de notificação push, como o Apple Push Notification Service (APNs) ou o Firebase Cloud Messaging (FCM), tem para entregar a mensagem ao endpoint.

5. Na seção Message body (Corpo da mensagem), siga um destes procedimentos:

   a. Escolha Identical payload for all delivery protocols (Carga útil idêntica para todos os protocolos de entrega) e, em seguida, insira uma mensagem.

   b. Escolha Custom payload for each delivery protocol (Carga útil personalizada para cada protocolo de entrega) e, depois, insira um objeto JSON para definir a mensagem a ser enviada a cada protocolo de entrega.

   Para obter mais informações, consulte Publicar notificações do Amazon SNS com cargas úteis específicas da plataforma.

6. Na seção Message attributes (Atributos da mensagem), adicione todos os atributos que deseja que o Amazon SNS corresponda ao atributo de assinatura `FilterPolicy` para decidir se o endpoint inscrito está interessado na mensagem publicada.

   a. Em Type Tipo, escolha um tipo de atributo, como String.Array.

   > **Note**
   >
   > Para o tipo de atributo String.Array, coloque a matriz entre colchetes (`[]`). Dentro da matriz, coloque valores de strings entre aspas duplas. As aspas não são necessárias para números nem para as palavras-chave `true`, `false` e `null`.

   b. Insira um atributo Name (Nome), por exemplo, `customer_interests`.

   c. Insira um atributo Value (Valor), por exemplo, `["soccer", "rugby", "hockey"]`.

   Se o tipo do atributo for String, String.Array ou Number, o Amazon SNS avaliará o atributo de mensagem em relação à política de filtro de uma assinatura (se existir) antes de enviar a mensagem à assinatura, considerando que o escopo da política de filtro não está explicitamente definido como `MessageBody`.

   Para obter mais informações, consulte Atributos de mensagem do Amazon SNS.

7. Selecione Publish message (Publicar mensagem).

   A mensagem é publicada no tópico e o console abre a página Details (Detalhes).

Para publicar uma mensagem em um tópico usando um AWS SDK

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

Os exemplos de código a seguir mostram como usar o `Publish`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem em um tópico.

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

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

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

Aplique as seleções do usuário à ação de publicação.

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

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

Publicar uma mensagem com um atributo.

```
        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;
        }
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Exemplo 1: Para publicar uma mensagem em um tópico:

O exemplo `publish` a seguir publica a mensagem específica no tópico do SNS especificado. A mensagem é proveniente de um arquivo de texto, o que permite incluir quebras de linha.

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

Conteúdo de `message.txt`:

```
Hello World
Second Line
```

Saída:

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

Exemplo 2: Para publicar uma mensagem SMS em um número de telefone

O exemplo `publish` a seguir publica a mensagem `Hello world!` no número de telefone `+1-555-555-0100`.

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

Saída:

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

- Para obter detalhes da API, consulte Publish na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

```
  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();
    }
  }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


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

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

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

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para PHP .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

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

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$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
```

- Para obter detalhes da API, consulte Publicar no Ferramentas da AWS para PowerShell Cmdlet Reference (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

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

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$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
```

- Para obter detalhes da API, consulte Publicar no Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem com atributos para que uma assinatura possa filtrar com base em atributos.

```
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
```

Publique uma mensagem que assume diferentes formas com base no protocolo do assinante.

```
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
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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

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

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

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Ruby .

Rust

SDK for Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter os detalhes da API, consulte Publicar na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)")
```

- Para obter detalhes da API, consulte Publicar no AWS SDK para referência da API Swift.

Publicar mensagens grandes com o Amazon SNS e o Amazon S3

Para publicar mensagens grandes do Amazon SNS, você pode usar a Biblioteca do cliente em versão ampliada para Java do Amazon SNS ou a Biblioteca do cliente em versão ampliada para Python do Amazon SNS. Essas bibliotecas são úteis para mensagens maiores do que o máximo atual de 256 KB, até o máximo de 2 GB. As bibliotecas salvam a carga útil real em um bucket do Amazon S3 e publicam a referência do objeto armazenado do Amazon S3 no tópico do Amazon SNS. As filas do Amazon SQS inscritas podem usar a biblioteca do cliente em versão ampliada do

Amazon SQS para Java para cancelar a referência e recuperar cargas úteis do Amazon S3. Outros endpoints, como o Lambda, podem usar a Payload Offloading Java Common Library for AWS para cancelar a referência e recuperar a carga útil.

> **Note**
>
> As bibliotecas do cliente em versão ampliada do Amazon SNS são compatíveis com tópicos comuns e FIFO.

Biblioteca do cliente em versão ampliada para Java do Amazon SNS

Pré-requisitos

Veja a seguir os pré-requisitos para usar a Biblioteca do cliente em versão ampliada para Java do Amazon SNS:

- Um AWS SDK. O exemplo nesta página usa o AWS Java SDK. Para instalar e configurar o SDK, consulte Configurar o AWS SDK para Java no Guia AWS SDK para Java do desenvolvedor.
- E Conta da AWS com as credenciais adequadas. Para criar uma Conta da AWS, navegue até a página AWS inicial e escolha Criar uma AWS conta. Siga as instruções.

  Para obter informações sobre credenciais, consulte Configurar AWS credenciais e região para desenvolvimento no Guia do AWS SDK para Java desenvolvedor.
- Java 8 ou posterior.
- A Biblioteca do cliente em versão ampliada para Java (também disponível no Maven).

Configurar o armazenamento de mensagens

A biblioteca Amazon SNS Extended Client usa a biblioteca comum Java Payload Offloading AWS para armazenamento e recuperação de mensagens. Você pode configurar as seguintes opções de armazenamento de mensagens do Amazon S3:

- Limite de tamanhos de mensagem personalizados — Mensagens com cargas e atributos que excedem esse tamanho são armazenadas automaticamente no Amazon S3.
- **`alwaysThroughS3`**flag — Defina esse valor para `true` forçar todas as cargas de mensagens a serem armazenadas no Amazon S3. Por exemplo:

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

- Chave KMS personalizada — A chave a ser usada para criptografia do lado do servidor em seu bucket Amazon S3.
- Nome do bucket — O nome do bucket do Amazon S3 para armazenar cargas de mensagens.

Exemplo: publicação de mensagens no Amazon SNS com carga útil armazenada no Amazon S3

O código de exemplo a seguir mostra como:

- Criar um tópico de exemplo e uma fila.
- Inscrever a fila para receber mensagens do tópico.
- Publicar uma mensagem de teste.

A carga útil da mensagem é armazenada no Amazon S3 e a referência a ela é publicada. O Amazon SQS Extended Client é usado para receber a mensagem.

SDK para Java 1.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para publicar uma mensagem grande, use a Amazon SNS Extended Client Library for Java. A mensagem que você envia faz referência a um objeto do Amazon S3 que contém o conteúdo real da mensagem.

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

Outros protocolos de endpoint

Tanto as bibliotecas do Amazon SNS quanto do Amazon SQS usam a Payload Offloading Java Common Library for AWS para armazenar e recuperar cargas úteis de mensagens com o Amazon S3. Qualquer endpoint habilitado para Java (por exemplo, um endpoint HTTPS implementado em Java) pode usar a mesma biblioteca para cancelar a referência ao conteúdo da mensagem.

Os endpoints que não podem usar a biblioteca comum Java para descarregamento de carga ainda AWS podem publicar mensagens com cargas armazenadas no Amazon S3. Veja a seguir um exemplo de uma referência do Amazon S3 que é publicada pelo exemplo de código acima:

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

Biblioteca do cliente em versão ampliada para Python

Pré-requisitos

Veja a seguir os pré-requisitos para usar a Biblioteca do cliente em versão ampliada para Python do Amazon SNS:

- Um AWS SDK. O exemplo nesta página usa AWS Python SDK Boto3. Para instalar e configurar o SDK, consulte a documentação do AWS SDK para Python.
- E Conta da AWS com as credenciais adequadas. Para criar uma Conta da AWS, navegue até a página AWS inicial e escolha Criar uma AWS conta. Siga as instruções.

Para obter informações sobre credenciais, consulte Credenciais no Guia do desenvolvedor do AWS SDK para Python.

- Python 3.x (ou posterior) e pip.
- A Biblioteca do cliente em versão ampliada para Python (também disponível no PyPI).

Configurar o armazenamento de mensagens

Os atributos abaixo estão disponíveis no Boto3 Amazon SNS Client, Topic PlatformEndpointe objetos para configurar as opções de armazenamento de mensagens do Amazon S3.

- **`large_payload_support`**— O nome do bucket do Amazon S3 que armazenará mensagens grandes.
- **`use_legacy_attribute`**— Se sim`True`, todas as mensagens publicadas usarão o atributo de mensagem reservada antiga (`SQSLargePayloadSize`) em vez do atributo de mensagem reservada atual (`ExtendedPayloadSize`).
- **`message_size_threshold`**: o limite para armazenar a mensagem no bucket de mensagens grandes. Não pode ser menor `0` ou maior que`262144`. O padrão é `262144`.
- **`always_through_s3`**: se `True`, então todas as mensagens serão armazenadas no Amazon S3. O padrão é `False`.
- **`s3_client`**— O objeto Boto3 Amazon `client` S3 a ser usado para armazenar objetos no Amazon S3. Use isso se quiser controlar o cliente Amazon S3 (por exemplo, configurações ou credenciais personalizadas do Amazon S3). O padrão é `boto3.client("s3")` no primeiro uso, se não tiver sido definido anteriormente.

Exemplo: publicação de mensagens no Amazon SNS com a carga útil armazenada no Amazon S3

O código de exemplo a seguir mostra como:

- Crie um exemplo de tópico do Amazon SNS e uma fila do Amazon SQS.
- Anexe a política à fila do Amazon SQS para receber a mensagem do tópico do Amazon SNS.
- Inscrever a fila para receber mensagens do tópico.
- Publique uma mensagem de teste usando o cliente estendido, o recurso Topic e PlatformEndpoint o recurso do Amazon SNS.
- A carga útil da mensagem é armazenada no Amazon S3 e a referência a ela é publicada.

- Imprima a mensagem publicada da fila junto com a mensagem original recuperada do Amazon S3.

Para publicar uma mensagem grande, use a Biblioteca do cliente em versão ampliada para Python. A mensagem que você envia faz referência a um objeto do Amazon S3 que contém o conteúdo real da mensagem.

```
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)
```

Saída

```
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
```

Atributos de mensagem do Amazon SNS

O Amazon SNS oferece suporte à entrega de atributos de mensagem, o que permite que você forneça itens de metadados estruturados (como time stamps, dados geoespaciais, assinaturas e identificadores) sobre a mensagem. Para assinaturas SQS, no máximo dez atributos de mensagem podem ser enviados quando Raw Message Delivery (Entrega de mensagens brutas) estiver habilitado. Para enviar mais de dez atributos de mensagem, é necessário desabilitar Entrega de mensagens brutas. As mensagens com mais de 10 atributos de mensagem direcionadas para

assinaturas do Amazon SQS habilitadas para entrega bruta de mensagens serão descartadas como erros do lado do cliente.

Os atributos de mensagem são opcionais e separados do corpo da mensagem, mas são enviados junto com ele. O destinatário pode usar essas informações para decidir como lidar com ela sem ter de primeiro processar o corpo da mensagem.

Para obter informações sobre o envio de mensagens com atributos usando o AWS Management Console ou o AWS SDK para Java, consulte o Para publicar mensagens nos tópicos do Amazon SNS usando o AWS Management Console tutorial.

> **Note**
>
> Os atributos de mensagens são enviados somente quando a estrutura da mensagem é String, e não JSON.

Também é possível usar atributos de mensagem para ajudar a estruturar a mensagem de notificação por push para endpoints móveis. Nesse cenário, os atributos de mensagens são usados somente para ajudar a estruturar a mensagem de notificação por push. Os atributos não são entregues ao endpoint do mesmo modo que são ao enviar mensagens com atributos de mensagem para endpoints do Amazon SQS.

Você também pode usar atributos de mensagens para que suas mensagens possam ser filtradas usando políticas de filtro de assinatura. Você pode aplicar as políticas de filtro a inscrições em tópicos. Quando uma política de filtro é aplicada com o escopo de políticas de filtro definido como `MessageAttributes` (padrão), uma assinatura recebe somente aquelas mensagens que têm atributos aceitos pela política. Para obter mais informações, consulte Filtragem de mensagens do Amazon SNS.

> **Note**
>
> Quando atributos de mensagem são usados para filtragem, o valor deve ser uma string JSON válida. Isso garante que a mensagem seja entregue a uma assinatura com a filtragem de atributos de mensagem ativada.

Validação e itens de atributos de mensagem

Cada atributo de mensagem consiste nos seguintes itens:

- Nome: o nome do atributo de mensagem pode conter os seguintes caracteres: A-Z, a-z, 0-9, sublinhado (_), hífen (-) e ponto (.). O nome não deve iniciar ou terminar com um ponto, e nem ter pontos sucessivos. O nome diferencia maiúsculas e minúsculas e deve ser exclusivo entre todos os nomes de atributos para a mensagem. O nome pode ter até 256 caracteres. O nome não pode começar com `AWS.` ou `Amazon.` (ou quaisquer variações em maiúsculas e minúsculas) porque esses prefixos são reservados para uso da Amazon Web Services.
- Tipo: os tipos de dados de atributo de mensagem compatíveis são `String`, `String.Array`, `Number` e `Binary`. O tipo de dados tem as mesmas restrições no conteúdo que o corpo da mensagem. Para obter mais informações, consulte a seção Tipos de dados de atributo de mensagem e validação.
- Valor: o valor do atributo de mensagem especificado pelo usuário. Para tipos de dados de string, o atributo value deve seguir as mesmas restrições de conteúdo do corpo da mensagem. No entanto, se o atributo da mensagem for usado para filtragem, o valor deverá ser uma string JSON válida para garantir a compatibilidade com as políticas de filtro de assinatura do Amazon SNS. Para obter mais informações, consulte a ação Publish na Referência da API do Amazon Simple Notification Service.

Nome, tipo e valor não podem estar vazios ou nulos. Além disso, o corpo da mensagem não pode estar vazio ou nulo. Todas as partes do atributo da mensagem, incluindo o nome, o tipo e o valor, são incluídas na restrição de tamanho da mensagem, que é de 256 KB.

Tipos de dados de atributo de mensagem e validação

Os tipos de dados de atributo de mensagem identificam como os valores de atributo de mensagem são processados pelo Amazon SNS. Por exemplo, se o tipo for um número, o Amazon SNS validará que é um número.

O Amazon SNS é compatível com os seguintes tipos de dados lógicos em todos os endpoints, com exceção daqueles com observação contrária:

- String: as strings são Unicode com codificação binária UTF-8. Para obter uma lista de valores de código, consulte http://en.wikipedia. org/wiki/ASCII#ASCII_printable_characters.

> **Note**
>
> Valores substitutos não são aceitos nos atributos da mensagem. Por exemplo, usar um valor substituto para representar um emoji apresentará o seguinte erro: `Invalid attribute value was passed in for message attribute`.

- String.Array: uma matriz, formatada como uma string, que pode conter vários valores. Os valores podem ser strings, números ou as palavras-chave `true`, `false` e `null`. Um String.Array de número ou tipo booliano não precisa de aspas. Vários valores String.Array são separados por vírgulas.

  Esse tipo de dados não é compatível com AWS Lambda assinaturas. Se você especificar esse tipo de dados para endpoints do Lambda, ele será passado como o tipo de dados `String` na carga útil JSON que o Amazon SNS entrega ao Lambda.
- Number: números são inteiros positivos ou negativos ou com ponto flutuante. Números têm alcance e precisão suficientes para abranger a maioria dos possíveis valores que números inteiros, flutuantes e duplicados normalmente comportam. Um número pode ter um valor de -10^9 a 10^9, com 5 dígitos de precisão após o ponto decimal. Zeros iniciais e finais são cortados.

  Esse tipo de dados não é compatível com AWS Lambda assinaturas. Se você especificar esse tipo de dados para endpoints do Lambda, ele será passado como o tipo de dados `String` na carga útil JSON que o Amazon SNS entrega ao Lambda.
- Binary: os atributos de tipo binário podem armazenar qualquer dado binário, por exemplo, dados compactados, dados criptografados ou imagens.

Atributos de mensagens reservados para notificações móveis por push

A tabela a seguir lista os atributos de mensagens reservadas para serviços de notificação móvel por push que você pode usar para estruturar a mensagem de notificação por push:

Serviço de notificação por push	Atributo de mensagem reservada
ADM	`AWS.SNS.MOBILE.ADM.TTL`
APNs[1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`

Serviço de notificação por push	Atributo de mensagem reservada
	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
	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

Serviço de notificação por push	Atributo de mensagem reservada
	`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] A Apple rejeitará as notificações do Amazon SNS se os atributos das mensagens não atenderem aos requisitos. Para obter detalhes adicionais, consulte Envio de solicitações de notificação para o APNs site de desenvolvedores da Apple.

Agrupamento de mensagens do Amazon SNS em lotes

O que é o agrupamento de mensagens em lotes?

Uma alternativa à publicação de mensagens em tópicos padrão ou FIFO em solicitações individuais da API `Publish` é usar a API `PublishBatch` do Amazon SNS para publicar até 10 mensagens em uma única solicitação de API. O envio de mensagens em lotes pode ajudar a reduzir os custos associados à conexão de aplicações distribuídas (sistema de mensagens A2A) ou ao envio de notificações a pessoas (sistema de mensagens A2P) com o Amazon SNS em um fator de até 10. O Amazon SNS tem cotas que definem quantas mensagens é possível publicar em um tópico por segundo com base na região de operação. Consulte a página Endpoints e cotas do Amazon SNS no guia Referência geral da AWS para obter mais informações sobre cotas de API.

> **Note**
>
> O tamanho total agregado de todas as mensagens enviadas em uma única solicitação de `PublishBatch` API não pode exceder 262.144 bytes (256 KiB).
> A API `PublishBatch` isa a mesma ação da API `Publish` para políticas do IAM.

Como funciona o agrupamento de mensagens em lotes?

A publicação de mensagens com a API `PublishBatch` é semelhante à publicação de mensagens com a API `Publish`. A principal diferença é que cada mensagem em uma solicitação da API `PublishBatch` precisa ser atribuída a um ID de lote exclusivo com até 80 caracteres. Dessa maneira, podem ser retornadas respostas individuais da API pelo Amazon SNS para cada mensagem em um lote para confirmar se cada mensagem foi publicada ou se ocorreu uma falha. Para mensagens que estão sendo publicadas em tópicos FIFO, além de incluir a atribuição de um ID de lote exclusivo, ainda é necessário incluir um `MessageDeduplicationID` e `MessageGroupId` para cada mensagem individual.

Exemplos

Publicação de um lote de 10 mensagens em um tópico padrão

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

Publicação de um lote de 10 mensagens em um tópico FIFO

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

Excluir uma assinatura e um tópico do Amazon SNS

Quando um tópico é excluído, suas assinaturas associadas são excluídas de forma assíncrona. Embora os clientes ainda possam acessar essas assinaturas, elas não estão mais associadas ao tópico, mesmo que você recrie o tópico usando o mesmo nome. Se um publicador tentar publicar uma mensagem no tópico excluído, o publicador receberá uma mensagem de erro indicando que o tópico não existe. Da mesma forma, qualquer tentativa de se inscrever no tópico excluído também resultará em uma mensagem de erro. Não é possível excluir uma assinatura que esteja pendente de confirmação. Após 48 horas, o Amazon SNS exclui automaticamente a assinatura não confirmada.

Para excluir um tópico ou assinatura do Amazon SNS usando o AWS Management Console

A exclusão de um tópico ou assinatura do Amazon SNS garante o gerenciamento eficiente dos recursos, evitando o uso desnecessário e mantendo o console do Amazon SNS organizado. Essa etapa ajuda a evitar possíveis custos de recursos ociosos e simplifica a administração ao remover tópicos ou assinaturas que não são mais necessários.

Para excluir um tópico usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação à esquerda, selecione Tópicos.
3. Na página Topics (Tópicos), selecione um tópico e, em seguida, escolha Delete (Excluir).
4. Na caixa de diálogo Delete topic (Excluir tópico), insira `delete me` e escolha Delete (Excluir).

   O console exclui o tópico.

Para excluir uma assinatura usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação à esquerda, selecione Assinaturas.
3. Na página Assinaturas, escolha uma assinatura com o status de Confirmada e escolha Excluir.
4. Na caixa de diálogo Delete subscription (Excluir assinatura), escolha Delete (Excluir).

   O console exclui a assinatura.

Para excluir uma assinatura e um tópico usando o AWS SDK

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

Os exemplos de código a seguir mostram como usar o `DeleteTopic`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um tópico por meio do respectivo ARN.

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

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um tópico do SNS

O exemplo `delete-topic` a seguir exclui o tópico do SNS especificado.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteTopicem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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
  //   }
  // }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DeleteTopicReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

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

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

- Para obter detalhes da API, consulte DeleteTopica referência da API AWS SDK for Swift.

Próximas etapas

Agora que você criou um tópico com uma assinatura e enviou mensagens para o tópico, talvez queira experimentar o seguinte:

- Explorar o Centro do desenvolvedor da AWS.
- Saiba como proteger seus dados na seção Segurança.
- Habilite a criptografia do lado do servidor para um tópico.
- Habilitar a criptografia no lado do servidor para um tópico com uma fila criptografada do Amazon Simple Queue Service (Amazon SQS inscrita.

- Assine AWS Event Fork Pipelines em um tópico.

Ordenação de mensagens e estratégias de desduplicação usando tópicos FIFO do Amazon SNS

Este tópico fornece informações sobre os recursos e funcionalidades dos tópicos FIFO (First-In-First-Out) do Amazon SNS e como eles se integram com filas FIFO do Amazon SQS. Você aprenderá a usar esses serviços juntos para garantir a ordenação e a desduplicação estritas de mensagens, essenciais para aplicativos que exigem consistência de dados. Este conteúdo aborda os casos de uso específicos em que os tópicos FIFO do Amazon SNS são benéficos, fornecendo informações sobre cenários em que a ordem e a exclusividade das mensagens são essenciais.

Você também aprenderá sobre os detalhes técnicos da ordem e do agrupamento de mensagens e como eles afetam a entrega das mensagens. O tópico de desduplicação de mensagens explica os mecanismos que evitam mensagens duplicadas, garantindo que cada mensagem seja processada somente uma vez. Além disso, você aprenderá sobre filtragem, segurança e durabilidade de mensagens, que são importantes para manter a integridade e a confiabilidade do seu sistema de mensagens. Esse conteúdo também inclui informações sobre arquivamento e reprodução de mensagens, oferecendo estratégias para gerenciar históricos de mensagens. Exemplos práticos de códigos também são fornecidos para ajudá-lo a implementar esses recursos em seus próprios aplicativos, oferecendo experiência prática com os tópicos FIFO do Amazon SNS e sua integração com as filas FIFO do Amazon SQS.

Tópicos de FIFO de alto rendimento no Amazon SNS

Tópicos de FIFO de alta taxa de transferência no Amazon SNS gerenciam com eficiência a alta taxa de transferência de mensagens, mantendo uma ordem estrita de mensagens, garantindo confiabilidade e escalabilidade para aplicativos que processam várias mensagens. Essa solução é ideal para cenários que exigem throughput alto e entrega ordenada de mensagens. Para melhorar a taxa de transferência de mensagens usando tópicos FIFO de alta taxa de transferência, é recomendável aumentar o número de grupos de mensagens. Para obter mais informações sobre cotas de mensagens de alta taxa de transferência, consulte as cotas de serviço do Amazon SNS no. Referência geral da Amazon Web Services

Casos de uso de alta taxa de transferência para tópicos FIFO do Amazon SNS

Os casos de uso a seguir destacam as diversas aplicações de tópicos de FIFO de alto rendimento, mostrando sua eficácia em todos os setores e cenários:

- Processamento de dados em tempo real: aplicativos que lidam com fluxos de dados em tempo real, como processamento de eventos ou ingestão de dados de telemetria, podem se beneficiar de tópicos de FIFO de alto rendimento para lidar com o fluxo contínuo de mensagens e, ao mesmo tempo, preservar sua ordem para uma análise precisa.
- Processamento de pedidos de comércio eletrônico: em plataformas de comércio eletrônico em que manter a ordem das transações do cliente é fundamental, os tópicos de FIFO de alto rendimento garantem que os pedidos sejam entregues sequencialmente e sem atrasos, mesmo durante os períodos de pico de compras.
- Serviços financeiros: instituições financeiras que lidam com dados comerciais ou transacionais de alta frequência dependem de tópicos de FIFO de alto rendimento para processar dados e transações de mercado com latência mínima, ao mesmo tempo em que cumprem os rígidos requisitos regulatórios para pedidos de mensagens.
- Streaming de mídia: plataformas de streaming e serviços de distribuição de mídia utilizam tópicos FIFO de alto rendimento para gerenciar a entrega de arquivos de mídia e conteúdo de streaming, garantindo experiências de reprodução suaves para os usuários e mantendo a ordem correta de entrega do conteúdo

Partições e distribuição de dados para alta taxa de transferência para tópicos FIFO do Amazon SNS

Com tópicos de alta taxa de transferência, o Amazon SNS distribui dados de tópicos FIFO entre partições. Uma partição é uma alocação de capacidade para um tópico que é replicada automaticamente em várias zonas de disponibilidade em um. Região da AWS Você não gerencia partições. Em vez disso, o Amazon SNS gerencia automaticamente as partições em seu nome, com base na taxa de entrada.

Para tópicos de FIFO, o Amazon SNS modifica o número de partições em um tópico nas seguintes situações:

- Se a taxa de publicação atual se aproximar ou exceder o que as partições existentes podem suportar, partições adicionais serão alocadas até que o tópico atinja a cota regional. Para obter informações sobre cotas, consulte as cotas de serviço do Amazon SNS no. Referência geral da Amazon Web Services
- Se as partições atuais tiverem baixa utilização, o número de partições poderá ser reduzido.

O gerenciamento de partições ocorre automaticamente em segundo plano e é transparente para as aplicações. Seu tópico e suas mensagens estão sempre disponíveis.

> **Note**
>
> A limitação temporária da API de publicação pode ocorrer se você aumentar repentinamente e significativamente o tráfego para seu tópico enquanto envia várias vezes o volume normal. Essa limitação pode durar até a duração da janela de desduplicação, enquanto o tópico se expande para acomodar o aumento do tráfego.

Distribuindo dados por grupo de mensagens IDs

Ao publicar uma mensagem em um tópico FIFO, o Amazon SNS usa o valor do ID do grupo de mensagens de cada mensagem como entrada para uma função hash interna. O valor de saída da função hash determina qual partição processa a mensagem; um ou mais grupos de mensagens IDs podem ser manipulados por uma determinada partição.

> **Note**
>
> O Amazon SNS é otimizado para distribuição uniforme de itens nas partições de um tópico FIFO, independentemente do número de partições. AWS recomenda que você use um grupo de mensagens IDs que pode ter um grande número de valores distintos.

Habilite uma alta taxa de transferência em seu tópico FIFO do Amazon SNS

Por padrão, os tópicos FIFO do Amazon SNS são configurados para desduplicação em nível de tópico. Isso é controlado pelo atributo de tópico `FifoThroughputScope`definido como `Topic` e

tem cotas de taxa de transferência mais restritas. Consulte as cotas de serviço do Amazon SNS no. Referência geral da Amazon Web Services

Para habilitar uma alta taxa de transferência para seu tópico FIFO do Amazon SNS, `FifoThroughputScope` atualize o atributo para. `MessageGroup` Essa alteração pode ser feita por meio do console ou usando o SDK AWS CLI e também pode ser definida durante a criação do tópico, o que o Amazon SNS recomenda para a melhor experiência do cliente e para reduzir as chances de seu tópico ser limitado.

> **Important**
>
> Depois de habilitar um tópico `FifoThroughputScope` para`MessageGroup`, ele não pode ser revertido para a `Topic` taxa de transferência.

Ative o modo de alta taxa de transferência para qualquer fila FIFO do Amazon SQS inscrita

Ao publicar em seu tópico FIFO do Amazon SNS com alta taxa de transferência habilitada e uma ou mais filas FIFO do Amazon SQS estarem inscritas, é recomendável que você habilite a alta taxa de transferência em suas filas FIFO do Amazon SQS para permitir que seu tópico FIFO de alto rendimento do Amazon SNS seja entregue sem problemas. Para obter mais informações, consulte Alta taxa de transferência para filas FIFO no Guia do desenvolvedor do Amazon Simple Queue Service.

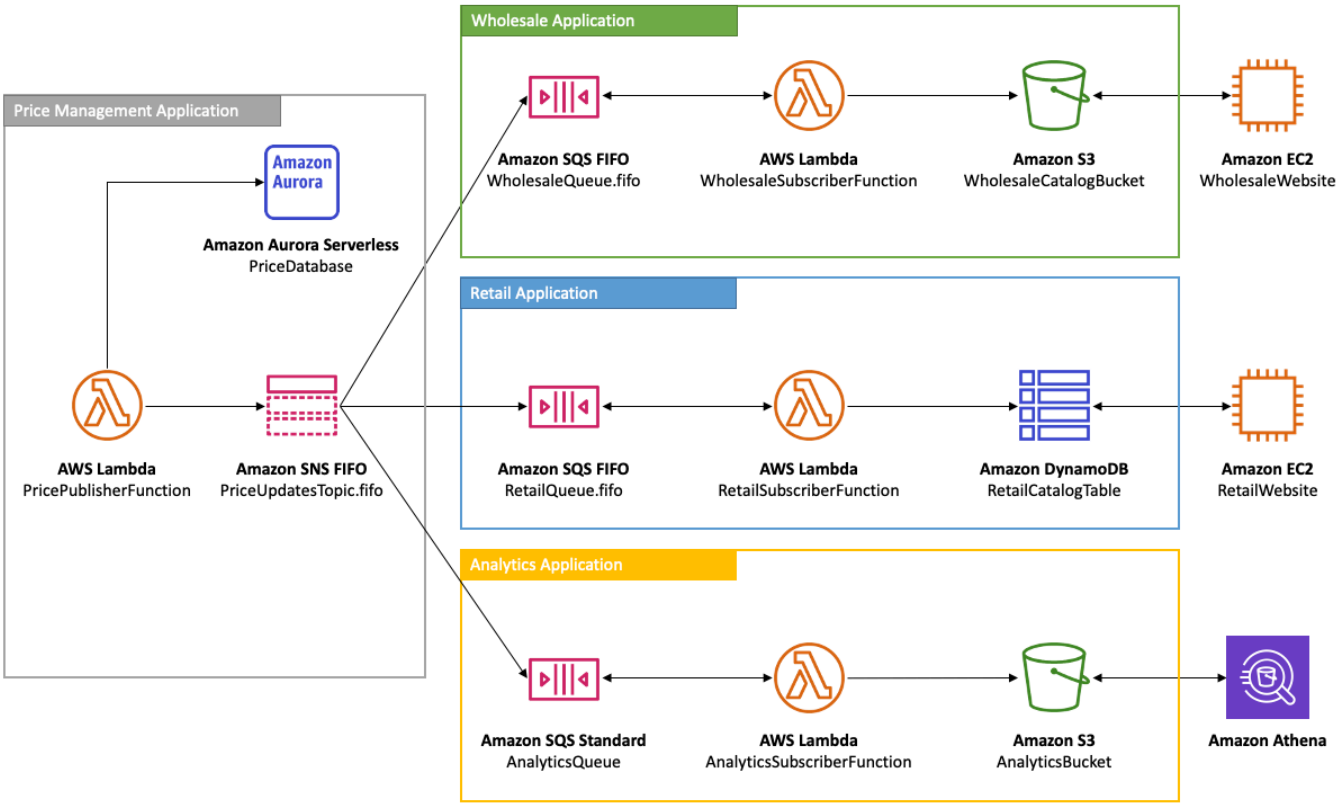
Exemplo de caso de uso do tópico FIFO do Amazon SNS

O exemplo a seguir descreve uma plataforma de comércio eletrônico criada por um fabricante de autopeças usando tópicos FIFO do Amazon SNS e filas do Amazon SQS. A plataforma é composta de quatro aplicações sem servidor:

- Os gerentes de inventário usam uma aplicação de gerenciamento de preços para definir o preço de cada item em estoque. Nessa empresa, os preços dos produtos podem mudar com base na flutuação cambial, na demanda do mercado e em mudanças na estratégia de vendas. A aplicação de gerenciamento de preços usa uma função do AWS Lambda que publica atualizações de preços em um tópico FIFO do Amazon SNS sempre que os preços mudam.
- Uma aplicação de atacado fornece o backend para um site em que oficinas mecânicas e fabricantes de automóveis podem comprar peças automotivas da empresa a granel. Para obter

notificações sobre alteração de preço, a aplicação de atacado inscreve sua fila FIFO do Amazon SQS no tópico FIFO do Amazon SNS da aplicação de gerenciamento de preços.

- Uma aplicação de varejo fornece o backend para outro site em que os proprietários de carros e entusiastas de carros tunados podem comprar peças automotivas individuais para seus veículos. Para obter notificações sobre alteração de preço, a aplicação de varejo também inscreve sua fila FIFO do Amazon SQS no tópico FIFO do Amazon SNS da aplicação de gerenciamento de preços.
- Uma aplicação de análise que agrega atualizações de preços e as armazena em um bucket do Amazon S3, permitindo que o Amazon Athena consulte o bucket para fins de business intelligence (BI). Para obter notificações sobre alteração de preço, a aplicação de análise inscreve sua fila padrão do Amazon SQS no tópico FIFO do Amazon SNS da aplicação de gerenciamento de preços. Ao contrário de outras aplicações, a de análise não exige que as atualizações de preços sejam ordenadas estritamente.

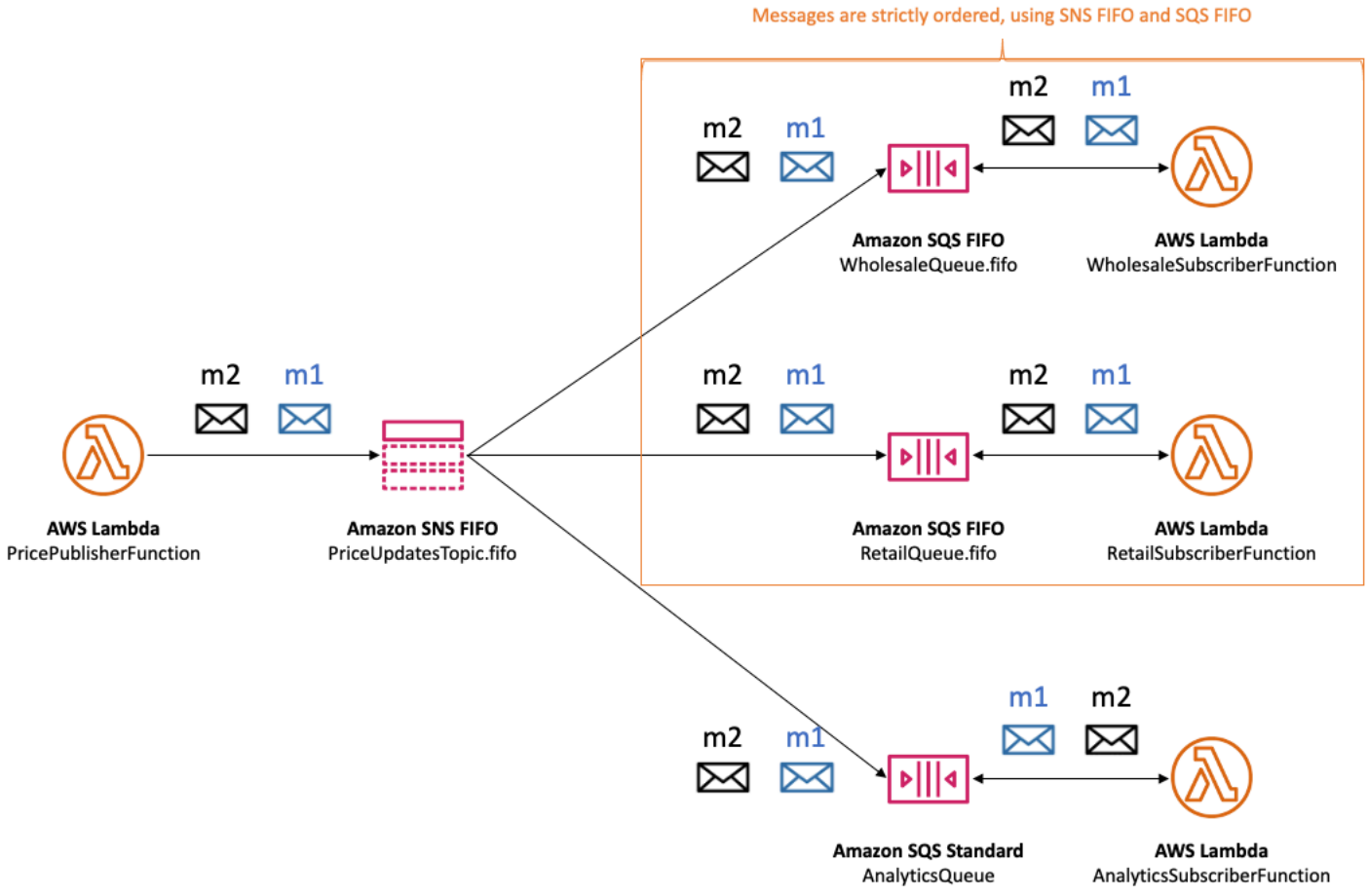


Para que as aplicações de atacado e varejo recebam atualizações de preços na ordem correta, a aplicação de gerenciamento de preços deve usar um sistema de distribuição de mensagens estritamente ordenado. O uso de tópicos FIFO do Amazon SNS e filas FIFO do Amazon SQS permite o processamento de mensagens em ordem e sem duplicação. Para obter mais informações, consulte Detalhes de ordenação de mensagens do Amazon SNS para tópicos FIFO. Para obter

trechos de código que implementem esse caso de uso, consulte Exemplos de código do Amazon SNS para tópicos FIFO.

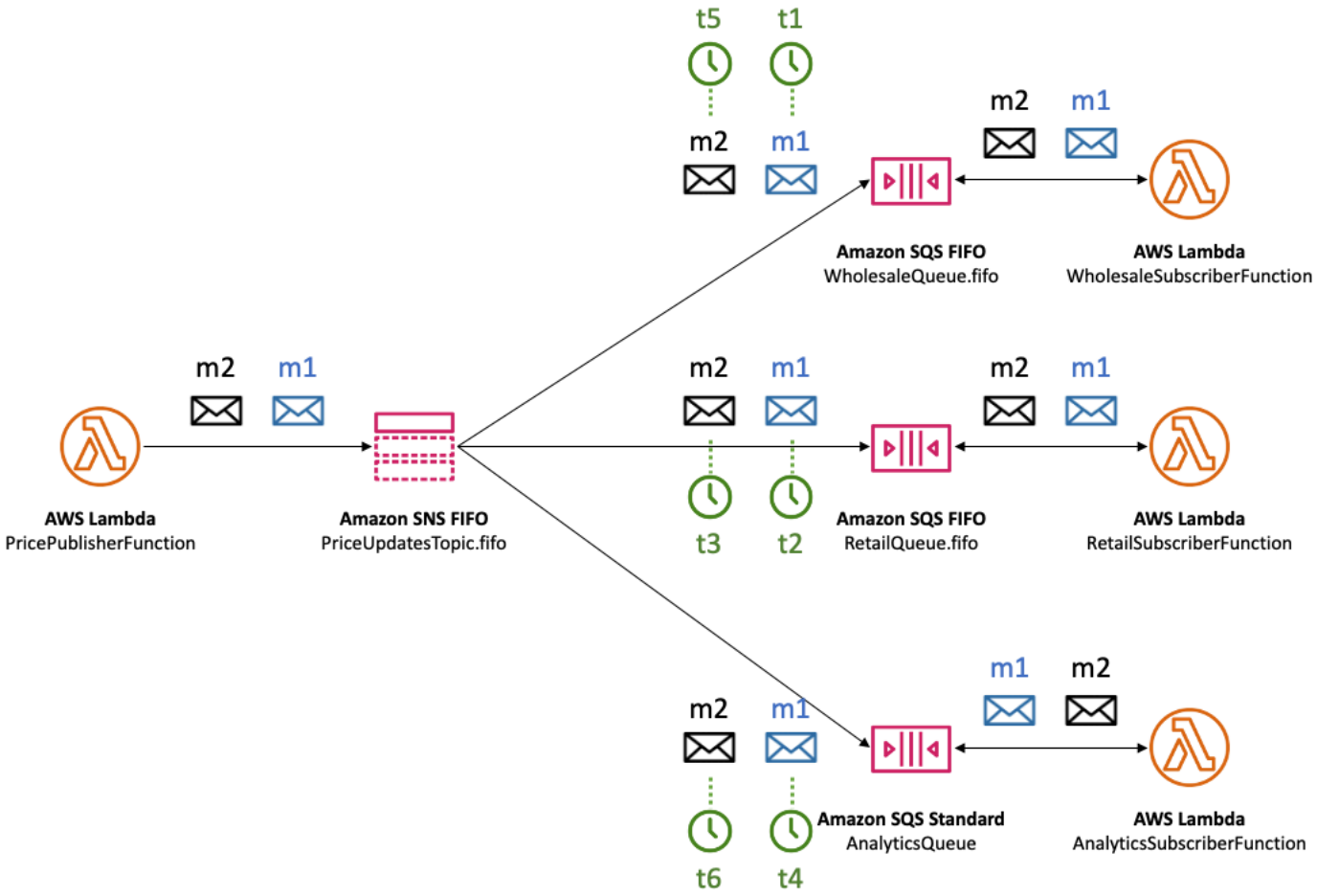
Detalhes de ordenação de mensagens do Amazon SNS para tópicos FIFO

Um tópico FIFO do Amazon SNS sempre entrega mensagens para filas do Amazon SQS inscritas na ordem exata em que as mensagens são publicadas no tópico e somente uma vez. Com uma fila FIFO do Amazon SQS assinada, o consumidor da fila recebe as mensagens na ordem exata em que são entregues à fila, sem duplicatas. No entanto, com uma fila padrão do Amazon SQS assinada, o consumidor da fila pode receber mensagens fora de ordem e mais de uma vez. Isso permite uma maior dissociação entre assinantes e editores, oferecendo aos assinantes maior flexibilidade em termos de consumo de mensagens e otimização de custos, conforme mostrado no diagrama a seguir, com base no Exemplo de caso de uso do tópico FIFO do Amazon SNS.

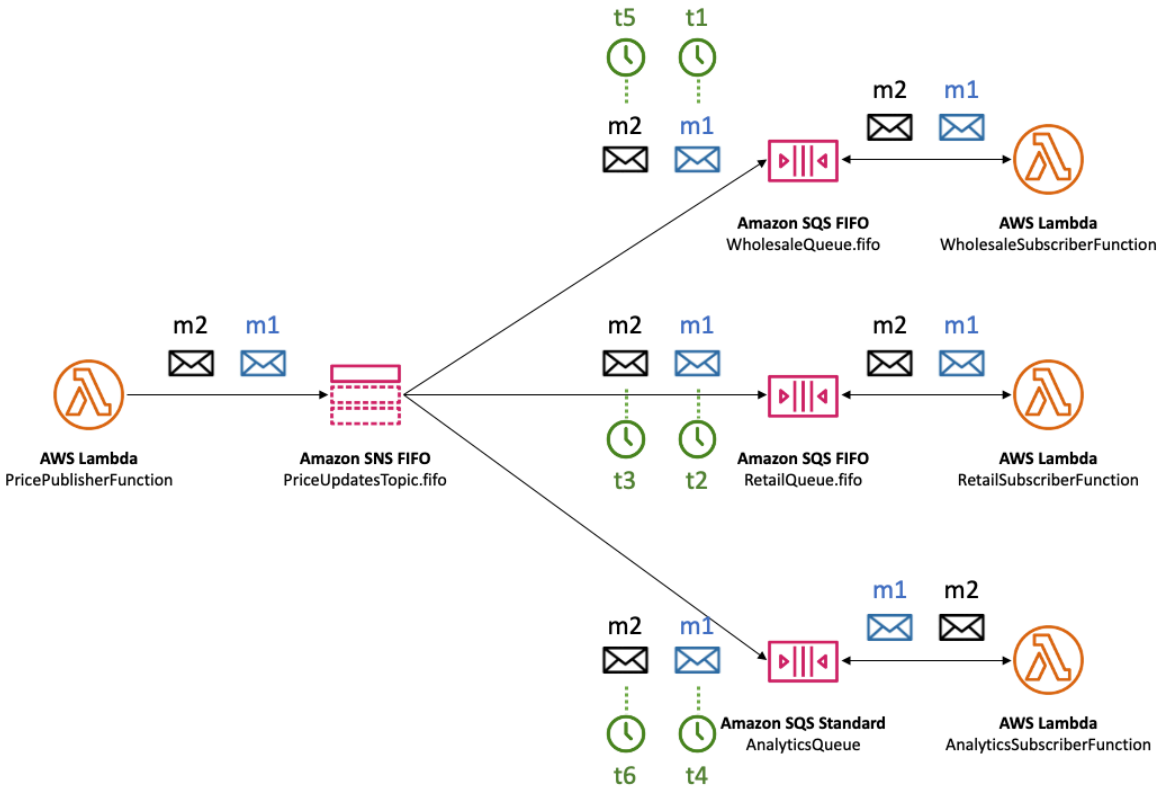


Observe que não há nenhuma ordenação implícita dos assinantes. O exemplo a seguir mostra que essa mensagem m1 é entregue primeiro ao assinante atacadista e, depois, ao assinante varejista e,

então ao assinante analista. A mensagem m2 é entregue primeiro ao assinante varejista, depois, ao assinante atacadista e, por fim, ao assinante analista. Embora as duas mensagens sejam entregues aos assinantes em uma ordem diferente, a sequência das mensagens é preservada para cada assinante de FIFO do Amazon SQS. Cada assinante é percebido isoladamente de qualquer outro assinante.

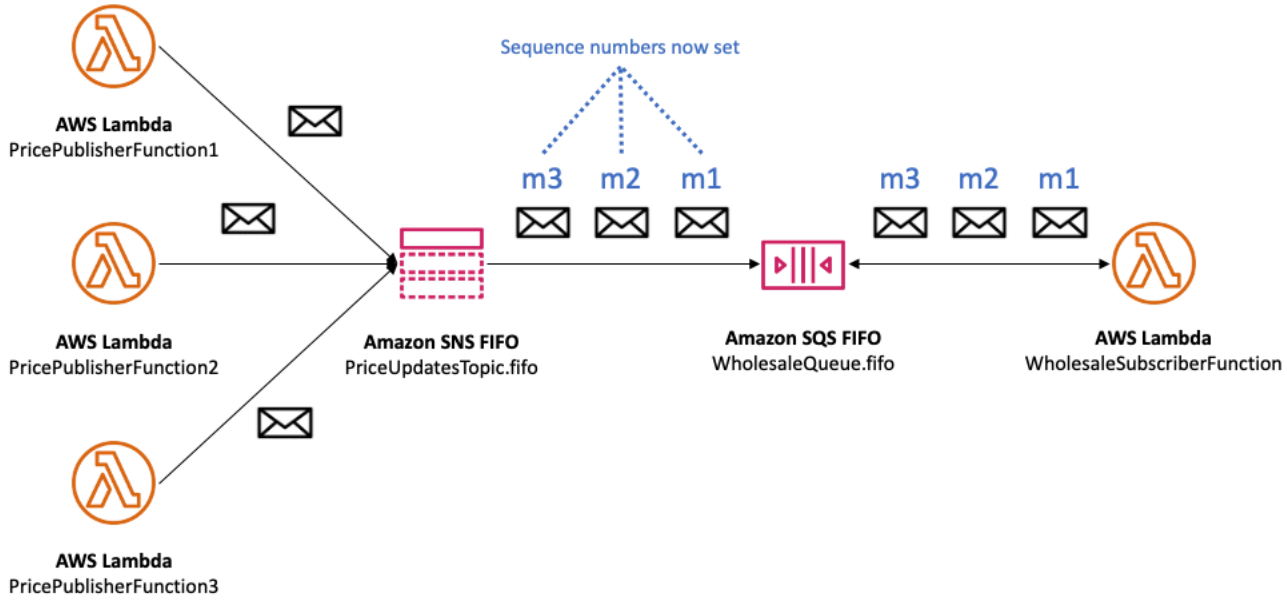


Se um assinante da fila do Amazon SQS se tornar inacessível, ele poderá ficar fora de sincronia. Por exemplo, digamos que o proprietário da fila da aplicação de atacado altere erroneamente a política de fila do Amazon SQS de uma maneira que impede que o principal de serviço do Amazon SNS entregue mensagens para a fila. Nesse caso, as entregas de atualização de preço à fila de atacado falham, enquanto as entregas às filas de varejo e análise são bem-sucedidas, fazendo com que os assinantes fiquem fora de sincronia. Quando o proprietário da fila da aplicação de atacado corrige a política de fila, o Amazon SNS retoma a entrega de mensagens para a fila inscrita. Todas as mensagens publicadas no tópico que se destinam à fila configurada incorretamente são descartadas, a menos que a assinatura tenha uma fila de mensagens não entregues configurada.



Você pode ter várias aplicações (ou várias threads dentro da mesma aplicação) publicando mensagens em um tópico FIFO do SNS em paralelo. Ao fazer isso, você efetivamente delega o sequenciamento de mensagens ao serviço Amazon SNS. Para determinar a sequência estabelecida de mensagens, você pode verificar o número de sequência.

O número de sequência é um número grande, não consecutivo que o Amazon SNS atribui a cada mensagem. O tamanho do número de sequência é de 128 bits e continua aumentando para cada grupo de mensagens. O número de sequência é passado para as filas do Amazon SQS inscritas como parte do corpo da mensagem. No entanto, se você habilitar a entrega de mensagens brutas, a mensagem entregue à fila do Amazon SQS não incluirá o número de sequência nem nenhum outro metadado de mensagens do Amazon SNS.



Os tópicos FIFO do Amazon SNS definem pedidos no contexto de um grupo de mensagens. Para obter mais informações, consulte Agrupamento de mensagens do Amazon SNS para tópicos FIFO.

Agrupamento de mensagens do Amazon SNS para tópicos FIFO

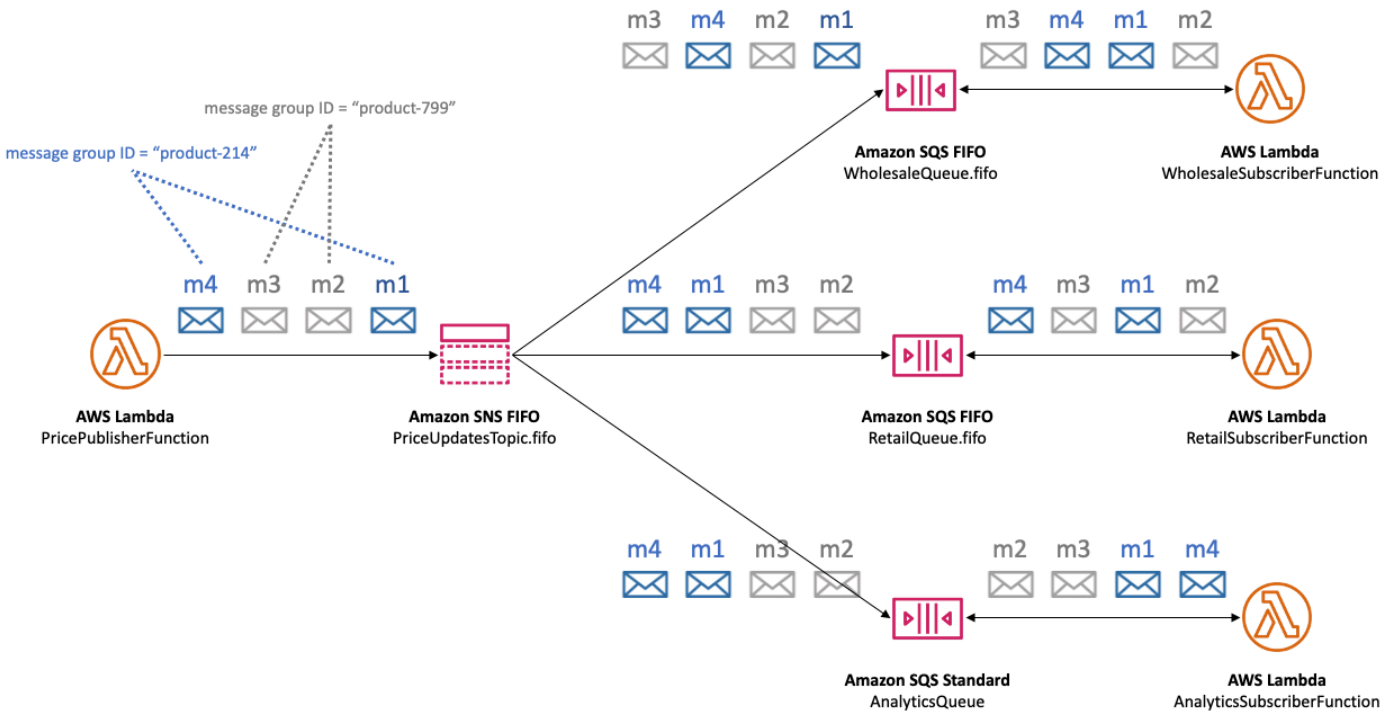
As mensagens que pertencem ao mesmo grupo são processadas uma a uma, em uma ordem estrita em relação ao grupo.

Ao publicar mensagens em um tópico FIFO do Amazon SNS, você define o ID do grupo de mensagens. O ID do grupo é um token obrigatório que determina que uma mensagem pertence a um grupo de mensagens específico. O tópico FIFO do SNS passa o ID do grupo para as filas FIFO inscritas do Amazon SQS. Não há limite para o número de grupos IDs nos tópicos do SNS FIFO ou nas filas do SQS FIFO. O ID do grupo de mensagens não é passado para filas padrão do Amazon SQS.

Não há afinidade entre um grupo de mensagens e uma assinatura. Portanto, as mensagens publicadas em qualquer grupo de mensagens são entregues a todas as filas inscritas, sujeitas a quaisquer políticas de filtro anexadas a assinaturas. Para ter mais informações, consulte Entrega de mensagens do Amazon SNS para tópicos FIFO e Filtragem de mensagens do Amazon SNS para tópicos FIFO do SNS.

No exemplo de caso de uso de gerenciamento de preços de peças automotivas, há um grupo de mensagens dedicado para cada produto vendido na plataforma. O mesmo tópico FIFO do Amazon SNS é usado para processar todas as atualizações de preços. A sequência de atualizações de

preços é preservada no contexto de um único produto de peças automotivas, mas não em vários produtos. O diagrama a seguir mostra como isso funciona. Observe que, para o produto cujo ID de grupo de mensagens é produto-214, a mensagem m1 é processada antes de m4. Essa sequência é preservada em todos os fluxos de trabalho que usam FIFO do Amazon SNS a FIFO do Amazon SQS. Da mesma forma, para o produto cujo ID do grupo de mensagens é product-799, a mensagem m2 é processada antes de m3. No entanto, ao usar as filas padrão do Amazon SQS, a ordem das mensagens não é mais garantida e os grupos de mensagens não existem. Os grupos de mensagens product-214 e product-799 são independentes entre si, portanto, não há relação entre a forma como as mensagens são sequenciadas.



Distribuindo dados por grupo de mensagens IDs para melhorar o desempenho

Para otimizar o throughput de entrega, os tópicos FIFO do Amazon SNS entregam mensagens de diferentes grupos de mensagens em paralelo, enquanto a ordem das mensagens é estritamente mantida dentro de cada grupo de mensagens. Cada grupo de mensagens individual pode entregar no máximo 300 mensagens por segundo. Portanto, para obter alta taxa de transferência para um único tópico, use um grande número de grupos IDs de mensagens distintos. Ao utilizar um

conjunto diversificado de grupos de mensagens, os tópicos FIFO do Amazon SNS distribuem automaticamente as mensagens em um número maior de partições paralelas.

> **Note**
>
> Os tópicos FIFO do Amazon SNS são otimizados para distribuição uniforme de mensagens entre grupos de mensagens IDs, independentemente do número de grupos. AWS recomenda que você use um grande número de grupos de mensagens distintos IDs para otimizar o desempenho.

Ao publicar em seu tópico FIFO do Amazon SNS com alto throughput e uma ou mais filas FIFO do Amazon SQS estarem assinadas, é recomendável que você habilite o alto throughput nas filas. Para obter mais informações, consulte Alta taxa de transferência para filas FIFO no Guia do desenvolvedor do Amazon Simple Queue Service.

Entrega de mensagens do Amazon SNS para tópicos FIFO

Os tópicos FIFO (first in, first out) do Amazon SNS são compatíveis com a entrega para as filas padrão e FIFO do Amazon SQS para oferecer aos clientes flexibilidade e controle ao integrar aplicações distribuídas que exigem consistência de dados quase em tempo real.

Para workloads que precisam preservar a ordem estrita de mensagens ou a desduplicação, a combinação dos tópicos FIFO do Amazon SNS com filas FIFO do Amazon SQS, inscritas como endpoint de entrega, aprimora o sistema de mensagens entre aplicações quando a ordem das operações e dos eventos é crucial ou quando não se admitem duplicatas.

Para cargas de trabalho que toleram pedidos e at-least-once entregas com o melhor esforço possível, inscrever filas padrão do Amazon SQS nos tópicos FIFO do Amazon SNS oferece a capacidade de reduzir custos, além de compartilhar filas entre cargas de trabalho que não utilizam FIFO.

> **Note**
>
> Para distribuir mensagens de tópicos FIFO do Amazon SNS para AWS Lambda funções, etapas adicionais são necessárias. Primeiro, inscreva as filas padrão ou FIFO do Amazon SQS no tópico. Em seguida, configure as filas para acionar as funções. Para obter mais

informações, consulte SQS FIFO as an event source (“FIFO do SQS como fonte de eventos”) no Blog de computação da AWS .

Os tópicos FIFO do SNS não podem entregar mensagens para endpoints gerenciados pelo cliente, como endereços de e-mail, aplicativos móveis, números de telefone para sistemas de mensagens de texto (SMS) ou endpoints HTTP(S). Esses tipos de endpoint não têm garantia de preservar a ordenação estrita de mensagens. As tentativas de inscrever endpoints gerenciados pelo cliente nos tópicos FIFO do SNS resultam em erros.

Os tópicos FIFO do SNS oferecem suporte aos mesmos recursos de filtragem de mensagens que os tópicos padrão. Para obter mais informações, consulte Filtragem de mensagens do Amazon SNS para tópicos FIFO do SNS e Simplify Your Pub/Sub Messaging with Amazon SNS Message Filtering (“Simplificar seu sistema de mensagens Pub/Sub com a filtragem de mensagens do Amazon SNS”) no Blog de computação da AWS .

Filtragem de mensagens do Amazon SNS para tópicos FIFO do SNS

Os tópicos FIFO do Amazon SNS oferecem suporte à filtragem de mensagens. O uso da filtragem de mensagens simplifica sua arquitetura descarregando a lógica de roteamento de mensagens dos sistemas do editor e a lógica de filtragem de mensagens de seus sistemas assinantes.

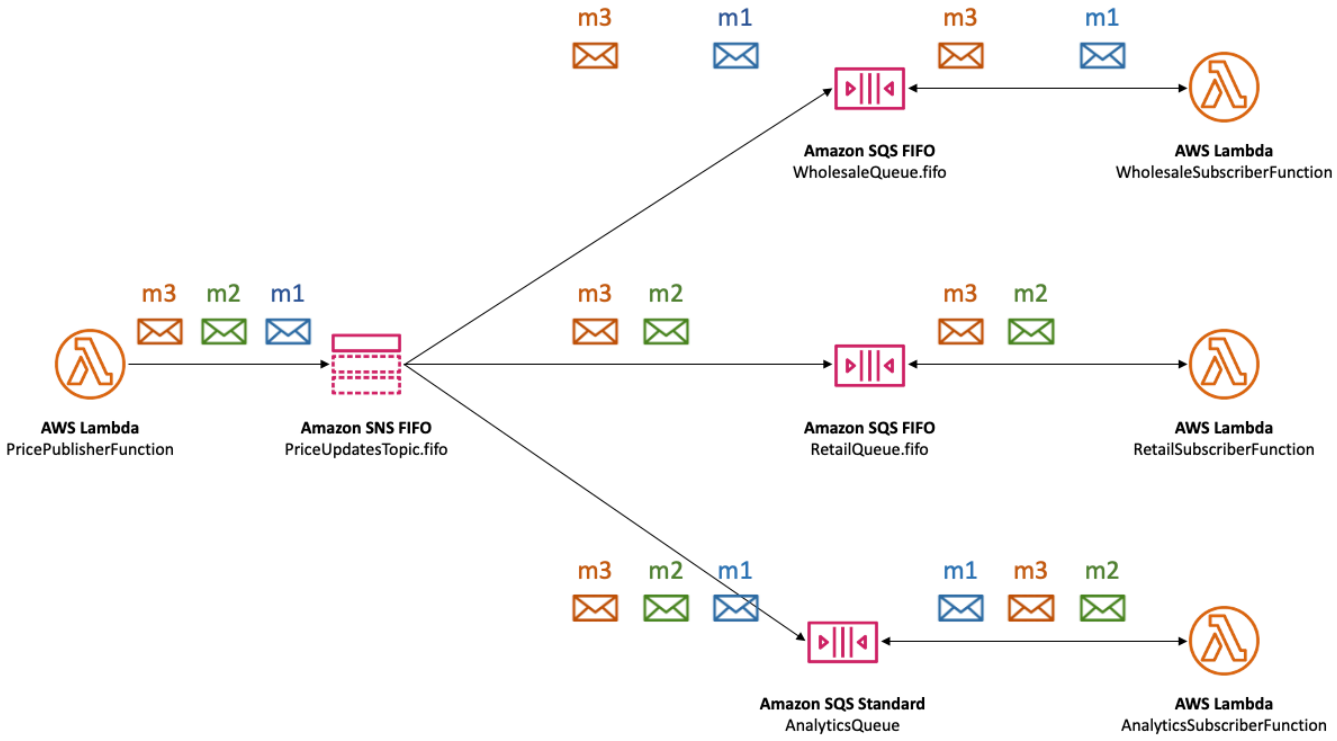
Ao inscrever uma fila FIFO ou padrão do Amazon SQS em um tópico FIFO do SNS, você pode usar a filtragem de mensagens para especificar que o assinante receba um subconjunto de mensagens, em vez de todas elas. Cada assinante pode definir sua própria política de filtro como atributos de assinatura. Com base no escopo da política de filtro, a política de filtro é comparada aos atributos ou ao corpo da mensagem recebida. Se a política de filtro estabelecer correspondência, o tópico entregará uma cópia da mensagem ao assinante. Se não houver correspondência, o tópico não entregará uma cópia da mensagem.

No exemplo de caso de uso de gerenciamento de preços de peças automotivas, suponha que as seguintes políticas de filtro do Amazon SNS estejam definidas e que o escopo de política seja `MessageBody`:

- Para a fila de atacado, a política de filtro `{"business":["wholesale"]}` estabelece correspondência com cada mensagem que contém uma chave chamada `business` e com `wholesale` no conjunto de valores. No diagrama a seguir, uma das chaves na mensagem m1 é

`business` com um valor de `wholesale`. Uma das chaves na mensagem m3 é `business` com um valor de `["wholesale,retail"]`. Assim, tanto m1 quanto m3 correspondem aos critérios da política de filtro e ambas as mensagens são entregues à fila de atacado.

- Para a fila de varejo, a política de filtro `{"business":["retail"]}` estabelece correspondência com cada mensagem que contém uma chave chamada `business` e com `retail` no conjunto de valores. No diagrama, uma das chaves na mensagem m2 é `business` com um valor de `retail`. Uma das chaves na mensagem m3 é `business` com o valor de `["wholesale,retail"]`. Assim, tanto m2 quanto m3 correspondem aos critérios da política de filtro e ambas as mensagens são entregues à fila de varejo.
- Com relação à fila de análise, queremos que o Amazon Athena receba todos os registros, para que nenhuma política de filtro seja aplicada.



Os tópicos FIFO do SNS oferecem suporte a uma variedade de operadores correspondentes, incluindo valores de string de atributo, valores numéricos de atributo e chaves de atributo. Para obter mais informações, consulte Filtragem de mensagens do Amazon SNS.

Os tópicos FIFO do SNS não entregam mensagens duplicadas para endpoints inscritos. Para obter mais informações, consulte Desduplicação de mensagens do Amazon SNS para tópicos FIFO.

Desduplicação de mensagens do Amazon SNS para tópicos FIFO

Os tópicos FIFO do Amazon SNS e as filas FIFO do Amazon SQS oferecem suporte à desduplicação de mensagens, que fornece entrega e processamento de mensagens exatamente uma vez, desde que as seguintes condições sejam atendidas:

- A fila FIFO do Amazon SQS inscrita existe e tem permissões para que a entidade principal de serviço do Amazon SNS entregue mensagens para a fila.
- O consumidor de fila FIFO do Amazon SQS processa a mensagem e a exclui da fila antes que o tempo limite de visibilidade expire.
- O tópico de assinatura do Amazon SNS não tem filtragem de mensagens. Quando você configura a filtragem de mensagens, os tópicos FIFO do Amazon SNS at-most-once oferecem suporte à entrega, pois as mensagens podem ser filtradas com base nas suas políticas de filtro de assinatura.
- Não há interrupções de rede que impeçam a confirmação da entrega da mensagem.

> **Note**
>
> A desduplicação de mensagens se aplica a um tópico FIFO inteiro do Amazon SNS quando o atributo do tópico está definido como. `FifoThroughputScope` `Topic` Quando o atributo de tópico `FifoThroughputScope` é definido como`MessageGroup`, a desduplicação de mensagens se aplica a cada grupo de mensagens individual.

Quando você publica uma mensagem em um tópico FIFO do Amazon SNS, a mensagem deve incluir um ID de desduplicação. Esse ID está incluído na mensagem que o tópico FIFO do Amazon SNS entrega às filas FIFO do Amazon SQS inscritas.

Se uma mensagem com uma determinada ID de desduplicação for publicada com sucesso em um tópico FIFO do Amazon SNS, qualquer mensagem publicada com a mesma ID de desduplicação, dentro do intervalo de desduplicação de cinco minutos, será aceita, mas não entregue. O tópico FIFO do Amazon SNS continua rastreando o ID de desduplicação da mensagem, no escopo de desduplicação configurado pelo atributo tópico`FifoThroughputScope`, mesmo depois que a mensagem é entregue aos endpoints assinados.

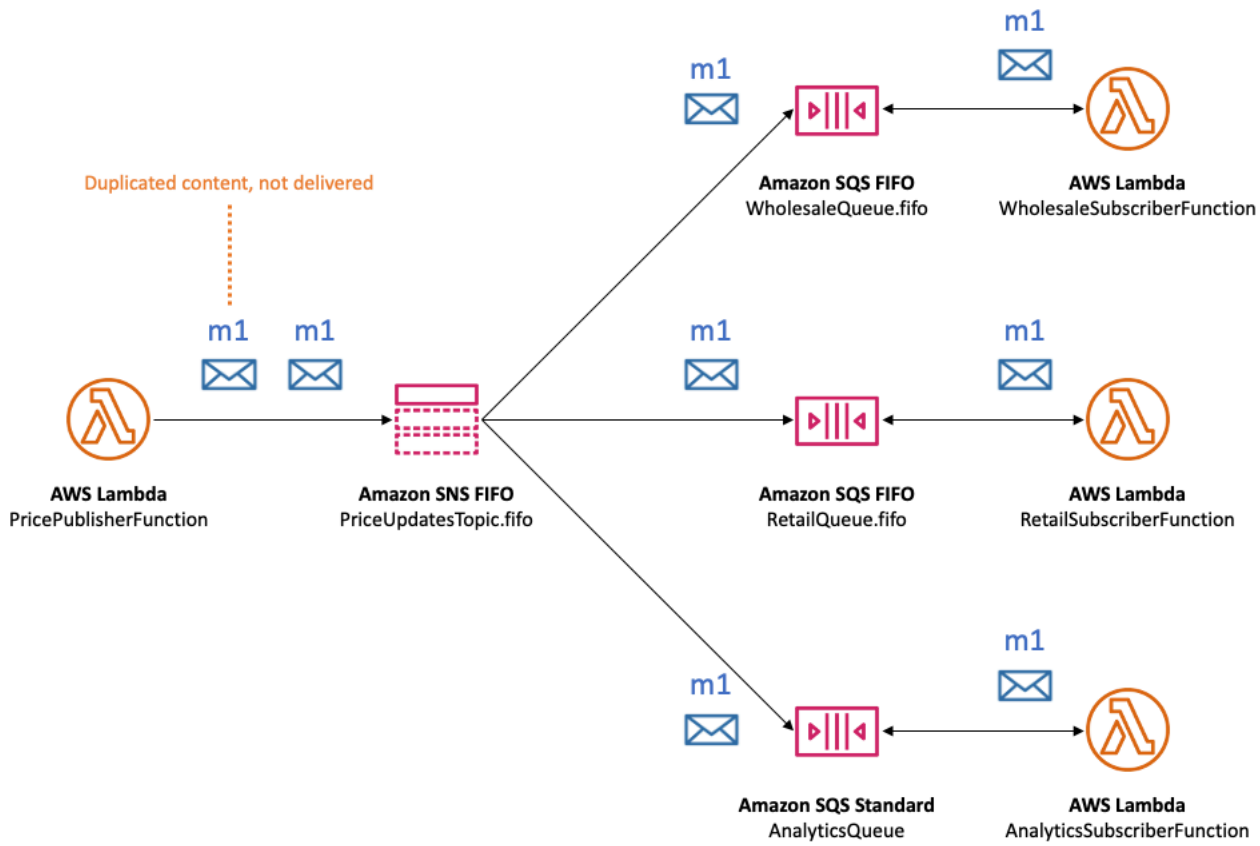
Se houver garantia de que o corpo da mensagem será exclusivo para cada mensagem publicada, você poderá habilitar a desduplicação baseada em conteúdo para um tópico FIFO do Amazon SNS

e as filas FIFO do Amazon SQS inscritas. O Amazon SNS usa o corpo da mensagem para gerar um valor de hash exclusivo a ser usado como o ID de desduplicação para cada mensagem, portanto, você não precisa definir um ID de desduplicação ao enviar cada mensagem.

> **Note**
>
> Os atributos de mensagem não estão incluídos no cálculo de hash.

Quando a desduplicação baseada em conteúdo é habilitada para um tópico FIFO do Amazon SNS e uma mensagem é publicada com uma ID de desduplicação, a ID de desduplicação publicada substitui a ID de desduplicação baseada em conteúdo gerada.

No caso de uso de exemplo de gerenciamento de preços de peças automotivas, a empresa deve definir um ID de desduplicação universalmente exclusivo para cada atualização de preço. Isso ocorre porque o corpo da mensagem pode ser idêntico mesmo quando o atributo de mensagem é diferente para atacado e varejo. No entanto, se a empresa adicionasse o tipo de negócio (atacado ou varejo) ao corpo da mensagem com o ID e o preço do produto, ela poderia permitir a duplicação baseada em conteúdo no tópico FIFO do Amazon SNS e nas filas FIFO do Amazon SQS inscritas.



Além da ordenação e desduplicação de mensagens, os tópicos FIFO do Amazon SNS oferecem suporte à criptografia do lado do servidor (SSE) de mensagens com chaves e à privacidade de mensagens por meio de VPC endpoints AWS KMS com. AWS PrivateLink Para obter mais informações, consulte Segurança de mensagens do Amazon SNS para tópicos FIFO.

Segurança de mensagens do Amazon SNS para tópicos FIFO

Você pode habilitar a criptografia para tópicos FIFO do Amazon SNS e filas FIFO do Amazon SQS usando AWS Key Management Service () chaves mestras do cliente ()AWS KMS. CMKs

- Você pode criar novos tópicos e filas de FIFO criptografados ou ativar a criptografia para os existentes.
- Somente o corpo da mensagem é criptografado. Os atributos da mensagem, os metadados dos recursos e as métricas dos recursos permanecem sem criptografia.

> **Note**
>
> Adicionar criptografia a um tópico ou fila FIFO existente não criptografa nenhuma mensagem armazenada no backlog, e remover criptografia de um tópico ou fila deixa as mensagens armazenadas em backlog criptografadas.

Os tópicos FIFO do SNS descriptografam as mensagens imediatamente antes de entregá-las aos endpoints inscritos. As filas FIFO do SQS descriptografam a mensagem antes de retorná-las à aplicação consumidora. Para obter mais informações, consulte Criptografia de dados do Amazon SNS e a publicação Mensagens de criptografia publicadas no Amazon SNS com AWS KMS no Blog de computação da AWS .

Além disso, os tópicos FIFO do SNS e as filas FIFO do SQS oferecem suporte à privacidade de mensagens com endpoints da VPC de interface com tecnologia do AWS PrivateLink. Usando endpoints de interface, você pode enviar mensagens de sub-redes da Amazon Virtual Private Cloud (Amazon VPC) para tópicos e filas FIFO sem atravessar a Internet pública. Esse modelo mantém suas mensagens dentro da AWS infraestrutura e da rede, o que aumenta a segurança geral do seu aplicativo. Ao usar AWS PrivateLink, você não precisa configurar um gateway de internet, tradução de endereços de rede (NAT) ou rede privada virtual (VPN). Para obter mais informações, consulte Proteger o tráfego do Amazon SNS com endpoints da VPC e a publicação Mensagens de segurança publicadas no Amazon SNS com AWS PrivateLink no Blog de computação da AWS .

Os tópicos FIFO do SNS também oferecem suporte a filas de mensagens mortas e ao armazenamento de mensagens nas zonas de disponibilidade. Para obter mais informações, consulte Durabilidade de mensagens do Amazon SNS para tópicos FIFO.

Durabilidade de mensagens do Amazon SNS para tópicos FIFO

Os tópicos FIFO do Amazon SNS e as filas FIFO do Amazon SQS são duráveis. Os dois tipos de recursos armazenam mensagens de maneira redundante em várias zonas de disponibilidade e fornecem filas de mensagens mortas para lidar com casos excepcionais.

No Amazon SNS, a entrega de mensagens falha quando o tópico do Amazon SNS não pode acessar uma fila do Amazon SQS inscrita devido a um erro do lado do cliente ou um erro do lado do servidor:

- Os erros do lado do cliente ocorrem quando o tópico FIFO do Amazon SNS tem metadados de inscrição obsoletos. Duas causas comuns de erros do lado do cliente ocorrem quando o proprietário da fila do Amazon SQS executa um dos seguintes procedimentos:

  - Exclui a fila.
  - Altera a política de filas impedindo que o principal de serviço do Amazon SNS entregue mensagens a ela.

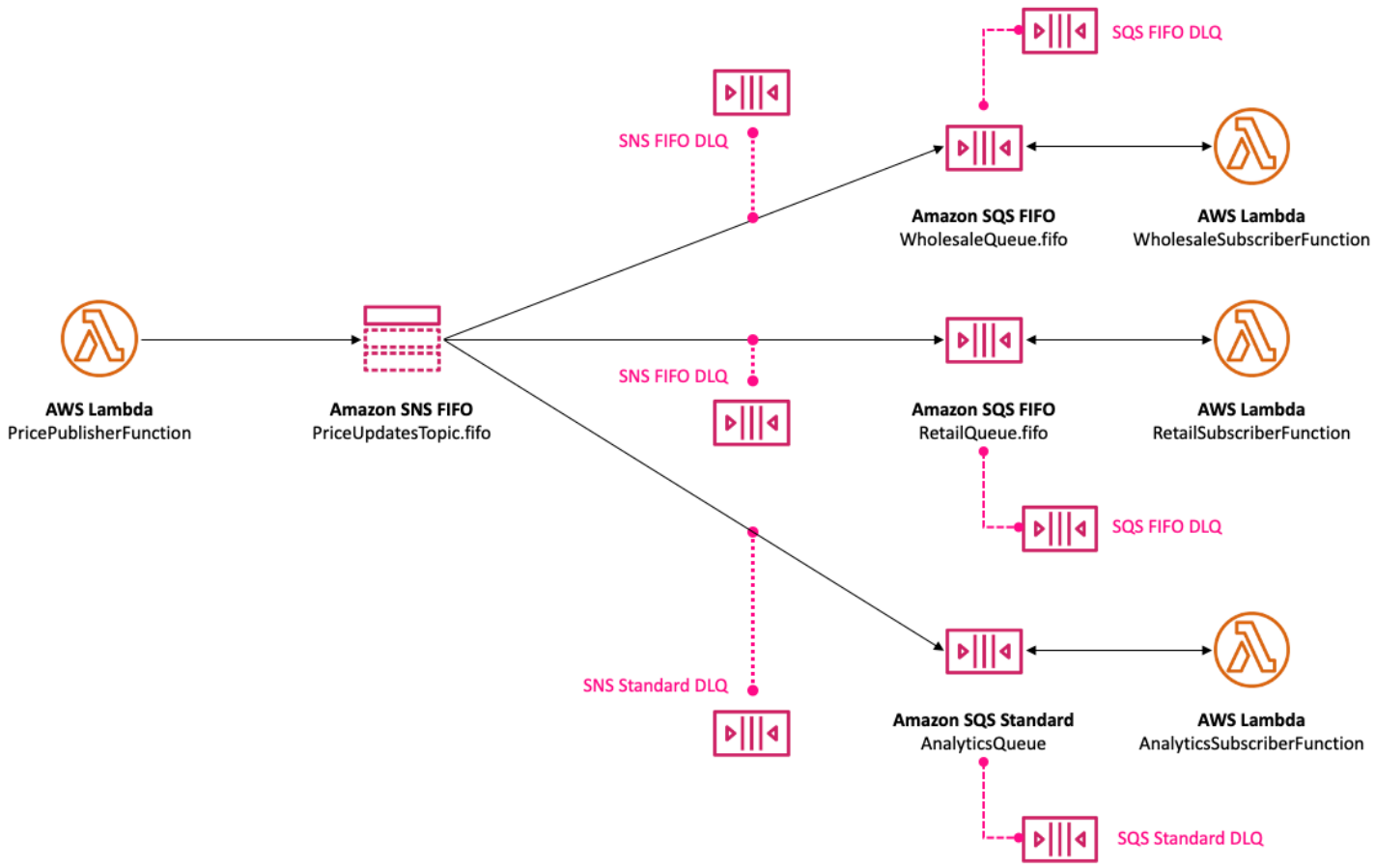
  O Amazon SNS não tenta entregar mensagens que falharam devido a erros do lado do cliente.
- Erros do lado do servidor podem ocorrer nestas situações:
  - O serviço Amazon SQS não está disponível.
  - O Amazon SQS falha ao processar uma solicitação válida do serviço Amazon SNS.

  Quando ocorrem erros do lado do servidor, os tópicos FIFO do Amazon SNS tentam executar as entregas com falha novamente até 100.015 vezes ao longo de 23 dias. Para obter mais informações, consulte Novas tentativas de entrega de mensagens do Amazon SNS.

Para qualquer tipo de erro, o Amazon SNS pode segregar mensagens nas filas de mensagens mortas do Amazon SQS para que os dados não sejam perdidos.

No Amazon SQS, o processamento de mensagens falha quando a aplicação consumidora não consegue receber a mensagem, processá-la e excluí-la da fila. Quando o número máximo de solicitações de recebimento falhar, o Amazon SQS pode colocar mensagens em filas de mensagens mortas para que os dados não sejam perdidos.

No exemplo de caso de uso de gerenciamento de preços de peças automotivas, a empresa pode atribuir uma fila de mensagens não entregues (DLQ) do Amazon SQS a cada assinatura de tópico FIFO do Amazon SNS, bem como a cada fila do Amazon SQS inscrita. Isso protege a empresa contra qualquer perda de atualização de preço.



A fila de mensagens não entregues associada a uma assinatura do Amazon SNS deve ser uma fila do Amazon SQS do mesmo tipo da fila de assinatura. Por exemplo, a assinatura FIFO do Amazon SNS de uma fila FIFO do Amazon SQS deve ter uma fila FIFO do Amazon SQS como a fila de mensagens não entregues. Da mesma forma, a assinatura FIFO do Amazon SNS de uma fila padrão do Amazon SQS deve ter uma fila padrão do Amazon SQS como fila de mensagens não entregues. Para obter mais informações, consulte a publicação Projetando aplicativos duráveis sem servidor DLQs para Amazon SNS Filas de mensagens não entregues do Amazon SNS e Amazon SQS no Compute AWS Lambda Blog.AWS

Para obter maior durabilidade para auxiliar na recuperação de falhas subsequentes, os proprietários de tópicos também podem usar tópicos FIFO para arquivar mensagens por até 365 dias. Os assinantes de tópicos podem reproduzir essas mensagens arquivadas em um endpoint inscrito com o objetivo de recuperar mensagens perdidas em decorrência de uma falha em uma aplicação subsequente ou para replicar o estado de uma aplicação existente. Para saber mais, consulte Arquivamento e reprodução de mensagens do Amazon SNS de tópicos FIFO.

Arquivamento e reprodução de mensagens do Amazon SNS de tópicos FIFO

O que é arquivamento e reprodução de mensagens?

O Amazon SNS oferece um recurso de arquivamento e reprodução de mensagens sem código, projetado especificamente para tópicos FIFO (First-In-First-Out). Esse recurso permite que os proprietários de tópicos armazenem mensagens diretamente no arquivo de tópicos por até 365 dias e as reproduzam para os assinantes quando necessário. O arquivamento e a reprodução de mensagens são essenciais para recuperar mensagens perdidas e sincronizar aplicativos entre regiões ou sistemas por meio da replicação de estados.

Essa funcionalidade pode ser acessada por meio da AWS API AWS CloudFormation, SDK e. AWS Management Console

Casos de uso principais

- Recuperação de mensagens: recupere mensagens perdidas devido a falhas de aplicativos posteriores, reproduzindo-as no endpoint do assinante.
- Replicação de estado: replique o estado de um sistema existente em um novo ambiente reproduzindo mensagens a partir de um timestamp específico.
- Correção de erros: reenvie mensagens perdidas durante interrupções para garantir que todos os eventos sejam processados corretamente.

Componentes do arquivamento e reprodução de mensagens

Gerencie o arquivamento e a repetição de mensagens para tópicos FIFO do Amazon SNS, incluindo definir períodos de retenção, monitorar o CloudWatch uso de mensagens arquivadas, iniciar repetições por meio de atributos de assinatura e entender as permissões necessárias para modificar e iniciar repetições.

Arquivamento de mensagens

- O proprietário do tópico ativa o recurso de arquivamento e define o período de retenção de mensagens, que pode ser de até 365 dias. Para saber mais, consulte Arquivamento de mensagens do Amazon SNS para proprietários de tópicos FIFO
- CloudWatch as métricas ajudam a monitorar as mensagens arquivadas.

Reprodução de mensagens

- Um assinante inicia uma repetição, selecionando a janela de tempo para as mensagens serem reprocessadas no endpoint inscrito. Para saber mais, consulte Reprodução de mensagens do Amazon SNS para assinantes de tópicos FIFO.
- Você gerencia o replay por meio de atributos de assinatura usando o recurso `ReplayPolicy`.

Permissões relevantes

- **`SetSubscriptionAttributes`**: necessário para definir ou modificar as configurações de repetição usando o atributo `ReplayPolicy` em uma assinatura.
- **`Subscribe`**: necessário para anexar uma nova assinatura e iniciar os replays.
- **`GetTopicAttributes`**: permite visualizar as propriedades do tópico, mas o início da repetição gira principalmente em torno do gerenciamento de assinaturas.

Arquivamento de mensagens do Amazon SNS para proprietários de tópicos FIFO

O arquivamento de mensagens oferece a capacidade de arquivar uma única cópia de todas as mensagens publicadas no tópico. É possível armazenar mensagens publicadas no tópico habilitando a política de arquivamento de mensagens no tópico, o que permite o arquivamento de mensagens para todas as assinaturas vinculadas a esse tópico. As mensagens podem ser arquivadas por um período mínimo de 1 dia a, no máximo, 365 dias.

Há cobranças adicionais ao definir uma política de arquivamento. Para obter informações sobre preços, consulte Definição de preço do Amazon SNS.

Crie uma política de arquivamento de mensagens usando o AWS Management Console

Use essa opção para criar uma política de arquivamento de mensagens com o AWS Management Console.

1. Faça login no console do Amazon SNS.
2. Selecione um tópico ou crie um. Para saber mais sobre como criar tópicos, consulte Criar um tópico do Amazon SNS.

> **Note**
>
> O arquivamento e a reprodução de mensagens do Amazon SNS só estão disponíveis para tópicos de FIFO application-to-application (A2A).

3. Na página Editar tópico, expanda a seção Política de arquivamento.
4. Ative o recurso Política de arquivamento e insira o número de dias durante os quais você deseja arquivar mensagens no tópico.
5. Escolha Salvar alterações.

Como visualizar, editar e desativar uma política de tópico de arquivamento de mensagens

- Na página Detalhes do tópico, a Política de retenção exibe o status da política de arquivamento, incluindo o número de dias para os quais ela está definida. Selecione a guia Política de arquivamento para ver os seguintes detalhes do arquivamento de mensagens:
  - Status: o status de arquivamento e reprodução aparece como ativo quando uma política de arquivamento é aplicada. O status de arquivamento e reprodução aparece como inativo quando a política de arquivamento é definida como um objeto JSON vazio.
  - Período de retenção de mensagens: o número especificado de dias de retenção de mensagens.
  - Data de início do arquivamento: a data a partir da qual os assinantes podem reproduzir as mensagens.
  - Pré-visualização de JSON: a pré-visualização JSON da política de arquivamento.
- (Opcional) Para editar uma política de arquivamento, acesse a página de resumo do tópico e selecione Editar.
- (Opcional) Para desativar uma política de arquivamento, acesse a página de resumo do tópico e selecione Editar. Desative a Política de arquivamento e selecione Salvar alterações.
- (Opcional) Para excluir um tópico com uma política de arquivamento, é necessário primeiro desativar a política de arquivamento conforme descrito anteriormente.

> **Important**
>
> Para evitar exclusões acidentais de mensagens, você não pode excluir um tópico com uma política de arquivamento de mensagens ativa. A política de arquivamento de mensagens do tópico deve ser desativada para que o tópico possa ser excluído. Quando você desativa uma política de arquivamento de mensagens, o Amazon SNS exclui todas as mensagens

arquivadas. Ao excluir um tópico, as assinaturas são removidas e nenhuma mensagem em trânsito pode ser entregue.

Criar uma política de arquivamento de mensagens usando a API

Para criar uma política de arquivamento de mensagens com a API, é necessário adicionar o atributo `ArchivePolicy` ao tópico. É possível definir uma `ArchivePolicy` usando as ações da API `CreateTopic` e `SetTopicAttributes`. A `ArchivePolicy` tem um valor único, `MessageRetentionPeriod`, que representa o número de dias pelos quais o Amazon SNS retém as mensagens. Para ativar o arquivamento de mensagens para o tópico, defina o `MessageRetentionPeriod` como um valor inteiro maior que zero. Por exemplo, para reter mensagens em seu arquivo por 30 dias, defina a `ArchivePolicy` como:

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

Para desativar o arquivamento de mensagens do tópico e limpar o arquivo, cancele a definição da `ArchivePolicy` da seguinte forma:

```
{}
```

Criar uma política de arquivamento de mensagens usando o SDK

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Compartilhados `config` e `credentials` arquivos no Guia de referência de ferramentas AWS SDKs e ferramentas.

O exemplo de código a seguir mostra como definir a `ArchivePolicy` para um tópico do Amazon SNS com o objetivo de reter todas as mensagens publicadas no tópico por 30 dias.

```
// 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);
```

Crie uma política de arquivamento de mensagens usando AWS CloudFormation

Para criar uma política de arquivamento usando, AWS CloudFormation consulte `AWS::SNS::Topic`o Guia AWS CloudFormation do Usuário.

Conceder acesso a um arquivo criptografado

Para que um assinante possa começar a reproduzir mensagens de um tópico criptografado, siga as etapas abaixo. Como as mensagens passadas são reproduzidas, o Amazon SNS precisa ter acesso `Decrypt` provisionado à chave do KMS que foi usada para criptografar as mensagens no arquivo.

1. Ao criptografar mensagens com uma chave do KMS e armazená-las no tópico, é necessário conceder ao Amazon SNS a capacidade de descriptografar essas mensagens por meio da política de chave. Para saber mais, consulte Conceder permissões de descriptografia ao Amazon SNS.
2. Habilite AWS KMS para o Amazon SNS. Para saber mais, consulte Configurando permissões AWS KMS.

> **Important**
>
> Quando você adicionar as novas seções à política de chave do KMS, não altere as seções existentes na política. Se a criptografia estiver habilitada em um tópico e a chave do KMS estiver desabilitada ou tiver sido excluída, ou a política de chave do KMS não estiver configurada corretamente para o Amazon SNS, o Amazon SNS não poderá reproduzir mensagens para seus assinantes.

Conceder permissões de descriptografia ao Amazon SNS

Para que o Amazon SNS acesse mensagens criptografadas de dentro do arquivo do tópico e as reproduza nos endpoints inscritos, é necessário habilitar o princípio de serviço do Amazon SNS para descriptografar essas mensagens.

Veja a seguir um exemplo da política que é necessária para permitir que a entidade principal do serviço do Amazon SNS descriptografe mensagens armazenadas durante uma reprodução de mensagens históricas do tópico.

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

Monitore métricas de arquivamento de mensagens usando a Amazon CloudWatch

Você pode monitorar mensagens arquivadas usando a Amazon CloudWatch usando as seguintes métricas. Para ser notificado sobre anomalias em suas cargas de trabalho e ajudar a evitar impactos, você pode configurar os CloudWatch alarmes da Amazon com base nessas métricas. Consulte mais detalhes em Registrar em log e monitorar no Amazon SNS.

Métrica	Descrição
ApproximateNumberOfMessagesArchived	Fornece ao proprietário do tópico o número agregado de mensagens arquivadas no arquivo de tópicos, com resolução de 60 minutos.
ApproximateNumberOfBytesArchived	Fornece ao proprietário do tópico o número agregado de bytes arquivados, em todas as mensagens no arquivo de tópicos, com resolução de 60 minutos.

Métrica	Descrição
NumberOfMessagesArchiveProcessing	Fornece ao proprietário do tópico o número de mensagens salvas no arquivo do tópico durante o intervalo em resolução de 1 minuto.
NumberOfBytesArchiveProcessing	Fornece ao proprietário do tópico o número agregado de bytes salvos no arquivo do tópico durante o intervalo em resolução de 1 minuto.

A API `GetTopicAttributes` tem uma propriedade `BeginningArchiveTime`, que representa o carimbo de data e hora mais antigo no qual um assinante pode iniciar uma reprodução. O seguinte exemplo representa uma resposta para essa ação de API:

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

Reprodução de mensagens do Amazon SNS para assinantes de tópicos FIFO

O Amazon SNS replay permite que assinantes de tópicos recuperem e reentreguem mensagens arquivadas do armazenamento de dados de tópicos para um endpoint inscrito.

- As mensagens podem ser reproduzidas imediatamente após a criação da assinatura.
- Uma mensagem repetida mantém o mesmo conteúdo e o mesmo que `Timestamp` a original. `MessageId`
- A mensagem inclui um `Replayed` atributo para indicar que é uma mensagem repetida.
- Para reproduzir somente mensagens específicas, aplique uma política de filtro à sua assinatura.

Para obter mais informações sobre como filtrar mensagens, consulte Filtrar mensagens reproduzidas.

Crie uma política de repetição de mensagens usando o AWS Management Console

Use essa opção para criar uma política de reprodução de mensagens com o AWS Management Console.

1. Faça login no console do Amazon SNS.
2. Selecione uma assinatura de tópico ou crie uma. Para saber mais sobre como criar assinaturas, consulte Criação de uma assinatura em um tópico do Amazon SNS.
3. Para iniciar a reprodução de mensagens, acesse o menu suspenso Reproduzir e selecione Iniciar reprodução.
4. No modal Prazo de repetição, selecione:
   a. Escolha a data e a hora de início da repetição — Escolha a data (YYYY/MM/DDformato) e a hora (formato hh:mm:ss de 24 horas) a partir das quais você deseja começar a reproduzir as mensagens arquivadas. A hora de início deve ser posterior ao início da hora de arquivamento aproximada.
   b. (Opcional) Escolha a data e a hora de término da repetição — Escolha a data (YYYY/MM/DDformato) e a hora (formato hh:mm:ss de 24 horas) em que você deseja parar de reproduzir as mensagens arquivadas.
   c. Selecione Iniciar repetição.
5. (Opcional) Para interromper a reprodução de mensagens, acesse a página Detalhes da assinatura e selecione Interromper repetição no menu suspenso Reproduzir.
6. (Opcional) Para monitorar métricas de repetição de mensagens de dentro desse fluxo de trabalho usando CloudWatch, consulteMonitore as métricas de reprodução de mensagens usando a Amazon CloudWatch.

Como visualizar e editar uma política de reprodução de mensagens

É possível realizar as seguintes ações na página Detalhes da assinatura:

- Para visualizar o status de reprodução de mensagens, o campo Status de reprodução exibe os seguintes valores:
  - Concluído: a reprodução reenviou com êxito todas as mensagens e agora está enviando mensagens recém-publicadas.
  - Em andamento: no momento, as mensagens selecionadas estão sendo reproduzidas.
  - Com falha: não foi possível concluir a reprodução.

  - Pendente: o estado padrão durante o início da reprodução.
- (Opcional) Para modificar a política de reprodução de mensagens, acesse a página Detalhes da assinatura e selecione Iniciar reprodução no menu suspenso Reproduzir. Ao iniciar uma reprodução, a reprodução existente será substituída.

Adicionar uma política de reprodução à assinatura com a API

Para reproduzir mensagens arquivadas, use o atributo `ReplayPolicy`. A `ReplayPolicy` pode ser usada com as ações `Subscribe` e `SetSubscriptionAttributes` da API. Essa política tem os seguintes valores:

- **`StartingPoint`** (obrigatório): sinaliza de onde começar a reproduzir as mensagens.
- **`EndingPoint`** (opcional): sinaliza quando parar de reproduzir mensagens. Se `EndingPoint` for omitido, a reprodução continuará até a hora atual.
- **`PointType`** (obrigatório): define o tipo dos pontos de início e término. No momento, o valor compatível para `PointType` é `Timestamp`.

Por exemplo, para se recuperar de uma falha subsequente e reenviar todas as mensagens por um período de duas horas em 1.º de outubro de 2023, use a ação `SetSubscriptionAttributes` da API para definir uma `ReplayPolicy` da seguinte maneira:

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

Para reproduzir todas as mensagens enviadas ao tópico a partir de 1.º de outubro de 2023 e continuar recebendo todas as mensagens recém-publicadas no tópico, use a ação `SetSubscriptionAttributes` da API para definir uma `ReplayPolicy` na assinatura da seguinte maneira:

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

Para verificar se uma mensagem foi reproduzida, o atributo booliano `Replayed` é adicionado a cada mensagem reproduzida.

Adicionar uma política de reprodução à assinatura com o SDK

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Compartilhados `config` e `credentials` arquivos no Guia de referência de ferramentas AWS SDKs e ferramentas.

O exemplo de código a seguir mostra como definir a `ReplayPolicy` em uma assinatura para reenviar mensagens do arquivo do tópico FIFO do Amazon SNS por um período de duas horas em 1.º de outubro de 2023.

```
// 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);
```

Entendendo o EndingPoint

Quando você aplica uma `ReplayPolicy` para uma assinatura do Amazon SNS, o valor `EndingPoint` é opcional. Se não `EndingPoint` for fornecido, a repetição começará do `StartingPoint`especificado e continuará até atingir a hora atual, incluindo o processamento de qualquer mensagem recém-publicada. Depois de concluída, a assinatura funcionará como uma assinatura regular, recebendo novas mensagens à medida que forem publicadas.

Se um `EndingPoint` for especificado, o serviço reproduzirá as mensagens do `StartingPoint` até o `EndingPoint` e depois parará. Essa ação efetivamente pausa a assinatura. Enquanto a

assinatura estiver pausada, as mensagens recém-publicadas não serão entregues ao endpoint inscrito.

Para retomar a entrega de mensagens, aplique uma nova `ReplayPolicy` sem fornecer um `EndingPoint` e defina o `StartingPoint` para o momento desejado a partir do qual continuar recebendo mensagens. Por exemplo, para retomar uma assinatura após o término de uma repetição anterior, defina o novo `StartingPoint` ao `EndingPoint` fornecido anteriormente.

Filtrar mensagens reproduzidas

A filtragem de mensagens do Amazon SNS permite que você controle as mensagens reproduzidas que o Amazon SNS reproduz no endpoint do assinante. Quando a filtragem e o arquivamento de mensagens estão habilitados, o Amazon SNS primeiro recupera a mensagem do datastore do tópico e, depois, aplica a mensagem à `FilterPolicy` da assinatura. A mensagem é enviada ao endpoint inscrito quando há uma correspondência. Caso contrário, ela é filtrada. Para obter mais informações, consulte Políticas de filtro de assinatura do Amazon SNS.

Monitore as métricas de reprodução de mensagens usando a Amazon CloudWatch

Você pode monitorar mensagens de repetição usando a Amazon CloudWatch usando as seguintes métricas. Para ser notificado sobre anomalias em suas cargas de trabalho e ajudar a evitar impactos, você pode configurar os CloudWatch alarmes da Amazon com base nessas métricas. Consulte mais detalhes em Registrar em log e monitorar no Amazon SNS.

Métrica	Descrição
NumberOfReplayedNotificationsDelivered	Fornece ao assinante o número agregado de mensagens reproduzidas do arquivo de tópicos, com resolução de 1 minuto.
NumberOfReplayedNotificationsFailed	Fornece ao assinante o número agregado de mensagens reproduzidas que não foram entregues do arquivo de tópicos, com resolução de 1 minuto.

Exemplos de código do Amazon SNS para tópicos FIFO

Use os exemplos de código a seguir para integrar o exemplo de caso de uso do gerenciamento de preços de autopeças com um tópico FIFO do Amazon SNS e uma fila FIFO do Amazon SQS ou uma fila padrão.

Usando um AWS SDK

Usando um AWS SDK, você cria um tópico FIFO do Amazon SNS `FifoTopic` definindo seu atributo como. **`true`** Você cria uma fila FIFO do Amazon SQS definindo seu atributo `FifoQueue` como **`true`**. Além disso, você deve adicionar o sufixo **`.fifo`** ao nome de cada recurso FIFO. Depois de criar um tópico ou fila FIFO, não é possível convertê-lo em um tópico ou fila padrão.

O seguinte exemplo de código cria esses recursos de fila FIFO e padrão:

- O tópico FIFO do Amazon SNS que distribui as atualizações de preços
- As filas FIFO do Amazon SQS que fornecem essas atualizações para as aplicações de atacado e varejo
- A fila padrão do Amazon SQS para a aplicação de análise que armazena registros, que podem ser consultados para fins de business intelligence (BI)
- As assinaturas FIFO do Amazon SNS que conectam as três filas ao tópico

Esse exemplo define políticas de filtro nas assinaturas. Se você testar o exemplo publicando uma mensagem no tópico, faça isso com o atributo `business`. Especifique `retail` ou `wholesale` como valor do atributo. Caso contrário, a mensagem será filtrada e não será entregue às filas inscritas. Para obter mais informações, consulte Filtragem de mensagens do Amazon SNS para tópicos FIFO do SNS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O teste verifica o recebimento da mensagem em cada fila. O exemplo completo também mostra a adição de políticas de acesso e exclui os recursos no final.

```
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);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTopic
  - Publicar
  - Assinar

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
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
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTopic
  - Publicar
  - Assinar

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

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

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

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

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

```
iv_stringvalue = 'High' ).
       INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
  - CreateTopic
  - Publicar
  - Assinar

Receber mensagens de assinaturas FIFO

Agora você pode receber atualizações de preços nas três aplicações inscritas. Conforme mostrado emthe section called “Caso de uso de tópico FIFO”, o ponto de entrada para cada aplicativo do consumidor é a fila do Amazon SQS, que sua AWS Lambda função correspondente pode pesquisar automaticamente. Quando uma fila do Amazon SQS é uma fonte de eventos para uma função do Lambda, o Lambda escala sua frota de sondas conforme necessário para consumir mensagens de forma eficiente.

Para obter mais informações, consulte Como usar AWS Lambda com o Amazon SQS no Guia do AWS Lambda desenvolvedor. Para obter informações sobre como criar seus próprios questionadores de filas, consulte Recomendações para filas padrão e FIFO do Amazon SQS no Amazon Simple

Queue Service Developer Guide e ReceiveMessagena Amazon Simple Queue Service API Reference.

Usando AWS CloudFormation

AWS CloudFormation permite que você use um arquivo de modelo para criar e configurar uma coleção de AWS recursos juntos como uma única unidade. Esta seção tem um modelo de exemplo que cria o seguinte:

- O tópico FIFO do Amazon SNS que distribui as atualizações de preços
- As filas FIFO do Amazon SQS que fornecem essas atualizações para as aplicações de atacado e varejo
- A fila padrão do Amazon SQS para a aplicação de análise que armazena registros, que podem ser consultados para fins de business intelligence (BI)
- As assinaturas FIFO do Amazon SNS que conectam as três filas ao tópico
- A política de filtros que especifica que as aplicações de assinante recebem somente as atualizações de preço de que precisam

> **Note**
>
> Se você testar esse exemplo de código publicando uma mensagem no tópico, faça isso com o atributo `business`. Especifique `retail` ou `wholesale` como valor do atributo. Caso contrário, a mensagem será filtrada e não será entregue às filas inscritas.

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

Para obter mais informações sobre como implantar AWS recursos usando um AWS CloudFormation modelo, consulte Introdução no Guia do AWS CloudFormation usuário.

Filtragem de mensagens do Amazon SNS

Por padrão, um assinante de tópico do Amazon SNS recebe todas as mensagens publicadas no tópico. Para receber apenas um subconjunto de mensagens, um assinante deve atribuir uma política de filtro à assinatura do tópico.

Uma política de filtro é um objeto JSON que contém propriedades que definem quais mensagens o assinante recebe. O Amazon SNS oferece suporte a políticas que atuam nos atributos da mensagem ou no corpo da mensagem, de acordo com o escopo da política de filtro que você definiu para a assinatura. As políticas de filtro para o corpo da mensagem pressupõem que a carga útil da mensagem seja um objeto JSON bem formado.

Se uma assinatura não tiver uma política de filtro, o assinante receberá todas as mensagens publicadas ao tópico. Quando você publica uma mensagem em um tópico com uma política de filtro em vigor, o Amazon SNS compara os atributos da mensagem ou o corpo da mensagem com as propriedades na política de filtro para cada assinatura do tópico. Se houver satisfação com as condições especificadas na política de filtro, o Amazon SNS enviará a mensagem para o assinante. Caso contrário, o Amazon SNS não enviará a mensagem para esse assinante.

Para obter mais informações, consulte Filtre mensagens publicadas em tópicos.

Escopo de política de filtro de assinaturas do Amazon SNS

O atributo de `FilterPolicyScope` assinatura permite que você defina o escopo da filtragem definindo um dos seguintes valores:

- `MessageAttributes`— Aplica a política de filtro aos atributos da mensagem (configuração padrão).
- `MessageBody`— Aplica a política de filtro ao corpo da mensagem.

> **Note**
>
> Se nenhum escopo de política de filtro for definido para uma política de filtro existente, o escopo assumirá `MessageAttributes` como padrão.

Políticas de filtro de assinatura do Amazon SNS

Uma política de filtro de assinatura permite especificar nomes de propriedades e atribuir uma lista de valores para cada nome de propriedade. Para obter mais informações, consulte Filtragem de mensagens do Amazon SNS.

Quando o Amazon SNS avalia os atributos ou o corpo da mensagem em relação à política de filtro de assinatura, ele ignora atributos que não estão especificados na política.

> **Important**
>
> AWS serviços como IAM e Amazon SNS usam um modelo de computação distribuída chamado consistência eventual. As adições ou alterações a uma política de filtro de assinatura podem levar até 15 minutos para entrarem em vigor.

Uma assinatura aceita uma mensagem nas seguintes condições:

- Quando o escopo da política de filtro está definido como `MessageAttributes`, cada nome de propriedade na política de filtro corresponde ao nome de um atributo da mensagem. Para cada nome de propriedade correspondente na política de filtro, pelo menos um valor de propriedade corresponde ao valor do atributo da mensagem.
- Quando o escopo da política de filtro está definido como `MessageBody`, cada nome de propriedade na política de filtro corresponde ao nome de uma propriedade do corpo da mensagem. Para cada nome de propriedade correspondente na política de filtro, pelo menos um valor de propriedade corresponde ao valor da propriedade do corpo da mensagem.

Atualmente, o Amazon SNS é compatível com os seguintes operadores de filtro:

- Lógica E
- Lógica OU
- Operador OU
- Correspondência de chaves
- Correspondência exata do valor numérico
- Correspondência tudo-exceto de valor numérico
- Correspondência de valores numéricos

- Correspondência de valores de string
- Correspondência tudo-exceto de valor de string
- Correspondência de string usando um prefixo com o operador tudo-exceto
- Correspondência de igual-ignorar-capitalização do valor da string
- Correspondência do endereço IP do valor da string
- Correspondência de prefixo de valores de string
- Correspondência de sufixo de valores de string

Exemplo de políticas de filtro do Amazon SNS

O exemplo a seguir mostra uma carga útil da mensagem entregue por um tópico do Amazon SNS que processa as transações de clientes.

O primeiro exemplo inclui o campo `MessageAttributes` com atributos que descrevem a transação:

- Interesses do cliente
- Nome do repositório
- Estado do evento
- Preço de compra em USD

Como essa mensagem inclui o campo `MessageAttributes`, qualquer assinatura de tópico que defina uma `FilterPolicy` pode aceitar ou rejeitar seletivamente a mensagem, desde que `FilterPolicyScope` esteja definido como `MessageAttributes` na assinatura. Para obter mais informações sobre a aplicação de atributos a uma mensagem, consulte Atributos de mensagem do Amazon SNS.

```
{
   "Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "message-body-with-transaction-details",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url",
   "MessageAttributes": {
```

```
      "customer_interests": {
         "Type": "String.Array",
         "Value": "[\"soccer\", \"rugby\", \"hockey\"]"
      },
      "store": {
         "Type": "String",
         "Value":"example_corp"
      },
      "event": {
         "Type": "String",
         "Value": "order_placed"
      },
      "price_usd": {
         "Type": "Number",
         "Value": "210.75"
      }
   }
}
```

O exemplo a seguir mostra os mesmos atributos incluídos no campo `Message`, também chamado de carga útil da mensagem ou corpo da mensagem. Qualquer assinatura de tópico que inclui uma `FilterPolicy` pode aceitar ou rejeitar seletivamente a mensagem, desde que `FilterPolicyScope` esteja definido como `MessageBody` na assinatura.

```
{
"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"
}
```

As políticas de filtro a seguir aceitam ou rejeitam mensagens com base em seus nomes e valores de propriedades.

Uma política que aceita a mensagem de exemplo

As propriedades na política de filtro de assinatura a seguir correspondem aos atributos da mensagem de exemplo. Observe que a mesma política de filtro funciona para um `FilterPolicyScope` definido como `MessageAttributes` ou `MessageBody`. Cada assinante escolhe seu escopo de filtragem de acordo com a composição das mensagens que recebe do tópico.

Se qualquer propriedade única nessa política não corresponder a um atributo da mensagem, a política rejeitará a mensagem.

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

Uma política que rejeita a mensagem de exemplo

A política de filtro de assinatura a seguir tem várias incompatibilidades entre seus atributos e as propriedades da mensagem de exemplo. Por exemplo, como o nome da propriedade `encrypted` não está presente nos atributos da mensagem, essa propriedade de política faz com que a mensagem seja rejeitada, independentemente do valor atribuído a ela.

Se ocorrer qualquer incompatibilidade, a política rejeitará a mensagem.

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

Restrições da política de filtro no Amazon SNS

Ao configurar políticas de filtro no Amazon SNS, há algumas regras importantes que você deve ter em mente. Essas regras ajudam a garantir a aplicação efetiva das políticas de filtro, mantendo o desempenho e a compatibilidade do sistema.

Restrições da política de filtro

Ao configurar políticas de filtro no Amazon SNS, siga estas regras importantes para garantir que elas funcionem de forma eficaz, mantendo o desempenho e a compatibilidade do sistema:

- Correspondência de strings — Para correspondência de strings na política de filtro, a comparação diferencia maiúsculas de minúsculas.
- Correspondência numérica — Para correspondência numérica, o valor pode variar de -10^9 a 10^9 (-1 bilhão a 1 bilhão), com cinco dígitos de precisão após o ponto decimal.
- Complexidade da política de filtro — A combinação total de valores em uma política de filtro não deve exceder 150. Para calcular a combinação total, multiplique o número de valores em cada matriz na política de filtro.
- Limitar o número de chaves — Uma política de filtro pode ter no máximo cinco chaves.

Considerações adicionais

- O JSON da política de filtro pode conter o seguinte:
  - Strings entre aspas
  - Números
  - As palavras-chave `true`, `false` e `null`, sem aspas
- Ao usar a API do Amazon SNS, você deve passar o JSON da política de filtro como uma string UTF-8 válida.
- O tamanho máximo de uma política de filtro é 256 KB.
- Por padrão, você pode ter até 200 políticas de filtro por tópico e 10.000 políticas de filtro por AWS conta.

Esse limite de política não impedirá que as assinaturas de filas do Amazon SQS sejam criadas com a API. `Subscribe` No entanto, falhará quando você anexar a política de filtro à chamada da API `Subscribe` (ou na chamada da API `SetSubscriptionAttributes`).

Para aumentar essa cota, use AWS Service Quotas.

Restrições de política para filtragem baseada em atributo

A filtragem baseada em atributo é a opção padrão. `FilterPolicyScope` é definido como `MessageAttributes` na assinatura.

- O Amazon SNS não aceita uma política de filtro aninhada para filtragem baseada em atributo.
- O Amazon SNS compara as propriedades de política somente com atributos de mensagens que tenham os seguintes tipos de dados:
  - `String`
  - `String.Array`

> **Important**
>
> Ao usar a filtragem baseada em atributos no Amazon SNS, você deve escapar duas vezes de certos caracteres especiais, especificamente:
>
> - Aspas duplas (“)
> - Barras invertidas ()
>
> A falha em escapar duas vezes desses caracteres fará com que a política de filtro não corresponda aos atributos de uma mensagem publicada e a notificação não será entregue.

Considerações adicionais

- Passar objetos em matrizes não é recomendado porque pode gerar resultados inesperados devido ao aninhamento, que não é suportado pela filtragem baseada em atributos. Usar filtragem baseada em carga para políticas aninhadas.
- `Number`é compatível com valores de atributos numéricos.
- O Amazon SNS ignora os atributos da mensagem com o tipo de dados binários.

Exemplo de política para complexidade:

No exemplo de política a seguir, a primeira chave tem três operadores de correspondência, a segunda tem um operador de correspondência e a terceira tem dois operadores de correspondência.

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

A combinação total é calculada como o produto do número de operadores de correspondência para cada chave na política de filtro:

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

Restrições de política para filtragem baseada em carga útil

Para mudar da filtragem baseada em atributo (padrão) para a filtragem baseada em carga útil, defina `FilterPolicyScope` como `MessageBody` na assinatura.

- O Amazon SNS aceita uma política de filtro aninhada para filtragem baseada em carga útil.
- Para uma política aninhada, somente as chaves foliares são contabilizadas no limite de cinco chaves.

Exemplo de política para limite de chaves:

No exemplo de política a seguir:

- Existem duas teclas de folha: `key_c` `key_e` e.
- `key_c`tem quatro operadores de correspondência com um nível aninhado de três e **`key_e`** três operadores de correspondência com um nível aninhado de dois.

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

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

A combinação total é calculada como o produto do número de operadores de correspondência e do nível aninhado de cada chave na política de filtro:

```
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
```

Lógica E/OU

Use a lógica AND/OR em políticas de filtro para corresponder aos atributos da mensagem ou às propriedades do corpo da mensagem no Amazon SNS. Isso permite uma filtragem de mensagens mais precisa e flexível.

Lógica E

Aplique a lógica E usando vários nomes de propriedade.

Considere a seguinte política:

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

Ela estabelece correspondência com qualquer propriedade de atributo ou corpo de mensagem com o valor de `customer_interests` definido como `rugby` e o valor de `price_usd` definido como um número maior que 100.

> **Note**
>
> Não é possível aplicar a lógica AND aos valores do mesmo atributo.

Lógica OU

Aplique a lógica OU atribuindo vários valores a um nome de propriedade.

Considere a seguinte política:

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

Ela estabelece correspondência com qualquer propriedade de atributo ou corpo de mensagem com o valor de `customer_interests` definido como `rugby`, `football` ou `baseball`.

Operador OU

Você pode usar o operador `"$or"` para definir explicitamente uma política de filtro para expressar a relação OR entre vários atributos na política.

O Amazon SNS só reconhece uma relação `"$or"` quando a política atende a todas as condições a seguir. Quando todas essas condições não são atendidas, `"$or"` é tratado como um nome de atributo regular, o mesmo que qualquer outra string na política.

- Há um atributo do campo `"$or"` na regra seguido por uma matriz, por exemplo `“$or” : []`.
- Há pelo menos dois objetos na matriz `"$or"`: `"$or": [{}, {}]`.
- Nenhum dos objetos na matriz `"$or"` tem nomes de campo que sejam palavras-chave reservadas.

Caso contrário, `"$or"` é tratado como um nome de atributo normal, o mesmo que outras strings na política.

A política a seguir não é analisada como uma relação OR porque numérico e prefixo são palavras-chave reservadas.

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

Exemplos de operador **`OR`**

0R padrão:

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

A lógica do filtro para essa política é:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

or

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

or

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

Restrições de políticas que incluem relações **`OR`**

Considere a seguinte política:

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

A lógica dessa política também pode ser simplificada como:

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

O cálculo da complexidade para políticas com relacionamentos OR pode ser simplificado como a soma das complexidades da combinação para cada declaração OR.

A combinação total é calculada da seguinte forma:

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

`source` tem um valor, `metricName` tem dois valores, `metricType` tem um valor, `metricId` tem dois valores e `spaceId` tem três valores.

Considere a seguinte política de filtro aninhada:

```
{
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      { "namespace": [ "AWS/EC2", "AWS/ES" ] }
    ],
```

```
    "detail" : {
      "scope" : [ "Service" ],
      "$or": [
        { "source": [ "aws.cloudwatch" ] },
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

A lógica dessa política pode ser simplificada como:

```
("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")
```

O cálculo das combinações totais é o mesmo para políticas não aninhadas, exceto que precisamos considerar o nível de aninhamento de uma chave.

A combinação total é calculada da seguinte forma:

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

`metricName` tem dois valores, `namespace` tem dois valores, `scope` é uma chave aninhada de dois níveis com um valor, `source` é uma chave aninhada de dois níveis com um valor e `type` é uma chave aninhada de dois níveis com um valor.

Correspondência de chaves

Use o `exists` operador em uma política de filtro para combinar as mensagens recebidas com base na presença ou ausência de uma propriedade específica.

- `exists`funciona somente nos nós da folha (atributos finais na estrutura).
- Ela não se aplica a nós intermediários dentro de uma estrutura JSON aninhada.

- Use `"exists": true` para estabelecer correspondência com mensagens recebidas que incluam a propriedade especificada. A chave deve ter um valor não nulo e não vazio.

Por exemplo, a propriedade de política seguinte usa o operador `exists` com um valor de `true`:

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

Ela estabelece correspondência com qualquer atributo de mensagem que contenha a chave de atributo `store`, como a seguinte:

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

No entanto, não estabelece correspondência com nenhum atributo de mensagem sem a chave de atributo `store`, como o seguinte:

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

Nem corresponde ao seguinte corpo de mensagem:

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

- Use `"exists": false` para estabelecer correspondência com mensagens recebidas que não incluam a propriedade especificada.

> **Note**
>
> `"exists": false` só estabelecerá correspondência se pelo menos um atributo estiver presente. Um conjunto vazio de atributos impossibilita que o filtro estabeleça correspondência.

Por exemplo, a propriedade de política seguinte usa o operador `exists` com um valor de `false`:

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

Ela não estabelece correspondência com nenhum atributo de mensagem que contenha a chave de atributo `store`, como o seguinte:

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

Também não estabelece correspondência com o seguinte corpo de mensagem:

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

No entanto, estabelece correspondência com qualquer atributo de mensagem sem a chave de atributo `store`, como o seguinte:

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

Também estabelece correspondência com o seguinte corpo de mensagem:

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

Correspondência de valores numéricos

Filtre mensagens combinando valores numéricos com os valores dos atributos da mensagem ou com os valores das propriedades do corpo da mensagem. Os valores numéricos não são inseridos entre aspas duplas na política JSON. Você pode usar as operações numéricas a seguir para filtragem.

> **Note**
>
> Os prefixos são compatíveis somente com correspondência de string.

Correspondência exata

Quando um valor de propriedade de política inclui a palavra-chave `numeric` e o operador `=`, estabelece correspondência com qualquer valor de propriedade de atributo ou corpo de mensagem que tenha o mesmo nome e o mesmo valor numérico.

Considere a seguinte propriedade de política:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

Correspondência anything-but

Quando o valor de uma propriedade de política inclui a palavra-chave `anything-but`, estabelece correspondência com qualquer valor de propriedade de atributo ou corpo de mensagem que não inclua nenhum dos valores de propriedade de política.

Considere a seguinte propriedade de política:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

Além disso, estabelece correspondência com o seguinte atributo de mensagem (pois contém um valor que não é `100` nem `500`):

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

Também estabelece correspondência com o seguinte corpo de mensagem (pois contém um valor que não é `100` nem `500`):

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

No entanto, não corresponde ao seguinte atributo de mensagem:

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

Nem corresponde ao seguinte corpo de mensagem:

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

Correspondência de intervalo de valores

Além do operador =, uma propriedade de política numérica pode incluir os seguintes operadores: <, <=, > e >=.

Considere a seguinte propriedade de política:

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

Estabelece correspondência com qualquer propriedade de atributo ou corpo de mensagem com valores numéricos negativos.

Considere outro atributo de mensagem:

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

Estabelece correspondência com qualquer propriedade de atributo ou corpo de mensagem com números positivos até e incluindo 150.

Correspondência de valores de string

Filtre mensagens combinando valores de string com valores de atributos da mensagem ou valores de propriedades do corpo da mensagem. Os valores de string são inseridos entre aspas duplas na política JSON. Você pode usar as seguintes operações de string para corresponder aos atributos da mensagem ou às propriedades do corpo da mensagem:

Correspondência exata

A correspondência exata ocorre quando um valor de propriedade de política corresponde a um ou mais valores de atributos de mensagens. Para atributos `String.Array` de tipo, cada elemento na matriz é tratado como uma string separada para fins de correspondência.

Considere a seguinte propriedade de política:

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

Ele corresponde aos seguintes atributos de mensagem:

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

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

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

Também corresponde aos seguintes corpos de mensagem:

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

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

No entanto, ele não corresponde aos seguintes atributos de mensagem:

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

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

Nem corresponde ao seguinte corpo de mensagem:

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

Correspondência anything-but

Quando o valor de uma propriedade de política inclui a palavra-chave `anything-but`, corresponde a qualquer valor de atributo ou corpo de mensagem que não inclua nenhum dos valores de

propriedade de política. `anything-but` pode ser combinado com `"exists": false`. Para atributos `String.Array` de tipo, corresponde se nenhum dos elementos da matriz estiver listado na propriedade da política.

Considere a seguinte propriedade de política:

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

Ele corresponde a qualquer um dos seguintes atributos de mensagem:

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

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

Além disso, estabelece correspondência com o seguinte atributo de mensagem (pois contém um valor que não é `rugby` nem `tennis`):

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

Também estabelece correspondência com o seguinte corpo de mensagem (pois contém um valor que não é `rugby` nem `tennis`):

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

No entanto, ele não corresponde aos seguintes atributos de mensagem:

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

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

Nem corresponde ao seguinte corpo de mensagem:

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

Usar um prefixo com o operador **anything-but**

Para correspondência de string, também é possível usar um prefixo com o operador `anything-but`. Por exemplo, a política de propriedade seguinte nega o prefixo `order-`:

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

Ele corresponde a um dos seguintes atributos:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

No entanto, não corresponde ao seguinte atributo de mensagem:

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

Nem corresponde ao seguinte corpo de mensagem:

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

Equals-ignore-case combinando

Quando a propriedade de uma política inclui a palavra-chave `equals-ignore-case`, ela realizará uma correspondência que não diferencia letras maiúsculas de minúsculas a qualquer valor de propriedade do corpo ou atributo da mensagem.

Considere a seguinte propriedade de política:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

Correspondência de endereço IP

Você pode usar o operador `cidr` para verificar se uma mensagem de entrada é originada de um endereço IP ou sub-rede específico.

Considere a seguinte propriedade de política:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

No entanto, não corresponde ao seguinte atributo de mensagem:

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

Nem corresponde ao seguinte corpo de mensagem:

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

Correspondência de prefixo

Quando uma propriedade de política inclui a palavra-chave `prefix`, ela corresponde a qualquer valor de atributo de mensagem ou corpo de mensagem que comece com os caracteres especificados.

Considere a seguinte propriedade de política:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

No entanto, não corresponde ao seguinte atributo de mensagem:

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

Nem corresponde ao seguinte corpo de mensagem:

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

Correspondência de sufixo

Quando a propriedade de uma política inclui a palavra-chave `suffix`, ela corresponde a qualquer valor de propriedade do corpo ou atributo da mensagem que termine com os caracteres especificados.

Considere a seguinte propriedade de política:

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

Ele corresponde a um dos seguintes atributos de mensagens:

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

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

Também estabelece correspondência com um dos seguintes corpos de mensagem:

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

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

No entanto, não corresponde ao seguinte atributo de mensagem:

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

Nem corresponde ao seguinte corpo de mensagem:

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

Aplicar uma política de filtro de assinatura no Amazon SNS

A filtragem de mensagens no Amazon SNS permite que você entregue mensagens seletivamente aos assinantes com base nas políticas de filtro. Essas políticas definem as condições que as mensagens devem atender para serem entregues a uma assinatura. Embora a entrega de mensagens brutas seja uma opção que possa afetar o processamento de mensagens, ela não é necessária para que os filtros de assinatura funcionem.

É possível aplicar uma política de filtro a uma assinatura do Amazon SNS usando o console do Amazon SNS. Ou, para aplicar políticas de forma programática, você pode usar a API do Amazon SNS, AWS Command Line Interface o AWS CLI() ou AWS qualquer SDK compatível com o Amazon SNS. Você também pode usar AWS CloudFormation.

Habilitação da entrega de mensagens brutas

A entrega de mensagens brutas garante que as cargas úteis de mensagens sejam entregues como estão aos assinantes, sem qualquer codificação ou transformação adicional. Isso pode ser útil quando os assinantes precisam do formato original da mensagem para processamento. No entanto, a entrega de mensagens brutas não está diretamente relacionada à funcionalidade dos filtros de assinatura.

Aplicação de filtros de assinatura

Para aplicar filtros de mensagens a uma assinatura, você define uma política de filtro usando a sintaxe JSON. Essa política especifica as condições que uma mensagem deve atender para ser entregue à assinatura. Os filtros podem ser baseados em atributos da mensagem, como atributos da mensagem, estrutura da mensagem ou até mesmo conteúdo da mensagem.

Relação entre entrega de mensagens brutas e filtros de assinatura

Embora a ativação da entrega de mensagens brutas possa afetar a forma como as mensagens são entregues e processadas pelos assinantes, não é um pré-requisito para o uso de filtros de assinatura. No entanto, em cenários em que os assinantes exigem o formato original da mensagem sem nenhuma modificação, habilitar a entrega de mensagens brutas pode ser benéfico junto com os filtros de assinatura.

Considerações para uma filtragem eficaz

Ao implementar a filtragem de mensagens, considere os requisitos específicos do seu aplicativo e dos assinantes. Defina políticas de filtro que correspondam com precisão aos critérios de entrega de mensagens para garantir uma distribuição eficiente e direcionada de mensagens.

> **Important**
>
> AWS serviços como IAM e Amazon SNS usam um modelo de computação distribuída chamado consistência eventual. As adições ou alterações a uma política de filtro de assinatura podem levar até 15 minutos para entrarem em vigor.

AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Subscriptions (Assinaturas).
3. Selecione uma assinatura e, em seguida, escolha Edit (Editar).
4. Na página Editar expanda a seção Política de filtros de assinatura.
5. Escolha entre filtragem baseada em atributo ou filtragem baseada em carga útil.
6. No campo Editor JSON, informe o Corpo do JSON de sua política de filtro.
7. Escolha Save changes (Salvar alterações).

   O Amazon SNS aplica a política de filtro à assinatura.

AWS CLI

Para aplicar uma política de filtro com o AWS Command Line Interface (AWS CLI), use o `set-subscription-attributes`comando, conforme mostrado no exemplo a seguir. Para a opção `--attribute-name`, especifique `FilterPolicy`. Em `--attribute-value`, especifique sua política JSON.

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

Para fornecer JSON válido para sua política, coloque os nomes e valores dos atributos entre aspas duplas. Também é necessário colocar o argumento completo da política entre aspas. Para evitar aspas de escape, você pode usar aspas simples para colocar a política e aspas duplas para os nomes e valores de JSON, como mostrado no exemplo.

Se você quiser alternar da filtragem de mensagens baseada em atributos (padrão) para a baseada em carga, você também pode usar o comando. set-subscription-attributes Para a opção `--attribute-name`, especifique `FilterPolicyScope`. Em `--attribute-value`, especifique `MessageBody`.

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

Para verificar se a política de filtro foi aplicada, use o comando `get-subscription-attributes`. Os atributos na saída do terminal devem mostrar a política de filtro para a chave `FilterPolicy`, conforme mostrado no exemplo a seguir:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
    "Attributes": {
        "Endpoint": "endpoint . . .",
        "Protocol": "https",
        "RawMessageDelivery": "false",
        "EffectiveDeliveryPolicy": "delivery policy . . .",
        "ConfirmationWasAuthenticated": "true",
        "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed
\"]}",
        "FilterPolicyScope": "MessageAttributes",
        "Owner": "111122223333",
```

```
        "SubscriptionArn": "arn:aws:sns: . . .",
        "TopicArn": "arn:aws:sns: . . ."
    }
}
```

AWS SDKs

Os exemplos de código a seguir mostram como usar o `SetSubscriptionAttributes`.

> **Important**
>
> Se você estiver usando o exemplo do SDK para Java 2.x, a classe `SNSMessageFilterPolicy` não estará disponível por padrão. Para obter instruções sobre como instalar essa classe, consulte o exemplo do GitHub site.

CLI

AWS CLI

Para definir atributos de assinatura

O exemplo `set-subscription-attributes` a seguir define o atributo `RawMessageDelivery` para uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Este comando não produz saída.

O exemplo `set-subscription-attributes` a seguir define um atributo `FilterPolicy` para uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Este comando não produz saída.

O exemplo `set-subscription-attributes` a seguir remove o atributo `FilterPolicy` de uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetSubscriptionAttributesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

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

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

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);
```

```
            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

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

- Para obter detalhes da API, consulte SetSubscriptionAttributesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


    @staticmethod
```

```
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Para obter detalhes da API, consulte a SetSubscriptionAttributesReferência da API AWS SDK for Python (Boto3).

API do Amazon SNS

Para aplicar um política de filtro com a API do Amazon SNS, faça uma solicitação para a ação `SetSubscriptionAttributes`. Defina o parâmetro `AttributeName` como `FilterPolicy` e defina o parâmetro `AttributeValue` para a política de filtro JSON.

Se você quiser mudar da filtragem baseada em atributos (padrão) para a filtragem baseada em carga útil, também poderá usar a ação `SetSubscriptionAttributes`. Defina o parâmetro `AttributeName` como `FilterPolicyScope` e defina o parâmetro `AttributeValue` como `MessageBody`.

AWS CloudFormation

Para aplicar uma política de filtro usando AWS CloudFormation, use um modelo JSON ou YAML para criar uma AWS CloudFormation pilha. Para obter mais informações, consulte a `FilterPolicy`propriedade do `AWS::SNS::Subscription` recurso no Guia do AWS CloudFormation usuário e no AWS CloudFormation modelo de exemplo.

1. Faça login no console do AWS CloudFormation.
2. Escolha Create Stack (Criar pilha).
3. Na página Select Template (Selecionar modelo), escolha Upload a template to Amazon S3 (Carregar um modelo no Amazon S3) e, em seguida, o arquivo e Next (Avançar).
4. Na página Especificar detalhes, faça o seguinte:
   a. Em Nome da pilha, digite `MyFilterPolicyStack`.
   b. Para myHttpEndpoint, digite o endpoint HTTP a ser inscrito em seu tópico.

   > **Tip**
   >
   > Se você não tiver um endpoint HTTP, crie um.

5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Review (Revisar), escolha Create (Criar).

Remover uma política de filtro de assinatura no Amazon SNS

Para interromper a filtragem de mensagens que são enviadas para uma assinatura, remova a política de filtro da assinatura substituindo-a por um corpo JSON vazio. Após a remoção da política, a assinatura aceitará todas as mensagens que forem publicadas nela.

Usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Subscriptions (Assinaturas).
3. Selecione uma assinatura e, em seguida, escolha Edit (Editar).
4. Na *`EXAMPLE1-23bc-4567-d890-ef12g3hij456`* página Editar, expanda a seção Política de filtro de assinatura.

5. No campo Editor JSON, forneça um corpo do JSON vazio para seu filtro de política: `{}`.
6. Escolha Salvar alterações.

   O Amazon SNS aplica a política de filtro à assinatura.

Usando o AWS CLI

Para remover uma política de filtro com o AWS CLI, use o `set-subscription-attributes`comando e forneça um corpo JSON vazio para o `--attribute-value` argumento:

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

Uso da API do Amazon SNS

Para remover uma política de filtro com a API do Amazon SNS, faça uma solicitação para a ação `SetSubscriptionAttributes`. Defina o parâmetro `AttributeName` como `FilterPolicy` e forneça um corpo JSON vazio para o parâmetro `AttributeValue`.

Proteção de dados de mensagens no Amazon SNS

O que é proteção de dados de mensagens?

A proteção de dados de mensagens protege os dados publicados em seus tópicos do Amazon SNS usando políticas de proteção de dados para auditar, mascarar, redigir ou bloquear as informações confidenciais que se movem entre aplicativos ou serviços. AWS

A proteção de dados de mensagens verifica dados em movimento em busca de informações de identificação pessoal (PII) e informações de saúde protegidas (PHI) usando identificadores de dados. É possível optar por usar identificadores de dados predefinidos (ou gerenciados pelo Amazon SNS) (por exemplo, nomes, endereços, números de cartão de crédito e códigos de medicamentos prescritos) ou criar seus próprios identificadores de dados personalizados especificamente para o caso de uso de negócios. Usando as informações verificadas, a proteção de dados de mensagens fornece logs de auditoria detalhados e permite que você tome medidas para proteger esses dados.

A proteção de dados de mensagens comporta as seguintes ações para ajudar a proteger as informações sigilosas do cliente:

- Audit (Auditoria): audite até 99% dos dados publicados em um tópico do Amazon SNS. Em seguida, você pode optar por enviar as descobertas para a Amazon CloudWatch, Amazon S3 ou Amazon Data Firehose.
- Desidentificar: mascara ou edita dados confidenciais sem interromper a publicação ou a entrega de mensagens.
- Negar — bloqueie a transmissão de dados entre aplicativos e AWS recursos se houver dados confidenciais na carga útil.

> **Note**
>
> O Amazon SNS é compatível com a proteção de dados de mensagens somente para tópicos padrão do Amazon SNS.

Por que devo usar a proteção de dados de mensagens?

Ao introduzir a proteção de dados de mensagens em seus programas de governança, gerenciamento de riscos e conformidade, você pode implementar políticas de proteção de dados que ajudam a identificar e evitar o vazamento de dados. Isso oferece às suas equipes ferramentas que podem ajudar a reduzir riscos financeiros, jurídicos e regulatórios por meio do cumprimento de regulamentos de privacidade como HIPAA, RGPD, PCI e FedRAMP. Também libera seus desenvolvedores da sobrecarga operacional associada à criação e ao gerenciamento de suas próprias ferramentas para proteger dados sigilosos.

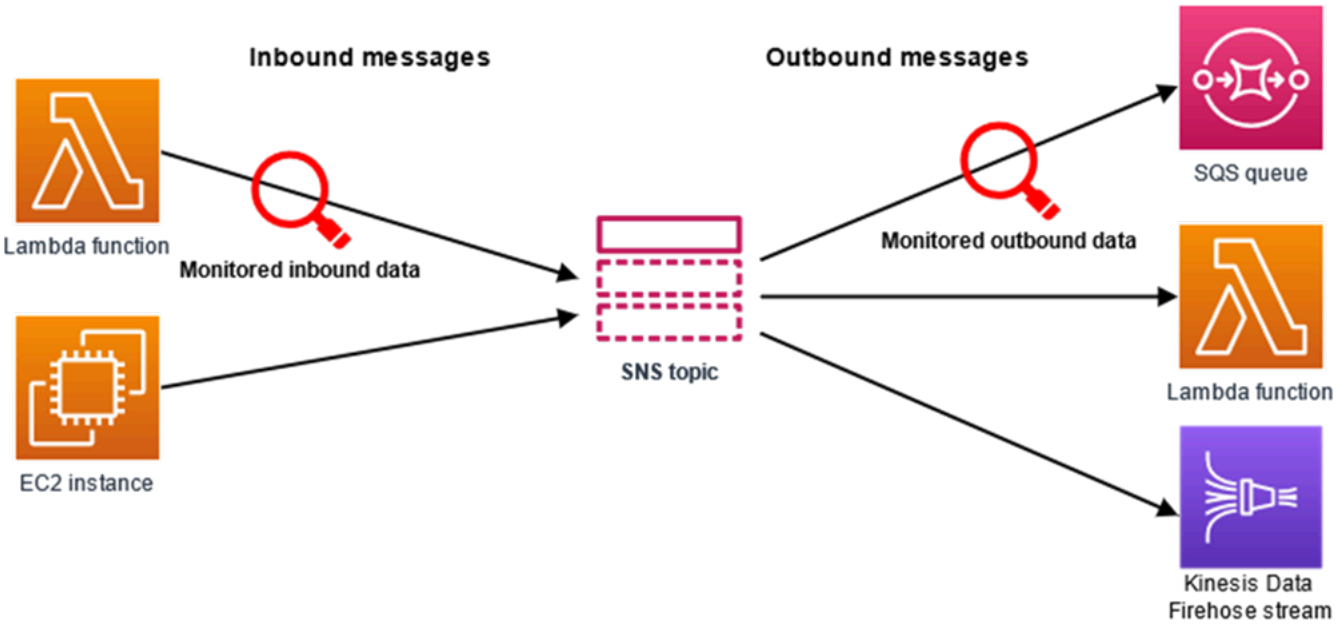
Por exemplo, você pode usar a proteção de dados de mensagens para criar uma política de auditoria destinada a determinar se algum de seus sistemas está acidentalmente enviando ou recebendo dados sigilosos. Se os resultados da auditoria mostrarem que os sistemas estão enviando informações de cartão de crédito a sistemas que não precisam delas, você pode usar uma política de bloqueio para impedir a entrega dos dados.

> **Note**
>
> O Amazon SNS é compatível com a proteção de dados de mensagens somente para tópicos padrão do Amazon SNS.

Compreender as políticas de proteção de dados do Amazon SNS

O que são políticas de proteção de dados?

O Amazon SNS usa políticas de proteção de dados para selecionar os dados sigilosos que você deseja verificar e as ações que quer realizar para evitar que esses dados sejam trocados por seus tópicos do Amazon SNS. Para selecionar os dados sigilosos de interesse, você deve usar identificadores de dados. Depois, a proteção de dados de mensagens do Amazon SNS detecta os dados sigilosos usando machine learning e correspondência de padrões. Para agir de acordo com os identificadores de dados encontrados, você pode definir uma operação de auditoria, desidentificação ou negação. Essas operações permitem que você registre os dados confidenciais encontrados (ou não encontrados), mascare ou elimine os dados confidenciais ou negue a entrega de mensagens.

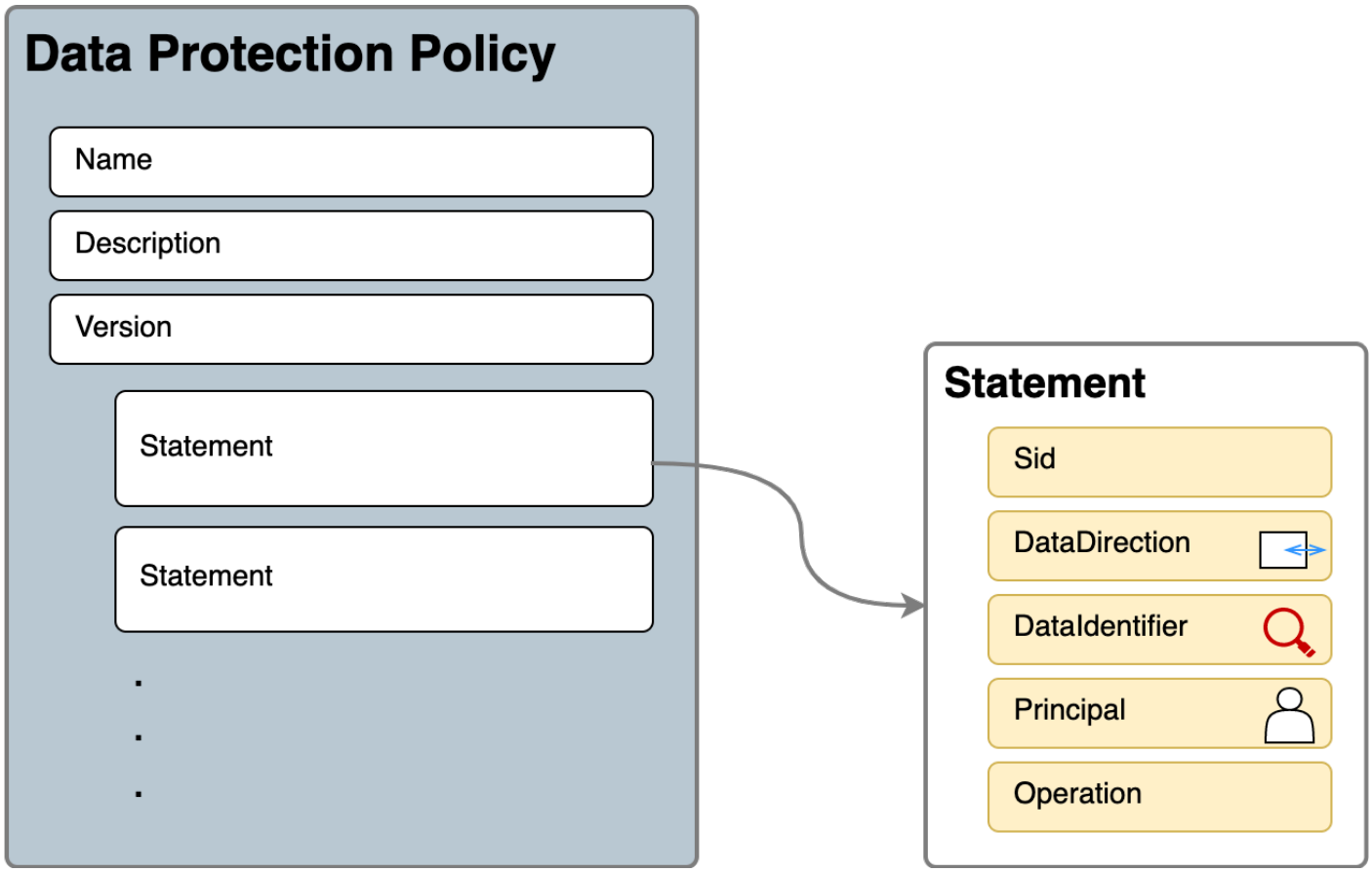


Como a política de proteção de dados é estruturada?

Como mostrado na figura abaixo, um documento de política de proteção de dados inclui os seguintes elementos:

- Informações opcionais da política na parte superior do documento
- Uma ou mais instruções individuais

Cada instrução inclui informações sobre uma única permissão.



Só é possível definir uma política de proteção de dados por tópico do Amazon SNS. A política de proteção de dados pode ter uma ou mais instruções de negação ou desidentificação, mas somente uma instrução de auditoria.

Propriedades JSON para a política de proteção de dados

Uma política de proteção de dados requer as seguintes informações básicas de política para identificação:

- Nome: o nome da política.
- Descrição (Opcional): a descrição da política.
- Versão: a versão do idioma das políticas. A versão atual é 2021-06-01.
- Declaração: uma lista de declarações que especificam as ações da política de proteção de dados.

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

```
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

Propriedades JSON para uma declaração de política

Uma declaração de política define o contexto de detecção para a operação de proteção de dados.

- Sid (Opcional): o identificador da declaração.
- DataDirection— Entrada (para solicitações de API de publicação) ou saída (para entrega de notificações) com relação ao tópico do Amazon SNS.
- DataIdentifier— Os dados confidenciais que o tópico do Amazon SNS deve verificar. Por exemplo, nome, endereço ou número de telefone.
- Entidade principal: a entidade principal do IAM que publicou no tópico ou a entidade principal do IAM que assinou o tópico.
- Operação: a ação subsequente, seja Audit (Auditor), De-identify (Desidentificar) (mascarar ou eliminar) ou Deny (Negar), que o tópico do Amazon SNS realiza quando encontra dados confidenciais.

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

Propriedades JSON para uma operação de declaração de política

Uma declaração de política define uma das operações de proteção de dados a seguir.

- Audit (Auditoria): emite métricas e localiza logs sem interromper a publicação ou a entrega de mensagens.
- Desidentificar: mascara ou elimina dados confidenciais sem interromper a publicação de mensagens.
- Deny (Negação): bloqueia a solicitação de publicação do Amazon SNS ou não entrega a mensagem.

Como determino as entidades principais do IAM para minha política de proteção de dados?

A proteção de dados de mensagens usa duas entidades principais do IAM que interagem com o Amazon SNS.

1. Entidade principal da API de publicação (entrada): a entidade principal autenticada do IAM que chama a API `Publish` do Amazon SNS.
2. Entidades principais de assinatura (saída): a entidade principal autenticada do IAM que chamou a API `Subscribe` durante a criação da assinatura.

A `SubscriptionPrincipal` é uma propriedade de assinatura do Amazon SNS disponível ao público que pode ser recuperada na API `GetSubscriptionAttributes`.

```
{
  "Attributes": {
    "SubscriptionPrincipal": "arn:aws:iam::123456789012:user/NoNameAccess",
    "Owner": "123412341234",
    "RawMessageDelivery": "true",
    "TopicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
    "Endpoint": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
    "Protocol": "sqs",
    "PendingConfirmation": "false",
    "ConfirmationWasAuthenticated": "true",
    "SubscriptionArn": "arn:aws:sns:us-east-1:123412341234:PII-data-
topic:5d8634ef-67ef-49eb-a824-4042b28d6f55"
  }
}
```

Operações de política de proteção de dados no Amazon SNS

Veja a seguir as políticas de proteção de dados que você pode usar para auditar e negar dados sigilosos. Para ver um tutorial completo que inclui uma aplicação de exemplo, consulte a publicação do blog Introducing message data protection for Amazon SNS (“Apresentar a proteção de dados de mensagens para o Amazon SNS”).

Operação de auditoria

A operação de auditoria coleta amostras de mensagens de entrada de tópicos e registra as descobertas de dados confidenciais em um AWS destino. A taxa de amostragem pode ser um número inteiro entre 0 e 99. Essa operação requer um dos seguintes tipos de destino de registro em log:

1. FindingsDestination— O destino do registro quando o tópico do Amazon SNS encontra dados confidenciais na carga útil.
2. NoFindingsDestination— O destino do registro quando o tópico do Amazon SNS não encontra dados confidenciais na carga útil.

Você pode usar o seguinte Serviços da AWS em cada um dos seguintes tipos de destino de log:

- Amazon CloudWatch Logs (opcional) — Eles `LogGroup` devem estar na região do tópico e o nome deve começar com /aws/vendedlogs/.
- Amazon Data Firehose (opcional): o `DeliveryStream` deve estar na região do tópico e ter Direct PUT como fonte de fluxo de entrega. Para obter detalhes adicionais, consulte Source, Destination, and Name (“Origem, destino e nome”) no Guia do desenvolvedor do Amazon Kinesis Data Firehose.
- Amazon S3 (Opcional): um nome de bucket do Amazon S3. Ações extras são necessárias para usar o bucket do Amazon S3 com a criptografia SSE-KMS ativada.

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

Permissões necessárias ao especificar destinos de log

Ao especificar destinos de registro em log na política de proteção de dados, você deve adicionar as seguintes permissões à política de identidade do IAM da entidade principal do IAM que está chamando o Amazon SNS: API `PutDataProtectionPolicy` ou `CreateTopic` com o parâmetro `--data-protection-policy`.

Destino de auditoria	Permissão do IAM
Padrão	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy

Destino de auditoria	Permissão do IAM
	logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Ações extras são necessárias para usar o bucket do Amazon S3 com a criptografia SSE-KMS ativada.

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

Política de chaves obrigatórias para uso com SSE-KMS

Se você usar um bucket do Amazon S3 como destino de log, poderá proteger os dados em seu bucket habilitando a criptografia do lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3) ou a criptografia do lado do servidor com (SSE-KMS). AWS KMS keys Para obter mais informações, consulte Proteger dados usando criptografia do lado do servidor no Manual do usuário do Amazon S3.

Se você escolher SSE-S3, nenhuma configuração adicional será necessária. O Amazon S3 lida com a chave de criptografia.

Se você escolher o SSE-KMS, deverá usar uma chave gerenciada pelo cliente. Você deve atualizar a política de chaves para a chave gerenciada pelo cliente para que a conta de entrega de logs possa gravar no bucket do S3. Para obter mais informações sobre a política de chaves necessária para uso com o SSE-KMS, consulte a criptografia do lado do servidor do bucket Amazon S3 no Guia do usuário do Amazon Logs. CloudWatch

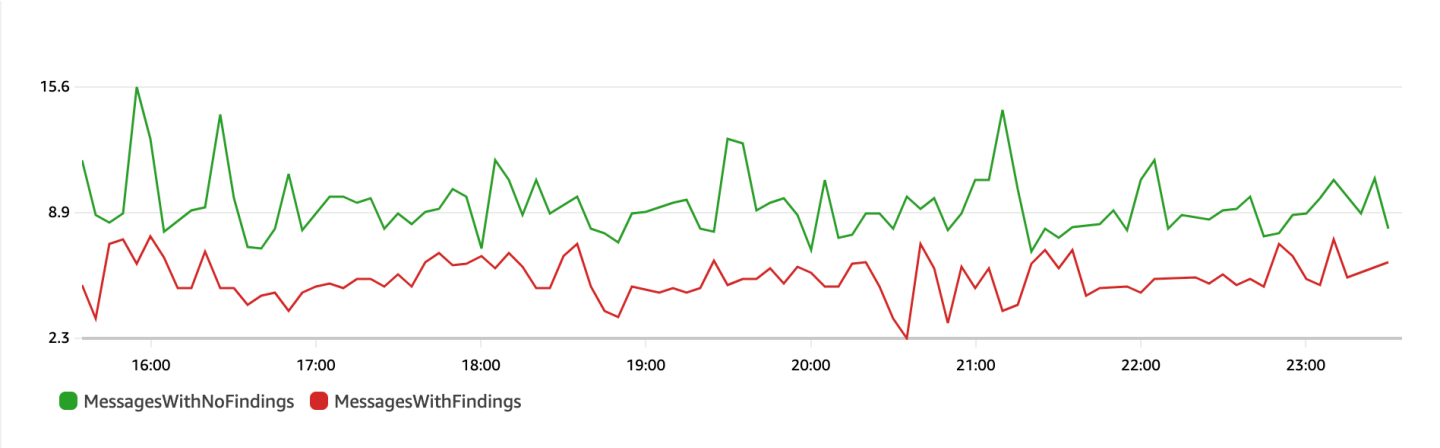
Exemplo de log de destino de auditoria

No exemplo a seguir, `callerPrincipal` é usado para identificar a origem do conteúdo sigiloso e `messageID` como referência para comparar com a resposta da API `Publish`.

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

Métricas de operação de auditoria

Quando uma operação de auditoria especifica a `FindingsDestination` ou a `NoFindingsDestination` propriedade, os proprietários do tópico também recebem CloudWatch `MessagesWithFindings` `MessagesWithNoFindings` métricas.



Operação de desidentificação

A operação Remoção de identificação mascara ou elimina dados sigilosos das mensagens publicadas ou entregues. Essa operação está disponível para mensagens de entrada e saída e requer um dos seguintes tipos de configuração:

- MaskConfig— Máscara usando um caractere compatível da tabela a seguir. Por exemplo, ssn: `123-45-6789` se torna ssn: `###########`.

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

Caractere de máscara compatível	Name
*	Asterisco
A–Z, a–z e 0–9	Alfanumérico
	Espaço
!	Ponto de exclamação

Caractere de máscara compatível	Name
$	Sinal de dólar
%	Sinal de porcentagem
&	E comercial
()	Parênteses
+	Sinal de adição
,	Vírgula
-	Hífen
.	Período
/\	Barra, barra invertida
#	Sinal numérico
:	Dois pontos
;	Ponto e vírgula
=, <>	Igual, menor que ou maior que
@	Arroba
[]	Colchetes
^	Circunflexo
_	Sublinhado
`	Acento grave
|	Barra vertical
~	Til

- RedactConfig— Redija removendo totalmente os dados. Por exemplo, ssn: `123-45-6789` se torna ssn: .

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

Em uma mensagem de entrada, os dados confidenciais são desidentificados após a operação de auditoria, e o chamador da API `SNS:Publish` recebe o erro de parâmetro inválido a seguir quando toda a mensagem é confidencial.

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

Operação de negação

A operação Deny (Negação) interromperá a solicitação da API `Publish` ou a entrega da mensagem se a mensagem contiver dados sigilosos. O objeto da operação Deny (Negação) está em branco, pois não requer configuração adicional.

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

Em uma mensagem de entrada, o chamador da API `SNS:Publish` recebe um erro de autorização.

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

Em uma mensagem de saída, o tópico do Amazon SNS não entrega a mensagem para a assinatura. Para monitorar entregas não autorizadas, habilite o registro em log do status de entrega do tópico. Veja a seguir um exemplo de log de status de entrega:

```
{
    "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"
}
```

Exemplos de políticas de proteção de dados do Amazon SNS

Veja a seguir as políticas de proteção de dados que você pode usar para auditar e negar dados sigilosos. Para ver um tutorial completo que inclui uma aplicação de exemplo, consulte a publicação do blog Introducing message data protection for Amazon SNS (“Apresentar a proteção de dados de mensagens para o Amazon SNS”).

Exemplo de política para auditoria

As políticas de auditoria permitem auditar até 99% das mensagens recebidas e enviar descobertas para a Amazon CloudWatch, Amazon Data Firehose e Amazon S3.

Por exemplo, você pode criar uma política de auditoria para avaliar se algum de seus sistemas está acidentalmente enviando ou recebendo dados sigilosos. Se os resultados da auditoria mostrarem que os sistemas estão enviando informações de cartão de crédito a sistemas que não precisam delas, você pode implementar uma política de proteção de dados para bloquear a entrega dos dados.

O exemplo a seguir audita 99% das mensagens que fluem pelo tópico procurando números de cartão de crédito e enviando as descobertas para CloudWatch Logs, Firehose e Amazon S3.

Política de proteção de dados:

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

Exemplo de formato de resultados de auditoria:

```
{
    "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"
}
```

Exemplo de política com instrução de máscara de desidentificação de entrada

O exemplo a seguir impede que um usuário publique uma mensagem em um tópico com `CreditCardNumber` ao mascarar os dados confidenciais do conteúdo da mensagem.

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

Exemplo de resultados da máscara de desidentificação de entrada:

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

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

Exemplo de política com instrução de eliminação da desidentificação de entrada

O exemplo a seguir impede que um usuário publique uma mensagem em um tópico com `CreditCardNumber` ao eliminar os dados confidenciais do conteúdo da mensagem.

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

Exemplo de resultados de eliminação de desidentificação de entrada:

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

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

Exemplo de política com instrução de máscara da desidentificação de saída

O exemplo a seguir impede que um usuário receba uma mensagem com `CreditCardNumber` ao mascarar os dados sigilosos do respectivo conteúdo.

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

Exemplo de resultados da máscara de desidentificação de saída:

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

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

Exemplo de política com instrução de edição da desidentificação de saída

O exemplo a seguir impede que um usuário receba uma mensagem com `CreditCardNumber` ao eliminar os dados sigilosos do conteúdo da mensagem.

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

Exemplo de resultados de eliminação de desidentificação de saída:

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

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

Exemplo de política com instrução de negação de entrada

O exemplo a seguir impede que um usuário publique uma mensagem em um tópico com `CreditCardNumber` no conteúdo da mensagem. As cargas negadas na resposta da API têm um código de status de”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": {}
      }
    }
  ]
```

```
}
```

Exemplo de política com instrução de negação de saída

O exemplo a seguir impede que uma AWS conta receba mensagens que contenham`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": {}
      }
    }
  ]
}
```

Exemplo de resultados de negação de saída, registrados na 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"
}
```

Criar políticas de proteção de dados no Amazon SNS

As políticas de proteção de dados ajudam você a proteger os dados publicados em seus tópicos do Amazon SNS auditando, desidentificando (mascarando ou eliminando) e negando (bloqueando) informações confidenciais passadas entre aplicações ou Serviços da AWS. Você pode usar a AWS API, AWS CLI, AWS CloudFormation, ou AWS Management Console para criar políticas de proteção de dados no Amazon SNS. Somente uma política pode ser definida por tópico do Amazon SNS. Cada política de proteção de dados pode ter uma ou mais instruções de desidentificação e negação, mas somente uma instrução de auditoria.

Tópicos

- Criar políticas de proteção de dados no Amazon SNS use o API
- Criar políticas de proteção de dados no Amazon SNS use o CLI
- Criação de políticas de proteção de dados no Amazon SNS usando CloudFormation
- Criar políticas de proteção de dados no Amazon SNS use o console
- Criar políticas de proteção de dados do Amazon SNS para proteger os dados das mensagens usando o SDK

Criar políticas de proteção de dados no Amazon SNS use o API

O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Amazon Simple Notification Service endpoints and quotas (Endpoints e cotas do Amazon Simple Notification Service).

Criar uma política de proteção de dados usando API

Crie uma política de proteção de dados do Amazon SNS usando a AWS API.

Como criar uma política de proteção de dados com um tópico do Amazon SNS (API da AWS)

Use a propriedade `DataProtectionPolicy` de um tópico padrão do Amazon SNS:

- `CreateTopic`

Para recuperar ou criar uma política de proteção de dados para um tópico existente do Amazon SNS (API da AWS)

Chame uma das seguintes operações:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Criar políticas de proteção de dados no Amazon SNS use o CLI

O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Amazon Simple Notification Service endpoints and quotas (Endpoints e cotas do Amazon Simple Notification Service).

Criar políticas de proteção de dados usando AWS CLI

Crie uma política de proteção de dados do Amazon SNS usando o. AWS Command Line Interface

Como criar uma política de proteção de dados com um tópico do Amazon SNS (AWS CLI)

Use essa opção para criar uma política de proteção de dados com um tópico padrão do Amazon SNS:

- create-topic

Como recuperar ou criar uma política de proteção de dados para um tópico existente do Amazon SNS (AWS CLI)

Chame uma das seguintes operações:

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

Criação de políticas de proteção de dados no Amazon SNS usando CloudFormation

O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Amazon Simple Notification Service endpoints and quotas (Endpoints e cotas do Amazon Simple Notification Service).

Criar políticas de proteção de dados (CloudFormation)

Crie uma política de proteção de dados do Amazon SNS usando. AWS CloudFormation

Como criar uma política de proteção de dados com um tópico do Amazon SNS (CloudFormation)

Use essa opção para criar uma política de proteção de dados com um tópico padrão do Amazon SNS:

- AWS::SNS::Topic

Criar políticas de proteção de dados no Amazon SNS use o console

O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Amazon Simple Notification Service endpoints and quotas (Endpoints e cotas do Amazon Simple Notification Service).

Como criar uma política de proteção de dados com um tópico do Amazon SNS (console)

Use essa opção para criar uma política de proteção de dados com um tópico padrão do Amazon SNS.

1. Faça login no console do Amazon SNS.
2. Selecione um tópico ou crie um. Para obter mais detalhes sobre como criar tópicos, consulte Criar um tópico do Amazon SNS.
3. Na página Create topic (Criar tópico), na seção Details (Detalhes), selecione Standard (Padrão).
   a. Insira um Nome para o tópico.
   b. (Opcional) Insira um Nome de exibição para o tópico.
4. Amplie Data protection policy (Política de proteção de dados).
5. Selecione um Configuration mode (Modo de configuração):
   - Basic (Básico): defina uma política de proteção de dados usando um menu simples.
   - Avançado: defina uma política de proteção de dados personalizada usando JSON.
6. (Opcional) Para criar o próprio identificador de dados personalizado, expanda a seção Configuração personalizada do identificador de dados e faça o seguinte:

a. Insira um nome exclusivo para o identificador de dados personalizado. Os nomes de identificadores de dados personalizados aceitam caracteres alfanuméricos, sublinhado (_) e hífen (-). São aceitos até 128 caracteres. Esse nome não pode compartilhar o mesmo nome de um identificador de dados gerenciado. Para obter uma lista completa de limitações de identificadores de dados personalizados, consulte Restrições de identificadores de dados personalizados.

b. Insira uma expressão regular (RegEx) para o identificador de dados personalizado. RegExsuporta caracteres alfanuméricos, caracteres RegEx reservados e símbolos. RegEx tem um comprimento máximo de 200 caracteres. Se RegEx for muito complicado, o Amazon SNS falhará na chamada da API. Para obter uma lista completa das RegEx limitações, consulteRestrições de identificadores de dados personalizados.

c. (Opcional) Selecione Adicionar identificador de dados personalizado para adicionar outros identificadores de dados, conforme necessário. As políticas de proteção de dados são compatíveis com dez identificadores de dados personalizados, no máximo.

7. Selecione as declarações que você gostaria de adicionar à sua política de proteção de dados. Você pode adicionar os tipos de instrução audit (auditar), de-identify (desidentificar) (mascarar ou eliminar) e deny (negar) (bloquear) à mesma política de proteção de dados.

   a. Adicionar declaração de auditoria: configure quais dados sigilosos devem ser auditados, qual porcentagem de mensagens você deseja auditar nesses dados e para onde enviar os logs de auditoria.

      > **Note**
      >
      > Somente uma declaração de auditoria é permitida por política ou tópico de proteção de dados.

      i. Selecione data identifiers (identificadores de dados) para definir os dados sigilosos que você deseja auditar.

      ii. Em Audit sample rate (Taxa de amostragem de auditoria), insira a porcentagem de mensagens a serem auditadas quanto a informações sigilosas, até no máximo 99%.

      iii. Em Destino da auditoria, selecione qual Serviços da AWS deseja enviar os resultados da descoberta de auditoria e insira um nome de destino para cada um AWS service

(Serviço da AWS) que você usa. Você pode selecionar entre os seguintes serviços da Amazon Web Services:

- Amazon CloudWatch — CloudWatch Logs é a solução de registro AWS padrão. Usando o CloudWatch Logs, você pode realizar análises de registros usando o Logs Insights (veja exemplos aqui) e criar métricas e alarmes. CloudWatch É nos registros que muitos serviços publicam registros, o que facilita a agregação de todos os registros usando uma única solução. Para obter informações sobre a Amazon CloudWatch, consulte o Guia CloudWatch do usuário da Amazon.
- Amazon Data Firehose — O Firehose satisfaz as demandas de streaming em tempo real para o Splunk e o OpenSearch Amazon Redshift para análises adicionais de log. Para obter informações sobre o Amazon Data Firehose, consulte o Guia do usuário do Amazon Data Firehose.
- Amazon Simple Storage Service: o Amazon S3 é um destino de log com bom custo-benefício para fins de arquivamento. Pode ser necessário manter os logs por um período de anos. Nesse caso, você pode colocar os logs no Amazon S3 para reduzir os custos. Para obter informações sobre o Amazon Simple Storage Service, consulte o Guia do usuário do Amazon Simple Storage Service.

b. Adicionar uma instrução de desidentificação: configure os dados confidenciais que você deseja desidentificar na mensagem, se deseja ou não mascarar ou eliminar esses dados, e as contas para interromper a entrega desses dados.

   i. Em Identificadores de dados, selecione os dados confidenciais que deseja desidentificar.

   ii. Em Definir essa declaração de desidentificação para, selecione as AWS contas ou os diretores do IAM aos quais essa declaração de desidentificação se aplica. Você pode aplicá-la a todas as AWS contas ou a AWS contas específicas ou entidades do IAM (raízes da conta, funções ou usuários) que usam conta IDs ou entidade do IAM ARNs. Separe vários IDs ou ARNs use uma vírgula (,).

   As seguintes entidades principais do IAM são compatíveis:

   - IAM account principals (Entidades principais de conta do IAM): por exemplo, `arn:aws:iam::AWS-account-ID:root`.
   - IAM role principals (Entidades principais de perfil do IAM): por exemplo, `arn:aws:iam::AWS-account-ID:role/role-name`.

- IAM user principals (Entidades principais de usuário do IAM): por exemplo, `arn:aws:iam::AWS-account-ID:user/user-name`.

iii. Em De-identify Option (Opção de desidentificação), selecione como deseja desidentificar os dados confidenciais. Há compatibilidade com as seguintes opções:

- Redact (Eliminar): remoção total dos dados. Por exemplo, e-mail: `classified@amazon.com` se torna e-mail: .
- Mask (Máscara): substitui os dados por caracteres únicos. Por exemplo, e-mail: `classified@amazon.com` se torna e-mail: `**********`.

iv. (Opcional) Continue adicionando instruções de desidentificação conforme necessário.

c. Adicionar declaração de negação: configure quais dados sigilosos não devem percorrer o tópico e quais entidades principais não devem entregar esses dados.

i. Em data direction (direcionamento de dados), escolha o direcionamento das mensagens para a instrução de negação:

- Mensagens de entrada: aplique essa declaração de negação às mensagens enviadas ao tópico.
- Mensagens de saída: aplique essa declaração de negação às mensagens que o tópico entrega aos endpoints de assinatura.

ii. Selecione data identifiers (identificadores de dados) para definir os dados sigilosos que você deseja negar.

iii. Selecione as IAM principals (Entidades principais do IAM) que se aplicam a essa instrução de negação. Você pode aplicá-la a todas as AWS contas, a entidades específicas Contas da AWS ou do IAM (por exemplo, raízes da conta, funções ou usuários) que usam conta IDs ou entidade do IAM ARNs. Separe vários IDs ou ARNs use uma vírgula (,). As seguintes entidades principais do IAM são compatíveis:

- IAM account principals (Entidades principais de conta do IAM): por exemplo, `arn:aws:iam::AWS-account-ID:root`.
- IAM role principals (Entidades principais de perfil do IAM): por exemplo, `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM user principals (Entidades principais de usuário do IAM): por exemplo, `arn:aws:iam::AWS-account-ID:user/user-name`.

iv. (Opcional) Continue adicionando declarações de negação conforme necessário.

Criar políticas de proteção de dados do Amazon SNS para proteger os dados das mensagens usando o SDK

O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Amazon Simple Notification Service endpoints and quotas (Endpoints e cotas do Amazon Simple Notification Service).

Criar políticas de proteção de dados para proteger os dados das mensagens usando o AWS SDK

Crie uma política de proteção de dados do Amazon SNS usando o AWS SDK.

Como criar uma política de proteção de dados com um tópico do Amazon SNS (AWS SDK)

Use as seguintes opções para criar uma política de proteção de dados com um tópico padrão do Amazon SNS:

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();
```

Como recuperar ou criar uma política de proteção de dados para um tópico existente do Amazon SNS (AWS SDK)

Use as seguintes opções para criar ou recuperar uma política de proteção de dados com um tópico padrão do Amazon SNS:

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();
```

Excluir políticas de proteção de dados no Amazon SNS

Você pode excluir as políticas de proteção de dados do Amazon SNS usando a AWS API,, AWS CLI AWS CloudFormation, ou. AWS Management Console

Para obter informações gerais sobre políticas de proteção de dados do Amazon SNS, consulte Compreender as políticas de proteção de dados do Amazon SNS.

O número e o tamanho dos recursos da política de proteção de dados do Amazon SNS em uma conta da AWS são limitados. Para obter mais informações, consulte Controle de utilização da API do Amazon SNS na Referência geral da AWS.

Excluir políticas de proteção de dados usando o console

Como excluir uma política de proteção de dados gerenciados usando o console

1. Faça login no console do Amazon SNS.

2. Selecione o tópico que contém a política de proteção de dados que você deseja excluir.
3. Selecione Editar.
4. Expanda a seção Data protection policy (Política de proteção de dados).
5. Selecione Remove (Remover) ao lado da declaração da política de proteção de dados que você deseja remover.
6. Escolha Salvar alterações.

Excluir uma política de proteção de dados usando uma string JSON em branco

Você pode excluir uma política de proteção de dados atualizando-a para uma string JSON em branco.

Excluindo uma política de proteção de dados usando o AWS CLI

Você pode excluir uma política de proteção de dados usando a AWS CLI.

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

Identificadores de dados do Amazon SNS

O Amazon SNS usa uma combinação de critérios e técnicas, inclusive machine learning e correspondência de padrões, para detectar dados sigilosos. Esses critérios e técnicas, coletivamente denominados identificadores de dados, podem detectar uma lista extensa e crescente de tipos de dados sigilosos para muitos países e regiões. Os identificadores de dados gerenciados do Amazon SNS oferecem tipos de dados predefinidos para proteger dados financeiros, informações pessoais de saúde (PHI) e informações de identificação pessoal (PII). Também é possível usar identificadores de dados personalizados para criar seus próprios identificadores de dados personalizados para um caso de uso específico.

Usar identificadores de dados gerenciados no Amazon SNS

O que são identificadores de dados gerenciados?

Os identificadores de dados gerenciados do Amazon SNS são projetados para detectar um tipo específico de dados confidenciais, como números de cartão de crédito, chaves de acesso AWS secretas ou números de passaportes de um determinado país ou região. Ao criar uma política de proteção de dados, você pode configurar o Amazon SNS para usar esses identificadores a

fim de analisar mensagens que estão passando pelo tópico e tomar medidas quando elas forem detectadas.

O Amazon SNS pode detectar as seguintes categorias de dados sigilosos usando identificadores de dados gerenciados:

- Credenciais, como chaves privadas ou chaves de acesso AWS secretas
- Identificadores de dispositivos, como endereço IP ou MAC.
- Informações financeiras, como números de cartão de crédito
- Informações de saúde, para PHI, como seguro de saúde ou números de identificação médica.
- Informações pessoais, para PII, como carteiras de habilitação ou números de previdência social.

Em cada categoria, o Amazon SNS pode detectar vários tipos de dados sigilosos. Os tópicos dessa seção listam e descrevem cada tipo e todos os requisitos relevantes para detectá-lo. Para cada tipo, eles também indicam o identificador exclusivo (ID) do identificador de dados gerenciados projetado para detectar os dados. Ao criar uma política de proteção de dados, você pode usar esse ID para incluir o identificador de dados gerenciados a ser detectado pela proteção de dados de mensagens.

Requisitos de palavras-chave

Para detectar determinados tipos de dados sigilosos, o Amazon SNS verifica as palavras-chave próximas aos dados. Se esse for o caso de determinado tipo de dado, um tópico subsequente nessa seção indicará requisitos específicos de palavras-chave para esses dados.

As palavras-chave não diferenciam maiúsculas de minúsculas. Além disso, se uma palavra-chave contiver um espaço, o Amazon SNS faz automaticamente a correspondência com as variações de palavras-chave que não contêm o espaço ou contêm um sublinhado (_) ou um hífen (-) em vez do espaço. Em determinados casos, o Amazon SNS também estende ou abrevia uma palavra-chave para abordar variações comuns da palavra-chave.

Identificadores de dados gerenciados pelo Amazon SNS para tipos de dados sigilosos

A tabela a seguir lista e descreve os tipos de credencial, dispositivo, informações financeiras, médicas e pessoais de saúde (PHI) que o Amazon SNS pode detectar usando identificadores de dados gerenciados. Eles são uma adição a determinados tipos de dados que também podem ser qualificados como informações de identificação pessoal (PII).

Os identificadores de dados dependentes da região exigem o nome do identificador com um hífen e os códigos de duas letras (ISO 3166-1 alfa-2). Por exemplo, DriversLicense -US.

Identificador	Categoria	Países/idiomas
BankAccountNumber	Financeiro	DE, ES, FR, GB, IT
CepCode	Pessoal	BR
Cnpj	Pessoal	BR
CpfCode	Pessoal	BR
DriversLicense	Pessoal	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	Integridade	EUA
ElectoralRollNumber	Pessoal	GB
HealthInsuranceCardNumber	Integridade	UE
HealthInsuranceClaimNumber	Integridade	EUA
HealthInsuranceNumber	Integridade	FR
HealthcareProcedureCode	Integridade	EUA
IndividualTaxIdentificationNumber	Pessoal	EUA
InseeCode	Pessoal	FR
MedicareBeneficiaryNumber	Integridade	EUA
NationalDrugCode	Integridade	EUA
NationalIdentificationNumber	Pessoal	DE, ES, IT
NationalInsuranceNumber	Pessoal	GB

Identificador	Categoria	Países/idiomas
NationalProviderId	Integridade	EUA
NhsNumber	Integridade	GB
NieNumber	Pessoal	ES
NifNumber	Pessoal	ES
PassportNumber	Pessoal	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	Pessoal	CA
PersonalHealthNumber	Integridade	CA
PhoneNumber	Pessoal	BR, DE, ES, FR, GB, IT, US
PostalCode	Pessoal	CA
RgNumber	Pessoal	BR
SocialInsuranceNumber	Pessoal	CA
Ssn	Pessoal	ES, US
TaxId	Pessoal	DE, ES, FR, GB
ZipCode	Pessoal	EUA

Identificadores compatíveis que são independentes do idioma/região

Identificador	Categoria
Endereço	Pessoal
AwsSecretKey	Credenciais
CreditCardExpiration	Financeiro

Identificador	Categoria
CreditCardNumber	Financeiro
CreditCardSecurityCode	Financeiro
EmailAddress	Pessoal
IpAddress	Pessoal
LatLong	Pessoal
Name	Pessoal
OpenSshPrivateKey	Credenciais
PgpPrivateKey	Credenciais
PkcsPrivateKey	Credenciais
PuttyPrivateKey	Credenciais
VehicleIdentificationNumber	Pessoal

Tipos de dados sigilosos do Amazon SNS: credenciais

A tabela a seguir lista e descreve os tipos de credencial que o Amazon SNS pode detectar usando identificadores de dados gerenciados.

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Países e regiões
AWS chave de acesso secreta	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Any

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Países e regiões
Chave privada OpenSSH	OpenSshPrivateKey	Não	Any
Chave privada PGP	PgpPrivateKey	Não	Any
Chave privada do padrão de criptografia de chave pública (PKCS)	PkcsPrivateKey	Não	Any
Chave privada PuTTY	PuttyPrivateKey	Não	Any

Identificador de dados ARNs para tipos de dados de credenciais

A seguir, são listados os nomes de recursos da Amazon (ARNs) para os identificadores de dados que você pode adicionar às suas políticas de proteção de dados.

Identificador de dados de credencial ARNs
arn:aws:dataprotection: :aws:data-identifier/ AwsSecretKey
arn:aws:dataprotection: :aws:data-identifier/ OpenSshPrivateKey
arn:aws:dataprotection: :aws:data-identifier/ PgpPrivateKey
arn:aws:dataprotection: :aws:data-identifier/ PkcsPrivateKey
arn:aws:dataprotection: :aws:data-identifier/ PuttyPrivateKey

Tipos de dados sigilosos do Amazon SNS: dispositivos

A tabela a seguir lista e descreve os tipos de identificador de dispositivos que o Amazon SNS pode detectar usando identificadores de dados gerenciados.

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Países e regiões
Endereço IP	IpAddress	Não	Any

Identificador de dados ARNs para tipos de dados de dispositivos

A seguir, são listados os nomes de recursos da Amazon (ARNs) para os identificadores de dados que você pode adicionar às suas políticas de proteção de dados.

ARN de identificador de dados de dispositivos
arn:aws:dataprotection: :aws:data-identifier/ IpAddress

Tipos de dados sigilosos do Amazon SNS: financeiros

A tabela a seguir lista e descreve os tipos de informações financeiras que o Amazon SNS pode detectar usando identificadores de dados gerenciados.

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de conta bancária	BankAccountNumber BankAccountNumber-US	Sim, consulte Palavras-chave para números de conta bancária.	Isso inclui: Números de contas bancárias internacionais (IBANs) que consistem em até 34 caracteres alfanuméricos, incluindo elementos como o código do país.	França, Alemanha, Itália, Espanha, Reino Unido

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Data de validade do cartão de crédito	CreditCardExpiration	exp d, exp m, exp y, expiração, expirar	–	Any
Dados da faixa magnética do cartão de crédito	CreditCardMagneticStripe	Sim, incluindo: card data, iso7813, mag, magstripe, stripe, swipe.	Isso inclui as faixas 1 e 2.	Any

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Números de cartão de crédito	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	A detecção exige que os dados sejam uma sequência de 13 a 19 dígitos que siga a fórmula de cheque de Luhn e use um prefixo de número de cartão padrão para qualquer um dos seguintes tipos de cartão de crédito: American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card Bureau (JCB), Mastercard e Visa (link sobrescrito abaixo de 1). UnionPay	Any

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Código de verificação do cartão de crédito	CreditCardSecurityCode	id do cartão, código de identificação do cartão, número de identificação do cartão, código de segurança do cartão, código de validação do cartão, número de validação do cartão, dados de verificação do cartão, valor de verificação do cartão, cvc, cvc2, cvv, cvv2, código de verificação elo	–	Any

1. O Amazon SNS não relata ocorrências das seguintes sequências, que os emissores de cartão de crédito reservaram para testes públicos:

   122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,

5204740009900014, 5420923878724339, 5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194, 5555555555554444, 5610591081018250, 6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441, 630495060000000000, 6331101999990016, 6759649826438453, 6799990100000000019 e 76009244561.

Palavras-chave para números de conta bancária

Use as seguintes palavras-chave para detectar números de contas bancárias internacionais (IBANs) que consistem em até 34 caracteres alfanuméricos, incluindo elementos como o código do país.

País ou região	Palavras-chave
França	account code, account number, accountno #, accountnu mber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte
Alemanha	account code, account number, accountno #, accountnu mber#, bankleitz ahl, bban, customer

País ou região	Palavras-chave			
	account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
Itália	account code, account number, accountno #, accountnumber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

País ou região	Palavras-chave			
Espanha	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			
Reino Unido	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

País ou região	Palavras-chave			
EUA	bank account, bank acct, checking account, checking acct, deposit account, deposit acct, savings account, savings acct, chequing account, chequing acct			

Identificador de dados ARNs para tipos de dados financeiros

A seguir, são listados os nomes de recursos da Amazon (ARNs) para os identificadores de dados que você pode adicionar às suas políticas de proteção de dados.

Identificador de dados financeiros ARNs
arn:aws:dataprotection: :aws:data-identifier/ -DE BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR BankAccountNumber
arn:aws:proteção de dados: :aws:data-identifier/ -GB BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -US BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ CreditCardExpiration
arn:aws:dataprotection: :aws:data-identifier/ CreditCardNumber

Identificador de dados financeiros ARNs
arn:aws:dataprotection: :aws:data-identifier/ CreditCardSecurityCode

Tipos de dados sigilosos do Amazon SNS: informações de saúde protegidas (PHI)

A tabela a seguir lista e descreve os tipos de informações de saúde protegidas (PHI) que o Amazon SNS pode detectar usando identificadores de dados gerenciados.

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Países e regiões
Número de registro da Agência Antidrogas (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	EUA
Número do cartão de seguro de saúde (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicuraz ione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber# , gesundheitskarte, hälsokort, cartão de saúde, número do cartão de saúde, cartão de seguro de saúde, número do seguro de saúde, número do cartão de seguro, krankenve rsicherungskarte, krankenversicherun gsnummer, número	UE

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Países e regiões
		da conta médica, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	EUA
Número de identificação médica ou do seguro de saúde	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Países e regiões
Código do Healthcare Common Procedure Coding System (HCPCS)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	EUA
Número do beneficiário do Medicare (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	EUA
National Drug Code (NDC)	NationalDrugCode	national drug code, ndc	EUA
National Provider Identifier (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	EUA
Número do Serviço Nacional de Saúde (NHS)	NhsNumber	national health service, NHS	GB
Personal Health Number (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Palavras-chave para números de identificação médica e do seguro de saúde

Para detectar vários tipos de número de identificação médica e de seguro de saúde, o Amazon SNS exige que uma palavra-chave esteja próxima aos números. Isso inclui números do Cartão Europeu de Seguro de Doença (UE, Finlândia), números de seguro de saúde (França), identificadores de beneficiários do Medicare (EUA), números de seguro nacional (Reino Unido), números do NHS (Reino Unido) e números pessoais de saúde (Canadá).

A tabela a seguir lista as palavras-chave que o Amazon SNS reconhece para países e regiões específicos.

País ou região	Palavras-chave
Canadá	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
UE	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
Finlândia	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskor

País ou região	Palavras-chave
	t, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
França	carte d'assuré social, carte vitale, insurance card
Reino Unido	serviço nacional de saúde, NHS
EUA	mbi, medicare beneficiary

Identificador de dados ARNs para tipos de dados de informações de saúde protegidas (PHI)

O seguinte lista o identificador de dados Amazon Resource Names (ARNs) que pode ser usado nas políticas de proteção de dados de PHI.

Identificador de dados PHI ARNs
arn:aws:dataprotection: :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:dataprotection: :aws:data-identifier/ -US HealthcareProcedureCode
arn:aws:dataprotection: :aws:data-identifier/ -EU HealthInsuranceCardNumber
arn:aws:dataprotection: :aws:data-identifier/ -US HealthInsuranceClaimNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR HealthInsuranceNumber
arn:aws:dataprotection: :aws:data-identifier/ -US MedicareBeneficiaryNumber
arn:aws:dataprotection: :aws:data-identifier/ -US NationalDrugCode
arn:aws:proteção de dados: :aws:data-identifier/ -GB NationalInsuranceNumber
arn:aws:dataprotection: :aws:data-identifier/ -US NationalProviderId
arn:aws:proteção de dados: :aws:data-identifier/ -GB NhsNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PersonalHealthNumber

Tipos de dados sigilosos do Amazon SNS: informações de identificação pessoal (PII)

A tabela a seguir lista e descreve os tipos de informações de identificação pessoal (PII) que o Amazon SNS pode detectar usando identificadores de dados gerenciados.

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Datas de nascimento	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	O suporte inclui a maioria dos formatos de data, como todos os dígitos e combinações de dígitos e nomes de meses. Os componentes de data podem ser separados por espaços, barras (/) ou hífens (-).	Any
Código de endereçamento postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brasil
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	Brasil

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Cadastro de Pessoa Física (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	Brasil

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de identificação da carteira de habilitação	DriversLicense	Sim, consulte Palavras-chave para números de identificação da carteira de habilitação.	–	Austrália, Áustria, Bélgica, Bulgária, Canadá, Croácia, Chipre, República Tcheca, Dinamarca, Estônia, Finlândia, França, Alemanha, Grécia, Hungria, Irlanda, Itália, Letônia, Lituânia, Luxemburgo, Malta, Holanda, Polônia, Portugal, Romênia, Eslováquia, Eslovênia, Espanha, Suécia, Reino Unido, EUA

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de registro eleitoral	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	Reino Unido
Identificação do contribuinte individual	IndividualTaxIdentificationNumber	Sim, consulte Palavras-chave para identificação do contribuinte e números de referência.	–	EUA
Instituto Nacional de Estatística e Estudos Econômicos (INSEE)	InseeCode	Sim, consulte Palavras-chave para números de identificação nacional.	–	França

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de identificação nacional	NationalIdentificationNumber	Sim, consulte Palavras-chave para números de identificação nacional.	Isso inclui identificadores do Documento Nacional de Identidade (DNI) (Espanha), códigos fiscais (Itália) e números de carteira de identidade nacional (Alemanha).	Alemanha, Itália, Espanha
Número do Seguro Nacional (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	Reino Unido
Número de identidad de extranjero (NIE)	NieNumber	Sim, consulte Palavras-chave para identificação do contribuinte e números de referência.	–	Espanha

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de Identificación Fiscal (NIF)	NifNumber	Sim, consulte Palavras-chave para identificação do contribuinte e números de referência.	–	Espanha
Número de passaporte	PassportNumber	Sim, consulte Palavras-chave para números de passaporte.	–	Canadá, França, Alemanha, Itália, Espanha, Reino Unido, EUA

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de residência permanente	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	–	Canadá

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número de telefone	PhoneNumber	Brasil: as palavras-chave também incluem: cel, celular, fone, móvel, número residencial, numero residencial, telefone Outros: celular, contato, fax, número de fax, celular, fone, número de fone, telefone, número de telefone	Isso inclui números de chamada gratuita nos EUA e números de fax. Se uma palavra-chave estiver próxima dos dados, o número não precisará incluir o código do país. Se uma palavra-chave estiver próxima dos dados, o número não precisará incluir o código do país.	Brasil, Canadá, França, Alemanha, Itália, Espanha, Reino Unido, EUA
CEP	PostalCode	No	–	Canadá
Registro Geral (RG)	RgNumber	Sim, consulte Palavras-chave para números de identificação nacional.	–	Brasil

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Número do Seguro Social (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Canadá
Número da Previdência Social (SSN)	Ssn	Espanha: número de la seguridad social, social security no., social security no. número de la seguridad social, social security number, socialsecurityno#, ssn, ssn# EUA: previdência social, ss#, ssn	–	Espanha, EUA
Identificação do contribuinte ou número de referência	TaxId	Sim, consulte Palavras-chave para identificação do contribuinte e números de referência.	Isso inclui TIN (França); Steueridentifikationsnummer (Alemanha); CIF (Espanha) e TRN, UTR (Reino Unido).	França, Alemanha, Espanha, Reino Unido

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Código postal dos EUA	ZipCode	zip code, zip+4	–	EUA
Endereço postal	Address	No	Embora uma palavra-chave não seja necessária, a detecção exige que o endereço inclua o nome de uma cidade ou local e um CEP.	Austrália, Canadá, França, Alemanha, Itália, Espanha, Reino Unido, EUA
Endereço de correio eletrônico	EmailAddress	email, endereço de email, e-mail, endereço de e-mail	–	Any

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Coordenadas do sistema de posicionamento global (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	O Amazon SNS poderá detectar coordenadas de GPS se as coordenadas de latitude e longitude estiverem armazenadas como um par e estiverem no formato de graus decimais (DD), por exemplo, 41.948614 ,-87.655311. O suporte não inclui coordenadas no formato graus, decimais, minutos (DDM), por exemplo, 41°56.9168'N 87°39.3187'W, ou no formato graus, minutos, segundos (DMS), por exemplo, 41°56'55.0104"N 87°39'19.1196"W.	Any

Tipo de detecção	ID do identificador de dados gerenciados	Palavra-chave obrigatória	Mais informações	Países e regiões
Nome completo	Name	No	O Amazon SNS consegue detectar somente nomes completos. O suporte é limitado aos conjuntos de caracteres latinos.	Any
Número de identificação de veículo (VIN)	VehicleIdentificationNumber	Fahrgestellnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	O Amazon SNS pode detectar VINs que consistem em uma sequência de 17 caracteres e aderem aos padrões ISO 3779 e 3780. Esses padrões foram projetados para uso em todo o mundo.	Any

Palavras-chave para números de identificação da carteira de habilitação

Para detectar vários tipos de número de identificação da carteira de habilitação, o Amazon SNS exige que uma palavra-chave esteja próxima aos números. A tabela a seguir lista as palavras-chave que o Amazon SNS reconhece para países e regiões específicos.

País ou região	Palavras-chave
Austrália	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
Áustria	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Bélgica	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgária	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Canadá	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,

País ou região	Palavras-chave
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
Croácia	vozačka dozvola
Chipre	άδεια οδήγησης
República Tcheca	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
Dinamarca	kørekort, kørekortnummer
Estônia	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finlândia	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
França	permis de conduire
Alemanha	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Grécia	δεια οδήγησης, adeia odigisis
Hungria	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Irlanda	ceadúnas tiomána
Itália	patente di guida, patente di guida numero, patente guida, patente guida numero

País ou região	Palavras-chave
Letônia	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Lituânia	vairuotojo pažymėjimas
Luxemburgo	fahrerlaubnis, führerschäin
Malta	liċenzja tas-sewqan
Holanda	permis de conduire, rijbewijs, rijbewijsnummer
Polônia	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
Portugal	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
Romênia	numărul permisului de conducere, permis de conducere
Eslováquia	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Eslovênia	vozniško dovoljenje

País ou região	Palavras-chave
Espanha	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
Suécia	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.
Reino Unido	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
EUA	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

Palavras-chave para números de identificação nacional

Para detectar vários tipos de número de identificação nacional, o Amazon SNS exige que uma palavra-chave esteja próxima aos números. Isso inclui identificadores do Documento Nacional

de Identidad (DNI) (Espanha), códigos do Instituto Nacional de Estatística e Estudos Econômicos (INSEE) da França, números da carteira de identidade nacional alemã e números do Registro Geral (RG) (Brasil).

A tabela a seguir lista as palavras-chave que o Amazon SNS reconhece para países e regiões específicos.

País ou região	Palavras-chave
Brasil	registro geral, rg
França	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Alemanha	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Itália	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Espanha	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#

Palavras-chave para números de passaporte

Para detectar vários tipos de número de passaporte, o Amazon SNS exige que uma palavra-chave esteja próxima aos números. A tabela a seguir lista as palavras-chave que o Amazon SNS reconhece para países e regiões específicos.

País ou região	Palavras-chave
Canadá	passeport, passeport#, passport, passport#, passportno, passportno#
França	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
Alemanha	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Itália	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Espanha	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
Reino Unido	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
EUA	passport, travel document

Palavras-chave para identificação do contribuinte e números de referência

Para detectar vários tipos de número de referência e identificação de contribuinte, o Amazon SNS exige que uma palavra-chave esteja próxima aos números. A tabela a seguir lista as palavras-chave que o Amazon SNS reconhece para países e regiões específicos.

País ou região	Palavras-chave
Brasil	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
França	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Alemanha	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
Espanha	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#
Reino Unido	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
EUA	número de identificação de contribuinte individual, itin, i.t.i.n.

Identificador de dados ARNs para informações de identificação pessoal (PII)

A tabela a seguir lista os nomes de recursos da Amazon (ARNs) para os identificadores de dados que você pode adicionar às suas políticas de proteção de dados.

Identificador de dados PII ARNs
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:proteção de dados: :aws:data-identifier/ -BR CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:proteção de dados: :aws:data-identifier/ -BR CpfCode
arn:aws:dataprotection: :aws:data-identifier/ DateOfBirth
arn:aws:dataprotection: :aws:data-identifier/ -AT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -AU DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -BE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -BG DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -CA DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -CY DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -CZ DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -DE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -DK DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -EE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -ES DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ FI DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -FR DriversLicense

Identificador de dados PII ARNs
arn:aws:proteção de dados: :aws:data-identifier/ -GB DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -GR DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -HR DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -HU DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -IE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -IT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -LT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -LU DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -LV DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -MT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -NL DriversLicense
arn:aws:proteção de dados: :aws:data-identifier/ -PL DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -PT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -RO DriversLicense
arn:aws:proteção de dados: :aws:data-identifier/ -SE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -SI DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -SK DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -US DriversLicense
arn:aws:proteção de dados: :aws:data-identifier/ -GB ElectoralRollNumber
arn:aws:dataprotection: :aws:data-identifier/ EmailAddress

Identificador de dados PII ARNs
arn:aws:dataprotection: :aws:data-identifier/ -US IndividualTaxIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR InseeCode
arn:aws:dataprotection: :aws:data-identifier/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:dataprotection: :aws:data-identifier/ -DE NationalIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES NationalIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT NationalIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES NieNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES NifNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -DE PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR PassportNumber
arn:aws:proteção de dados: :aws:data-identifier/ -GB PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -US PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PermanentResidenceNumber
arn:aws:proteção de dados: :aws:data-identifier/ -BR PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -DE PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES PhoneNumber

Identificador de dados PII ARNs
arn:aws:dataprotection: :aws:data-identifier/ -FR PhoneNumber
arn:aws:proteção de dados: :aws:data-identifier/ -GB PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -US PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PostalCode
arn:aws:proteção de dados: :aws:data-identifier/ -BR RgNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA SocialInsuranceNumber
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:dataprotection: :aws:data-identifier/ -DE TaxId
arn:aws:dataprotection: :aws:data-identifier/ -ES TaxId
arn:aws:dataprotection: :aws:data-identifier/ -FR TaxId
arn:aws:proteção de dados: :aws:data-identifier/ -GB TaxId
arn:aws:dataprotection: :aws:data-identifier/ VehicleIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -US ZipCode

Usar identificadores de dados personalizados no Amazon SNS

Os identificadores de dados personalizados (CDIs) permitem que você defina suas próprias expressões regulares personalizadas que podem ser usadas em sua política de proteção de dados. Usando identificadores de dados personalizados, é possível segmentar casos de uso de informações de identificação pessoal (PII) específicos da empresa que os identificadores de dados gerenciados não podem fornecer. Por exemplo, você pode usar um identificador de dados personalizado para

procurar funcionários específicos da empresa. IDs Identificadores de dados personalizados podem ser usados em conjunto com identificadores de dados gerenciados.

O que são identificadores de dados personalizados?

Os identificadores de dados personalizados (CDIs) permitem que você defina suas próprias expressões regulares personalizadas que podem ser usadas em sua política de proteção de dados. Usando identificadores de dados personalizados, é possível segmentar casos de uso de informações de identificação pessoal (PII) específicos da empresa que os identificadores de dados gerenciados não podem fornecer. Por exemplo, você pode usar um identificador de dados personalizado para procurar funcionários específicos da empresa. IDs Identificadores de dados personalizados podem ser usados em conjunto com identificadores de dados gerenciados.

Usar identificadores de dados personalizados na política de proteção de dados

A política de proteção de dados a seguir instrui o tópico do Amazon SNS a detectar cargas que transportam IDs funcionários específicos da empresa e, em seguida, IDs mascará-las usando o símbolo de hash (#).

1. Crie um bloco `Configuration` na política de proteção de dados.
2. Insira um `Name` para o identificador de dados personalizado. Por exemplo, **`EmployeeId`**.
3. Insira um `Regex` para o identificador de dados personalizado. Por exemplo, **`EID-\d{9}-US`**.
4. Consulte o identificador de dados personalizado a seguir em uma declaração de política.

```
{
 "Name": "__example_data_protection_policy",
 "Description": "Example data protection policy",
 "Version": "2021-06-01",
 "Configuration": {
   "CustomDataIdentifier": [
     {"Name": "EmployeeId", "Regex": "EID-\d{9}-US"}
   ]
 },
 "Statement": [
   {
     "DataDirection": "Inbound",
     "Principal": ["*"],
     "DataIdentifier": [
        "EmployeeId"
     ],
```

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

5. (Opcional) Continue adicionando identificadores de dados personalizados ao bloco `Configuration`, conforme necessário. No momento, as políticas de proteção de dados são compatíveis com dez identificadores de dados personalizados, no máximo.

Restrições de identificadores de dados personalizados

Os identificadores de dados personalizados do Amazon SNS têm as seguintes limitações:

- As políticas de proteção de dados são compatíveis com dez identificadores de dados personalizados, no máximo.
- O tamanho máximo dos nomes de identificadores de dados personalizados é 128 caracteres. Os seguintes caracteres são aceitos:
  - Alfanuméricos: (a – z; A – Z; 0 – 9)
  - Símbolos: (“_” | “-”)
- RegEx tem um comprimento máximo de 200 caracteres. Os seguintes caracteres são aceitos:
  - Alfanuméricos: (a – z; A – Z; 0 – 9)
  - Símbolos: (“_” | “#” | “=” | “@” | “/” | “;” | “,” | “-” | “ ”)
  - RegEx caracteres reservados: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- Os identificadores de dados personalizados não podem compartilhar o mesmo nome de um identificador de dados gerenciados.
- É necessário especificar identificadores de dados personalizados em cada política de proteção de dados para cada tópico do Amazon SNS.

Entrega de mensagens do Amazon SNS

Este tópico descreve como o Amazon SNS lida com a entrega de mensagens em vários cenários. Você aprenderá sobre a entrega de mensagens brutas, em que o Amazon SNS entrega mensagens em seu formato original e não modificado para o endpoint. Você também descobrirá como enviar mensagens de um tópico do Amazon SNS para uma fila do Amazon SQS de Conta da AWS outra forma, fornecendo informações sobre mensagens entre contas.

Este tópico fornece informações sobre a entrega de mensagens do Amazon SNS para uma fila do Amazon SQS ou uma função Regiões da AWS Lambda de forma diferente, como funciona a entrega entre regiões e as considerações envolvidas.

Além disso, você aprenderá a monitorar e interpretar o status de entrega de mensagens, que fornece informações essenciais sobre se as mensagens foram entregues com sucesso ou se encontraram problemas. Nos casos em que a entrega de mensagens falhar, você entenderá o processo de nova tentativa de entrega de mensagens, incluindo como o Amazon SNS tenta reenviar mensagens automaticamente para garantir que elas cheguem aos destinos pretendidos. Este tópico também discute o uso de filas de mensagens não entregues para capturar mensagens que não puderam ser entregues após várias tentativas, permitindo que você analise e solucione essas falhas de forma eficaz.

Entrega de mensagens brutas do Amazon SNS

Para evitar que os endpoints HTTP/S, do Amazon Data Firehose e do Amazon SQS processem a formatação JSON das mensagens, o Amazon SNS permite a entrega de mensagens brutas:

- Ao habilitar a entrega de mensagens brutas para um endpoint do Amazon Data Firehose ou do Amazon SQS, todos os metadados do Amazon SNS são removidos da mensagem publicada e a mensagem é enviada como está.
- Quando você habilita a entrega de mensagens brutas para endpoints HTTP/S, o cabeçalho HTTP `x-amz-sns-rawdelivery` com o valor definido como `true` é adicionado à mensagem, indicando que a mensagem foi publicada sem formatação JSON.
- Quando você habilita a entrega de mensagens brutas para endpoints HTTP/S, o corpo da mensagem, o IP do cliente e os cabeçalhos necessários são entregues. Quando você especifica atributos de mensagem, ela não é enviada.
- Quando você habilita a entrega de mensagens brutas para endpoints do Firehose, o corpo da mensagem é entregue. Quando você especifica atributos de mensagem, ela não é enviada.

Para habilitar a entrega de mensagens brutas usando um AWS SDK, você deve usar a ação da `SetSubscriptionAttribute` API e definir o valor do `RawMessageDelivery` atributo como`true`.

Habilitar a entrega de mensagens brutas usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Na página Tópicos, escolha um tópico assinado para um endpoint HTTP/S, do Firehose ou do Amazon SQS.
4. Na ***MyTopic*** página, na seção Assinatura, escolha uma assinatura e escolha Editar.
5. Na *EXAMPLE1-23bc-4567-d890-ef12g3hij456* página Editar, na seção Detalhes, escolha Habilitar entrega de mensagens brutas.
6. Escolha Salvar alterações.

Exemplos de formatos de mensagens

Nos exemplos a seguir, a mesma mensagem é enviada para a mesma fila do Amazon SQS duas vezes. A única diferença é que a entrega de mensagens brutas está desabilitada para a primeira mensagem e habilitada para a segunda.

- Entrega de mensagens brutas desabilitada

```
{
  "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"
}
```

- Entrega de mensagens brutas habilitada

```
This is a test message.
```

Atributos de mensagem e entrega de mensagens brutas para assinaturas do Amazon SQS

O Amazon SNS oferece suporte à entrega de atributos de mensagem, o que permite que você forneça itens de metadados estruturados, como time stamps, dados geoespaciais, assinaturas e identificadores sobre a mensagem. Para assinaturas Amazon SQS, com Entrega de mensagens brutas habilitado, no máximo dez atributos de mensagem podem ser enviados. Para enviar mais de dez atributos de mensagem, é necessário desabilitar Entrega de mensagens brutas. Porém, o Amazon SNS descarta mensagens com mais de 10 atributos de mensagem direcionadas para assinaturas do Amazon SQS com Entrega de mensagens brutas habilitadas, tratando elas como erros do lado do cliente.

Enviar mensagens do Amazon SNS para uma fila do SQS em uma conta diferente

Este documento descreve como publicar uma notificação em um tópico do Amazon SNS com uma ou mais inscrições em filas do Amazon SQS em outra conta. Você pode configurar o tópico e as filas da mesma forma que faria se eles estivessem na mesma conta (consulte Fanout de notificações do Amazon SNS para filas do Amazon SQS para processamento assíncrono). A maior diferença é a forma de lidar com a confirmação de inscrição, e isso depende de como você inscreve a fila no tópico.

É prática recomendada seguir as etapas mencionadas na seção Proprietário da fila cria a assinatura quando possível, porque a confirmação é automática quando o proprietário da fila cria a inscrição.

> **Note**
>
> Se a fila do Amazon SQS tiver um grande volume de mensagens, recomendamos que o proprietário da fila crie a assinatura.

Proprietário da fila cria a assinatura

A conta que criou a fila do Amazon SQS é proprietária da fila. Quando o proprietário da fila cria uma assinatura, ela não requer confirmação. A fila começa a receber notificações do tópico assim que a ação `Subscribe` é concluída. Para permitir que o proprietário da fila assine o tópico do proprietário, este último deve conceder à conta do proprietário da fila permissão para chamar a ação `Subscribe` no tópico.

Etapa 1: Definir a política de tópico usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Selecione um tópico e escolha Editar.
4. Na *`MyTopic`* página Editar, expanda a seção Política de acesso.
5. Insira a seguinte política:

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

Essa política concede à conta `111122223333` permissão para chamar `sns:Subscribe` em `MyTopic` na conta `123456789012`.

Um usuário com as credenciais da conta `111122223333` pode assinar `MyTopic`. Essa permissão possibilita que o ID da conta delegue a permissão ao usuário ou à função do IAM. Somente os usuários da conta raiz ou administrador terão permissão para chamar `sns:Subscribe`. O usuário ou a função do IAM também precisará ter `sns:subscribe` para permitir a assinatura da respectiva fila.

6. Escolha Salvar alterações.

   Um usuário com as credenciais da conta `111122223333` pode se inscrever MyTopic.

Etapa 2: Para adicionar uma assinatura de fila do Amazon SQS a um tópico em outro usando o Conta da AWSAWS Management Console

Antes de começar, verifique se você tem o ARNs tópico e a fila e se deu permissão ao tópico para enviar mensagens para a fila.

1. Faça login no console do Amazon SQS.
2. No painel de navegação, escolha Queues (Filas).
3. Na lista de filas, escolha a fila que você deseja assinar em um tópico do Amazon SNS.
4. Escolha Subscribe to Amazon SNS topic (Inscrever-se no tópico do Amazon SNS).
5. No menu Especificar um tópico do Amazon SNS disponível para esta fila, escolha o Tópico do Amazon SNS para a fila.
6. Selecione Inserir o ARN do tópico do Amazon SNS e insira o Nome do recurso da Amazon (ARN) do tópico.
7. Escolha Salvar.

> **Note**
>
> - Para poder se comunicar com o serviço, a fila deve ter permissões para o Amazon SNS.
> - Como você é o proprietário da fila, você não precisa confirmar a assinatura.

Um usuário que não é proprietário da fila cria uma assinatura

Qualquer usuário que criar uma assinatura, mas não for o proprietário da fila, deverá confirmar a assinatura.

Quando você usa a ação `Subscribe`, o Amazon SNS envia uma confirmação de assinatura para a fila. A assinatura é exibida no console do Amazon SNS, com seu ID de assinatura definido como Confirmação pendente.

Para confirmar a inscrição, o usuário com permissão para ler mensagens da fila precisa recuperar o URL de confirmação da inscrição e o proprietário da inscrição precisa confirmá-la usando o URL de confirmação. Até que a inscrição seja confirmada, nenhuma notificação publicada no tópico é enviada para a fila. Para confirmar a assinatura, você pode usar o console do Amazon SQS ou a ação `ReceiveMessage`.

> **Note**
>
> Antes de inscrever um endpoint no tópico, certifique-se de que a fila possa receber mensagens do tópico definindo a permissão `sqs:SendMessage` para a fila. Para obter mais informações, consulte Etapa 2: Conceder permissão ao tópico do Amazon SNS para enviar mensagens à fila do Amazon SQS.

Etapa 1: Para adicionar uma assinatura de fila do Amazon SQS a um tópico em outro usando o Conta da AWSAWS Management Console

Antes de começar, verifique se você tem o ARNs tópico e a fila e se deu permissão ao tópico para enviar mensagens para a fila.

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Subscriptions (Assinaturas).
3. Na página Assinaturas, escolha Criar assinatura.
4. Na página Criar assinatura, na seção Detalhes, faça o seguinte:
   a. Para ARN de tópico, insira o ARN do tópico.
   b. Em Protocol (Protocolo), escolha Amazon SQS.
   c. Em Endpoint, insira o ARN de uma fila.
   d. Selecione Create subscription.

> **Note**
>
> - Para poder se comunicar com o serviço, a fila deve ter permissões para o Amazon SNS.

Veja a seguir um exemplo de instrução de política que permite ao tópico do Amazon SNS enviar mensagem à fila do Amazon SQS.

```
{
   "Sid": "Stmt1234",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "sqs:SendMessage",
   "Resource": "arn:aws:sqs:us-west-2:111111111111:QueueName",
   "Condition": {
      "ArnEquals": {
         "aws:SourceArn": "arn:aws:sns:us-west-2:555555555555:TopicName"
      }
   }
}
```

Etapa 2: Para confirmar uma assinatura usando o AWS Management Console

1. Faça login no console do Amazon SQS.
2. Selecione a fila que tem uma inscrição pendente no tópico.
3. Escolha Send and receive messages (Enviar e receber mensagens), depois escolha Poll for messages (Pesquisar mensagens).

   Uma mensagem com a confirmação de assinatura é recebida na fila.
4. Na coluna Corpo, faça o seguinte:

   a. Escolha Mais detalhes.

   b. Na caixa de diálogo Message Details (Detalhes da mensagem), localize e anote o valor SubscribeURL. Este é seu link de assinatura (exemplo abaixo). Para obter detalhes adicionais sobre a validação de token de API, consulte `ConfirmSubscription` na Referência da API do Amazon SNS.

```
https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

c. Anote o link de confirmação da inscrição. O URL deve ser transmitido do proprietário da fila para o proprietário da assinatura. O proprietário da assinatura precisa inserir o URL no console do Amazon SNS.

5. Faça login como o proprietário da assinatura no console do Amazon SNS. O proprietário da assinatura executa a confirmação.
6. Escolha o tópico pertinente.
7. Escolha a inscrição pertinente na tabela de listas de assinatura do tópico. Ela é identificada como “Confirmação pendente”.
8. Clique em Confirm subscription (Confirmar assinatura).
9. Uma janela modal é exibida solicitando o link de confirmação da assinatura. Cole o link de confirmação da inscrição.
10. Selecione Confirm subscription (Confirmar assinatura) na janela modal.

    Uma resposta em XML é exibida, por exemplo:

```
<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>
```

    A fila inscrita está pronta para receber mensagens do tópico.
11. (Opcional) Se você exibir a assinatura do tópico no console do Amazon SNS, poderá ver que a mensagem Confirmação pendente foi substituída pelo ARN da assinatura na coluna ID da assinatura.

Como faço para forçar uma assinatura a exigir autenticação em solicitações de cancelamento de assinatura?

O proprietário da assinatura precisa definir o sinalizador `AuthenticateOnUnsubscribe` como true na confirmação de assinatura.

- `AuthenticateOnUnsubscribe` é definido automaticamente como true quando o proprietário da fila cria a inscrição.
- `AuthenticateOnUnsubscribe` não pode ser definido como true quando o link de confirmação da assinatura é acessado sem autenticação.

Enviar mensagens do Amazon SNS para uma fila do Amazon SQS ou uma função do AWS Lambda em uma conta diferente

O Amazon SNS é compatível com entregas entre regiões, tanto para regiões habilitadas por padrão quanto para regiões de adesão. Para obter a lista atual de regiões da AWS compatíveis com o Amazon SNS, incluindo regiões de adesão, consulte Endpoints e cotas do Amazon Simple Notification Service na Referência geral da Amazon Web Services.

O Amazon SNS oferece suporte à entrega entre regiões de notificações para filas do Amazon SQS e funções do AWS Lambda . Quando uma das regiões é uma região de adesão, você deve especificar um principal de serviço do Amazon SNS diferente na política do recurso inscrito.

O comando de assinatura do Amazon SNS deve ser executado na região em que o Amazon SNS está hospedado, na região correspondente. Por exemplo, se o Amazon SNS estiver na conta “A” na região us-east-1 e a função do Lambda estiver na conta “B” na região us-east-2, o comando da CLI da assinatura deverá ser executado na conta “A” na região us-east-1.

Regiões de adesão

O Amazon SNS é compatível com as seguintes regiões de adesão:

Nome da região	Região
Região África (Cidade do Cabo)	af-south-1
Região Ásia-Pacífico (Hong Kong)	ap-east-1

Nome da região	Região
Ásia-Pacífico (Haiderabade)	ap-south-2
Região Ásia-Pacífico (Jacarta)	ap-southeast-3
Região Ásia-Pacífico (Melbourne)	ap-southeast-4
Região Europa (Milão)	eu-south-1
Região Europa (Espanha)	eu-south-2
Região Europa (Zurique)	eu-central-2
Região de Israel (Tel Aviv)	il-central-1
Região Oriente Médio (Bahrein)	me-south-1
Região do Oriente Médio (Emirados Árabes Unidos)	me-central-1

Para obter informações sobre como habilitar uma região opcional, consulte Gerenciando AWS regiões no Referência geral da Amazon Web Services.

Quando você usa o Amazon SNS para entregar mensagens de regiões de adesão para regiões habilitadas por padrão, é necessário alterar a política de recurso criada para a fila. Substitua o principal `sns.amazonaws.com` por `sns.<opt-in-region>.amazonaws.com`. Por exemplo:

- Para inscrever uma fila do Amazon SQS no Leste dos EUA (N. da Virgínia) em um tópico do Amazon SNS na Ásia-Pacífico (Hong Kong), altere a entidade principal na política de fila para `sns.ap-east-1.amazonaws.com`. As regiões de adesão incluem as regiões lançadas após 20 de março de 2019, que abrangem Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Jacarta), Oriente Médio (Bahrein), Europa (Milão) e África (Cidade do Cabo). As regiões lançadas antes de 20 de março de 2019 são habilitadas por padrão.

Compatibilidade da entrega entre regiões com o Amazon SQS

Tipo de entrega entre regiões	Compatível/não compatível
Região habilitada por padrão para região de adesão	Compatível com o uso de `sns.<opt-in-region>.amazonaws.com` na entidade principal do serviço para a fila
Região de adesão para região habilitada por padrão	Compatível com o uso de `sns.<opt-in-region>.amazonaws.com` na entidade principal do serviço para a fila
Região de adesão para região de adesão	Sem compatibilidade

Veja a seguir um exemplo de uma declaração de política de acesso que permite que um tópico do Amazon SNS em uma região opcional (af-south-1) seja entregue em uma fila do Amazon SQS em uma região (us-east-1). enabled-by-default Ele contém a configuração da entidade principal regionalizada do serviço que é necessária no caminho `Statement/Principal/Service`.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.af-south-1.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
        }
```

```
      }
    },
    ...
  ]
}
```

- Para inscrever uma AWS Lambda função no Leste dos EUA (Norte da Virgínia) em um tópico do Amazon SNS na Ásia-Pacífico (Hong Kong), altere o principal na política da AWS Lambda função para. `sns.ap-east-1.amazonaws.com` As regiões de adesão incluem as regiões lançadas após 20 de março de 2019, que abrangem Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Jacarta), Oriente Médio (Bahrein), Europa (Milão) e África (Cidade do Cabo). As regiões lançadas antes de 20 de março de 2019 são habilitadas por padrão.

  Suporte de entrega entre regiões para AWS Lambda

Tipo de entrega entre regiões	Compatível/não compatível	
Região habilitada por padrão para região de adesão	Sem compatibilidade	
Região de adesão para região habilitada por padrão	Compatível com o uso de `sns.<opt-in-region>.amazonaws.com` na entidade principal do serviço para a função do Lambda	
Região de adesão para região de adesão	Sem compatibilidade	

Status de entrega de mensagens do Amazon SNS

O Amazon SNS fornece suporte para registrar o status de entrega das mensagens de notificação enviadas aos tópicos com os seguintes endpoints do Amazon SNS:

- Amazon Data Firehose
- Amazon Simple Queue Service
- AWS Lambda

- HTTPS
- Endpoint da aplicação da plataforma

Os registros de status de entrega são enviados para o Amazon CloudWatch Logs, fornecendo informações sobre as operações de entrega de mensagens. Esses registros ajudam você a:

- Determine se uma mensagem foi entregue com sucesso em um endpoint.
- Identifique a resposta do endpoint para o Amazon SNS.
- Meça o tempo de permanência da mensagem (tempo entre a data e hora da publicação e a entrega para o endpoint).

Você pode configurar o registro do status de entrega usando a AWS Management Console AWS SDKs,, Query API ou AWS CloudFormation.

Pré-requisitos para o registro do status de entrega

Este tópico descreve as permissões do IAM necessárias para permitir que o Amazon SNS grave registros de entrega e explica a convenção padrão de CloudWatch nomenclatura de grupos de registros. Isso garante que você tenha a configuração e o acesso corretos para monitorar e analisar os registros de entrega de mensagens nos CloudWatch registros.

Permissões obrigatórias do IAM

A função do IAM anexada ao registro do status de entrega deve incluir as seguintes permissões para permitir que o Amazon SNS grave nos registros. CloudWatch Você pode usar uma função existente com essas permissões ou criar uma nova função durante a configuração.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
```

```
  ]
}
```

Convenção de nomenclatura de grupos de registros

Por padrão, o Amazon SNS cria CloudWatch grupos de registros para registros de status de entrega usando a seguinte convenção de nomenclatura. Os fluxos de log dentro desse grupo correspondem aos protocolos de endpoint (por exemplo, Lambda, Amazon SQS). Verifique se você tem permissões para visualizar esses registros no console de CloudWatch registros.

```
sns/<region>/<account-id>/<topic-name>
```

Configurar registro em log do status de entrega usando o AWS Management Console

Este tópico explica como habilitar o registro de status de entrega de mensagens para tópicos do Amazon SNS, incluindo a definição de configurações de registro, a atribuição de funções do IAM e a verificação de que CloudWatch os registros capturam registros de entrega para monitoramento e solução de problemas.

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Selecione o tópico desejado e escolha Editar.
4. Expanda a seção Registro do status de entrega.
5. Escolha o protocolo para o qual você deseja ativar o registro (por exemplo, HTTP, Lambda, Amazon SQS).
6. Insira a taxa de amostragem de sucesso, que é a porcentagem de mensagens bem-sucedidas das quais você deseja receber CloudWatch registros.
7. Na seção de funções do IAM, você deve configurar as funções para o registro de sucesso e de falha:

   - Use uma função de serviço existente — Selecione uma função do IAM existente que tenha as permissões necessárias para o Amazon SNS gravar registros. CloudWatch

- Crie uma nova função de serviço — escolha Criar novas funções para definir as funções do IAM para entregas bem-sucedidas e malsucedidas no console do IAM. Para obter detalhes sobre a permissão, consulte Pré-requisitos para o registro do status de entrega.

8. Escolha Salvar alterações.

   Depois de ativar o registro, você pode visualizar e analisar os CloudWatch registros que contêm o status de entrega da mensagem. Para obter mais informações sobre o uso CloudWatch, consulte a CloudWatchdocumentação.

Verificando a configuração do registro

1. Faça login no console CloudWatch de registros.
2. Localize o grupo de registros chamado`sns/<region>/<account-id>/<topic-name>`.
3. Certifique-se de que existam fluxos de log para o protocolo de endpoint configurado.
4. Envie uma mensagem de teste para seu tópico e confirme se as entradas de registro aparecem, indicando entregas bem-sucedidas ou malsucedidas.

Configurando o registro do status de entrega usando o AWS SDKs

Eles AWS SDKs fornecem, APIs em vários idiomas, a definição de atributos de tópicos para o registro do status de entrega de mensagens. Por exemplo, use a SetTopicAttributesAPI para configurar:

- `LambdaSuccessFeedbackRoleArn`— Função do IAM para entrega bem-sucedida de mensagens para endpoints Lambda.
- `LambdaSuccessFeedbackSampleRate`— Taxa de amostragem de mensagens bem-sucedidas para endpoints Lambda.
- `LambdaFailureFeedbackRoleArn`— Função do IAM para falha na entrega de mensagens para endpoints Lambda.

Exemplo de AWS CLI comando

```
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
```

Atributos de tópicos

Use os seguintes valores de nome de atributo de tópico para o status de entrega da mensagem:

HTTP

- `HTTPSuccessFeedbackRoleArn`— Status de entrega de mensagem bem-sucedida para um tópico do Amazon SNS que está inscrito em um endpoint HTTP.
- `HTTPSuccessFeedbackSampleRate`— Porcentagem de mensagens bem-sucedidas para amostra de um tópico do Amazon SNS que está inscrito em um endpoint HTTP.
- `HTTPFailureFeedbackRoleArn`— Falha no status de entrega de mensagens para um tópico do Amazon SNS que está inscrito em um endpoint HTTP.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— Status de entrega de mensagens bem-sucedido para um tópico do Amazon SNS que está inscrito em um endpoint do Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate`— Porcentagem de mensagens bem-sucedidas para amostra de um tópico do Amazon SNS que está inscrito em um endpoint do Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn`— Falha no status de entrega de mensagens para um tópico do Amazon SNS que está inscrito em um endpoint do Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— Status de entrega de mensagens bem-sucedido para um tópico do Amazon SNS que está inscrito em um endpoint Lambda.
- `LambdaSuccessFeedbackSampleRate`— Porcentagem de mensagens bem-sucedidas para amostra de um tópico do Amazon SNS que está inscrito em um endpoint Lambda.
- `LambdaFailureFeedbackRoleArn`— Falha no status de entrega de mensagens para um tópico do Amazon SNS que está inscrito em um endpoint Lambda.

Endpoints de aplicativos da plataforma

- `ApplicationSuccessFeedbackRoleArn`— Status de entrega de mensagens bem-sucedido para um tópico do Amazon SNS que está inscrito em um AWS endpoint do aplicativo.
- `ApplicationSuccessFeedbackSampleRate`— Porcentagem de mensagens bem-sucedidas para amostra de um tópico do Amazon SNS que está inscrito em um AWS endpoint de aplicativo.
- `ApplicationFailureFeedbackRoleArn`— Falha no status de entrega de mensagens para um tópico do Amazon SNS que está inscrito em um AWS endpoint do aplicativo.

> **Note**
>
> Além disso, você pode configurar os atributos do aplicativo para registrar o status de entrega diretamente nos serviços de notificação push. Para obter mais informações, consulte Usar atributos de aplicativo do Amazon SNS para obter o status de entrega de mensagens.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— Status de entrega de mensagem bem-sucedida para um tópico do Amazon SNS que está inscrito em um endpoint do Amazon SQS.
- `SQSSuccessFeedbackSampleRate`— Porcentagem de mensagens bem-sucedidas para amostra de um tópico do Amazon SNS que está inscrito em um endpoint do Amazon SQS.
- `SQSFailureFeedbackRoleArn`— Falha no status de entrega de mensagens para um tópico do Amazon SNS que está inscrito em um endpoint do Amazon SQS.

Os registros dos endpoints do aplicativo da plataforma são gravados no mesmo grupo de CloudWatch registros dos outros endpoints.

> **Note**
>
> Os `<ENDPOINT>FailureFeedbackRoleArn` atributos `<ENDPOINT>SuccessFeedbackRoleArn` e são usados para dar ao Amazon SNS acesso de gravação para usar CloudWatch Logs em seu nome. O atributo `<ENDPOINT>SuccessFeedbackSampleRate` é para especificar a porcentagem de taxa de amostra (0-100) de mensagens bem-sucedidas. Depois de configurar o `<ENDPOINT>FailureFeedbackRoleArn` atributo, todas as entregas de mensagens com falha geram CloudWatch registros.

AWS Exemplos de SDK para configurar atributos de tópicos

Os exemplos de código a seguir mostram como usar o `SetTopicAttributes`.

CLI

AWS CLI

Para definir um atributo para um tópico

O exemplo `set-topic-attributes` a seguir define o atributo `DisplayName` para o tópico especificado.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetTopicAttributesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte a SetTopicAttributesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para PHP da API.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a SetTopicAttributesreferência da API AWS SDK for SAP ABAP.

Configurar registro em log do status de entrega usando o AWS CloudFormation

Para configurar `DeliveryStatusLogging` usando AWS CloudFormation, use um modelo JSON ou YAML para criar uma AWS CloudFormation pilha. Para obter mais informações, consulte a `DeliveryStatusLogging` propriedade do `AWS::SNS::Topic` recurso no Guia do AWS CloudFormation usuário. Abaixo estão exemplos de AWS CloudFormation modelos em JSON e YAML para criar um novo tópico ou atualizar um tópico existente com todos os `DeliveryStatusLogging` atributos do protocolo Amazon SQS.

Garanta que os papéis do IAM referenciados em `SuccessFeedbackRoleArn` e `FailureFeedbackRoleArn` tenham as permissões de CloudWatch registros necessárias.

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
```

Novas tentativas de entrega de mensagens do Amazon SNS

O Amazon SNS define uma política de entrega para cada protocolo de entrega. A política de entrega define como o Amazon SNS tenta novamente a entrega de mensagens quando ocorrem erros do lado do servidor (quando o sistema que hospeda o endpoint inscrito se torna indisponível). Quando a política de entrega estiver esgotada, o Amazon SNS parará de tentar novamente a entrega e descartará a mensagem, a menos que uma fila de mensagens mortas esteja anexada à inscrição. Para obter mais informações, consulte Filas de mensagens não entregues do Amazon SNS.

Protocolos e políticas de entrega

> **Note**
>
> - Com exceção do HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S suporte a políticas personalizadas. Consulte Como criar uma política de entrega HTTP/S.
> - O Amazon SNS aplica instabilidade às novas tentativas de entrega. Para obter mais informações, consulte a publicação Recuo exponencial e instabilidade no Blog de arquitetura da AWS .
> - O tempo total de repetição da política para um endpoint HTTP/S não pode ser superior a 3,6 mil segundos. Esse limite é fixo e não pode ser alterado.

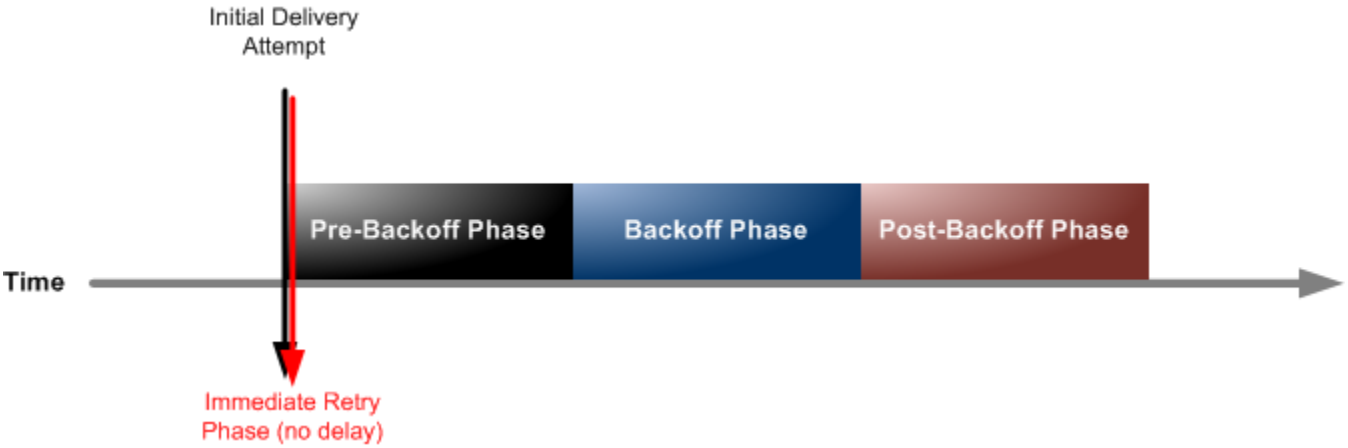
Tipo de endpoint	Protocolos de entrega	Fase de nova tentativa imediata (sem atraso)	Fase de pré-recuo	Fase de recuo	Fase de pós-recuo	Total de tentativas
AWS endpoints gerenciados	Amazon Data Firehose[1] AWS Lambda	3 vezes, sem atraso	2 vezes, 1 segundo de intervalo	10 vezes, com recuo exponencial, de 1 segundo	100.000 vezes, 20 segundos de intervalo	100.015 vezes, ao longo de 23 dias

Tipo de endpoint	Protocolos de entrega	Fase de nova tentativa imediata (sem atraso)	Fase de pré-recuo	Fase de recuo	Fase de pós-recuo	Total de tentativas
	Amazon SQS			a 20 segundos		
Endpoints gerenciados pelo cliente	SMTP SMS Push para dispositivos móveis	0 vezes, sem atraso	2 vezes, 10 segundos de intervalo	10 vezes, com recuo exponencial, de 10 segundos a 600 segundos (10 minutos)	38 vezes, 600 segundos (10 minutos) de intervalo	50 tentativas, mais de 6 horas

¹ Para erros de controle de utilização com o protocolo do Firehose, o Amazon SNS usa a mesma política de entrega empregada em endpoints gerenciados pelo cliente.

Estágios da política de entrega

O diagrama a seguir mostra as fases de uma política de entrega.



Cada política de entrega é composta por quatro fases.

1. Fase de nova tentativa imediata (sem demora) — Essa fase ocorre imediatamente após a tentativa inicial de entrega. Não há um intervalo entre novas tentativas nessa fase.
2. Fase de pré-recuo — Essa fase segue a fase de repetição imediata. Essa fase é usada pelo Amazon SNS para executar um conjunto de novas tentativas antes da aplicação de uma função de recuo. Essa fase especifica o número de novas tentativas e a quantidade de atraso entre elas.
3. Fase de recuo — Essa fase controla o atraso entre as novas tentativas usando a função retry-backoff. Essa fase define um atraso mínimo, um atraso máximo e uma função de recuo de nova tentativa que define a rapidez com que o atraso aumenta do atraso mínimo para o máximo. A função de recuo pode ser aritmética, exponencial, geométrica ou linear.
4. Fase pós-recuo — Esta fase segue a fase de recuo. Ela especifica um número de novas tentativas e a quantidade de atraso entre elas. Esta é a fase final.

Como criar uma política de entrega HTTP/S

Você pode definir como o Amazon SNS tenta novamente a entrega de mensagens para endpoints HTTP/S usando uma política de entrega com quatro fases: sem atraso, pré-recuo, recuo e pós-recuo. Essa política permite que você substitua as configurações padrão de repetição e as personalize de acordo com a capacidade do seu servidor HTTP.

Você pode definir sua política de entrega HTTP/S como um objeto JSON no nível do tópico ou da assinatura:

- Política em nível de tópico — aplica-se a todas as assinaturas HTTP/S vinculadas ao tópico. Use a ação `CreateTopic`ou `SetTopicAttributes`API para definir essa política.
- Política em nível de assinatura — aplica-se somente a uma assinatura específica. Use a ação `Subscribe`ou `SetSubscriptionAttributes`API para definir essa política.

Como alternativa, você também pode usar o AWS::SNS::Subscriptionrecurso em seus AWS CloudFormation modelos.

Você deve personalizar sua política de entrega com base na capacidade do seu servidor HTTP/S:

- Servidor único para todas as assinaturas — Se todas as assinaturas HTTP/S em um tópico usarem o mesmo servidor, defina a política de entrega como um atributo do tópico para garantir a consistência em todas as assinaturas.

- Servidores diferentes para assinaturas — Se as assinaturas tiverem como alvo servidores diferentes, crie uma política de entrega exclusiva para cada assinatura, adaptada à capacidade do servidor específico.

Você também pode definir o `Content-Type` cabeçalho na política de solicitação para especificar o tipo de mídia da notificação. Por padrão, o Amazon SNS envia todas as notificações para endpoints HTTP/S com o tipo de conteúdo definido como. `text/plain; charset=UTF-8` No entanto, você pode substituir esse padrão usando o `headerContentType`campo na política de solicitação.

O objeto JSON a seguir define uma política de entrega com novas tentativas estruturadas em quatro fases:

1. Fase sem demora — tente novamente 3 vezes imediatamente.
2. Fase de pré-recuo — tente novamente 2 vezes com um intervalo de 1 segundo.
3. Fase de recuo — tente novamente 10 vezes com atrasos exponenciais que variam de 1 a 60 segundos.
4. Fase pós-recuo — Tente novamente 35 vezes com um intervalo fixo de 60 segundos.

O Amazon SNS faz um total de 50 tentativas para entregar uma mensagem antes de descartá-la. Para reter as mensagens que não podem ser entregues após todas as novas tentativas, configure sua assinatura para mover as mensagens não entregues para uma fila de mensagens mortas (DLQ). Para obter mais informações, consulte Filas de mensagens não entregues do Amazon SNS.

O Amazon SNS considera todos os erros 5XX e 429 (muitas solicitações enviadas) como passíveis de repetição. Esses erros estão sujeitos à política de entrega. Todos os outros erros são considerados falhas permanentes e não serão feitas novas tentativas.

> **Note**
>
> Essa política de entrega usa a `maxReceivesPerSecond` propriedade para reduzir o tráfego de entrega para uma média de 10 mensagens por segundo por assinatura. Embora esse mecanismo ajude a evitar que seu endpoint HTTP/S fique sobrecarregado pelo alto tráfego, ele foi projetado para manter uma taxa média de entrega e não impõe um limite rígido. Podem ocorrer picos ocasionais de tráfego de entrega acima do limite especificado, especialmente se sua taxa de publicação for significativamente maior do que o limite de limitação.

Quando o tráfego de publicação (entrada) excede a taxa de entrega (saída), isso pode resultar em um acúmulo de mensagens e maior latência de entrega. Para evitar esses problemas, certifique-se de que o `maxReceivesPerSecond` valor esteja alinhado com os requisitos de capacidade e carga de trabalho do seu servidor HTTP/S.

O exemplo de política de entrega a seguir substitui o tipo de conteúdo padrão para notificação HTTP/S. `application/json`

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

A política de entrega é composta por uma política de repetição, uma política de aceleração e uma política de solicitação. No total, há 9 atributos em uma política de entrega.

Política	Descrição	Restrição
`minDelayTarget`	O atraso mínimo para uma nova tentativa. Unidade: segundos	1 a atraso máximo Padrão: 20
`maxDelayTarget`	O atraso máximo para uma nova tentativa.	Atraso mínimo a 3.600 Padrão: 20

Política	Descrição	Restrição
	Unidade: segundos	
`numRetries`	O número total de novas tentativas, incluindo repetições imediatas, pré-recuo, recuo e pós-recuo.	0 a 100 Padrão: 3
`numNoDelayRetries`	O número de novas tentativas a serem feitas imediatamente, sem atraso entre elas.	0 ou mais Padrão: 0
`numMinDelayRetries`	O número de tentativas na fase de pré-recuo, com o atraso mínimo especificado entre elas.	0 ou mais Padrão: 0
`numMaxDelayRetries`	O número de novas tentativa s na fase pós-recuo, com o atraso máximo entre elas.	0 ou mais Padrão: 0
`backoffFunction`	O modelo para recuo entre novas tentativas.	Uma das quatro opções: • aritmética • exponencial • geométrica • linear Padrão: linear
`maxReceivesPerSecond`	O número médio máximo de entregas de mensagens por segundo, por assinatura.	1 ou mais Padrão: Sem limitação (sem limite na taxa de entrega)

Política	Descrição	Restrição
`headerContentType`	O tipo de conteúdo da notificação que está sendo enviada aos endpoints HTTP/S.	Se a política de solicitação não estiver definida, o tipo de conteúdo usará `text/plain; charset=UTF-8` como padrão. Quando a entrega de mensagens brutas é desativada para uma assinatura (padrão), ou quando a política de entrega é definida no nível do tópico, os tipos de conteúdo de cabeçalho compatíveis são `application/json` e `text/plain`. Quando a entrega de mensagens brutas é ativada para uma assinatura, os seguintes tipos de conteúdo são compatíveis: • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • aplicação/ x-www-form-urlencoded

Política	Descrição	Restrição
		• application/xhtml+xml • application/xml

O Amazon SNS usa a seguinte fórmula para calcular o número de novas tentativas na fase de recuo:

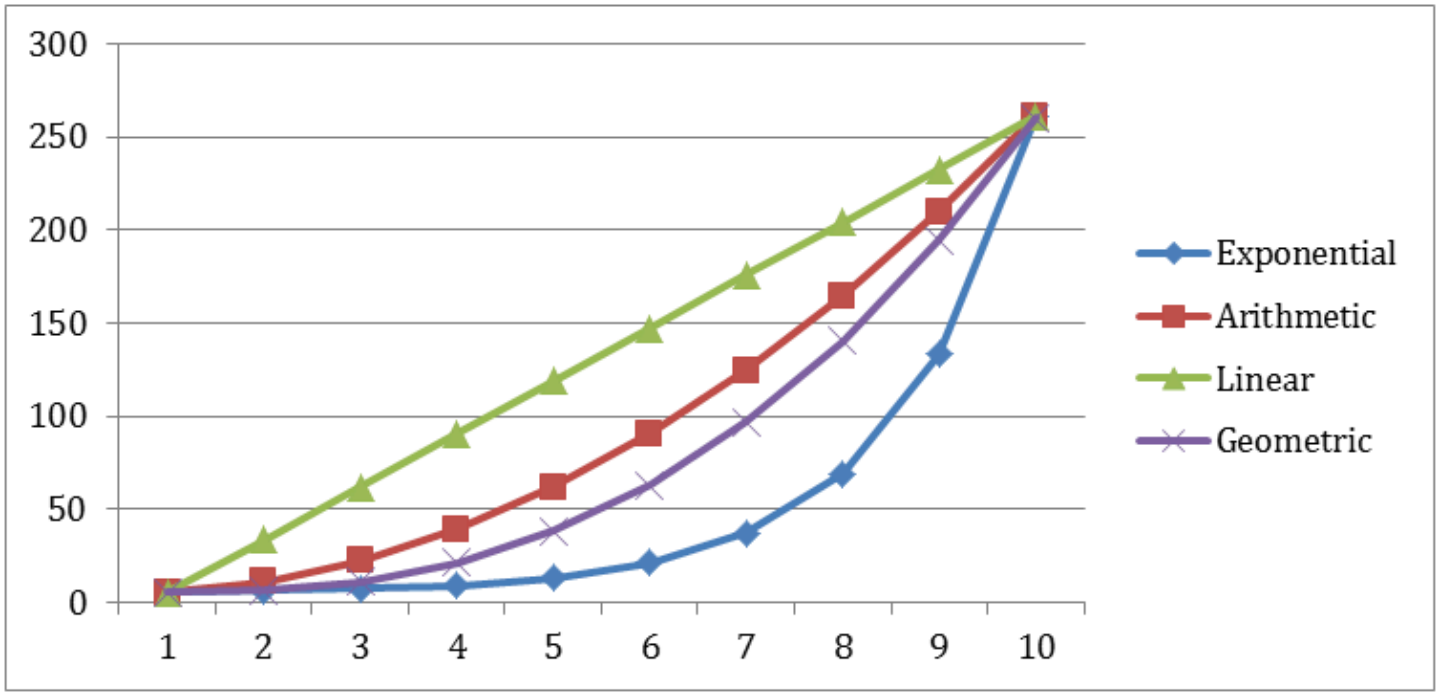
```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

Você pode controlar a frequência de novas tentativas durante a fase de recuo usando três parâmetros:

- **`minDelayTarget`**— Define o atraso para a primeira tentativa de repetição na fase de recuo.
- **`maxDelayTarget`**— Define o atraso para a tentativa final de nova tentativa na fase de recuo.
- **`backoffFunction`**— Determina o algoritmo que o Amazon SNS usa para calcular os atrasos de todas as tentativas de repetição entre a primeira e a última tentativa. Você pode escolher entre quatro funções de retry-backoff disponíveis.

O diagrama a seguir ilustra como as diferentes funções de recuo de repetição afetam os atrasos entre as novas tentativas durante a fase de recuo. A política de entrega usada neste exemplo inclui as seguintes configurações: total de 10 tentativas, um atraso mínimo de 5 segundos e um atraso máximo de 260 segundos.

- O eixo vertical mostra o atraso (em segundos) para cada tentativa de nova tentativa.
- O eixo horizontal representa a sequência de repetição, variando da primeira à décima tentativa.



Filas de mensagens não entregues do Amazon SNS

Uma fila de mensagens mortas é uma fila do Amazon SQS para a qual uma assinatura do Amazon SNS pode enviar mensagens que não podem ser entregues aos assinantes com êxito. As mensagens que não podem ser entregues devido a erros do cliente ou erros do servidor são mantidas na fila de mensagens mortas para análise ou reprocessamento adicionais. Para ter mais informações, consulte Configurar uma fila de mensagens não entregues do Amazon SNS para uma assinatura e Novas tentativas de entrega de mensagens do Amazon SNS.

> **Note**
>
> - A assinatura do Amazon SNS e a fila do Amazon SQS devem estar na mesma conta e região AWS .
> - Em um tópico FIFO, você pode usar uma fila do Amazon SQS como uma fila de mensagens não entregues para a assinatura do Amazon SNS. As assinaturas de tópicos FIFO usam filas FIFO e as de tópicos padrão usam filas padrão.
> - Para usar uma fila criptografada do Amazon SQS como fila de mensagens sem saída, você deve usar um KMS personalizado com uma política de chaves que conceda ao serviço Amazon SNS acesso principal às ações da API. AWS KMS Para obter mais informações, consulte Segurança dos dados do Amazon SNS com a criptografia do lado

do servidor neste guia e Proteção de dados do Amazon SQS usando criptografia do lado do servidor (SSE) e AWS KMS no Guia do desenvolvedor do Amazon Simple Queue Service.

Por que há falha nas entregas de mensagens?

Em geral, há falha na entrega de mensagens quando o Amazon SNS não pode acessar um endpoint inscrito devido a um erro do lado do cliente ou um erro do lado do servidor. Quando o Amazon SNS recebe um erro do lado do cliente ou continua a receber um erro do lado do servidor para uma mensagem além do número de tentativas especificado pela política de novas tentativas correspondente, o Amazon SNS descarta a mensagem, a menos que uma fila de mensagens mortas esteja anexada à inscrição. Entregas com falha não alteram o status de suas inscrições. Para obter mais informações, consulte Novas tentativas de entrega de mensagens do Amazon SNS.

Erros no lado do cliente,

Erros do lado do cliente podem ocorrer quando o Amazon SNS tem metadados de inscrição obsoletos. Esses erros geralmente ocorrem quando um proprietário exclui o endpoint (por exemplo, uma função do Lambda inscrita em um tópico do Amazon SNS) ou quando um proprietário altera a política anexada ao endpoint inscrito de uma forma que impede que o Amazon SNS entregue as mensagens para o ponto de endpoint. O Amazon SNS não tenta novamente a entrega da mensagem que falha como resultado de um erro do lado do cliente.

Erros no lado do servidor

Erros do lado do servidor podem ocorrer quando o sistema responsável pelo endpoint inscrito se torna indisponível ou retorna uma exceção que indica que ele não pode processar uma solicitação válida do Amazon SNS. Quando ocorrem erros do lado do servidor, o Amazon SNS tenta novamente as entregas com falha usando uma função de recuo exponencial ou linear. Para erros do lado do servidor causados por endpoints AWS gerenciados apoiados pelo Amazon SQS ou, o AWS Lambda Amazon SNS tenta novamente a entrega até 100.015 vezes, em 23 dias.

Os endpoints gerenciados pelo cliente (como HTTP, SMTP, SMS ou push em dispositivos móveis) também podem causar erros no lado do servidor. O Amazon SNS tenta novamente a entrega para esses tipos de endpoints também. Enquanto os endpoints HTTP oferecem suporte a políticas de novas tentativas definidas pelo cliente, o Amazon SNS define uma política de novas tentativas de entrega interna como 50 vezes ao longo de 6 horas, para SMTP, SMS e endpoints de push para dispositivos móveis.

Como as filas de mensagens não entregues funcionam?

Uma fila de mensagens não entregues é anexada a uma inscrição do Amazon SNS (em vez de um tópico) porque as entregas de mensagens acontecem no nível de inscrição. Isso permite identificar o endpoint de destino original para cada mensagem com mais facilidade.

Uma fila de mensagens mortas associada a uma inscrição do Amazon SNS é uma fila comum do Amazon SQS. Para obter mais informações sobre o período de retenção de mensagens, consulte Quotas Related to Messages (“Cotas relacionadas a mensagens”) no Guia do desenvolvedor do Amazon Simple Queue Service. É possível alterar o período de retenção de mensagens usando a ação da API `SetQueueAttributes` do Amazon SQS. Para tornar seus aplicativos mais resilientes, recomendamos definir o período máximo de retenção para filas de mensagens não entregues como 14 dias.

Como as mensagens são movidas para uma fila de mensagens não entregues?

As mensagens são movidas para uma fila de mensagens não entregues por meio de uma política de redirecionamento. Uma política de redirecionamento é um objeto JSON que se refere ao ARN da fila de mensagens não entregues. O atributo `deadLetterTargetArn` especifica o ARN. O ARN deve apontar para uma fila do Amazon SQS na Conta da AWS mesma região da sua assinatura do Amazon SNS. Para obter mais informações, consulte Configurar uma fila de mensagens não entregues do Amazon SNS para uma assinatura.

O objeto JSON a seguir é uma política de redirecionamento de exemplo, anexada a uma assinatura SNS.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Como posso mover mensagens de uma fila de mensagens não entregues?

É possível mover mensagens para fora de uma fila de mensagens não entregues de duas maneiras:

- Evitar escrever lógica do consumidor do Amazon SQS: defina sua fila de mensagens mortas como uma fonte de evento para a função do Lambda drenar sua fila de mensagens mortas.

- Escreva a lógica de consumo do Amazon SQS — Use a API AWS , o SDK do Amazon SQS AWS CLI ou para escrever uma lógica de consumidor personalizada para pesquisar, processar e excluir as mensagens na fila de mensagens mortas.

Como posso monitorar e registrar em log filas de mensagens não entregues?

Você pode usar CloudWatch as métricas da Amazon para monitorar filas de cartas mortas associadas às suas assinaturas do Amazon SNS. Todas as filas do Amazon SQS emitem CloudWatch métricas em intervalos de um minuto. Para obter mais informações, consulte CloudWatch Métricas disponíveis para o Amazon SQS no Guia do desenvolvedor do Amazon Simple Queue Service. Todas as assinaturas do Amazon SNS com filas de mensagens sem saída também emitem métricas. CloudWatch Para obter mais informações, consulte Monitorando tópicos do Amazon SNS usando CloudWatch.

Para ser notificado sobre atividades em suas filas de cartas mortas, você pode usar CloudWatch métricas e alarmes. Configurar um alarme para a métrica `NumberOfMessagesSent` não é adequado porque essa métrica não captura mensagens enviadas para uma DLQ como resultado de tentativas de processamento malsucedidas. Em vez disso, use a métrica `ApproximateNumberOfMessagesVisible`, que captura todas as mensagens atualmente disponíveis no DLQ, incluindo aquelas movidas devido a falhas de processamento.

Exemplo de configuração CloudWatch de alarme

1. Crie um CloudWatchalarme para a **`ApproximateNumberOfMessagesVisible`** métrica.
2. Defina o limite de alarme como 1 (ou outro valor apropriado com base em suas expectativas e tráfego de DLQ).
3. Especifique um tópico a ser notificado quando o alarme desligar. Esse tópico do Amazon SNS pode entregar sua notificação de alarme a qualquer tipo de ponto de endpoint (como um endereço de e-mail, número de telefone ou aplicativo de pager móvel).

Você pode usar o CloudWatch Logs para investigar as exceções que fazem com que qualquer entrega do Amazon SNS falhe e para que as mensagens sejam enviadas para filas de cartas mortas. O Amazon SNS pode registrar entregas bem-sucedidas e malsucedidas. CloudWatch Para obter mais informações, consulte Atributos de aplicativo móvel do Amazon SNS.

Configurar uma fila de mensagens não entregues do Amazon SNS para uma assinatura

Uma fila de mensagens mortas é uma fila do Amazon SQS para a qual uma assinatura do Amazon SNS pode enviar mensagens que não podem ser entregues aos assinantes com êxito. As mensagens que não podem ser entregues devido a erros do cliente ou erros do servidor são mantidas na fila de mensagens mortas para análise ou reprocessamento adicionais. Para ter mais informações, consulte Filas de mensagens não entregues do Amazon SNS e Novas tentativas de entrega de mensagens do Amazon SNS.

Esta página mostra como você pode usar o AWS Management Console, um AWS SDK AWS CLI, o e AWS CloudFormation para configurar uma fila de mensagens mortas para uma assinatura do Amazon SNS.

> **Note**
>
> Em um tópico FIFO, você pode usar uma fila do Amazon SQS como uma fila de mensagens não entregues para a assinatura do Amazon SNS. As assinaturas de tópicos FIFO usam filas FIFO e as de tópicos padrão usam filas padrão.

Pré-requisitos

Antes de configurar uma fila de mensagens não entregues, complete os seguintes pré-requisitos:

1. Crie um tópico do Amazon SNS chamado `MyTopic`.
2. Crie uma fila do Amazon SQS chamada `MyEndpoint`, para ser usada como endpoint para a assinatura do Amazon SNS.
3. (Ignorar para AWS CloudFormation) Inscreva-se na fila do tópico.
4. Crie outra fila do Amazon SQS chamada `MyDeadLetterQueue`, para ser usada como a fila de mensagens mortas para a assinatura do Amazon SNS.
5. Para conceder à entidade principal do Amazon SNS acesso à ação da API do Amazon SQS, defina a seguinte política de fila para `MyDeadLetterQueue`.

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

Para configurar uma fila de mensagens mortas para uma assinatura do Amazon SNS usando o AWS Management Console

Verifique os pré-requisitos antes de começar este tutorial.

1. Faça login no console do Amazon SQS.
2. Crie uma fila do Amazon SQS ou use uma fila existente e anote o ARN da fila na guia Detalhes da fila, por exemplo:

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

3. Faça login no console do Amazon SNS.
4. No painel de navegação, escolha Subscriptions (Assinaturas).
5. Na página Assinaturas selecione uma inscrição existente e escolha Editar.
6. Na *1234a567-bc89-012d-3e45-6fg7h890123i* página Editar, expanda a seção Política do Redrive (fila de mensagens mortas) e faça o seguinte:

   a. Selecione Ativado.

   b. Especifique o ARN de uma fila do Amazon SQS.

7. Escolha Salvar alterações.

   A inscrição estará configurada para usar uma fila de mensagens não entregues.

Para configurar uma fila de mensagens mortas para uma assinatura do Amazon SNS usando um SDK AWS

Antes de executar este exemplo, verifique se você preencheu os pré-requisitos.

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

O código de exemplo a seguir mostra como usar `SetSubscriptionAttributesRedrivePolicy`.

Java

SDK para Java 1.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Para configurar uma fila de mensagens mortas para uma assinatura do Amazon SNS usando o AWS CLI

Verifique os pré-requisitos antes de começar este tutorial.

1. Instale e configure a AWS CLI. Para obter mais informações, consulte o Guia do usuário do AWS Command Line Interface .
2. Use o seguinte comando.

```
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\"}"
```

Para configurar uma fila de mensagens mortas para uma assinatura do Amazon SNS usando AWS CloudFormation

Verifique os pré-requisitos antes de começar este tutorial.

1. Copie o seguinte código JSON em um arquivo denominado `MyDeadLetterQueue.json`.

```
{
  "Resources": {
    "mySubscription": {
      "Type" : "AWS::SNS::Subscription",
      "Properties" : {
        "Protocol": "sqs",
        "Endpoint": "arn:aws:sqs:us-east-2:123456789012:MyEndpoint",
        "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
        "RedrivePolicy": {
          "deadLetterTargetArn":
            "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
        }
      }
    }
  }
}
```

2. Faça login no console do AWS CloudFormation.
3. Na página Selecionar modelo, selecione Fazer upload de um modelo no Amazon S3, selecione o arquivo `MyDeadLetterQueue.json` e escolha Próximo.
4. Na página Especificar detalhes, digite `MyDeadLetterQueue` em Nome da pilha e escolha Próximo.
5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Revisar, escolha Criar.

   AWS CloudFormation começa a criar a `MyDeadLetterQueue` pilha e exibe o status CREATE_IN_PROGRESS. Quando o processo estiver concluído, AWS CloudFormation exibirá o status CREATE_COMPLETE.

Arquivamento, reprodução e análise de mensagens do Amazon SNS

Os tópicos padrão do Amazon SNS comportam arquivamento e análise de mensagens por meio do Amazon Data Firehose. Você pode divulgar notificações aos fluxos de entrega do Firehose, o que permite enviar notificações a destinos de armazenamento e análise compatíveis com o Firehose, incluindo Amazon Simple Storage Service (Amazon S3), Amazon Redshift e muito mais.

Os tópicos FIFO do Amazon SNS comportam um arquivamento de mensagens local, sem código, que permite aos proprietários de tópicos armazenar (ou arquivar) mensagens publicadas em um tópico por até 365 dias. No caso de tópicos com uma `ArchivePolicy` ativa, os assinantes podem então criar uma `ReplayPolicy` para recuperar (ou reproduzir) as mensagens arquivadas em um endpoint inscrito. Para saber mais sobre esse atributo, consulte Arquivamento e reprodução de mensagens do Amazon SNS de tópicos FIFO.

Atributos	Tópicos padrão	Tópicos FIFO
Arquivamento de mensagens	Fanout de fluxos de entrega do Firehose	Arquivamento de mensagens do Amazon SNS para proprietários de tópicos FIFO
Reprodução de mensagens	A reprodução de tópicos padrão não é um recurso incorporado. Muitos clientes criam o próprio recurso com base no arquivo de mensagens.	Reprodução de mensagens do Amazon SNS para assinantes de tópicos FIFO

Gerenciamento e otimização de recursos no Amazon SNS

Este tópico fornece orientação sobre como aproveitar todo o potencial do Amazon SNS garantindo um desempenho ideal, reduzindo custos desnecessários e mantendo recursos bem organizados.

Tópicos

Marcação de tópicos do Amazon SNS

O Amazon SNS oferece suporte à marcação de tópicos do Amazon SNS. Isso pode ajudá-lo a monitorar e gerenciar os custos associados aos seus tópicos, fornecer segurança aprimorada em suas políticas de AWS Identity and Access Management (IAM) e permitir que você pesquise ou filtre facilmente milhares de tópicos. A marcação permite que você gerencie seus tópicos do Amazon SNS usando Resource AWS Groups. Para obter mais informações sobre o Resource Groups, consulte o Guia do usuário do AWS Resource Groups.

Marcação para alocação de custos

Para organizar e identificar seus tópicos do Amazon SNS para alocação de custos, você pode adicionar etiquetas que identificam o objetivo de um tópico. Isso é especialmente útil quando você tem vários tópicos. Você pode usar etiquetas de alocação de custos para organizar sua AWS fatura de forma a refletir sua própria estrutura de custos. Para fazer isso, inscreva-se para receber a fatura AWS da sua conta e incluir as chaves e valores da tag. Para obter mais informações, consulte Configurar um relatório de alocação de custos mensal, no Guia do usuário do AWS Billing and Cost Management.

Por exemplo, é possível adicionar etiquetas que representam o centro de custos e o objetivo dos seus tópicos do Amazon SNS, da seguinte maneira.

Recurso	Chave	Valor
Tópico 1	Centro de custos	43289
	Aplicação	Processamento de pedidos
Tópico 2	Centro de custos	43289

Recurso	Chave	Valor
	Aplicação	Processamento de pagamentos
Tópico 3	Centro de custos	76585
	Aplicação	Arquivamento

Esse esquema de marcação permite que você agrupe dois tópicos executando tarefas relacionadas no mesmo centro de custos e, ao mesmo tempo, marcar uma atividade não relacionada com outra etiqueta de alocação de custos.

Marcação para controle de acesso

AWS Identity and Access Management suporta o controle do acesso a recursos com base em tags. Após a marcação dos seus recursos, forneça informações sobre as etiquetas de recurso no elemento de condição de uma política do IAM para gerenciar o acesso baseado em etiquetas. Para obter informações sobre como marcar seus recursos usando o console do Amazon SNS ou o AWS SDK, consulte Configurar etiquetas.

É possível restringir o acesso de uma identidade do IAM. Por exemplo, você pode restringir o acesso de `Publish` e `PublishBatch` a todos os tópicos do Amazon SNS que incluem uma etiqueta com a chave `environment` e o valor `production`, enquanto permite acesso a todos os outros tópicos do Amazon SNS. No exemplo abaixo, a política restringe a capacidade de publicar mensagens em tópicos marcados com`production`, enquanto permite que mensagens sejam publicadas em tópicos marcados com `development`. Para obter mais informações, consulte Controlar o acesso usando etiquetas, no Guia do usuário do IAM.

> **Note**
>
> Configurar a permissão do IAM para `Publish` define a permissão para `Publish` e `PublishBatch`.

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

```
    "Action": [
 "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
}
```

Marcação para pesquisa e filtragem de recursos

Uma AWS conta pode ter dezenas de milhares de tópicos do Amazon SNS (consulte Cotas do Amazon SNS para obter detalhes). Por meio da marcação de tópicos, é possível simplificar o processo de pesquisa ou filtragem de tópicos.

Por exemplo, você pode ter centenas de tópicos que estão associados ao seu ambiente de produção. Em vez de ter que pesquisar manualmente esses tópicos, você pode consultar todos eles com uma determinada etiqueta:

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

Configurar etiquetas de tópicos do Amazon SNS

Este tópico explica como configurar tags para um tópico do Amazon SNS usando o AWS Management Console, um AWS SDK ou o. AWS CLI

> **Important**
>
> Não adicione informações de identificação pessoal (PII) nem outras informações confidenciais ou sigilosas em tags. As etiquetas são acessíveis a outros Amazon Web Services, incluindo o faturamento. As tags não devem ser usadas para dados privados ou confidenciais.

Listar, adicionar e remover tags para um tópico do Amazon SNS usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Na página Topics (Tópicos), escolha um tópico e selecione Edit (Editar).

4. Expanda a seção Tags.

   As tags adicionadas ao tópico serão listadas.

5. Modifique as tags do tópico:

   - Para adicionar uma etiqueta, escolha Add tag (Adicionar etiqueta) e insira uma Key (Chave) e um Value (Valor) (opcional).
   - Para remover uma tag, escolha Remove tag (Remover tag) ao lado de um par chave/valor.

6. Escolha Salvar alterações.

Adicionar etiquetas a um tópico usando um AWS SDK

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Para adicionar uma tag a um tópico

O exemplo `tag-resource` a seguir adiciona uma tag de metadados ao tópico do Amazon SNS especificado.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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

                Usage:    <topicArn>

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

                """;

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

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

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

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

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

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

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

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

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

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

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

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

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

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

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for Kotlin.

Gerenciar etiquetas com ações de API do Amazon SNS

Para gerenciar etiquetas usando a API do Amazon SNS, use as seguintes ações de API:

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

Ações de API com suporte para o ABAC

A lista a seguir inclui as ações de API que oferecem suporte ao controle de acesso baseado em atributos (ABAC). Para obter mais detalhes sobre o ABAC, consulte Para que serve o ABAC? AWS no Guia do usuário do IAM.

- `AddPermission`
- `ConfirmSubscription`
- `DeleteTopic`
- `GetDataProtectionPolicy`
- `GetSubscriptionAttributes`
- `GetTopicAttributes`
- `ListSubscriptionsByTopic`
- `ListTagsForResource`
- `Publish`
- `PublishBatch`
- `PutDataProtectionPolicy`
- `RemovePermission`
- `SetSubscriptionAttributes`
- `SetTopicAttributes`
- `Subscribe`
- `TagResource`

- Unsubscribe
- UntagResource

Origens e destinos de eventos do Amazon SNS

O Amazon SNS conecta Serviços da AWS sistemas externos ao rotear notificações orientadas por eventos. O Amazon SNS recebe eventos de vários Serviços da AWS, como atualizações de pipeline de dados, ações de EC2 escalabilidade da Amazon ou alertas de segurança, e publica esses eventos em tópicos do Amazon SNS. Esses tópicos então enviam notificações para destinos designados.

O Amazon SNS oferece suporte a dois tipos principais de destinos: Application-to-Application (A2A) e Application-to-Person (A2P). Nas mensagens A2A, o Amazon SNS pode enviar eventos para o Lambda para acionar uma lógica de negócios personalizada, para o Amazon SQS para enfileirar mensagens e para o Amazon Kinesis Data Firehose para transmitir dados para serviços de armazenamento e análise. Para mensagens A2P, o Amazon SNS pode enviar notificações via SMS, e-mail e notificações push para dispositivos móveis, garantindo que os usuários ou equipes recebam alertas oportunos.

Ao atuar como um hub central, o Amazon SNS encaminha as notificações para os lugares certos, ajudando você a automatizar e gerenciar sua AWS infraestrutura com mais eficiência. Essa configuração permite a integração perfeita entre os serviços e a comunicação confiável com usuários e sistemas.

Fontes de eventos do Amazon SNS

O Amazon SNS se integra a uma ampla variedade de várias Serviços da AWS categorias, permitindo que esses serviços publiquem eventos em tópicos do Amazon SNS. Essa integração fornece notificações em tempo real dos principais eventos, como mudanças na infraestrutura, no desempenho do aplicativo e no gerenciamento de custos.

> **Note**
>
> O Amazon SNS introduziu os tópicos FIFO em outubro de 2020. Atualmente, a maioria dos AWS serviços oferece suporte ao envio de eventos somente para tópicos padrão.

Serviços de análise

A tabela a seguir descreve como o Amazon SNS se integra a serviços de AWS análise como Athena e AWS Data Pipeline Amazon Redshift para fornecer notificações em tempo real para

eventos importantes, incluindo violações de limites de controle, atualizações de status de pipeline e atividades de data warehouse.

Você pode aproveitar essas integrações para automatizar as respostas e manter uma supervisão eficaz de suas operações de dados.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon Athena: permite analisar dados no Amazon S3 utilizando o SQL padrão.	Receba notificações se os limites de controle forem excedidos. Para obter mais informaçõ es, consulte Setting Data Usage Control Limits (“Definir limites de controle de uso de dados”) no Manual do usuário do Amazon Athena.
AWS Data Pipeline: ajuda a automatizar a movimentação e a transformação de dados.	Receba notificações sobre o status dos componentes do pipeline. Para obter mais informações, consulte SnsAlarm no Guia do desenvolvedor do AWS Data Pipeline .
Amazon Redshift: gerencia todo o trabalho de configuração, operação e escalabilidade de um data warehouse.	Receba notificações de eventos do Amazon Redshift. Para obter mais informações, consulte Notificações de eventos do Amazon Redshift no Guia de gerenciamento do Amazon Redshift.

Serviços de integração de aplicações

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de integração de aplicativos, como EventBridge e AWS Step Functions, permitindo o roteamento e as notificações de dados em tempo real para aplicativos essenciais aos negócios.

Você pode aproveitar essas integrações para receber alertas de EventBridge eventos e orquestrar fluxos de trabalho usando Step Functions, aprimorando a automação e a capacidade de resposta de seus aplicativos.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon EventBridge — entrega um fluxo de dados em tempo real de seus próprios aplicativos, aplicativos software-as-a-service (SaaS) e AWS serviços e encaminha esses dados para destinos, incluindo o Amazon SNS. EventBridge era anteriormente chamado de CloudWatch Eventos.	Receba notificações de EventBridge eventos. Para obter mais informações, consulte as EventBridgemetas da Amazon no Guia EventBridge do usuário da Amazon.
AWS Step Functions— Permite combinar AWS Lambda funções e outros AWS serviços para criar aplicativos essenciais para os negócios.	Receba notificação de eventos do Step Functions. Para obter mais informações, consulte Call Amazon SNS with Step Functions (“Chamar o Amazon SNS com Step Functions”) no Guia do desenvolvedor do AWS Step Functions .

Serviços de faturamento e gerenciamento de custos

A tabela a seguir descreve como Gerenciamento de Faturamento e Custos da AWS se integra ao Amazon SNS para fornecer notificações sobre orçamentos, alterações de preços e anomalias de custo.

Você pode aproveitar essa integração para configurar tópicos do Amazon SNS para receber alertas em tempo real sobre AWS seus gastos, ajudando você a monitorar custos e responder a cobranças inesperadas com eficiência.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Gerenciamento de Faturamento e Custos da AWS: fornece recursos que ajudam a monitorar os custos e pagar sua fatura.	Receba notificações sobre orçamento e alteração de preços, além de alertas sobre anomalias. Para obter mais informações, consulte as seguintes páginas do Manual do usuário do AWS Billing : •

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
	Criar um tópico do Amazon SNS para notificações de orçamento • Configuração de notificações • Detectando gastos incomuns com AWS Cost Anomaly Detection

Serviços de aplicações de negócios

A tabela a seguir descreve como o Amazon Chime se integra ao Amazon SNS para enviar notificações de eventos importantes de reuniões, permitindo que você se mantenha informado sobre suas comunicações e agendamento.

Você pode aproveitar essa integração para utilizar as notificações de eventos do Amazon Chime SDK para aprimorar suas ferramentas de colaboração dentro e fora da sua organização.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon Chime: permite conhecer, conversar e fazer chamadas de negócios dentro e fora da sua organização.	Receba notificações sobre eventos de reunião importantes. Para obter mais informações, consulte Amazon Chime SDK event notifications (“Notificações de eventos do SDK do Amazon Chime”) no Guia do desenvolvedor do Amazon Chime.

Serviços de computação

A tabela a seguir descreve como o Amazon SNS se integra a vários serviços de AWS computação, permitindo que você receba notificações sobre eventos importantes, como ações do Auto Scaling, conclusões do Image EC2 Builder, mudanças no ambiente do Elastic Beanstalk, saídas da função Lambda e limites métricos do Lightsail.

Você pode aproveitar essas integrações para gerenciar com eficiência seus aplicativos e recursos, mantendo-se informado sobre atualizações e ações críticas em todos os Serviços da AWS.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon EC2 Auto Scaling — Ajuda você a ter o número correto de instâncias do Amazon Elastic Compute Cloud (Amazon EC2) disponíveis para lidar com a carga do seu aplicativo.	Receba notificações quando o Auto Scaling iniciar ou encerrar instâncias da EC2 Amazon em seu grupo de Auto Scaling. Para obter mais informações, consulte Receber notificações do Amazon SNS quando seu grupo de Auto Scaling escalar no Guia do usuário do Amazon Auto EC2 Scaling.
EC2 Image Builder — ajuda a automatizar a criação, o gerenciamento e a implantação de imagens personalizadas, seguras e de up-to-date servidor pré-instaladas e pré-configuradas com software e configurações para atender a padrões específicos de TI.	Receba notificações quando as compilaçõ es estiverem concluídas. Para obter mais informações, consulte Rastreamento das imagens mais recentes do servidor nos pipelines do EC2 Image Builder no blog AWS Compute.
AWS Elastic Beanstalk: gerencia os detalhes de provisionamento de capacidade, balanceam ento de carga, escalabilidade e monitoramento do integridade da aplicação.	Receba notificações sobre eventos important es que afetam sua aplicação. Para obter mais informações, consulte Notificações do ambiente do Elastic Beanstalk com o Amazon SNS no Guia do desenvolvedor do AWS Elastic Beanstalk .
AWS Lambda: permite executar código sem provisionar ou gerenciar servidores.	Receba dados de saída de funções definindo um tópico do SNS como uma fila de mensagens mortas do Lambda ou um destino do Lambda. Para obter mais informações, consulte Invocação assíncronano Guia do desenvolvedor do AWS Lambda .
Amazon Lightsail — ajuda os desenvolvedores a começar a AWS usar para criar sites ou aplicativos web.	Receba notificações quando uma métrica de uma de suas instâncias, bancos de dados ou balanceadores de carga ultrapassar um limite especificado. Para obter mais informações,

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
	consulte Adicionar contatos de notificação no Amazon Lightsail no Guia do desenvlvedor do Amazon Lightsail.

Serviços de contêineres

A tabela a seguir descreve como o Amazon SNS se integra a serviços de AWS contêineres, como o Amazon EKS Distro e o Amazon ECS, permitindo que você acompanhe atualizações e patches de segurança para clusters do Amazon EKS e receba notificações sobre novas versões de AMI otimizadas para ECS.

Você pode aproveitar essas integrações para manter a segurança e a eficiência de suas implantações de contêineres, mantendo-se informado sobre atualizações e mudanças importantes.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon EKS Distro: permite criar clusters confiáveis e seguros onde quer que suas aplicações sejam implantadas.	Rastreie as atualizações e os patches de segurança dos clusters criados com o Amazon EKS Distro. Para obter mais informações, consulte Introducing Amazon EKS Distro - an open source Kubernetes distribution used by Amazon EKS (“Apresentando o Amazon EKS Distro - uma distribuição de código aberto do Kubernetes usada pelo Amazon EKS”).
Amazon Elastic Container Service (Amazon ECS): permite executar, interromper e gerenciar os contêineres em um cluster.	Receba notificações quando uma nova AMI otimizada para o Amazon ECS estiver disponível. Para obter mais informações, consulte Subscribing to Amazon ECS-optimized AMI update notifications (“Assinar notificações sobre atualizações da AMI otimizada para Amazon ECS”) no Guia do desenvolvedor do Amazon Elastic Container Service.

Serviços de envolvimento com o cliente

A tabela a seguir descreve como o Amazon SNS aprimora os serviços de engajamento do cliente por meio da integração com o Amazon Connect e o Amazon Simple Email Service (SES), permitindo que você receba alertas e validações, configure mensagens SMS bidirecionais e monitore notificações por e-mail para devoluções, reclamações e entregas. AWS End User Messaging SMS

Essas integrações ajudam você a gerenciar as comunicações com clientes em vários canais.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon Connect: permite configurar uma central de atendimento em nuvem omnichannel para interagir com seus clientes.	Receba alertas e validações. Para obter mais informações, consulte O poder do AWS Amazon Connect no Guia do administrador do Amazon Connect.
AWS End User Messaging SMS: ajuda você a envolver seus clientes enviando e-mails, SMS e mensagens de voz, além de notificações por push.	Configure o SMS bidirecional, que permite receber mensagens dos clientes. Para obter mais informações, consulte Mensagem de SMS bidirecional no Guia do usuário do AWS End User Messaging SMS .
Amazon Simple Email Service (Amazon SES): oferece uma forma econômica de enviar e receber e-mails usando seus próprios endereços de e-mail e domínios.	Receba notificações sobre entregas, devoluções e reclamações. Para obter mais informações sobre o controle de acesso do Amazon SNS, consulte Configuring Amazon SNS notifications for Amazon SES (“Casos de exemplo para o controle de acesso do Amazon SNS”) no Guia do desenvolvedor do Amazon Simple Notification Service.

Serviços de banco de dados

A tabela a seguir descreve como o Amazon SNS se integra a serviços de AWS banco de dados como AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon, Amazon ElastiCache Neptune, Amazon Redshift e Amazon Relational Database Service (RDS) Amazon Relational

Database Service (RDS) para enviar notificações sobre eventos importantes, como migrações de dados, atividades de manutenção, atualizações de cache e alterações no banco de dados.

Essas integrações ajudam você a monitorar e gerenciar seus ambientes de banco de dados com mais eficiência, fornecendo alertas oportunos sobre os principais eventos operacionais.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS Database Migration Service: migra dados de bancos de dados on-premises para a Nuvem AWS.	Receba notificações quando ocorrerem AWS DMS eventos; por exemplo, quando uma instância de replicação for criada ou excluída. Para obter mais informações, consulte Trabalhar com eventos e notificações no AWS Database Migration Service no Guia do usuário do AWS Database Migration Service .
Amazon DynamoDB: fornece performan ce rápida e previsível com escalabilidade integrada no serviço de banco de dados NoSQL totalmente gerenciado.	Receba notificações quando ocorrerem eventos de manutenção. Para obter mais informações, consulte Personalização das configurações do cluster DAX no Guia do desenvolvedor do Amazon DynamoDB.
Amazon ElastiCache — Fornece um cache na memória de alto desempenho, redimensionável e econômico, ao mesmo tempo em que remove a complexidade associada à implantação e ao gerenciamento de um ambiente de cache distribuído.	Receba notificações quando ocorrerem eventos significativos. Para obter mais informações, consulte Notificações de eventos e Amazon SNS no Guia do usuário do Amazon ElastiCache (Memcached).
Amazon Neptune: permite criar e executar aplicações que funcionam com conjuntos de dados altamente conectados.	Receba notificações quando ocorrer um evento do Neptune. Para obter mais informações, consulte Using Neptune event notification (“Usar a notificação de eventos do Neptune”) no Manual do usuário do Neptune.
Amazon Redshift: gerencia todo o trabalho de configuração, operação e escalabilidade de um data warehouse.	Receba notificações de eventos do Amazon Redshift. Para obter mais informações, consulte Notificações de eventos do Amazon

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
	Redshift no Guia de gerenciamento do Amazon Redshift.
Amazon Relational Database Service — Facilita a configuração, a operação e a escalabilidade de um banco de dados relacional na AWS nuvem.	Receba notificações de eventos do Amazon RDS. Para obter mais informações, consulte Usar a notificação de evento do Amazon RDS no Manual do usuário do Amazon RDS.

Serviços das ferramentas de desenvolvedor

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de ferramentas para AWS desenvolvedores, como, AWS CodeBuild,, AWS CodeDeploy Amazon AWS CodeCommit, e AWS CodePipeline fornece notificações para eventos críticos CodeGuru, como alterações de status de compilação, atualizações de repositórios, progresso da implantação, anomalias de desempenho e ações de pipeline.

Essas integrações ajudam você a monitorar e gerenciar com eficiência seus fluxos de trabalho de desenvolvimento de software, recebendo alertas oportunos sobre eventos importantes.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS CodeBuild: compila o código-fonte, executa testes de unidade e produz artefatos prontos para implantação.	Receba notificações se as compilações forem bem-sucedidas, falharem ou passarem de uma fase de compilação para outra. Para obter mais informações, consulte Exemplo de criação de notificações CodeBuild no Guia AWS CodeBuild do usuário.
AWS CodeCommit: fornece controle de versões para armazenar e gerenciar ativos de maneira privativa na nuvem.	Receba notificações sobre eventos CodeCommit do repositório. Para obter mais informações, consulte Exemplo: Criar um AWS CodeCommit gatilho para um tópico do Amazon SNS no Guia do AWS CodeCommit usuário.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS CodeDeploy— Automatiza implantaç ões de aplicativos em EC2 instâncias da Amazon, instâncias locais, funções Lambda sem servidor ou serviços do Amazon ECS.	Receba notificações sobre CodeDeploy implantações ou eventos de instância. Para obter mais informações, consulte Criar um gatilho para um CodeDeploy evento no Guia AWS CodeDeploy do usuário.
Amazon CodeGuru — Coleta dados de desempenho em tempo de execução de seus aplicativos ativos e fornece recomendações que podem ajudá-lo a ajustar o desempenho do seu aplicativo.	Receba notificações quando ocorrerem anomalias. Para obter mais informações, consulte Trabalho com anomalias e relatório s de recomendação no Guia CodeGuru do usuário da Amazon.
AWS CodePipeline: automatiza as etapas necessárias para lançar alterações de software de maneira contínua.	Receba notificações sobre ações de aprovação . Para obter mais informações, consulte Gerenciar ações de aprovação CodePipeline no Guia AWS CodePipeline do usuário.

Serviços Web e móveis de front-end

A tabela a seguir descreve como o Amazon SNS se integra AWS End User Messaging SMS para aprimorar o engajamento do cliente enviando e-mails, SMS, mensagens de voz e notificações push, incluindo a capacidade de configurar SMS bidirecionais para receber mensagens do cliente.

Essa integração permite que você interaja de forma mais eficaz com seus clientes em vários canais de comunicação.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS End User Messaging SMS: ajuda você a envolver seus clientes enviando e-mails, SMS e mensagens de voz, além de notificações por push.	Configure o SMS bidirecional, que permite receber mensagens dos clientes. Para obter mais informações, consulte Mensagem de SMS bidirecional no Guia do usuário do AWS End User Messaging SMS .

Serviços de desenvolvimento de jogos

A tabela a seguir descreve como o Amazon SNS se integra aos GameLift servidores da Amazon para fornecer notificações para eventos de matchmaking e fila em servidores de jogos multijogador baseados em sessões.

Essa integração ajuda os desenvolvedores de jogos a automatizar e monitorar a implantação, a operação e o escalonamento de seus servidores de jogos, garantindo uma experiência de jogo perfeita.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon GameLift Servers — Fornece soluções para hospedar servidores de jogos multijogador baseados em sessões na nuvem, incluindo um serviço totalmente gerenciado para implantação, operação e escalabilidade de servidores de jogos.	Receba notificações sobre eventos de matchmaking e fila. Para obter mais informações, consulte as páginas a seguir: • Para notificações de matchmaking, consulte Configurar notificação de FlexMatch evento no Amazon GameLift Servers FlexMatch Developer Guide. • Para notificações de fila, consulte Configurar notificação de evento para colocação de sessão de jogo no Guia do desenvolvedor de GameLift servidores da Amazon.

Serviços da Internet das Coisas

A tabela a seguir descreve como o Amazon SNS se integra a serviços, AWS IoT Core como,, AWS IoT Greengrass e AWS IoT Device Defender AWS IoT Events, para AWS IoT fornecer notificações para eventos e alertas de IoT.

Essas integrações permitem monitorar com eficácia o comportamento do dispositivo, receber alertas de atividades anormais e gerenciar dispositivos de IoT com atualizações e ações em tempo real.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS IoT Core— Fornece os serviços em nuvem que conectam seus dispositivos de IoT a outros dispositivos e Nuvem AWS serviços.	Receba notificações de AWS IoT Core eventos. Para obter mais informações, consulte Creating an Amazon SNS rule (“Criar uma regra do Amazon SNS”) no Guia do desenvolvedor do AWS IoT .
AWS IoT Device Defender: permite auditar a configuração de seus dispositivos, monitorar os dispositivos conectados para detectar comportamentos anormais e reduzir os riscos de segurança.	Receba alarmes se um dispositivo violar um comportamento. Para obter mais informaçõ es, consulte Como usar a AWS IoT Device Defender detecção no Guia do AWS IoT desenvolvedor.
AWS IoT Events: permite monitorar os equipamento e as frotas de dispositivos em busca de falhas ou alterações na operação, além de acionar ações caso esses eventos ocorram.	Receba notificações de AWS IoT Events eventos. Para obter mais informações, consulte Amazon Simple Notification Service no Guia do desenvolvedor do AWS IoT Events .
AWS IoT Greengrass— se AWS estende aos dispositivos físicos para que eles possam agir localmente nos dados que geram, enquanto ainda usam a nuvem para gerenciamento, análise e armazenamento durável.	Receba notificações de AWS IoT Greengras s eventos. Para mais informações, consulte SNS connector (“Conector SNS”) no Guia do desenvolvedor do AWS IoT Greengrass Version 1 .

Serviços de machine learning

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de aprendizado de AWS máquina, como Amazon, Amazon DevOps Guru, CodeGuru Amazon Lookout for Metrics, Amazon Rekognition SageMaker e Amazon AI, para fornecer notificações sobre anomalias, insights operacionais e atividades de rotulagem de dados.

Essas integrações permitem monitorar o desempenho do aplicativo, receber alertas sobre irregularidades nos dados e agilizar a implantação de modelos de machine learning com atualizações em tempo real.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon CodeGuru — Coleta dados de desempenho em tempo de execução de seus aplicativos ativos e fornece recomendações que podem ajudá-lo a ajustar o desempenho do seu aplicativo.	Receba notificações quando ocorrerem anomalias. Para obter mais informações, consulte Trabalho com anomalias e relatórios de recomendação no Guia CodeGuru do usuário da Amazon.
Amazon DevOps Guru — Gera insights operacionais usando aprendizado de máquina para ajudar você a melhorar o desempenho de seus aplicativos operacionais.	Encaminhe os insights e as confirmações. Para obter mais informações, consulte Forneça insights operacionais baseados em ML para suas equipes de plantão usando o PagerDuty Amazon DevOps Guru no blog de AWS gerenciamento e governança.
Amazon Lookout for Metrics: localiza anomalias nos dados, determina as causas raiz e permite tomar medidas rapidamente.	Receba notificações sobre anomalias. Para obter mais informações, consulte Using Amazon SNS with Lookout for Metrics (“Usar o Amazon SNS com Lookout for Metrics”) no Guia do desenvolvedor do Amazon Lookout for Metrics.
Amazon Rekognition: permite adicionar análises de imagens e vídeos às aplicações	Receba notificações sobre os resultados de solicitações. Para obter mais informações, consulte Reference: Video analysis results notification (“Referência: notificação sobre resultados da análise de vídeo”) no Guia do desenvolvedor do Amazon Rekognition.
Amazon SageMaker AI — permite que cientistas de dados e desenvolvedores criem e treinem modelos de aprendizado de máquina e, em seguida, os implantem diretamente em um ambiente hospedado pronto para produção.	Receba notificações se um objeto de dados for rotulado. Para obter mais informações, consulte Criação de um trabalho de rotulagem de streaming no Amazon SageMaker AI Developer Guide.

Serviços de gerenciamento e governança

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de AWS gerenciamento e governança, como o Amazon Q Developer, em aplicativos de bate-papo AWS CloudFormation, CloudTrail, CloudWatch, AWS Config,, AWS Control Tower, AWS License Manager, AWS Service Catalog AWS Systems Manager, e fornecendo notificações para eventos importantes, como mudanças na infraestrutura, alertas de conformidade e insights operacionais.

Essas integrações ajudam você a monitorar e gerenciar seus AWS ambientes com eficiência, fornecendo alertas e atualizações oportunos para equipes e sistemas relevantes.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon Q Developer em aplicativos de bate-papo — Permite que DevOps as equipes de desenvolvimento de software usem as salas de bate-papo do Amazon Chime e do Slack para monitorar e responder aos eventos operacion ais no. Nuvem AWS	Envie notificações para salas de conversa. Para obter mais informações, consulte Configuração do Amazon Q Developer em aplicativos de bate-papo no Guia do administr ador do Amazon Q Developer em aplicativos de bate-papo.
AWS CloudFormation— Permite criar e provisionar implantações de AWS infraestrutura de forma previsível e repetida.	Receba notificações quando as pilhas forem criadas e atualizadas. Para obter mais informações, consulte Configurar opções de pilha do AWS CloudFormation no Guia do usuário do AWS CloudFormation .
AWS CloudTrail: fornece o histórico de eventos da atividade em sua Conta da AWS .	Receba notificações ao CloudTrail publicar novos arquivos de log em seu bucket do Amazon S3. Para obter mais informaçõ es, consulte Configuração de notificações CloudTrail do Amazon SNS no Guia AWS CloudTrail do usuário.
Amazon CloudWatch — Monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real.	Receba notificações quando os alarmes mudarem de estado. Para obter mais informaçõ es, consulte Usando CloudWatch alarmes da Amazon no Guia do CloudWatch usuário da Amazon.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS Config— Fornece uma visão detalhada da configuração dos AWS recursos em seu Conta da AWS.	Receba notificações quando os recursos forem atualizados ou quando o AWS Config avaliar as regras gerenciadas ou personalizadas em comparação com seus recursos. Para obter mais informações, consulte Notificações AWS Config enviadas para um tópico do SNS e Exemplos de notificações de alteração de item de configuração no Guia do AWS Config desenvolvedor.
AWS Control Tower— Permite que você configure e administre um ambiente seguro, compatível e com várias AWS contas.	Use alertas para ajudar a evitar desvios em sua zona de aterrissagem e receba notificaç ões sobre compatibilidade. Para obter mais informações, consulte Tracking alerts through Amazon Simple Notification Service (“Acompan har alertas pelo Amazon Simple Notificat ion Service”) no Manual do usuário do AWS Control Tower .
AWS License Manager— Ajuda você a gerenciar suas licenças de software de fornecedores de software de forma centralizada em seus ambientes AWS locais.	Receba notificações e alertas do License Manager. Para obter mais informações, consulte Configurações no License Manager no Guia do Usuário do License Manager e Criação de ServiceNow incidentes para AWS License Manager notificações no Blog AWS de Gerenciamento e Governança.
AWS Service Catalog: permite que administr adores de TI criem, gerenciem e distribuam portfólios de produtos aprovados para usuários finais, que podem, então, acessar os produtos de que precisam em um portal personalizado.	Receba notificações sobre eventos de pilha. Para obter mais informações, consulte Limitações das notificações do AWS Service Catalog no Guia do administrador do Service Catalog.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS Systems Manager— Permite que você visualize e controle sua infraestrutura em AWS.	Receba notificações sobre o status dos comandos. Para obter mais informações, consulte Monitoring Systems Manager status changes using Amazon SNS notifications (“Monitorar alterações de status do Systems Manager usando as notificações do Amazon SNS”) no Manual do usuário do AWS Systems Manager .

Serviços de mídia

A tabela a seguir descreve como o Amazon SNS se integra ao Amazon Elastic Transcoder para enviar notificações quando as tarefas de transcodificação de mídia mudam de status, permitindo que você monitore e gerencie com eficiência a conversão de arquivos de mídia armazenados no Amazon S3 em formatos adequados para dispositivos de reprodução de consumidores.

Essa integração ajuda você a simplificar os fluxos de trabalho de processamento de mídia fornecendo alertas em tempo real sobre o status do trabalho.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon Elastic Transcoder: permite converter arquivos de mídia armazenados no Amazon S3 em arquivos de mídia nos formatos usados em dispositivos de reprodução dos consumidores.	Receba notificações quando os trabalhos mudarem de status. Para obter mais informaçõ es, consulteNotifications of job status (“Notific ações de status de trabalhos”) no Guia do desenvolvedor do Amazon Elastic Transcoder.

Serviços de migração e transferência

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de AWS migração e transferência, como, AWS Database Migration Service (DMS) e AWS Application Discovery Service, para fornecer notificações para eventos como coleta de dados do servidor AWS Snowball Edge, atividades de migração de banco de dados e trabalhos de transferência de dados.

Essas integrações ajudam você a gerenciar e monitorar com eficiência seus processos de migração para a nuvem, oferecendo alertas e atualizações em tempo real sobre tarefas críticas de migração.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS Application Discovery Service— Ajuda você a planejar sua migração para o Nuvem AWS coletando dados de uso e configuração sobre seus servidores locais.	Receba notificações de eventos por meio de AWS CloudTrail. Para obter mais informaçõ es, consulte Registrar chamadas de API do Application Discovery Service com o AWS CloudTrail no Guia do usuário do Application Discovery Service.
AWS Database Migration Service: migra dados de bancos de dados on-premises para a Nuvem AWS.	Receba notificações quando ocorrerem AWS DMS eventos; por exemplo, quando uma instância de replicação for criada ou excluída. Para obter mais informações, consulte Trabalhar com eventos e notificações no AWS Database Migration Service no Guia do usuário do AWS Database Migration Service .
AWS Snowball Edge— Usa dispositivos de armazenamento físico para transferir grandes quantidades de dados entre o Amazon S3 e seu local de armazenamento de dados no local em alta velocidade. faster-than-internet	Receba notificações sobre trabalhos do Snowball Edge. Para obter mais informaçõ es, consulte Exemplo de notificações sobre compilação do CodeBuild no Manual do usuário do AWS Snowball Edge .

Serviços de rede e de entrega de conteúdo

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de AWS rede e entrega de conteúdo, como Amazon API Gateway, Amazon, Elastic Load Balancing CloudFront AWS Direct Connect, Amazon Route 53 e Amazon VPC, para enviar notificações de eventos como mensagens de API, alarmes métricos, alterações no estado da conexão CloudFront , eventos do balanceador de carga, status de verificação de integridade e atividades de endpoints da VPC.

Essas integrações ajudam você a monitorar e gerenciar suas operações de rede e entrega de conteúdo, fornecendo alertas e atualizações oportunos.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon API Gateway — permite que você crie e implante seu próprio REST WebSocket APIs em qualquer escala.	Receba mensagens publicadas em um endpoint do API Gateway. Para obter mais informações, consulte Tutorial: Crie uma API REST do API Gateway com AWS integração no Guia do desenvolvedor do API Gateway.
Amazon CloudFront — Acelera a distribuição do seu conteúdo web estático e dinâmico, como arquivos.html, .css, .php, imagens e arquivos de mídia.	Receba notificações quando ocorrerem alarmes com base em CloudFront métricas especificadas. Para obter mais informações, consulte Configuração de alarmes para receber notificações no Amazon CloudFront Developer Guide.
AWS Direct Connect— Vincula sua rede interna a um AWS Direct Connect local por meio de um cabo de fibra óptica Ethernet padrão.	Receba notificações quando houver alteração de estado nos alarmes que monitoram o estado de uma conexão do AWS Direct Connect . Para obter mais informações, consulte Criação de CloudWatch alarmes para monitorar AWS Direct Connect conexões no Guia do AWS Direct Connect usuário.
Elastic Load Balancing — distribui automatic amente seu tráfego de entrada entre vários destinos, como EC2 instâncias, contêineres e endereços IP da Amazon, em mais ou mais zonas de disponibilidade.	Receba notificações sobre os alarmes criados para eventos de balanceador de carga. Para obter mais informações, consulte Criar CloudWatch alarmes para seu balanceador de carga no Guia do usuário para balanceadores de carga clássicos.
Amazon Route 53: fornece registro de domínios, roteamento de DNS e verificação de integridade.	Receba notificações quando o status da verificação e integridade for não íntegro. Para obter mais informações, consulte To receive an Amazon SNS notification when a health check status is unhealthy (console) (“Para receber uma notificação do Amazon SNS quando um status de verificação de integridade for não

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
	íntegro (console) no Guia do desenvolvedor do Amazon Route 53.
Amazon Virtual Private Cloud (Amazon VPC) — Permite que você lance AWS recursos em uma rede virtual que você definiu.	Receba notificações relacionadas a eventos específicos que ocorrem nos endpoints da interface. Para obter mais informações, consulte Criar e gerenciar uma notificação para um serviço de endpoint no Manual do usuário da Amazon VPC.

Serviços de segurança, identidade e compatibilidade

A tabela a seguir descreve como o Amazon SNS se integra aos serviços de AWS segurança, identidade e conformidade, como Amazon, AWS Directory Service Amazon GuardDuty Inspector e, para fornecer notificações sobre alterações de status de diretórios AWS Security Hub, descobertas de segurança, eventos do Inspector e anúncios do hub de segurança.

Essas integrações ajudam você a manter práticas de segurança robustas, oferecendo alertas e atualizações oportunos sobre eventos de segurança e conformidade.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS Directory Service— Fornece várias maneiras de usar o Microsoft Active Directory (AD) com outros Serviços da AWS.	Receba mensagens de e-mail ou texto (SMS) quando houver alteração no status de seu diretório. Para obter mais informações, consulte Configure directory status notifications (“Configurar notificações de status do diretório ”) no Guia de administração do AWS Directory Service .
Amazon GuardDuty — Fornece monitoramento contínuo de segurança para ajudar a identific ar atividades inesperadas e potencialmente não autorizadas ou maliciosas em seu AWS ambiente.	Receba notificações sobre os tipos de descoberta recém-lançados, as atualizações nos tipos de descoberta existentes e outras alterações de funcionalidade. Para obter mais informações, consulte o tópico Assinatura

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
	de GuardDuty anúncios do SNS no Guia do usuário da Amazon. GuardDuty
Amazon Inspector — Testa a acessibilidade de rede de suas EC2 instâncias da Amazon e o estado de segurança de seus aplicativos que são executados nessas instâncias.	Receba notificações referentes a eventos do Amazon Inspector. Para obter mais informaçõ es, consulte Configurar um tópico do SNS para obter notificações do Amazon Inspector no Manual do usuário do Amazon Inspector.
AWS Security Hub: automatiza as verificações de segurança da AWS e centraliza os alertas de segurança.	Receba notificações sobre AWS Security Hub anúncios, incluindo notificações sobre AWS Security Hub controles ou padrões que foram adicionados, editados ou retirados. Para obter mais informações, consulte Assinatura de AWS Security Hub anúncios com o Amazon SNS.

Serviços sem servidor

A tabela a seguir descreve como o Amazon SNS se integra a serviços como Amazon DynamoDB, Amazon EventBridge e Lambda para enviar notificações para eventos importantes, como atualizações de manutenção, fluxos de dados em tempo real e saídas de funções Lambda.

Essas integrações ajudam você a monitorar e gerenciar com eficiência seus aplicativos, recebendo alertas oportunos sobre operações críticas e automatizando as respostas a esses eventos.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
Amazon DynamoDB: fornece performan ce rápida e previsível com escalabilidade integrada no serviço de banco de dados NoSQL totalmente gerenciado.	Receba notificações quando ocorrerem eventos de manutenção. Para obter mais informações, consulte Personalização das configurações do cluster DAX no Guia do desenvolvedor do Amazon DynamoDB.
Amazon EventBridge — entrega um fluxo de dados em tempo real de seus próprios aplicativos, aplicativos software-as-a-service	Receba notificações de EventBridge eventos. Para obter mais informações, consulte as

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
(SaaS) Serviços da AWS e encaminha esses dados para destinos, incluindo o Amazon SNS. EventBridge era anteriormente chamado de CloudWatch Eventos.	EventBridgemetas da Amazon no Guia EventBridge do usuário da Amazon.
AWS Lambda: permite executar código sem provisionar ou gerenciar servidores.	Receba dados de saída de funções definindo um tópico do SNS como uma fila de mensagens mortas do Lambda ou um destino do Lambda. Para obter mais informações, consulte Invocação assíncronano Guia do desenvolvedor do AWS Lambda .

Serviços de armazenamento

A tabela a seguir descreve como o Amazon SNS se integra a serviços de AWS armazenamento AWS Backup, como Amazon Elastic File System (EFS), Amazon S3 Glacier, Amazon S3, além de fornecer notificações para vários eventos, como atividades de backup, alarmes do sistema de arquivos, trabalhos de recuperação de dados, alterações de bucket AWS Snowball Edge e operações de transferência de dados.

Essas integrações ajudam você a monitorar e gerenciar com eficiência suas soluções de armazenamento, recebendo alertas oportunos sobre eventos críticos de armazenamento.

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
AWS Backup: ajuda a centralizar e automatizar o backup de dados entre em Serviços da AWS e na nuvem e on-premises	Receba notificações de AWS Backup eventos. Para obter mais informações, consulte Usando o Amazon SNS para rastrear AWS Backup eventos no Guia do AWS Backup desenvolv edor.
Amazon Elastic File System — Fornece armazenamento de arquivos para suas EC2 instâncias da Amazon.	Receba notificações sobre os alarmes criados para eventos do Amazon EFS. Para obter mais informações, consulte Automated monitorin g tools (“Ferramentas de monitoramento

AWS service (Serviço da AWS)	Benefício do uso com o Amazon SNS
	automatizadas") no Manual do usuário do Amazon Elastic File System.
Amazon S3 Glacier: fornece armazenamento para dados usados com pouca frequência.	Defina uma configuração de notificação em um cofre, de maneira que, quando um trabalho for concluído, uma mensagem será enviada para um tópico do SNS. Para obter mais informaçõ es, consulte Configuring vault notifications in Amazon S3 Glacier ("Configurar notificações de cofre no Amazon S3 Glacier") no Guia do desenvolvedor do Amazon S3 Glacier.
Amazon Simple Storage Service (Amazon S3): fornece armazenamento de objetos.	Receba notificações quando ocorrerem alterações em um bucket do Amazon S3 ou na rara ocasião em que os objetos não forem replicados para a região de destino. Para obter mais informações, consulte Demonstra ção: configurar um bucket para notificações (tópico do SNS ou fila do SQS) e Monitoram ento do progresso com métricas de replicaçã o e notificações de eventos do Amazon S3 no Manual do usuário do Amazon Simple Storage Service.
AWS Snowball Edge— Usa dispositivos de armazenamento físico para transferir grandes quantidades de dados entre o Amazon S3 e seu local de armazenamento de dados no local em alta velocidade. faster-than-internet	Receba notificações sobre trabalhos do Snowball Edge. Para obter mais informaçõ es, consulte Exemplo de notificações sobre compilação do CodeBuild no Manual do usuário do AWS Snowball Edge .

Fontes de eventos adicionais

A tabela a seguir descreve como o Amazon SNS pode ser usado para receber notificações oportunas sobre atualizações AWS diárias de recursos e alterações nos intervalos de endereços

AWS IP, garantindo que você seja informado sobre os últimos lançamentos de AWS serviços, tipos de instância, VPC endpoints e alterações de endereço IP público.

Essas integrações ajudam você a up-to-date acompanhar as mudanças na AWS infraestrutura e gerenciar seus recursos de forma eficaz.

Origem	Benefício do uso com o Amazon SNS
AWS Atualizações diárias de recursos	Receba informações detalhadas em tempo hábil sobre lançamentos e atualizações da AWS por meio de um tópico do Amazon SNS. Esses lançamentos incluem endpoints Amazon VPC Regiões da AWS Serviços da AWS, Serviços da AWS integrados com Service AWS Quotas, tipos de instância Amazon, tipos de instância Amazon AI, EC2 tipos de SageMaker instância Amazon Nimble Studio, versões do mecanismo de banco de dados Amazon RDS e versões do Amazon MSK Apache Kafka. Para obter mais informações, consulte Inscrever-se para receber atualizações AWS diárias de recursos via Amazon SNS no blog de AWS notícias.
AWS Intervalos de endereços IP	Receba notificações de alterações nos intervalos de AWS IP por meio de um tópico do Amazon SNS. Para obter mais informaçõ es, consulte as notificações de intervalos de endereços AWS IP no Referência geral da Amazon Web Servicese Inscreva-se para receber alterações de endereço IP AWS público via Amazon SNS no blog de AWS notíc ias.

Consulte as seguintes fontes para obter mais informações sobre a computação baseada em eventos:

- O que é uma arquitetura orientada a eventos?

- Computação orientada a eventos com o Amazon SNS AWS e serviços de computação, armazenamento, banco de dados e rede no blog de computação AWS
- Enriquecimento de arquiteturas orientadas por eventos com pipelines de bifurcação de AWS eventos no blog de computação AWS

Destinos de eventos do Amazon SNS

Este tópico lista todos os destinos de eventos, organizados por mensagens application-to-application (A2A) e notificações application-to-person (A2P).

> **Note**
>
> O Amazon SNS introduziu os tópicos FIFO em outubro de 2020. Atualmente, a maioria dos AWS serviços oferece suporte ao recebimento de eventos somente de tópicos padrão do SNS. O Amazon SQS oferece suporte ao recebimento de eventos de tópicos padrão e FIFO do SNS.

Destinos A2A

A tabela a seguir descreve como o Amazon SNS pode entregar eventos para vários destinos application-to-application (A2A), como Amazon Data Firehose, Lambda, Amazon SQS, Event Fork Pipelines e endpoints AWS HTTP/S.

Essas integrações permitem arquivar e analisar dados, acionar uma lógica de negócios personalizada, facilitar a integração de aplicativos e rotear eventos para webhooks externos, aumentando a eficiência e a flexibilidade das arquiteturas orientadas por eventos.

Destino do evento	Benefício do uso com o Amazon SNS
Amazon Data Firehose	Entregue eventos para fluxos de entrega para fins de arquivamento e análise. Por meio de fluxos de entrega, você pode entregar eventos para AWS destinos como Amazon Simple Storage Service (Amazon S3), Amazon Redshift e OpenSearch Amazon Service OpenSearch (Service), ou para destinos

Destino do evento	Benefício do uso com o Amazon SNS
	de terceiros, como Datadog, New Relic, MongoDB e Splunk. Para obter mais informaçõ es, consulte Fanout de fluxos de entrega do Firehose.
AWS Lambda	Entregue eventos a funções para acionar a execução da lógica de negócios personalizada. Para obter mais informações, consulte Fanout notificações do Amazon SNS para funções do Lambda para processamento automatizado.
Amazon SQS	Entregue eventos para filas para fins de integração de aplicações. Para obter mais informações, consulte Fanout de notificações do Amazon SNS para filas do Amazon SQS para processamento assíncrono.
AWS Event Fork Pipelines	Entregue eventos para backup e armazenam ento de eventos, pesquisa e análise de eventos ou pipelines de repetição de eventos. Para obter mais informações, consulte Fanout eventos do Amazon SNS no AWS Event Fork Pipelines.
HTTP/S	Entregue eventos para webhooks externos. Para obter mais informações, consulte Fanout de notificações do Amazon SNS para endpoints HTTPS.

Destinos A2P

A tabela a seguir descreve como o Amazon SNS entrega notificações application-to-person (A2P) para vários destinos, incluindo telefones celulares via SMS e notificações push nativas, caixas de entrada de e-mail, salas de bate-papo do Amazon Chime, canais do Slack e informações operacionais para equipes de plantão via. PagerDuty

Essas integrações aprimoram a comunicação e a eficiência operacional, permitindo alertas e atualizações em tempo real em várias plataformas e canais de comunicação.

Destino do evento	Benefício do uso com o Amazon SNS
SMS	Entregue eventos para telefones celulares como mensagens de texto. Para obter mais informações, consulte Mensagens de texto em dispositivos móveis com o Amazon SNS.
E-mail	Entregue eventos para caixas de entrada como mensagens de e-mail. Para obter mais informações, consulte Configuração e gerenciamento de assinaturas de e-mail do Amazon SNS.
Endpoint da plataforma	Entregue eventos para telefones celulares como notificações por push nativas. Para obter mais informações, consulte Enviar notificações por push para dispositivos móveis com o Amazon SNS.
Amazon Q Developer em aplicações de chat	Entregue eventos para salas de bate-papo do Amazon Chime ou canais do Slack. Para obter mais informações, consulte as seguintes páginas no Guia do administrador do Amazon Q Developer em aplicativos de bate-papo: • Configurando o Amazon Q Developer em aplicativos de bate-papo com o Amazon Chime • Configurando o Amazon Q Developer em aplicativos de bate-papo com o Slack • Usando o Amazon Q Developer em aplicativos de bate-papo com outros AWS serviços

Destino do evento	Benefício do uso com o Amazon SNS
PagerDuty	Forneça insights operacionais para equipes de plantão. Para obter mais informações, consulte Forneça insights operacionais baseados em ML para suas equipes de plantão por meio do PagerDuty Amazon DevOps Guru no blog de AWS gerenciamento e governança.

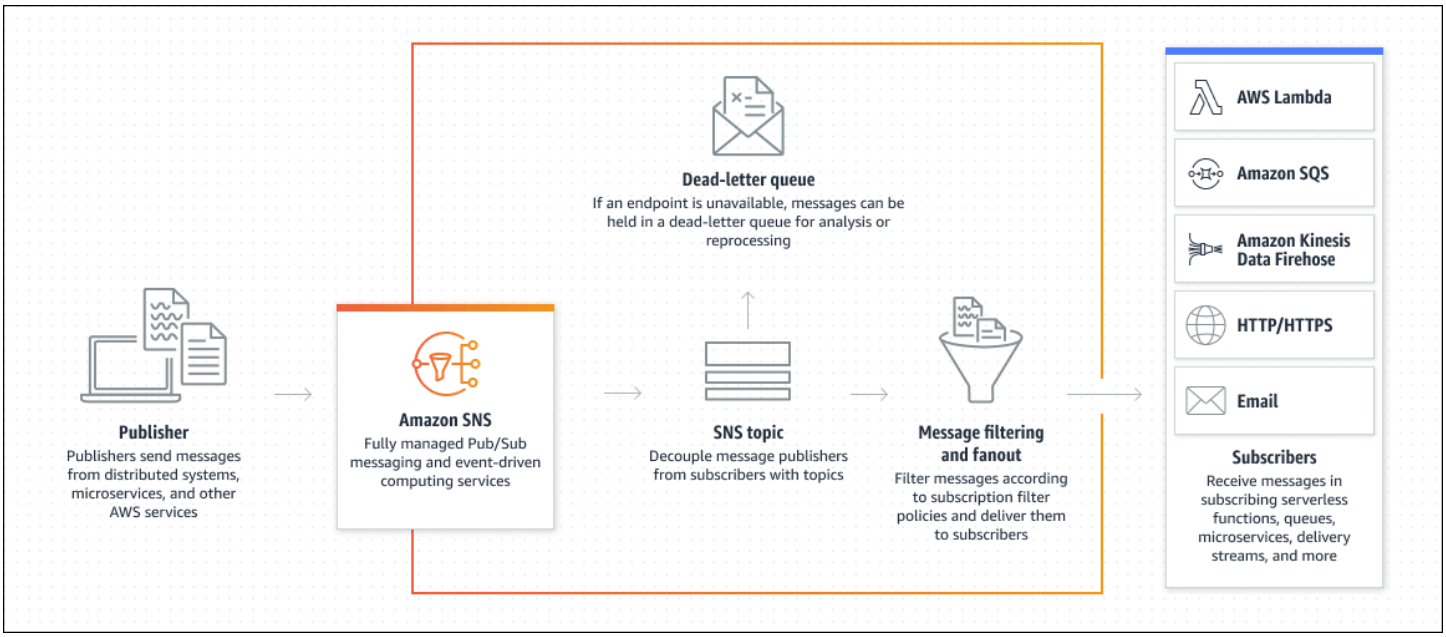
Note

Você pode oferecer AWS eventos nativos e personalizados para aplicativos de bate-papo:

- AWS Eventos nativos — Você pode usar o Amazon Q Developer em aplicativos de bate-papo para enviar AWS eventos nativos, por meio de tópicos do Amazon SNS, para o Amazon Chime e o Slack. O conjunto suportado de AWS eventos nativos inclui eventos da Gerenciamento de Faturamento e Custos da AWS, AWS Health, AWS CloudFormation CloudWatch, Amazon e muito mais. Para obter mais informações, consulte Usando o Amazon Q Developer em aplicativos de bate-papo com outros serviços no Guia do administrador do Amazon Q Developer em aplicativos de bate-papo.
- Eventos personalizados: você também pode enviar seus eventos personalizados, por meio de tópicos do Amazon SNS, para o Amazon Chime, Slack e Microsoft Teams. Para fazer isso, você publica eventos personalizados em um tópico do SNS, que entrega os eventos para uma função do Lambda inscrita. A função do Lambda usa o webhook da aplicação de conversa para entregar os eventos aos destinatários. Para obter mais informações, consulte How do I use webhooks to publish Amazon SNS messages to Amazon Chime, Slack, or Microsoft Teams? (“Como uso webhooks para publicar mensagens do Amazon SNS no Amazon Chime, Slack ou Microsoft Teams?”).

Usando o Amazon SNS para mensagens application-to-application

O Amazon SNS simplifica as mensagens application-to-application (A2A) separando editores de assinantes, o que oferece suporte a microsserviços, sistemas distribuídos e aplicativos sem servidor. As mensagens são enviadas para tópicos do Amazon SNS, onde podem ser filtradas e entregues aos assinantes, como Lambda, Amazon SQS ou endpoints HTTP. Se a entrega falhar, as mensagens são armazenadas em uma fila de mensagens não entregues para análise ou reprocessamento adicionais.



Fanout de fluxos de entrega do Firehose

Você pode inscrever streams de entrega do Amazon Data Firehose para tópicos do Amazon SNS, permitindo que você envie notificações para endpoints adicionais de armazenamento e análise. As mensagens publicadas em um tópico do Amazon SNS são enviadas para o stream de entrega do Firehose assinante e entregues ao destino conforme configurado no Firehose. O proprietário de uma assinatura pode assinar até cinco fluxos de entrega do Firehose em um tópico do Amazon SNS. Cada stream de entrega do Firehose tem uma cota padrão para solicitações e throughput por segundo. Esse limite poderia resultar em mais mensagens publicadas (tráfego de entrada) do que entregues (tráfego de saída). Se houver mais tráfego de entrada que de saída, sua assinatura pode acumular um grande backlog de menagens, causando alta latência na entrega de mensagens. Você

pode solicitar um aumento na cota com base na taxa de publicação para evitar impactos adversos na workload.

Por meio dos streams de entrega do Firehose, você pode distribuir notificações do Amazon SNS para o Amazon Simple Storage Service (Amazon S3), Amazon Redshift, OpenSearch Amazon Service (Service) e para provedores de serviços terceirizados, como OpenSearch Datadog, New Relic, MongoDB e Splunk.

Por exemplo, você pode usar essa funcionalidade para armazenar permanentemente mensagens enviadas para um tópico em um bucket do Amazon S3 para fins de conformidade, arquivamento ou outros. Para fazer isso, crie um stream de entrega do Firehose com um destino de bucket do Amazon S3 e inscreva esse stream de entrega no tópico do Amazon SNS. Como outro exemplo, para realizar análises em mensagens enviadas para um tópico do Amazon SNS, crie um stream de entrega com um destino de índice OpenSearch de serviços. Depois, inscreva o fluxo de entrega do Firehose no tópico do Amazon SNS.

O Amazon SNS também oferece suporte a registro em log do status da entrega de mensagens para notificações enviadas para endpoints do Firehose. Para obter mais informações, consulte Status de entrega de mensagens do Amazon SNS.

Pré-requisitos para assinatura de fluxos de entrega do Firehose para tópicos do Amazon SNS

Para assinar um stream de entrega do Amazon Data Firehose para um tópico do SNS, você Conta da AWS deve ter:

- Um tópico do SNS padrão. Para obter mais informações, consulte Criar um tópico do Amazon SNS.
- Fluxo de entrega do Firehose. Para obter mais informações, consulte Creating an Amazon Data Firehose Delivery Stream e Grant Your Application Access to Your Firehose Resources no Guia do desenvolvedor do Amazon Kinesis Data Firehose.
- Uma função AWS Identity and Access Management (IAM) que confia no responsável pelo serviço principal do Amazon SNS e tem permissão para gravar no stream de entrega. Você deverá inserir o nome do recurso da Amazon (ARN) dessa função como `SubscriptionRoleARN` quando cria a assinatura. O Amazon SNS assume essa função, o que permite que o Amazon SNS coloque registros no fluxo de entrega do Firehose.

  O exemplo de política seguinte mostra as permissões recomendadas:

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

Para fornecer permissão total para usar o Firehose, você também pode usar a política AWS gerenciada. `AmazonKinesisFirehoseFullAccess` Ou, para fornecer permissões mais estritas para usar o Firehose, você pode criar sua própria política. No mínimo, a política deve fornecer permissão para executar a operação `PutRecord` em um fluxo de entrega específico.

Em todos os casos, você também deve editar o relacionamento de confiança para incluir o principal de serviço do Amazon SNS. Por exemplo:

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

Para obter mais informações sobre a criação de funções, consulte Como criar uma função para delegar permissões a um AWS serviço no Guia do usuário do IAM.

Depois de concluir esses requisitos, você pode assinar o fluxo de entrega do tópico do SNS.

Inscrever um fluxo de entrega do Firehose em um tópico do Amazon SNS

Para entregar notificações do Amazon SNS em fluxos de entrega do Amazon Data Firehose, primeiro certifique-se de que você abordou todos os pré-requisitos. Para obter uma lista de endpoints compatíveis, consulte Endpoints e cotas do Amazon Data Firehose no Referência geral da Amazon Web Services.

Para inscrever um fluxo de entrega do Firehose em um tópico

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Assinaturas.
3. Na página Assinaturas, escolha Criar assinatura.
4. Na página Criar assinatura, na seção Detalhes, faça o seguinte:
   a. Em ARN do tópico, escolha o nome do recurso da Amazon (ARN) de um tópico padrão.
   b. Em Protocolo, escolha Kinesis Data Firehose.
   c. Em Endpoint, escolha o ARN de um fluxo de entrega do Firehose que pode receber notificações do Amazon SNS.
   d. Em ARN da função de assinatura, especifique o ARN da função do AWS Identity and Access Management (IAM) que você criou para gravar em fluxos de entrega do Firehose. Para obter mais informações, consulte Pré-requisitos para assinatura de fluxos de entrega do Firehose para tópicos do Amazon SNS.
   e. (Opcional) Para remover quaisquer metadados do Amazon SNS das mensagens publicadas, escolha Habilitar a entrega de mensagens. Para obter mais informações, consulte Entrega de mensagens brutas do Amazon SNS.
5. (Opcional) Para configurar uma política de filtros, expanda a seção Subscription filter policy (Política de filtro de assinatura). Para obter mais informações, consulte Políticas de filtro de assinatura do Amazon SNS.
6. (Opcional) Para configurar uma fila de mensagens não entregues para a assinatura, expanda a seção Redrive policy (dead-letter queue) (Política de redirecionamento (fila de mensagens

não entregues)). Para obter mais informações, consulte Filas de mensagens não entregues do Amazon SNS.

7. Selecione Create subscription.

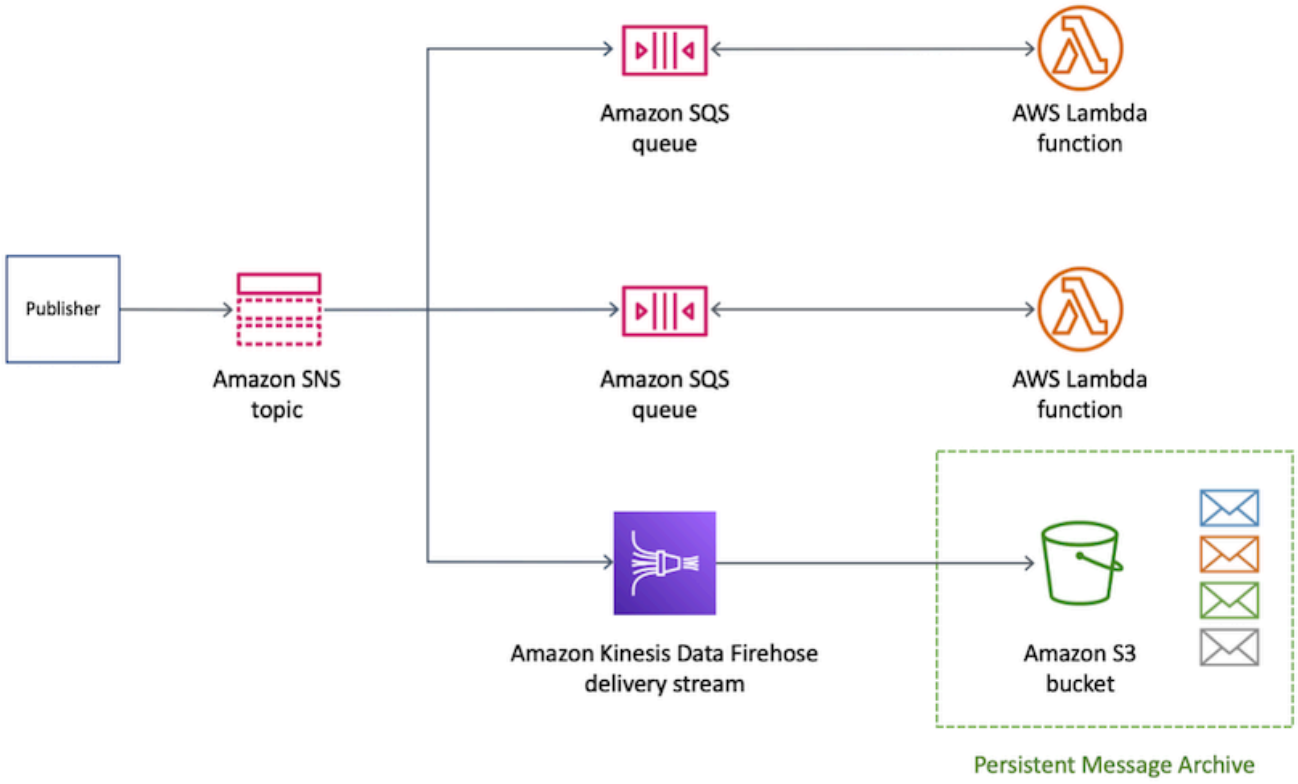
O console cria a assinatura e abre a página Details (Detalhes) da assinatura.

Gerenciamento de mensagens do Amazon SNS em vários destinos de fluxo de entrega

Os streams de entrega do Amazon Data Firehose permitem que você gerencie mensagens do Amazon SNS em vários destinos, permitindo a integração com Amazon S3, Amazon Service OpenSearch , Amazon Redshift e endpoints HTTP para armazenamento, indexação e análise. Ao configurar adequadamente a formatação e a entrega de mensagens, você pode armazenar notificações do Amazon SNS no Amazon S3 para processamento posterior, analisar dados estruturados de mensagens usando o Amazon Athena, indexar mensagens para pesquisa e visualização OpenSearch em tempo real e estruturar arquivos no Amazon Redshift para consultas avançadas.

Armazenar e analisar mensagens do Amazon SNS em destinos do Amazon S3

Este tópico explica como os streams de entrega do Amazon Data Firehose publicam dados no Amazon Simple Storage Service (Amazon S3).



Tópicos

- Formatação de notificações do Amazon SNS para armazenamento em destinos do Amazon S3
- Analisar mensagens do Amazon SNS armazenadas no Amazon S3 usando o Athena

Formatação de notificações do Amazon SNS para armazenamento em destinos do Amazon S3

O exemplo a seguir mostra uma notificação do Amazon SNS enviada para um bucket do Amazon Simple Storage Service (Amazon S3), com indentação para facilitar a leitura.

> **Note**
>
> Neste exemplo, a entrega de mensagens brutas está desativada para a mensagem publicada. Quando a entrega de mensagens brutas é desativada, o Amazon SNS adiciona metadados JSON à mensagem, incluindo estas propriedades:
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- `Subject`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Para obter mais informações sobre a entrega de mensagens brutas, consulte Entrega de mensagens brutas do Amazon SNS.

```
{
    "Type": "Notification",
    "MessageId": "719a6bbf-f51b-5320-920f-3385b5e9aa56",
    "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-kinesis-test-topic",
    "Subject": "My 1st subject",
    "Message": "My 1st body",
    "Timestamp": "2020-11-26T23:48:02.032Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:333333333333:my-kinesis-test-
topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5",
    "MessageAttributes": {
        "myKey1": {
            "Type": "String",
            "Value": "myValue1"
        },
        "myKey2": {
            "Type": "String",
            "Value": "myValue2"
        }
    }
 }
```

O exemplo a seguir mostra três mensagens do SNS enviadas por meio de um fluxo de entrega do Amazon Data Firehose para o mesmo bucket do Amazon S3. O buffer é aplicado e as quebras de linha separam cada mensagem.

```
{"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"}
```


Analisar mensagens do Amazon SNS armazenadas no Amazon S3 usando o Athena

Esta página explica como analisar as mensagens do Amazon SNS que são enviadas pelos fluxos de entrega do Amazon Data Firehose para destinos do Amazon Simple Storage Service (Amazon S3).

Para analisar mensagens do SNS enviadas por meio de fluxos de entrega do Firehose para destinos do Amazon S3

1. Configure seus recursos do Amazon S3. Para obter instruções, consulte Criação de buckets no Manual do usuário do Amazon Simple Storage Service e Como trabalhar com buckets do Amazon S3 no Manual do usuário Amazon Simple Storage Service.
2. Configure seu fluxo de entrega. Para obter instruções, consulte Escolher o Amazon S3 para seu destino no Guia do desenvolvedor do Amazon Data Firehose.
3. Use o Amazon Athena para consultar os objetos do Amazon S3 com o SQL padrão. Para obter mais informações, consulte Conceitos básicos no Manual do usuário do Amazon Athena.

Consulta de exemplo

Para esta consulta de exemplo, suponha o seguinte:

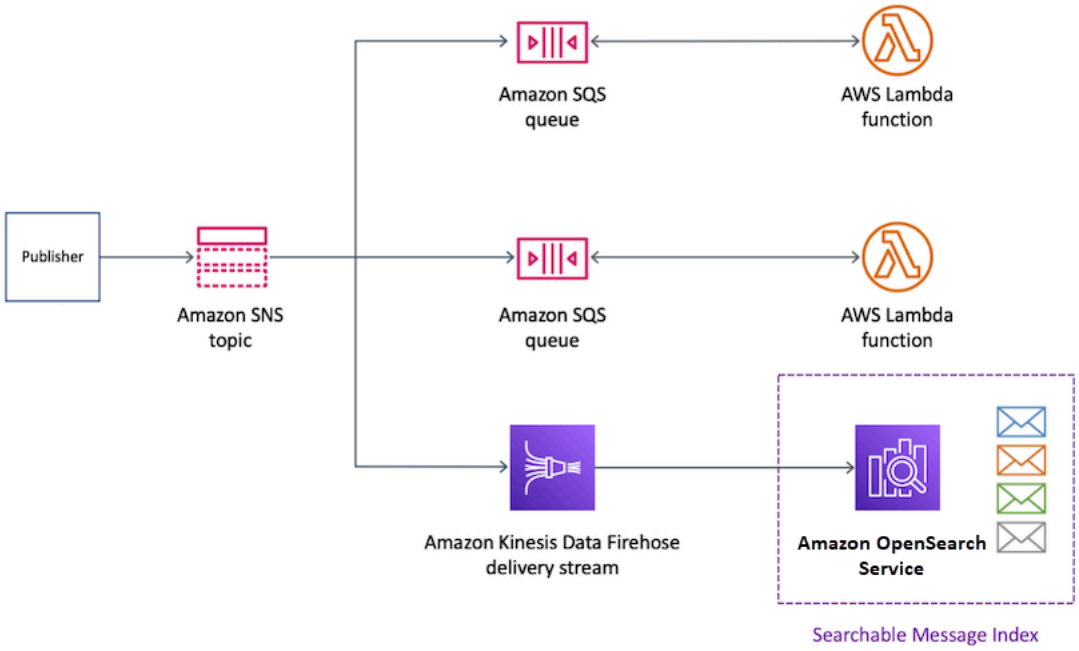
- As mensagens são armazenadas na tabela `notifications` no esquema `default`.
- A tabela `notifications` inclui uma coluna `timestamp` com um tipo de `string`.

A consulta a seguir retorna todas as mensagens do SNS recebidas no intervalo de datas especificado:

```
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';
```

Integração de mensagens do Amazon SNS com destinos do Amazon Service OpenSearch

Esta seção explica como os streams de entrega do Amazon Data Firehose publicam dados no Amazon OpenSearch Service (Service)OpenSearch .



Tópicos

- Armazenamento e formatação de notificações OpenSearch do Amazon SNS em índices de serviço
- Analisando mensagens do Amazon SNS para destinos de serviços OpenSearch

Armazenamento e formatação de notificações OpenSearch do Amazon SNS em índices de serviço

O exemplo a seguir demonstra uma notificação do Amazon SNS enviada para um índice do OpenSearch Amazon Service OpenSearch (Service) chamado. `my-index` Esse índice tem um

campo de filtro de tempo no campo `Timestamp`. A notificação SNS é colocada na propriedade `_source` da carga útil.

> **Note**
>
> Neste exemplo, a entrega de mensagens brutas está desativada para a mensagem publicada. Quando a entrega de mensagens brutas é desativada, o Amazon SNS adiciona metadados JSON à mensagem, incluindo estas propriedades:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Para obter mais informações sobre a entrega de mensagens brutas, consulte Entrega de mensagens brutas do Amazon SNS.


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "49613100963111323203250405402193283794773886550985932802.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
```

```
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

Analisando mensagens do Amazon SNS para destinos de serviços OpenSearch

Este tópico explica como analisar mensagens do Amazon SNS enviadas por meio de fluxos de entrega do Amazon Data Firehose para destinos do Amazon OpenSearch Service (Service). OpenSearch

Para analisar mensagens de SNS enviadas por meio de fluxos OpenSearch de entrega do Firehose para destinos de serviço

1. Configure seus recursos OpenSearch de serviço. Para obter instruções, consulte Conceitos básicos do Amazon OpenSearch Service no Amazon OpenSearch Service Developer Guide.
2. Configure seu fluxo de entrega. Para obter instruções, consulte Escolha o OpenSearch serviço para seu destino no Guia do desenvolvedor do Amazon Data Firehose.
3. Execute uma consulta usando consultas OpenSearch de serviço e Kibana. Para obter mais informações, consulte Etapa 3: Pesquisar documentos em um domínio de OpenSearch serviço e Kibana no Amazon OpenSearch Service Developer Guide.

Consulta de exemplo

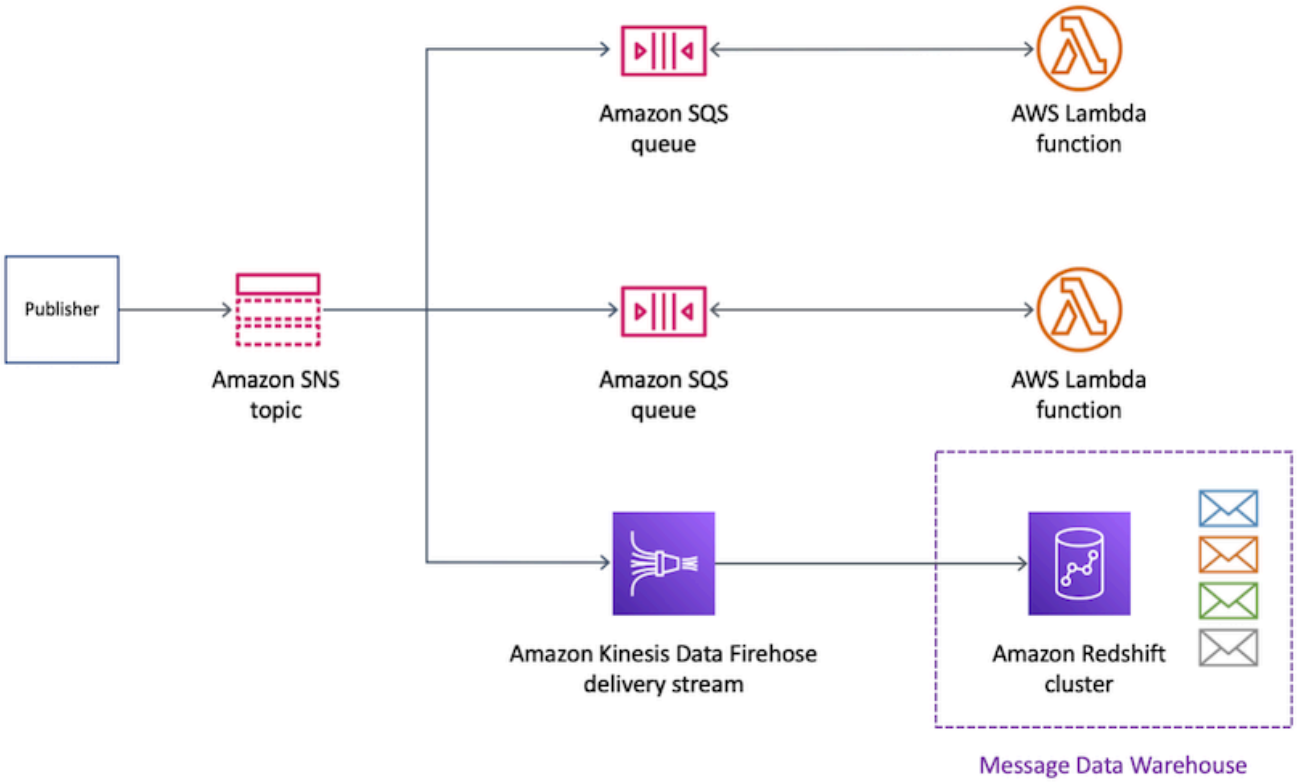
O exemplo a seguir cria o índice `my-index` para todas as mensagens do SNS recebidas no intervalo de datas especificado:

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

Configuração da entrega e análise de mensagens do Amazon SNS nos destinos do Amazon Redshift

Este tópico explica como distribuir notificações do Amazon SNS para um stream de entrega do Amazon Data Firehose, que então publica dados no Amazon Redshift. Com essa configuração, você pode se conectar ao banco de dados do Amazon Redshift e usar uma ferramenta de consulta SQL para recuperar mensagens do Amazon SNS que correspondam a critérios específicos.



Tópicos

- Estruturação de arquivos de mensagens do Amazon SNS em tabelas do Amazon Redshift
- Analisar mensagens do Amazon SNS armazenadas em destinos do Amazon Redshift

Estruturação de arquivos de mensagens do Amazon SNS em tabelas do Amazon Redshift

Para endpoints do Amazon Redshift, as mensagens do Amazon SNS são arquivadas como linhas em uma tabela. Aqui está um exemplo de como os dados são armazenados:

> **Note**
>
> Neste exemplo, a entrega de mensagens brutas está desativada para a mensagem publicada. Quando a entrega de mensagens brutas é desativada, o Amazon SNS adiciona metadados JSON à mensagem, incluindo estas propriedades:
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- `Subject`
- `Message`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Para obter mais informações sobre a entrega de mensagens brutas, consulte Entrega de mensagens brutas do Amazon SNS.
Embora o Amazon SNS adicione propriedades à mensagem usando a capitalização mostrada nesta lista, os nomes de colunas nas tabelas do Amazon Redshift aparecem em todos os caracteres minúsculos. Para transformar os metadados JSON para o endpoint do Amazon Redshift, você pode usar o comando `COPY`. Para obter mais informações, consulte Copy from JSON examples (“Copiar de exemplos JSON”) e Load from JSON data using the 'auto ignorecase' option (“Carregar de dados JSON usando a opção “auto ignorecase””) no Guia do desenvolvedor de banco de dados do Amazon Redshift.

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notificação	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Exemplo do assunto	Exemplo de mensagem	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Ação=CancelarInscrição&=arn:aws:sns:us-east-SubscriptionArn1:1111111	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						11111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	
Notificação	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Exemplo de assunto 2	Exemplo de mensagem 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Ação=CancelarInscrição&=arn:aws:sns:us-east-SubscriptionArn1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notificação	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Exemplo de assunto 3	Exemplo de mensagem 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Ação=CancelarInscrição&=arn:aws:sns:us-east-SubscriptionArn1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Para obter mais informações sobre divulgação de notificações de eventos para o Amazon Redshift, consulte Configuração da entrega e análise de mensagens do Amazon SNS nos destinos do Amazon Redshift.

Analisar mensagens do Amazon SNS armazenadas em destinos do Amazon Redshift

Este tópico descreve como analisar mensagens do Amazon SNS que são enviadas por meio de fluxos de entrega do Amazon Data Firehose para destinos do Amazon Redshift.

Para analisar mensagens do SNS enviadas por meio de fluxos de entrega do Firehose para destinos do Amazon Redshift

1. Configure seus recursos do Amazon Redshift. Para obter instruções, consulte Conceitos básicos do Amazon Redshift no Guia de conceitos básicos do Amazon Redshift.
2. Configure seu fluxo de entrega. Para obter instruções, consulte Escolher o Amazon Redshift para seu destino no Guia do desenvolvedor do Amazon Data Firehose.
3. Execute uma consulta. Para obter mais informações, confira Consultar um banco de dados usando o editor de consultas no Guia de gerenciamento do Amazon Redshift.

Consulta de exemplo

Para esta consulta de exemplo, suponha o seguinte:

- As mensagens são armazenadas na tabela `notifications` no esquema `public` padrão.
- A propriedade `Timestamp` da mensagem SNS é armazenada na coluna `timestamp` da tablea com um tipo de dados de coluna `timestamptz`.

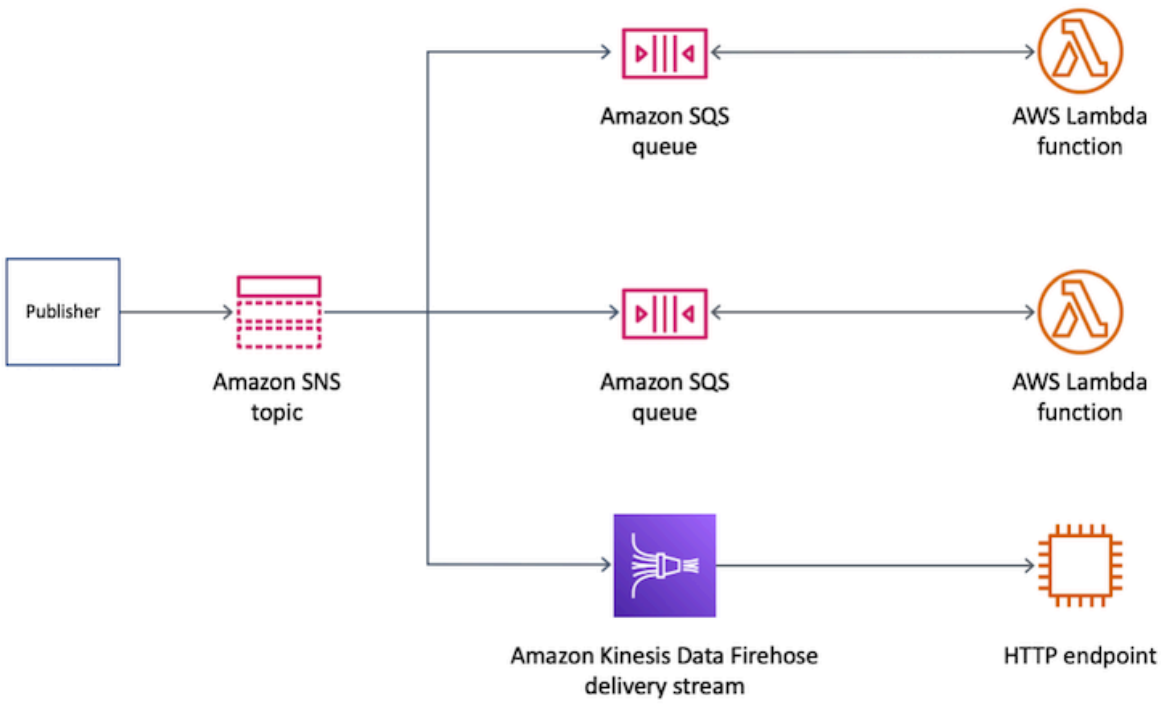
> **Note**
>
> Para transformar os metadados JSON para o endpoint do Amazon Redshift, você pode usar o comando `COPY`. Para obter mais informações, consulte Copy from JSON examples (“Copiar de exemplos JSON”) e Load from JSON data using the 'auto ignorecase' option (“Carregar de dados JSON usando a opção “auto ignorecase”“) no Guia do desenvolvedor de banco de dados do Amazon Redshift.

A consulta a seguir retorna todas as mensagens do SNS recebidas no intervalo de datas especificado:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

Configurar a entrega de mensagens do Amazon SNS para destinos HTTP usando o Amazon Data Firehose

Este tópico explica como os streams de entrega do Amazon Data Firehose publicam dados em endpoints HTTP.



Tópicos

- Formato de notificação do Amazon SNS para entrega em destinos HTTP

Formato de notificação do Amazon SNS para entrega em destinos HTTP

Aqui está um exemplo de um corpo de solicitação HTTP POST do Amazon SNS, enviado por meio de um stream de entrega do Amazon Data Firehose para um endpoint HTTP. A notificação do Amazon SNS é codificada como uma carga base64 dentro da propriedade de registros.

> **Note**
>
> Neste exemplo, a entrega de mensagens brutas está desativada para a mensagem publicada. Para obter mais informações sobre a entrega de mensagens brutas, consulte Entrega de mensagens brutas do Amazon SNS.


```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
        "data":
 "eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```

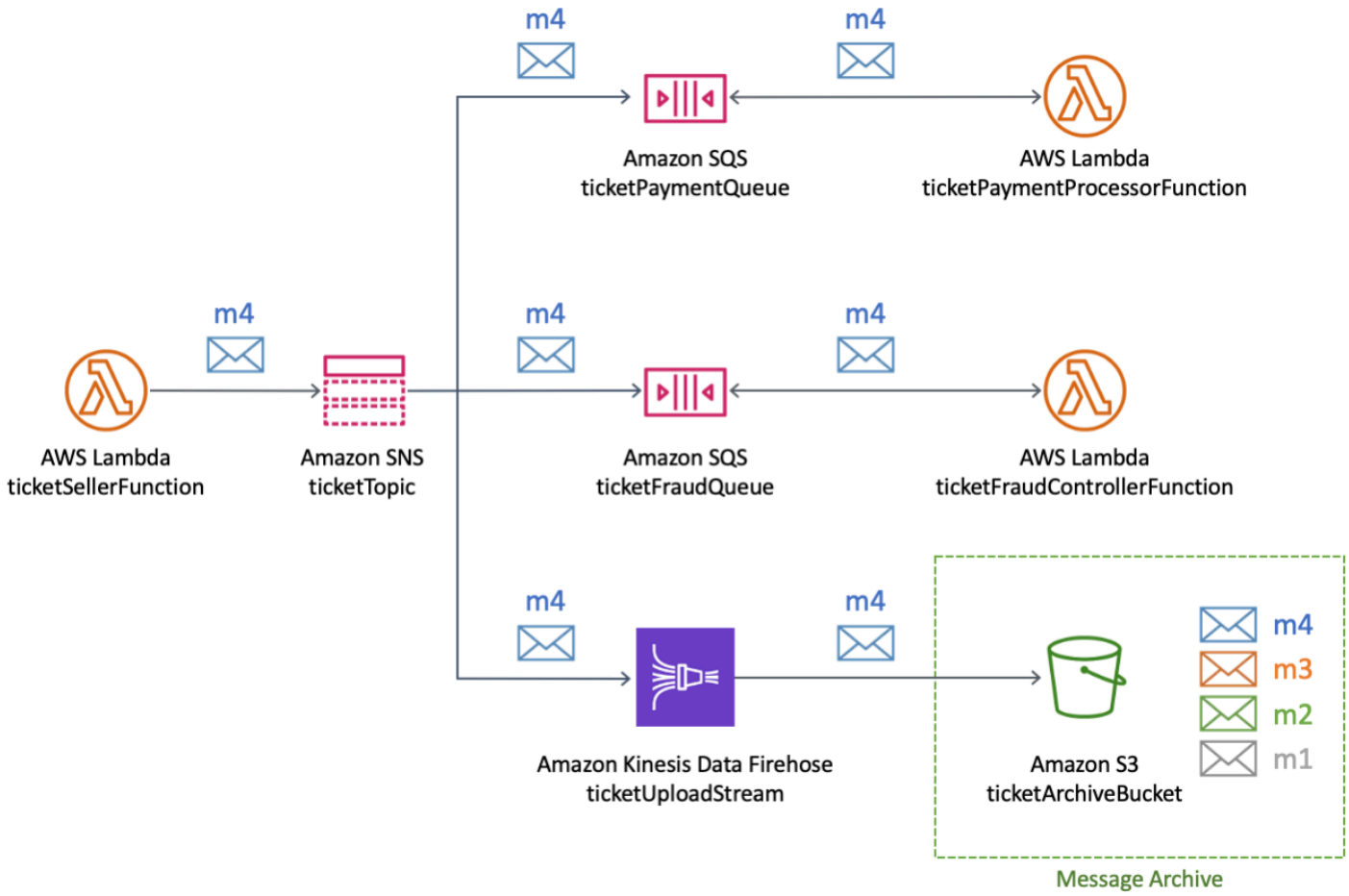

Arquivamento e análise de mensagens do Amazon SNS: um exemplo de uso para plataformas de emissão de passagens aéreas

Este tópico fornece um tutorial para um caso de uso comum de arquivamento e análise de mensagens do Amazon SNS.

A configuração deste caso de uso é uma plataforma de emissão de bilhetes de avião que opera em um ambiente regulamentado.

1. A plataforma está sujeita a um framework de conformidade que exige que a empresa arquive todas as vendas de ingressos por pelo menos cinco anos.
2. Para atingir a meta de conformidade na retenção de dados, a empresa subscreve um stream de entrega do Amazon Data Firehose para um tópico existente do Amazon SNS.
3. O destino do fluxo de entrega é um bucket do Amazon Simple Storage Service (Amazon S3). Com essa configuração, todos os eventos publicados no tópico SNS são arquivados no bucket do Amazon S3.

O diagrama seguinte mostra a arquitetura dessa configuração:



Para executar análises e obter insights sobre vendas de ingressos, a empresa executa consultas SQL usando o Amazon Athena. Por exemplo, a empresa pode consultar para saber mais sobre os destinos mais populares e os passageiros mais frequentes.

Para criar os AWS recursos para esse caso de uso, você pode usar o AWS Management Console ou um AWS CloudFormation modelo.

Tópicos

- Configurar os recursos iniciais da AWS para arquivamento e análise de mensagens do Amazon SNS
- Configurar um stream de entrega do Firehose para arquivamento de mensagens do Amazon SNS
- Assinar o fluxo de entrega do Firehose no tópico do Amazon SNS
- Testar e consultar uma configuração do Amazon SNS para um gerenciamento de dados eficaz
- Automatizar o arquivamento de mensagens do Amazon SNS com um modelo AWS CloudFormation

Configurar os recursos iniciais da AWS para arquivamento e análise de mensagens do Amazon SNS

Este tópico descreve como criar os recursos necessários para o exemplo de caso de uso de arquivamento e análise de mensagens:

- Um bucket do Amazon Simple Storage Service (Amazon S3)
- Duas filas do Amazon Simple Queue Service (Amazon SQS)
- Um tópico do Amazon SNS
- Duas inscrições do Amazon SQS no tópico do Amazon SNS

Para criar os recursos iniciais

1. Criar o bucket do Amazon S3:

   a. Abra o console Amazon S3.

   b. Escolha Criar bucket.

   c. Em Bucket name (Nome do bucket), insira um nome globalmente exclusivo. Mantenha os outros campos como os padrões.

   d. Escolha Criar bucket.

   Para obter mais informações sobre buckets do Amazon S3, consulte Criação de buckets no Manual do usuário do Amazon Simple Storage Service e Como trabalhar com buckets do Amazon S3 no Manual do usuário do Amazon Simple Storage Service.

2. Crie as duas filas do Amazon SQS:

   a. Abra o console do Amazon SQS.

   b. Selecione Criar fila.

   c. Em Tipo, escolha Padrão.

   d. Em Nome, digite **`ticketPaymentQueue`**.

   e. Em Access policy (Política de acesso), para Choose method (Escolher método), escolha Advanced (Avançado).

   f. Na caixa JSON policy (Política de JSON), cole a seguinte política:

```
{
```

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

Nessa política de acesso, substitua o Conta da AWS número (*123456789012*) pelo seu e altere a AWS Região (*us-east-1*) de acordo.

g. Selecione Criar fila.

h. Repita essas etapas para criar uma segunda fila do SQS chamada **ticketFraudQueue**.

Para obter mais informações sobre a criação de filas do SQS, consulte Creating an Amazon SQS queue (console) no Guia do desenvolvedor do Amazon Simple Queue Service.

3. Criar o tópico do SNS:

a. Abra a página Topics (Tópicos) no console do Amazon SNS.

b. Escolha Criar tópico.

c. Em Details (Detalhes), para Type (Tipo), escolha Standard (Padrão).

d. Em Nome, digite **ticketTopic**.

e. Escolha Criar tópico.

Para obter mais informações sobre a criação de tópicos do SNS, consulte Criar um tópico do Amazon SNS.

4. Inscreva as duas filas do SQS no tópico do SNS:

a. No console do Amazon SNS, na página de detalhes do tópico ticketTopic, escolha Create subscription (Criar assinatura).

b. Em Details (Detalhes), para Protocol (Protocolo), escolha Amazon SQS.

c. Para Endpoint, escolha o Amazon Resource Name (ARN) da ticketPaymentQueuefila.

d. Selecione Criar assinatura.

e. Repita essas etapas para criar uma segunda assinatura usando o ARN da ticketFraudQueuefila.

Para obter mais informações sobre a inscrição em tópicos do SNS, consulte Criação de uma assinatura em um tópico do Amazon SNS. Você também pode assinar filas SQS para tópicos do SNS no console do Amazon SQS. Para obter mais informações,consulte Subscribing an Amazon SQS queue to an Amazon SNS topic (console) no Guia do desenvolvedor do Amazon Simple Queue Service.

Você criou os recursos iniciais para este exemplo de caso de uso. Para continuar, consulte Configurar um stream de entrega do Firehose para arquivamento de mensagens do Amazon SNS.

Configurar um stream de entrega do Firehose para arquivamento de mensagens do Amazon SNS

Este tópico explica como criar o stream de entrega do Amazon Data Firehose para o exemplo de caso de uso de arquivamento e análise de mensagens.

Para criar o fluxo de entrega do Firehose

1. Abra o console de serviços do Amazon Kinesis.
2. Escolha Firehose e, depois, escolha Criar fluxo de entrega.
3. Na página Novo fluxo de entrega, em Nome do fluxo de entrega, insira **`ticketUploadStream`** e, depois, escolha Próximo.
4. Na página Process records (Processar registros), escolha Next Step (Próxima etapa).
5. Na página Choose a Destination (Selecionar um destino), faça o seguinte:

   a. Para Destination (Destino), escolha Amazon S3.

   b. Em S3 destination (Destino do S3), em S3 bucket (Bucket do S3), escolha o bucket do S3 que você criou inicialmente.

c. Escolha Próximo.

6. Na página Configure settings (Definir configurações), para S3 buffer conditions (Condições de buffer do S3), faça o seguinte:

   - Em Buffer size (Tamanho do buffer), insira **1**.
   - Em Buffer interval (Intervalo buffer), insira **60**.

   O uso desses valores para o buffer do Amazon S3 permite testar rapidamente a configuração. A primeira condição que é atendida aciona a entrega de dados para o bucket do S3.

7. Na página Definir configurações, em Permissões, escolha criar uma função AWS Identity and Access Management (IAM) com as permissões necessárias atribuídas automaticamente. Escolha Próximo.
8. Na página Review (Revisar), selecione Create delivery stream (Criar fluxo de entrega).
9. Na página de streams de entrega do Kinesis Data Firehose, escolha o stream de entrega que você acabou de criar (). ticketUploadStream Na guia Details (Detalhes), anote o nome do recurso da Amazon (ARN) do fluxo para mais tarde.

Para obter mais informações sobre a criação de fluxos de entrega, consulte Como criar um fluxo de entrega do Amazon Firehose Data no Guia do desenvolvedor do Amazon Data Firehose. Para obter mais informações sobre a criação de funções do IAM, consulte Como criar uma função para delegar permissões a um AWS serviço no Guia do usuário do IAM.

Você criou o fluxo de entrega do Firehose com as permissões necessárias. Para continuar, consulte Assinar o fluxo de entrega do Firehose no tópico do Amazon SNS.

Assinar o fluxo de entrega do Firehose no tópico do Amazon SNS

Este tópico explica como criar os seguintes recursos para o exemplo de caso de uso de arquivamento e análise de mensagens:

- A função AWS Identity and Access Management (IAM) que permite que a assinatura do Amazon SNS coloque registros no stream de entrega do Amazon Data Firehose.
- A assinatura do stream de entrega do Firehose para o tópico Amazon SNS.

Para criar o perfil do IAM para a assinatura do Amazon SNS

1. Abra a página Funções no console do IAM.
2. Selecione Criar função.
3. Em Selecionar tipo de entidade confiável, selecione serviço da AWS .
4. Para Choose a use case (Escolher um caso de uso), escolha SNS. Então, escolha Próximo: permissões.
5. Escolha Próximo: etiquetas.
6. Selecione Próximo: revisar.
7. Na página Revisar, em Nome da função, insira **`ticketUploadStreamSubscriptionRole`**. Então, escolha Criar perfil.
8. Quando a função for criada, escolha seu nome (ticketUploadStreamSubscriptionRole).
9. Na página Summary (Resumo), escolha Add inline policy (Adicionar política inline).
10. Na página Create policy (Criar política), escolha a guia JSON e cole a seguinte política JSON na caixa:

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

Nessa política, substitua o Conta da AWS número (*123456789012*) pelo seu e altere a AWS Região (*us-east-1*) de acordo.

11. Escolha Revisar política.
12. Na página Review policy (Revisar política), em Name (Nome), insira **`FirehoseSnsPolicy`**. Selecione Criar política.
13. Na página Summary (Resumo) da função, anote o Role ARN (ARN da função) para mais tarde.

Para obter mais informações sobre a criação de funções do IAM, consulte Como criar uma função para delegar permissões a um AWS serviço no Guia do usuário do IAM.

Para inscrever o fluxo de entrega do Firehose no tópico do SNS

1. Abra a página Topics (Tópicos) no console do Amazon SNS.
2. Na guia Assinaturas, escolha Criar assinatura.
3. Em Detalhes, em Protocolo, escolha Amazon Data Firehose.
4. Para Endpoint, insira o Amazon Resource Name (ARN) ticketUploadStreamdo stream de entrega que você criou anteriormente. Por exemplo, digite **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Em ARN da função de assinatura, insira o ARN da função do ticketUploadStreamSubscriptionRoleIAM que você criou anteriormente. Por exemplo, digite **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Marque a caixa de seleção Enable raw message delivery (Habilitar a entrega de mensagens).
7. Selecione Create subscription.

Você criou a função do IAM e a assinatura do tópico do SNS. Para continuar, consulte Testar e consultar uma configuração do Amazon SNS para um gerenciamento de dados eficaz.

Testar e consultar uma configuração do Amazon SNS para um gerenciamento de dados eficaz

Este tópico explica como testar o exemplo de caso de uso de arquivamento e análise de mensagens publicando uma mensagem no tópico do Amazon SNS. As instruções incluem uma consulta de exemplo que você pode executar e se adaptar às suas próprias necessidades.

Para testar sua configuração

1. Abra a página Topics (Tópicos) no console do Amazon SNS.
2. Selecione o tópico **`ticketTopic`**.

3. Selecione Publish message (Publicar mensagem).
4. Na página Publicar mensagem no tópico, insira o seguinte para o corpo da mensagem. Adicione um caractere de nova linha no final da mensagem.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

   Mantenha todas as outras opções como seus valores padrão.
5. Selecione Publish message (Publicar mensagem).

   Para obter mais informações sobre publicação de mensagens, consulte Publicar uma mensagem do Amazon SNS.
6. Após o intervalo de fluxo de entrega de 60 segundos, abra a caixa de diálogo Console do Amazon Simple Storage Service (Amazon S3) e escolha o bucket do Amazon S3 que você criou inicialmente.

   A mensagem publicada é exibida no bucket.

Para consultar os dados

1. Abra o console do Amazon Athena.
2. Execute uma consulta.

   Por exemplo, suponha que a tabela `notifications` no esquema `default` contenha os seguintes dados:

```
{"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"}
```

   Para localizar o destino superior, execute a seguinte consulta:

```
SELECT destination
```

```
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

Para consultar tickets vendidos durante um intervalo de data e hora específico, execute uma consulta como a seguinte:

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

Você pode adaptar ambas as consultas de exemplo para suas próprias necessidades. Para obter mais informações sobre como usar o Athena para executar consultas, consulte Conceitos básicos no Manual do usuário do Amazon Athena.

Liberar

Para evitar incorrer em cobranças de uso depois de terminar o teste, exclua os seguintes recursos criados durante o tutorial:

- Assinaturas do Amazon SNS
- Tópico do Amazon SNS
- Filas do Amazon Simple Queue Service (Amazon SQS)
- Bucket do Amazon S3
- Fluxo de entrega do Amazon Data Firehose
- AWS Identity and Access Management Funções e políticas (IAM)

Automatizar o arquivamento de mensagens do Amazon SNS com um modelo AWS CloudFormation

Para automatizar a implantação do caso de uso de exemplo de arquivamento e análise de mensagens do Amazon SNS, você pode usar o seguinte modelo YAML:

```
---
```

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Template for creating an SNS archiving use case
Resources:
  ticketUploadStream:
    DependsOn:
    - ticketUploadStreamRolePolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN: !Sub 'arn:${AWS::Partition}:s3:::${ticketArchiveBucket}'
        BufferingHints:
          IntervalInSeconds: 60
          SizeInMBs: 1
        CompressionFormat: UNCOMPRESSED
        RoleARN: !GetAtt ticketUploadStreamRole.Arn
  ticketArchiveBucket:
    Type: AWS::S3::Bucket
  ticketTopic:
    Type: AWS::SNS::Topic
  ticketPaymentQueue:
    Type: AWS::SQS::Queue
  ticketFraudQueue:
    Type: AWS::SQS::Queue
  ticketQueuePolicy:
    Type: AWS::SQS::QueuePolicy
    Properties:
      PolicyDocument:
        Statement:
          Effect: Allow
          Principal:
            Service: sns.amazonaws.com
          Action:
            - sqs:SendMessage
          Resource: '*'
          Condition:
            ArnEquals:
              aws:SourceArn: !Ref ticketTopic
      Queues:
        - !Ref ticketPaymentQueue
        - !Ref ticketFraudQueue
  ticketUploadStreamSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
```

```
      Endpoint: !GetAtt ticketUploadStream.Arn
      Protocol: firehose
      SubscriptionRoleArn: !GetAtt ticketUploadStreamSubscriptionRole.Arn
  ticketPaymentQueueSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketPaymentQueue.Arn
      Protocol: sqs
  ticketFraudQueueSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketFraudQueue.Arn
      Protocol: sqs
  ticketUploadStreamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Sid: ''
          Effect: Allow
          Principal:
            Service: firehose.amazonaws.com
          Action: sts:AssumeRole
  ticketUploadStreamRolePolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: FirehoseticketUploadStreamRolePolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - s3:AbortMultipartUpload
          - s3:GetBucketLocation
          - s3:GetObject
          - s3:ListBucket
          - s3:ListBucketMultipartUploads
          - s3:PutObject
          Resource:
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}'
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}/*'
```

```
      Roles:
      - !Ref ticketUploadStreamRole
  ticketUploadStreamSubscriptionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - sns.amazonaws.com
          Action:
          - sts:AssumeRole
      Policies:
      - PolicyName: SNSKinesisFirehoseAccessPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Action:
            - firehose:DescribeDeliveryStream
            - firehose:ListDeliveryStreams
            - firehose:ListTagsForDeliveryStream
            - firehose:PutRecord
            - firehose:PutRecordBatch
            Effect: Allow
            Resource:
            - !GetAtt ticketUploadStream.Arn
```

Fanout notificações do Amazon SNS para funções do Lambda para processamento automatizado

O Amazon SNS se integra ao AWS Lambda, permitindo que você acione funções Lambda em resposta às notificações do Amazon SNS. Quando uma mensagem é publicada em um tópico do SNS que tem uma função do Lambda inscrita, a função do Lambda é chamada com a carga útil da mensagem publicada. A função Lambda recebe a carga da mensagem como um parâmetro de entrada e pode manipular as informações na mensagem, publicá-la em outros tópicos do SNS ou enviá-la para outros serviços. AWS

O Amazon SNS também oferece suporte a atributos de status de entrega para notificações por mensagens enviadas a endpoints do Lambda. Para obter mais informações, consulte Status de entrega de mensagens do Amazon SNS.

Tópicos

Pré-requisitos para integrar o Amazon SNS às funções do Lambda em todas as regiões

Para chamar funções do Lambda usando notificações do Amazon SNS, você precisa do seguinte:

- Uma função Lambda
- Um tópico do Amazon SNS

Para obter informações sobre a criação de uma função do Lambda para usar com o Amazon SNS, consulte Usar o Lambda com o Amazon SNS. Para obter informações sobre a criação de um tópico do Amazon SNS, consulte Criar um tópico.

Quando você usa o Amazon SNS para entregar mensagens de regiões de adesão para regiões habilitadas por padrão, é necessário alterar a política criada na função do AWS Lambda substituindo o principal `sns.amazonaws.com` por `sns.<opt-in-region>.amazonaws.com`.

Por exemplo, se você quiser inscrever uma função do Lambda no Leste dos EUA (Norte da Virgínia) em um tópico do SNS na Ásia-Pacífico (Hong Kong), altere o principal na política de função do AWS Lambda para `sns.ap-east-1.amazonaws.com`. As regiões de adesão incluem as regiões lançadas após 20 de março de 2019, que abrangem Ásia-Pacífico (Hong Kong), Oriente Médio (Bahrein), UE (Milão) e África (Cidade do Cabo). As regiões lançadas antes de 20 de março de 2019 são habilitadas por padrão.

> **Note**
>
> AWS não oferece suporte à entrega entre regiões para o Lambda de uma região habilitada por padrão para uma região opcional. Além disso, não há suporte para o encaminhamento de mensagens do SNS entre duas regiões de adesão.

Assinar tópico do Amazon SNS de uma função do Lambda

Este tópico explica como inscrever uma função Lambda em um tópico do Amazon SNS, permitindo que a função seja acionada por mensagens publicadas.

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Na página Topics (Tópicos), escolha um tópico.
4. Na seção Inscrições, escolha Criar inscrição.
5. Na página Criar assinatura, na seção Detalhes, faça o seguinte:
    a. Verifique o ARN do tópico escolhido.
    b. Para Protocolo, escolha AWS Lambda.
    c. Em Endpoint, insira o ARN de uma função.
    d. Selecione Create subscription.

Quando uma mensagem é publicada em um tópico do SNS que tem uma função do Lambda inscrita, a função do Lambda é chamada com a carga útil da mensagem publicada. Para obter informações sobre como usar AWS Lambda com o Amazon SNS, incluindo um tutorial, consulte Usando AWS Lambda com o Amazon SNS.

Fanout de notificações do Amazon SNS para filas do Amazon SQS para processamento assíncrono

O Amazon SNS funciona em conjunto com o Amazon Simple Queue Service (Amazon SQS). Esses serviços oferecem benefícios diferentes para os desenvolvedores. O Amazon SNS permite que aplicativos enviem mensagens urgentes para vários inscritos através de um mecanismo de “push”, eliminando a necessidade de verificar ou “pesquisar” periodicamente atualizações. O Amazon SQS é um serviço de fila de mensagens usado por aplicativos distribuídos para trocar mensagens por meio de um modelo de pesquisa e pode ser usado para decompor os componentes de envio e recebimento — sem a necessidade de cada componente estar disponível simultaneamente. Ao usar o Amazon SNS e o Amazon SQS em conjunto, as mensagens podem ser entregues a aplicativos que exigem uma notificação imediata de um evento e também podem continuar em uma fila do Amazon SQS para que outros aplicativos as processem posteriormente.

Ao inscrever uma fila do Amazon SQS em um tópico do Amazon SNS, é possível publicar uma mensagem no tópico e o Amazon SNS envia uma mensagem do Amazon SQS para a fila inscrita. A mensagem do Amazon SQS contém o assunto e a mensagem publicados no tópico juntamente com os metadados sobre a mensagem em um documento JSON. A mensagem do Amazon SQS será semelhante ao seguinte documento JSON.

```
{
   "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"
}
```

Inscrever uma fila do Amazon SQS em um tópico do Amazon SNS

Para permitir que um tópico do Amazon SNS envie mensagens para uma fila do Amazon SQS, escolha uma das seguintes opções:

- Use o console do Amazon SQS, o que simplifica o processo. Para obter mais informações,consulte Subscribing an Amazon SQS queue to an Amazon SNS topic no Guia do desenvolvedor do Amazon Simple Queue Service.
- Use as seguintes etapas:
  1. Obtenha o Nome de recurso da Amazon (ARN) da fila para a qual deseja enviar mensagens e o tópico no qual você deseja inscrever a fila.
  2. Conceda a permissão `sqs:SendMessage` ao tópico do Amazon SNS para que ele possa enviar mensagens para a fila.
  3. Inscreva a fila no tópico do Amazon SNS.
  4. Forneça aos usuários do IAM ou às Contas da AWS as permissões apropriadas para publicar no tópico do Amazon SNS e ler mensagens da fila do Amazon SQS.

5. Faça um teste publicando uma mensagem no tópico e lendo a mensagem a partir da fila.

Para saber mais sobre como configurar um tópico para enviar mensagens a uma fila que está em uma conta da AWS diferente, consulte Enviar mensagens do Amazon SNS para uma fila do SQS em uma conta diferente.

Para ver um AWS CloudFormation modelo que cria um tópico que envia mensagens para duas filas, consulteAutomatize as mensagens do Amazon SNS para o Amazon SQS com AWS CloudFormation.

Etapa 1: Obter o ARN da fila e do tópico

Ao inscrever uma fila em seu tópico, você precisará de uma cópia do ARN da fila. Da mesma forma, ao conceder permissão para o tópico enviar mensagens para a fila, você precisará de uma cópia do ARN do tópico.

Para obter o ARN da fila, você pode usar o console do Amazon SQS ou a ação da API. GetQueueAttributes

Para obter o ARN da fila do console do Amazon SQS

1. Faça login no AWS Management Console e abra o console do Amazon SQS em. https://console.aws.amazon.com/sqs/
2. Marque a caixa de seleção da fila cujo ARN você deseja obter.
3. Na seção Details (Detalhes), copie o valor do ARN e use-o para se inscrever no tópico do Amazon SSN.

Para obter o ARN do tópico, use o console do Amazon SNS, o comando `sns-get-topic-attributes` ou a ação da API `GetQueueAttributes`.

Para obter o ARN do tópico do console do Amazon SNS

1. Faça login no console do Amazon SNS.
2. No painel de navegação, selecione o tópico do qual você deseja obter o ARN.
3. Na seção Details (Detalhes), copie o valor de ARN a fim de usá-lo para conceder permissão ao tópico do Amazon SNS para enviar mensagens à fila.

Etapa 2: Conceder permissão ao tópico do Amazon SNS para enviar mensagens à fila do Amazon SQS

Para que um tópico do Amazon SNS possa de enviar mensagens a uma fila, você deve definir uma política na fila que permita que o tópico do Amazon SNS execute a ação `sqs:SendMessage`.

Antes de inscrever uma fila em um tópico, você precisa de um tópico e uma fila. Se ainda não criou um tópico ou uma fila, crie-os agora. Para obter mais informações, consulte Criar um tópico e Criar uma fila no Guia do desenvolvedor do Amazon Simple Queue Service.

Para definir uma política em uma fila, você pode usar o console do Amazon SQS ou SetQueueAttributesa ação da API. Antes de começar, verifique se você tem o ARN do tópico que você deseja permitir que envie mensagens para a fila. Se você estiver assinando uma fila para vários tópicos, sua política deverá conter um elemento `Statement` para cada tópico.

Para definir uma SendMessage política em uma fila usando o console do Amazon SQS

1. Faça login no AWS Management Console e abra o console do Amazon SQS em. https://console.aws.amazon.com/sqs/
2. Marque a caixa da fila cuja política você deseja definir, escolha a guia Access policy (Política de acesso) e escolha Edit (Editar).
3. NoPolítica de acesso, defina quem pode acessar sua fila.

   - Adicione uma condição que permite a ação para o tópico.
   - Defina `Principal` como o serviço do Amazon SNS, conforme exibido no exemplo abaixo.
   - Use as chaves de condição globais `aws:SourceArn` ou `aws:SourceAccount` para se proteger contra o cenário confused deputy. Para usar essas chaves de condição, defina o valor como o ARN de seu tópico. Se sua fila estiver inscrita em vários tópicos, use `aws:SourceAccount` em seu lugar.

   Por exemplo, a política a seguir MyTopic permite enviar mensagens para 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"
        }
      }
    }
  ]
}
```

Etapa 3: Inscrever a fila no tópico do Amazon SNS

Para enviar mensagens para uma fila por meio de um tópico, você deve inscrever a fila no tópico do Amazon SNS. Você especifica a fila por meio de sua ARN. Para se inscrever em um tópico, use o console do Amazon SNS, o comando `sns-subscribe` da CLI ou a ação da API `Subscribe`. Antes de começar, verifique se você tem o ARN para a fila que deseja inscrever.

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Na página Topics (Tópicos), escolha um tópico.
4. Na ***MyTopic*** página, na página Assinaturas, escolha Criar assinatura.
5. Na página Criar assinatura, na seção Detalhes, faça o seguinte:

    a. Verifique o ARN do tópico.

    b. Em Protocol (Protocolo), escolha Amazon SQS.

    c. Em Endpoint, insira o ARN de uma fila do Amazon SQS.

    d. Escolha Create Subscription (Criar assinatura).

    Quando a inscrição for confirmada, o ID da inscrição da nova inscrição exibirá o ID da inscrição. Se o proprietário da fila criar a inscrição, ela será automaticamente confirmada e deverá ser ativada quase que imediatamente.

    Geralmente, você inscreverá sua própria fila em seu próprio tópico em sua própria conta. No entanto, você também pode inscrever uma fila de uma conta diferente em seu tópico. Se o usuário que criar a inscrição não for o proprietário da fila (por exemplo, se um usuário da conta

A inscreve uma fila da conta B para um tópico na conta A), a inscrição deverá ser confirmada. Para obter mais informações sobre como inscrever uma fila de uma conta diferente e confirmar a inscrição, consulte Enviar mensagens do Amazon SNS para uma fila do SQS em uma conta diferente.

Etapa 4: Conceder aos usuários permissões para o tópico apropriado e as ações da fila

Você deve usar AWS Identity and Access Management (IAM) para permitir que somente usuários apropriados publiquem no tópico do Amazon SNS e leiam ou excluam mensagens da fila do Amazon SQS. Para obter mais informações sobre como controlar ações em tópicos e filas para usuários do IAM, consulteUsar políticas baseadas em identidade com o Amazon SNS, eIdentity and Access Management no Amazon SQSno Guia do desenvolvedor do Amazon Simple Queue Service.

Há duas maneiras de controlar o acesso a um tópico ou uma fila:

- Adicione uma política para um usuário ou grupo do IAM. A maneira mais simples de conceder aos usuários permissões para tópicos ou filas é criar um grupo e adicionar a política adequada e os usuários ao grupo. É muito mais fácil adicionar e remover usuários de um grupo do que controlar as políticas definidas para usuários individualmente.
- Adicione uma política ao tópico ou à fila. Se você quiser conceder permissões para um tópico ou fila para outra AWS conta, a única maneira de fazer isso é adicionando uma política que tenha como principal a pessoa à qual Conta da AWS você deseja conceder permissões.

Você deve usar o primeiro método para a maioria dos casos (aplicar políticas a grupos e gerenciar permissões para usuários adicionando ou removendo os usuários adequados aos grupos). Se você precisa conceder permissões para um usuário em outra conta, use o segundo método.

Adicionar uma política a um usuário ou grupo do IAM

Se você adicionasse a política a seguir a um usuário ou grupo do IAM, concederia permissão a esse usuário ou membros desse grupo para realizar a `sns:Publish` ação no tópico MyTopic.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
```

```
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Se você adicionasse a política a seguir a um usuário ou grupo do IAM, concederia a esse usuário ou membros desse grupo permissão para realizar as `sqs:DeleteMessage` ações `sqs:ReceiveMessage` e nas filas MyQueue 1 e MyQueue 2.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

Adição de uma política a um tópico ou uma fila

Os exemplos de políticas a seguir mostram como conceder a outra conta permissões para um tópico e uma fila.

> **Note**
>
> Ao conceder outro Conta da AWS acesso a um recurso em sua conta, você também concede aos usuários do IAM que têm acesso em nível de administrador (acesso curinga) permissões a esse recurso. Todos os outros usuários do IAM na outra conta têm automaticamente o acesso negado a seus recursos. Se desejar conceder a usuários específicos do IAM dessa Conta da AWS acesso a seu recurso, a conta ou um usuário do IAM com acesso de nível de administrador deverá delegar permissões para o recurso para esses usuários do IAM. Para obter mais informações sobre a delegação entre contas, consulte Habilitar acesso entre contas no Guia de uso do IAM.

Se você adicionasse a política a seguir a um tópico MyTopic na conta 123456789012, concederia permissão à conta 111122223333 para realizar a ação nesse tópico. `sns:Publish`

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

Se você adicionasse a política a seguir a uma fila MyQueue na conta 123456789012, concederia permissão à conta 111122223333 para realizar as ações e nessa fila. `sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Etapa 5: Testar as assinaturas de fila do tópico

Você pode testar as inscrições de fila de um tópico publicando nele e visualizando a mensagem que ele envia para a fila.

Para publicar um tópico usando o console do Amazon SNS

1. Usando as credenciais do usuário Conta da AWS ou do IAM com permissão para publicar no tópico, faça login no AWS Management Console e abra o console do Amazon SNS em. https://console.aws.amazon.com/sns/
2. No painel de navegação, escolha o tópico e selecione Publicar no tópico.
3. Na caixa Assunto, digite um assunto (por exemplo, **`Testing publish to queue`**), na caixa Mensagem, digite algum texto (por exemplo, **`Hello world!`**) e escolha Publicar mensagem. Será exibida a seguinte mensagem: Your message has been successfully published.

Para visualizar a mensagem do tópico usando o console do Amazon SQS

1. Usando as credenciais do usuário Conta da AWS ou do IAM com permissão para visualizar mensagens na fila, faça login no AWS Management Console e abra o console do Amazon SQS em. https://console.aws.amazon.com/sqs/
2. Escolha uma fila que esteja inscrita no tópico.
3. Escolha Send and receive messages (Enviar e receber mensagens), depois escolha Poll for messages (Pesquisar mensagens). Uma mensagem com um tipo de Notificação é exibida.
4. Na coluna Corpo, escolha Mais detalhes. A caixa Detalhes da mensagem inclui um documento JSON que contém o assunto e a mensagem publicada no tópico. A mensagem é semelhante ao seguinte documento JSON.

```
{
  "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. Escolha Fechar. Você publicou com sucesso em um tópico que envia mensagens de notificação para uma fila.

Automatize as mensagens do Amazon SNS para o Amazon SQS com AWS CloudFormation

AWS CloudFormation permite que você use um arquivo de modelo para criar e configurar uma coleção de AWS recursos juntos como uma única unidade. Esta seção tem um modelo de exemplo que facilita a implantação de tópicos que publicam em filas. Os modelos processam as etapas de configuração para você, criando duas filas, criando um tópico com inscrições das filas, adicionando uma política às filas para que o tópico possa enviar mensagens para as filas e criando usuários e grupos do IAM para controlar o acesso a esses recursos.

Para obter mais informações sobre como implantar AWS recursos usando um AWS CloudFormation modelo, consulte Introdução no Guia do AWS CloudFormation usuário.

Usando um AWS CloudFormation modelo para configurar tópicos e filas em um Conta da AWS

O modelo de exemplo cria um tópico do Amazon SNS que pode enviar mensagens para duas filas do Amazon SQS com as permissões adequadas para que os membros de um grupo do IAM publiquem no tópico e os membros do outro leiam mensagens das filas. O modelo também cria usuários do IAM que são adicionados a cada grupo.

Copie o conteúdo do modelo em um arquivo. Você também pode baixar o modelo na página AWS CloudFormation Modelos. Na página de modelos, escolha Procurar modelos de amostra por AWS serviço e, em seguida, escolha Amazon Simple Queue Service.

SNSTopic O My está configurado para publicar em dois endpoints inscritos, que são duas filas do Amazon SQS MyQueue (1 e 2). MyQueue MyPublishTopicGroup é um grupo do IAM cujos membros têm permissão para publicar no My SNSTopic usando a ação da API Publish ou o comando sns-publish. O modelo cria os usuários do IAM MyPublishUser MyQueueUser e fornece a eles perfis de login e chaves de acesso. O usuário que cria uma pilha com esse modelo especifica as senhas para os perfis de login como parâmetros de entrada. O modelo cria chaves de acesso para os dois usuários do IAM com MyPublishUserKey MyQueueUserKey e. AddUserToMyPublishTopicGroup adiciona MyPublishUser ao MyPublishTopicGroup para que o usuário tenha as permissões atribuídas ao grupo.

My RDMessage QueueGroup é um grupo do IAM cujos membros têm permissão para ler e excluir mensagens das duas filas do Amazon SQS usando as ações ReceiveMessagee DeleteMessageda API. AddUserToMyQueueGroup adiciona MyQueueUser ao Meu RDMessage QueueGroup para que o usuário tenha as permissões atribuídas ao grupo. MyQueuePolicy atribui permissão para SNSTopic que My publique suas notificações nas duas filas.

A lista a seguir mostra o conteúdo do AWS CloudFormation modelo.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template SNSToSQS: This Template creates
 an SNS topic that can send messages to
  two SQS queues with appropriate permissions for one IAM user to publish to the topic
 and another to read messages from the queues.
  MySNSTopic is set up to publish to two subscribed endpoints, which are two SQS queues
 (MyQueue1 and MyQueue2). MyPublishUser is an IAM user
  that can publish to MySNSTopic using the Publish API. MyTopicPolicy assigns that
 permission to MyPublishUser. MyQueueUser is an IAM user
  that can read messages from the two SQS queues. MyQueuePolicy assigns those
 permissions to MyQueueUser. It also assigns permission for
  MySNSTopic to publish its notifications to the two queues. The template creates
 access keys for the two IAM users with MyPublishUserKey
  and MyQueueUserKey. ***Warning*** you will be billed for the AWS resources used if
 you create a stack from this template.",

  "Parameters": {
    "MyPublishUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description": "Password for the IAM user MyPublishUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern": "[a-zA-Z0-9]*",
      "ConstraintDescription": "must contain only alphanumeric characters."
    },
    "MyQueueUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description": "Password for the IAM user MyQueueUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern": "[a-zA-Z0-9]*",
```

```
      "ConstraintDescription": "must contain only alphanumeric characters."
    }
  },


  "Resources": {
    "MySNSTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "Subscription": [{
            "Endpoint": {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "Protocol": "sqs"
          },
          {
            "Endpoint": {
              "Fn::GetAtt": ["MyQueue2", "Arn"]
            },
            "Protocol": "sqs"
          }
        ]
      }
    },
    "MyQueue1": {
      "Type": "AWS::SQS::Queue"
    },
    "MyQueue2": {
      "Type": "AWS::SQS::Queue"
    },
    "MyPublishUser": {
      "Type": "AWS::IAM::User",
      "Properties": {
        "LoginProfile": {
          "Password": {
            "Ref": "MyPublishUserPassword"
          }
        }
      }
    },
    "MyPublishUserKey": {
      "Type": "AWS::IAM::AccessKey",
      "Properties": {
        "UserName": {
```

```
        "Ref": "MyPublishUser"
      }
    }
  },
  "MyPublishTopicGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
        "PolicyName": "MyTopicGroupPolicy",
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Action": [
              "sns:Publish"
            ],
            "Resource": {
              "Ref": "MySNSTopic"
            }
          }]
        }
      }]
    }
  },
  "AddUserToMyPublishTopicGroup": {
    "Type": "AWS::IAM::UserToGroupAddition",
    "Properties": {
      "GroupName": {
        "Ref": "MyPublishTopicGroup"
      },
      "Users": [{
        "Ref": "MyPublishUser"
      }]
    }
  },
  "MyQueueUser": {
    "Type": "AWS::IAM::User",
    "Properties": {
      "LoginProfile": {
        "Password": {
          "Ref": "MyQueueUserPassword"
        }
      }
    }
  },
```

```
    "MyQueueUserKey": {
      "Type": "AWS::IAM::AccessKey",
      "Properties": {
        "UserName": {
          "Ref": "MyQueueUser"
        }
      }
    },
    "MyRDMessageQueueGroup": {
      "Type": "AWS::IAM::Group",
      "Properties": {
        "Policies": [{
          "PolicyName": "MyQueueGroupPolicy",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
              ],
              "Resource": [{
                  "Fn::GetAtt": ["MyQueue1", "Arn"]
                },
                {
                  "Fn::GetAtt": ["MyQueue2", "Arn"]
                }
              ]
            }]
          }
        }]
      }
    },
    "AddUserToMyQueueGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
          "Ref": "MyRDMessageQueueGroup"
        },
        "Users": [{
          "Ref": "MyQueueUser"
        }]
      }
    },
    "MyQueuePolicy": {
```

```
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": {
              "Service": "sns.amazonaws.com"
            },
            "Action": ["sqs:SendMessage"],
            "Resource": "*",
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": {
                  "Ref": "MySNSTopic"
                }
              }
            }
          }]
        },
        "Queues": [{
          "Ref": "MyQueue1"
        }, {
          "Ref": "MyQueue2"
        }]
      }
    }
  },
  "Outputs": {
    "MySNSTopicTopicARN": {
      "Value": {
        "Ref": "MySNSTopic"
      }
    },
    "MyQueue1Info": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "URL:",
            {
```

```
            "Ref": "MyQueue1"
          }
        ]
      ]
    }
  },
  "MyQueue2Info": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyQueue2", "Arn"]
          },
          "URL:",
          {
            "Ref": "MyQueue2"
          }
        ]
      ]
    }
  },
  "MyPublishUserInfo": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyPublishUser", "Arn"]
          },
          "Access Key:",
          {
            "Ref": "MyPublishUserKey"
          },
          "Secret Key:",
          {
            "Fn::GetAtt": ["MyPublishUserKey", "SecretAccessKey"]
          }
        ]
      ]
    }
  },
```

```
    "MyQueueUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueueUser", "Arn"]
            },
            "Access Key:",
            {
              "Ref": "MyQueueUserKey"
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyQueueUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    }
  }
}
```

Fanout de notificações do Amazon SNS para endpoints HTTPS

Use o Amazon SNS para enviar mensagens de notificação a um ou mais endpoints HTTP ou HTTPS. Quando você inscrever um endpoint em um tópico, será possível publicar uma notificação no tópico, e o Amazon SNS envia uma solicitação HTTP POST entregando o conteúdo da notificação para o endpoint inscrito. Ao inscrever o endpoint, você seleciona se o Amazon SNS usa HTTP ou HTTPS para enviar a solicitação POST ao endpoint. Se você usar HTTPS, poderá aproveitar o suporte do Amazon SNS para o seguinte:

- Server Name Indication (SNI): permite que o Amazon SNS ofereça suporte a endpoints HTTPS que exigem SNI, como um servidor que exige vários certificados para hospedar vários domínios. Para obter mais informações sobre SNI, consulte Server Name Indication.

- Autenticação de acesso Basic e Digest: permite que você especifique um nome de usuário e uma senha no URL HTTPS para a solicitação HTTP POST, como `https://user:password@domain.com` ou `https://user@domain.com`. O nome do usuário e a senha

são criptografados pela conexão SSL estabelecida ao usar HTTPS. Somente o nome de domínio é enviado em texto sem formatação. Para obter mais informações sobre autenticação e controle de acesso Basic e Digest, consulte a RFC-2617.

> **Important**
>
> No momento, o Amazon SNS não é compatível com endpoints HTTP(S) privados. HTTPS só podem ser URLs recuperados da ação de API do Amazon `GetSubscriptionAttributes` SNS, para entidades às quais você concedeu acesso à API.

> **Note**
>
> O serviço de cliente deve poder oferecer suporte à resposta de cabeçalho `HTTP/1.1 401 Unauthorized`

A solicitação contém o assunto e a mensagem que foram publicados no tópico juntamente com os metadados sobre a notificação em um documento JSON. A solicitação será semelhante à seguinte solicitação POST HTTP. Para obter detalhes sobre o cabeçalho HTTP e o formato JSON do corpo de solicitação, consulte Cabeçalhos HTTP/HTTPS e Formato JSON de notificação HTTP/HTTPS.

> **Note**
>
> O Amazon SNS considera todos os erros 5XX e 429 (muitas solicitações enviadas) como passíveis de repetição. Esses erros estão sujeitos à política de entrega. Todos os outros erros são considerados falhas permanentes e não serão feitas novas tentativas.

```
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"
}
```

Inscrever um endpoint HTTPS em um tópico do Amazon SNS

Este tópico explica como inscrever endpoints HTTP/S em tópicos do Amazon SNS.

Tópicos

- Etapa 1: Verificar se o endpoint está pronto para processar mensagens do Amazon SNS
- Etapa 2: Inscrever o endpoint HTTP/HTTPS no tópico do Amazon SNS
- Etapa 3: confirmar sua assinatura do Amazon SNS
- Etapa 4: opcional - definir a política de entrega para a assinatura do Amazon SNS.
- Etapa 5: opcional - conceder aos usuários permissões para publicar no tópico do Amazon SNS
- Etapa 6: enviar mensagens do Amazon SNS para o endpoint HTTP/HTTPS

Etapa 1: Verificar se o endpoint está pronto para processar mensagens do Amazon SNS

Antes de inscrever um endpoint HTTP ou HTTPS em um tópico, você deve garantir que ele tem a capacidade de lidar com as solicitações HTTP POST que o Amazon SNS usa para enviar a confirmação de inscrição e as mensagens de notificação. Geralmente, isso significa a criação e

a implantação de uma aplicação web (por exemplo, um servlet Java se o host do endpoint está executando Linux com Apache e Tomcat) que processa solicitações HTTP do Amazon SNS. Quando você inscreve um endpoint HTTP, o Amazon SNS envia uma solicitação de confirmação de inscrição. Seu endpoint deve estar preparado para receber e processar essa solicitação quando você criar a assinatura, porque o Amazon SNS envia essa solicitação nesse momento. O Amazon SNS não enviará notificações ao endpoint até que a assinatura seja confirmada. Assim que você confirmar a inscrição, o Amazon SNS enviará notificações para o endpoint quando uma ação de publicação for realizada no tópico inscrito.

Para configurar o endpoint para processar as mensagens de notificação e de confirmação de inscrição

1. Seu código deve conter os cabeçalhos HTTP de solicitações HTTP POST que o Amazon SNS envia para seu endpoint. O código deve procurar o campo do cabeçalho `x-amz-sns-message-type`, que mostra o tipo de mensagem que o Amazon SNS enviou a você. Ao visualizar o cabeçalho, você pode determinar o tipo da mensagem sem a necessidade de analisar o corpo da solicitação HTTP. Há dois tipos com os quais você precisa lidar: `SubscriptionConfirmation` e `Notification`. A mensagem `UnsubscribeConfirmation` é usada somente quando a assinatura é excluída do tópico.

   Para obter detalhes sobre o cabeçalho HTTP, consulte Cabeçalhos HTTP/HTTPS. A seguinte solicitação HTTP POST é um exemplo de uma mensagem de confirmação de inscrição.

```
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. Seu código deve analisar o documento JSON no corpo da solicitação HTTP POST e o tipo de conteúdo texto/sem formatação para ler os pares de nome/valor que compõem a mensagem do Amazon SNS. Use um analisador JSON que faça a conversão da representação em sequência de escape dos caracteres de controle de volta para seus valores de caracteres ASCII (por exemplo, converter \n para um caractere de nova linha). Você pode usar um analisador JSON existente, como o ProcessadorJackson JSON ou escrever seu próprio analisador. Para enviar o texto nos campos de assunto e mensagem em formato JSON válido, o Amazon SNS deve converter alguns caracteres de controle para representações em sequência de escape para inclusão no documento JSON. Ao receber o documento JSON no corpo da solicitação POST enviada para o endpoint, você deve converter os caracteres em sequência de escape de volta para seus valores de caractere originais se quiser uma representação exata do assunto e das mensagens originais publicadas no tópico. Isso é essencial se você deseja verificar a assinatura de uma notificação, pois a assinatura usa a mensagem e o assunto em seus formulários originais como parte da string para assinar.
3. Seu código deve verificar a autenticidade de uma mensagem de notificação, confirmação de inscrição ou confirmação de cancelamento de inscrição enviada pelo Amazon SNS. o usar informações contidas na mensagem do Amazon SNS, o endpoint pode recriar a assinatura para que você possa verificar o conteúdo da mensagem comparando sua assinatura com a assinatura enviada pelo Amazon SNS junto com a mensagem. Para obter mais informações sobre como verificar a assinatura de uma mensagem, consulte Verificação das assinaturas de mensagens do Amazon SNS.
4. Com base no tipo especificado pelo campo de cabeçalho `x-amz-sns-message-type`, seu código deve ler o documento JSON contido no corpo da solicitação HTTP e processar a mensagem. A seguir, são apresentadas as diretrizes para lidar com os dois tipos principais de mensagens:

SubscriptionConfirmation

Leia o valor para `SubscribeURL` e acesse o URL. Para confirmar a assinatura e começar a receber notificações no endpoint, é necessário visitar o URL`SubscribeURL` (por exemplo, enviando uma solicitação HTTP GET para o URL). Consulte o exemplo de solicitação HTTP na etapa anterior para ver qual a aparência de `SubscribeURL`. Para obter mais informações sobre o formato da mensagem `SubscriptionConfirmation`, consulte Formato JSON de confirmação de assinatura HTTP/HTTPS. Ao acessar o URL, você receberá de volta uma resposta que se parece com o seguinte documento XML. O documento retorna o ARN da inscrição para o endpoint no elemento `ConfirmSubscriptionResult`.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55</
SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

Como alternativa à visita ao`SubscribeURL`, você pode confirmar a assinatura usando a ConfirmSubscriptionação com o valor correspondente `Token` definido na `SubscriptionConfirmation` mensagem. Se você deseja permitir que somente o proprietário do tópico e o proprietário da inscrição cancelem o recebimento para o endpoint, chame a ação `ConfirmSubscription` com uma assinatura AWS .

Notificação

Leia os valores de `Subject` e `Message` para obter as informações da notificação que foi publicada no tópico.

Para obter mais detalhes sobre o formato da mensagem `Notification`, consulte Cabeçalhos HTTP/HTTPS. A seguinte solicitação HTTP POST é um exemplo de uma mensagem de notificação enviada para o endpoint example.com.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
```

```
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
    Content-Length: 773
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

5. Certifique-se de que o endpoint responde à mensagem de HTTP POST do Amazon SNS com o código de status apropriado. A conexão atingirá o tempo limite em aproximadamente 15 segundos. Se o endpoint não responder antes da conexão atingir o tempo limite ou retornar um código de status fora do intervalo de 200–4xx, o Amazon SNS considerará que houve falha na entrega da mensagem.
6. Certifique-se de que o seu código pode lidar com novas tentativas de entrega de mensagens do Amazon SNS. Se o Amazon SNS não receber uma resposta bem-sucedida do endpoint, ele tenta entregar a mensagem novamente. Isso se aplica a todas as mensagens, incluindo a mensagem de confirmação de inscrição. Por padrão, se a entrega inicial da mensagem falhar, o Amazon SNS faz até três novas tentativas com uma espera de 20 segundos entre as tentativas malsucedidas.

> **Note**
>
> A solicitação da mensagem atinge o tempo limite de aproximadamente 15 segundos. Isso significa que se a falha na entrega da mensagem for causada por exceder o tempo

limite, o Amazon SNS tentará outra vez por aproximadamente 35 segundos após a última tentativa de entrega. Você pode definir uma política de entrega diferente para o endpoint.

O Amazon SNS usa o campo de cabeçalho `x-amz-sns-message-id` para identificar de forma exclusiva cada mensagem publicada em um tópico do Amazon SNS. Ao comparar as IDs mensagens que você processou com as mensagens recebidas, você pode determinar se a mensagem é uma tentativa de nova tentativa.

7. Se estiver inscrevendo um endpoint HTTPS, verifique se o endpoint tem um certificado do servidor de uma autoridade de certificado (CA) confiável. O Amazon SNS só envia mensagens a endpoints HTTPS que têm um certificado de servidor assinado por uma CA confiável para o Amazon SNS.
8. Implante o código que você criou para receber mensagens do Amazon SNS. Quando você assinar o endpoint, ele deve estar pronto para receber pelo menos a mensagem de confirmação de inscrição.

Etapa 2: Inscrever o endpoint HTTP/HTTPS no tópico do Amazon SNS

Para enviar mensagens a um endpoint HTTP ou HTTPS por meio de um tópico, inscreva o endpoint no tópico do Amazon SNS. Você especifica o endpoint usando seu URL. Para inscrever-se em um tópico, você pode usar o console do Amazon SNS, o comando sns-subscribe ou a ação da API Subscribe. Antes de começar, verifique se você tem o URL do endpoint que deseja inscrever e se o endpoint está preparado para receber a confirmação e as mensagens de notificação, como descrito na Etapa 1.

Para inscrever um endpoint HTTP ou HTTPS em um tópico usando o console do Amazon SNS

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Subscriptions (Assinaturas).
3. Selecione Create subscription (Criar assinatura).
4. Na lista suspensa Protocolo, selecione HTTP ou HTTPS.
5. Na caixa Endpoint, cole a URL do endpoint para o qual você deseja que o tópico envie mensagens e escolha Create subscription (Criar inscrição).
6. A mensagem de confirmação é exibida. Escolha Fechar.

O ID de assinatura da sua nova assinatura é exibido PendingConfirmation. Quando você confirmar a inscrição, o ID da inscrição exibirá o ID da inscrição.

Etapa 3: confirmar sua assinatura do Amazon SNS

Para confirmar sua assinatura do Amazon SNS, siga estas etapas para garantir que seu endpoint possa receber mensagens com sucesso. Esse processo envolve configurar seu endpoint para lidar com as mensagens de confirmação recebidas, recuperar o URL de confirmação e confirmar a assinatura. Você pode confirmar a assinatura automática ou manualmente, dependendo da sua configuração.

1. Depois de assinar um tópico do Amazon SNS, o Amazon SNS envia uma mensagem de confirmação para o seu endpoint. Essa mensagem contém um `SubscribeURL` que você deve usar para confirmar a assinatura.
2. Seu endpoint deve estar configurado para ouvir as mensagens recebidas do Amazon SNS. Quando a mensagem de confirmação chegar, extraia a **`SubscribeURL`**da mensagem.
3. Depois de ter o`SubscribeURL`, você pode confirmar a assinatura de uma das duas maneiras:

   - Confirmação automática — Seu endpoint pode confirmar automaticamente a assinatura enviando uma solicitação HTTP GET para o. `SubscribeURL`

     Este método não requer intervenção manual.
   - Confirmação manual — Se a confirmação automática não estiver configurada, copie a mensagem **`SubscribeURL`**de confirmação e cole-a na barra de endereço do seu navegador.

     Isso confirmará a assinatura manualmente.
4. Você também pode verificar o status da assinatura usando o console do Amazon SNS:

   a. Faça login no console do Amazon SNS.

   b. No painel de navegação, escolha Assinaturas.

   c. Encontre sua assinatura na lista.

      - Se confirmado, o `SubscriptionArn` será exibido.
      - Se ainda não for confirmado, ele será exibido como`PendingConfirmation`.

Etapa 4: opcional - definir a política de entrega para a assinatura do Amazon SNS.

Por padrão, se a entrega inicial da mensagem falhar, o Amazon SNS faz até três novas tentativas com uma espera de 20 segundos entre as tentativas malsucedidas. Conforme foi dito na Etapa 1, o endpoint deve ter o código que pode processar novas tentativas de mensagens. Ao definir a política de entrega em um tópico ou inscrição, você pode controlar a frequência e o intervalo com que o Amazon SNS tentará entregar novamente mensagens com falha. Você também pode especificar o tipo de conteúdo para as notificações HTTP/S em `DeliveryPolicy`. Para obter mais informações, consulte Como criar uma política de entrega HTTP/S.

Etapa 5: opcional - conceder aos usuários permissões para publicar no tópico do Amazon SNS

Por padrão, o proprietário do tópico tem permissão para publicar no tópico. Para permitir que outros usuários ou aplicativos publiquem no tópico, você deve usar AWS Identity and Access Management (IAM) para dar permissão de publicação ao tópico. Para obter mais informações sobre como conceder permissões para ações do Amazon SNS a usuários do IAM, consulte Usar políticas baseadas em identidade com o Amazon SNS.

Há duas maneiras de controlar o acesso a um tópico:

- Adicione uma política para um usuário ou grupo do IAM. A maneira mais simples de conceder aos usuários permissões para tópicos é criar um grupo e adicionar a política adequada e os usuários ao grupo. É muito mais fácil adicionar e remover usuários de um grupo do que controlar as políticas definidas para usuários individualmente.
- Adicione uma política ao tópico. Se deseja conceder permissões para um tópico para outra AWS , a única maneira de fazer isso é adicionando uma política com a Conta da AWS para a qual você deseja conceder permissões como principal.

Você deve usar o primeiro método para a maioria dos casos (aplicar políticas a grupos e gerenciar permissões para usuários adicionando ou removendo os usuários adequados aos grupos). Se você precisa conceder permissões a um usuário em outra conta, use o segundo método.

Se você adicionasse a política a seguir a um usuário ou grupo do IAM, concederia permissão a esse usuário ou membros desse grupo para realizar a `sns:Publish` ação no tópico MyTopic.

```
{
  "Statement":[{
    "Sid":"AllowPublishToMyTopic",
```

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

O exemplo a seguir de política mostra como conceder permissão à outra conta para um tópico.

> **Note**
>
> Ao conceder outro Conta da AWS acesso a um recurso em sua conta, você também concede aos usuários do IAM que têm acesso em nível de administrador (acesso curinga) permissões a esse recurso. Todos os outros usuários do IAM na outra conta têm automaticamente o acesso negado a seus recursos. Se você quiser dar a usuários específicos do IAM esse Conta da AWS acesso ao seu recurso, a conta ou um usuário do IAM com acesso em nível de administrador deve delegar permissões para o recurso a esses usuários do IAM. Para obter mais informações sobre a delegação entre contas, consulte Habilitar acesso entre contas no Guia de uso do IAM.

Se você adicionasse a política a seguir a um tópico MyTopic na conta 123456789012, concederia permissão à conta 111122223333 para realizar a ação nesse tópico. `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"
  }]
}
```

Etapa 6: enviar mensagens do Amazon SNS para o endpoint HTTP/HTTPS

É possível enviar uma mensagem para inscrições em um tópico publicando no tópico. Para publicar um tópico, use o console do Amazon SNS, o comando `sns-publish` da CLI ou a API `Publish`.

Se seguiu a Etapa 1, o código que você implantou no endpoint deve processar a notificação.

Para publicar um tópico usando o console do Amazon SNS

1. Usando as credenciais do usuário Conta da AWS ou do IAM com permissão para publicar no tópico, faça login no AWS Management Console e abra o console do Amazon SNS em. https://console.aws.amazon.com/sns/
2. No painel de navegação, escolha Topics (Tópicos) e selecione um tópico.
3. Escolha o botão Publicar mensagem.
4. Na caixa Subject (Assunto), insira um assunto (por exemplo, **`Testing publish to my endpoint`**).
5. Na caixa Message (Mensagem), insira texto (por exemplo, **`Hello world!`**) e escolha Publish message (Publicar mensagem).

   Será exibida a seguinte mensagem: Your message has been successfully published.

Verificação das assinaturas de mensagens do Amazon SNS

O Amazon SNS usa assinaturas de mensagens para confirmar a autenticidade das mensagens enviadas ao seu endpoint HTTP. Para garantir a integridade da mensagem e evitar falsificações, você deve verificar a assinatura antes de processar qualquer mensagem do Amazon SNS.

Quando você deve verificar as assinaturas do Amazon SNS?

Você deve verificar as assinaturas de mensagens do Amazon SNS nos seguintes cenários:

- Quando o Amazon SNS envia uma mensagem de notificação para seu endpoint HTTP (S).
- Quando o Amazon SNS envia uma mensagem de confirmação para seu endpoint após uma chamada `Subscribe`ou `Unsubscribe`uma chamada de API.

O Amazon SNS oferece suporte a duas versões de assinatura:

- SignatureVersion1 — Usa um SHA1 hash da mensagem.
- SignatureVersion2 — Usa um SHA256 hash da mensagem. Isso fornece maior segurança e é a opção recomendada.

Para verificar corretamente as assinaturas de mensagens do SNS, siga estas práticas recomendadas:

- Sempre recupere o certificado de assinatura usando HTTPS para evitar ataques de interceptação não autorizados.
- Verifique se o certificado foi emitido pelo Amazon SNS.
- Confirme se a cadeia de confiança do certificado é válida.
- O certificado deve vir de uma URL assinada pelo SNS.
- Não confie em nenhum certificado fornecido na mensagem sem validação.
- Rejeite qualquer mensagem com uma mensagem inesperada `TopicArn` para evitar falsificação.
- O AWS SDKs for Amazon SNS fornece lógica de validação integrada, reduzindo o risco de implementação incorreta.

Configurando a versão da assinatura da mensagem nos tópicos do Amazon SNS

Configurar a versão da assinatura da mensagem nos tópicos do Amazon SNS permite que você aprimore a segurança e a compatibilidade do seu processo de verificação de mensagens.

Selecione entre `SignatureVersion` 1 (SHA1) e `SignatureVersion` 2 (SHA256) para controlar o algoritmo de hash usado para assinar mensagens. Os tópicos do Amazon SNS são padronizados para **`SignatureVersion`** 1. Você pode definir essa configuração usando a ação `SetTopicAttributes`da API.

Use o exemplo a seguir para definir o atributo de tópico `SignatureVersion` usando o AWS CLI:

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

Verificando a assinatura de uma mensagem do Amazon SNS ao usar solicitações baseadas em consultas HTTP

A verificação da assinatura de uma mensagem do Amazon SNS ao usar solicitações baseadas em consultas HTTP garante a autenticidade e a integridade da mensagem. Esse processo confirma que a mensagem é originária do Amazon SNS e não foi adulterada durante o trânsito. Ao analisar a mensagem, criar a string correta para assinar e validar a assinatura em relação a uma chave pública confiável, você protege seu sistema contra falsificações e alterações não autorizadas de mensagens.

1. Extraia pares de valores-chave do documento JSON no corpo da solicitação HTTP POST enviada pelo Amazon SNS. Esses campos são obrigatórios para construir a string a ser assinada.

   - `Message`
   - `Subject` (se presente)
   - `MessageId`
   - `Timestamp`
   - `TopicArn`
   - `Type`

   Por exemplo:

   ```
   MESSAGE_FILE="message.json"
   FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
   ```

   > **Note**
   >
   > Se algum campo contiver caracteres de escape (por exemplo,\n), converta-os em seu formato original para garantir uma correspondência exata.

2. Localize o `SigningCertURL` campo na mensagem do Amazon SNS. Esse certificado contém a chave pública necessária para verificar a assinatura da mensagem. Por exemplo:

   ```
   SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
   ```

3. Certifique-se de que `SigningCertURL` seja de um AWS domínio confiável (por exemplo, https://sns.us-east-1.amazonaws.com). Rejeite qualquer AWS domínio URLs externo por motivos de segurança.

4. Faça o download do certificado X.509 do URL fornecido. Por exemplo:

   ```
   curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
   ```

5. Extraia a chave pública do certificado X.509 baixado. A chave pública permite decifrar a assinatura da mensagem e verificar sua integridade. Por exemplo:

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. Diferentes tipos de mensagem exigem pares de valores-chave diferentes na cadeia de caracteres para serem assinados. Identifique o tipo de mensagem (`Type`campo na mensagem do Amazon SNS) para determinar quais pares de valores-chave incluir:

   - Mensagem de notificação — Inclui `Message`,`MessageId`, `Subject` (se presente) `Timestamp`,`TopicArn`, `Type` e.
   - SubscriptionConfirmationou UnsubscribeConfirmation mensagem — Inclui `Message`,`MessageId`,`SubscribeURL`, `Timestamp`,`Token`,`TopicArn`, `Type` e.

7. O Amazon SNS exige que a string seja assinada para seguir uma ordem de campo fixa e rígida para verificação. Somente os campos explicitamente obrigatórios devem ser incluídos — nenhum campo extra pode ser adicionado. Campos opcionais`Subject`, como, devem ser incluídos somente se estiverem presentes na mensagem e devem aparecer na posição exata definida pela ordem dos campos obrigatórios. Por exemplo:

   ```
   KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
   ```

   > **Important**
   >
   > Não adicione um caractere de nova linha no final da string.

8. Organize os pares de valores-chave em ordem de classificação por bytes (alfabética pelo nome da chave).

9. Construa a string a ser assinada usando o seguinte exemplo de formato:

   ```
   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
   ```

   Exemplo de mensagem de notificação:

```
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 exemplo:

```
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. O `Signature` campo na mensagem é codificado em Base64. Você precisa decodificá-lo para comparar sua forma binária bruta com o hash derivado. Por exemplo:

```
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
echo "$SIGNATURE" | base64 -d > signature.bin
```

11. Use o `SignatureVersion` campo para selecionar o algoritmo de hash:

    - Para `SignatureVersion` 1, use SHA1(por exemplo,`-sha1`).
    - Para `SignatureVersion` 2, use SHA256(por exemplo,`-sha256`).

12. Para confirmar a autenticidade da mensagem do Amazon SNS, gere um hash da string construída e verifique a assinatura usando a chave pública.

```
openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
 "$STRING_TO_SIGN"
```

Se a assinatura for válida, a saída será`Verified OK`. Caso contrário, a saída será`Verification Failure`.

Exemplo de script com tratamento de erros

O script de exemplo a seguir automatiza o processo de verificação:

```
#!/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)
```

Analisar formatos de mensagem do Amazon SNS

Quando o Amazon SNS envia mensagens para endpoints HTTP/HTTPS, eles contêm cabeçalhos HTTP e um corpo de mensagem JSON. Essas mensagens seguem um formato estruturado que inclui metadados, como tipo de mensagem, ARN do tópico, carimbos de data/hora e assinaturas digitais. Ao analisar corretamente as mensagens do Amazon SNS, você pode determinar se uma mensagem é uma confirmação de assinatura, notificação ou confirmação de cancelamento de assinatura, extrair dados relevantes e verificar a autenticidade usando a validação de assinatura.

Cabeçalhos HTTP/HTTPS

Quando o Amazon SNS envia uma confirmação de assinatura, uma notificação ou uma mensagem de confirmação de cancelamento da assinatura para endpoints HTTP/HTTPS, ele envia uma mensagem POST com uma série de valores de cabeçalho específicos do Amazon SNS. É possível usar valores de cabeçalho para tarefas como identificar o tipo de mensagem sem precisar analisar o corpo da mensagem JSON para ler o valor `Type`. Por padrão, o Amazon SNS envia todas as notificações para endpoints HTTP/S com `Content-Type` definido como `text/plain; charset=UTF-8`. Para escolher um `Content-Type` diferente de texto/simples (padrão), consulte `headerContentType` emComo criar uma política de entrega HTTP/S.

`x-amz-sns-message-type`

O tipo de mensagem. Os valores possíveis são `SubscriptionConfirmation`, `Notification` e `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Um Identificador universalmente exclusivo (UUID), exclusivo para cada mensagem publicada. Para uma notificação que o Amazon SNS reenvia durante uma nova tentativa, o ID da mensagem original é usado.

`x-amz-sns-topic-arn`

O Nome de recurso da Amazon (ARN) para o tópico em que essa mensagem foi publicada.

`x-amz-sns-subscription-arn`

O ARN para a inscrição desse endpoint.

O cabeçalho HTTP POST a seguir é um exemplo de um cabeçalho para uma mensagem de `Notification` para um endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

Formato JSON de confirmação de assinatura HTTP/HTTPS

Depois de assinar um HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS endpoint. Essa mensagem contém um valor `SubscribeURL` que você deve consultar para confirmar a assinatura (como alternativa, é possível usar o valor `Token` com o `ConfirmSubscription`).

> **Note**
>
> O Amazon SNS não enviará notificações para esse endpoint enquanto a inscrição não for confirmada.

A mensagem de confirmação da inscrição é uma mensagem POST com um corpo de mensagem que contém um documento JSON com os pares de nome/valor a seguir.

`Type`

O tipo de mensagem. Para uma confirmação da inscrição, o tipo é `SubscriptionConfirmation`.

`MessageId`

Um Identificador universalmente exclusivo (UUID), exclusivo para cada mensagem publicada. Para uma mensagem que o Amazon SNS reenvia durante uma nova tentativa, o ID da mensagem original é usado.

`Token`

Um valor que pode ser usado com a ação `ConfirmSubscription` para confirmar a assinatura. Como alternativa, você pode simplesmente acessar o `SubscribeURL`.

`TopicArn`

O Nome de recurso da Amazon (ARN) para o tópico em que este endpoint está inscrito.

`Message`

Uma string que descreve a mensagem. Para confirmação da inscrição, esta string é semelhante a:

```
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`

O URL que você deve acessar para confirmar a inscrição. Como alternativa, é possível usar o `Token` com a ação `ConfirmSubscription` para confirmar a assinatura.

`Timestamp`

A hora (GMT) quando a confirmação da inscrição foi enviada.

`SignatureVersion`

Versão da assinatura do Amazon SNS usada.

- Se `SignatureVersion` for 1, `Signature` será uma assinatura `SHA1withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.
- Se `SignatureVersion` for 2, `Signature` será uma assinatura `SHA256withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.

`Signature`

Assinatura `SHA1withRSA` ou `SHA256withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.

`SigningCertURL`

O URL do certificado que foi usado para assinar a mensagem.

A mensagem HTTP POST a seguir é um exemplo de uma mensagem de `SubscriptionConfirmation` para um endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

Formato JSON de notificação HTTP/HTTPS

Quando o Amazon SNS envia uma notificação para um endpoint HTTP ou HTTPS inscrito, a mensagem POST enviada para o endpoint tem um corpo da mensagem que contém um documento JSON com os seguintes pares de nome-valor.

`Type`

O tipo de mensagem. Para uma notificação, o tipo é `Notification`.

`MessageId`

Um Identificador universalmente exclusivo (UUID), exclusivo para cada mensagem publicada. Para uma notificação que o Amazon SNS reenvia durante uma nova tentativa, o ID da mensagem original é usado.

`TopicArn`

O Nome de recurso da Amazon (ARN) para o tópico em que essa mensagem foi publicada.

`Subject`

O parâmetro `Subject` especificado quando a notificação foi publicada no tópico.

> **Note**
>
> Esse parâmetro é opcional. Se nenhum `Subject` foi especificado, esse par de nome/valor não é exibido nesse documento JSON.

`Message`

O valor `Message` especificado quando a notificação foi publicada no tópico.

`Timestamp`

A hora (GMT) quando a notificação foi publicada.

`SignatureVersion`

Versão da assinatura do Amazon SNS usada.

- Se `SignatureVersion` for 1, `Signature` será uma assinatura `SHA1withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Subject` (se presente), `Type`, `Timestamp` e `TopicArn`.
- Se `SignatureVersion` for 2, `Signature` será uma assinatura `SHA256withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Subject` (se presente), `Type`, `Timestamp` e `TopicArn`.

`Signature`

Assinatura `SHA1withRSA` ou `SHA256withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Subject` (se presente), `Type`, `Timestamp` e `TopicArn`.

`SigningCertURL`

O URL do certificado que foi usado para assinar a mensagem.

`UnsubscribeURL`

Um URL que você pode usar para cancelar a inscrição do endpoint com base neste tópico. Se você acessar este URL, o Amazon SNS cancela a inscrição do endpoint e interrompe o envio de notificações para esse endpoint.

A mensagem HTTP POST a seguir é um exemplo de uma mensagem de `Notification` para um endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
```

```
}
```

Formato JSON de confirmação de cancelamento de assinatura HTTP/HTTPS

Depois que a inscrição de um endpoint HTTP/HTTPS é cancelada em um tópico, o Amazon SNS envia uma mensagem de confirmação de cancelamento de inscrição para o endpoint.

A mensagem de confirmação do cancelamento da inscrição é uma mensagem POST com um corpo de mensagem que contém um documento JSON com os pares de nome/valor a seguir.

`Type`

O tipo de mensagem. Para uma confirmação de cancelamento de inscrição, o tipo é `UnsubscribeConfirmation`.

`MessageId`

Um Identificador universalmente exclusivo (UUID), exclusivo para cada mensagem publicada. Para uma mensagem que o Amazon SNS reenvia durante uma nova tentativa, o ID da mensagem original é usado.

`Token`

Um valor que pode ser usado com a ação `ConfirmSubscription` para confirmar novamente a assinatura. Como alternativa, você pode simplesmente acessar o `SubscribeURL`.

`TopicArn`

O Nome de recurso da Amazon (ARN) para o tópico do qual esse endpoint teve sua inscrição cancelada.

`Message`

Uma string que descreve a mensagem. Para a confirmação de cancelamento de inscrição, esta string é semelhante a:

```
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`

O URL que você deve acessar para confirmar novamente a inscrição. Como alternativa, é possível usar o `Token` com a ação `ConfirmSubscription` para confirmar novamente a assinatura.

`Timestamp`

A hora (GMT) quando a confirmação de cancelamento da inscrição foi enviada.

`SignatureVersion`

Versão da assinatura do Amazon SNS usada.

- Se `SignatureVersion` for 1, `Signature` será uma assinatura `SHA1withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.
- Se `SignatureVersion` for 2, `Signature` será uma assinatura `SHA256withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.

`Signature`

Assinatura `SHA1withRSA` ou `SHA256withRSA` codificada em Base64 dos valores `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.

`SigningCertURL`

O URL do certificado que foi usado para assinar a mensagem.

A mensagem HTTP POST a seguir é um exemplo de uma mensagem de `UnsubscribeConfirmation` para um endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "UnsubscribeConfirmation",
```

```
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37fb6...",
  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJm...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

SetSubscriptionAttributesformato JSON da política de entrega

Se você envia uma solicitação para a ação `SetSubscriptionAttributes` e define o parâmetro `AttributeName` para um valor de `DeliveryPolicy`, o valor do parâmetro `AttributeValue` deve ser um objeto JSON válido. Por exemplo, o exemplo a seguir define a política de entrega para um total de cinco novas tentativas.

```
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}}
...
```

Use o formato JSON a seguir para o valor do parâmetro `AttributeValue`.

```
{
    "healthyRetryPolicy" : {
        "minDelayTarget" :  int,
        "maxDelayTarget" : int,
        "numRetries" : int,
        "numMaxDelayRetries" : int,
        "backoffFunction" : "linear|arithmetic|geometric|exponential"
    },
```

```
    "throttlePolicy" : {
        "maxReceivesPerSecond" : int
    },
    "requestPolicy" : {
        "headerContentType" : "text/plain | application/json | application/xml"
    }
}
```

Para obter mais informações sobre a `SetSubscriptionAttribute` ação, acesse a Referência SetSubscriptionAttributesde API do Amazon Simple Notification Service. Para obter mais informações sobre os cabeçalhos do tipo de conteúdo HTTP compatíveis, consulte Como criar uma política de entrega HTTP/S.

SetTopicAttributes formato JSON da política de entrega

Se você envia uma solicitação para a ação `SetTopicAttributes` e define o parâmetro `AttributeName` para um valor de `DeliveryPolicy`, o valor do parâmetro `AttributeValue` deve ser um objeto JSON válido. Por exemplo, o exemplo a seguir define a política de entrega para um total de cinco novas tentativas.

```
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}}}
...
```

Use o formato JSON a seguir para o valor do parâmetro `AttributeValue`.

```
{
    "http" : {
        "defaultHealthyRetryPolicy" : {
            "minDelayTarget":  int,
            "maxDelayTarget": int,
            "numRetries": int,
            "numMaxDelayRetries": int,
            "backoffFunction": "linear|arithmetic|geometric|exponential"
        },
        "disableSubscriptionOverrides" : Boolean,
        "defaultThrottlePolicy" : {
            "maxReceivesPerSecond" : int
        },
```

```
        "defaultRequestPolicy" : {
            "headerContentType" : "text/plain | application/json | application/xml"
        }
    }
}
```

Para obter mais informações sobre a `SetTopicAttribute` ação, acesse a Referência SetTopicAttributesde API do Amazon Simple Notification Service. Para obter mais informações sobre os cabeçalhos do tipo de conteúdo HTTP compatíveis, consulte Como criar uma política de entrega HTTP/S.

Fanout eventos do Amazon SNS no AWS Event Fork Pipelines

Para arquivamento e análise de eventos, o Amazon SNS agora recomenda o uso de sua integração nativa com o Amazon Data Firehose. Você pode inscrever streams de entrega do Firehose em tópicos do SNS, o que permite enviar notificações para endpoints de arquivamento e análise, como buckets do Amazon Simple Storage Service (Amazon S3), tabelas do Amazon Redshift, Amazon Service (Service) e muito mais. OpenSearch OpenSearch Usar o Amazon SNS com os streams de entrega do Firehose é uma solução totalmente gerenciada e sem código que não exige o uso de funções. AWS Lambda Para obter mais informações, consulte Fanout de fluxos de entrega do Firehose.

Você pode usar o Amazon SNS para desenvolver aplicações orientadas por eventos que usam serviços de assinante para executar o trabalho automaticamente em resposta a eventos acionados por serviços do editor. Esse padrão de arquitetura pode tornar os serviços mais reutilizáveis, interoperáveis e dimensionáveis. No entanto, pode ser trabalhoso usar o processamento de eventos em pipelines que abordam os requisitos de manipulação de eventos comuns, como armazenamento de eventos, backup, pesquisa, análise e reprodução.

Para acelerar o desenvolvimento de seus aplicativos orientados a eventos, você pode inscrever pipelines de tratamento de eventos — desenvolvidos pelo AWS Event Fork Pipelines — em tópicos do Amazon SNS. AWS O Event Fork Pipelines é um conjunto de aplicativos aninhados de código aberto, baseado no AWS Serverless Application Model (AWS SAM), que você pode implantar diretamente do pacote AWS Event Fork Pipelines (escolha Mostrar aplicativos que criam funções personalizadas do IAM ou políticas de recursos) em sua conta. AWS

Para um AWS caso de uso do Event Fork Pipelines, consulte. Implantar e testar a aplicação de exemplo de event fork pipelines do Amazon SNS

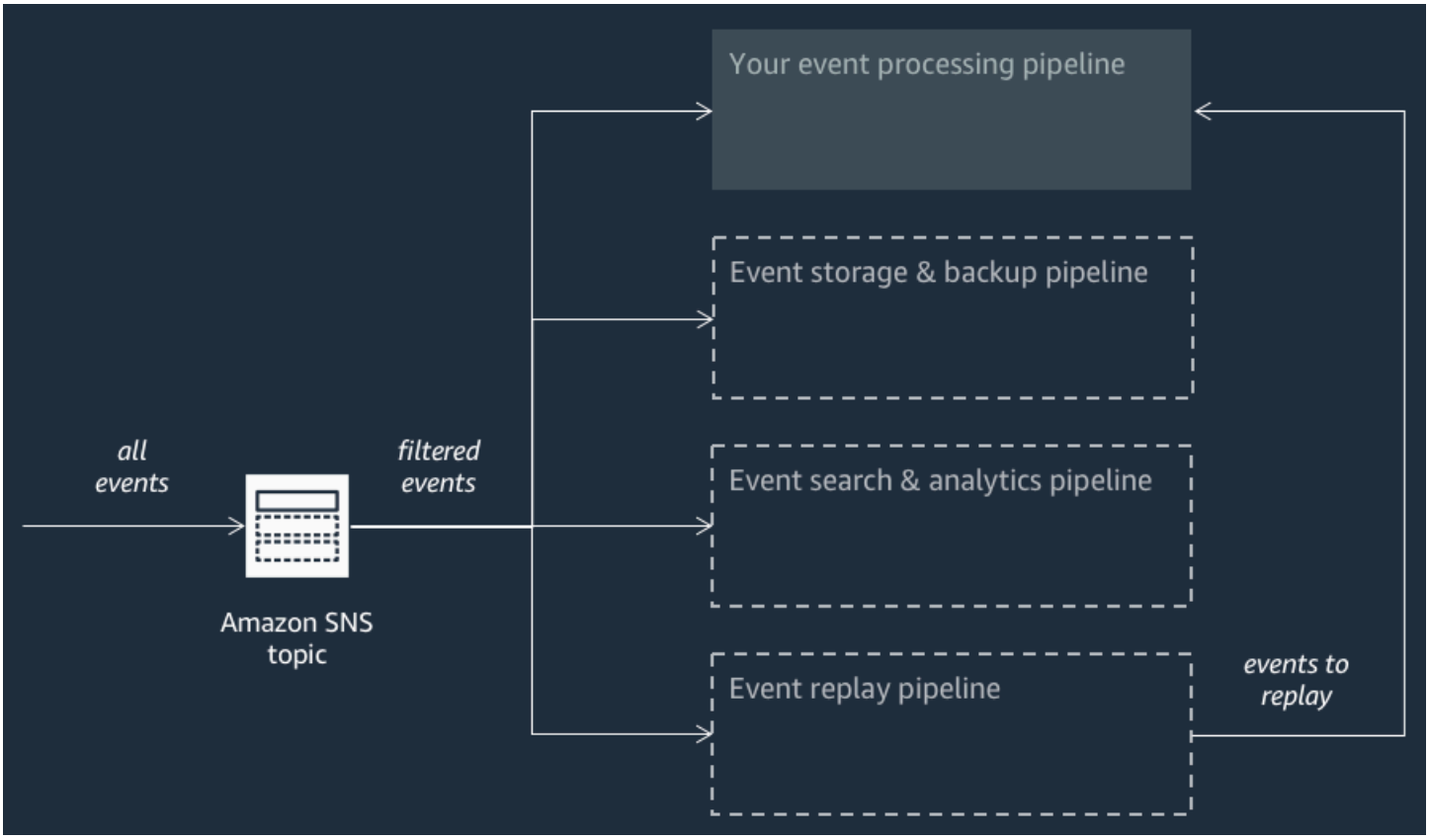
Tópicos

Como funciona o AWS Event Fork Pipelines

AWS O Event Fork Pipelines é um padrão de design sem servidor. No entanto, também é um conjunto de aplicativos sem servidor aninhados baseados no AWS SAM (que você pode implantar diretamente do (AWS SAR) para o AWS Serverless Application Repository seu, a fim de enriquecer suas Conta da AWS plataformas orientadas a eventos). Você pode implantar essas aplicações aninhadas individualmente, conforme sua arquitetura exigir.

Tópicos

O diagrama a seguir mostra um aplicativo AWS Event Fork Pipelines complementado por três aplicativos aninhados. Você pode implantar qualquer um dos pipelines do pacote AWS Event Fork Pipelines no AWS SAR de forma independente, conforme sua arquitetura exige.



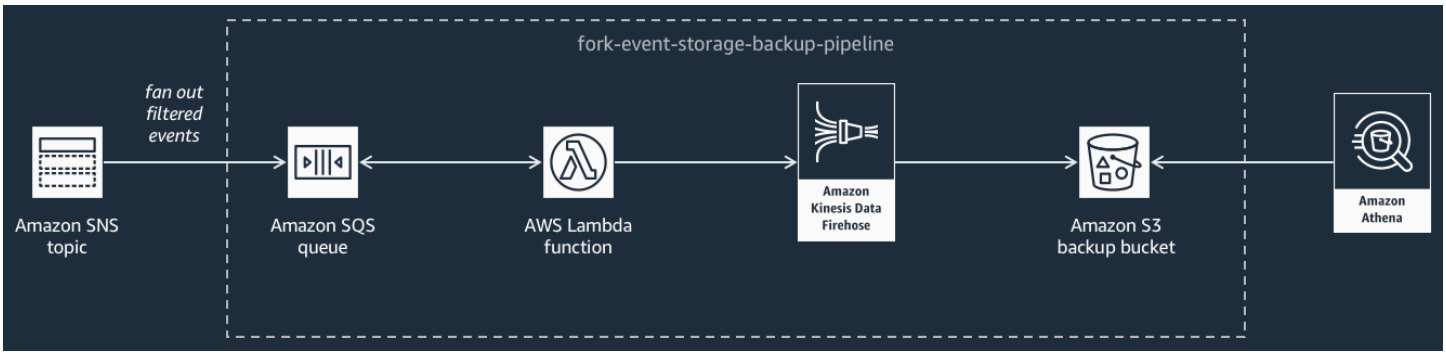
Cada pipeline é inscrito no mesmo tópico do Amazon SNS, o que permite que ele processe eventos em paralelo conforme esses eventos são publicados no tópico. Cada pipeline é independente e pode definir sua própria política de filtro de assinatura. Isso permite que um pipeline processe apenas um subconjunto dos eventos em que está interessado (em vez de todos os eventos publicados no tópico).

> **Note**
>
> Como você coloca os três AWS Event Fork Pipelines ao lado de seus pipelines regulares de processamento de eventos (possivelmente já inscritos no tópico do Amazon SNS), você não precisa alterar nenhuma parte do seu editor de mensagens atual para aproveitar os AWS Event Fork Pipelines em suas cargas de trabalho existentes.

O pipeline de armazenamento e backup de eventos

O diagrama a seguir mostra o pipeline de armazenamento e backup de eventos. Você pode inscrever esse pipeline em seu tópico do Amazon SNS para fazer backup automaticamente dos eventos que passam pelo sistema.

Esse pipeline é composto por uma fila do Amazon SQS que armazena em buffer os eventos entregues pelo tópico do Amazon SNS, AWS Lambda uma função que pesquisa automaticamente esses eventos na fila e os envia para um stream do Amazon Data Firehose e um bucket do Amazon S3 que faz backup duradouro dos eventos carregados pelo stream.

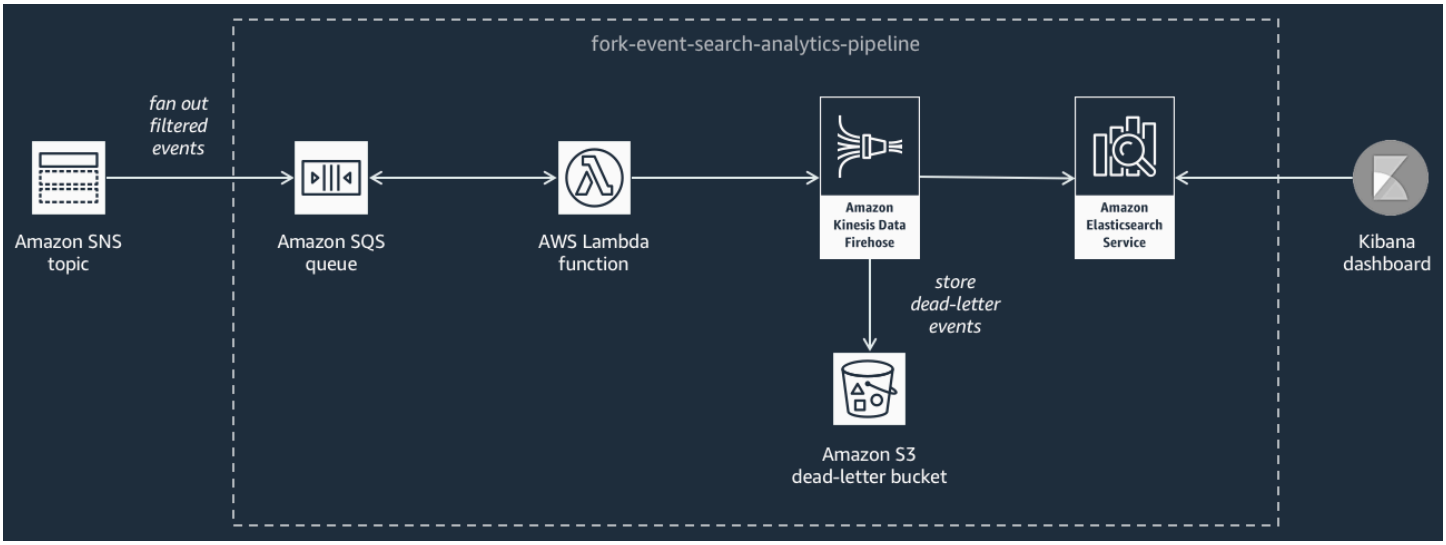


Para ajustar o comportamento de seu fluxo do Firehose, configure-o para armazenar em buffer, transformar e compactar seus eventos antes de carregá-los no bucket. À medida que os eventos forem carregados, use o Amazon Athena para consultar o bucket usando consultas SQL padrão. Você também pode configurar o pipeline para reutilizar um bucket do Simple Storage Service (Amazon S3) existente ou criar um novo.

O pipeline de pesquisa e análise de eventos

O diagrama a seguir mostra o pipeline de pesquisa e análise de eventos. Você pode inscrever esse pipeline em seu tópico do Amazon SNS para indexar os eventos que passam pelo sistema em um domínio de pesquisa e analisá-los.

Esse pipeline é composto por uma fila do Amazon SQS que armazena em buffer os eventos entregues pelo tópico do Amazon SNS, AWS Lambda uma função que pesquisa eventos da fila e os envia para um stream do Amazon Data Firehose, um domínio do Amazon OpenSearch Service que indexa os eventos carregados pelo stream do Firehose e um bucket do Amazon S3 que armazena os eventos sem saída que não podem ser indexado no domínio de pesquisa.



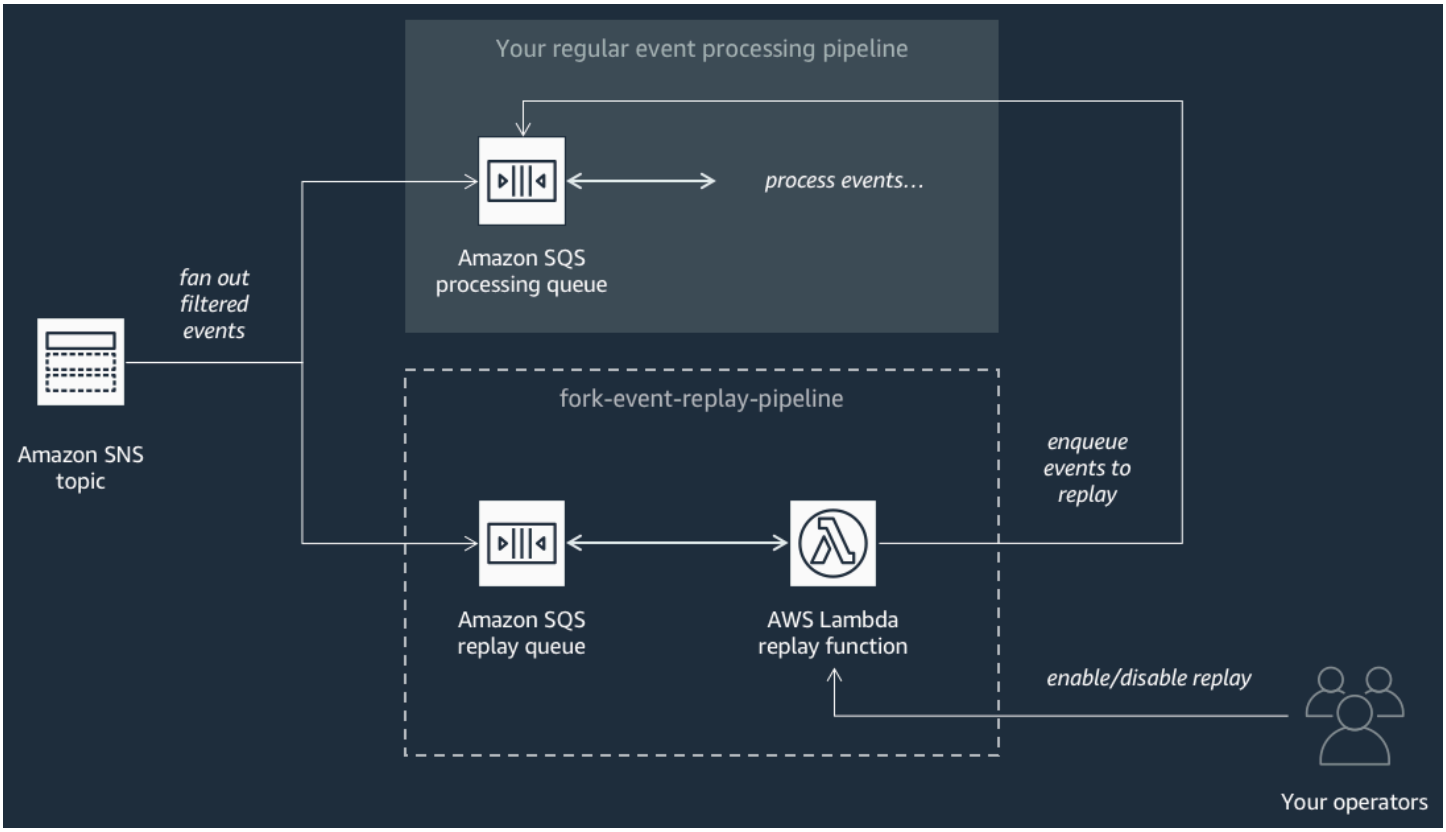
Para ajustar o fluxo do Firehose em termos de armazenamento de eventos em buffer, transformação e compactação, você pode configurar esse pipeline.

Você também pode configurar se o pipeline deve reutilizar um OpenSearch domínio existente no seu Conta da AWS ou criar um novo para você. À medida que os eventos forem indexados no domínio de pesquisa, use o Kibana para executar a análise de seus eventos e atualizar os painéis visuais em tempo real.

O pipeline de reprodução de eventos

O diagrama a seguir mostra o pipeline de reprodução de eventos. Para registrar os eventos que foram processados pelo sistema nos últimos 14 dias (por exemplo, quando sua plataforma precisar se recuperar de uma falha), inscreva esse pipeline em seu tópico do Amazon SNS e processe novamente os eventos.

Esse pipeline é composto por uma fila do Amazon SQS que armazena em buffer os eventos entregues pelo tópico do Amazon SNS e AWS Lambda uma função que pesquisa eventos da fila e os redireciona para seu pipeline regular de processamento de eventos, que também está inscrito em seu tópico.



> **Note**
>
> Por padrão, a função de reprodução está desabilitada e não redireciona seus eventos. Se precisar reprocessar os eventos, você deve habilitar a fila de reprodução do Amazon SQS como uma origem de eventos para a função de reprodução do AWS Lambda .

Implantação de tubulações AWS Event Fork

O pacote AWS Event Fork Pipelines (escolha Mostrar aplicativos que criam funções personalizadas do IAM ou políticas de recursos) está disponível como um grupo de aplicativos públicos no AWS Serverless Application Repository, de onde você pode implantá-los e testá-los manualmente usando o AWS Lambda console. Para obter informações sobre como implantar pipelines usando o AWS Lambda console, consulte. Inscrevendo AWS Event Fork Pipelines em um tópico do Amazon SNS

Em um cenário de produção, recomendamos incorporar o AWS Event Fork Pipelines ao modelo geral de SAM do AWS seu aplicativo. O recurso de aplicativo aninhado permite que você faça isso adicionando o recurso `AWS::Serverless::Application` ao seu modelo AWS SAM, referenciando o AWS SAR `ApplicationId` e o `SemanticVersion` do aplicativo aninhado.

Por exemplo, você pode usar o Event Storage and Backup Pipeline como um aplicativo aninhado adicionando o seguinte trecho de YAML à `Resources` seção do seu modelo do SAM. AWS

```
Backup:
    Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-2:123456789012:applications/fork-
event-storage-backup-pipeline
      SemanticVersion: 1.0.0
    Parameters:
      #The ARN of the Amazon SNS topic whose messages should be backed up to the Amazon
 S3 bucket.
      TopicArn: !Ref MySNSTopic
```

Ao especificar valores de parâmetros, você pode usar funções AWS CloudFormation intrínsecas para referenciar outros recursos em seu modelo. Por exemplo, no trecho YAML acima, o `TopicArn` parâmetro faz referência ao `AWS::SNS::Topic` recurso`MySNSTopic`, definido em outra parte do modelo. AWS SAM Para obter mais informações, consulte o Referência à função intrínseca no Guia do usuário do usuário do AWS CloudFormation .

> **Note**
>
> A página do AWS Lambda console do seu aplicativo AWS SAR inclui o botão Copiar como recurso do SAM, que copia o YAML necessário para aninhar um aplicativo AWS SAR na área de transferência.

Implantar e testar a aplicação de exemplo de event fork pipelines do Amazon SNS

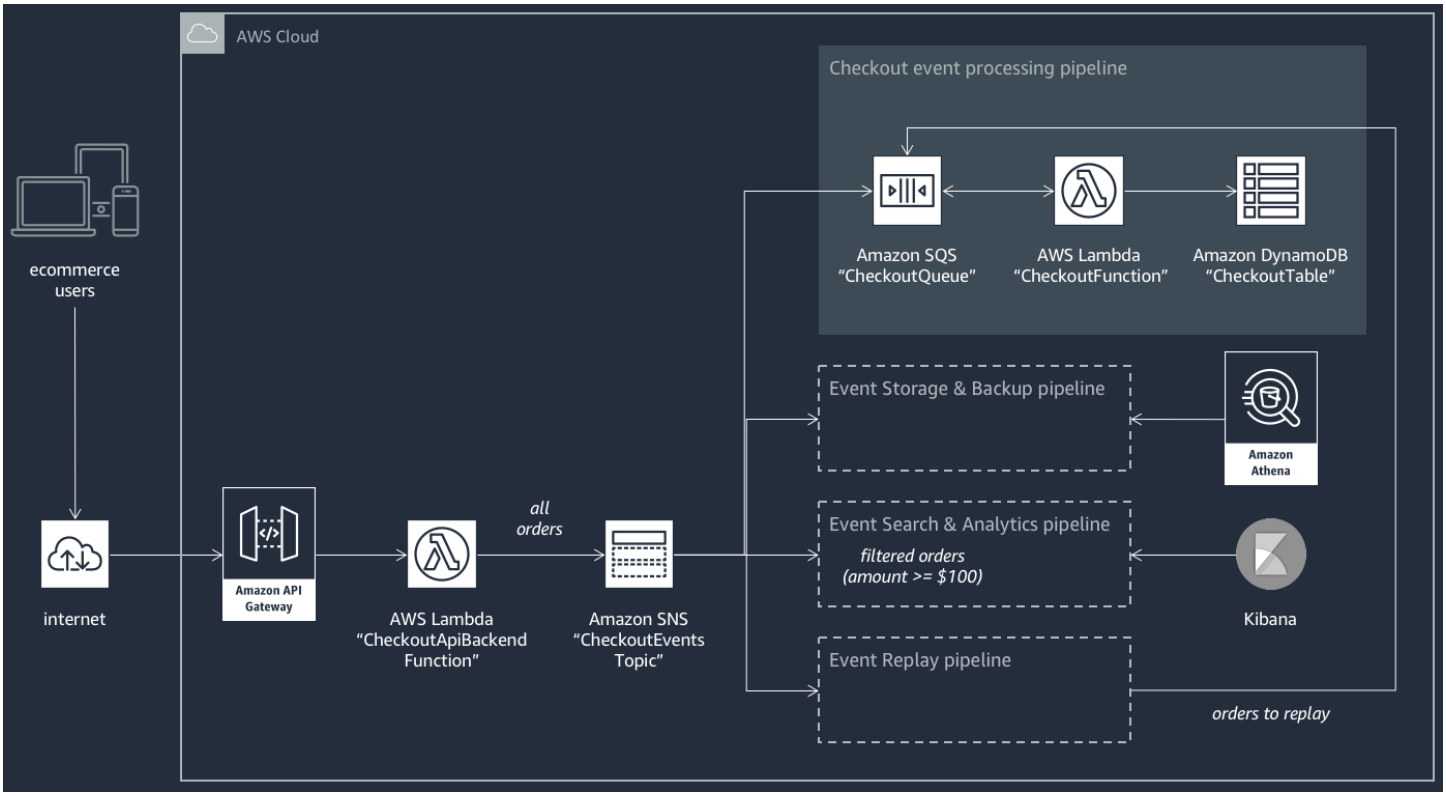
Para acelerar o desenvolvimento de seus aplicativos orientados a eventos, você pode inscrever pipelines de tratamento de eventos — desenvolvidos pelo AWS Event Fork Pipelines — em tópicos do Amazon SNS. AWS O Event Fork Pipelines é um conjunto de aplicativos aninhados de código aberto, baseado no AWS Serverless Application Model (AWS SAM), que você pode implantar diretamente do pacote AWS Event Fork Pipelines (escolha Mostrar aplicativos que criam funções personalizadas do IAM ou políticas de recursos) em sua conta. AWS Para obter mais informações, consulte Como funciona o AWS Event Fork Pipelines.

Esta página mostra como você pode usar o AWS Management Console para implantar e testar o aplicativo de amostra AWS Event Fork Pipelines.

> **Important**
>
> Para evitar custos indesejados após concluir a implantação do aplicativo de amostra AWS Event Fork Pipelines, exclua sua pilha. AWS CloudFormation Para obter mais informações, consulte Excluir uma pilha no console do AWS CloudFormation no Manual do usuário do AWS CloudFormation .

AWS Exemplo de caso de uso do Event Fork Pipelines

O cenário a seguir descreve um aplicativo de comércio eletrônico sem servidor e orientado por eventos que usa AWS o Event Fork Pipelines. Você pode usar esse exemplo de aplicativo de comércio eletrônico no AWS Serverless Application Repository e depois implantá-lo no AWS Lambda console, onde você pode testá-lo e examinar seu código-fonte GitHub. Conta da AWS



Esse aplicativo de comércio eletrônico recebe pedidos de compradores por meio de uma RESTful API hospedada pelo API Gateway e apoiada pela AWS Lambda função`CheckoutApiBackendFunction`. Essa função publica todos os pedidos recebidos em um

tópico do Amazon SNS chamado `CheckoutEventsTopic` que, por sua vez, envia os pedidos para quatro pipelines diferentes.

O primeiro pipeline é o pipeline de processamento de checkout usual projetado e implementado pelo proprietário do aplicativo de comércio eletrônico. Esse pipeline tem a fila do Amazon SQS `CheckoutQueue` que armazena em buffer todos os pedidos recebidos, uma AWS Lambda função chamada `CheckoutFunction` que pesquisa a fila para processar esses pedidos e a tabela do DynamoDB que salva com segurança todos os pedidos feitos. `CheckoutTable`

Aplicação de AWS tubulações Event Fork

Os componentes do aplicativo de comércio eletrônico processam a lógica de negócios central. No entanto, o proprietário do aplicativo de comércio eletrônico também precisa abordar o seguinte:

- Conformidade: backups seguros e compactados, criptografados em repouso e limpeza de informações confidenciais
- Resiliência: repetição dos pedidos mais recentes em caso de interrupção do processo de atendimento
- Capacidade de pesquisa: execução de análise e geração de métricas nos pedidos realizados

Em vez de implementar essa lógica de processamento de eventos, o proprietário do aplicativo pode inscrever o AWS Event Fork Pipelines no tópico do Amazon `CheckoutEventsTopic` SNS

- O O pipeline de armazenamento e backup de eventos é configurado com o intuito de transformar dados para remover detalhes de cartões de crédito, armazenar os dados em buffer durante 60 segundos, compactá-los usando GZIP e criptografá-los usando a chave mestra de cliente (CMK) padrão do Amazon S3. Essa chave é gerenciada AWS e alimentada pelo AWS Key Management Service (AWS KMS).

  Para obter mais informações, consulte Escolher o Amazon S3 para seu destino, Transformação de dados do Amazon Data Firehose e Ajustar configurações no Guia do desenvolvedor do Amazon Data Firehose.
- O O pipeline de pesquisa e análise de eventos é configurado com uma duração de nova tentativa do índice de 30 segundos, um bucket para armazenar os pedidos que não são indexados no domínio de pesquisa e uma política de filtro para restringir o conjunto de pedidos indexados.

  Para obter mais informações, consulte Escolha o OpenSearch serviço para seu destino no Guia do desenvolvedor do Amazon Data Firehose.

- O O pipeline de reprodução de eventos é configurado com a parte da fila do Amazon SQS do pipeline de processamento de pedidos usual projetado e implementado pela proprietária da aplicação de comércio eletrônico.

  Para obter mais informações, consulte Queue Name and URL (“Nome e URL da fila”) no Guia do desenvolvedor do Amazon Simple Queue Service.

A seguinte política de filtro JSON é definida na configuração do pipeline de pesquisa e análise de eventos. Ela corresponde apenas aos pedidos de entrada em que a quantidade total for de 100 USD ou superior. Para obter mais informações, consulte Filtragem de mensagens do Amazon SNS.

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

Usando o padrão AWS Event Fork Pipelines, o proprietário do aplicativo de comércio eletrônico pode evitar a sobrecarga de desenvolvimento que geralmente segue a lógica indiferenciadora de codificação para tratamento de eventos. Em vez disso, ela pode implantar o AWS Event Fork Pipelines diretamente de AWS Serverless Application Repository dentro dela. Conta da AWS

Etapa 1: implantar a aplicação de amostra do Amazon SNS

1. Faça login no console do AWS Lambda.
2. No painel de navegação, escolha Functions (Funções) e selecione Create function (Criar função).
3. Na página Create function (Criar função), faça o seguinte:

   a. Escolha Browse serverless app repository (Procurar no repositório de aplicações sem servidor), Public applications (Aplicações públicas), Show apps that create custom IAM roles or resource policies (Mostrar aplicações que criam funções do IAM ou políticas de recursos).

   b. Procure `fork-example-ecommerce-checkout-api` e escolha o aplicativo.
4. Na página fork-example-ecommerce-checkout-api, faça o seguinte:

   a. Na seção Application settings (Configurações do aplicativo), insira um Application name (Nome de aplicativo) (por exemplo, `fork-example-ecommerce-my-app`).

> **Note**
>
> - Para encontrar facilmente seus recursos posteriormente, mantenha o prefixo `fork-example-ecommerce`.
> - Para cada implantação, o nome do aplicativo deve ser exclusivo. Se você reutilizar o nome de um aplicativo, a implantação atualizará somente a AWS CloudFormation pilha implantada anteriormente (em vez de criar uma nova).

b. (Opcional) Insira uma das seguintes LogLevelconfigurações para a execução da função Lambda do seu aplicativo:

- `DEBUG`
- `ERROR`
- `INFO` (padrão)
- `WARNING`

5. Escolha I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Eu reconheço que este aplicativo cria funções personalizadas do IAM, políticas de recursos e implanta aplicativos aninhados.) e, na parte inferior da página, selecione Deploy (Implantar).

Na *`my-app`* página Status de implantação para fork-example-ecommerce -, o Lambda exibe o status Seu aplicativo está sendo implantado.

Na seção Recursos, AWS CloudFormation começa a criar a pilha e exibe o status CREATE_IN_PROGRESS de cada recurso. Quando o processo estiver concluído, AWS CloudFormation exibirá o status CREATE_COMPLETE.

> **Note**
>
> Pode levar entre 20–30 minutos para que todos os recursos sejam implantados.

Quando a implantação for concluída, o Lambda exibirá o status Your application has been deployed (Sua aplicação foi implantada).

Etapa 2: executar a aplicação de amostra vinculado ao SNS

1. No AWS Lambda console, no painel de navegação, escolha Aplicativos.
2. Na página Aplicativos, no campo de pesquisa, busque `serverlessrepo-fork-example-ecommerce-`*`my-app`* e escolha o aplicativo.
3. Na seção Recursos, faça o seguinte:

   a. Para encontrar o recurso cujo tipo é ApiGatewayRestApi, classifique os recursos por Tipo, por exemplo`ServerlessRestApi`, e expanda o recurso.

   b. Dois recursos aninhados são exibidos, dos tipos ApiGatewayDeployment e ApiGatewayStage.

   c. Copie o link Endpoint da API de produção e anexe `/checkout` a ele, por exemplo:

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

4. Copie o JSON a seguir em um arquivo denominado `test_event.json`.

   ```
   {
      "id": 15311,
      "date": "2019-03-25T23:41:11-08:00",
      "status": "confirmed",
      "customer": {
         "id": 65144,
     "quantity": 2,
         "price": 25.00,
         "subtotal": 50.00
      }]
   }
   ```

5. Para enviar uma solicitação HTTPS ao endpoint da API, envie a carga do evento de exemplo como entrada, executando um comando `curl`, por exemplo:

   ```
   curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
   east-2.amazonaws.com/Prod/checkout
   ```

   A API retorna a seguinte resposta vazia, indicando uma execução bem-sucedida:

   ```
   { }
   ```

Etapa 3: verificar o desempenho do aplicativo e do pipeline do Amazon SNS

Etapa 1: verificar a execução do pipeline de checkout de exemplo

1. Faça login no console do Amazon DynamoDB.
2. No painel de navegação, escolha Tables (Tabelas).
3. Busque `serverlessrepo-fork-example` e escolha `CheckoutTable`.
4. Na página de detalhes da tabela, escolha Items (Itens) e selecione o item criado.

   Os atributos armazenados são exibidos.

Etapa 2: verificar a execução do pipeline de armazenamento e backup de eventos

1. Faça login no console do Amazon S3.
2. No painel de navegação, selecione Buckets.
3. Busque `serverlessrepo-fork-example` e escolha `CheckoutBucket`.
4. Navegue pela hierarquia de diretórios até encontrar um arquivo com a extensão `.gz`.
5. Para fazer download do arquivo, escolha Ações, Abrir.
6. O pipeline está configurado com uma função do Lambda que limpa as informações de cartão de crédito por motivos de conformidade.

   Para verificar se a carga JSON armazenada não contém informações de cartão de crédito, descompacte o arquivo.

Etapa 3: verificar a execução do pipeline de pesquisa e análise de eventos

1. Faça login no console OpenSearch de serviço.
2. No painel de navegação, em Meus domínios, escolha o domínio prefixado com `serverl-analyt`.
3. O pipeline está configurado com uma política de filtro de assinatura do Amazon SNS que define uma condição de correspondência numérica.

   Para verificar se o evento está indexado porque se refere a um pedido cujo valor é maior que USD $100, na ***abcdefgh1ijk*** página servidor-analista, escolha Índices, checkout_events.

Etapa 4: verificar a execução do pipeline de repetição de eventos

1. Faça login no console do Amazon SQS.
2. Na lista de filas, busque `serverlessrepo-fork-example` e escolha `ReplayQueue`.
3. Escolha Enviar e receber mensagens.
4. Na caixa de *123ABCD4E5F6* diálogo Enviar e receber mensagens em fork-example-ecommerce -***my-app***... ReplayP- ReplayQueue -, escolha Sondagem de mensagens.
5. Para verificar se o evento está enfileirado, escolha Mais detalhes ao lado da mensagem que aparece na fila.

Etapa 4: simular um problema e repetir eventos para recuperação

Etapa 1: habilitar o problema simulado e enviar uma segunda solicitação da API

1. Faça login no console do AWS Lambda.
2. No painel de navegação, escolha Functions (Funções).
3. Busque `serverlessrepo-fork-example` e escolha `CheckoutFunction`.
4. No fork-example-ecommerce- ***my-app*** - CheckoutFunction -***ABCDEF***... página, na seção Variáveis de ambiente, defina a variável BUG_ENABLED como verdadeira e escolha Salvar.
5. Copie o JSON a seguir em um arquivo denominado `test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
"quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Para enviar uma solicitação HTTPS ao endpoint da API, envie a carga do evento de exemplo como entrada, executando um comando `curl`, por exemplo:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

A API retorna a seguinte resposta vazia, indicando uma execução bem-sucedida:

```
{ }
```

Etapa 2: verificar a corrupção de dados simulados

1. Faça login no console do Amazon DynamoDB.
2. No painel de navegação, escolha Tables (Tabelas).
3. Busque `serverlessrepo-fork-example` e escolha `CheckoutTable`.
4. Na página de detalhes da tabela, escolha Items (Itens) e selecione o item criado.

   Os atributos armazenados são exibidos, alguns marcados como CORROMPIDOS!

Etapa 3: desabilitar o problema simulado

1. Faça login no console do AWS Lambda.
2. No painel de navegação, escolha Functions (Funções).
3. Busque `serverlessrepo-fork-example` e escolha `CheckoutFunction`.
4. No fork-example-ecommerce- ***my-app*** - CheckoutFunction -***ABCDEF***... página, na seção Variáveis de ambiente, defina a variável BUG_ENABLED como falsa e escolha Salvar.

Etapa 4: habilitar a repetição a fim de recuperar do problema

1. No AWS Lambda console, no painel de navegação, escolha Funções.
2. Busque `serverlessrepo-fork-example` e escolha `ReplayFunction`.
3. Expanda a seção Designer, escolha o bloco SQS e, na seção SQS, selecione Habilitado.

> **Note**
>
> Leva aproximadamente 1 minuto para que o acionador da origem de eventos do Amazon SQS seja habilitado.

4. Escolha Salvar.
5. Para visualizar os atributos recuperados, volte ao console do Amazon DynamoDB.
6. Para desativar a repetição, retorne ao AWS Lambda console e desative o gatilho de origem de eventos do Amazon SQS para. `ReplayFunction`

Inscrevendo AWS Event Fork Pipelines em um tópico do Amazon SNS

Para acelerar o desenvolvimento de seus aplicativos orientados a eventos, você pode inscrever pipelines de tratamento de eventos — desenvolvidos pelo AWS Event Fork Pipelines — em tópicos do Amazon SNS. AWS O Event Fork Pipelines é um conjunto de aplicativos aninhados de código aberto, baseado no AWS Serverless Application Model (AWS SAM), que você pode implantar diretamente do pacote AWS Event Fork Pipelines (escolha Mostrar aplicativos que criam funções personalizadas do IAM ou políticas de recursos) em sua conta. AWS Para obter mais informações, consulte Como funciona o AWS Event Fork Pipelines.

Esta seção mostra como você pode usar o AWS Management Console para implantar um pipeline e depois inscrever o AWS Event Fork Pipelines em um tópico do Amazon SNS. Antes de começar, crie um tópico do Amazon SNS.

Para excluir os recursos que compõem um pipeline, localize o pipeline na página Aplicativos do AWS Lambda console, expanda a seção de modelos do SAM, escolha CloudFormationpilha e, em seguida, escolha Outras ações, Excluir pilha.

Implantar e inscrever o pipeline de armazenamento e backup de eventos para o Amazon SNS

Para arquivamento e análise de eventos, o Amazon SNS agora recomenda o uso de sua integração nativa com o Amazon Data Firehose. Você pode inscrever streams de entrega do Firehose em tópicos do SNS, o que permite enviar notificações para endpoints de arquivamento e análise, como buckets do Amazon Simple Storage Service (Amazon S3), tabelas do Amazon Redshift, Amazon Service (Service) e muito mais. OpenSearch OpenSearch Usar o Amazon SNS com os streams de entrega do Firehose é uma solução totalmente gerenciada e sem código que não exige o uso de funções. AWS Lambda Para obter mais informações, consulte Fanout de fluxos de entrega do Firehose.

Este tutorial mostra como implantar o pipeline de armazenamento e backup de eventos e inscrevê-lo em um tópico do Amazon SNS. Esse processo transforma automaticamente o AWS SAM modelo associado ao pipeline em uma AWS CloudFormation pilha e, em seguida, implanta a pilha na sua. Conta da AWS Esse processo também cria e configura o conjunto de recursos que compõe o pipeline de armazenamento e backup de eventos, incluindo o seguinte:

- Fila do Amazon SQS
- Função do Lambda
- Fluxo de entrega do Firehose
- Bucket de backup do Amazon S3

Para obter mais informações sobre como configurar um stream com um bucket do Amazon S3 como destino, `S3DestinationConfiguration` consulte a Amazon Data Firehose API Reference.

Para obter mais informações sobre como transformar eventos e configurar o armazenamento em buffer, a compactação e a criptografia de eventos, consulte Criar um fluxo de entrega do Amazon Kinesis Data Firehose no Guia do desenvolvedor do Amazon Data Firehose.

Para obter mais informações sobre como filtrar eventos, consulte Políticas de filtro de assinatura do Amazon SNS neste guia.

1. Faça login no console do AWS Lambda.
2. No painel de navegação, escolha Functions (Funções) e selecione Create function (Criar função).
3. Na página Create function (Criar função), faça o seguinte:
   a. Escolha Browse serverless app repository (Procurar no repositório de aplicações sem servidor), Public applications (Aplicações públicas), Show apps that create custom IAM roles or resource policies (Mostrar aplicações que criam funções do IAM ou políticas de recursos).
   b. Procure `fork-event-storage-backup-pipeline` e escolha o aplicativo.
4. Na página fork-event-storage-backup-pipeline, faça o seguinte:
   a. Na seção Application settings (Configurações do aplicativo), insira um Application name (Nome de aplicativo) (por exemplo, `my-app-backup`).

> **Note**
>
> - Para cada implantação, o nome do aplicativo deve ser exclusivo. Se você reutilizar o nome de um aplicativo, a implantação atualizará somente a AWS CloudFormation pilha implantada anteriormente (em vez de criar uma nova).

b. (Opcional) Para BucketArn, insira o ARN do bucket do Amazon S3 no qual os eventos recebidos são carregados. Se você não inserir um valor, um novo bucket do Amazon S3 será criado em sua AWS conta.

c. (Opcional) Para DataTransformationFunctionArn, insira o ARN da função Lambda por meio da qual os eventos recebidos são transformados. Se você não inserir um valor, a transformação de dados será desativada.

d. (Opcional) Insira uma das seguintes LogLevelconfigurações para a execução da função Lambda do seu aplicativo:

- `DEBUG`
- `ERROR`
- `INFO` (padrão)
- `WARNING`

e. Para TopicArn, insira o ARN do tópico do Amazon SNS no qual essa instância do fork pipeline deve ser assinada.

f. (Opcional) StreamBufferingSizeInMBsEm StreamBufferingIntervalInSecondse, insira os valores para configurar o armazenamento em buffer de eventos recebidos. Se nenhum valor for inserido, são usados 300 segundos e 5 MB.

g. (Opcional) Insira uma das seguintes StreamCompressionFormatconfigurações para compactar eventos recebidos:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (padrão)
- `ZIP`

h. (Opcional) Para StreamPrefix, insira o prefixo da string para nomear os arquivos armazenados no bucket de backup do Amazon S3. Se você não inserir um valor, nenhum prefixo será usado.

i. (Opcional) Para SubscriptionFilterPolicy, insira a política de filtro de assinatura do Amazon SNS, no formato JSON, a ser usada para filtrar eventos recebidos. A política de filtro decide quais eventos são indexados no índice OpenSearch de serviços. Se nenhum valor for inserido, nenhuma filtragem será usada (todos os eventos serão indexados).

j. (Opcional) Para SubscriptionFilterPolicyScope, insira a string `MessageBody` ou habilite `MessageAttributes` a filtragem de mensagens baseada em carga ou atributo.

k. Escolha I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Eu reconheço que este aplicativo cria funções personalizadas do IAM, políticas de recursos e implanta aplicativos aninhados.) e selecione Deploy (Implantar).

Na *`my-app`* página Status de implantação para, o Lambda exibe o status Seu aplicativo está sendo implantado.

Na seção Recursos, AWS CloudFormation começa a criar a pilha e exibe o status CREATE_IN_PROGRESS de cada recurso. Quando o processo estiver concluído, AWS CloudFormation exibirá o status CREATE_COMPLETE.

Quando a implantação for concluída, o Lambda exibirá o status Your application has been deployed (Sua aplicação foi implantada).

As mensagens publicadas em seu tópico do Amazon SNS são armazenadas automaticamente no bucket de backup do Amazon S3 provisionado pelo pipeline de armazenamento de eventos e backup.

Implantar e inscrever o pipeline de pesquisa e análise de eventos para o Amazon SNS

Para arquivamento e análise de eventos, o Amazon SNS agora recomenda o uso de sua integração nativa com o Amazon Data Firehose. Você pode inscrever streams de entrega do Firehose em tópicos do SNS, o que permite enviar notificações para endpoints de arquivamento e análise, como buckets do Amazon Simple Storage Service (Amazon S3), tabelas do Amazon Redshift, Amazon Service (Service) e muito mais. OpenSearch OpenSearch Usar o Amazon SNS com os streams de entrega do Firehose é uma solução totalmente gerenciada e sem código que não exige o uso de funções. AWS Lambda Para obter mais informações, consulte Fanout de fluxos de entrega do Firehose.

Esta página mostra como implantar o pipeline de pesquisa e análise de eventos e inscrevê-lo em um tópico do Amazon SNS. Esse processo transforma automaticamente o AWS SAM modelo associado ao pipeline em uma AWS CloudFormation pilha e, em seguida, implanta a pilha na sua. Conta da AWS Esse processo também cria e configura o conjunto de recursos que compõe o pipeline de pesquisa e análise de eventos, incluindo o seguinte:

- Fila do Amazon SQS
- Função do Lambda
- Fluxo de entrega do Firehose
- Domínio do Amazon OpenSearch Service
- Bucket de mensagens mortas do Amazon S3

Para obter mais informações sobre como configurar um fluxo com um índice como um destino, consulte `ElasticsearchDestinationConfiguration` na Referência de API do Amazon Data Firehose.

Para obter mais informações sobre como transformar eventos e configurar o armazenamento em buffer, a compactação e a criptografia de eventos, consulte Criar um fluxo de entrega do Amazon Kinesis Data Firehose no Guia do desenvolvedor do Amazon Data Firehose.

Para obter mais informações sobre como filtrar eventos, consulte Políticas de filtro de assinatura do Amazon SNS neste guia.

1. Faça login no console do AWS Lambda.
2. No painel de navegação, escolha Functions (Funções) e selecione Create function (Criar função).
3. Na página Create function (Criar função), faça o seguinte:

    a. Escolha Browse serverless app repository (Procurar no repositório de aplicações sem servidor), Public applications (Aplicações públicas), Show apps that create custom IAM roles or resource policies (Mostrar aplicações que criam funções do IAM ou políticas de recursos).

    b. Procure `fork-event-search-analytics-pipeline` e escolha o aplicativo.
4. Na página fork-event-search-analytics-pipeline, faça o seguinte:

    a. Na seção Application settings (Configurações do aplicativo), insira um Application name (Nome de aplicativo) (por exemplo, `my-app-search`).

> **Note**
>
> Para cada implantação, o nome do aplicativo deve ser exclusivo. Se você reutilizar o nome de um aplicativo, a implantação atualizará somente a AWS CloudFormation pilha implantada anteriormente (em vez de criar uma nova).

b. (Opcional) Para DataTransformationFunctionArn, insira o ARN da função Lambda usada para transformar eventos recebidos. Se você não inserir um valor, a transformação de dados será desativada.

c. (Opcional) Insira uma das seguintes LogLevelconfigurações para a execução da função Lambda do seu aplicativo:

- `DEBUG`
- `ERROR`
- `INFO` (padrão)
- `WARNING`

d. (Opcional) Em SearchDomainArn, insira o ARN do domínio de OpenSearch serviço, um cluster que configura a funcionalidade necessária de computação e armazenamento. Se você não inserir um valor, um novo domínio será criado com a configuração padrão.

e. Para TopicArn, insira o ARN do tópico do Amazon SNS no qual essa instância do fork pipeline deve ser assinada.

f. Para SearchIndexName, insira o nome do índice de OpenSearch serviços para pesquisa e análise de eventos.

> **Note**
>
> As seguintes cotas se aplicam a nomes de índice:
>
> - Não é possível incluir letras maiúsculas
> - Não é possível incluir os seguintes caracteres: `\ / * ? " < > | ` , #`
> - Não é possível começar com os seguintes caracteres: `- + _`
> - Não podem ser o seguinte: `. ..`
> - Não podem ter mais que 80 caracteres
> - Não podem ter mais que 255 bytes

- Não é possível conter dois pontos (do OpenSearch Serviço 7.0)

g. (Opcional) Insira uma das seguintes SearchIndexRotationPeriodconfigurações para o período de rotação do índice de OpenSearch serviços:

- `NoRotation` (padrão)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

A rotação de índice anexa um timestamp ao nome do índice, facilitando a expiração de dados antigos.

h. Para SearchTypeName, insira o nome do tipo de OpenSearch serviço para organizar os eventos em um índice.

> **Note**
>
> - OpenSearch Os nomes dos tipos de serviço podem conter qualquer caractere (exceto bytes nulos), mas não podem começar com _ eles.
> - Para o OpenSearch Service 6.x, só pode haver um tipo por índice. Se você especificar um novo tipo para um índice existente que já tem outro tipo, o Firehose retornará um erro runtime.

i. (Opcional) StreamBufferingSizeInMBsEm StreamBufferingIntervalInSecondse, insira os valores para configurar o armazenamento em buffer de eventos recebidos. Se nenhum valor for inserido, são usados 300 segundos e 5 MB.

j. (Opcional) Insira uma das seguintes StreamCompressionFormatconfigurações para compactar eventos recebidos:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (padrão)
- `ZIP`

k. (Opcional) Para StreamPrefix, insira o prefixo da string para nomear os arquivos armazenados no bucket de cartas mortas do Amazon S3. Se você não inserir um valor, nenhum prefixo será usado.

l. (Opcional) Para StreamRetryDurationInSecons, insira a duração da nova tentativa para casos em que o Firehose não consegue indexar eventos OpenSearch no índice de serviços. Se você não inserir um valor, será usado 300 segundos.

m. (Opcional) Para SubscriptionFilterPolicy, insira a política de filtro de assinatura do Amazon SNS, no formato JSON, a ser usada para filtrar eventos recebidos. A política de filtro decide quais eventos são indexados no índice OpenSearch de serviços. Se nenhum valor for inserido, nenhuma filtragem será usada (todos os eventos serão indexados).

n. Escolha I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Eu reconheço que este aplicativo cria funções personalizadas do IAM, políticas de recursos e implanta aplicativos aninhados.) e selecione Deploy (Implantar).

Na *my-app-search* página Status de implantação para, o Lambda exibe o status Seu aplicativo está sendo implantado.

Na seção Recursos, AWS CloudFormation começa a criar a pilha e exibe o status CREATE_IN_PROGRESS de cada recurso. Quando o processo estiver concluído, AWS CloudFormation exibirá o status CREATE_COMPLETE.

Quando a implantação for concluída, o Lambda exibirá o status Your application has been deployed (Sua aplicação foi implantada).

As mensagens publicadas em seu tópico do Amazon SNS são indexadas automaticamente no índice de OpenSearch serviços provisionado pelo pipeline de pesquisa e análise de eventos. Se o pipeline não puder indexar um evento, ele o armazenará em um bucket de cartas mortas do Amazon S3.

Implantação do Event Replay Pipeline com a integração do Amazon SNS

Esta página mostra como implantar o pipeline de repetição de eventos e inscrevê-lo em um tópico do Amazon SNS. Esse processo transforma automaticamente o AWS SAM modelo associado ao pipeline em uma AWS CloudFormation pilha e, em seguida, implanta a pilha na sua. Conta da AWS Esse processo também cria e configura o conjunto de recursos que compõe o pipeline de repetição de eventos, incluindo uma fila do Amazon SQS e uma função do Lambda.

Para obter mais informações sobre como filtrar eventos, consulte Políticas de filtro de assinatura do Amazon SNS neste guia.

1. Faça login no console do AWS Lambda.
2. No painel de navegação, escolha Functions (Funções) e selecione Create function (Criar função).
3. Na página Create function (Criar função), faça o seguinte:

   a. Escolha Browse serverless app repository (Procurar no repositório de aplicações sem servidor), Public applications (Aplicações públicas), Show apps that create custom IAM roles or resource policies (Mostrar aplicações que criam funções do IAM ou políticas de recursos).

   b. Procure `fork-event-replay-pipeline` e escolha o aplicativo.
4. Na página fork-event-replay-pipeline, faça o seguinte:

   a. Na seção Application settings (Configurações do aplicativo), insira um Application name (Nome de aplicativo) (por exemplo, `my-app-replay`).

   > **Note**
   >
   > Para cada implantação, o nome do aplicativo deve ser exclusivo. Se você reutilizar o nome de um aplicativo, a implantação atualizará somente a AWS CloudFormation pilha implantada anteriormente (em vez de criar uma nova).

   b. (Opcional) Insira uma das seguintes LogLevelconfigurações para a execução da função Lambda do seu aplicativo:

      - `DEBUG`
      - `ERROR`
      - `INFO` (padrão)
      - `WARNING`

   c. (Opcional) Para ReplayQueueRetentionPeriodInSeconds, insira o tempo, em segundos, durante o qual a fila de repetição do Amazon SQS mantém a mensagem. Se você não inserir um valor, será usado 1.209.600 segundos (14 dias).

   d. Para TopicArn, insira o ARN do tópico do Amazon SNS no qual essa instância do fork pipeline deve ser assinada.

e. Para DestinationQueueName, insira o nome da fila do Amazon SQS para a qual a função de reprodução do Lambda encaminha as mensagens.

f. (Opcional) Para SubscriptionFilterPolicy, insira a política de filtro de assinatura do Amazon SNS, no formato JSON, a ser usada para filtrar eventos recebidos. A política de filtro decide quais eventos serão armazenados em buffer para repetição. Se nenhum valor for inserido, nenhuma filtragem será usada (todos os eventos serão armazenados em buffer).

g. Escolha I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Eu reconheço que este aplicativo cria funções personalizadas do IAM, políticas de recursos e implanta aplicativos aninhados.) e selecione Deploy (Implantar).

Na *my-app-replay* página Status de implantação para, o Lambda exibe o status Seu aplicativo está sendo implantado.

Na seção Recursos, AWS CloudFormation começa a criar a pilha e exibe o status CREATE_IN_PROGRESS de cada recurso. Quando o processo estiver concluído, AWS CloudFormation exibirá o status CREATE_COMPLETE.

Quando a implantação for concluída, o Lambda exibirá o status Your application has been deployed (Sua aplicação foi implantada).

As mensagens publicadas em seu tópico do Amazon SNS são armazenadas em buffer para repetição na fila do Amazon SQS provisionada pelo pipeline de repetição de eventos automaticamente.

> **Note**
>
> Por padrão, a repetição está desabilitada. Para habilitar a repetição, navegue até a página da função no console do Lambda, expanda a seção Designer, escolha o bloco SQS e, na seção SQS, escolha Enabled (Habilitado).

Usando o Amazon EventBridge Scheduler com o Amazon SNS

O Amazon EventBridge Scheduler é um programador sem servidor que permite criar, executar e gerenciar tarefas a partir de um serviço gerenciado central. Com o EventBridge Scheduler, você pode criar agendas usando Cron e classificar expressões para padrões recorrentes ou configurar

invocações únicas. Você pode configurar janelas de tempo flexíveis para entregas, definir limites de novas tentativas e o tempo máximo de retenção de invocações de API com falha.

Esta página explica como usar o EventBridge Scheduler para publicar uma mensagem de um tópico do Amazon SNS em uma programação.

Configurar o perfil de execução

Quando você cria um novo EventBridge agendamento, o Scheduler deve ter permissão para invocar sua operação de API de destino em seu nome. Você concede essas permissões ao EventBridge Scheduler usando uma função de execução. A política de permissão que você anexa ao perfil de execução da programação define as permissões necessárias. Essas permissões dependem da API de destino que você deseja que o EventBridge Scheduler invoque.

Quando você usa o console do EventBridge Scheduler para criar um agendamento, como no procedimento a seguir, o EventBridge Scheduler configura automaticamente uma função de execução com base no destino selecionado. Se você quiser criar um EventBridge agendamento usando um dos Agendadores, o, ou SDKs AWS CLI AWS CloudFormation, você deve ter uma função de execução existente que conceda as permissões que o EventBridge Agendador exige para invocar um alvo. Para obter mais informações sobre como configurar manualmente uma função de execução para sua agenda, consulte Configurando uma função de execução no Guia do usuário do EventBridge Scheduler.

Criar uma programação

Para criar uma programação usando o console

1. Abra o console do Amazon EventBridge Scheduler em https://console.aws.amazon.com/scheduler/casa.
2. Na página Programações, clique em Criar programação.
3. Na página Especificar detalhes da programação, na seção Nome e descrição da programação, faça o seguinte:

   a. Em Nome da programação, insira um nome para a programação. Por exemplo, **`MyTestSchedule`**.

   b. (Opcional) Em Descrição, insira a descrição da programação. Por exemplo, **`My first schedule`**.

c. Em Grupo de programação, escolha um grupo de programação na lista suspensa. Se você não tiver um grupo, escolha padrão. Para criar um grupo de programação, escolha criar sua própria programação.

   Para adicionar tags a grupos de programação, você usa os grupos de programação.

4. • Escolha as opções de programação.

Ocorrência	Fazer isso...	
Programação única A programação única invoca o destino somente uma vez na data e hora que você especificar.	Em Data e hora, faça o seguinte: • Insira uma data válida no formato `YYYY/MM/DD` . • Insira um carimbo de data/hora no formato de 24 horas `hh:mm`. • Em Fuso horário, escolha o fuso horário.	
Programação recorrente A programação recorrente invoca o destino em uma taxa especificada por você usando uma expressão cron ou rate.	a. Em Tipo de programação, siga um dos procedimentos a seguir. • Para usar uma expressão cron para definir a programação, escolha Programação baseada em cron e insira a expressão cron. • Para usar uma expressão rate para definir a programação, escolha Programação baseada em rate	

Ocorrência	Fazer isso...	
	e insira a expressão rate. Para obter mais informações sobre expressões cron e de taxa, consulte Tipos de programação no EventBridge Scheduler no Guia do usuário do Amazon EventBridge Scheduler. b. Em Janela de tempo flexível, escolha Desativar para desativar a opção ou escolha uma das janelas de tempo predefinidas. Por exemplo, se você escolher 15 minutos e definir uma programação recorrente para invocar o destino uma vez a cada hora, a programação será executada em até 15 minutos após o início de cada hora.	

5. (Opcional) Se você escolher Programação recorrente na etapa anterior, na seção Período, faça o seguinte:

   a. Em Fuso horário, escolha um fuso horário.

   b. Em Data e hora de início, insira uma data válida no formato `YYYY/MM/DD` e, em seguida, especifique um carimbo de data/hora no formato de 24 horas `hh:mm`.

c. Para Data e hora de término, insira uma data válida no formato `YYYY/MM/DD` e, em seguida, especifique um carimbo de data/hora no formato 24 horas `hh:mm`.

6. Escolha Próximo.
7. Na página Selecionar destino, escolha a operação de AWS API que o EventBridge Scheduler invoca:

   a. Selecione Publicação do Amazon SNS.

   b. Na seção Publicar, selecione um SNS tópico ou escolha Criar novo SNS tópico.

   c. (Opcional) Insira um JSON carga útil. Se você não inserir uma carga, o EventBridge Scheduler usará um evento vazio para invocar a função.

8. Escolha Próximo.
9. Na página Configurações, faça o seguinte:

   a. Para ativar a programação, em Estado da programação, mude para Ativar programação.

   b. Para configurar uma política de novas tentativas para a programação, em Política de novas tentativas e fila de mensagens não entregues (DLQ), faça o seguinte:

      - Mude para Tentar novamente.
      - Em Idade máxima do evento, insira o (s) máximo (s) de hora (s) e minuto (s) em que o EventBridge Agendador deve manter um evento não processado.
      - O período máximo é de 24 horas.
      - Em Máximo de tentativas, insira o número máximo de vezes que o EventBridge Scheduler repete o agendamento se o alvo retornar um erro.

        O valor máximo é 185 tentativas.

      Com políticas de repetição, se um agendamento falhar em invocar seu destino, o EventBridge Scheduler executará novamente o agendamento. Se configurado, você deve definir o tempo máximo de retenção e as novas tentativas da programação.

   c. Escolha onde o EventBridge Scheduler armazena os eventos não entregues.

Opção Fila de mensagens não entregues (DLQ)	Fazer isso...	
Não armazene	Selecione Nenhum.	

Opção Fila de mensagens não entregues (DLQ)	Fazer isso...
Armazene o evento no mesmo Conta da AWS local em que você está criando a programação	a. Escolha Selecionar uma fila do Amazon SQS em my Conta da AWS as a DLQ. b. Escolha o nome do recurso da Amazon (ARN) da fila do Amazon SQS.
Armazene o evento em um local Conta da AWS diferente de onde você está criando a programação	a. Escolha Especificar uma fila do Amazon SQS em outra Contas da AWS como DLQ. b. Insira o nome do recurso da Amazon (ARN) da fila do Amazon SQS.

d. Para usar uma chave gerenciada pelo cliente para criptografar a entrada de destino, em Criptografia, escolha Personalizar as configurações de criptografia (avançado).

   Se você escolher essa opção, insira o ARN da chave do KMS existente ou escolha Criar AWS KMS key para navegar até o console do AWS KMS . Para obter mais informações sobre como o EventBridge Scheduler criptografa seus dados em repouso, consulte Criptografia em repouso no Guia do usuário do Amazon EventBridge Scheduler.

e. Para que o EventBridge Scheduler crie uma nova função de execução para você, escolha Criar nova função para esta agenda. Depois, insira um nome em Nome do perfil. Se você escolher essa opção, o EventBridge Scheduler anexará as permissões necessárias para seu alvo modelado à função.

10. Escolha Próximo.
11. Na página Revisar e criar programação, revise os detalhes da programação. Em cada seção, escolha Editar para voltar a essa etapa e editar seus detalhes.
12. Clique em Criar programação.

Você pode ver a lista com as programações novas e existentes na página Programações. Na coluna Status, verifique se a nova programação está Ativada.

Recursos relacionados

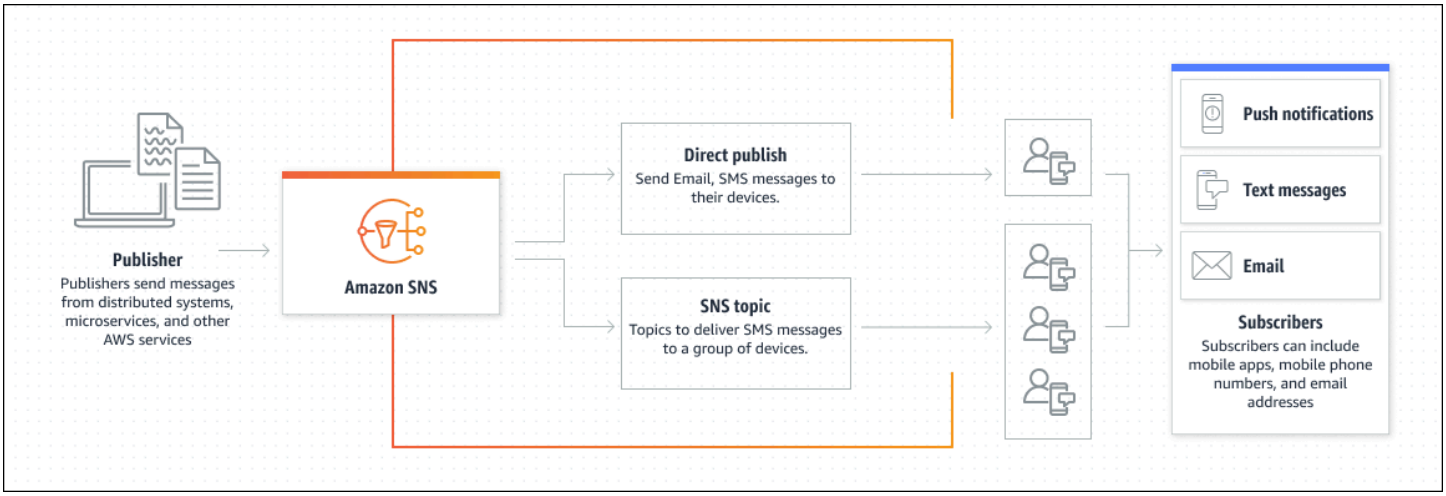
Para obter mais informações sobre o EventBridge Scheduler, consulte o seguinte:

- EventBridge Guia do usuário do Scheduler
- EventBridge Referência da API Scheduler
- EventBridge Preços do Scheduler

Usando o Amazon SNS para mensagens application-to-person

O sistema de mensagens do Amazon SNS application-to-person (A2P) permite que você envie notificações e alertas diretamente para os dispositivos móveis de seus clientes por meio de SMS (Short Message Service). Usando esse recurso, você pode enviar notificações push para aplicativos móveis, mensagens de texto para números de telefone celular e e-mails de texto sem formatação para endereços de e-mail. Além disso, você tem a flexibilidade de distribuir mensagens usando tópicos para alcançar vários destinatários ou publicar mensagens diretamente endpoints móveis individuais para comunicação personalizada.

Este tópico explica como usar o Amazon SNS para notificações de usuários com assinantes, como aplicativos móveis, números de telefone celular e endereços de e-mail.



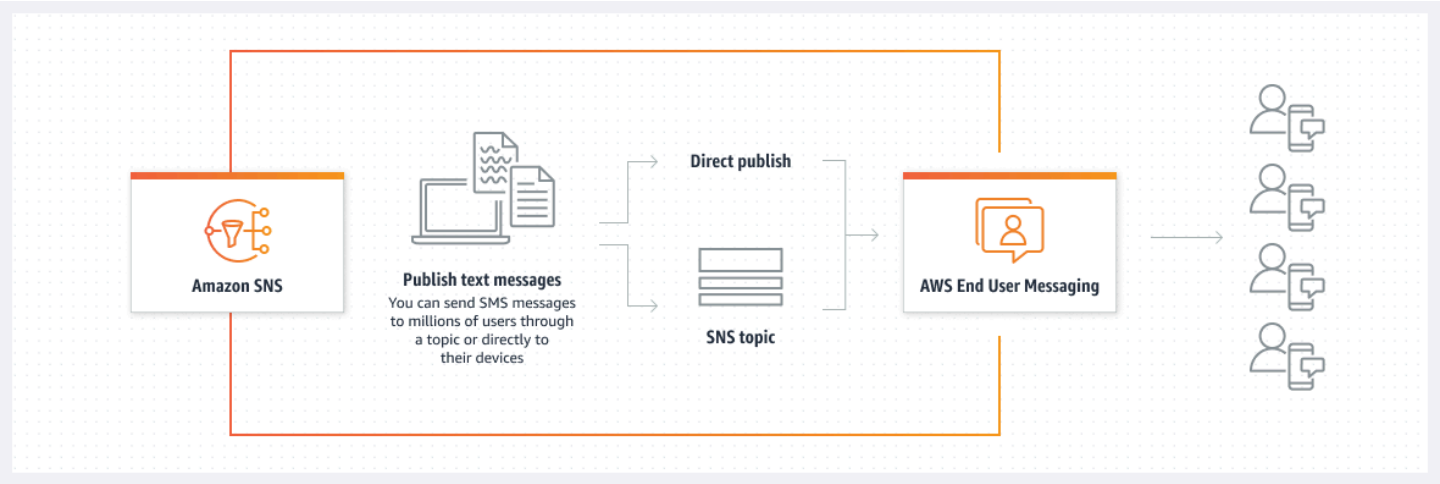
Mensagens de texto em dispositivos móveis com o Amazon SNS

> **Important**
>
> O Guia do desenvolvedor de SMS do Amazon SNS foi atualizado. O Amazon SNS foi integrado com AWS End User Messaging SMS para a entrega de mensagens SMS. Este Guia contém as informações mais recentes sobre como criar, configurar e gerenciar as mensagens SMS do Amazon SNS.

As mensagens de texto móveis (SMS) do Amazon SNS foram projetadas para facilitar a entrega de mensagens em várias plataformas, como aplicativos web, móveis e comerciais que oferecem suporte a SMS. Os usuários podem enviar mensagens para um ou vários números de telefone inscrevendo-os em um tópico, simplificando o processo de distribuição.

As mensagens do Amazon SNS são entregues por AWS End User Messaging SMS, o que garante uma transmissão confiável de mensagens. No Amazon SNS APIs, você pode definir várias propriedades, como tipos de mensagens (promocionais ou transacionais), limites de gastos mensais, listas de exclusão e otimização da entrega de mensagens.

AWS End User Messaging SMS gerencia a transmissão de mensagens para o número de telefone de destino por meio de sua rede global de fornecimento de SMS. Ele gerencia o roteamento, o status da entrega e qualquer conformidade necessária com os regulamentos regionais. Para acessar recursos adicionais de SMS, como permissões granulares, pools telefônicos, conjuntos de configurações, simulador de SMS e regras de país, consulte o Guia do usuário do AWS End User Messaging SMS.



Os principais recursos a seguir ajudam você a enviar mensagens SMS do Amazon SNS escaláveis e facilmente extensíveis:

Personalize as preferências de mensagens

Personalize as entregas de SMS para você Conta da AWS configurando as preferências de SMS com base em seu orçamento e caso de uso. Por exemplo, você pode escolher se as mensagens priorizam o custo ou entrega confiável.

Defina cotas de gastos

Defina suas entregas SMS ao especificar cotas de gastos para entregas de mensagens individuais e cotas de gastos mensais para sua Conta da AWS. Quando exigido pelas leis e regulamentações locais (como os EUA e o Canadá), os destinatários de SMS podem optar por não receber mais mensagens SMS de você. Conta da AWS Depois que um destinatário cancela o recebimento de mensagens, você pode, com limitações, incluir o número de telefone para retomar o envio de mensagens.

Envie mensagens SMS globalmente

O Amazon SNS é compatível com sistemas de mensagens SMS em várias regiões, e permite enviar mensagens para mais de 240 países e regiões.

Como o Amazon SNS entrega minhas mensagens SMS?

Quando você solicita que o Amazon SNS envie SMS em seu nome, as mensagens são enviadas usando AWS End User Messaging SMS. A integração entre o Amazon SNS AWS End User Messaging SMS oferece os seguintes benefícios:

Políticas do IAM

Você pode aproveitar o IAM e as políticas de recursos para controlar e distribuir o acesso aos seus recursos de SMS em outros AWS serviços e regiões.

Configurações do AWS End User Messaging SMS

Todas as configurações relacionadas à ID de originação (criação, atualização de configuração, provisionamento de nova originação IDs, alteração de modelos de registro) são usadas. AWS End User Messaging SMS

Faturamento AWS End User Messaging SMS

No entanto AWS End User Messaging SMS, todo o faturamento por SMS é feito. Você pode consolidar seus AWS gastos com suas cargas de trabalho de SMS e, ao mesmo tempo, adquirir e gerenciar seus recursos de SMS em um local central.

Conceitos básicos do Amazon SNS SMS

> **Important**
>
> O Guia do desenvolvedor de SMS do Amazon SNS foi atualizado. O Amazon SNS foi integrado com AWS End User Messaging SMS para a entrega de mensagens SMS. Este Guia contém as informações mais recentes sobre como criar, configurar e gerenciar as mensagens SMS do Amazon SNS.

Este tópico orienta você no gerenciamento de seu sandbox de SMS e na configuração de IAM e políticas baseadas em recursos para conceder ao Amazon SNS as permissões necessárias para acessar e utilizar o. AWS End User Messaging SMS APIs

Pré-requisitos

O Amazon SNS recomenda atualizar sua política do IAM para incluir as seguintes ações para garantir controle e visibilidade abrangentes sobre seus recursos do Amazon SNS:

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

Usar sandbox de SMS do Amazon SNS

As contas SMS recém-criadas do Amazon SNS são automaticamente colocadas na sandbox de SMS para garantir a segurança de AWS clientes e destinatários, reduzindo o risco de fraude e abuso. Esse ambiente serve como um espaço seguro para fins de teste e desenvolvimento. Ao operar no sandbox de SMS, você tem acesso a todos os recursos do Amazon SNS, mas está sujeito a certas limitações:

- As mensagens SMS poderão ser enviadas apenas para números de telefone de destino verificados.
- É possível ter até dez números de telefone de destino verificados.
- A exclusão de números de telefone de destino só é possível após 24 horas ou mais desde a verificação ou desde a última tentativa de verificação.

Quando a conta for movida para fora da sandbox, essas restrições serão removidas e será possível enviar mensagens SMS para qualquer destinatário.

Primeiras etapas

Novas contas do Amazon SNS SMS são colocadas em uma sandbox de SMS. Use as etapas a seguir para criar e gerenciar números de telefone em sua sandbox, criar números de origem e remetente IDs e registrar sua empresa.

1. Adicione um número de telefone de destino à sandbox de SMS. Para obter detalhes sobre como adicionar, gerenciar e mover números de telefone do sandbox de SMS do Amazon SNS, consulte Adicionar e verificar números de telefone na sandbox de Amazon SNS SMS.
2. Criar uma identidade de origem que seus destinatários veem nos dispositivos quando você envia uma mensagem SMS. Para saber mais sobre identidades de origem, incluindo os diferentes tipos que você pode usar, consulte a documentação Identidades de origem para mensagens do Amazon SNS SMS.
3. Cadastre sua empresa. Alguns países exigem que você registre a identidade da sua empresa para poder comprar números de telefone ou remetente IDs e revisar as mensagens enviadas aos destinatários em seus países. Para obter informações sobre quais países precisam de registro, consulte Países e regiões compatíveis com mensagem de SMS com AWS End User Messaging SMS no Guia do usuário do AWS End User Messaging SMS .
4. Envie suas mensagens para um tópico ou telefone celular. Para obter mais informações, consulte Enviar mensagens SMS usando o Amazon SNS.

Adicionar e verificar números de telefone na sandbox de Amazon SNS SMS

Antes de começar a enviar mensagens SMS Conta da AWS enquanto estiver na sandbox de SMS, você deve concluir as etapas de configuração a seguir. Isso garante que sua conta esteja pronta para receber mensagens SMS e que seus números de telefone de destino sejam devidamente verificados.

1. Crie uma ID de originação. Semelhante às contas fora do sandbox de SMS, uma ID de origem é necessária antes que você possa enviar mensagens SMS para destinatários em alguns países ou regiões.
2. Adicione os números de telefone de destino para os quais você deseja enviar mensagens na sandbox de SMS.

3. Verifique os números de telefone para garantir que os números de telefone de destino sejam válidos para uso em suas mensagens SMS.

Adicione e verifique os números de telefone de destino

1. Faça login no console do Amazon SNS.
2. No menu do console, selecione uma região compatível com mensagens SMS.
3. No painel de navegação, escolha Mensagens de texto (SMS).
4. Na seção Números de telefone de destino do Sandbox, selecione Adicionar número de telefone.
5. Em Detalhes do destino, forneça as seguintes informações e selecione Adicionar número de telefone:
   - Código do país e número de telefone do destino.
   - O idioma em que você deseja que a mensagem de verificação seja enviada.
6. Depois de adicionar o número de telefone, o Amazon SNS enviará uma OTP para o número de telefone de destino fornecido. Essa OTP é necessária para verificação.
7. Você receberá o OTP como uma mensagem SMS padrão no número de telefone de destino fornecido.
   - Se você não receber a OTP em 15 minutos, selecione Reenviar código de verificação no console do Amazon SNS.
   - Você pode reenviar a OTP até cinco vezes em um período de 24 horas.
8. Depois de receber a OTP, insira-a na caixa Código de verificação e selecione Verificar número de telefone.
9. Verifique o status da verificação.
   - Depois de verificar o número de telefone com sucesso, o número de telefone e seu status de verificação aparecerão na seção de números de telefone de destino do Sandbox.
   - Se o status for Pendente, a verificação não foi bem-sucedida. Isso pode acontecer se, por exemplo, você não inserir o código do país corretamente.
   - Você só pode excluir números de telefone pendentes ou verificados após 24 horas ou mais desde a última tentativa de verificação.
10. Se você quiser usar o mesmo número de telefone de destino em outras regiões, repita as etapas anteriores para cada região em que pretende usá-lo.

Solução de problemas do não recebimento de um texto OTP

Solucione problemas comuns que podem impedir que um número de telefone receba mensagens de texto OTP.

- Limite de gastos com SMS do Amazon SNS: se sua Conta da AWS excedeu o limite de gastos para enviar mensagens SMS, outras mensagens, incluindo textos OTP, podem não ser entregues até que o limite seja aumentado ou o problema de cobrança seja resolvido.
- Número de telefone não ativado para receber notificações por SMS: em alguns países ou regiões, os destinatários devem optar por receber mensagens SMS a partir de códigos curtos, que são comumente usados para textos OTP. Se o número de telefone do destinatário não estiver ativado, ele não receberá o texto OTP.
- Restrições ou filtragem da operadora: algumas operadoras de celular podem ter restrições ou mecanismos de filtragem que impedem a entrega de certos tipos de mensagens SMS, incluindo textos OTP. Isso pode ser devido a políticas de segurança ou medidas anti-spam implementadas pela operadora.
- Número de telefone inválido ou incorreto: se o número de telefone fornecido pelo destinatário estiver incorreto ou inválido, o texto OTP não será entregue.
- Problemas de rede: problemas ou interrupções temporárias na rede podem impedir a entrega de mensagens SMS, incluindo textos OTP, ao telefone do destinatário.
- Entrega atrasada: em alguns casos, as mensagens SMS podem sofrer atrasos na entrega devido ao congestionamento da rede ou a outros fatores. O texto do OTP pode eventualmente ser entregue, mas pode ser adiado além do prazo esperado.
- Suspensão ou encerramento da conta: Se houver problemas com você Conta da AWS, como falta de pagamento ou violação dos AWS termos de serviço, os recursos de mensagens do Amazon SNS, incluindo textos OTP, podem ser suspensos ou encerrados.

Excluir números de telefone da sandbox de Amazon SNS SMS

É possível excluir tanto números de telefone de destino pendentes como verificados da sandbox de SMS.

> **Important**
>
> Você pode apenas excluir o número de telefone 24 horas após verificar o número de telefone ou 24 horas após a última tentativa de verificação.

Para excluir números de telefone de destino da sandbox de SMS

1. Faça login no console do Amazon SNS.
2. No menu do console, selecione uma região compatível com mensagens SMS em que você adicionou um número de telefone de destino.
3. No painel de navegação, selecione Mensagens de texto (SMS).
4. Na página Mensagens de texto móveis (SMS), navegue até a seção Números de telefone de destino da sandbox.
5. Escolha o número de telefone específico que você deseja excluir e, em seguida, escolha Excluir número de telefone.
6. Para confirmar que deseja excluir o número de telefone, insira **`delete me`** e escolha Excluir.

   Verifique se passaram 24 horas ou mais desde que você verificou ou tentou verificar o número de telefone de destino antes de prosseguir com a exclusão.
7. Repita essas etapas em todas as regiões em que esse número de telefone de destino deve ser utilizado.

Saída da sandbox do Amazon SNS SMS

Para Conta da AWS sair da sandbox de SMS, você precisa primeiro adicionar, verificar e testar os números de telefone de destino. Depois de fazer isso, crie um caso com AWS Support.

Para solicitar que sua AWS conta seja removida da sandbox de SMS

1. Verificar número de telefone
   a. Enquanto você Conta da AWS estiver na sandbox do SMS, abra o console do Amazon SNS.
   b. No painel de navegação, em Dispositivo móvel, escolha Mensagens de texto (SMS).
   c. Na seção de números de telefone de destino da sandbox, adicione e verifique um ou mais números de telefone de destino. Essa verificação garante que você possa enviar e receber mensagens com sucesso.
2. Testar a publicação de SMS
   - Confirme se é possível publicar e receber mensagens em pelo menos um número de telefone de destino verificado. Para obter instruções mais detalhadas sobre como publicar

mensagens SMS, consulte Publicação de mensagens SMS em um telefone celular usando o Amazon SNS.

3. Iniciar a edição do sandbox

   - Na página Mobile text messaging (SMS) (Mensagens de texto móveis (SMS)) do console do Amazon SNS, em Account information (Informações da conta), escolha Exit SMS sandbox (Sair da sandbox de SMS). Essa ação redireciona você para o Central de suporte da Amazon e cria automaticamente um caso de suporte com a opção de aumento da cota de serviço selecionada.

4. Preencher o formulário

   - No formulário de suporte em Aumento da cota de serviços, faça o seguinte:

     i. Escolha escolher Mensagens de texto SNS como serviço.

     ii. Forneça o URL do site ou o nome do aplicativo a partir do qual você pretende enviar mensagens SMS.

     iii. Especifique o tipo de mensagem que você pretende enviar: Senha de uso único, Promocional ou Transacional.

     iv. Escolha a Região da AWS a partir da qual você enviará mensagens SMS.

     v. Liste os países ou as regiões para os quais você pretende enviar mensagens SMS.

     vi. Descreva como seus clientes optam por receber mensagens.

     vii. Inclua qualquer modelo de mensagem que você pretenda usar.

5. Especificar cota e região

   - Em Solicitações, faça o seguinte:

     i. Escolha para Região da AWSonde você deseja mover seu Conta da AWS.

     ii. Escolha Limites gerais para Tipo de recurso.

     iii. Escolha Sair da sandbox de SMS para Cota.

     iv. (Opcional) Para solicitar aumentos adicionais ou outros ajustes, escolha Adicionar outra solicitação e especifique os detalhes necessários.

     v. Em Novo valor de cota, insira o limite em USD que você está solicitando.

6. Outros detalhes

   a. Na descrição do caso, forneça detalhes adicionais relevantes à sua solicitação.

b. Expanda Opções de contato, e escolha o Idioma de contato preferido.

7. Envie a solicitação

   - Escolha Enviar para enviar a solicitação para Suporte.

A Suporte equipe fornece uma resposta inicial à sua solicitação em 24 horas.

Para evitar que nossos sistemas sejam usados para enviar conteúdo indesejado ou malicioso, consideramos cuidadosamente cada solicitação. Se for possível, atenderemos à sua solicitação dentro desse período de 24 horas. No entanto, se for necessário solicitar mais informações a você, o tempo de resolução poderá ser mais longo.

Se o seu caso de uso não estiver alinhado com nossas políticas, talvez não seja possível atender à sua solicitação.

Identidades de origem para mensagens do Amazon SNS SMS

> **Important**
>
> O Guia do desenvolvedor de SMS do Amazon SNS foi atualizado. O Amazon SNS foi integrado com AWS End User Messaging SMS para a entrega de mensagens SMS. Este Guia contém as informações mais recentes sobre como criar, configurar e gerenciar as mensagens SMS do Amazon SNS.

As identidades de origem de mensagens SMS são identificadores usados para representar o remetente de uma mensagem SMS. Você pode se identificar para seus destinatários usando os seguintes tipos de identidades de origem:

Números de origem

Uma sequência numérica que identifica o número de telefone do remetente da mensagem SMS. Existem vários tipos de números de origem, incluindo códigos longos (números de telefone padrão que normalmente têm 10 ou mais dígitos), códigos longos de 10 dígitos (10DLC), números gratuitos (TFN) e códigos curtos (números de telefone que contêm entre 4 e 7 dígitos).

O suporte para números de origem não está disponível em países onde as leis locais exijam o uso do remetente IDs em vez dos números de origem. Quando você envia uma mensagem SMS

usando um número de origem, o dispositivo do destinatário mostra o número de origem como o número de telefone do remetente. É possível especificar diferentes números de origem por caso de uso.

Para obter mais informações, consulte Números de telefone no Guia do usuário do AWS End User Messaging SMS .

> **Tip**
>
> Para ver uma lista de todos os números de origem existentes em sua AWS conta, no painel de navegação do console do Amazon SNS, escolha Números de origem.

Remetente IDs

Um nome alfabético que identifica o remetente de uma mensagem SMS. Quando você envia uma mensagem SMS usando um ID de remetente, e o destinatário está em uma área que oferece suporte à autenticação de ID de remetente, o ID desse remetente é exibido no dispositivo do destinatário em lugar de seu número de telefone. Um ID de remetente fornece aos destinatários de SMS mais informações sobre o remetente do que um número de telefone, código longo ou código simplificado.

IDs Os remetentes são suportados em vários países e regiões ao redor do mundo. Em alguns lugares, se a sua empresa envia mensagens SMS para clientes individuais, você deve usar um ID de remetente pré-registrado em uma agência reguladora ou grupo do setor. Para obter uma lista completa de países e regiões que oferecem suporte ou exigem remetente IDs, consulte Países e regiões com suporte para mensagens SMS AWS End User Messaging SMS no Guia do AWS End User Messaging SMS usuário.

Não há cobrança adicional pelo uso do remetente IDs. No entanto, o suporte e os requisitos para autenticação de ID de remetente variam de acordo com o país. Vários mercados importantes (incluindo Canadá, China e Estados Unidos da América) não suportam IDs o uso do remetente. Algumas áreas exigem que as empresas que enviam mensagens SMS para clientes individuais usem um ID de remetente pré-registrado junto a uma agência reguladora ou grupo do setor.

Para obter informações adicionais, consulte Remetente IDs no Guia do AWS End User Messaging SMS Usuário.

Configurar mensagens SMS no Amazon SNS

> **Important**
>
> O Guia do desenvolvedor de SMS do Amazon SNS foi atualizado. O Amazon SNS foi integrado com AWS End User Messaging SMS para a entrega de mensagens SMS. Este Guia contém as informações mais recentes sobre como criar, configurar e gerenciar as mensagens SMS do Amazon SNS.

Você pode usar as configurações no Amazon SNS SMS para definir preferências de SMS de acordo com suas necessidades, como ajustar cotas de gastos e configurar o registro do status de entrega. Este tópico também fornece detalhes sobre como publicar mensagens SMS em tópicos usando o console e o AWS SDK do Amazon SNS, lidar com cotas de forma eficiente e recuperar estatísticas detalhadas sobre a atividade de SMS.

Enviar mensagens SMS usando o Amazon SNS

Esta seção descreve como enviar mensagens SMS usando o Amazon SNS, incluindo publicar em um tópico, inscrever números de telefone em tópicos, definir atributos em mensagens e publicar diretamente em telefones celulares.

Publicar mensagens de SMS em um tópico do Amazon SNS

É possível publicar uma única mensagem SMS em vários números de telefone de uma só vez inscrevendo esses números de telefone em um tópico do Amazon SNS. Um tópico do Amazon SNS é um canal de comunicação para o qual é possível adicionar inscritos e publicar mensagens para todos esses inscritos. Um assinante recebe todas as mensagens publicadas no tópico até que você cancele a assinatura ou até que o assinante opte por não receber mensagens SMS da sua conta. AWS

Enviar uma mensagem para um tópico usando o console da AWS

Para criar um tópico

Realize as seguintes etapas se você ainda não tiver um tópico para o qual deseja enviar mensagens SMS.

1. Faça login no console do Amazon SNS.

2. No menu do console, selecione uma região compatível com mensagens SMS.
3. No painel de navegação, escolha Tópicos.
4. Na página Topics (Tópicos), escolha Create topic (Criar tópico).
5. Na página Create topic (Criar tópico), em Details (Detalhes), faça o seguinte:

   a. Em Tipo, escolha Padrão.
   b. Em Name (Nome), insira um nome para o tópico.
   c. (Opcional) Para Display name (Nome de exibição), digite um prefixo personalizado para suas mensagens SMS. Quando você envia uma mensagem para o tópico, o Amazon SNS acrescenta o nome de exibição seguido por uma seta para a direita (>) e um espaço. Os nomes de exibição não fazem distinção entre maiúsculas e minúsculas e o Amazon SNS converte os nomes de exibição para caracteres maiúsculos. Por exemplo, se o nome de exibição de um tópico é `MyTopic` e a mensagem é `Hello World!`, a mensagem é exibida como:

      ```
      MYTOPIC> Hello World!
      ```

6. Escolha Criar tópico. O nome do tópico e o nome do recurso da Amazon (ARN) aparecem na página Topics (Tópicos).

Para criar uma assinatura de SMS

Use assinaturas para enviar uma mensagem SMS para vários destinatários publicando a mensagem somente uma vez em seu tópico.

> **Note**
>
> Quando você começa a usar o Amazon SNS para enviar mensagens SMS, sua AWS conta está na sandbox de SMS. A sandbox de SMS oferece um ambiente seguro para testar os recursos do Amazon SNS sem arriscar a reputação como remetente do SMS. Enquanto sua conta estiver na sandbox de SMS, é possível usar todos os recursos do Amazon SNS, mas as mensagens SMS podem ser enviadas somente para números de telefone de destino verificados. Para obter mais informações, consulte Usar sandbox de SMS do Amazon SNS.

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Assinaturas.

3. Na página Assinaturas, escolha Criar assinatura.
4. Na página Create subscription (Criar assinatura), na seção Details (Detalhes), faça o seguinte:

   a. Em Topic ARN (ARN do tópico), insira ou escolha o nome do recurso da Amazon (ARN) do tópico para o qual você deseja enviar mensagens SMS.
   b. Para Protocol (Protocolo), escolha SMS.
   c. Para Endpoint digite o número de telefone que você deseja assinar para seu tópico.
5. Selecione Criar assinatura. As informações da assinatura são exibidas na página Subscriptions (Assinaturas).

   Para adicionar mais números de telefone, repita essas etapas. Também é possível adicionar outros tipos de assinaturas, como e-mail.

Para enviar uma mensagem

Quando você publica uma mensagem em um tópico, o Amazon SNS tenta entregar essa mensagem para cada número de telefone inscrito no tópico.

1. No console do Amazon SNS, na página Topics (Tópicos), escolha o nome do tópico para o qual você deseja enviar mensagens SMS.
2. Na página de detalhes do tópico, selecione Publicar mensagem.
3. Na página Publish message to topic (Publicar mensagem no tópico), em Message details (Detalhes da mensagem), faça o seguinte:

   a. Em Subject (Assunto), deixe o campo em branco, a menos que o tópico contenha inscrições de e-mail e você queira publicar nas inscrições por e-mail e por SMS. O Amazon SNS usará o Subject (Assunto) que você inserir como linha de assunto do e-mail.
   b. (Opcional) ParaTime to Live (TTL) (Vida útil (TTL)), insira o número de segundos que o Amazon SNS tem para enviar sua mensagem SMS para qualquer assinante de endpoint de aplicação móvel.
4. Em Message body (Corpo da mensagem), faça o seguinte:

   a. Em Message structure (Estrutura de mensagem), escolha Identical payload for all delivery protocols (Carga útil idêntica para todos os protocolos de entrega) para enviar a mesma mensagem para todos os tipos de protocolo inscritos no tópico. Ou escolha Custom payload for each delivery protocol (Carga útil personalizada para cada protocolo de entrega) para personalizar a mensagem direcionada a assinantes de diferentes tipos de protocolo. Por

exemplo, é possível inserir uma mensagem padrão para assinantes de número de telefone e uma mensagem personalizada para assinantes de e-mail.

b. Em Message body to send to the endpoint (Corpo da mensagem a ser enviada para o endpoint), insira sua mensagem ou suas mensagens personalizadas por protocolo de entrega.

   Se o tópico tem um nome de exibição, o Amazon SNS adiciona-o à mensagem, o que aumenta o tamanho da mensagem. O tamanho do nome de exibição é o número de caracteres no nome, mais dois caracteres para a seta (>) e o espaço adicionado pelo Amazon SNS.

   Para obter mais informações sobre as cotas de tamanho de mensagens SMS, consulte Publicação de mensagens SMS em um telefone celular usando o Amazon SNS.

5. (Opcional) Para atributos de mensagem, adicione metadados de mensagem, como carimbos de data/hora, assinaturas e. IDs
6. Selecione Publish message (Publicar mensagem). O Amazon SNS envia a mensagem SMS e exibe uma mensagem de êxito.

Enviar uma mensagem para um tópico usando o AWS SDKs

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

O código de exemplo a seguir mostra como:

- Criar um tópico do Amazon SNS.
- Inscrever números de telefone no tópico.
- Publicar mensagens SMS no tópico para que todos os números de telefone inscritos recebam a mensagem de uma só vez.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar um tópico e retorne seu 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 "";
    }
}
```

Inscreva um endpoint em um tópico.

```
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);
        }
    }
}
```

Defina atributos na mensagem, como o ID do remetente, o preço máximo e seu tipo. Os atributos de mensagem são opcionais.

```
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);
        }
    }
}
```

Publique uma mensagem em um tópico. A mensagem é enviada para todos os assinantes.

```
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);
        }
    }
}
```

Publicação de mensagens SMS em um telefone celular usando o Amazon SNS

É possível usar o Amazon SNS para enviar mensagens SMS diretamente para um telefone celular sem inscrever o número de telefone em um tópico do Amazon SNS.

> **Note**
>
> A inscrição de números de telefone em um tópico é útil se você quer enviar uma mensagem para vários números de telefone de uma só vez. Para obter instruções sobre como publicar uma mensagem SMS em um tópico, consulte Publicar mensagens de SMS em um tópico do Amazon SNS.

Ao enviar uma mensagem, é possível controlar se a mensagem é otimizada para custos ou confiabilidade de entrega. Também é possível especificar um ID do remetente ou número de origem. Se você enviar a mensagem programaticamente usando a API do Amazon SNS ou AWS SDKs a, você pode especificar um preço máximo para a entrega da mensagem.

Cada mensagem SMS pode ter até 140 bytes, e a cota de caracteres depende do esquema de codificação. Por exemplo, uma mensagem SMS pode incluir:

- 160 caracteres GSM
- 140 caracteres ASCII
- 70 caracteres UCS-2

Se você publicar uma mensagem que exceda a cota de tamanho, o Amazon SNS a enviará como várias mensagens, cada uma delas respeitando a cota de tamanho. Para isso, as palavras são mantidas inteiras na mensagem, não cortadas. O tamanho total da cota de uma única ação de publicação de SMS é de 1.600 bytes.

Ao enviar uma mensagem SMS, você especifica o número de telefone com o formato E.164, que é uma estrutura padrão de numeração de telefones para telecomunicação internacional. Os números de telefone que seguem esse formato podem conter no máximo 15 dígitos junto como prefixo de um sinal de adição (+) e o código do país. Por exemplo, um número de telefone dos EUA no formato E.164 aparece como +1 XXX555 0100.

Enviar uma mensagem (console)

1. Faça login no console do Amazon SNS.

2. No menu do console, selecione uma região compatível com mensagens SMS.
3. No painel de navegação, escolha Mensagens de texto (SMS).
4. Na página Mobile text messaging (SMS) (Mensagens de texto para dispositivos móveis (SMS)), selecione Publish text message (Publicar mensagem de texto).
5. Na página Publish SMS message (Publicar mensagem SMS), para Message type (Tipo de mensagem), escolha uma das seguintes opções:

   - Promotional (Promocional): mensagens não essenciais, como mensagens de marketing.
   - Transactional (Transacional): mensagens urgentes que oferecem suporte para transações do cliente, como senhas únicas para autenticação multifator.

   > **Note**
   >
   > Essa configuração em nível de mensagem substitui o tipo de mensagem padrão em nível de conta. É possível definir um tipo de mensagem padrão em nível de conta na seção Text messaging preferences (Opções de mensagens de texto) da página Mobile text messaging (SMS) (Mensagens de texto para dispositivos móveis (SMS)).

   Para obter informações sobre mensagens promocionais e transacionais, consulte Worldwide SMS Pricing (Preço global para SMS).
6. Em Destination phone number (Número de telefone de destino), digite o número de telefone para o qual você deseja enviar a mensagem.
7. Em Message (Mensagem), digite a mensagem a ser enviada.
8. (Opcional) Em Origination identities (Identidades de origem), especifique como você se identificará para seus destinatários:

   - Para especificar um Sender ID (ID do remetente), digite um ID personalizado que contenha de 3 a 11 caracteres alfanuméricos sem espaços, com pelo menos uma letra. O ID do remetente é exibido como o remetente da mensagem no dispositivo receptor. Por exemplo, é possível usar a marca de sua empresa para tornar a origem da mensagem mais fácil de reconhecer.

     Support para remetente IDs varia de acordo com o país e/ou região. Por exemplo, as mensagens enviadas para os números de telefone dos EUA não exibirão o ID do remetente. Para os países e regiões que oferecem suporte ao remetente IDs, consulte Países e regiões

com suporte para mensagens SMS AWS End User Messaging SMS no Guia do AWS End User Messaging SMS usuário.

Se você não especificar um ID do remetente, uma das seguintes informações será exibida como identidade de origem:

- Em países com suporte para códigos longos, o código longo será exibido.
- Em países onde somente o remetente IDs é suportado, o AVISO é exibido.

Esse ID do remetente no nível de mensagem substitui o ID de remetente padrão, que você define na página Preferências de mensagens de texto.

- Para especificar um Origination number (Número de origem), insira uma string de 5 a 14 números para ser exibida como número de telefone do remetente no dispositivo do receptor. Essa sequência de caracteres deve corresponder a um número de origem configurado em seu Conta da AWS para o país de destino. O número de origem pode ser um número 10DLC, número gratuito, código person-to-person longo ou códigos curtos. Para obter mais informações, consulte Identidades de origem para mensagens do Amazon SNS SMS.

  Se você não especificar um número de origem, o Amazon SNS selecionará um número de origem a ser usado para a mensagem de texto SMS, com base na configuração de sua Conta da AWS .

9. Se você estiver enviando mensagens SMS para destinatários na Índia, expanda Country-specific attributes (Atributos específicos do país) e especifique estes atributos:

   - Entity ID (ID da entidade): o ID da entidade ou ID da Principal Entity (PE – Entidade principal) para enviar mensagens SMS a destinatários na Índia. Esse ID é uma string exclusiva de 1 a 50 caracteres que a Telecom Regulatory Authority of India (TRAI) fornece para identificar a entidade que você registrou junto a ela.
   - Template ID (ID do modelo): o ID de modelo para enviar mensagens SMS a destinatários na Índia. Esse ID é uma string exclusiva de 1 a 50 caracteres fornecida pela TRAI que identifica o modelo que você registrou junto a ela. O ID do modelo deve ser associado ao ID do remetente que você especificou para a mensagem.

   Para obter mais informações sobre como enviar mensagens SMS para destinatários na Índia, consulte o processo de registro de IDs do remetente no Guia do Usuário do AWS End User Messaging SMS .

10. Selecione Publish message (Publicar mensagem).

> **Tip**
>
> Para enviar mensagens SMS de um número de origem, você também pode escolher Origination numbers (Números de origem) no painel de navegação do console do Amazon SNS. Escolha um número de origem que inclua SMS na coluna Capacities (Recursos) e, em seguida, escolha Publish text message (Publicar mensagem de texto).

Enviando uma mensagem (AWS SDKs)

Para enviar uma mensagem SMS usando um dos AWS SDKs, use a operação de API nesse SDK que corresponde à `Publish` solicitação na API do Amazon SNS. Com essa solicitação, você pode enviar uma mensagem SMS diretamente para um número de telefone. Você também pode usar o parâmetro `MessageAttributes` para definir valores para os seguintes nomes de atributos:

`AWS.SNS.SMS.SenderID`

Um ID personalizado que contenha de 3 a 11 caracteres alfanuméricos ou caracteres de hífen (-) sem espaços, com pelo menos uma letra. O ID do remetente é exibido como o remetente da mensagem no dispositivo receptor. Por exemplo, é possível usar a marca de sua empresa para ajudar a tornar a origem da mensagem mais fácil de reconhecer.

Support para remetente IDs varia de acordo com o país ou a região. Por exemplo, as mensagens enviadas para os números de telefone dos EUA não exibem o ID do remetente. Para obter uma lista dos países ou regiões que oferecem suporte ao remetente IDs, consulte Países e regiões com suporte para mensagens SMS AWS End User Messaging SMS no Guia do AWS End User Messaging SMS usuário.

Se você não especificar um ID do remetente, um código longo será exibido como o ID do remetente nos países ou regiões compatíveis. Para países ou regiões que exigem um ID do remetente alfabético, um AVISO será exibido como o ID do remetente.

Esse atributo de mensagem substitui o atributo no nível da contas `DefaultSenderID` que você pode definir usando a solicitação `SetSMSAttributes`.

`AWS.MM.SMS.OriginationNumber`

Uma string personalizada de 5 a 14 números, que pode incluir um sinal de adição opcional à esquerda (+). Essa string numérica aparece como o número de telefone do remetente no dispositivo receptor. A sequência de caracteres deve corresponder a um número de origem

configurado na sua AWS conta para o país de destino. O número de origem pode ser um número 10DLC, número gratuito, código longo person-to-person (P2P) ou código curto. Para obter mais informações, consulte Números de telefone no Guia do usuário do AWS End User Messaging SMS .

Se você não especificar um número de origem, o Amazon SNS escolherá um número de origem com base na configuração da sua conta. AWS

`AWS.SNS.SMS.MaxPrice`

O preço máximo em USD que você está disposto a gastar para enviar a mensagem SMS. Se o Amazon SNS determinar que enviar a mensagem pode gerar um custo que ultrapassaria o preço máximo, ele não enviará a mensagem.

Esse atributo não tem efeito se seus custos de month-to-date SMS já tiverem excedido a cota definida para o `MonthlySpendLimit` atributo. É possível definir o atributo `MonthlySpendLimit` usando a solicitação `SetSMSAttributes`.

Se você está enviando a mensagem para um tópico do Amazon SNS, o preço máximo se aplica a cada entrega de mensagem para cada número de telefone inscrito no tópico.

`AWS.SNS.SMS.SMSType`

O tipo de mensagem que você está enviando:

- **`Promotional`** (padrão): mensagens não essenciais, como mensagens de marketing.
- **`Transactional`**: mensagens essenciais que oferecem suporte a transações do cliente, como senhas únicas para autenticação multifator.

Esse atributo de mensagem substitui o atributo no nível da contas `DefaultSMSType` que você pode definir usando a solicitação `SetSMSAttributes`.

`AWS.MM.SMS.EntityId`

Esse atributo é necessário somente para enviar mensagens SMS a destinatários na Índia.

Esse é o ID da entidade ou ID da Principal Entity (PE – Entidade principal) para enviar mensagens SMS a destinatários na Índia. Esse ID é uma string exclusiva de 1 a 50 caracteres que a Telecom Regulatory Authority of India (TRAI) fornece para identificar a entidade que você registrou junto a ela.

`AWS.MM.SMS.TemplateId`

Esse atributo é necessário somente para enviar mensagens SMS a destinatários na Índia.

Este é o seu modelo para enviar mensagens SMS a destinatários na Índia. Esse ID é uma string exclusiva de 1 a 50 caracteres fornecida pela TRAI que identifica o modelo que você registrou junto a ela. O ID do modelo deve ser associado ao ID do remetente que você especificou para a mensagem.

Enviar uma mensagem

Os exemplos de código a seguir mostram como publicar mensagens SMS usando o Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para C++ .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Python (Boto3).

Definir preferências de mensagens SMS no Amazon SNS

Use o Amazon SNS para especificar preferências para mensagens SMS. Por exemplo, é possível especificar se as entregas serão otimizadas para fins de custo ou confiabilidade, o limite de gastos mensais, como as entregas serão registradas e a inscrição em relatórios diários de uso de SMS.

Essas preferências entrarão em vigor para cada mensagem SMS que você enviar de sua conta, mas será possível substituir algumas delas quando enviar uma mensagem individual. Para obter mais informações, consulte Publicação de mensagens SMS em um telefone celular usando o Amazon SNS.

Definir preferências de mensagens SMS usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. Escolha uma região que ofereça suporte a mensagens SMS.
3. No painel de navegação, escolha Dispositivo móvel e Mensagens de texto (SMS).
4. Na página Mobile text messaging (SMS) [Mensagens de texto (SMS) em dispositivos móveis], na seção Text messaging preferences (Preferências de mensagens de texto), escolha Edit (Editar).
5. Na página Editar preferências de mensagens SMS, na seção Detalhes, faça o seguinte:

    a. Em Tipo de mensagem padrão, selecione uma das seguintes opções:

- Promocional: mensagens não essenciais (por exemplo, de marketing). O Amazon SNS otimiza a entrega de mensagens para gerar o custo mais baixo.
- Transnacional (padrão): mensagens urgentes que comportam transações do cliente, como senhas únicas para autenticação multifator. O Amazon SNS otimiza a entrega de mensagens para gerar a mais alta confiabilidade.

Para obter informações sobre a definição de preços para mensagens promocionais e transacionais, consulte Definição global de preço para SMS.

b. (Opcional) Em Limite de gastos da conta, insira a quantidade máxima (em dólares americanos) que você deseja gastar em mensagens SMS a cada mês.

> **Important**
>
> - Por padrão, a cota de gasto é definida como 1,00 USD. Se desejar aumentar a cota de serviço, envie uma solicitação.
> - Se o valor definido no console exceder sua cota de serviço, o Amazon SNS interromperá a publicação de mensagens SMS.
> - Como o Amazon SNS é um sistema distribuído, ele interromperá o envio de mensagens SMS minutos depois que a cota de gasto for excedida. Durante esse intervalo, se você continuar a enviar mensagens SMS, poderá incorrer em custos que excederão sua cota.

6. (Opcional) Em ID do remetente padrão, insira um ID personalizado, como a marca de sua empresa, que será exibido como o remetente do dispositivo receptor.

> **Note**
>
> Support para remetente IDs varia de acordo com o país.

7. (Opcional) Digite o Amazon S3 bucket name for usage reports (Nome do bucket do Amazon S3 para relatórios de uso).

> **Note**
>
> A política de bucket do Amazon S3 deve conceder acesso de gravação ao Amazon SNS.

8. Escolha Salvar alterações.

Definindo preferências (AWS SDKs)

Para definir suas preferências de SMS usando um dos AWS SDKs, use a ação nesse SDK que corresponde à `SetSMSAttributes` solicitação na API do Amazon SNS. Com essa solicitação, você atribui valores para os diferentes atributos de SMS, como sua cota de gasto mensal e seu tipo de SMS padrão (promocional ou transacional). Para todos os atributos de SMS, consulte Definir SMSAttributes na referência da API do Amazon Simple Notification Service.

Os exemplos de código a seguir mostram como usar o `SetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Como usar o Amazon SNS para definir o atributo padrãoSMSType .

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

- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK para C++ da API.

CLI

AWS CLI

Para definir atributos de mensagens SMS

O exemplo `set-sms-attributes` a seguir define o ID do remetente padrão para mensagens SMS como `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Este comando não produz saída.

- Para obter detalhes da API, consulte Definir SMSAttributes na Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
```

```
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

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

- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK para PHP da API.

Configurar preferências de mensagens SMS para entrega específica do país

Você pode gerenciar e controlar seu tráfego de SMS enviando mensagens somente para países de destino específicos. Isso garante que suas mensagens sejam enviadas somente para países aprovados, evitando cobranças indesejadas por SMS. As instruções a seguir usam a configuração do Amazon Pinpoint Protect para especificar os países que você deseja permitir ou bloquear.

1. Abra o AWS SMS console em https://console.aws.amazon.com/sms-voice/.
2. No painel de navegação, em Visão geral, na seção Início rápido, escolha Criar uma configuração de proteção.
3. Em Detalhes da configuração de proteção, insira um nome comercial adequado para sua configuração de proteção (por exemplo, Allow-Only-AU).
4. Em Regras de país do SMS, marque a caixa de seleção Região/País para bloquear o envio de mensagens para todos os países compatíveis.
5. Desmarque as caixas de seleção dos países para onde você deseja enviar mensagens. Por exemplo, para permitir mensagens somente para a Austrália, desmarque a caixa de seleção para a Austrália.
6. Na seção Proteger associações de configuração, em Tipo de associação, selecione Conta padrão. Isso garantirá que a configuração do AWS End User Messaging SMS Protect afete todas as mensagens enviadas por meio do Amazon SNS, do Amazon Cognito e da chamada de API do Amazon Pinpoint. `SendMessages`
7. Selecione Criar para salvar suas configurações.

   A seguinte mensagem de confirmação é exibida:

   ```
   Success Protect configuration protect-abc0123456789 has been created.
   ```

8. Faça login no console do Amazon SNS.
9. Publique uma mensagem em um dos países bloqueados, como a Índia.

   A mensagem não será entregue. Você pode verificar isso nos registros de falha de entrega usando CloudWatch. Pesquise um grupo de registros sns/region/AccountID/DirectPublishToPhoneNumber/Failurepara obter uma resposta semelhante ao exemplo a seguir:

   ```
   {
   "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"
}
```

Gerenciar assinaturas de números de telefone do Amazon SNS

O Amazon SNS oferece várias opções para gerenciar quem recebe mensagens SMS de sua conta. Com uma frequência limitada, você poderá incluir números de telefone que cancelaram o recebimento de mensagens SMS de sua conta. Para interromper o envio de mensagens para inscrições de SMS, você pode remover as inscrições ou os tópicos que publicam neles.

Cancelar recebimento de mensagens SMS

Quando exigido pelas leis e regulamentos locais (como nos Estados Unidos e Canadá), os destinatários de SMS podem usar seus dispositivos para cancelar a inscrição respondendo à mensagem com uma destas opções:

- ARRET (Francês)
- CANCEL
- END
- CANCELAR
- CANCELAR
- QUIT
- REMOVE
- STOP
- TD
- CANCELAR INSCRIÇÃO

Para cancelar o recebimento, o destinatário deve responder ao mesmo número de origem que o Amazon SNS usou para entregar a mensagem. Depois de cancelar, o destinatário não receberá mais mensagens SMS enviadas por você, a Conta da AWS menos que você opte pelo número de telefone.

Se o número de telefone estiver inscrito em um tópico do Amazon SNS, o cancelamento não removerá a inscrição, mas mensagens SMS apresentarão falha na entrega para essa inscrição, a menos que você inclua o número de telefone.

Gerenciar números de telefone e assinaturas usando o console do Amazon SNS

É possível usar o console do Amazon SNS para controlar quais números de telefone recebem mensagens SMS de sua conta.

Incluir um número de telefone que cancelou a assinatura do console do Amazon SNS

É possível ver quais números de telefone cancelaram o recebimento de mensagens SMS de sua conta, e incluí-los para retomar o envio de mensagens para eles.

Você pode incluir um número de telefone somente uma vez a cada 30 dias.

1. Faça login no console do Amazon SNS.
2. No menu do console, defina a seleção de região para uma região que comporte mensagens SMS.
3. No painel de navegação, escolha Text messaging (SMS) [Mensagens de texto (SMS)].
4. Na página Mensagens de texto móveis (SMS), na seção Números de telefone excluídos, os números de telefone desativados são exibidos.
5. Marque a caixa de seleção para o número de telefone que você deseja incluir e selecione Incluir. O número de telefone não terá mais o recebimento cancelado e receberá mensagens SMS que você enviar a ele.

Excluir uma assinatura de SMS no console do Amazon SNS

Exclua uma inscrição de SMS para interromper o envio de mensagens SMS para aquele número de telefone quando você publicar nos tópicos.

1. No painel de navegação, escolha Subscriptions (Assinaturas).
2. Marque as caixas de seleção para as inscrições que você deseja excluir. Escolha Ações e escolha Excluir inscrições.
3. Na janela Delete (Excluir), escolha Delete (Excluir). O Amazon SNS exclui a assinatura e exibe uma mensagem de êxito.

Excluir um tópico do console do Amazon SNS

Exclua um tópico quando você não desejar mais publicar mensagens em seus endpoints inscritos.

1. No painel de navegação, escolha Tópicos.

2. Marque as caixas de seleção para os tópicos que você deseja excluir. Escolha Ações e escolha Excluir tópicos.
3. Na janela Delete (Excluir), escolha Delete (Excluir). O Amazon SNS exclui o tópico e exibe uma mensagem de êxito.

Gerenciar números de telefone e assinaturas usando o AWS SDK

Você pode usar o AWS SDKs para fazer solicitações programáticas ao Amazon SNS e gerenciar quais números de telefone podem receber mensagens SMS da sua conta.

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Arquivos de configuração e credenciais compartilhados no Guia de referência de ferramentas AWS SDKs e ferramentas.

Visualizando todos os números de telefone desativados usando o SDK AWS

Para visualizar todos os números de telefone cancelados, envie uma solicitação `ListPhoneNumbersOptedOut` com a API do Amazon SNS.

Os exemplos de código a seguir mostram como usar o `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Para listar as opções de cancelamento de mensagens SMS

O exemplo `list-phone-numbers-opted-out` a seguir lista os números de telefone que optaram por não receber mensagens SMS.

```
aws sns list-phone-numbers-opted-out
```

Saída:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOutem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOuta Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte ListPhoneNumbersOptedOuta Referência AWS SDK para PHP da API.

Verificar se um número de telefone foi excluído usando o SDK AWS

Para verificar se um número de telefone cancelou o recebimento, envie uma solicitação `CheckIfPhoneNumberIsOptedOut` com a API do Amazon SNS.

Os exemplos de código a seguir mostram como usar o `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para verificar o cancelamento de mensagens SMS para um número de telefone

O `check-if-phone-number-is-opted-out` exemplo a seguir verifica se o número de telefone especificado optou por não receber mensagens SMS da AWS conta atual.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Saída:

```
{
    "isOptedOut": false
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOutem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

```
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para PHP da API.

Incluir um número de telefone que cancelou a assinatura usando o Amazon SNS API

Para incluir um número de telefone que cancelou o recebimento, envie uma solicitação `OptInPhoneNumber` com a API do Amazon SNS.

Você pode incluir um número de telefone somente uma vez a cada 30 dias.

Excluindo uma assinatura de SMS usando o SDK AWS

Para excluir uma inscrição de SMS de um tópico do Amazon SNS, obtenha o ARN da inscrição enviando uma solicitação `ListSubscriptions` com a API do Amazon SNS e, em seguida, informe o ARN em uma solicitação `Unsubscribe`.

Os exemplos de código a seguir mostram como usar o `Unsubscribe`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cancele a assinatura de um tópico por meio de um ARN de assinatura.

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

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para cancelar a assinatura de um tópico

O exemplo `unsubscribe` a seguir exclui a assinatura especificada de um tópico.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Este comando não produz saída.

- Para obter detalhes da API, consulte Unsubscribe na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte Cancelar assinatura na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


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

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

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Cancelar assinatura na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
```

- Para obter detalhes da API, consulte Cancelar assinatura na referência da API AWS SDK for Swift.

Excluir um tópico usando o AWS SDK

Para excluir um tópico e todas as suas inscrições, obtenha o ARN do tópico enviando uma solicitação `ListTopics` com a API do Amazon SNS e, em seguida, informe o ARN em uma solicitação `DeleteTopic`.

Os exemplos de código a seguir mostram como usar o `DeleteTopic`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um tópico por meio do respectivo ARN.

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
```

```
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um tópico do SNS

O exemplo `delete-topic` a seguir exclui o tópico do SNS especificado.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteTopicem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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
  //   }
  // }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DeleteTopicReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

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

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

- Para obter detalhes da API, consulte DeleteTopica referência da API AWS SDK for Swift.

Monitoramento de atividades de SMS do Amazon SNS

Ao monitorar a atividade do SMS, é possível manter o controle dos números de telefone de destino, das entregas bem-sucedidas ou com falha, dos motivos da falha, dos custos e de outras informações. O Amazon SNS ajuda resumindo estatísticas no console, enviando informações para a Amazon CloudWatch e enviando relatórios diários de uso de SMS para um bucket do Amazon S3 que você especificar.

Visualizar estatísticas de entrega de SMS do Amazon SNS

É possível usar o console do Amazon SNS para exibir estatísticas sobre suas entregas recentes de SMS.

1. Faça login no console do Amazon SNS.
2. No menu do console, defina a seleção de região para uma região que comporte mensagens SMS.
3. No painel de navegação, escolha Text messaging (SMS) [Mensagens de texto (SMS)].
4. Na página Mensagens de texto (SMS), na seção Estatísticas da conta, veja os gráficos para suas entregas de mensagens SMS promocionais e transacionais. Cada gráfico mostra os seguintes dados para os 15 dias anteriores:

   - Taxa de entrega (porcentagem de entregas bem-sucedidas)
   - Enviadas (número de tentativas de entrega)
   - Falhas (número de falhas na entrega)

Nessa página, você também pode escolher o botão Usage (Uso) para acessar o bucket do Amazon S3 em que você armazena seus relatórios de uso diário. Para obter mais informações, consulte Como assinar relatórios diários de uso de SMS no Amazon SNS.

Monitoramento de entrega de SMS do Amazon SNS com CloudWatch métricas e registros da Amazon

Você pode usar a Amazon CloudWatch e o Amazon CloudWatch Logs para monitorar suas entregas de mensagens SMS.

Visualizando CloudWatch métricas da Amazon

O Amazon SNS coleta automaticamente métricas sobre suas entregas de mensagens SMS e as envia para a Amazon. CloudWatch Você pode usar CloudWatch para monitorar essas métricas e criar alarmes para alertá-lo quando uma métrica ultrapassa um limite. Por exemplo, você pode monitorar CloudWatch métricas para saber sua taxa de entrega de SMS e suas cobranças de month-to-date SMS.

Para obter informações sobre CloudWatch métricas de monitoramento, configuração de CloudWatch alarmes e os tipos de métricas disponíveis, consulteMonitorando tópicos do Amazon SNS usando CloudWatch.

Visualizando CloudWatch registros

Você pode coletar informações sobre entregas de mensagens SMS bem-sucedidas e malsucedidas ao permitir que o Amazon SNS grave no Amazon Logs. CloudWatch Para cada mensagem SMS que você enviar, o Amazon SNS gravará um log que inclui o preço da mensagem, o status de sucesso ou falha, o motivo da falha (se a mensagem falhou), o tempo de permanência da mensagem e outras informações.

Para ativar e visualizar CloudWatch os registros de suas mensagens SMS

1. Faça login no console do Amazon SNS.
2. No menu do console, defina a seleção de região para uma região que comporte mensagens SMS.
3. No painel de navegação, escolha Text messaging (SMS) [Mensagens de texto (SMS)].
4. Na página Mobile text messaging (SMS) [Mensagens de texto (SMS) em dispositivos móveis], na seção Text messaging preferences (Preferências de mensagens de texto), escolha Edit (Editar).

5. Na página seguinte, expanda a seção Registro do status da entrega.
6. Para taxa de amostragem de sucesso, especifique a porcentagem de entregas de SMS bem-sucedidas para as quais o Amazon SNS gravará registros em registros CloudWatch . Por exemplo:

   - Para gravar logs somente para entregas com falha, defina esse valor como 0.
   - Para gravar logs para 10% de suas entregas bem-sucedidas, defina o valor como 10.

   Se você não especificar uma porcentagem, o Amazon SNS gravará logs para todas as entregas bem-sucedidas.
7. Para fornecer as permissões necessárias, use uma das seguintes opções:

   - Para criar um novo perfil de serviço, escolha Criar novo perfil de serviço e, em seguida, Criar novos perfis. Na página seguinte, escolha Permitir para que o Amazon SNS tenha acesso de gravação aos recursos de sua conta.
   - Para usar um perfil de serviço existente, escolha Usar perfil de serviço existente e, em seguida, cole o ARN na caixa Perfil do IAM para entregas bem-sucedidas e com falha.

     A perfil de serviço especificado deve permitir acesso de gravação aos recursos de sua conta. Para obter mais informações sobre a criação de funções do IAM, consulte Como criar uma função para um AWS serviço no Guia do usuário do IAM.
8. Escolha Salvar alterações.
9. Novamente na página Mensagens de texto para dispositivos móveis (SMS), acesse a seção Logs de status de entrega para exibir todos os logs disponíveis.

> **Note**
>
> Dependendo da operadora do número de telefone de destino, pode demorar até 72 horas para que os logs de entrega apareçam no console do Amazon SNS.

Exemplo de log para entrega de SMS bem-sucedida

O log de status de entrega para uma entrega de SMS bem-sucedida será semelhante ao exemplo a seguir:

```
{
```

```
    "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"
}
```


Exemplo de log para entrega de SMS com falha

O log de status de entrega para uma entrega de SMS com falha será semelhante ao exemplo a seguir:


```
{
    "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"
}
```

Motivos para falha de entrega de SMS

O motivo da falha é fornecido com o atributo `providerResponse`. As mensagens SMS podem não ser entregues pelos seguintes motivos:

- Bloqueada como spam pela operadora de telefonia
- O destino está em uma lista bloqueada
- Número de telefone inválido
- O corpo da mensagem é inválido
- A operadora de telefonia bloqueou essa mensagem
- A operadora de telefonia está inacessível/indisponível no momento
- O telefone bloqueou SMS
- O telefone está em uma lista bloqueada
- O telefone está inacessível/indisponível no momento
- O número de telefone solicitou o cancelamento do recebimento
- Essa entrega excede o preço máximo
- Erro desconhecido ao tentar entrar em contato com o telefone

Como assinar relatórios diários de uso de SMS no Amazon SNS

É possível monitorar suas entregas de SMS inscrevendo-se para relatórios diários de uso do Amazon SNS. Para cada dia que você enviar pelo menos uma mensagem SMS, o Amazon SNS fornecerá um relatório de uso como um arquivo CSV para o bucket do Amazon S3 especificado. São necessárias 24 horas para que o relatório de uso do SMS esteja disponível no bucket do Amazon S3.

Informações sobre relatório de uso diário

O relatório de uso inclui as seguintes informações para cada mensagem SMS que foi enviada de sua conta.

O relatório não inclui as mensagens que são enviadas aos destinatários que recusaram a opção.

- Hora da publicação da mensagem (em UTC)
- ID de mensagem
- Número de telefone de destino

- Tipo de mensagem
- Status da entrega
- Preço da mensagem (em USD)
- Número da parte (uma mensagem é dividida em várias partes se for muito longa para uma única mensagem)
- Número total de partes

> **Note**
>
> Se o Amazon SNS não recebeu o número da peça, definimos seu valor como zero.

Assinar relatórios de uso diário

Para inscrever-se para relatórios de uso diário, você deve criar um bucket do Amazon S3 com as permissões apropriadas.

Para criar um bucket do Amazon S3 para seus relatórios de uso diário

1. Do Conta da AWS que envia mensagens SMS, faça login no console do Amazon S3.
2. Escolha Criar bucket.
3. Em Nome do bucket, recomendamos inserir um nome exclusivo para sua conta e organização. Por exemplo, use o padrão `<my-bucket-prefix>-<account_id>-<org-id>`.

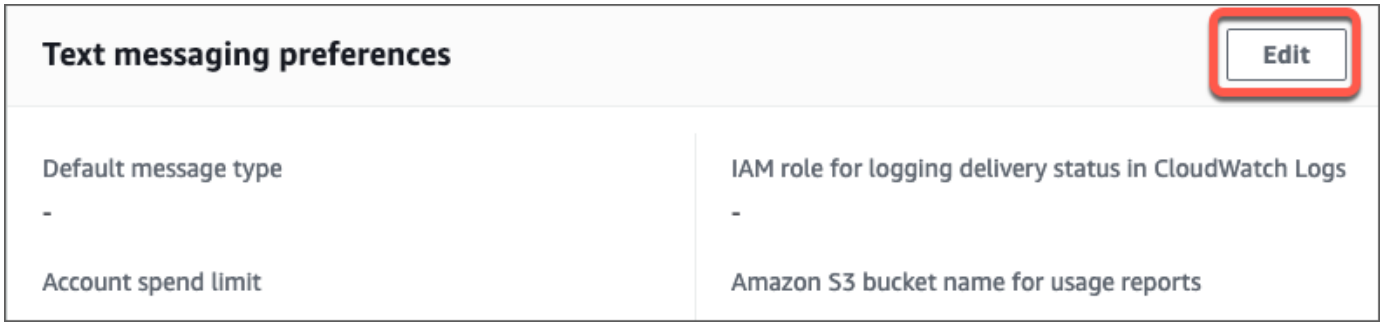
   Para obter informações sobre convenções e restrições de nomes de buckets, consulte Regras de nomenclatura de buckets no Guia do desenvolvedor do Amazon Simple Storage Service.
4. Escolha Criar.
5. Na tabela All Buckets (Todos os buckets), escolha o bucket.
6. Na guia Permissions (Permissões), escolha Bucket policy (Política de bucket).
7. Na janela Bucket Policy Editor (Editor de política de bucket), forneça uma política que permita ao principal do serviço Amazon SNS gravar no bucket. Para obter um exemplo, consulte Exemplo de política de bucket.

   Se você usar a política de exemplo, lembre-se de *`my-s3-bucket`* substituí-la pelo nome do bucket que você escolheu na Etapa 3.

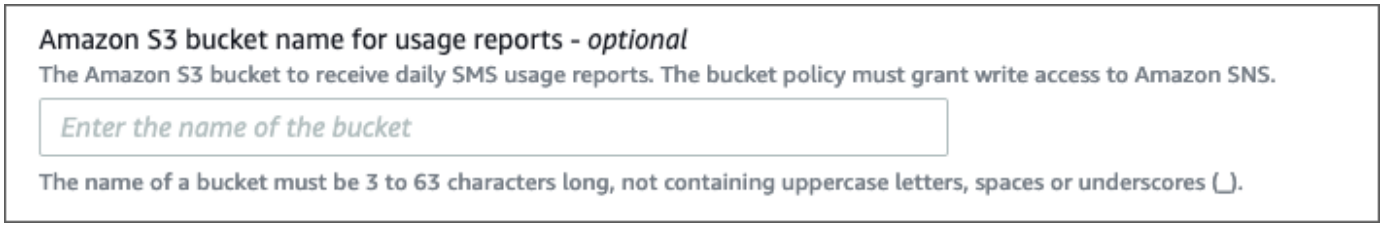
8. Escolha Salvar.

Para inscrever-se para relatórios de uso diário

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Text messaging (SMS) [Mensagens de texto (SMS)].
3. Na página Mensagens de texto (SMS), na seção Preferências de mensagens de texto, escolha Editar.



4. Na página Editar preferências de mensagens de texto, na seção Detalhes, especifique o Nome do bucket do Amazon S3 para relatórios de uso.



5. Escolha Salvar alterações.

Exemplo de política de bucket

A política a seguir permite que o principal do serviço Amazon SNS execute as ações `s3:PutObject`, `s3:GetBucketLocation` e `s3:ListBucket`.

AWS fornece ferramentas para todos os serviços com diretores de serviços que receberam acesso aos recursos em sua conta. Quando o principal em uma declaração de política de bucket do Amazon S3 é um problema de representante confuso. Para limitar de qual região e conta o bucket pode receber relatórios de uso diário, use `aws:SourceArn`, como mostra o exemplo abaixo. Se não quiser limitar quais regiões podem gerar esses relatórios, use `aws:SourceAccount` para limitar com base em qual conta está gerando os relatórios. Se você não conhece o ARN do recurso, use `aws:SourceAccount`.

Use o exemplo a seguir, que inclui proteção contra representante confuso quando você cria um bucket do Amazon S3 para receber relatórios de uso diário de SMS do Amazon SNS.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
                }
            }
        },
        {
            "Sid": "AllowGetBucketLocation",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:GetBucketLocation",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
                }
            }
        },
        {
            "Sid": "AllowListBucket",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
                }
            }
        }
    ]
}
```

> **Note**
>
> É possível publicar relatórios de uso em buckets do Amazon S3 que pertencem à Conta da AWS especificada no elemento `Condition` na política do Amazon S3. Para publicar relatórios de uso em um bucket do Amazon S3 de Conta da AWS propriedade de outra pessoa, consulte Como posso copiar objetos do Amazon S3 de outro? Conta da AWS.

Exemplo de relatório de uso diário

Depois de inscrever-se nos relatórios de uso diários, a cada dia o Amazon SNS coloca um arquivo CSV com os dados de uso no seguinte local:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Cada arquivo pode conter até 50.000 registros. Se os registros de um dia excederem essa cota, o Amazon SNS adicionará vários arquivos. A seguir, um exemplo de relatório:

```
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
```

Solicitar suporte para sistema de mensagens SMS do Amazon SNS

> **Important**
>
> O Guia do desenvolvedor de SMS do Amazon SNS foi atualizado. O Amazon SNS foi integrado com AWS End User Messaging SMS para a entrega de mensagens SMS. Este Guia contém as informações mais recentes sobre como criar, configurar e gerenciar as mensagens SMS do Amazon SNS.

Certas opções de SMS com o Amazon SNS não estão disponíveis para sua AWS conta até que você entre em contato. Suporte Crie um caso na AWS Support Center para solicitar qualquer um dos itens a seguir:

- Um aumento de seu limite mensal de gastos de SMS

  Por padrão, o limite de gastos mensal é 1,00 USD. O limite de gastos determina o volume de mensagens que é possível enviar com o Amazon SNS. Você pode solicitar um limite de gastos que atenda ao volume mensal de mensagens esperado para seu caso de uso de SMS.
- Uma migração da sandbox de SMS para que você possa enviar mensagens SMS sem restrições. Para obter mais informações, consulte Saída da sandbox do Amazon SNS SMS.
- Um número de origem dedicado
- Um ID do remetente dedicado. Um ID do remetente é um ID personalizado mostrado como remetente no dispositivo do destinatário. Por exemplo, é possível usar a marca de sua empresa para tornar a origem da mensagem mais fácil de reconhecer. Support para remetente IDs varia de acordo com o país ou a região. Para obter mais informações, consulte Países e regiões compatíveis com mensagem de SMS com AWS End User Messaging SMS no Guia do usuário do AWS End User Messaging SMS .

Solicitar aumentos de sua cota de gastos mensais de SMS para o Amazon SNS

O Amazon SNS fornece cotas de gastos para ajudar você a gerenciar o custo máximo por mês incorrido pelo envio de SMS usando sua conta. A cota de gastos limita seu risco em caso de ataque malicioso e impede que sua aplicação upstream envie mais mensagens do que o esperado. Você pode configurar o Amazon SNS para parar o envio de mensagens SMS quando ele determinar que esse envio tiver um custo que ultrapassará sua cota de gastos do mês.

Para garantir que suas operações não sejam afetadas, recomendamos solicitar uma cota de gastos alta o suficiente para suportar suas cargas de trabalho de produção. Para obter mais informações, consulte Etapa 1: abrir um caso de SMS do Amazon SNS. Depois de receber a cota, você pode gerenciar seu risco aplicando a cota completa ou um valor menor, conforme descrito em Etapa 2: atualizar suas configurações de SMS. Ao aplicar um valor mais baixo, você pode controlar seus gastos mensais com a opção de escalar, se necessário.

> **Important**
>
> Como o Amazon SNS é um sistema distribuído, ele interromperá o envio de mensagens SMS minutos depois que a cota de gasto for excedida. Durante esse intervalo, se você continuar a enviar mensagens SMS, poderá gerar custos que ultrapassam sua cota.

Definimos a cota de gastos para todas as novas contas como 1,00 USD por mês. Essa cota tem como objetivo permitir testar os recursos de envio de mensagens do Amazon SNS. Para solicitar um aumento na cota de gastos com SMS da sua conta, abra um caso de aumento de cota no AWS Support Center.

Tópicos

Etapa 1: abrir um caso de SMS do Amazon SNS

Você pode solicitar um aumento na sua cota de gastos mensais abrindo um caso de aumento de cota no AWS Support Center.

> **Note**
>
> Alguns campos no formulário de solicitação são marcados como "opcionais". No entanto, o Suporte exige todas as informações mencionadas nas etapas a seguir para processar sua solicitação. Se você não fornecer todas as informações necessárias, poderá haver atrasos no processamento de sua solicitação.

1. Faça login no AWS Management Console em https://console.aws.amazon.com/.
2. No menu Suporte, escolha Support Center.
3. No painel Seus casos de suporte, escolha Criar caso.
4. Escolha o link Procurando aumentos no limite de serviço? e preencha o seguinte:
   - Para Tipo de limite, escolha SNS Mensagens de texto.
   - (Opcional) Em Fornecer um link ao site ou à aplicação que enviará mensagens SMS, forneça informações sobre o site, a aplicação ou o serviço que enviará mensagens SMS.
   - (Opcional) Em Qual tipo de mensagem você pretende enviar, escolha o tipo de mensagem que pretende enviar utilizando o código longo:
     - Senhas de uso único: mensagens que fornecem senhas que seus clientes usam para se autenticarem em seu site ou aplicação.
     - Promocional: mensagens não críticas que promovem o seu negócio ou serviço, como ofertas especiais ou anúncios.
     - Transacional: mensagens informativas importantes que oferecem suporte para transações do cliente, como confirmações de pedidos ou alertas de contas. As mensagens transacionais não devem conter conteúdo promocional nem de marketing.
   - (Opcional) Para qual AWS região você enviará mensagens, escolha a região de onde você enviará mensagens.
   - (Opcional) Em Para quais países você planeja enviar mensagens, insira o país ou a região em que deseja comprar códigos simplificados.
   - (Opcional) Em Como seus clientes optam por receber suas mensagens, forneça detalhes sobre seu processo de consentimento.
   - (Opcional) No campo Forneça o modelo de mensagem que você planeja usar para enviar mensagens aos seus clientes, inclua o modelo que será usado.
5. Em Solicitações, preencha as seguintes seções:

- Para Região, escolha a região da qual você enviará mensagens.

> **Note**
>
> A região é obrigatória na seção Solicitações. Mesmo que você tenha fornecido essas informações na seção Detalhes do caso, também é necessário incluí-las aqui.

- Em Tipo de recurso, escolha Limites gerais.
- Em Limite, escolha Aumento de limite de gasto da conta.

6. Em “Novo valor de limite”, insira o valor máximo (em USD) que você pode gastar em SMS a cada mês.
7. Em Descrição do caso, em Descrição do caso de uso, forneça os seguintes detalhes:

   - O site ou o aplicativo da empresa ou do serviço que está enviando mensagens SMS.
   - O serviço que é fornecido pelo seu site ou aplicativo e como suas mensagens SMS contribuem para esse serviço.
   - Como os usuários se cadastram para receber voluntariamente suas mensagens SMS no seu site, aplicativo ou em outro local.

   Se a cota de gastos solicitada (o valor que você especificou em New quota value (Novo valor de cota)) exceder 10.000 USD, forneça os detalhes adicionais a seguir para cada país de destino do sistema de mensagens:

   - Se você estiver usando um ID do remetente ou código simplificado. Se você está usando um ID do remetente, forneça:
     - O ID do remetente.
     - Se o ID do remetente está registrado com operadoras sem fio no país.
   - O máximo esperado transactions-per-second (TPS) para suas mensagens.
   - O tamanho médio das mensagens.
   - O modelo para as mensagens que você envia para o país.
   - (Opcional) Necessidades de codificação de caracteres, se houver.
8. (Opcional) Se você quiser enviar outras solicitações, escolha Adicionar outra solicitação. Se você incluir várias solicitações, forneça as informações necessárias para cada uma delas. Para

obter as informações necessárias, consulte as outras seções em Solicitar suporte para sistema de mensagens SMS do Amazon SNS.

9. Em Opções de contato, para Idioma de contato preferencial, escolha o idioma no qual você deseja receber as comunicações sobre esse caso.
10. Quando terminar, escolha Enviar.

A Suporte equipe fornece uma resposta inicial à sua solicitação em 24 horas.

Para evitar que nossos sistemas sejam usados para enviar conteúdo indesejado ou malicioso, consideramos cuidadosamente cada solicitação. Se for possível, atenderemos à sua solicitação dentro desse período de 24 horas. No entanto, se for necessário solicitar mais informações a você, o tempo de resolução poderá ser mais longo.

Se o seu caso de uso não estiver alinhado com nossas políticas, talvez não seja possível atender à sua solicitação.

Etapa 2: atualizar suas configurações de SMS no console do Amazon SNS

Depois de receber uma notificação de que a cota de gastos mensais foi aumentada, ajuste a cota de gastos de sua conta no console do Amazon SNS.

> **Important**
>
> Você deve concluir as etapas a seguir para que seu limite de gastos de SMS seja aumentado.

Como ajustar sua cota de gastos no console

1. Faça login no console do Amazon SNS.
2. Abra o menu de navegação à esquerda, expanda Dispositivos móveis e escolha Mensagens de texto (SMS).
3. Na página Mobile text messaging (SMS) [Mensagens de texto (SMS) em dispositivos móveis], na seção Text messaging preferences (Preferências de mensagens de texto), escolha Edit (Editar).
4. Na página Editar preferências de mensagens de texto, na seção Detalhes, insira o novo limite de gastos de SMS no campo Limite de gastos da conta.

> **Note**
>
> É possível receber um aviso de que o valor inserido é maior que o limite de gastos padrão. Pode ignorar esse erro.

5. Escolha Salvar alterações.

> **Note**
>
> Se receber um erro “Parâmetro inválido”, verifique o contato do AWS Support e confirme se o novo limite de gastos de SMS foi inserido corretamente. Se você ainda tiver algum problema, abra um caso no AWS Support Center.

Ao criar seu caso no Suporte Centro, não se esqueça de incluir todas as informações necessárias para o tipo de solicitação que você está enviando. Caso contrário, Suporte deverá entrar em contato com você para obter essas informações antes de continuar. Ao enviar um caso detalhado, você ajuda a garantir que seu caso seja realizado sem atrasos. Para ver os detalhes necessários para tipos específicos de solicitações de SMS, consulte os tópicos a seguir.

Para obter mais informações sobre o remetente IDs, consulte a seguinte documentação no Guia do AWS End User Messaging SMS usuário:

AWS End User Messaging SMS Tópico	Descrição
Solicitar um aumento de cota de gastos	Sua cota de gastos determina quanto dinheiro você pode gastar enviando mensagens SMS por AWS End User Messaging SMS mês.
Abra um caso na central de suporte para obter um ID de remetente	Se você planeja enviar mensagens aos destinatários de um país onde o remetente IDs é obrigatório, você pode solicitar um ID de remetente criando um novo caso no Centro. Suporte

Melhores práticas para mensagens SMS do Amazon SNS

> **Important**
>
> O Guia do desenvolvedor de SMS do Amazon SNS foi atualizado. O Amazon SNS foi integrado com AWS End User Messaging SMS para a entrega de mensagens SMS. Este Guia contém as informações mais recentes sobre como criar, configurar e gerenciar as mensagens SMS do Amazon SNS.

Os usuários de telefone celular tendem a ter uma tolerância muito baixa com mensagens SMS não solicitadas. As taxas de resposta para campanhas SMS não solicitadas quase sempre serão baixas e, portanto, o retorno do investimento será fraco.

Além disso, as operadoras de celular continuamente fazem auditoria em remetentes de SMS em massa. Elas limitam ou bloqueiam mensagens de números que elas determinam que estão enviando mensagens não solicitadas.

O envio de conteúdo não solicitado também é uma violação da Política de uso aceitável da AWS. A equipe do Amazon SNS audita campanhas de SMS de maneira rotineira e poderá limitar ou bloquear a sua capacidade de enviar mensagens se você estiver enviando mensagens não solicitadas.

Por fim, em muitos países, regiões e jurisdições, há graves penalidades para o envio de mensagens SMS não solicitadas. Por exemplo, nos Estados Unidos, o Telephone Consumer Protection Act (TCPA) afirma que os consumidores podem ganhar indenizações de US$ 500 a US$ 1.500 (pagos pelo remetente) para cada mensagem não solicitada que recebem.

Esta seção descreve várias práticas recomendadas que podem ajudar você a melhorar seu envolvimento com os clientes e evitar multas pesadas. No entanto, observe que esta seção não contém orientação jurídica. Sempre consulte seu advogado para obter orientação jurídica.

Cumprir as leis, os regulamentos e os requisitos da operadora

Você pode enfrentar multas e penalidades significativas se violar as leis e os regulamentos dos lugares onde seus clientes residem. Por esse motivo, é importante compreender as leis relacionadas a mensagens SMS em cada país ou região em que você faz negócios.

A lista a seguir inclui links para as principais leis que se aplicam às comunicações SMS em mercados importantes em todo o mundo.

- Estados Unidos: o Telephone Consumer Protection Act de 1991, também conhecido como TCPA, aplica-se a certos tipos de mensagens SMS. Para obter mais informações, consulte regras e regulamentos no site da Federal Communications Commission.
- Reino Unido: as Privacy and Electronic Communications (EC Directive) Regulations 2003, também conhecidas como PECR, aplicam-se a certos tipos de mensagens SMS. Para obter mais informações, consulte O que são PECR? no site do Information Commissioner's Office do Reino Unido.
- União Europeia: a Privacy and Electronic Communications Directive 2002, também conhecida como ePrivacy Directive, aplica-se a alguns tipos de mensagens SMS. Para obter mais informações, consulte o texto completo da lei no site Europa.eu.
- Canadá: o Fighting Internet and Wireless Spam Act, mais comumente conhecido como Anti-Spam Law ou CASL do Canadá, aplica-se a certos tipos de mensagens SMS. Para obter mais informações, consulte o texto completo da lei no site do Parlamento do Canadá.
- Japão: o Act on Regulation of Transmission of Specific Electronic Mail pode ser aplicável a determinados tipos de mensagens SMS. Para obter mais informações, consulte Contramedidas contra spam do Japão no site do Ministry of Internal Affairs and Communications japonês.

Como remetente, essas leis podem se aplicar a você mesmo que sua empresa ou organização não esteja sediada em um desses países. Algumas das leis nesta lista foram originalmente criadas para abordar e-mails ou telefonemas não solicitados, mas foram interpretados ou expandidos para que fossem aplicados às mensagens SMS. Outros países e regiões podem ter suas próprias leis relacionadas à transmissão de mensagens SMS. Consulte um advogado em cada país ou região em que seus clientes estão localizados para obter orientação jurídica.

Em muitos países, as operadoras locais têm autoridade para determinar que tipo de tráfego flui em suas redes. Isso significa que elas podem impor restrições ao conteúdo de SMS que exceda os requisitos mínimos das leis locais.

Obter permissão

Nunca envie mensagens a destinatários que não tenham solicitado explicitamente o recebimento dos tipos específicos de mensagens que você planeja enviar. Não compartilhe listas de aceitação, mesmo entre organizações da mesma empresa.

Se os destinatários puderem se cadastrar para receber suas mensagens usando um formulário on-line, adicione sistemas para evitar que scripts automatizados inscrevam pessoas sem o

conhecimento delas. Você também deve limitar o número de vezes que um usuário pode enviar um número de telefone em uma única sessão.

Ao receber uma solicitação de aceitação de SMS, envie uma mensagem ao destinatário pedindo para confirmar que ele deseja receber mensagens de você. Não envie a esse destinatário mensagens adicionais enquanto ele não confirmar a assinatura. Uma mensagem de confirmação de inscrição pode ser parecida com o exemplo a seguir:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may apply. Reply HELP for help, STOP to cancel.
```

Mantenha registros que incluem a data, a hora e a origem de cada solicitação de inclusão e confirmação de inscrição. Isso pode ser útil se uma operadora ou uma agência reguladora solicitar, e também pode ajudar você a realizar auditorias de rotina de sua lista de clientes.

Fluxo de trabalho de aceitação

Em alguns casos (como o registro de código curto ou de chamada gratuita dos EUA), as operadoras de celular exigem que você forneça modelos ou capturas de tela de todo o seu fluxo de trabalho de aceitação. Os modelos ou as capturas de tela devem ser bem semelhantes ao fluxo de trabalho de aceitação que seus destinatários concluirão.

Seus modelos ou capturas de tela devem incluir todas as divulgações necessárias listadas abaixo para manter o mais alto nível de conformidade.

Divulgações obrigatórias

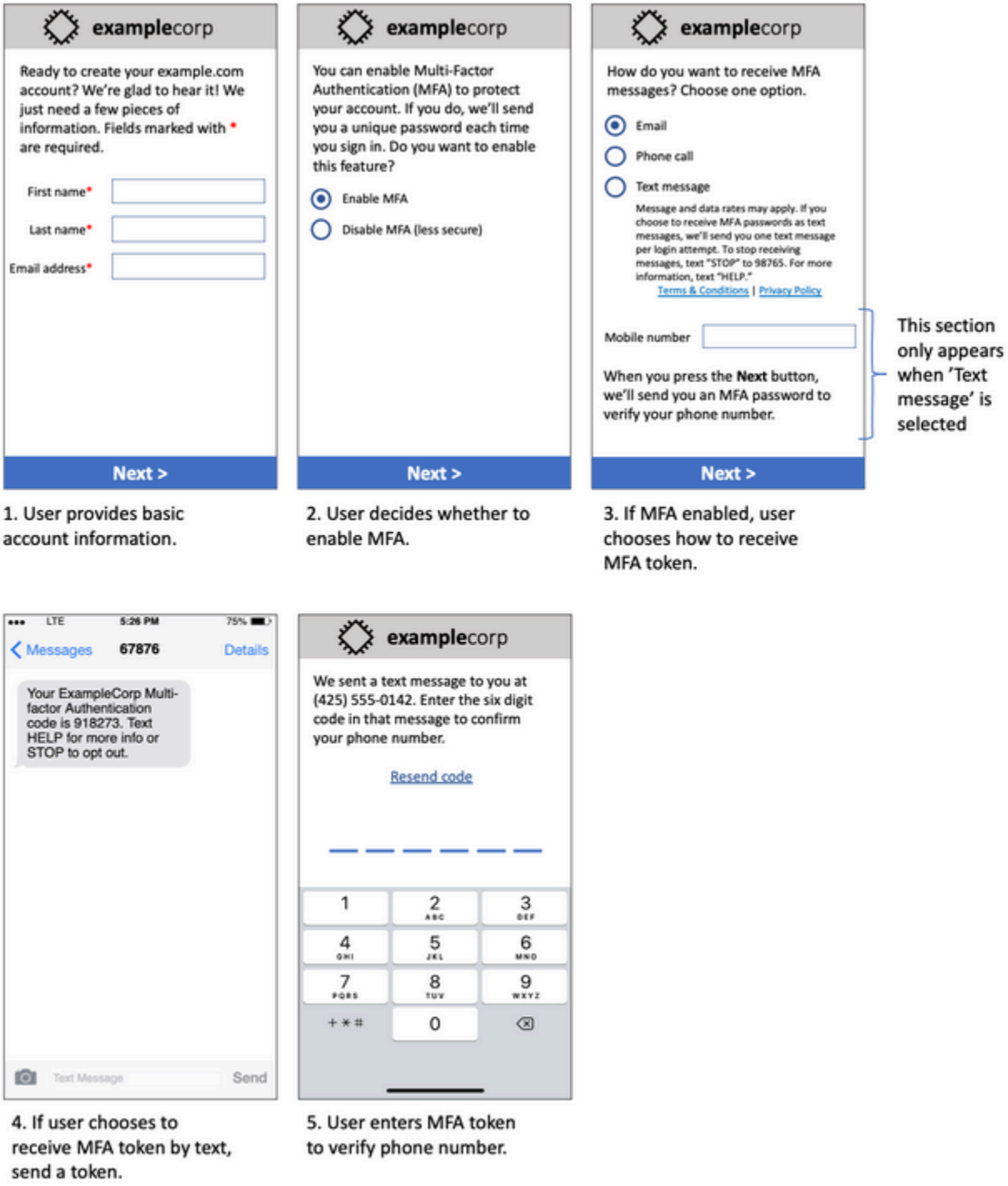
- Uma descrição do caso de uso de mensagens que você enviará por meio de seu programa.
- A frase “Taxas de mensagens e dados podem ser aplicadas”.
- Uma indicação da frequência com que os destinatários receberão suas mensagens. Por exemplo, um programa de mensagens recorrentes pode indicar “uma mensagem por semana”. Um caso de uso de senha de uso único ou autenticação multifator pode indicar “a frequência da mensagem varia” ou “uma única mensagem por tentativa de login”.
- Links para seus documentos de termos e condições e política de privacidade.

Motivos comuns de rejeição de aceitação não compatíveis

- Se o nome da empresa fornecido não corresponder ao fornecido no modelo ou na captura de tela. Qualquer relação não óbvia deve ser explicada na descrição do fluxo de trabalho de aceitação.

- Se parecer que uma mensagem será enviada ao destinatário, mas nenhum consentimento for obtido explicitamente antes disso. O consentimento explícito é um requisito de todas as mensagens.
- Se parecer que é necessário receber uma mensagem de texto para cadastramento em um serviço. Isso não será compatível se o fluxo de trabalho não fornecer nenhuma alternativa ao recebimento de uma mensagem de aceitação em outro formato, como e-mail ou chamada de voz.
- Se o teor da aceitação for apresentado inteiramente nos termos de serviço. As divulgações devem sempre ser apresentadas ao destinatário no momento da aceitação, em vez de armazenadas em um documento de política vinculado.
- Se um cliente consentir em receber um tipo de mensagem sua e você enviar a ele outros tipos de mensagem de texto. Por exemplo, ele consente em receber senhas de uso único, mas também recebe mensagens sobre enquetes e pesquisas.
- Se as divulgações exigidas (listadas acima) não forem apresentadas aos destinatários.

O exemplo a seguir está em conformidade com os requisitos das operadoras de celular para um caso de uso de autenticação multifator.



Modelo de caso de uso da autenticação multifator

Ele contém texto e imagens finalizados e mostra todo o fluxo de aceitação, complementado por anotações. No fluxo de aceitação, o cliente precisa tomar medidas distintas e intencionais para

fornecer seu consentimento para receber mensagens de texto e todas as divulgações necessárias estão incluídas.

Outros tipos de fluxo de trabalho de aceitação

As operadoras de celular também aceitarão fluxos de trabalho de aceitação fora de aplicações e sites, como aceitação verbal ou por escrito, se estiverem em conformidade com o descrito acima. Um fluxo de trabalho de aceitação compatível e um script verbal ou por escrito obterão o consentimento explícito do destinatário para receber um tipo específico de mensagem. São exemplos o script verbal que um agente de suporte usa para obter consentimento antes de gravar em um banco de dados de serviços ou um número de telefone listado em um folheto promocional. Para fornecer uma maquete desses tipos de fluxo de trabalho de aceitação, você pode fornecer uma captura de tela do seu script de aceitação, material de marketing ou banco de dados onde os números são coletados. As operadoras de celular poderão ter perguntas adicionais sobre esses casos de uso se uma aceitação não estiver clara ou se o caso de uso exceder determinados volumes.

Não enviar para listas antigas

As pessoas mudam de número de telefone com frequência. Um número de telefone pelo qual você obteve consentimento para entrar em contato há dois anos pode pertencer a outra pessoa hoje. Não use listas antigas de números de telefone para um novo programa de mensagens; se você fizer isso, é provável que algumas mensagens sejam malsucedidas porque o número não está mais funcionando e algumas pessoas optam por não aceitar as mensagens porque não se lembram de ter dado consentimento.

Fazer auditoria em suas listas de clientes

Se você envia campanhas de SMS recorrentes, faça uma auditoria em suas listas de cliente regularmente. Fazer uma auditoria nas listas de clientes garante que somente os clientes interessados em receber as mensagens são aqueles que as recebem.

Ao fazer uma auditoria em sua lista, envie a cada cliente incluído uma mensagem que lembre a ele que está inscrito e ofereça informações sobre o cancelamento da inscrição. Uma mensagem de lembrete pode ser parecida com o exemplo a seguir:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

Manter registros

Mantenha registros que mostram quando cada cliente solicitou o recebimento de mensagens SMS e quais mensagens você enviou para cada cliente. Muitos países e regiões do mundo exigem remetentes de SMS para manter esses registros de uma forma que possam ser facilmente recuperados. Operadoras de celular também podem solicitar essas informações a qualquer momento. As informações exatas que você deve fornecer variam de acordo com o país ou região. Para obter mais informações sobre os requisitos para manter registros, analise os regulamentos sobre o envio de mensagens SMS comerciais em cada país ou região dos clientes.

Às vezes, uma operadora ou agência reguladora pedem para fornecermos um comprovante de que um cliente optou por receber mensagens de você. Nessas situações, entre em Suporte contato com você com uma lista das informações solicitadas pela operadora ou agência. Se você não conseguir fornecer as informações necessárias, poderemos pausar a sua capacidade de enviar outras mensagens SMS.

Torne suas mensagens claras, sinceras e concisas

O SMS é um meio que tem especificidades. O character-per-message limite de 160 significa que suas mensagens precisam ser concisas. Técnicas que você pode usar em outros canais de comunicação, como e-mail, podem não se aplicar ao canal de SMS e até parecer desonestas ou enganosas quando usadas com mensagens SMS. Se o conteúdo de suas mensagens não estiver de acordo com as práticas recomendadas, os destinatários poderão ignorá-las; na pior das hipóteses, as operadoras de celular poderão identificar suas mensagens como spam e bloquear futuras mensagens de seu número de telefone.

Esta seção fornece algumas dicas e ideias para criar um corpo de mensagem SMS eficaz.

Identificar-se como remetente

Seus destinatários devem saber imediatamente que uma mensagem é sua. Os remetentes que seguem essa prática recomendada incluem um nome de identificação (“nome do programa”) no início de cada mensagem.

Não elabore mensagens deste tipo:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Em vez disso, tente:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

Não tente fazer com que sua mensagem pareça uma person-to-person mensagem

Alguns profissionais de marketing ficam tentados a dar um toque pessoal às mensagens SMS fazendo com que pareçam vir de uma pessoa. No entanto, essa técnica pode fazer sua mensagem parecer uma tentativa de phishing.

Não elabore mensagens deste tipo:

```
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.
```

Em vez disso, tente:

```
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.
```

Ter cuidado ao falar sobre dinheiro

Os golpistas geralmente se aproveitam do desejo das pessoas de economizar e receber dinheiro. Evite que as ofertas pareçam muito boas para serem verdadeiras. Não use a sedução de dinheiro para enganar as pessoas. Não use símbolos monetários para indicar dinheiro.

Não elabore mensagens deste tipo:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Em vez disso, tente:

```
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.
```

Usar somente os caracteres necessários

Muitas vezes, as empresas tendem a proteger suas marcas registradas incluindo símbolos como ™ ou ® nas mensagens. No entanto, esses símbolos não fazem parte do conjunto padrão de caracteres (conhecido como alfabeto GSM) que pode ser incluído em uma mensagem SMS de 160 caracteres. Quando você envia uma mensagem com esses caracteres, ela é enviada automaticamente usando um sistema de codificação de caracteres diferente, que aceita apenas 70 caracteres por parte da mensagem. Por isso, sua mensagem pode ser dividida em várias partes. Como há cobrança por cada parte da mensagem enviada, isso pode ter um custo superior ao que você espera para enviar a mensagem completa. Além disso, os destinatários podem receber várias mensagens sequenciais de você em vez de uma única mensagem. Para obter mais informações sobre a codificação de caracteres do SMS, consulte Limites de caracteres de SMS no Amazon SNS.

Não elabore mensagens deste tipo:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Em vez disso, tente:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> Os dois exemplos anteriores são quase idênticos, mas o primeiro exemplo contém um símbolo de marca registrada (®), o qual não faz parte do alfabeto GSM. Por isso, o primeiro exemplo é enviado como duas partes da mensagem, enquanto o segundo é enviado como uma parte da mensagem.

Usar links válidos e seguros

Se a sua mensagem incluir links, confira-os para verificar se eles funcionam. Teste os links em um dispositivo fora da rede corporativa para garantir que eles sejam resolvidos corretamente. Devido ao limite de 160 caracteres das mensagens SMS, mensagens muito longas URLs podem ser divididas em várias mensagens. Você deve usar domínios de redirecionamento para fornecer abreviações. URLs No entanto, você não deve usar serviços gratuitos de encurtamento de links, como tinyurl.com

ou bitly.com, porque as operadoras tendem a filtrar mensagens que incluem links nesses domínios. No entanto, você pode usar serviços pagos de encurtamento de links, desde que os links apontem para um domínio dedicado ao uso exclusivo de sua empresa ou organização.

Não elabore mensagens deste tipo:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Em vez disso, tente:

```
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.
```

Limitar o número de abreviações utilizadas

A limitação de 160 caracteres do canal de SMS leva alguns remetentes a acreditar que precisam usar abreviações extensivamente em suas mensagens. No entanto, o uso excessivo de abreviações pode parecer pouco profissional para muitos leitores e fazer com que alguns usuários denunciem sua mensagem como spam. É plenamente possível escrever uma mensagem coerente sem usar um número excessivo de abreviações.

Não elabore mensagens deste tipo:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Em vez disso, tente:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Responder de maneira apropriada

Quando um destinatário responde suas mensagens, responda com informações úteis. Por exemplo, quando um cliente responder a uma de suas mensagens com a palavra-chave "AJUDA", envie informações sobre o programa no qual ele está inscrito, o número de mensagens que você vai enviar por mês e as maneiras com as quais ele pode entrar em contato com você para obter mais informações. Uma resposta a uma mensagem de AJUDA pode ser parecida com o exemplo a seguir:

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Quando um cliente responder com a palavra-chave "PARAR", explique que ele não receberá mais mensagens. Uma resposta a uma mensagem de PARAR pode ser parecida com o exemplo a seguir:

```
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.
```

Ajustar seu envio com base no envolvimento

As prioridades de seus clientes podem mudar ao longo do tempo. Se os clientes não acham mais suas mensagens úteis, eles podem querer cancelar a inscrição para suas mensagens ou até mesmo informar suas mensagens como não solicitadas. Por esses motivos, é importante que você ajuste suas práticas de envio com base no envolvimento do cliente.

Você precisa ajustar a frequência de suas mensagens para clientes que raramente interagem com elas. Por exemplo, se você envia mensagens semanais para clientes envolvidos, pode criar uma compilação mensal separada para os clientes com menos envolvimento.

Por fim, remova das suas listas de clientes aqueles que não têm nenhum envolvimento. Essa etapa impede que os clientes fiquem frustrados com suas mensagens. Isso também gera economia e ajuda a proteger sua reputação como remetente.

Enviar em momentos adequados

Somente envie mensagens durante o horário comercial normal. Se você envia mensagens na hora do jantar ou no meio da noite, há uma boa chance de que seus clientes cancelarão a inscrição de suas listas para não serem mais perturbados. Além disso, não faz sentido enviar mensagens SMS quando os clientes não podem responder a elas imediatamente.

Se você enviar campanhas ou jornadas para públicos muito grandes, confira as taxas de throughput de seus números de originação. Divida o número de destinatários pela taxa de throughput para determinar quanto tempo levará para enviar mensagens a todos os destinatários.

Evitar a fadiga entre canais

Se você usa vários canais de comunicação (como e-mail, SMS e mensagens push) em suas campanhas, não envie a mesma mensagem em cada canal. Quando você envia a mesma

mensagem ao mesmo tempo em mais de um canal, seus clientes provavelmente percebem o comportamento de envio como irritante em vez de útil.

Usar códigos curtos dedicados

Se você usa códigos simplificados, mantenha um separado para cada marca e cada tipo de mensagem. Por exemplo, se a sua empresa tem duas marcas, use um código simplificado separado para cada uma. Da mesma forma, se você envia mensagens promocionais e transacionais, use um código simplificado separado para cada tipo de mensagem. Para saber mais sobre como solicitar códigos curtos, consulte Solicitar códigos curtos para mensagens SMS AWS End User Messaging SMS no Guia do usuário do AWS End User Messaging SMS .

Verificar seus números de telefone de destino

Ao enviar mensagens SMS pelo Amazon SNS, cada parte da mensagem enviada é cobrada. O preço pago por parte da mensagem varia de acordo com o país ou a região do destinatário. Para obter mais informações sobre preços do SMS, consulte Definição de preço do SMS da AWS mundial.

Quando o Amazon SNS aceita uma solicitação para enviar uma mensagem SMS (como resultado de uma chamada para a SendMessagesAPI ou como resultado do lançamento de uma campanha ou jornada), você é cobrado pelo envio dessa mensagem. Essa afirmação se aplica mesmo que o destinatário pretendido não receba realmente a mensagem. Por exemplo, se o número de telefone do destinatário não estiver mais funcionando ou se o número para o qual você enviou a mensagem não for um número de celular válido, mesmo assim haverá cobrança pelo envio da mensagem.

O Amazon SNS aceita solicitações válidas para enviar mensagens SMS e tenta entregá-las. Por esse motivo, você deve validar se os números de telefone para os quais você envia mensagens são números de celular válidos. Você pode usar AWS End User Messaging SMS para enviar uma mensagem de teste para determinar se um número de telefone é válido e que tipo de número é (como celular, telefone fixo ou VoIP). Para obter mais informações, consulte Enviar uma mensagem de teste com o simulador de SMS no Guia do usuário do AWS End User Messaging SMS .

Ter a redundância em mente ao projetar

Para programas de mensagens essenciais, recomendamos que você configure o Amazon SNS em mais de uma Região da AWS. O Amazon SNS está disponível em vários. Regiões da AWS Para obter uma lista de regiões onde o Amazon SNS está disponível, consulte a Referência geral da AWS.

Os números de telefone que você usa para mensagens SMS, inclusive códigos curtos, códigos longos, números gratuitos e números 10DLC, não podem ser replicados entre Regiões da AWS. Por

isso, para usar o Amazon SNS em várias regiões, você deve solicitar números de telefone separados em cada região em que deseja usar o Amazon SNS. Por exemplo, se você usar um código curto para enviar mensagens de texto para destinatários nos Estados Unidos da América, precisará solicitar códigos curtos separados em Região da AWS cada um dos que planeja usar.

Em alguns países, você também pode usar vários tipos de número de telefone para aumentar a redundância. Por exemplo, nos Estados Unidos, você pode solicitar códigos curtos, números 10DLC e números gratuitos. Cada um desses tipos de número de telefone segue um caminho diferente até o destinatário. Ter vários tipos de números de telefone disponíveis — no mesmo Região da AWS ou distribuídos em vários Regiões da AWS— fornece uma camada adicional de redundância, que pode ajudar a melhorar a resiliência.

Limites e restrições de SMS

Para ver os limites e as restrições de SMS, consulte Limites e restrições de SMS e MMS no Guia do usuário do AWS End User Messaging SMS .

Gerenciamento de palavras-chave de cancelamento

Os destinatários de SMS podem usar seus dispositivos para cancelar o recebimento de mensagens respondendo com uma palavra-chave. Para obter mais informações, consulte Cancelar recebimento de mensagens SMS.

CreatePool

Use a ação da API `CreatePool` para criar um grupo e associar uma identidade de origem especificada ao grupo. Para obter mais informações, consulte CreatePoolem Referência AWS End User Messaging SMS da API.

PutKeyword

Use a ação da API `PutKeyword` para criar ou atualizar uma configuração de uma palavra-chave em um grupo ou número de telefone de origem. Para obter mais informações, consulte PutKeywordem Referência AWS End User Messaging SMS da API.

Gerenciar configurações de números

Para gerenciar as configurações dos códigos curtos e longos dedicados que você solicitou ao AWS Support e atribuiu à sua conta, consulte Alterar os recursos de um número de telefone com a AWS CLI entrada AWS End User Messaging SMS.

Limites de caracteres de SMS no Amazon SNS

Uma mensagem SMS pode ter até 140 bytes de informação. O número de caracteres que você pode incluir em uma única mensagem SMS depende do tipo de caracteres contidos na mensagem.

Se a mensagem tiver somente caracteres do conjunto de caracteres GSM 03.38, também conhecido como alfabeto GSM de 7 bits, ela poderá ter até 160 caracteres. Se a mensagem tiver caracteres que não façam parte do conjunto de caracteres GSM 03.38, ela poderá ter até 70 caracteres. Ao enviar uma mensagem SMS, o Amazon SNS determina automaticamente a codificação mais eficiente a ser usada.

Quando uma mensagem tiver mais do que o número máximo de caracteres, ela será dividida em várias partes. Quando as mensagens são divididas em várias partes, cada parte contém informações adicionais sobre a parte da mensagem que a precede. Quando o dispositivo do destinatário recebe partes da mensagem que estão separadas dessa forma, ele usa essas informações adicionais para garantir que todas as partes da mensagem sejam exibidas na ordem correta. Dependendo da operadora de celular e do dispositivo do destinatário, várias mensagens podem ser exibidas como uma única mensagem ou como uma sequência de mensagens separadas. Como resultado, o número de caracteres em cada parte da mensagem é reduzido a 153 (para mensagens que contenham somente caracteres GSM 03.38) ou 67 (para mensagens que contenham outros caracteres). É possível estimar quantas partes de mensagem sua mensagem contém antes de enviá-la usando ferramentas de calculadora de comprimento de SMS, muitas das quais estão disponíveis online. O tamanho máximo compatível para qualquer mensagem é de 1,6 mil caracteres GSM ou 630 caracteres não GSM. Para obter mais informações sobre throughput e tamanho de mensagens, consulte Limites de caracteres de SMS no Amazon Pinpoint no Guia do usuário do Amazon Pinpoint.

Para exibir o número de partes da mensagem para cada mensagem enviada, primeiro é necessário habilitar as configurações de streaming de eventos. Ao fazer isso, o Amazon SNS produzirá um evento `_SMS.SUCCESS` quando a mensagem for entregue à operadora de telefonia móvel do destinatário. O registro do evento `_SMS.SUCCESS` contém um atributo chamado `attributes.number_of_message_parts`. Esse atributo especifica o número de partes da mensagem que a mensagem continha.

> **Important**
>
> Quando enviar uma mensagem que contém mais de uma parte, você será cobrado pelo número de partes contidas na mensagem.

Conjunto de caracteres GSM 03.38

A tabela a seguir indica todos os caracteres presentes no conjunto de caracteres GSM 03.38. Se você enviar uma mensagem que inclui apenas os caracteres mostrados na tabela a seguir, a mensagem poderá ter até 160 caracteres.

Caracteres padrão GSM 03.38												
A	B	C	D	E	F	G	H	eu	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	p	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						

O conjunto de caracteres GSM 03.38 inclui vários símbolos além dos mostrados na tabela anterior. No entanto, cada um desses caracteres é contado como dois caracteres, pois também inclui um caractere de escape invisível:

- ^
- {
- }
- \
- [
-]

- ~
- |
- €

Por fim, o conjunto de caracteres GSM 03.38 também inclui os seguintes caracteres não impressos:

- Um caractere de espaço.
- Um controle de avanço de linha, que significa o fim de uma linha de texto e o início de outra.
- Um controle de retorno de linha, que move ao início de uma linha de texto (geralmente após um caractere de avanço de linha).
- Um controle de escape, que é adicionado automaticamente aos caracteres indicados na lista anterior.

Exemplos de mensagens

Esta seção contém vários exemplos de mensagens SMS. Para cada exemplo, a seção mostra o número total de caracteres, bem como o número de partes da mensagem.

Exemplo 1: uma mensagem longa que contém somente caracteres no alfabeto GSM 03.38

A mensagem a seguir contém somente caracteres que estão no alfabeto GSM 03.38.

```
Hello Carlos. Your Example Corp. bill of $100 is now available. Autopay is scheduled for next Thursday, April 9. To view the details of your bill, go to https://example.com/bill1.
```

A mensagem anterior contém 180 caracteres, portanto, deve ser dividida em várias partes. Quando uma mensagem é dividida em várias partes, cada parte pode conter 153 caracteres GSM 03.38. Como resultado, esta mensagem será enviada em 2 partes.

Exemplo 2: uma mensagem que contém caracteres multibyte

A mensagem a seguir contém vários caracteres chineses, todos fora do alfabeto GSM 03.38.

```
##################################################·####1994#7#########
```

A mensagem anterior contém 71 caracteres. No entanto, como quase todos os caracteres na mensagem estão fora do alfabeto GSM 03.38, ela será enviada em duas partes. Cada uma dessas partes da mensagem pode conter um máximo de 67 caracteres.

Exemplo 3: uma mensagem que contém um único caractere que não faz parte do GSM

A mensagem a seguir contém um único caractere que não faz parte do alfabeto GSM 03.38. Neste exemplo, o caractere é uma aspa simples de fechamento (’), que é um caractere diferente de um apóstrofo regular ('). Aplicativos de processamento de texto, como o Microsoft Word, muitas vezes substituem automaticamente os apóstrofos por aspas simples de fechamento. Se fizer rascunhos de suas mensagens SMS no Microsoft Word e colá-las no Amazon SNS, você deverá remover esses caracteres especiais e substituí-los por apóstrofos.

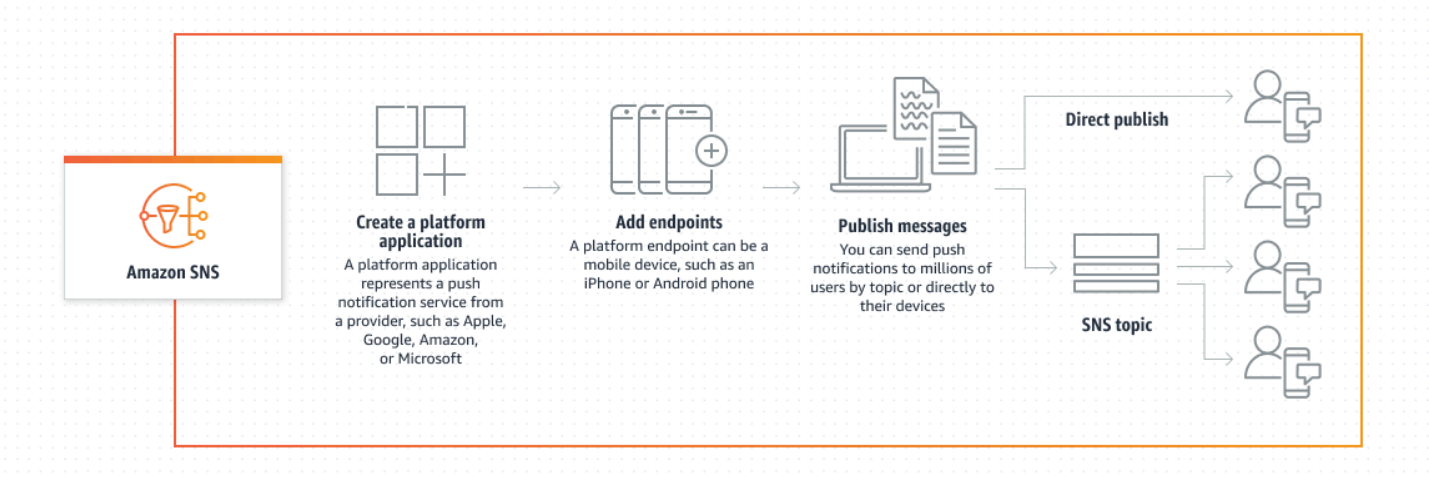
```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

A mensagem anterior contém 130 caracteres. No entanto, como ela contém o caractere de aspas simples de fechamento, que não faz parte do alfabeto GSM 03.38, ela será enviada em duas partes.

Se você substituir o caractere de aspas simples de fechamento nesta mensagem por um apóstrofo (que faz parte do alfabeto GSM 03.38), a mensagem será enviada em uma única parte.

Enviar notificações por push para dispositivos móveis com o Amazon SNS

Você pode usar o Amazon SNS para enviar mensagens de notificação por push diretamente para aplicações em dispositivos móveis. As mensagens de notificação por push enviadas para um endpoint móvel podem aparecer no aplicativo móvel como alertas de mensagem, atualizações de distintivo ou alertas com som.



Tópicos

Como funcionam as notificações ao usuário do Amazon SNS

Você envia as mensagens de notificação por push para dispositivos móveis e desktops usando um dos seguintes serviços de notificação por push compatíveis:

- Amazon Device Messaging (ADM)
- Serviço de notificação push da Apple (APNs) para iOS e Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Microsoft Push Notification Service for Windows Phone (MPNS)
- Windows Push Notification Services (WNS)

Os serviços de notificação push, como APNs o FCM, mantêm uma conexão com cada aplicativo e dispositivo móvel associado registrado para usar o serviço. Quando uma aplicação e um dispositivo móvel são registrados, o serviço de notificação por push retorna um token de dispositivo. O Amazon SNS usa o token de dispositivo para criar um endpoint móvel, para o qual é possível enviar mensagens diretas de notificação por push. Para que o Amazon SNS se comunique com os diferentes serviços de notificação por push, você envia suas credenciais de serviço de notificação por

push ao Amazon SNS para que sejam usadas em seu nome. Para obter mais informações, consulte Configurar notificações de push com o Amazon SNS.

Além de enviar mensagens de notificação por push diretas, você também pode usar o Amazon SNS para enviar mensagens para endpoints móveis inscritos em um tópico. O conceito é o mesmo que para inscrever outros tipos de endpoint, como Amazon SQS, HTTP/S, e-mail e SMS, em um tópico, conforme descrito em O que é o Amazon SNS?. A diferença é que o Amazon SNS se comunica usando os serviços de notificação por push para que os endpoints móveis inscritos recebam mensagens de notificação por push enviadas para o tópico.

Configurar notificações de push com o Amazon SNS

1. Obtenha as credenciais e o token de dispositivo para as plataformas móveis para as quais deseja oferecer suporte.
2. Use as credenciais para criar um objeto de aplicativo da plataforma (`PlatformApplicationArn`) usando o Amazon SNS. Para obter mais informações, consulte Criar uma aplicação da plataforma Amazon SNS.
3. Use as credenciais retornadas para solicitar um token para o dispositivo e aplicativo móvel dos serviços de notificação por push. O token que você recebe representa seu dispositivo e aplicativo móvel.
4. Use o token de dispositivo e o `PlatformApplicationArn` para criar um objeto de endpoint da plataforma (`EndpointArn`) usando o Amazon SNS. Para obter mais informações, consulte Configurar um endpoint da plataforma Amazon SNS para notificações móveis.
5. Use o `EndpointArn` para publicar uma mensagem em um aplicativo em um dispositivo móvel. Para obter mais informações, consulte Mensagens diretas para dispositivos móveis do Amazon SNS e a API Publish na Referência da API do Amazon Simple Notification Service.

Configurar uma aplicação móvel no Amazon SNS

Este tópico descreve como configurar aplicativos móveis AWS Management Console usando as informações descritas emPré-requisitos para notificações ao usuário do Amazon SNS.

Pré-requisitos para notificações ao usuário do Amazon SNS

Para começar a usar as notificações por push para dispositivos móveis do Amazon SNS, é necessário o seguinte:

- Um conjunto de credenciais para se conectar a um dos serviços de notificação push compatíveis: ADM, Baidu APNs, FCM, MPNS ou WNS.
- Um token de dispositivo ou ID de registro para o dispositivo e aplicativo móvel.
- Amazon SNS configurado para enviar mensagens de notificação por push para os endpoints móveis.
- Um aplicativo móvel que está registrado e configurado para usar um dos serviços de notificações por push compatíveis.

O registro da aplicação em um serviço de notificação por push requer várias etapas. O Amazon SNS precisa de algumas das informações que você fornece ao serviço de notificação por push para enviar mensagens de notificação por push diretas ao endpoint móvel. De modo geral, você precisa das credenciais necessárias para a conexão com o serviço de notificação por push, um token de dispositivo ou um ID de registro (que representam o dispositivo e o aplicativo móvel) recebido do serviço de notificação por push, e o aplicativo móvel registrado no serviço de notificação por push.

O formato exato das credenciais difere entre plataformas móveis, mas, em cada caso, essas credenciais devem ser enviadas durante uma conexão com a plataforma. Um conjunto de credenciais é emitido para cada aplicativo móvel e deve ser usado para enviar uma mensagem para qualquer instância do aplicativo.

Os nomes específicos variarão de acordo com qual serviço de notificação por push estiver sendo usado. Por exemplo, ao usar APNs como serviço de notificação push, você precisa de um token de dispositivo. Como alternativa, ao usar o FCM, o token de dispositivo equivalente será chamado de ID de registro. O token de dispositivo ou o ID de registro é uma string enviada para o aplicativo pelo sistema operacional do dispositivo móvel. Ela identifica unicamente uma instância de um aplicativo móvel em execução em um determinado dispositivo móvel e pode ser considerada como identificadores exclusivos desse par de dispositivo e aplicativo.

O Amazon SNS armazena as credenciais (além de outras configurações) como um recurso de aplicação de plataforma. Os tokens de dispositivo (novamente com algumas configurações extras) são representados como objetos chamados endpoints de plataforma. Cada endpoint de plataforma pertence a um aplicativo de plataforma específico, e cada endpoint de plataforma pode ser acessado usando as credenciais armazenadas em seu aplicativo de plataforma correspondente.

As seções a seguir incluem os pré-requisitos para cada um dos serviços de notificação por push com suporte. Depois de obter as informações de pré-requisito, você pode enviar uma mensagem de notificação push usando o push móvel AWS Management Console ou o push móvel do Amazon

SNS. APIs Para obter mais informações, consulte Configurar notificações de push com o Amazon SNS.

Criar uma aplicação da plataforma Amazon SNS

Para enviar notificações do Amazon SNS para endpoints móveis, seja diretamente ou por meio de assinaturas de um tópico, você deve primeiro criar um aplicativo de plataforma. Depois de registrar o aplicativo com AWS, você precisa criar um endpoint para o aplicativo e para o dispositivo móvel. Esse endpoint permite que o Amazon SNS envie mensagens para o dispositivo.

Para criar uma aplicação de plataforma

1. Faça login no console do Amazon SNS.
2. No painel de navegação, selecione Notificações push.
3. Na seção Aplicações de plataforma, selecione Criar aplicação de plataforma.
4. Escolha a Região da AWS. Para obter uma lista das regiões da AWS em que você pode criar aplicações para dispositivos móveis, consulte Regiões suportadas pelo aplicativo móvel Amazon SNS.
5. Insira os seguintes detalhes do aplicativo:
   - Nome do aplicativo — Forneça um nome para o aplicativo da sua plataforma. O nome deve ter entre 1 e 256 caracteres e pode conter letras maiúsculas e minúsculas, números, sublinhados, hífens e pontos.
   - Plataforma de notificação push — selecione o serviço de notificação apropriado no qual o aplicativo está registrado (por exemplo, Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM)).
6. Dependendo da plataforma selecionada, você precisará fornecer credenciais específicas:
   - Para APNs(Apple Push Notification Service) — Escolha entre autenticação baseada em token ou baseada em certificado.
     - Para autenticação baseada em token, faça upload de um arquivo.p8 (gerado por meio do Keychain Access).
     - Para autenticação baseada em certificado, faça upload de um arquivo.p12 (também exportado do Keychain Access).
   - Para FCM (Firebase Cloud Messaging) — insira a chave do servidor no Firebase Console.
   - Para outras plataformas (como ADM ou GCM) — Insira as respectivas chaves ou credenciais de API.

7. Depois de inserir os detalhes necessários, escolha Criar aplicativo de plataforma. Essa ação registra o aplicativo no Amazon SNS e cria o objeto de aplicativo da plataforma correspondente.
8. Após a criação, o Amazon SNS gera e retorna um (`PlatformApplicationArn`Amazon Resource Name). Esse ARN identifica de forma exclusiva o aplicativo da plataforma e é usado na criação de endpoints para dispositivos móveis.

Configurar um endpoint da plataforma Amazon SNS para notificações móveis

Quando um aplicativo e um dispositivo móvel se registram em um serviço de notificação por push (como APNs o Firebase Cloud Messaging), o serviço de notificação por push retorna um token do dispositivo. O Amazon SNS usa esse token de dispositivo para criar um endpoint de plataforma, que atua como um destino para enviar mensagens diretas de notificação push para o aplicativo no dispositivo. O endpoint da plataforma serve como uma ponte, roteando as mensagens enviadas pelo Amazon SNS para o serviço de notificação push para entrega no dispositivo móvel correspondente. Para ter mais informações, consulte Pré-requisitos para notificações ao usuário do Amazon SNS e Configurar notificações de push com o Amazon SNS.

Entendendo os tokens de dispositivos e os endpoints da plataforma

Um token de dispositivo identifica de forma exclusiva um dispositivo móvel registrado em um serviço de notificação push (por exemplo APNs, Firebase Cloud Messaging). Quando um aplicativo se registra no serviço de notificação push, ele gera um token de dispositivo específico para esse aplicativo e dispositivo. O Amazon SNS usa esse token de dispositivo para criar um endpoint de plataforma dentro do aplicativo de plataforma correspondente.

O endpoint da plataforma permite que o Amazon SNS envie mensagens de notificação push para o dispositivo por meio do serviço de notificação push, mantendo a conexão entre seu aplicativo e o dispositivo do usuário.

Criar um endpoint de plataforma

Para enviar notificações para uma aplicação com o Amazon SNS, o token do dispositivo da aplicação deve primeiro ser registrado no Amazon SNS chamando a ação de criação de endpoint de plataforma. Essa ação leva o Nome do recurso da Amazon (ARN) do aplicativo de plataforma e o token do dispositivo como parâmetros e retorna o ARN do endpoint de plataforma criado.

A ação do `CreatePlatformEndpoint` faz o seguinte:

- Se o endpoint de plataforma já existir, não o crie novamente. Retorne o ARN do endpoint de plataforma existente para o chamador.
- Se já existir um endpoint de plataforma com o mesmo token de dispositivo, mas com configurações diferentes, não crie o endpoint novamente. Gere uma exceção para o chamador.
- Se o endpoint de plataforma não existir, crie-o. Retorne o ARN do endpoint de plataforma recém-criado para o chamador.

Você não deve chamar a ação de criação do endpoint de plataforma imediatamente sempre que um aplicativo é iniciado, pois essa abordagem nem sempre fornece um endpoint que funciona. Isso pode acontecer, por exemplo, quando um aplicativo for desinstalado e reinstalado no mesmo dispositivo e o endpoint dele já existir mas estiver desabilitado. Para realizar um processo de registro bem-sucedido, você deve fazer o seguinte:

1. Verifique se um endpoint de plataforma existe para esta combinação de aplicativo-dispositivo.
2. Verifique se o token do dispositivo no endpoint de plataforma é o token do dispositivo válido mais recente.
3. Verifique se o endpoint de plataforma está ativado e pronto para uso.

Pseudocódigo

O seguinte pseudocódigo descreve uma prática recomendada para a criação de um endpoint de plataforma atual, em funcionamento e habilitado em uma ampla variedade de condições de inicialização. Essa abordagem funciona se essa for a primeira vez que o aplicativo estiver sendo registrado ou não, se o endpoint de plataforma para este aplicativo já existir e se o endpoint de plataforma estiver ativado, tiver o token de dispositivo correto e assim por diante. É seguro chamá-lo várias vezes consecutivas, já que ele não criará endpoints de plataforma duplicados ou alterará um endpoint de plataforma existente se ele já estiver atualizado e ativado.

```
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
```

Essa abordagem pode ser usada a qualquer momento que o aplicativo precisar se registrar ou efetuar um novo registro. Ela também pode ser usada ao notificar o Amazon SNS sobre uma alteração no token do dispositivo. Nesse caso, você pode chamar a ação com o valor de token do dispositivo mais recente. Alguns pontos a observar nessa abordagem são:

- Há dois casos em que ela pode chamar a ação de criação de endpoint de plataforma. Ela poderá ser chamada no início, quando o aplicativo não souber seu próprio ARN de endpoint de plataforma, como acontece durante um registro feito pela primeira vez. Ela também será chamada se a chamada inicial de ação de `GetEndpointAttributes` falhar com uma exceção não encontrada. Isso também acontecerá se o aplicativo souber o ARN de endpoint, mas ele foi excluído.
- A ação de `GetEndpointAttributes` é chamada para verificar o estado do endpoint de plataforma, mesmo se o endpoint de plataforma for recém-criado. Isso acontece quando o endpoint de plataforma já existe mas está desativado. Nesse caso, a ação de criação de endpoint de plataforma é bem-sucedida, mas não habilita o endpoint de plataforma. Portanto, você deve verificar o estado do endpoint da plataforma antes de retornar o sucesso.

AWS Exemplo de SDK

O código a seguir mostra como implementar o pseudocódigo anterior usando os clientes do Amazon SNS fornecidos pelo. AWS SDKs

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

CLI

AWS CLI

Para criar um endpoint de aplicação de plataforma

O exemplo `create-platform-endpoint` a seguir cria um endpoint para a aplicação de plataforma especificada usando o token especificado.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Saída:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s

            """;

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

       String token = args[0];
       String platformApplicationArn = args[1];
       SnsClient snsClient = SnsClient.builder()
           .region(Region.US_EAST_1)
           .build();
```

```
        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

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

Para obter mais informações, consulte Ações da API de push para dispositivos móveis.

Solução de problemas

Chamar repetidamente a criação de endpoint da plataforma com um token de dispositivo desatualizado

Especialmente para endpoints do FCM, você pode pensar que é melhor armazenar o primeiro token de dispositivo emitido pela aplicação e, em seguida, chamar a criação de endpoint de plataforma com esse token de dispositivo sempre na inicialização da aplicação. Isso pode parecer correto, pois a aplicação não precisará gerenciar o estado do token do dispositivo, e o Amazon SNS atualizará automaticamente o token do dispositivo para seu valor mais recente. No entanto, essa solução tem uma série de problemas sérios:

- O Amazon SNS utiliza o feedback do FCM para atualizar os tokens de dispositivo expirados para novos tokens de dispositivo. O FCM retém informações sobre tokens de dispositivo antigos por algum tempo, mas não indefinidamente. Quando o FCM esquecer a conexão entre o token do

dispositivo antigo e o novo token, o Amazon SNS não poderá mais atualizar o token do dispositivo armazenado no endpoint de plataforma para seu valor correto, ele apenas desabilitará o endpoint de plataforma.

- O aplicativo de plataforma conterá vários endpoints de plataforma correspondentes ao mesmo token de dispositivo.
- O Amazon SNS impõe uma cota para o número de endpoints de plataforma que podem ser criados a partir do mesmo token de dispositivo. Finalmente, a criação de novos endpoints falhará com uma exceção de parâmetro inválido e aparecerá a seguinte mensagem de erro: “This endpoint is already registered with a different token.”

Para obter mais informações sobre o gerenciamento de endpoints do FCM, consulte Gerenciamento de endpoints do Firebase Cloud Messaging pelo Amazon SNS.

Habilitar novamente um endpoint de plataforma associado a um token de dispositivo inválido

Quando uma plataforma móvel (como APNs ou FCM) informa ao Amazon SNS que o token do dispositivo usado na solicitação de publicação era inválido, o Amazon SNS desativa o endpoint da plataforma associado ao token do dispositivo. O Amazon SNS rejeitará as publicações posteriores nesse token de dispositivo. Você pode achar melhor simplesmente reativar o endpoint de plataforma e manter a publicação, mas, na maioria dos casos, isso não funcionará: as mensagens que são publicadas não são entregues e o endpoint de plataforma é desativado novamente logo em seguida.

Isso ocorre porque o token de dispositivo associado ao endpoint de plataforma é genuinamente inválido. As entregas destinadas a ele não terão sucesso porque ele não corresponde a nenhum aplicativo instalado. Na próxima vez em que for publicada, a plataforma móvel informará novamente ao Amazon SNS que o token de dispositivo é inválido, e o Amazon SNS desabilitará novamente o endpoint de plataforma.

Para ativar novamente um endpoint de plataforma desabilitada, ele precisa ser associado a um token de dispositivo válido (com uma chamada de ação de definição de atributos de endpoint) e depois habilitado. A partir de então, as entregas para o endpoint dessa plataforma terão sucesso. A única vez que a reabilitação de um endpoint de plataforma sem atualização do seu token de dispositivo funcionará será quando um token de dispositivo associado a esse endpoint for inválido, mas depois tornar-se válido novamente. Isso pode acontecer, por exemplo, quando um aplicativo for desinstalado e instalado novamente no mesmo dispositivo móvel e receber o mesmo token de dispositivo. A abordagem apresentada anteriormente faz isso, garantindo que a reativação de um endpoint de plataforma seja feita somente após verificar se o token de dispositivo associado a ele é o mais atual disponível.

Integração de tokens de dispositivos com o Amazon SNS para notificações móveis

Quando você registra pela primeira vez um aplicativo e um dispositivo móvel em um serviço de notificação, como o Apple Push Notification Service (APNs) e o Firebase Cloud Messaging (FCM), os tokens ou o registro do dispositivo IDs são devolvidos pelo serviço. Esses tokens/ IDs são adicionados ao Amazon SNS para criar um endpoint para o aplicativo e o dispositivo, usando a API. `PlatformApplicationArn` Depois que o endpoint é criado, um `EndpointArn`é retornado, que o Amazon SNS usa para direcionar as notificações para o aplicativo/dispositivo correto.

Você pode adicionar tokens de dispositivo ou registro IDs no Amazon SNS das seguintes formas:

- Adicione manualmente um único token por meio do AWS Management Console
- Fazer upload de vários tokens usando a API `CreatePlatformEndpoint`
- Registre tokens para futuros dispositivos

Para adicionar manualmente um token de dispositivo ou ID de registro

1. Faça login no console do Amazon SNS.
2. No painel de navegação, selecione Notificações push.
3. Na seção Aplicações de plataforma, selecione sua aplicação e escolha Editar. Se você ainda não criou um aplicativo de plataforma, siga o Criar uma aplicação da plataforma Amazon SNS guia para fazer isso agora.
4. Escolha Criar Endpoint.
5. Na caixa Endpoint Token, insira o token ou o ID de registro, dependendo do serviço de notificação que você está usando (por exemplo, ID de registro do FCM).
6. (Opcional) Insira dados adicionais no campo Dados do usuário. Esses dados devem ser codificados em UTF-8 e ter menos de 2 KB.
7. Escolha Criar Endpoint.

Depois que o endpoint for criado, você poderá enviar mensagens diretamente para o dispositivo móvel ou para dispositivos móveis inscritos em um tópico do Amazon SNS.

Para fazer upload de vários tokens usando a **`CreatePlatformEndpoint`** API

As etapas a seguir mostram como usar o aplicativo Java de amostra (`bulkuploadpacote`) fornecido pela AWS para fazer upload de vários tokens (tokens de dispositivo ou registro IDs) no Amazon

SNS. Você pode usar esse aplicativo de amostra para ajudá-lo a começar a fazer upload de tokens existentes.

> **Note**
>
> As etapas a seguir usam o IDE para Java Eclipse. As etapas pressupõem que você tenha instalado o AWS SDK para Java e tenha as credenciais AWS de segurança do seu Conta da AWS. Para obter mais informações, consulte AWS SDK para Java. Para obter mais informações sobre credenciais, consulte as credenciais AWS de segurança no Guia do usuário do IAM.

1. Faça o download e descompacte o arquivo snsmobilepush.zip.
2. Crie um novo projeto Java no Eclipse e importe a `SNSSamples` pasta para o projeto.
3. Baixe a biblioteca OpenCSV e adicione-a ao caminho de construção.
4. No `BulkUpload.properties` arquivo, especifique o seguinte:

   - Seu `ApplicationArn` (ARN do aplicativo da plataforma).
   - O caminho absoluto para seu arquivo CSV contendo os tokens.
   - Registrando nomes de arquivos para tokens bem-sucedidos e fracassados. Por exemplo, `goodTokens.csv` e `badTokens.csv`.
   - (Opcional) Uma configuração para delimitador, caractere de aspa e número de segmentos a serem usados.

   O `BulkUpload.properties` completo deve ser semelhante ao seguinte:

   ```
   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. Execute o BatchCreatePlatformEndpointSampleaplicativo.java para fazer o upload dos tokens para o Amazon SNS. Os tokens enviados com sucesso serão registrados`goodTokens.csv`, enquanto os tokens malformados serão registrados. `badTokens.csv`

Para registrar tokens de dispositivos para futuras instalações de aplicativos

Você tem duas opções para esse processo:

Use o serviço Amazon Cognito

Seu aplicativo móvel pode usar credenciais de segurança temporárias para criar endpoints. O Amazon Cognito é recomendado para gerar credenciais temporárias. Para obter mais informações, consulte o Guia do desenvolvedor do Amazon Cognito

Para rastrear registros de aplicativos, use os eventos do Amazon SNS para receber notificações quando um novo ARNs endpoint for criado.

Como alternativa, você pode usar a `ListEndpointByPlatformApplication`API para recuperar a lista de endpoints registrados.

Usar um servidor de proxy

Se a infraestrutura do seu aplicativo já suportar o registro de dispositivos na instalação, você poderá usar seu servidor como proxy. Ele encaminhará os tokens do dispositivo para o Amazon SNS por meio da `CreatePlatformEndpoint`API.

O ARN do endpoint criado pelo Amazon SNS será retornado e poderá ser armazenado pelo seu servidor para futura publicação de mensagens.

Amazon SNS: métodos de autenticação de notificação push da Apple

É possível autorizar o Amazon SNS a enviar notificações por push para sua aplicação iOS ou macOS fornecendo informações que identifiquem você como desenvolvedor da aplicação. Para autenticar, forneça uma chave ou um certificado ao criar uma aplicação de plataforma. Ambos estão disponíveis em sua conta de Desenvolvedor da Apple.

Chave de assinatura de token

Uma chave de assinatura privada que o Amazon SNS usa para assinar tokens de autenticação do Apple Push Notification Service (APNs).

Se você fornecer uma chave de assinatura, o Amazon SNS usará um token para se autenticar APNs para cada notificação push que você enviar. Com sua chave de assinatura, você pode enviar notificações push para ambientes APNs de produção e sandbox.

Sua chave de assinatura não expira e a mesma chave de assinatura pode ser usada para várias aplicações. Para obter mais informações, consulte Comunique-se com o APNs uso de tokens de autenticação na seção Ajuda da conta do desenvolvedor do site da Apple.

Certificado

Um certificado TLS que o Amazon SNS usa para se autenticar quando você APNs envia notificações push. O certificado pode ser obtido da sua conta de Desenvolvedor da Apple.

Os certificados expiram após um ano. Quando isso acontecer, crie um novo certificado e forneça-o ao Amazon SNS. Para obter mais informações, consulte Estabelecendo uma conexão baseada em certificado APNs no site do desenvolvedor da Apple.

Para gerenciar APNs configurações usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, selecione Notificações push.
3. Na seção Aplicativos da plataforma, selecione o aplicativo cujas APNs configurações você deseja editar e escolha Editar. Se você ainda não criou um aplicativo de plataforma, siga o Criar uma aplicação da plataforma Amazon SNS guia para fazer isso agora.
4. Escolha Editar para modificar as configurações do aplicativo da sua plataforma.
5. Na seção Tipo de autenticação, escolha uma das seguintes opções:
   - Autenticação baseada em tokens (recomendada para integrações modernas APNs)
   - Autenticação baseada em certificado (método antigo)
6. Configure suas credenciais com base no tipo de autenticação:
   - Para autenticação baseada em tokens:
     - Faça upload do arquivo.p8, que é a chave de assinatura do token de autenticação que você baixou da sua conta de desenvolvedor da Apple.
     - Insira o ID da chave de assinatura que você encontra na sua conta de desenvolvedor da Apple. Navegue até Certificados IDs e Perfis, Chaves e selecione a chave que você deseja usar.
     - Forneça o Identificador de Equipe da sua conta de desenvolvedor da Apple. Você pode encontrar isso na página de associação.
     - Insira o identificador do pacote atribuído ao seu aplicativo. Você pode encontrar isso em Certificados IDs e Perfis, Aplicativo IDs.

- Para autenticação baseada em certificado:
  - Faça upload do arquivo.p12 para seu certificado TLS. Esse arquivo pode ser exportado do Keychain Access no macOS depois de baixar o certificado da sua conta de desenvolvedor da Apple.
  - Se você atribuiu uma senha ao seu certificado.p12, insira-a aqui.

7. Depois de inserir as credenciais necessárias, escolha Salvar alterações para atualizar as configurações.

Integração do Amazon SNS à configuração de autenticação do Firebase Cloud Messaging

Este tópico descreve como obter as credenciais necessárias da API do FCM (HTTP v1) do Google para usar com a AWS API e a. AWS CLI AWS Management Console

> **Important**
>
> 26 de março de 2024: o Amazon SNS oferece suporte à API HTTP v1 do FCM para dispositivos Apple e destinos Webpush. Recomendamos que você migre suas aplicações móveis por push existentes para a API mais recente do FCM HTTP v1 até 1º de junho de 2024 para evitar interrupções nas aplicações.
> 18 de janeiro de 2024: o Amazon SNS introduziu o suporte à API HTTP v1 do FCM para entrega de notificações push móveis para dispositivos Android.
> 20 de junho de 2023: Google descontinuou a API HTTP legada do Firebase Cloud Messaging (FCM). O Amazon SNS agora oferece suporte à entrega para todos os tipos de dispositivos usando a API HTTP v1 do FCM. Recomendamos que você migre suas aplicações móveis por push existentes para a API mais recente do FCM HTTP v1 até 1º de junho de 2024 para evitar interrupções nas aplicações.

É possível autorizar o Amazon SNS a enviar notificações por push para suas aplicações fornecendo informações que identifiquem você como desenvolvedor da aplicação. Para autenticar, forneça uma chave de API ou um token ao criar uma aplicação de plataforma. É possível consultar as informações a seguir no console da aplicação do Firebase:

Chave de API

A chave de API é uma credencial usada ao chamar a API legada do Firebase. O FCM Legacy APIs será removido pelo Google em 20 de junho de 2024. Se você estiver usando uma chave de API como credencial da plataforma, atualize a credencial da plataforma selecionando Token como opção e fazendo upload do arquivo JSON associado à aplicação Firebase.

Token

Um token de acesso de curta duração é usado ao chamar a API HTTP v1. Essa é a API sugerida pelo Firebase para enviar notificações por push. Para gerar tokens de acesso, o Firebase fornece aos desenvolvedores um conjunto de credenciais na forma de um arquivo de chave privada (também conhecido como arquivo service.json).

Pré-requisito

Você deve obter as credenciais service.json do FCM antes de começar a gerenciar as configurações do FCM no Amazon SNS. Para obter suas credenciais service.json, consulte Migrar do FCM legado APIs para o HTTP v1 na documentação do Google Firebase.

Gerenciar configurações do FCM usando o CLI

Você pode criar notificações push do FCM usando a AWS API. O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Endpoints e cotas do Amazon Simple Notification Service no Guia do Referência geral da AWS .

Para criar uma notificação push do FCM junto com um tópico AWS (API) do Amazon SNS

Ao usar as credenciais de chave, `PlatformCredential` é `API key`. Ao usar credenciais de token, `PlatformCredential` é um arquivo de chave privada formatado em JSON:

- `CreatePlatformApplication`

Para recuperar um tipo de credencial do FCM para um tópico existente do Amazon SNS (API)AWS

Recupera o tipo de credencial `"AuthenticationMethod": "Token"` ou `"AuthenticationMethod": "Key"`:

- GetPlatformApplicationAttributes

Como definir um atributo do FCM para um tópico existente do Amazon SNS (AWS API)

Define o atributo do FCM:

- SetPlatformApplicationAttributes

Gerenciar configurações do FCM usando o console

Você pode criar notificações push do FCM usando a AWS Command Line Interface (CLI). O número e o tamanho dos recursos do Amazon SNS em uma AWS conta são limitados. Para obter mais informações, consulte Amazon Simple Notification Service endpoints and quotas (Endpoints e cotas do Amazon Simple Notification Service).

Como criar uma notificação por push do FCM junto com um tópico do Amazon SNS (AWS CLI)

Ao usar as credenciais de chave, `PlatformCredential` é `API key`. Ao usar credenciais de token, `PlatformCredential` é um arquivo de chave privada formatado em JSON: Ao usar a AWS CLI, o arquivo deve estar no formato de string e os caracteres especiais devem ser ignorados. Para formatar o arquivo corretamente, o Amazon SNS recomenda usar o seguinte comando: `SERVICE_JSON=`jq @json <<< cat service.json``:

- create-platform-application

Como recuperar um tipo de credencial do FCM para um tópico existente do Amazon SNS (AWS CLI)

Recupera o tipo de credencial `"AuthenticationMethod": "Token"` ou `"AuthenticationMethod": "Key"`:

- get-platform-application-attributes

Como definir um atributo do FCM para um tópico existente do Amazon SNS (AWS CLI)

Define o atributo do FCM:

- set-platform-application-attributes

Gerenciar configurações do FCM (console)

Use as etapas a seguir para inserir e gerenciar suas credenciais do Firebase Cloud Messaging (FCM) no Amazon SNS.

1. Faça login no console do Amazon SNS.

2. No painel de navegação, selecione Notificações push.
3. Na seção Aplicativos de plataforma, selecione o aplicativo da plataforma FCM cujas credenciais você deseja editar e, em seguida, escolha Editar.
4. Na seção Credenciais do Firebase Cloud Messaging, escolha uma das seguintes opções:

   - Autenticação baseada em token (método recomendado) — faça o upload do arquivo de chave privada (JSON) que você baixou do Firebase Console. Esse arquivo contém as credenciais necessárias para gerar tokens de acesso de curta duração para notificações do FCM. Para obter esse arquivo:

     1. Acesse seu console de aplicativos do Firebase.
     2. Nas Configurações do projeto, selecione Cloud Messaging.
     3. Baixe o arquivo JSON da chave privada (para uso no método de autenticação baseado em token).

   - Autenticação de chave de API — Se você preferir usar o método de autenticação de chave de API mais antigo, insira a chave de API do Google no campo fornecido. Para obter esse arquivo:

     1. Acesse seu console de aplicativos do Firebase.
     2. Em Configurações do projeto, selecione Cloud Messaging.
     3. Copie a chave do servidor (chave de API) a ser usada para enviar notificações.

5. Quando terminar, escolha Salvar alterações.

Tópicos relacionados

- Usar cargas úteis do Google Firebase Cloud Messaging v1 no Amazon SNS

Gerenciamento de endpoints do Firebase Cloud Messaging pelo Amazon SNS

Gerenciamento e manutenção de tokens de dispositivos

Você pode garantir a capacidade de entrega das notificações push do seu aplicativo móvel seguindo estas etapas:

1. Armazene todos os tokens do dispositivo, o endpoint ARNs correspondente do Amazon SNS e os carimbos de data/hora em seu servidor de aplicativos.
2. Remova todos os tokens obsoletos e exclua o endpoint correspondente do Amazon SNS ARNs.

Na inicialização inicial do seu aplicativo, você receberá um token de dispositivo (também conhecido como token de registro) para o dispositivo. Esse token de dispositivo é gerado pelo sistema operacional do dispositivo e está vinculado ao seu aplicativo FCM. Depois de receber esse token de dispositivo, você pode registrá-lo no Amazon SNS como um endpoint da plataforma. Recomendamos que você armazene o token do dispositivo, o ARN do endpoint da plataforma Amazon SNS e o carimbo de data/hora salvando-os em seu servidor de aplicativos ou em outro armazenamento persistente. Para configurar seu aplicativo FCM para recuperar e armazenar tokens de dispositivos, consulte Recuperar e armazenar tokens de registro na documentação do Firebase do Google.

É importante que você mantenha up-to-date os tokens. Os tokens de dispositivo do seu usuário podem ser alterados nas seguintes condições:

1. O aplicativo móvel é restaurado em um novo dispositivo.
2. O usuário desinstala ou atualiza o aplicativo.
3. O usuário limpa os dados do aplicativo.

Quando o token do seu dispositivo for alterado, recomendamos que você atualize o endpoint correspondente do Amazon SNS com o novo token. Isso permite que o Amazon SNS continue a comunicação com o dispositivo registrado. Você pode fazer isso implementando o seguinte pseudocódigo em seu aplicativo móvel. Ele descreve uma prática recomendada para criar e manter endpoints de plataforma habilitados. Essa abordagem pode ser executada sempre que os aplicativos móveis são iniciados ou como um trabalho agendado em segundo plano.

Pseudocódigo

Saiba como gerenciar e manter tokens de dispositivos usando o pseudocódigo do FCM.

```
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
```

Para saber mais sobre os requisitos de atualização de tokens, consulte Atualizar tokens regularmente na documentação do Firebase do Google.

Como detectar tokens inválidas

Quando uma mensagem é enviada para um endpoint do FCM v1 com um token de dispositivo inválido, o Amazon SNS receberá uma das seguintes exceções:

- `UNREGISTERED` (HTTP 404): quando o Amazon SNS receber essa exceção, você receberá um evento de falha na entrega com um `FailureType` de `InvalidPlatformToken` e um `FailureMessage` de Token de plataforma associado ao endpoint não é válido. O Amazon SNS desativará o endpoint da sua plataforma quando uma entrega falhar, com essa exceção.
- `INVALID_ARGUMENT` (HTTP 400): quando o Amazon SNS recebe essa exceção, significa que o token do dispositivo ou a carga útil da mensagem é inválida. Para obter mais informações, consulte ErrorCodea documentação do Firebase do Google.

Como `INVALID_ARGUMENT` pode ser devolvido em qualquer um desses casos, o Amazon SNS retornará um `FailureType` de `InvalidNotification` e um `FailureMessage` de Notificação de corpo é inválida. Quando você receber esse erro, verifique se sua carga útil está correta. Se estiver correto, verifique se o token do dispositivo está up-to-date. O Amazon SNS desativará o endpoint da sua plataforma quando uma entrega falhar, com essa exceção.

Outro caso em que você enfrentará um evento de falha na entrega do `InvalidPlatformToken` é quando o token do dispositivo registrado não pertence ao aplicativo que está tentando enviar essa mensagem. Nesse caso, o Google retornará um erro SENDER_ID_MISMATCH. O Amazon SNS desativará o endpoint da sua plataforma quando uma entrega falhar, com essa exceção.

Todos os códigos de erro observados recebidos da API FCM v1 estão disponíveis CloudWatch quando você configura o registro do status de entrega do seu aplicativo.

Para receber eventos de entrega para seu aplicativo, consulte Eventos do aplicativo disponíveis.

Remover tokens obsoletos

Os tokens são considerados obsoletos quando as entregas de mensagens ao dispositivo endpoint começam a falhar. O Amazon SNS define esses tokens obsoletos como endpoints desativados para seu aplicativo de plataforma. Quando você publica em um endpoint desativado, o Amazon SNS retornará `EventDeliveryFailure` um evento com `FailureType` de `EndpointDisabled` e um `FailureMessage` de Endpoint é desativado. Para receber eventos de entrega para seu aplicativo, consulte Eventos do aplicativo disponíveis.

Ao receber esse erro do Amazon SNS, você precisa remover ou atualizar o token obsoleto no aplicativo da sua plataforma.

Usar o Amazon SNS para notificações enviadas por push para dispositivos móveis

Esta seção descreve como enviar notificações por push móveis.

Publicar em um tópico

Também é possível usar o Amazon SNS para enviar mensagens para endpoints móveis inscritos em um tópico. O conceito é o mesmo que para inscrever outros tipos de endpoint, como Amazon SQS, HTTP/S, e-mail e SMS, em um tópico, conforme descrito em O que é o Amazon SNS?. A diferença é que o Amazon SNS se comunica por meio de serviços de notificação como o Apple Push Notification Service (APNS) e o Google Firebase Cloud Messaging (FCM). Por meio do serviço de notificações, os endpoints móveis inscritos recebem as notificações enviadas ao tópico.

Mensagens diretas para dispositivos móveis do Amazon SNS

Você pode enviar mensagens de notificação por push do Amazon SNS diretamente para um endpoint que representa uma aplicação em um dispositivo móvel.

Para enviar uma mensagem direta

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Push notifications (Notificações por push).
3. Na página de notificações push móveis, na seção Aplicativos da plataforma, escolha o nome do aplicativo, por exemplo***MyApp***.
4. Na ***MyApp*** página, na seção Endpoints, escolha um endpoint e escolha Publicar mensagem.

5. Na página Publish message to endpoint (Publicar mensagem no endpoint), insira a mensagem que aparecerá no aplicativo do dispositivo móvel e escolha Publish message (Publicar mensagem).

   O Amazon SNS envia a mensagem de notificação para o serviço de notificação da plataforma que, por sua vez, envia a mensagem para a aplicação.

Publicar notificações do Amazon SNS com cargas úteis específicas da plataforma

Você pode usar o AWS Management Console Amazon SNS APIs para enviar mensagens personalizadas com cargas específicas da plataforma para dispositivos móveis. Para obter informações sobre o uso do Amazon SNS APIs, consulte Ações da API de push para dispositivos móveis e o `SNSMobilePush.java` arquivo em. `snsmobilepush.zip`

Enviar mensagens formatadas em JSON

Ao enviar cargas específicas à plataforma, os dados devem estar formatados como strings de par de chave/valor JSON, com aspas em sequência de escape.

Os exemplos a seguir mostram uma mensagem personalizada para a plataforma do FCM.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

Enviar mensagens específicas à plataforma

Além de enviar dados personalizados como pares de chave/valor, é possível enviar pares de chave/valor específicos à plataforma.

O exemplo a seguir mostra a inclusão dos parâmetros `time_to_live` e `collapse_key` do FCM depois dos pares de chave/valor de dados personalizados no parâmetro `data` do FCM.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Para obter uma lista dos pares de chave/valor compatíveis em cada um dos serviços de notificação push compatíveis com o Amazon SNS, consulte o seguinte:

> **Important**
>
> O Amazon SNS agora oferece suporte à API HTTP v1 do Firebase Cloud Messaging (FCM) para enviar notificações push móveis para dispositivos Android.
> 26 de março de 2024: o Amazon SNS oferece suporte à API HTTP v1 do FCM para dispositivos Apple e destinos Webpush. Recomendamos que você migre suas aplicações móveis por push existentes para a API mais recente do FCM HTTP v1 até 1º de junho de 2024 para evitar interrupções nas aplicações.

- Referência da chave de carga útil na documentação APNs
- Protocolo HTTP do Firebase Cloud Messaging na documentação do FCM
- Send a Message (“Enviar uma mensagem”) na documentação do ADM

Enviar mensagens para um aplicativo em várias plataformas

Para enviar uma mensagem para um aplicativo instalado em dispositivos de várias plataformas, como o FCM e APNs, você deve primeiro inscrever os endpoints móveis em um tópico no Amazon SNS e depois publicar a mensagem no tópico.

O exemplo a seguir mostra uma mensagem para enviar aos endpoints móveis inscritos no FCM APNs e no ADM:

```
{
  "default": "This is the default message which must be present when publishing a
 message to a topic. The default message will only be used if a message is not present
 for
one of the notification platforms.",
  "APNS": "{\"aps\":{\"alert\": \"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"} }",
  "GCM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}",
  "ADM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}"
}
```

Envio de mensagens APNs como alerta ou notificações em segundo plano

O Amazon SNS pode enviar mensagens para APNs as `alert` ou `background` notificações (para obter mais informações, consulte Enviando atualizações em segundo plano para seu aplicativo na APNs documentação).

- Uma `alert` APNs notificação informa o usuário exibindo uma mensagem de alerta, reproduzindo um som ou adicionando um selo ao ícone do seu aplicativo.
- Uma `background` APNs notificação ativa ou instrui seu aplicativo a agir de acordo com o conteúdo da notificação, sem informar o usuário.

Especificando valores de APNs cabeçalho personalizados

Recomendamos especificar valores personalizados para o atributo de mensagem `AWS.SNS.MOBILE.APNS.PUSH_TYPE` reservada usando a ação AWS SDKs da API do Amazon `Publish` SNS ou o. AWS CLI O exemplo da CLI a seguir define `content-available` como 1 e `apns-push-type` como `background` para o tópico especificado.

```
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**
>
> Certifique-se de que a estrutura JSON seja válida. Adicione uma vírgula após cada par chave-valor, exceto o último.

Inferindo o cabeçalho do tipo APNs push a partir da carga

Se você não definir o `apns-push-type` APNs cabeçalho, o Amazon SNS define o cabeçalho como `alert` ou `background` dependendo da `content-available` chave no `aps` dicionário da sua configuração de carga em formato JSON APNs .

> **Note**
>
> O Amazon SNS é capaz de inferir somente os cabeçalhos `alert` ou `background`, embora o cabeçalho `apns-push-type` possa ser definido com outros valores.

- `apns-push-type` é definido como `alert`
  - Se o dicionário `aps` contiver `content-available` definida como `1` e uma ou mais chaves que acionem interações do usuário.
  - Se o dicionário `aps` contiver `content-available` definida como `0` ou se a chave `content-available` estiver ausente.
  - Se o valor da chave `content-available` não for um inteiro ou um booliano.
- `apns-push-type` é definido como `background`
  - Se o dicionário `aps` contiver somente a variável `content-available` definida como `1` e nenhuma outra chave que acione interações com o usuário.

> **Important**
>
> Se o Amazon SNS enviar um objeto de configuração bruto APNs como uma notificação somente em segundo plano, você deverá incluir `content-available` set to no dicionário. `1` `aps` Embora você possa incluir chaves personalizadas, o dicionário `aps` não deve conter chaves que acionem interações do usuário (por exemplo, alertas, distintivos ou sons).

Veja a seguir um exemplo de objeto de configuração bruto.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

Neste exemplo, o Amazon SNS define o `apns-push-type` APNs cabeçalho da mensagem como. `background` Quando o Amazon SNS detecta que o dicionário `apn` contém a chave `content-available` definida como `1` e não contém nenhuma outra chave que possa acionar interações do usuário, ele define o cabeçalho como `background`.

Usar cargas úteis do Google Firebase Cloud Messaging v1 no Amazon SNS

O Amazon SNS oferece suporte ao uso da API HTTP v1 do FCM para enviar notificações para destinos Android, iOS e Webpush. Este tópico fornece exemplos da estrutura de carga útil ao publicar notificações push móveis usando a CLI ou a API do Amazon SNS.

Você pode incluir os seguintes tipos de mensagem em sua carga útil ao enviar uma notificação do FCM:

- Mensagem de dados: uma mensagem de dados é gerenciada pelo seu aplicativo cliente e contém pares de valores-chave personalizados. Ao criar uma mensagem de dados, você deve incluir a chave `data` com um objeto JSON como valor e, em seguida, inserir seus pares de valores-chave personalizados.
- Mensagem de notificação ou mensagem de exibição: uma mensagem de notificação contém um conjunto predefinido de chaves gerenciadas pelo SDK do FCM. Essas chaves variam de acordo com o tipo de dispositivo para o qual você está entregando. Para obter mais informações sobre chaves de notificação específicas da plataforma, consulte a seguir:
  - Chaves de notificação do Android
  - Chaves de notificação do APNS
  - Chaves de notificação Webpush

Para obter mais informações sobre os tipos de mensagens do FCM, consulte Tipos de mensagens na documentação do Firebase do Google.

Usar a estrutura de carga útil do FCM v1 para enviar mensagens

Se você estiver criando um aplicativo FCM pela primeira vez ou quiser aproveitar os recursos do FCM v1, você pode optar por enviar uma carga útil no formato FCM v1. Para fazer isso, você deve incluir a chave `fcmV1Message` de nível superior. Para obter mais informações sobre como criar cargas do FCM v1, consulte Migrar do FCM legado APIs para HTTP v1 e Personalizar uma mensagem em várias plataformas na documentação do Firebase do Google.

Exemplo de carga útil do FCM v1 enviada para o Amazon SNS:

> **Note**
>
> O valor da chave `GCM` usado no exemplo a seguir deve ser codificado como uma string ao publicar uma notificação usando o Amazon SNS.

```
{
  "GCM": "{
    \"fcmV1Message\": {
      \"validate_only\": false,
      \"message\": {
        \"notification\": {
          \"title\": \"string\",
          \"body\": \"string\"
        },
        \"data\": {
          \"dataGen\": \"priority message\"
        },
        \"android\": {
          \"priority\": \"high\",
          \"notification\": {
            \"body_loc_args\": [\"string\"],
            \"title_loc_args\": [\"string\"],
            \"sound\": \"string\",
            \"title_loc_key\": \"string\",
            \"title\": \"string\",
            \"body\": \"string\",
            \"click_action\": \"clicky_clacky\",
            \"body_loc_key\": \"string\"
          },
          \"data\": {
            \"dataAndroid\": \"priority message\"
          },
          \"ttl\": \"10023.32s\"
        },
        \"apns\": {
          \"payload\": {
            \"aps\": {
              \"alert\": {
                \"subtitle\": \"string\",
                \"title-loc-args\": [\"string\"],
                \"title-loc-key\": \"string\",
```

```
              \"loc-args\": [\"string\"],
              \"loc-key\": \"string\",
              \"title\": \"string\",
              \"body\": \"string\"
            },
            \"category\": \"Click\",
            \"content-available\": 0,
            \"sound\": \"string\",
            \"badge\": 5
          }
        }
      },
      \"webpush\": {
        \"notification\": {
          \"badge\": \"5\",
          \"title\": \"string\",
          \"body\": \"string\"
        },
        \"data\": {
          \"dataWeb\": \"priority message\"
        }
      }
    }
  }"
}
```

Ao enviar uma carga útil JSON, não se esqueça de incluir o atributo `message-structure` em sua solicitação e configurá-lo como `json`.

Exemplo na CLI:

```
aws sns publish --topic $TOPIC_ARN --message '{"GCM": "{\"fcmV1Message\": {\"message\":
{\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"android\":{\"priority
\":\"high\",\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"data\":
{\"customAndroidDataKey\":\"custom key value\"},\"ttl\":\"0s\"},\"apns\":{\"payload
\":{\"aps\":{\"alert\":{\"title\":\"string\", \"body\":\"string\"},\"content-
available\":1,\"badge\":5}}},\"webpush\":{\"notification\":{\"badge\":\"URL\",\"body
\":\"Test\"},\"data\":{\"customWebpushDataKey\":\"priority message\"}},\"data\":
{\"customGeneralDataKey\":\"priority message\"}}}}", "default": "{\"notification\":
 {\"title\": \"test\"}"}' --region $REGION --message-structure json
```

Para obter mais informações sobre o envio de cargas úteis no formato FCM v1, consulte o seguinte na documentação do Firebase do Google:

- Migre do FCM antigo APIs para o HTTP v1
- Sobre as mensagens do FCM
- Recurso REST: projects.messages

Usar a estrutura de carga útil legada para enviar mensagens para a API FCM v1

Ao migrar para o FCM v1, você não precisa alterar a estrutura de carga útil que estava usando para suas credenciais legadas. O Amazon SNS transforma sua carga na nova estrutura de carga útil do FCM v1 e envia para o Google.

Formato da carga útil da mensagem de entrada:

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

Mensagem enviada ao Google:

```
{
  "message": {
    "token": "***",
    "notification": {
      "title": "string",
      "body": "string"
    },
    "android": {
      "priority": "high",
      "notification": {
        "body_loc_args": [
          "string"
        ],
        "title_loc_args": [
          "string"
        ],
```

```
        "color": "string",
        "sound": "string",
        "icon": "string",
        "tag": "string",
        "title_loc_key": "string",
        "title": "string",
        "body": "string",
        "click_action": "string",
        "channel_id": "string",
        "body_loc_key": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "apns": {
      "payload": {
        "aps": {
          "alert": {
            "title-loc-args": [
              "string"
            ],
            "title-loc-key": "string",
            "loc-args": [
              "string"
            ],
            "loc-key": "string",
            "title": "string",
            "body": "string"
          },
          "category": "string",
          "sound": "string"
        }
      }
    },
    "webpush": {
      "notification": {
        "icon": "string",
        "tag": "string",
        "body": "string",
        "title": "string"
      },
      "data": {
        "message": "priority message"
```

```
      }
    },
    "data": {
      "message": "priority message"
    }
  }
}
```

Riscos potenciais

- O mapeamento legado para v1 não é compatível com o Apple Push Notification Service (APNS) `headers` ou as chaves `fcm_options`. Se quiser usar esses campos, envie uma carga útil do FCM v1.
- Em alguns casos, os cabeçalhos das mensagens são exigidos pelo FCM v1 para enviar notificações silenciosas aos seus dispositivos. APNs Se você estiver enviando notificações silenciosas para seus APNs dispositivos, elas não funcionarão com a abordagem antiga. Em vez disso, recomendamos usar a carga útil do FCM v1 para evitar problemas inesperados. Para encontrar uma lista de APNs cabeçalhos e para que eles são usados, consulte Comunicação com APNs no Guia do Desenvolvedor da Apple.
- Se você estiver usando o atributo TTL Amazon SNS ao enviar sua notificação, ela só será atualizada no campo `android`. Se você quiser definir o atributo TTL APNS, use a carga útil do FCM v1.
- As chaves `android`, `apns` e `webpush` serão mapeadas e preenchidas com todas as chaves relevantes fornecidas. Por exemplo, se você fornecer o `title`, que é uma chave compartilhada entre as três plataformas, o mapeamento do FCM v1 preencherá as três plataformas com o título que você forneceu.
- Algumas chaves compartilhadas entre plataformas esperam diferentes tipos de valor. Por exemplo, a chave `badge` passada para `apns` espera um valor inteiro, enquanto a chave `badge` passada para `webpush` espera um valor String. Nos casos em que você fornecer a chave `badge`, o mapeamento do FCM v1 preencherá somente a chave para a qual você forneceu um valor válido.

Eventos de falha de entrega de FCM

A tabela a seguir fornece o tipo de falha do Amazon SNS que corresponde aos códigos de erro/status recebidos do Google para solicitações de notificação do FCM v1. Todos os códigos de erro observados recebidos da API FCM v1 estão disponíveis CloudWatch quando você configura o registro do status de entrega do seu aplicativo.

Código de erro/status do FCM	Tipo de falha do Amazon SNS	Mensagem de falha	Causa e mitigação
`UNREGISTERED`	`InvalidPlatformToken`	O token da plataforma associado ao endpoint não é válido.	O token do dispositivo anexado ao seu endpoint está obsoleto ou é inválido. O Amazon SNS desativou seu endpoint. Atualize o endpoint do Amazon SNS para o token de dispositivo mais novo.
`INVALID_ARGUMENT`	`InvalidNotification`	O corpo da notificação é inválido.	O token do dispositivo ou a carga útil da mensagem podem ser inválidos. Verifique se a carga útil da mensagem é válida. Se a carga útil da mensagem for válida, atualize o endpoint do Amazon SNS para o token de dispositivo mais novo.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	O token da plataforma associado ao endpoint não é válido.	O aplicativo da plataforma associado ao token do dispositivo não tem permissão para enviar para o token do dispositivo. Confira se está usando as credenciais do FCM corretas no aplicativo da

Código de erro/status do FCM	Tipo de falha do Amazon SNS	Mensagem de falha	Causa e mitigação
			plataforma do Amazon SNS.
UNAVAILABLE	DependencyUnavailable	A dependência não está disponível.	O FCM não pôde processar a solicitação a tempo. Todas as novas tentativas executadas pelo Amazon SNS falharam. Você pode armazenar essas mensagens em uma fila de mensagens não entregues (DLQ) e redirecioná-las posteriormente.
INTERNAL	UnexpectedFailure	Falha inesperada; entre em contato com a Amazon. Frase de falha [Erro interno].	O servidor FCM encontrou um erro ao tentar processar sua solicitação. Todas as novas tentativas executadas pelo Amazon SNS falharam. Você pode armazenar essas mensagens em uma fila de mensagens não entregues (DLQ) e redirecioná-las posteriormente.

Código de erro/status do FCM	Tipo de falha do Amazon SNS	Mensagem de falha	Causa e mitigação
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	As credenciais do aplicativo da plataforma não são válidas.	Uma mensagem direcionada a um dispositivo iOS ou a um dispositivo Webpush não pôde ser enviada. Verifique se suas credenciais de desenvolvimento e produção são válidas.
`QUOTA_EXCEEDED`	`Throttled`	Solicitação limitada por [gcm].	Uma cota de taxa de mensagem, cota de taxa de mensagem de dispositivo ou cota de taxa de mensagem de tópico foi excedida. Para obter informações sobre como resolver esse problema, consulte ErrorCode a documentação do Firebase do Google.

Código de erro/status do FCM	Tipo de falha do Amazon SNS	Mensagem de falha	Causa e mitigação
`PERMISSIO N_DENIED`	`InvalidNo tification`	O corpo da notificaç ão é inválido.	No caso de uma exceção `PERMISSIO N_DENIED` , o chamador (sua aplicação FCM) não tem permissão para executar a operação especificada na carga útil. Navegue até o console do FCM e verifique se suas credenciais têm as ações de API necessárias ativadas.

Atributos de aplicativo móvel do Amazon SNS

O Amazon Simple Notification Service (Amazon SNS) oferece suporte para o registro em log do status de entrega das mensagens de notificação por push. Depois de configurar os atributos do aplicativo, as entradas de registro serão enviadas ao CloudWatch Logs para mensagens enviadas do Amazon SNS para endpoints móveis. O registro de status de entrega de mensagens proporciona um melhor insight operacional, por exemplo:

- Saiba se uma mensagem de notificação por push foi entregue do Amazon SNS para o serviço de notificações por push.
- Identifique a resposta enviada do serviço de notificações por push para o Amazon SNS.
- Determinar o tempo de permanência da mensagem (o tempo entre o carimbo de data e hora da publicação e antes do envio para um serviço de notificações por push).

Para configurar os atributos do aplicativo para o status de entrega de mensagens, você pode usar os AWS Management Console kits de desenvolvimento de AWS software (SDKs) ou a API de consulta.

Configurando atributos de status de entrega de mensagens usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, aponte para Mobile (Dispositivos móveis), em seguida escolha Push notifications (Notificações por push).
3. Na seção Aplicativos da plataforma, escolha o aplicativo que contém os endpoints para os quais você deseja receber CloudWatch registros.
4. Escolha Application Actions (Ações do aplicativo) e, em seguida, selecione Delivery status (Status de entrega).
5. Na caixa de diálogo Delivery Status (Status da entrega), escolha Create IAM Roles (Criar funções do IAM).

   Você será redirecionado para o console do IAM.
6. Escolha Permitir para dar ao Amazon SNS acesso de gravação para usar CloudWatch os registros em seu nome.
7. Agora, de volta à caixa de diálogo Status de entrega, insira um número no campo Porcentagem de sucesso na amostra (0-100) para a porcentagem de mensagens bem-sucedidas enviadas para as quais você deseja receber CloudWatch registros.

   > **Note**
   >
   > Depois de configurar os atributos do aplicativo para o status de entrega de mensagens, todas as entregas de mensagens com falha geram CloudWatch registros.
8. Finalmente, escolha Save Configuration (Salvar configuração). Agora você poderá visualizar e analisar os CloudWatch registros que contêm o status de entrega da mensagem. Para obter mais informações sobre o uso CloudWatch, consulte a CloudWatchdocumentação.

Exemplos de registros de status CloudWatch de entrega de mensagens do Amazon SNS

Depois de configurar os atributos de status de entrega de mensagens para um endpoint do aplicativo, CloudWatch os registros serão gerados. Exemplos de logs, no formato JSON, são exibidos da seguinte forma:

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"
  }
}
```

Para obter uma lista de códigos de resposta do serviço de notificação por push, consulte Códigos de resposta da plataforma.

Configurando atributos de status de entrega de mensagens com o AWS SDKs

Eles AWS SDKsfornecem APIs em vários idiomas o uso de atributos de status de entrega de mensagens com o Amazon SNS.

O seguinte exemplo Java mostra como usar a API `SetPlatformApplicationAttributes` para configurar atributos de aplicativo para status de entrega de mensagens para as mensagens de notificação por push. Você pode usar os seguintes atributos para o status de entrega de mensagens: `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn` e `SuccessFeedbackSampleRate`. Os `FailureFeedbackRoleArn` atributos `SuccessFeedbackRoleArn` e são usados para dar ao Amazon SNS acesso de gravação para usar CloudWatch Logs em seu nome. O atributo `SuccessFeedbackSampleRate` é para especificar a porcentagem de taxa de amostra (0-100) de mensagens bem-sucedidas. Depois de configurar o `FailureFeedbackRoleArn` atributo, todas as entregas de mensagens com falha geram CloudWatch registros.

```
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);
```

Para obter mais informações sobre o SDK para Java, consulte Conceitos básicos do AWS SDK para Java.

Códigos de resposta da plataforma

A seguir, uma lista de links para os códigos de resposta do serviço de notificação por push:

Serviço de notificação por push	Códigos de resposta
Amazon Device Messaging (ADM)	Consulte Response Format (“Formato de resposta”) na documentação do ADM.

Serviço de notificação por push	Códigos de resposta
Serviço de notificação push da Apple (APNs)	Consulte a resposta HTTP/2 de APNs em Comunicação com APNs no Guia de programação de notificações locais e remotas.
Firebase Cloud Messaging (FCM)	Consulte Códigos de respostas de erros de mensagens downstream na documentação do Firebase Cloud Messaging.
Microsoft Push Notification Service for Windows Phone (MPNS)	Consulte Códigos de resposta do Serviço de notificação por push para Windows Phone 8 na documentação de desenvolvimento do Windows 8.
Windows Push Notification Services (WNS)	Consulte "Códigos de resposta" em Solicitação e cabeçalhos de resposta do Serviço de notificação por push (Aplicativos de tempo de execução do Windows) na documentação de desenvolvimento do Windows 8.

Notificações de eventos do aplicativo Amazon SNS para aplicativos móveis

O Amazon SNS fornece suporte para acionar as notificações quando determinados eventos da aplicação ocorrem. Em seguida, você pode executar algumas ações programáticas nesse evento. Seu aplicativo deve incluir suporte para um serviço de notificação push, como Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) e Windows Push Notification Services (WNS). Você define notificações de eventos do aplicativo usando o console do Amazon SNS ou o. AWS CLI AWS SDKs

Eventos do aplicativo disponíveis

As notificações de eventos do aplicativo rastreiam quando endpoints de plataforma individuais são criados, excluídos e atualizados, bem como as falhas de entrega. A seguir estão os nomes de atributo para os eventos do aplicativo.

Nome do atributo	Trigger para a notificação
`EventEndpointCreated`	Um novo endpoint de plataforma é adicionado ao seu aplicativo.
`EventEndpointDeleted`	Qualquer endpoint de plataforma associado ao seu aplicativo é excluído.
`EventEndpointUpdated`	Qualquer um dos atributos dos endpoints de plataforma associados ao seu aplicativo é alterado.
`EventDeliveryFailure`	Uma entrega para qualquer um dos endpoints de plataforma associados ao seu aplicativo encontra uma falha permanente. **Note** Para rastrear falhas de entrega no aplicativo da plataforma, inscreva-se em eventos de status de entrega de mensagens para o aplicativo. Para obter mais informações, consulte Usar atributos de aplicativo do Amazon SNS para obter o status de entrega de mensagens.

Você pode associar qualquer atributo a um aplicativo que poderá então receber essas notificações de eventos.

Enviar notificações por push para dispositivos móveis

Para enviar notificações de eventos do aplicativo, você especifica um tópico para receber notificações de cada tipo de evento. Como o Amazon SNS envia notificações, o tópico pode direcioná-las para endpoints que executarão ações programáticas.

Important

Aplicativos de grande volume criarão um grande número de notificações de eventos do aplicativo (por exemplo, dezenas de milhares), que sobrecarregarão os endpoints destinados ao uso humano, como endereços de e-mail, números de telefone e aplicativos móveis.

Considere as seguintes diretrizes ao enviar as notificações de eventos do aplicativo para um tópico:

- Cada tópico que recebe notificações deve conter somente assinaturas de endpoints programáticos, como endpoints HTTP ou HTTPS, filas ou funções do Amazon SQS. AWS Lambda
- Para reduzir a quantidade de processamento que é acionada pelas notificações, limite as inscrições de cada tópico em um pequeno número (por exemplo, cinco ou menos).

Você pode enviar notificações de eventos do aplicativo usando o console do Amazon SNS, o AWS Command Line Interface (AWS CLI) ou o. AWS SDKs

AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Mobile (Dispositivos móveis), Push notifications (Notificações por push).
3. Na página Notificações por push para dispositivos móveis, na seção Aplicativos de plataforma, selecione um aplicativo e escolha Editar.
4. Expanda a seção Notificações de eventos.
5. Escolha Ações, Configurar eventos.
6. Insira ARNs os quatro tópicos a serem usados nos seguintes eventos:
   - Endpoint criado
   - Endpoint excluído
   - Endpoint atualizado
   - Falha na entrega
7. Escolha Salvar alterações.

AWS CLI

Execute o comando set-platform-application-attributes.

O exemplo a seguir define o mesmo tópico do Amazon SNS para todos os quatro eventos da aplicação:

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDKs

Defina notificações de eventos do aplicativo enviando uma `SetPlatformApplicationAttributes` solicitação com a API do Amazon SNS usando AWS um SDK.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, incluindo ajuda para começar e informações sobre versões anteriores, consulteUsando o Amazon SNS com um SDK AWS.

Ações da API de push para dispositivos móveis

Para usar o push móvel do Amazon SNS APIs, você deve primeiro atender aos pré-requisitos do serviço de notificação push, como o Apple Push Notification Service (APNs) e o Firebase Cloud Messaging (FCM). Para obter mais informações sobre os pré-requisitos, consulte Pré-requisitos para notificações ao usuário do Amazon SNS.

Para enviar uma mensagem de notificação push para um aplicativo e dispositivo móvel usando o APIs, você deve primeiro usar a `CreatePlatformApplication` ação, que retorna um `PlatformApplicationArn` atributo. O atributo `PlatformApplicationArn` é, então, usado por `CreatePlatformEndpoint`, que retorna um atributo `EndpointArn`. Em seguida, você pode usar o atributo `EndpointArn` com a ação `Publish` para enviar uma mensagem de notificação para um dispositivo e aplicativo móvel, ou você pode usar o atributo `EndpointArn` com a ação `Subscribe` de inscrição em um tópico. Para obter mais informações, consulte Configurar notificações de push com o Amazon SNS.

Os push móveis do Amazon SNS APIs são os seguintes:

`CreatePlatformApplication`

Cria um objeto de aplicativo de plataforma para um dos serviços de notificação push compatíveis, como APNs o FCM, no qual dispositivos e aplicativos móveis podem se registrar. Retorna um atributo `PlatformApplicationArn`, que é usado pela ação `CreatePlatformEndpoint`.

`CreatePlatformEndpoint`

Cria um endpoint para um dispositivo e aplicativo móvel em um dos serviços de notificação por push compatíveis. O `CreatePlatformEndpoint` usa o atributo `PlatformApplicationArn` retornado da ação `CreatePlatformApplication`. O atributo `EndpointArn`, que é retornado ao usar `CreatePlatformEndpoint`, é, então, usado com a ação `Publish` para enviar uma mensagem de notificação para um dispositivo e aplicativo móvel.

`CreateTopic`

Cria um tópico no qual as mensagens podem ser publicadas.

`DeleteEndpoint`

Exclui o endpoint para um dispositivo e aplicativo móvel em um dos serviços de notificações por push compatíveis.

`DeletePlatformApplication`

Exclui um objeto de aplicativo de plataforma.

`DeleteTopic`

Exclui um tópico e todas as suas inscrições.

`GetEndpointAttributes`

Recupera os atributos do endpoint para um dispositivo e aplicativo móvel.

`GetPlatformApplicationAttributes`

Recupera os atributos da plataforma de objeto de aplicativo.

`ListEndpointsByPlatformApplication`

Lista os endpoints e atributos de endpoint para dispositivos e aplicativos móveis em um serviço de notificações por push compatível.

`ListPlatformApplications`

Lista os objetos do aplicativo da plataforma para os serviços de notificações por push compatíveis.

`Publish`

Envia uma mensagem de notificação para todos os endpoints inscritos em um tópico.

`SetEndpointAttributes`

Define os atributos para um endpoint para um dispositivo e aplicativo móvel.

`SetPlatformApplicationAttributes`

Define os atributos do objeto de aplicativo de plataforma.

`Subscribe`

Prepara para assinar um endpoint enviando ao endpoint uma mensagem de confirmação. Para realmente criar uma assinatura, o proprietário do endpoint deve ConfirmSubscription executar a ação com o token a partir da mensagem de confirmação.

`Unsubscribe`

Exclui a inscrição.

Erros comuns da API push móvel do Amazon SNS

Os erros retornados pelo Amazon SNS APIs para push móvel estão listados na tabela a seguir. Para obter mais informações sobre o Amazon SNS APIs para push móvel, consulte. Ações da API de push para dispositivos móveis

Erro	Descrição	Código de status HTTPS	Ação API
Nome do aplicativo é uma string nula	O nome do aplicativo necessário está definido como nulo.	400	`CreatePlatformApplication`
Nome da plataforma é uma string nula	O nome da plataforma necessária está definido como nulo.	400	`CreatePlatformApplication`
Nome da plataforma é inválido	Um out-of-range valor ou valor inválido	400	`CreatePlatformApplication`

Erro	Descrição	Código de status HTTPS	Ação API
	foi fornecido para o nome da plataforma.		
APNs — O principal não é um certificado válido	Um certificado inválido foi fornecido para o APNs principal, que é o certificado SSL. Para obter mais informações, consulte CreatePlatformApplicationa Referência da API do Amazon Simple Notification Service.	400	`CreatePlatformApplication`
APNs — Principal é um certificado válido, mas não em um formato.pem	Um certificado válido que não está no formato.pem foi fornecido para o APNs principal, que é o certificado SSL.	400	`CreatePlatformApplication`
APNs — O principal é um certificado expirado	Um certificado expirado foi fornecido para o APNs principal, que é o certificado SSL.	400	`CreatePlatformApplication`
APNs — O principal não é um certificado emitido pela Apple	Um certificado não emitido pela Apple foi fornecido para o principal do APNs, que é o certificado SSL.	400	`CreatePlatformApplication`

Erro	Descrição	Código de status HTTPS	Ação API
APNs — O principal não é fornecido	O APNs principal, que é o certificado SSL, não foi fornecido.	400	`CreatePlatformApplication`
APNs — A credencial não é fornecida	A APNs credencial, que é a chave privada, não foi fornecida. Para obter mais informações, consulte CreatePlatformApplication a Referência da API do Amazon Simple Notification Service.	400	`CreatePlatformApplication`
APNs — As credenciais não estão em um formato.pem válido	A APNs credencial, que é a chave privada, não está em um formato.pem válido.	400	`CreatePlatformApplication`
FCM — o servidor não APIKey é fornecido	As credenciais do FCM, que são a chave de API, não foram fornecidas. Para obter mais informações, consulte CreatePlatformApplication a Referência da API do Amazon Simple Notification Service.	400	`CreatePlatformApplication`

Erro	Descrição	Código de status HTTPS	Ação API
FCM — o servidor APIKey está vazio	As credenciais do FCM, que são a chave de API, estão vazias.	400	`CreatePlatformApplication`
FCM — o servidor APIKey é uma string nula	As credenciais do FCM, que são a chave de API, são nulas.	400	`CreatePlatformApplication`
FCM — o servidor APIKey é inválido	As credenciais do FCM, que são a chave de API, são inválidas.	400	`CreatePlatformApplication`
ADM: clientsecret não foi fornecido	O segredo do cliente necessário não foi fornecido.	400	`CreatePlatformApplication`
ADM: clientsecret é uma string nula	A string necessária para o segredo do cliente é nula.	400	`CreatePlatformApplication`
ADM: client_secret é uma string vazia	A string necessária para o segredo do cliente está vazia.	400	`CreatePlatformApplication`
ADM: client_secret não é válido	A string necessária para o segredo do cliente não é válida.	400	`CreatePlatformApplication`
ADM: client_id é uma string vazia	A string necessária para o ID do cliente está vazia.	400	`CreatePlatformApplication`

Erro	Descrição	Código de status HTTPS	Ação API
ADM: clientId não foi fornecido	A string necessária para o ID do cliente não foi fornecida.	400	`CreatePlatformApplication`
ADM: clientid é uma string nula	A string necessária para o ID do cliente é nula.	400	`CreatePlatformApplication`
ADM: client_id não é válido	A string necessária para o ID do cliente não é válida.	400	`CreatePlatformApplication`
EventEndpointCreated tem formato ARN inválido	EventEndpointCreated tem formato ARN inválido.	400	`CreatePlatformApplication`
EventEndpointDeleted tem formato ARN inválido	EventEndpointDeleted tem formato ARN inválido.	400	`CreatePlatformApplication`
EventEndpointUpdated tem formato ARN inválido	EventEndpointUpdated tem formato ARN inválido.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure tem formato ARN inválido	EventDeliveryAttemptFailure tem formato ARN inválido.	400	`CreatePlatformApplication`
EventDeliveryFailure tem formato ARN inválido	EventDeliveryFailure tem formato ARN inválido.	400	`CreatePlatformApplication`
EventEndpointCreated não é um tópico existente	EventEndpointCreated não é um tópico existente.	400	`CreatePlatformApplication`

Erro	Descrição	Código de status HTTPS	Ação API
EventEndpointDeleted não é um tópico existente	EventEndpointDeleted não é um tópico existente.	400	`CreatePlatformApplication`
EventEndpointUpdated não é um tópico existente	EventEndpointUpdated não é um tópico existente.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure não é um tópico existente	EventDeliveryAttemptFailure não é um tópico existente.	400	`CreatePlatformApplication`
EventDeliveryFailure não é um tópico existente	EventDeliveryFailure não é um tópico existente.	400	`CreatePlatformApplication`
O ARN da plataforma é inválido	O ARN da plataforma é inválido.	400	`SetPlatformAttributes`
O ARN da plataforma é válido, mas não pertence ao usuário	O ARN da plataforma é válido, mas não pertence ao usuário.	400	`SetPlatformAttributes`
APNs — O principal não é um certificado válido	Um certificado inválido foi fornecido para o APNs principal, que é o certificado SSL. Para obter mais informações, consulte CreatePlatformApplicationa Referência da API do Amazon Simple Notification Service.	400	`SetPlatformAttributes`

Erro	Descrição	Código de status HTTPS	Ação API
APNs — Principal é um certificado válido, mas não em um formato.pem	Um certificado válido que não está no formato.pem foi fornecido para o APNs principal, que é o certificado SSL.	400	`SetPlatformAttributes`
APNs — O principal é um certificado expirado	Um certificado expirado foi fornecido para o APNs principal, que é o certificado SSL.	400	`SetPlatformAttributes`
APNs — O principal não é um certificado emitido pela Apple	Um certificado não emitido pela Apple foi fornecido para o principal do APNs, que é o certificado SSL.	400	`SetPlatformAttributes`
APNs — O principal não é fornecido	O APNs principal, que é o certificado SSL, não foi fornecido.	400	`SetPlatformAttributes`
APNs — A credencial não é fornecida	A APNs credencial, que é a chave privada, não foi fornecida. Para obter mais informações, consulte CreatePlatformApplicationa Referência da API do Amazon Simple Notification Service.	400	`SetPlatformAttributes`

Erro	Descrição	Código de status HTTPS	Ação API
APNs — As credencia is não estão em um formato.pem válido	A APNs credencia l, que é a chave privada, não está em um formato.pem válido.	400	`SetPlatfo rmAttributes`
FCM — o servidor não APIKey é fornecido	As credenciais do FCM, que são a chave de API, não foram fornecidas. Para obter mais informações, consulte CreatePlatformAppl icationa Referência da API do Amazon Simple Notification Service.	400	`SetPlatfo rmAttributes`
FCM — o servidor APIKey é uma string nula	As credenciais do FCM, que são a chave de API, são nulas.	400	`SetPlatfo rmAttributes`
ADM: clientId não foi fornecido	A string necessária para o ID do cliente não foi fornecida.	400	`SetPlatfo rmAttributes`
ADM: clientid é uma string nula	A string necessária para o ID do cliente é nula.	400	`SetPlatfo rmAttributes`
ADM: clientsecret não foi fornecido	O segredo do cliente necessário não foi fornecido.	400	`SetPlatfo rmAttributes`

Erro	Descrição	Código de status HTTPS	Ação API
ADM: clientsecret é uma string nula	A string necessária para o segredo do cliente é nula.	400	`SetPlatformAttributes`
EventEndpointUpdated tem formato ARN inválido	EventEndpointUpdated tem formato ARN inválido.	400	`SetPlatformAttributes`
EventEndpointDeleted tem formato ARN inválido	EventEndpointDeleted tem formato ARN inválido.	400	`SetPlatformAttributes`
EventEndpointUpdated tem formato ARN inválido	EventEndpointUpdated tem formato ARN inválido.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure tem formato ARN inválido	EventDeliveryAttemptFailure tem formato ARN inválido.	400	`SetPlatformAttributes`
EventDeliveryFailure tem formato ARN inválido	EventDeliveryFailure tem formato ARN inválido.	400	`SetPlatformAttributes`
EventEndpointCreated não é um tópico existente	EventEndpointCreated não é um tópico existente.	400	`SetPlatformAttributes`
EventEndpointDeleted não é um tópico existente	EventEndpointDeleted não é um tópico existente.	400	`SetPlatformAttributes`
EventEndpointUpdated não é um tópico existente	EventEndpointUpdated não é um tópico existente.	400	`SetPlatformAttributes`

Erro	Descrição	Código de status HTTPS	Ação API
EventDeliveryAttemptFailure não é um tópico existente	EventDeliveryAttemptFailure não é um tópico existente.	400	`SetPlatformAttributes`
EventDeliveryFailure não é um tópico existente	EventDeliveryFailure não é um tópico existente.	400	`SetPlatformAttributes`
O ARN da plataforma é inválido	O ARN da plataforma é inválido.	400	`GetPlatformApplicationAttributes`
O ARN da plataforma é válido, mas não pertence ao usuário	O ARN da plataforma é válido, mas não pertence ao usuário.	403	`GetPlatformApplicationAttributes`
O token especificado é inválido	O token especificado é inválido.	400	`ListPlatformApplications`
O ARN da plataforma é inválido	O ARN da plataforma é inválido.	400	`ListEndpointsByPlatformApplication`
O ARN da plataforma é válido, mas não pertence ao usuário	O ARN da plataforma é válido, mas não pertence ao usuário.	404	`ListEndpointsByPlatformApplication`
O token especificado é inválido	O token especificado é inválido.	400	`ListEndpointsByPlatformApplication`

Erro	Descrição	Código de status HTTPS	Ação API
O ARN da plataforma é inválido	O ARN da plataforma é inválido.	400	`DeletePlatformApplication`
O ARN da plataforma é válido, mas não pertence ao usuário	O ARN da plataforma é válido, mas não pertence ao usuário.	403	`DeletePlatformApplication`
O ARN da plataforma é inválido	O ARN da plataforma é inválido.	400	`CreatePlatformEndpoint`
O ARN da plataforma é válido, mas não pertence ao usuário	O ARN da plataforma é válido, mas não pertence ao usuário.	404	`CreatePlatformEndpoint`
O token não foi especificado	O token não foi especificado.	400	`CreatePlatformEndpoint`
O token não é do tamanho certo	O token não tem o tamanho certo.	400	`CreatePlatformEndpoint`
Os dados do usuário do cliente são muito grandes	Os dados do usuário do cliente não podem ter mais de 2048 bytes em codificação UTF-8.	400	`CreatePlatformEndpoint`
O ARN do endpoint é inválido	O ARN do endpoint é inválido.	400	`DeleteEndpoint`
O ARN do endpoint é válido, mas não pertence ao usuário	O ARN do endpoint é válido, mas não pertence ao usuário.	403	`DeleteEndpoint`
O ARN do endpoint é inválido	O ARN do endpoint é inválido.	400	`SetEndpointAttributes`

Erro	Descrição	Código de status HTTPS	Ação API
O ARN do endpoint é válido, mas não pertence ao usuário	O ARN do endpoint é válido, mas não pertence ao usuário.	403	`SetEndpointAttributes`
O token não foi especificado	O token não foi especificado.	400	`SetEndpointAttributes`
O token não é do tamanho certo	O token não tem o tamanho certo.	400	`SetEndpointAttributes`
Os dados do usuário do cliente são muito grandes	Os dados do usuário do cliente não podem ter mais de 2048 bytes em codificação UTF-8.	400	`SetEndpointAttributes`
O ARN do endpoint é inválido	O ARN do endpoint é inválido.	400	`GetEndpointAttributes`
O ARN do endpoint é válido, mas não pertence ao usuário	O ARN do endpoint é válido, mas não pertence ao usuário.	403	`GetEndpointAttributes`
O ARN de destino é inválido	O ARN de destino é inválido.	400	`Publish`
O ARN de destino é válido, mas não pertence ao usuário	O ARN de destino é válido, mas não pertence ao usuário.	403	`Publish`
O formato da mensagem é inválido	O formato da mensagem é inválido.	400	`Publish`

Erro	Descrição	Código de status HTTPS	Ação API
O tamanho da mensagem é maior do que o comportado pelo protocolo/serviço final	O tamanho da mensagem é maior do que o comportado pelo protocolo/serviço final.	400	`Publish`

Usar atributo de mensagem Time to Live do Amazon SNS para notificações por push para dispositivos móveis

O Amazon Simple Notification Service (Amazon SNS) oferece suporte para configurar um atributo de mensagem Time to Live (TTL) para mensagens de notificação por push para dispositivos móveis. Isso é uma adição ao recurso atual de configuração do TTL dentro do corpo da mensagem do Amazon SNS para os serviços de notificação por push para dispositivos móveis que oferecem suporte a isso, como o Amazon Device Messaging (ADM) e o Firebase Cloud Messaging (FCM) quando enviar para Android.

O atributo de mensagem do TTL é usado para especificar os metadados de expiração sobre uma mensagem. Isso permite que você especifique quanto tempo o serviço de notificação push, como o Apple Push Notification Service (APNs) ou o FCM, tem para entregar a mensagem ao endpoint. Se por algum motivo (por exemplo, se o dispositivo móvel foi desativado) a mensagem não for entregue no material do TTL especificado, ela será descartada e não serão feitas mais tentativas de entrega. Para especificar o TTL nos atributos da mensagem, você pode usar os AWS Management Console kits de desenvolvimento de AWS software (SDKs) ou a API de consulta.

Atributos da mensagem do TTL para serviços de notificação por push

Veja a seguir uma lista dos atributos de mensagem TTL para serviços de notificação push que você pode usar para definir ao usar a API AWS SDKs ou query:

Serviço de notificação por push	Atributo de mensagem do TTL
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Serviço de notificação push da Apple (APNs)	`AWS.SNS.MOBILE.APNS.TTL`

Serviço de notificação por push	Atributo de mensagem do TTL
Sandbox do serviço de notificação push da Apple (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCM ao enviar para o Android)	`AWS.SNS.MOBILE.FCM.TTL`
Windows Push Notification Services (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Cada um dos serviços de notificação por push processa o TTL de maneira diferente. Com o Amazon SNS você tem uma visão abstrata do TTL em todos os serviços de notificação por push, o que facilita especificar o TTL. Ao usar o AWS Management Console para especificar o TTL (em segundos), você só precisa inserir o valor do TTL uma vez e o Amazon SNS calculará o TTL para cada um dos serviços de notificação push selecionados ao publicar a mensagem.

O TTL é relativo ao momento da publicação. Antes de enviar uma mensagem de notificação por push para um serviço de notificações por push específico, o Amazon SNS calcula o tempo de permanência (o tempo entre o carimbo de data e hora da publicação e antes do envio para um serviço de notificações por push) para as notificações por push e passa o TTL restante para o serviço de notificações por push específico. Se o TTL for menor do que o tempo de permanência, o Amazon SNS não tentará publicar.

Se você especificar um TTL para uma mensagem de notificação por push, o valor do TTL deverá ser um número inteiro positivo, a menos que o valor de `0` tenha um significado específico para o serviço de notificação por push, como com APNs e FCM (ao enviar para Android). Se o valor do TTL for definido como `0` e o serviço de notificação por push não tiver um significado específico para `0`, o Amazon SNS descartará a mensagem. Para obter mais informações sobre o parâmetro TTL definido como `0` quando usado APNs, consulte a Tabela A-3 Identificadores de itens para notificações remotas na documentação da API Binary Provider.

Ordem de precedência para determinar o TTL

A precedência que o Amazon SNS usa para determinar o TTL para uma mensagem de notificação por push é baseada na seguinte ordem, em que o número mais baixo tem prioridade mais alta:

1. TTL do atributo de mensagem

2. TTL do corpo da mensagem
3. TTL do serviço de notificação por push padrão (varia por serviço)
4. TTL padrão do Amazon SNS (4 semanas)

Se você definir valores diferentes de TTL (um em atributos de mensagem e outro no corpo da mensagem) para a mesma mensagem, o Amazon SNS modificará o TTL no corpo da mensagem de acordo com o TTL especificado no atributo de mensagem.

Especificando TTL usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Mobile (Dispositivos móveis), Push notifications (Notificações por push).
3. Na página Mobile push notifications (Notificações por push para dispositivos móveis), na seção Platform applications (Aplicativos de plataforma), selecione um aplicativo.
4. Na ***MyApplication*** página, na seção Endpoints, escolha um endpoint do aplicativo e escolha Publicar mensagem.
5. Na seção Message details Detalhes da mensagem), insira o TTL (o número de segundos que o serviço de notificação por push tem para entregar a mensagem ao endpoint).
6. Selecione Publish message (Publicar mensagem).

Regiões suportadas pelo aplicativo móvel Amazon SNS

No momento, as aplicações móveis podem ser criadas apenas nas seguintes regiões:

- Leste dos EUA (Ohio)
- Leste dos EUA (Norte da Virgínia)
- Oeste dos EUA (Norte da Califórnia)
- Oeste dos EUA (Oregon)
- África (Cidade do Cabo)
- Ásia-Pacífico (Hong Kong)
- Ásia-Pacífico (Jacarta)
- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Osaka)

- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Europa (Frankfurt)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milão)
- Europa (Paris)
- Europa (Estocolmo)
- Oriente Médio (Bahrein)
- Oriente Médio (Emirados Árabes Unidos)
- América do Sul (São Paulo)
- AWS GovCloud (Oeste dos EUA)

Práticas recomendadas para gerenciar notificações por push para dispositivos móveis do Amazon SNS

Esta seção descreve práticas recomendadas que podem ajudar você a melhorar seu envolvimento com os clientes.

Gerenciamento de endpoints

Problemas de entrega podem ocorrer em situações em que os tokens de dispositivo mudam devido à ação de um usuário no dispositivo (por exemplo, um aplicativo é reinstalado no dispositivo) ou a atualizações de certificado que afetam os dispositivos que utilizam determinada versão do iOS. É uma prática recomendada pela Apple registrar-se APNs sempre que seu aplicativo for iniciado.

Como o token do dispositivo não muda sempre que um aplicativo é aberto por um usuário, é possível usar a API `CreatePlatformEndpoint` idempotente. No entanto, isso pode introduzir duplicações do mesmo dispositivo nos casos em que o token em si é inválido, ou se o endpoint for válido, mas desabilitado (por exemplo, uma incompatibilidade entre ambientes de produção e sandbox).

Um mecanismo de gerenciamento de tokens de dispositivo, como o contido no pseudocódigo pode ser usado.

Para obter informações sobre como gerenciar e manter os tokens do dispositivo FCM v1, consulte Gerenciamento de endpoints do Firebase Cloud Messaging pelo Amazon SNS.

Registro em log do status de entrega

Para monitorar o status de entrega de notificações por push, recomendamos que você habilite o registro de status de entrega para a aplicação da plataforma Amazon SNS. Isso ajuda você a solucionar problemas de falhas de entrega porque os logs contêm códigos de resposta do provedor retornados do serviço de plataforma push. Para obter detalhes sobre como habilitar o registro de status de entrega, consulte Como faço para acessar os logs de entrega de tópicos do Amazon SNS para notificações por push?.

Notificações de eventos

Para gerenciar endpoints de forma orientada por eventos, você pode usar a funcionalidade notificações de eventos. Isso permite que o tópico configurado do Amazon SNS faça fanout de eventos para os assinantes, como uma função do Lambda, para eventos de aplicações de plataforma de criação de endpoint, exclusão, atualizações e falhas de entrega.

Configuração e gerenciamento de assinaturas de e-mail do Amazon SNS

Você pode inscrever um endereço de e-mail em um tópico do Amazon SNS usando o AWS Management Console, AWS SDK para Java, ou. AWS SDK para .NET

Observações

- A personalização do corpo da mensagem de e-mail não é suportada. O recurso de entrega de e-mail destina-se a fornecer alertas internos do sistema, não mensagens de marketing.
- A assinatura direta de endpoints de e-mail só é compatível com tópicos-padrão.
- O throughput da entrega de e-mail apresenta controle de utilização. Para obter mais informações, consulte as Cotas do Amazon SNS.

> **Important**
>
> Para evitar que os destinatários da lista de e-mails cancelem a assinatura de todos os destinatários dos e-mails de tópicos do Amazon SNS, consulte Configurar uma assinatura de e-mail que requer autenticação para cancelar a assinatura no AWS Support.

Inscrever um endereço de e-mail em um tópico do Amazon SNS usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. No painel de navegação à esquerda, selecione Assinaturas.
3. Na página Assinaturas, escolha Criar assinatura.
4. Na página Criar assinatura, na seção Detalhes, faça o seguinte:
   a. Em Topic ARN (ARN do tópico), escolha o nome do recurso da Amazon (ARN) de um tópico.
   b. Em Protocolo, escolha E-mail.
   c. Para Endpoint, insira o endereço de e-mail.
   d. (Opcional) Para configurar uma política de filtros, expanda a seção Subscription filter policy (Política de filtro de assinatura). Para obter mais informações, consulte Políticas de filtro de assinatura do Amazon SNS.
   e. (Opcional) Para habilitar a filtragem baseada em carga útil, configure `Filter Policy Scope` como `MessageBody`. Para obter mais informações, consulte Escopo de política de filtro de assinaturas do Amazon SNS.
   f. (Opcional) Para configurar uma fila de mensagens não entregues para a assinatura, expanda a seção Redrive policy (dead-letter queue) (Política de redirecionamento (fila de mensagens não entregues)). Para obter mais informações, consulte Filas de mensagens não entregues do Amazon SNS.
   g. Selecione Criar assinatura.

      O console cria a assinatura e abre a página Details (Detalhes) da assinatura.

Você deve confirmar a assinatura para que o endereço de e-mail comece a receber mensagens.

Para confirmar uma assinatura

1. Verifique sua caixa de entrada de e-mail e escolha Confirm subscription (Confirmar a assinatura) no e-mail do Amazon SNS.
2. O Amazon SNS abre seu navegador da Web e exibe uma confirmação de assinatura com seu ID de assinatura.

Inscrever um endereço de e-mail em um tópico do Amazon SNS usando um SDK AWS

Para usar um AWS SDK, você deve configurá-lo com suas credenciais. Para obter mais informações, consulte Os arquivos compartilhados de configuração e credenciais no Guia de referência de ferramentas AWS SDKs e ferramentas.

Os exemplos de código a seguir mostram como usar o `Subscribe`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

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

Inscreva uma fila em um tópico com filtros opcionais.

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

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

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

Inscrever uma aplicação móvel em um tópico.

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

Inscrever uma função do Lambda em um tópico.

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

Assinar uma fila do SQS em um tópico.

```
        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;
            }
```

Assinar com um filtro em um tópico.

```
        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;
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para inscrever-se em um tópico

O comando `subscribe` a seguir inscreve um endereço de e-mail no tópico especificado.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Saída:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Para obter detalhes da API, consulte Subscribe na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscreva uma fila em um tópico com filtros opcionais.

```
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
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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);
        }
    }
}
```

Inscrever um endpoint HTTP em um tópico.

```
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);
        }
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
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 "";
    }
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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'
  // }
};
```

Inscrever uma aplicação móvel em um tópico.

```
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;
};
```

Inscrever uma função do Lambda em um tópico.

```
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;
};
```

Assinar uma fila do SQS em um tópico.

```
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;
};
```

Assinar com um filtro em um tópico.

```
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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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()
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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());
}
```

Inscrever um endpoint HTTP em um tópico.

```
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());
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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
```

- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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(())
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
    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.
```

- Para obter detalhes da API, consulte Inscrever-se na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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.")
```

Inscreva um número de telefone em um tópico para receber notificações por SMS.

```
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.")
```

- Para obter detalhes da API, consulte Inscrever-se na referência da API do AWS SDK for Swift.

Exemplos de código para o Amazon SNS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon SNS com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos

Olá, Amazon SNS

Os exemplos de código a seguir mostram como começar a usar o Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o CMake arquivo CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)
```

```
# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_sns.cpp.

```
#include <aws/core/Aws.h>
```

```
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
```

```
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
  }
 }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicialize um cliente SNS e liste tópicos em sua conta.

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

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Kotlin.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
```

```
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O programa principal do Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"
```

```
    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
```

```
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa referência da API AWS SDK for Swift.

Exemplos de código

- Exemplos básicos para o uso do Amazon SNS AWS SDKs
  - Olá, Amazon SNS
  - Ações para o Amazon SNS usando AWS SDKs
    - Use CheckIfPhoneNumberIsOptedOut com um AWS SDK ou CLI
    - Use ConfirmSubscription com um AWS SDK ou CLI
    - Use CreateTopic com um AWS SDK ou CLI
    - Use DeleteTopic com um AWS SDK ou CLI
    - Use GetSMSAttributes com um AWS SDK ou CLI
    - Use GetTopicAttributes com um AWS SDK ou CLI
    - Use ListPhoneNumbersOptedOut com um AWS SDK ou CLI
    - Use ListSubscriptions com um AWS SDK ou CLI
    - Use ListTopics com um AWS SDK ou CLI
    - Use Publish com um AWS SDK ou CLI
    - Use SetSMSAttributes com um AWS SDK ou CLI

Exemplos básicos para o uso do Amazon SNS AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Notification Service com AWS SDKs.

Exemplos

- Use CheckIfPhoneNumberIsOptedOut com um AWS SDK ou CLI
- Use ConfirmSubscription com um AWS SDK ou CLI
- Use CreateTopic com um AWS SDK ou CLI
- Use DeleteTopic com um AWS SDK ou CLI
- Use GetSMSAttributes com um AWS SDK ou CLI
- Use GetTopicAttributes com um AWS SDK ou CLI
- Use ListPhoneNumbersOptedOut com um AWS SDK ou CLI
- Use ListSubscriptions com um AWS SDK ou CLI
- Use ListTopics com um AWS SDK ou CLI
- Use Publish com um AWS SDK ou CLI
- Use SetSMSAttributes com um AWS SDK ou CLI
- Use SetSubscriptionAttributes com um AWS SDK ou CLI
- Use SetSubscriptionAttributesRedrivePolicy com um AWS SDK
- Use SetTopicAttributes com um AWS SDK ou CLI
- Use Subscribe com um AWS SDK ou CLI
- Use TagResource com um AWS SDK ou CLI
- Use Unsubscribe com um AWS SDK ou CLI

Olá, Amazon SNS

Os exemplos de código a seguir mostram como começar a usar o Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o CMake arquivo CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_sns.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
```

```
            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
  }
 }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicialize um cliente SNS e liste tópicos em sua conta.

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

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Kotlin.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
```

```
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O programa principal do Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"
```

```
    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
```

```
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para o Amazon SNS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon SNS com. AWS SDKs Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon SNS e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon SNS usando AWS SDKs .

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Simple Notification Service.

Exemplos

- Use CheckIfPhoneNumberIsOptedOut com um AWS SDK ou CLI
- Use ConfirmSubscription com um AWS SDK ou CLI

- Use CreateTopic com um AWS SDK ou CLI
- Use DeleteTopic com um AWS SDK ou CLI
- Use GetSMSAttributes com um AWS SDK ou CLI
- Use GetTopicAttributes com um AWS SDK ou CLI
- Use ListPhoneNumbersOptedOut com um AWS SDK ou CLI
- Use ListSubscriptions com um AWS SDK ou CLI
- Use ListTopics com um AWS SDK ou CLI
- Use Publish com um AWS SDK ou CLI
- Use SetSMSAttributes com um AWS SDK ou CLI
- Use SetSubscriptionAttributes com um AWS SDK ou CLI
- Use SetSubscriptionAttributesRedrivePolicy com um AWS SDK
- Use SetTopicAttributes com um AWS SDK ou CLI
- Use Subscribe com um AWS SDK ou CLI
- Use TagResource com um AWS SDK ou CLI
- Use Unsubscribe com um AWS SDK ou CLI

Use **`CheckIfPhoneNumberIsOptedOut`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para verificar o cancelamento de mensagens SMS para um número de telefone

O `check-if-phone-number-is-opted-out` exemplo a seguir verifica se o número de telefone especificado optou por não receber mensagens SMS da AWS conta atual.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Saída:

```
{
    "isOptedOut": false
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOutna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

```
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para PHP da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ConfirmSubscription`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ConfirmSubscription`.

CLI

AWS CLI

Para confirmar uma assinatura

O comando `confirm-subscription` a seguir conclui o processo de confirmação iniciado quando você se inscreveu em um tópico do SNS chamado `my-topic`. O parâmetro --token vem da mensagem de confirmação enviada ao endpoint de notificação especificado na chamada de assinatura.

```
aws sns confirm-subscription \
```

```
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Saída:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Para obter detalhes da API, consulte ConfirmSubscriptionna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte ConfirmSubscriptiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ConfirmSubscriptiona Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte ConfirmSubscriptiona Referência AWS SDK para PHP da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`CreateTopic`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTopic`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e publicar em um tópico FIFO
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico com um nome específico.

```
    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;
        }
    }
```

Crie um tópico com um nome e atributos específicos de FIFO e desduplicação.

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

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um tópico do SNS

O exemplo `create-topic` a seguir cria um tópico do SNS chamado `my-topic`.

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

Saída:

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

Para obter mais informações, consulte Usando a interface de linha de AWS comando com o Amazon SQS e o Amazon SNS no Guia do usuário AWS da interface de linha de comando.

- Para obter detalhes da API, consulte CreateTopicna Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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 "";
    }
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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()
    }
```

```
}
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


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

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

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a CreateTopicReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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

```
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

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

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
```

```
}
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   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.
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
        }
```

- Para obter detalhes da API, consulte CreateTopica referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteTopic`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTopic`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um tópico por meio do respectivo ARN.

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

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um tópico do SNS

O exemplo `delete-topic` a seguir exclui o tópico do SNS especificado.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteTopicna Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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
  //   }
  // }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DeleteTopicReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

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

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

- Para obter detalhes da API, consulte DeleteTopica referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetSMSAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte Get SMSAttributes in AWS SDK para C++ API Reference.

CLI

AWS CLI

Para listar os atributos padrão da mensagem SMS

O exemplo `get-sms-attributes` a seguir lista os atributos padrão para o envio de mensagens SMS.

```
aws sns get-sms-attributes
```

Saída:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Para ver detalhes da API, consulte GetSMSAttributes na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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");
    }
}
```

- Para obter detalhes da API, consulte Get SMSAttributes in AWS SDK for Java 2.x API Reference.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Get SMSAttributes in AWS SDK para JavaScript API Reference.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Get SMSAttributes in AWS SDK para PHP API Reference.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetTopicAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetTopicAttributes`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar os atributos de um tópico

O exemplo `get-topic-attributes` a seguir exibe os atributos do tópico especificado.

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

Saída:

```
{
    "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"
    }
}
```

- Para obter detalhes da API, consulte GetTopicAttributesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a 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);
  });
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte a GetTopicAttributesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para PHP da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a GetTopicAttributesreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListPhoneNumbersOptedOut`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Para listar as opções de cancelamento de mensagens SMS

O exemplo `list-phone-numbers-opted-out` a seguir lista os números de telefone que optaram por não receber mensagens SMS.

```
aws sns list-phone-numbers-opted-out
```

Saída:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOutna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOuta Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte ListPhoneNumbersOptedOuta Referência AWS SDK para PHP da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListSubscriptions`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListSubscriptions`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! 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;
}
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar suas assinaturas do SNS

O `list-subscriptions` exemplo a seguir exibe uma lista das assinaturas do SNS em sua conta. AWS

```
aws sns list-subscriptions
```

Saída:

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

- Para obter detalhes da API, consulte ListSubscriptionsna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
```

```
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ListSubscriptionsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a ListSubscriptionsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# This class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
```

```
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a ListSubscriptionsreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListTopics`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTopics`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! 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;
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Listar os tópicos do SNS

O `list-topics` exemplo a seguir lista todos os tópicos do SNS em sua AWS conta.

```
aws sns list-topics
```

Saída:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListTopicsna Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
  }
 }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a ListTopicsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
```

```
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
}
```

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
            }
        }
```

- Para obter detalhes da API, consulte ListTopicsa referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`Publish`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Publish`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e publicar em um tópico FIFO
- Publicar uma mensagem de texto SMS
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem em um tópico.

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

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

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

Aplique as seleções do usuário à ação de publicação.

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

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

Publicar uma mensagem com um atributo.

```
        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;
    }
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Exemplo 1: Para publicar uma mensagem em um tópico:

O exemplo `publish` a seguir publica a mensagem específica no tópico do SNS especificado. A mensagem é proveniente de um arquivo de texto, o que permite incluir quebras de linha.

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

Conteúdo de `message.txt`:

```
Hello World
Second Line
```

Saída:

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

Exemplo 2: Para publicar uma mensagem SMS em um número de telefone

O exemplo `publish` a seguir publica a mensagem `Hello world!` no número de telefone `+1-555-555-0100`.

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

Saída:

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

- Para obter detalhes da API, consulte Publish na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

```
  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();
    }
  }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


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

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

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

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para PHP .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

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

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$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
```

- Para obter detalhes da API, consulte Publicar no Ferramentas da AWS para PowerShell Cmdlet Reference (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

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

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$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
```

- Para obter detalhes da API, consulte Publicar no Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem com atributos para que uma assinatura possa filtrar com base em atributos.

```
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
```

Publique uma mensagem que assume diferentes formas com base no protocolo do assinante.

```
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
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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

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

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

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Ruby .

Rust

SDK for Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter os detalhes da API, consulte Publicar na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)")
```

- Para obter detalhes da API, consulte Publicar no AWS SDK para referência da API Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SetSMSAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Como usar o Amazon SNS para definir o atributo padrãoSMSType .

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

- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK para C++ da API.

CLI

AWS CLI

Para definir atributos de mensagens SMS

O exemplo `set-sms-attributes` a seguir define o ID do remetente padrão para mensagens SMS como `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Este comando não produz saída.

- Para obter detalhes da API, consulte Definir SMSAttributes na Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Definir SMSAttributes na referência AWS SDK para PHP da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SetSubscriptionAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetSubscriptionAttributes`.

CLI

AWS CLI

Para definir atributos de assinatura

O exemplo `set-subscription-attributes` a seguir define o atributo `RawMessageDelivery` para uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Este comando não produz saída.

O exemplo `set-subscription-attributes` a seguir define um atributo `FilterPolicy` para uma assinatura do SQS.

```
aws sns set-subscription-attributes \
```

```
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Este comando não produz saída.

O exemplo `set-subscription-attributes` a seguir remove o atributo `FilterPolicy` de uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetSubscriptionAttributesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte SetSubscriptionAttributesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a SetSubscriptionAttributesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SetSubscriptionAttributesRedrivePolicy`** com um AWS SDK

O código de exemplo a seguir mostra como usar `SetSubscriptionAttributesRedrivePolicy`.

Java

SDK para Java 1.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// 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);
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SetTopicAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetTopicAttributes`.

CLI

AWS CLI

Para definir um atributo para um tópico

O exemplo `set-topic-attributes` a seguir define o atributo `DisplayName` para o tópico especificado.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetTopicAttributesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte a SetTopicAttributesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
}
```

- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para PHP da API.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a SetTopicAttributesreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`Subscribe`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Subscribe`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e publicar em um tópico FIFO
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

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

Inscreva uma fila em um tópico com filtros opcionais.

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

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

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

Inscrever uma aplicação móvel em um tópico.

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

Inscrever uma função do Lambda em um tópico.

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

Assinar uma fila do SQS em um tópico.

```
        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;
           }
```

Assinar com um filtro em um tópico.

```
       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;
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para inscrever-se em um tópico

O comando `subscribe` a seguir inscreve um endereço de e-mail no tópico especificado.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Saída:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Para obter detalhes da API, consulte Subscribe na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscreva uma fila em um tópico com filtros opcionais.

```
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
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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);
```

```
        }
    }
}
```

Inscrever um endpoint HTTP em um tópico.

```
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);
        }
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
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 "";
    }
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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'
  // }
};
```

Inscrever uma aplicação móvel em um tópico.

```
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;
};
```

Inscrever uma função do Lambda em um tópico.

```
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;
};
```

Assinar uma fila do SQS em um tópico.

```
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;
};
```

Assinar com um filtro em um tópico.

```
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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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()
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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());
}
```

Inscrever um endpoint HTTP em um tópico.

```
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());
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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
```

- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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(())
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
   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.
```

- Para obter detalhes da API, consulte Inscrever-se na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
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.")
```

Inscreva um número de telefone em um tópico para receber notificações por SMS.

```
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.")
```

- Para obter detalhes da API, consulte Inscrever-se na referência da API do AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`TagResource`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Para adicionar uma tag a um tópico

O exemplo `tag-resource` a seguir adiciona uma tag de metadados ao tópico do Amazon SNS especificado.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte TagResourcena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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")
    }
}
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for Kotlin.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`Unsubscribe`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Unsubscribe`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cancele a assinatura de um tópico por meio de um ARN de assinatura.

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

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para cancelar a assinatura de um tópico

O exemplo `unsubscribe` a seguir exclui a assinatura especificada de um tópico.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Este comando não produz saída.

- Para obter detalhes da API, consulte Unsubscribe na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
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({});
```

Importe o SDK e os módulos do cliente e chame a 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;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte Cancelar assinatura na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


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

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte Unsubscribe na Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Cancelar assinatura na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
```

- Para obter detalhes da API, consulte Cancelar assinatura na referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para o Amazon SNS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon SNS com. AWS SDKs Esses casos mostram como realizar tarefas específicas chamando várias funções dentro

do Amazon SNS ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação para enviar dados para uma tabela do DynamoDB
- Criar uma aplicação de publicação e assinatura que traduz mensagens
- Crie um endpoint de plataforma para notificações push do Amazon SNS usando um SDK AWS
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Criar uma aplicação de exploração do Amazon Textract
- Crie e publique em um tópico FIFO do Amazon SNS usando um SDK AWS
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS
- Publique mensagens SMS em um tópico do Amazon SNS usando um SDK AWS
- Publique uma mensagem grande no Amazon SNS com o Amazon S3 usando um SDK AWS
- Publique uma mensagem de texto SMS do Amazon SNS usando um SDK AWS
- Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS
- Usar o API Gateway para invocar uma função do Lambda
- Usar eventos programados para invocar uma função do Lambda

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

Os exemplos de código a seguir mostram como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

Java

SDK para Java 2.x

Mostra como criar uma aplicação Web dinâmica que envia dados usando a API Java do Amazon DynamoDB e envia uma mensagem de texto usando a API Java do Amazon Simple Notification Service.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Este exemplo mostra como criar uma aplicação que permite que os usuários enviem dados para uma tabela do Amazon DynamoDB e enviem uma mensagem de texto ao administrador usando o Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de publicação e assinatura que traduz mensagens

Os exemplos de código a seguir mostram como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon SNS
- Amazon Translate

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que usa a API Java Async, consulte o exemplo completo em. GitHub

Serviços utilizados neste exemplo

- Amazon SNS
- Amazon Translate

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e as instruções sobre como criar um aplicativo web, veja o exemplo completo em GitHub.

Para ver o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon SNS
- Amazon Translate

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Crie um endpoint de plataforma para notificações push do Amazon SNS usando um SDK AWS

Os exemplos de código a seguir mostram como criar um endpoint de plataforma para notificações por push do Amazon SNS.

CLI

AWS CLI

Para criar um endpoint de aplicação de plataforma

O exemplo `create-platform-endpoint` a seguir cria um endpoint para a aplicação de plataforma especificada usando o token especificado.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Saída:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
```

```
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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());
```

```
        }
    }
}
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple

Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use o modo assíncrono APIs para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Identidade do Amazon Cognito

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Crie e publique em um tópico FIFO do Amazon SNS usando um SDK AWS

Os exemplos de código a seguir mostram como criar e publicar em um tópico FIFO do Amazon SNS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O teste verifica o recebimento da mensagem em cada fila. O exemplo completo também mostra a adição de políticas de acesso e exclui os recursos no final.

```
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);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTopic
  - Publicar
  - Assinar

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
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
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTopic

- Publicar
- Assinar

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

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

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

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

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

 iv_stringvalue = 'High' ).
       INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

       DATA(lo_result) = lo_sns->publish(
            iv_topicarn = lv_topic_arn
            iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes ).
       ov_message_id = lo_result->get_messageid( ).                      "
 ov_message_id is returned for testing purposes. "
       MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
```

```
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
  - CreateTopic
  - Publicar
  - Assinar

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Publique mensagens SMS em um tópico do Amazon SNS usando um SDK AWS

O código de exemplo a seguir mostra como:

- Criar um tópico do Amazon SNS.
- Inscrever números de telefone no tópico.

- Publicar mensagens SMS no tópico para que todos os números de telefone inscritos recebam a mensagem de uma só vez.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar um tópico e retorne seu 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 "";
    }
}
```

Inscreva um endpoint em um tópico.

```
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);
        }
    }
}
```

Defina atributos na mensagem, como o ID do remetente, o preço máximo e seu tipo. Os atributos de mensagem são opcionais.

```
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);
        }
    }
}
```

Publique uma mensagem em um tópico. A mensagem é enviada para todos os assinantes.

```
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);
        }
    }
}
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Publique uma mensagem grande no Amazon SNS com o Amazon S3 usando um SDK AWS

O exemplo de código a seguir mostra como publicar uma mensagem grande no Amazon SNS usando o Amazon S3 para armazenar a carga útil da mensagem.

Java

SDK para Java 1.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para publicar uma mensagem grande, use a Amazon SNS Extended Client Library for Java. A mensagem que você envia faz referência a um objeto do Amazon S3 que contém o conteúdo real da mensagem.

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

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Publique uma mensagem de texto SMS do Amazon SNS usando um SDK AWS

Os exemplos de código a seguir mostram como publicar mensagens SMS usando o Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}");
            }
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para C++ .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

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

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


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';
```

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

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte Publish na Referência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS

Os exemplos de código a seguir mostram como:

- Crie um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

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

Crie uma classe que envolva operações do Amazon SQS.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;
```

```
    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
```

```
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
```

```
                                    "\"Effect\": \"Allow\"," +
                                    "\"Principal\": {" +
                                        $"\"Service\": " +
                                            "\"sns.amazonaws.com\"" +
                                           "}," +
                                    "\"Action\": \"sqs:SendMessage\"," +
                                    $"\"Resource\": \"{queueArn}\"," +
                                     "\"Condition\": {" +
                                          "\"ArnEquals\": {" +
                                              $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                           "}" +
                                       "}" +
                                  "}]" +
                               "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
```

```
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
}
```

Crie uma classe que envolva operações do Amazon SNS.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

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

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
```

```
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

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

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

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
```

```
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

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

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

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

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


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        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();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
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))
}
```

Defina um struct que envolva as ações do Amazon SNS usadas neste exemplo.

```
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
}
```

Defina um struct que envolva as ações do Amazon SQS usadas neste exemplo.

```
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
}
```

Limpar recursos.

```
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)
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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 "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada para esse cenário.

```
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();
};
```

O código anterior fornece as dependências necessárias e inicia o cenário. A próxima seção contém a maior parte do exemplo.

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar o API Gateway para invocar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Lambda Java. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma

mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3).

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon SNS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar eventos programados para invocar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

Java

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função Lambda usando a API de tempo de execução do JavaScript Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços utilizados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento programado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função Lambda o alvo.
- Concede permissão para permitir a EventBridge invocação da função Lambda.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações programadas.

- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos sem servidor para o Amazon SNS

Os exemplos de código a seguir mostram como usar o Amazon SNS com. AWS SDKs

Exemplos

- Invocar uma função do Lambda em um acionador do Amazon SNS

Invocar uma função do Lambda em um acionador do Amazon SNS

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
```

```
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
```

```
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
```

```
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do SNS com o JavaScript Lambda usando.

```
// 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;
  }
}
```

Consumindo um evento do SNS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
```

```
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SNS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon SNS com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Segurança do Amazon SNS

O modelo de responsabilidade AWS compartilhada se aplica à proteção de dados no Amazon Simple Notification Service. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração e gerenciamento de segurança dos AWS serviços que você usa. Para obter mais informações sobre a privacidade de dados, consulte as Data Privacy FAQ. Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog AWS Shared Responsibility Model and RGPD no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure contas de usuário individuais com AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Recomendamos usar o TLS 1.2 ou posterior.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções AWS de criptografia, juntamente com todos os controles de segurança padrão nos AWS serviços.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para ter mais informações sobre endpoints do FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-2.
- Proteção de dados de mensagens
  - A proteção de dados de mensagens é um recurso novo e importante do Amazon SNS.
  - Use o MDP para verificar mensagens quanto a informações sigilosas ou confidenciais.
  - Ofereça auditoria de mensagens para todo o conteúdo que flui pelo tópico.
  - Forneça controles de acesso ao conteúdo às mensagens publicadas no tópico e às mensagens entregues pelo tópico.

> **Important**
>
> É altamente recomendável que você nunca coloque informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com o Amazon SNS ou outros Amazon Web Services usando o console, a API ou. AWS CLI AWS SDKs Quaisquer dados inseridos em marcações ou campos de formato livre usados para nomes podem ser usados para logs de cobrança ou diagnóstico. Se fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia de dados do Amazon SNS

Proteção de dados protege os dados em trânsito (à medida que são transferidos para e do Amazon SNS) e em repouso (enquanto estão armazenados em discos em datacenters do Amazon SNS). Você pode proteger os dados em trânsito usando Secure Sockets Layer (SSL) ou criptografia no lado do cliente. Por padrão, o Amazon SNS armazena mensagens e arquivos usando criptografia de disco. Para proteger dados em repouso, solicite que o Amazon SNS criptografe as mensagens antes de salvá-las nos sistemas de arquivo criptografados em seus datacenters. O Amazon SNS recomenda o uso do SSE para otimizar a criptografia de dados.

Segurança dos dados do Amazon SNS com a criptografia do lado do servidor

A criptografia do lado do servidor (SSE) permite armazenar dados confidenciais em tópicos criptografados, protegendo o conteúdo das mensagens nos tópicos do Amazon SNS usando chaves gerenciadas em (). AWS Key Management Service AWS KMS

A SSE criptografa mensagens assim que o Amazon SNS as recebe. As mensagens são armazenadas no formato criptografado e são descriptografadas apenas quando são enviadas.

- Para obter informações sobre como gerenciar o SSE usando o AWS Management Console ou o AWS SDK para Java (definindo o `KmsMasterKeyId` atributo usando as ações `CreateTopic` e `SetTopicAttributes` da API), consulte Configuração da criptografia de tópico do Amazon SNS com a criptografia do lado do servidor.

- Para obter informações sobre como criar tópicos criptografados usando AWS CloudFormation (configurando a `KmsMasterKeyId` propriedade usando o `AWS::SNS::Topic` recurso), consulte o Guia AWS CloudFormation do usuário.

> **Important**
>
> Todas as solicitações para tópicos com SSE ativada devem usar HTTPS e o Signature versão 4.
> Para obter informações sobre a compatibilidade de outros serviços com filas criptografadas, consulte a documentação do seu serviço.
> O Amazon SNS só é compatível com chaves do KMS de criptografia simétrica. Você não pode usar nenhum outro tipo de chave do KMS para criptografar seus recursos de serviço. Para obter ajuda para determinar se uma chave do KMS é simétrica, consulte Identificar chaves assimétricas do KMS.

AWS KMS combina hardware e software seguros e de alta disponibilidade para fornecer um sistema de gerenciamento de chaves dimensionado para a nuvem. Quando você usa o Amazon SNS com AWS KMS, as chaves de dados que criptografam os dados da sua mensagem também são criptografadas e armazenadas com os dados que elas protegem.

Os benefícios de usar o AWS KMS são os seguintes:

- Você pode criar e gerenciar o AWS KMS key por conta própria.
- Você também pode usar chaves KMS AWS gerenciadas para o Amazon SNS, que são exclusivas para cada conta e região.
- Os padrões AWS KMS de segurança podem ajudá-lo a atender aos requisitos de conformidade relacionados à criptografia.

Para obter mais informações, consulte O que é AWS Key Management Service? no Guia do AWS Key Management Service desenvolvedor.

Escopo de criptografia

A SSE criptografa o corpo de uma mensagem em um tópico do Amazon SNS.

A SSE não criptografa o seguinte:

- Metadados de tópico (nome e atributos do tópico)
- Metadados de mensagens (assunto, ID de mensagem, carimbo de data/hora e atributos)
- Política de proteção de dados
- Métricas por tópico

> **Note**
>
> - Uma mensagem só será criptografada se for enviada após a habilitação da criptografia de um tópico. O Amazon SNS não criptografa mensagens com lista de pendências.
> - Qualquer mensagem criptografada permanecerá dessa forma mesmo se a criptografia de seu tópico for desabilitada.

Principais termos

Os seguintes termos-chave podem ajudar você a entender melhor a funcionalidade da SSE. Para obter descrições detalhadas, consulte a Referência da API do Amazon Simple Notification Service.

Chave de dados

A chave de criptografia dos dados (DEK) responsável por criptografar o conteúdo de mensagens do Amazon SNS.

Para obter mais informações, consulte Chaves de dados no Guia do desenvolvedor do AWS Key Management Service e Criptografia envelopada no Guia do desenvolvedor do AWS Encryption SDK .

AWS KMS key ID

O alias, o ARN do alias, o ID da chave ou o ARN da chave de um ou de um AWS KMS key personalizado AWS KMS— em sua conta ou em outra conta. Embora o alias do AWS gerenciado AWS KMS para o Amazon SNS seja `alias/aws/sns` sempre, o alias de um AWS KMS personalizado pode, por exemplo, ser. `alias/MyAlias` Você pode usar essas chaves do AWS KMS para proteger as mensagens em tópicos do Amazon SNS.

> **Note**
>
> Lembre-se do seguinte:

- A primeira vez que você usa o AWS Management Console para especificar o KMS AWS gerenciado para o Amazon SNS para um tópico AWS KMS , cria AWS o KMS gerenciado para o Amazon SNS.
- Como alternativa, na primeira vez que você usa a `Publish` ação em um tópico com o SSE ativado, AWS KMS cria o KMS AWS gerenciado para o Amazon SNS.

Você pode criar AWS KMS chaves, definir as políticas que controlam como AWS KMS as chaves podem ser usadas e auditar o AWS KMS uso usando a AWS KMS keysseção do AWS KMS console ou a `CreateKey` AWS KMS ação. Para obter mais informações, consulte AWS KMS keys e Criação de chaves no Guia do desenvolvedor do AWS Key Management Service . Para ver mais exemplos de AWS KMS identificadores, consulte KeyIda Referência da AWS Key Management Service API. Para obter informações sobre AWS KMS como encontrar identificadores, consulte Encontre o ID da chave e o ARN no Guia AWS Key Management Service do desenvolvedor.

Important

Há taxas adicionais pelo uso AWS KMS. Para obter mais informações, consulte Estimando custos AWS KMS e Definição de preço do AWS Key Management Service.

Gerenciar chaves e custos de criptografia do Amazon SNS

As seções a seguir contêm informações sobre como trabalhar com chaves gerenciadas pelo AWS Key Management Service (AWS KMS).

Note

O Amazon SNS só é compatível com chaves do KMS de criptografia simétrica. Você não pode usar nenhum outro tipo de chave do KMS para criptografar seus recursos de serviço. Para obter ajuda para determinar se uma chave do KMS é simétrica, consulte Identificar chaves assimétricas do KMS.

Estimando custos AWS KMS

Para prever custos e entender melhor sua AWS fatura, talvez você queira saber com que frequência o Amazon SNS usa sua. AWS KMS key

> **Note**
>
> Embora a fórmula a seguir possa dar a você uma boa ideia sobre os custos esperados, os custos reais poderão ser mais altos por conta da natureza distribuída do Amazon SNS.

Para calcular o número de solicitações de APIs (R) por tópico, use a seguinte fórmula:

```
R = B / D * (2 * P)
```

B é o período de faturamento (em segundos).

D é o período de reutilização da chave de dados (em segundos: o Amazon SNS reutiliza uma chave de dados por até cinco minutos).

P é o número de principais de publicação enviados para o tópico do Amazon SNS.

Estes são cálculos de exemplo. Para obter informações sobre a definição de preços, consulte Definição de preços do AWS Key Management Service.

Exemplo 1: cálculo do número de chamadas de AWS KMS API para 1 editor e 1 tópico

Este exemplo supõe o seguinte:

- O período de faturamento é de 1 a 31 de janeiro (2.678.400 segundos).
- O período de reutilização de chave de dados é de 5 minutos (300 segundos).
- Há 1 tópico.
- Há 1 entidade principal de publicação.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Exemplo 2: Cálculo do número de chamadas à API do AWS KMS para vários editores e 2 tópicos

Este exemplo supõe o seguinte:

- O período de faturamento é de 1 a 28 de fevereiro (2.419.200 segundos).
- O período de reutilização de chave de dados é de 5 minutos (300 segundos).
- Há 2 tópicos.
- O primeiro tópico tem 3 entidades principais de publicação.
- O segundo tópico tem 5 entidades principais de publicação.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Configurando permissões AWS KMS

Antes de usar o SSE, você deve configurar AWS KMS key políticas para permitir a criptografia de tópicos e criptografia e descriptografia de mensagens. Para obter exemplos e mais informações sobre as permissões do AWS KMS , consulte AWS KMS API Permissions: Actions and Resources Reference no Guia do desenvolvedor do AWS Key Management Service . Para obter detalhes sobre como configurar um tópico do Amazon SNS com a criptografia do lado do servidor, consulte Mais informações.

> **Note**
>
> Você também pode gerenciar permissões para criptografia simétrica de chaves do KMS usando políticas do IAM. Para obter mais informações, consulte Como usar políticas do IAM com AWS KMS.
> Embora você possa configurar permissões globais para enviar e receber do Amazon SNS, é AWS KMS necessário nomear explicitamente o ARN completo de regiões específicas KMSs na seção de uma política do `Resource` IAM.

Você também deve garantir que as principais políticas do AWS KMS key permitam as permissões necessárias. Para fazer isso, indique as entidades principais que produzem e consomem mensagens criptografadas no Amazon SNS como usuários na política de chaves do KMS.

Como alternativa, você pode especificar as AWS KMS ações necessárias e o ARN do KMS em uma política do IAM atribuída aos diretores que publicam e se inscrevem para receber mensagens criptografadas no Amazon SNS. Para obter mais informações, consulte Gerenciar o acesso ao AWS KMS no Guia do desenvolvedor do AWS Key Management Service .

Se você estiver selecionando uma chave gerenciada pelo cliente para seu tópico do Amazon SNS e estiver usando aliases para controlar o acesso às chaves do KMS usando políticas do IAM ou políticas de chave do KMS com a chave de condição `kms:ResourceAliases`, a chave gerenciada pelo cliente selecionada também deverá ter um alias associado. Para obter mais informações sobre o uso de alias para controlar o acesso às chaves do KMS, consulte Usar aliases para controlar o acesso às chaves do KMS no Guia do desenvolvedor do AWS Key Management Service .

Permitir que um usuário envie mensagens a um tópico com SSE

O editor deve ter as permissões `kms:GenerateDataKey*` e `kms:Decrypt` para o AWS KMS key.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Habilite a compatibilidade entre fontes de eventos de AWS serviços e tópicos criptografados

Vários AWS serviços publicam eventos em tópicos do Amazon SNS. Para permitir que essas fontes de eventos trabalhem com tópicos criptografados, você deve executar as seguintes etapas:

1. Use uma chave gerenciada pelo cliente. Para obter mais informações, consulte Criação de chaves no Guia do desenvolvedor AWS Key Management Service .
2. Para permitir que o AWS serviço tenha as `kms:Decrypt` permissões `kms:GenerateDataKey*` e, adicione a seguinte declaração à política do KMS.

   ```
   {
     "Statement": [{
   ```

```
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Origem do evento.	Entidade principal do serviço
Amazon CloudWatch	`cloudwatch.amazonaws.com`
CloudWatch Eventos da Amazon	`events.amazonaws.com`
AWS CodeCommit	`codecommit.amazonaws.com`
AWS Database Migration Service	`dms.amazonaws.com`
AWS Directory Service	`ds.amazonaws.com`
Amazon DynamoDB	`dynamodb.amazonaws.com`
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`
Amazon Simple Email Service	`ses.amazonaws.com`
Amazon Simple Storage Service	`s3.amazonaws.com`
AWS Snowball Edge	`importexport.amazonaws.com`

Origem do evento.	Entidade principal do serviço
AWS Systems Manager Gerenciador de incidentes	AWS O Systems Manager Incident Manager consiste em dois princípios de serviço: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Algumas fontes de eventos do Amazon SNS exigem que você forneça uma função do IAM (em vez da principal do serviço) na AWS KMS key política:
>
> - Amazon EC2 Auto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. Adicione as chaves de condição `aws:SourceAccount` e `aws:SourceArn` à política de recursos do KMS para proteger ainda mais a chave do KMS de ataques confused deputy. Consulte a lista de documentação específica do serviço (acima) para obter detalhes exatos em cada caso.

> **Important**
>
> Não há suporte `aws:SourceOrgID` para adicionar o `aws:SourceAccountaws:SourceArn`,, e a uma AWS KMS política para EventBridge-to-encrypted tópicos.

```
{
  "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. Habilite a SSE para seu tópico usando seu KMS.
5. Forneça o ARN do tópico criptografado para a fonte do evento.

AWS KMS erros

Ao trabalhar com o Amazon SNS AWS KMS, você pode encontrar erros. A lista a seguir descreve os erros e as possíveis soluções de problemas.

KMSAccessDeniedException

O texto cifrado faz referência a uma chave que não existe ou à qual você não tem acesso.

Código de status HTTP: 400

KMSDisabledExceção

A solicitação foi recusada porque o KMS especificado não está habilitado.

Código de status HTTP: 400

KMSInvalidStateException

A solicitação foi rejeitada porque o estado do recurso especificado não é válido para essa solicitação. Para obter mais informações, consulte Principais estados do AWS KMS keys no Guia do desenvolvedor do AWS Key Management Service .

Código de status HTTP: 400

KMSNotFoundException

A solicitação foi rejeitada porque a entidade ou o recurso especificado não pôde ser encontrado.

Código de status HTTP: 400

KMSOptInRequired

O ID da chave de AWS acesso precisa de uma assinatura para o serviço.

Código de status HTTP: 403

KMSThrottlingExceção

A solicitação foi negada devido à limitação da solicitação. Para obter mais informações sobre controle de utilização, consulte Cotas no Guia do desenvolvedor do AWS Key Management Service .

Código de status HTTP: 400

Configuração da criptografia de tópico do Amazon SNS com a criptografia do lado do servidor

O Amazon SNS oferece suporte à criptografia do lado do servidor (SSE) para proteger o conteúdo das mensagens usando (). AWS Key Management Service AWS KMS Siga as instruções abaixo para habilitar o SSE usando o console do Amazon SNS ou o CDK.

Opção 1: ativar a criptografia usando o AWS Management Console

1. Faça login no console do Amazon SNS.
2. Navegue até a página Tópicos, selecione seu tópico e escolha Editar.
3. Expanda a seção Criptografia e faça o seguinte:
   - Alterne a criptografia para Ativar.
   - Selecione o AWS gerenciado SNS Chave (alias/aws/sns) como chave de criptografia. Isso é selecionado por padrão.
4. Escolha Salvar alterações.

> **Note**
>
> - O Chave gerenciada pela AWS é criado automaticamente se ainda não existir.
> - Se você não vê a chave ou tem permissões insuficientes, peça ao administrador `kms:ListAliases` `kms:DescribeKey` e.

Opção 2: ativar a criptografia usando AWS CDK

Para usar o AWS gerenciado SNS chave em seu aplicativo CDK, adicione o seguinte trecho:

```
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();
    }
}
```

Mais informações

- Chave KMS personalizada — Você pode especificar uma chave personalizada, se necessário. No console do Amazon SNS, selecione sua chave KMS personalizada na lista ou insira o ARN.
- Permissões para chaves KMS personalizadas — Se estiver usando uma chave KMS personalizada, inclua o seguinte na política de chaves para permitir que o Amazon SNS criptografe e descriptografe mensagens:

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

Impacto nos consumidores

A ativação do SSE não altera a forma como os assinantes consomem mensagens. AWS gerencia a criptografia e a descriptografia de forma transparente. As mensagens permanecem criptografadas em repouso e são automaticamente descriptografadas antes da entrega aos assinantes. Para uma segurança ideal, AWS recomenda habilitar HTTPS para todos os endpoints para garantir a transmissão segura de mensagens.

Configuração da criptografia de tópicos do Amazon SNS com assinatura criptografada de filas do Amazon SQS

Você pode habilitar a criptografia no lado do servidor (SSE) para um tópico com o intuito de proteger seus dados. Para permitir que o Amazon SNS envie mensagens para filas do Amazon SQS criptografadas, a chave mestra de cliente (CMK) associada à fila do Amazon SQS deve ter uma declaração de política que conceda ao Amazon SNS o acesso principal ao serviço para as ações da API do AWS KMS , `GenerateDataKey` e `Decrypt`. Para obter mais informações sobre como usar a SSE, consulte Segurança dos dados do Amazon SNS com a criptografia do lado do servidor.

Este tópico explica como habilitar o SSE para um tópico do Amazon SNS com uma assinatura de fila criptografada do Amazon SQS usando o. AWS Management Console

Etapa 1: como criar uma chave do KMS

1. Faça login no console do AWS KMS com um usuário que tenha pelo menos a política `AWSKeyManagementServicePowerUser`.
2. Escolha Criar uma chave.
3. Para criar uma chave do KMS de criptografia simétrica, em Tipo de chave, selecione Simétrica.

   Para obter informações sobre como criar uma chave do KMS assimétrica no console do AWS KMS , consulte Criar chaves do KMS assimétricas (console).
4. Em Uso da chave, a opção Criptografar e descriptografar é selecionada para você.

   Para obter informações sobre como criar chaves do KMS que geram e verificam códigos MAC, consulte Criar chaves do KMS de HMAC.

   Para obter mais informações sobre as opções avançadas, consulte Chaves para fins especiais.
5. Escolha Próximo.
6. Digite um alias para a chave do KMS. O nome do alias não pode começar com **`aws/`**. O **`aws/`** prefixo é reservado pela Amazon Web Services para representar Chaves gerenciadas pela AWS em sua conta.

   > **Note**
   >
   > Adicionar, excluir ou atualizar um alias pode conceder ou negar uma permissão à chave do KMS. Para obter detalhes, consulte ABAC para AWS KMS e Uso de aliases para controlar o acesso às chaves do KMS.

   Um alias é um nome de exibição que identifica a chave do KMS. Recomendamos que você escolha um alias que indique o tipo de dados que pretende proteger ou a aplicação a ser usada com a chave do KMS.

   Aliases são necessários ao criar uma chave do KMS no AWS Management Console. Eles são opcionais quando você usa a CreateKeyoperação.
7. (Opcional) Digite uma descrição para a chave do KMS.

   Você pode adicionar uma descrição agora ou atualizá-la a qualquer momento, a não ser que o estado da chave seja `Pending Deletion` ou `Pending Replica Deletion`. Para adicionar,

alterar ou excluir a descrição de uma chave gerenciada pelo cliente existente, edite a descrição na operação AWS Management Console ou use a UpdateKeyDescriptionoperação.

8. (Opcional) Digite uma chave de tag e um valor de tag opcional. Para adicionar mais de uma etiqueta à chave do KMS, selecione Adicionar tag.

> **Note**
>
> Marcar ou desmarcar uma chave do KMS pode conceder ou negar uma permissão a essa chave do KMS. Para obter detalhes, consulte ABAC para AWS KMS e Uso de tags para controlar o acesso às chaves do KMS.

Quando você adiciona tags aos seus AWS recursos, AWS gera um relatório de alocação de custos com uso e custos agregados por tags. Tags também podem ser utilizadas para controlar o acesso a uma chave do KMS. Para informações sobre marcação de chaves do KMS, consulte Marcação de chaves e ABAC para AWS KMS.

9. Escolha Próximo.
10. Selecione os usuários e as funções do IAM que podem administrar a chave do KMS.

> **Note**
>
> Essa política de chaves dá o controle Conta da AWS total dessa chave KMS. Ela permite que os administradores de conta usem políticas do IAM para conceder a outras entidades principais a permissão para gerenciar a chave do KMS. Para obter mais detalhes, consulte Política de chaves padrão.
>
> As práticas recomendadas do IAM não encorajam o uso de usuários do IAM com credenciais de longo prazo. Sempre que possível, use os perfis do IAM, por fornecerem credenciais temporárias. Para obter detalhes, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

11. (Opcional) Para evitar que os usuários e funções do IAM selecionados excluam essa chave do KMS, na seção Exclusão de chaves na parte inferior da página, desmarque a caixa de seleção Permitir que os administradores de chaves excluam essa chave.
12. Escolha Próximo.

13. Selecione os usuários e os perfis do IAM que podem usar a chave em operações de criptografia. Escolha Próximo.
14. Na página Revisar e editar política de chaves, adicione a seguinte instrução à política de chaves e, em seguida, escolha Concluir.

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

Sua nova chave gerenciada pelo cliente será exibida na lista de chaves.

Etapa 2: como criar um tópico do Amazon SNS criptografado

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Escolha Criar tópico.
4. Na página Criar novo tópico, em Nome, digite um nome para o tópico (por exemplo, `MyEncryptedTopic`) e escolha Criar tópico.
5. Expanda a seção Criptografia e faça o seguinte:

   a. Escolha Habilitar criptografia no lado do servidor.

   b. Especifique a chave gerenciada pelo cliente. Para obter mais informações, consulte Principais termos.

      Para cada tipo de chave gerenciada pelo cliente, a Descrição, a Conta e o ARN da chave gerenciada pelo cliente são exibidos.

> **Important**
>
> Se você não for o proprietário da chave gerenciada pelo cliente ou se fizer login com uma conta que não tenha as permissões `kms:ListAliases` e `kms:DescribeKey`, não será possível visualizar as informações sobre a chave gerenciada pelo cliente no console do Amazon SNS.
> Peça ao proprietário da chave gerenciada pelo cliente para conceder essas permissões a você. Para obter mais informações, consulte Permissões da API do KMS: referência de ações e recursos do AWS KMS no Guia do desenvolvedor do AWS Key Management Service .

c. Para a chave gerenciada pelo cliente, escolha a MyCustomKeyque você criou anteriormente e, em seguida, escolha Ativar criptografia do lado do servidor.

6. Escolha Salvar alterações.

   O SSE está ativado para o seu tópico e a MyTopicpágina é exibida.

   O status de Criptografia, a Conta da AWS , a Chave mestra do cliente, o ARN e a Descrição do tópico são exibidos na guia Criptografia.

Seu novo tópico criptografado será exibido na lista de tópicos.

Etapa 3: como criar e se inscrever em filas criptografadas do Amazon SQS

1. Faça login no console do Amazon SQS.
2. Selecione Criar nova fila.
3. Na página Create New Queue (Criar nova fila), faça o seguinte:

   a. Insira um Queue Name (Nome da fila) (por exemplo, `MyEncryptedQueue1`).

   b. Escolha Standard Queue (Fila padrão) e, em seguida, escolha Configure Queue (Configurar fila).

   c. Selecione Use SSE (Usar SSE).

   d. Para AWS KMS key, escolha MyCustomKeyo que você criou anteriormente e, em seguida, escolha Criar fila.

4. Repita o processo para criar uma segunda fila (por exemplo, chamada `MyEncryptedQueue2`).

Suas novas filas criptografadas serão exibidas na lista de filas.

5. No console do Amazon SQS, selecione `MyEncryptedQueue1` e `MyEncryptedQueue2`, e escolha Queue Actions (Ações da fila), Subscribe Queues to SNS Topic (Inscrever filas em um tópico do SNS).
6. Na caixa de diálogo Inscrever-se em um tópico, em Escolher um tópico MyEncryptedTopic, selecione e, em seguida, escolha Inscrever-se.

   Suas inscrições das filas criptografadas para o seu tópico criptografado são exibidas na caixa de diálogo Resultado de inscrição do tópico.
7. Escolha OK.

Etapa 4: como publicar uma mensagem no seu tópico criptografado

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos.
3. Na lista de tópicos, escolha MyEncryptedTopice, em seguida, escolha Publicar mensagem.
4. Na página Publicar uma mensagem, faça o seguinte:

   a. (Opcional) Na seção Detalhes da mensagem, insira o Assunto (por exemplo, `Testing message publishing`).

   b. Na seção Corpo da mensagem, insira o corpo da mensagem (por exemplo, `My message body is encrypted at rest.`).

   c. Selecione Publish message (Publicar mensagem).

Sua mensagem será publicada nas suas filas criptografadas inscritas.

Etapa 5: como verificar entrega de mensagens

1. Faça login no console do Amazon SQS.
2. Na lista de filas, escolha MyEncryptedQueue1 e, em seguida, escolha Enviar e receber mensagens.
3. Na página Enviar e receber mensagens em MyEncryptedQueue 1, escolha Sondagem de mensagens.

   A mensagem que você enviou anteriormente será exibida.

4. Escolha Mais detalhes para visualizar sua mensagem.
5. Quando terminar, escolha Fechar.
6. Repita o processo para MyEncryptedQueue2.

Proteger o tráfego do Amazon SNS com endpoints da VPC

Um endpoint da Virtual Private Cloud (Amazon VPC) para Amazon SNS é uma entidade lógica dentro de uma VPC que permite conectividade apenas com o Amazon SNS. A VPC roteia as solicitações para o Amazon SNS e as respostas de volta para a VPC. As seções a seguir contêm informações sobre como trabalhar com VPC endpoints e criar políticas de VPC endpoint.

Se você usa a Amazon Virtual Private Cloud (Amazon VPC) para hospedar seus AWS recursos, você pode estabelecer uma conexão privada entre sua VPC e o Amazon SNS. Com essa conexão, você pode publicar mensagens nos tópicos do Amazon SNS sem enviá-las pela Internet pública.

O Amazon VPC é um AWS serviço que você pode usar para lançar AWS recursos em uma rede virtual que você define. Com a VPC, você tem controle sobre as configurações de rede, como o intervalo de endereços IP, sub-redes, tabelas de rotas e gateways de rede. Para conectar a VPC ao Amazon SNS, você define um VPC endpoint de interface. Esse tipo de endpoint permite que você conecte sua VPC AWS aos serviços. O endpoint fornece uma conectividade confiável e escalável ao Amazon SNS sem a necessidade de um gateway da Internet, de uma instância de conversão de endereço de rede (NAT) ou de uma conexão VPN. Para obter mais informações, consulte Acessar e AWS service (Serviço da AWS) usar uma interface VPC endpoint no Guia do usuário da Amazon VPC.

As informações nesta seção são para usuários da Amazon VPC. Para obter mais informações e começar a criar uma VPC, consulte Planejar sua VPC no Guia do usuário da Amazon VPC.

> **Note**
>
> VPC endpoints não permitem que você se inscreva em um tópico do Amazon SNS para um endereço IP privado.

Criar um endpoint da Amazon VPC para o Amazon SNS

Para publicar mensagens nos tópicos do Amazon SNS a partir de uma Amazon VPC, crie um VPC endpoint de interface. Em seguida, você poderá publicar mensagens nos seus tópicos, mantendo o tráfego na rede gerenciada com a VPC.

Use as informações a seguir para criar o endpoint e testar a conexão entre a VPC e o Amazon SNS. Consulte Publicar uma mensagem do Amazon SNS da Amazon VPC para ver instruções detalhadas sobre como começar do zero.

Criar o endpoint

Você pode criar um endpoint do Amazon SNS em sua VPC usando o AWS Management Console, o, um AWS SDK AWS CLI, a API do Amazon SNS ou. AWS CloudFormation

Para obter informações sobre como criar e configurar um endpoint usando o console da Amazon VPC ou a AWS CLI, consulte Creating an Interface Endpoint (“Criar um endpoint da interface”) no Manual do usuário da Amazon VPC.

> **Important**
>
> Você pode usar a Amazon Virtual Private Cloud somente com endpoints HTTPS do Amazon SNS.
> Ao criar um endpoint, especifique o Amazon SNS como o serviço ao qual a VPC deve se conectar. No console da Amazon VPC, os nomes de serviços variam de acordo com a região. Por exemplo, se você escolher Leste dos EUA (Norte da Virgínia), o nome do serviço será com.amazonaws.us-east-1.sns.
> Ao configurar o Amazon SNS para enviar mensagens pela Amazon VPC, habilite o DNS privado e especifique endpoints no formato `sns.us-east-2.amazonaws.com`.
> O DNS privado não oferece suporte a endpoints legados, como `queue.amazonaws.com` ou `us-east-2.queue.amazonaws.com`.

Para obter informações sobre como criar e configurar um endpoint usando AWS CloudFormation, consulte o `AWS::EC2::VPCEndpoint`recurso no Guia do AWS CloudFormation usuário.

Como testar a conexão entre a VPC e o Amazon SNS

Depois de criar um endpoint para o Amazon SNS, você pode publicar mensagens a partir da VPC nos tópicos do Amazon SNS. Para testar essa conexão, faça o seguinte:

1. Conecte-se a uma EC2 instância da Amazon que reside em sua VPC. Para obter informações sobre conexão, consulte Conecte-se à sua instância Linux ou Conecte-se à sua instância do Windows na EC2 documentação da Amazon.

   Por exemplo, para se conectar a uma instância do Linux usando um cliente SSH, execute o comando a seguir a partir de um terminal:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Em que:

   - ec2-key-pair.pem é o arquivo que contém o par de chaves que a Amazon EC2 forneceu quando você criou a instância.
   - instance-hostname é o nome de host público da instância. Para obter o nome do host no EC2console da Amazon: Escolha Instâncias, escolha sua instância e encontre o valor para o DNS público.

2. Na sua instância, use o comando `publish` do Amazon SNS com a AWS CLI. Você pode enviar uma mensagem simples para um tópico com o seguinte comando:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   Em que:

   - aws-region é a região em que AWS o tópico está localizado.
   - sns-topic-arné o Amazon Resource Name (ARN) do tópico. Para obter o ARN no Console do Amazon SNS: escolha Topics (Tópicos), localize o tópico e encontre o valor na coluna ARN.

   Se a mensagem for recebida com êxito pelo Amazon SNS, o terminal imprimirá um ID de mensagem, como o seguinte:

   ```
   {
      "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
   }
   ```

Criar uma política de endpoint da Amazon VPC para o Amazon SNS

É possível criar uma política para endpoints da Amazon VPC para o Amazon SNS na qual se especifica o seguinte:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para obter mais informações, consulte Controlar o acesso a serviços com endpoint da VPCs no Manual do usuário da Amazon VPC.

O exemplo de política de VPC endpoint a seguir especifica que o usuário `MyUser` do IAM tem permissão para publicar no tópico `MyTopic` do Amazon SNS.

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

O seguinte é negado:

- Outras ações de API do Amazon SNS, como `sns:Subscribe` e `sns:Unsubscribe`.
- Outros usuários e regras do IAM que tentam usar esse endpoint da VPC.
- `MyUser` publicando em um tópico diferente do Amazon SNS.

> **Note**
>
> O usuário do IAM ainda pode usar outras ações de API do Amazon SNS de fora da VPC.

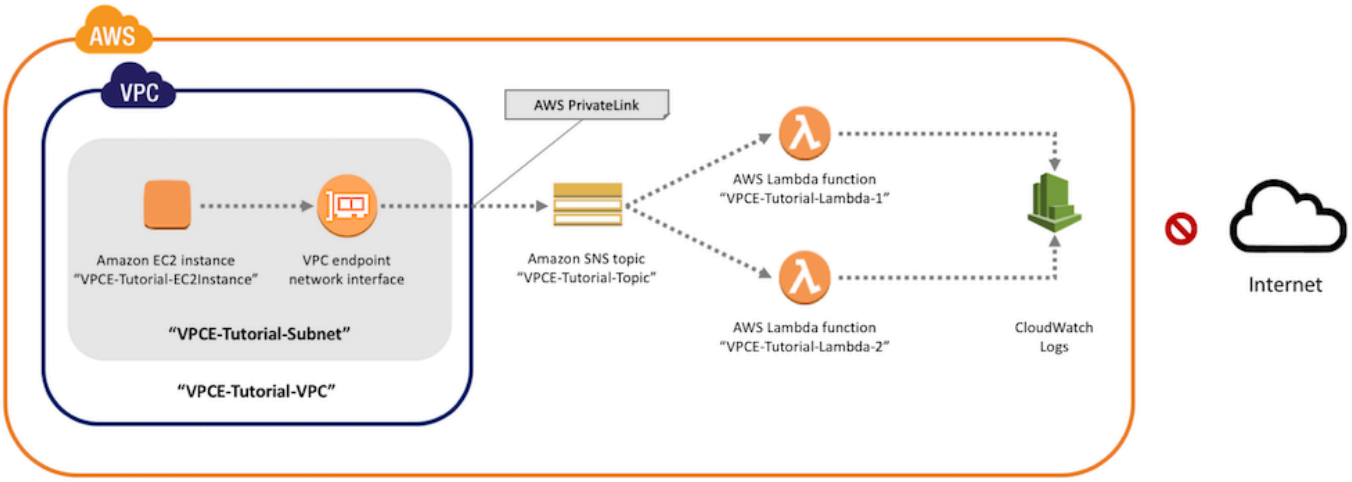
Publicar uma mensagem do Amazon SNS da Amazon VPC

Esta seção descreve como publicar em um tópico do Amazon SNS, mantendo as mensagens seguras em uma rede privada. Você publica uma mensagem de uma EC2 instância da Amazon que está hospedada na Amazon Virtual Private Cloud (Amazon VPC). A mensagem permanece na AWS rede sem viajar pela Internet pública. Ao publicar mensagens de forma privada a partir de uma VPC, você pode melhorar a segurança do tráfego entre seus aplicativos e o Amazon SNS. Essa segurança é importante ao publicar informações de identificação pessoal (PII) sobre seus clientes ou quando sua assinatura está sujeita a regulamentações de mercado. Por exemplo, publicar de maneira privada é útil se você tiver um sistema para área da saúde que precise estar em conformidade com a Lei de Portabilidade e Responsabilidade de Provedores de Saúde (HIPAA) ou um sistema financeiro que precise estar em conformidade com o Padrão de segurança de dados do setor de cartão de pagamento (Payment Card Industry Data Security Standard, PCI DSS).

As etapas gerais são as seguintes:

- Use um AWS CloudFormation modelo para criar automaticamente uma rede privada temporária no seu Conta da AWS.
- Crie um VPC endpoint que conecte a VPC com o Amazon SNS.
- Faça login em uma EC2 instância da Amazon e publique uma mensagem de forma privada em um tópico do Amazon SNS.
- Verifique se a mensagem foi entregue com sucesso.
- Exclua os recursos que você criou durante esse processo para que eles não permaneçam no seu Conta da AWS.

O diagrama a seguir mostra a rede privada que você cria em sua AWS conta ao concluir essas etapas:



Essa rede consiste em uma VPC que contém uma instância da Amazon EC2 . A instância se conecta ao Amazon SNS por meio de um endpoint da VPC de interface. Esse tipo de endpoint se conecta a serviços que são alimentados por AWS PrivateLink. Com essa conexão estabelecida, você pode fazer login na EC2 instância da Amazon e publicar mensagens no tópico do Amazon SNS, mesmo que a rede esteja desconectada da Internet pública. O tópico distribui as mensagens que recebe em duas AWS Lambda funções de assinatura. Essas funções registram as mensagens que recebem no Amazon CloudWatch Logs.

Demora cerca de 20 minutos para concluir essas etapas.

Tópicos

Antes de começar

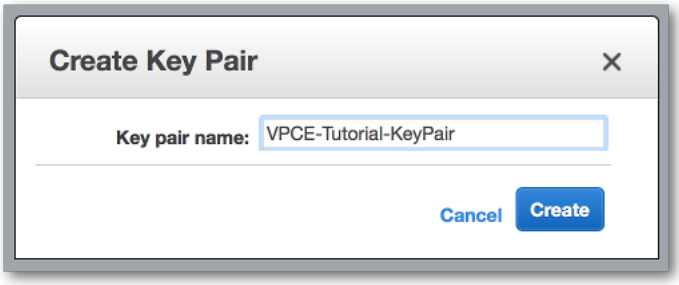
Antes de começar, você precisa de uma conta da Amazon Web Services (AWS). Quando você se inscreve, sua conta é automaticamente cadastrada em todos os serviços AWS, incluindo Amazon SNS e Amazon VPC. Caso ainda não tenha criado uma conta, acesse https://aws.amazon.com/ e escolha Create a Free Account (Criar conta gratuita).

Etapa 1: criar um par de EC2 chaves da Amazon

Um par de chaves é usado para fazer login em uma EC2 instância da Amazon. Esse par consiste em uma chave pública usada para criptografar suas informações de login e uma chave privada usada para descriptografá-la. Ao criar um par de chaves, você faz download de uma cópia da chave privada. Posteriormente, você usa o par de chaves para fazer login em uma EC2 instância da Amazon. Para efetuar login, especifique o nome do par de chaves e insira a chave privada.

Para criar o par de chaves

1. Faça login no AWS Management Console e abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. No menu de navegação à esquerda, localize a seção Rede e segurança. Em seguida, escolha Pares de chave).
3. Escolha Criar par de chaves.
4. Na janela Criar par de chaves, em Nome do par de chaves, digite **`VPCE-Tutorial-KeyPair`**. Escolha Criar.



5. O arquivo de chave privada é baixado automaticamente pelo navegador. Salve-o em um local seguro. A Amazon EC2 fornece ao arquivo uma extensão de `.pem`.
6. (Opcional) Se estiver usando um cliente de SSH em um computador Mac ou Linux para se conectar à sua instância, use o seguinte comando `chmod` para definir as permissões do arquivo de chave privada, de maneira que apenas você possa lê-lo:

   a. Abra um terminal e navegue até o diretório que contém a chave privada:

```
$ cd /filepath_to_private_key/
```

b. Defina as permissões usando o seguinte comando:

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

Etapa 2: criar os AWS recursos

Para configurar a infraestrutura, você usa um AWS CloudFormation modelo. Um modelo é um arquivo que atua como um modelo para criar AWS recursos, como EC2 instâncias da Amazon e tópicos do Amazon SNS. O modelo para esse processo é fornecido GitHub para você baixar.

Você fornece o modelo e AWS CloudFormation provisiona os recursos de que precisa como uma pilha em seu Conta da AWS. AWS CloudFormation Uma pilha é um conjunto de recursos que pode gerenciar como uma unidade. Ao concluir essas etapas, você pode usar AWS CloudFormation para excluir todos os recursos da pilha de uma só vez. Esses recursos não permanecem em seu Conta da AWS, a menos que você queira.

A pilha deste processo inclui os seguintes recursos:

- Uma VPC e os recursos de rede associados, incluindo uma sub-rede, um grupo de segurança, um gateway da Internet e uma tabela de rotas.
- Uma EC2 instância da Amazon que é iniciada na sub-rede na VPC.
- Um tópico do Amazon SNS.
- Duas AWS Lambda funções. Essas funções recebem mensagens que são publicadas no tópico do Amazon SNS e registram eventos em CloudWatch Logs.
- CloudWatch Métricas e registros da Amazon.
- Uma função do IAM que permite que a EC2 instância da Amazon use o Amazon SNS e uma função do IAM que permite que as funções do Lambda gravem nos registros. CloudWatch

Para criar os AWS recursos

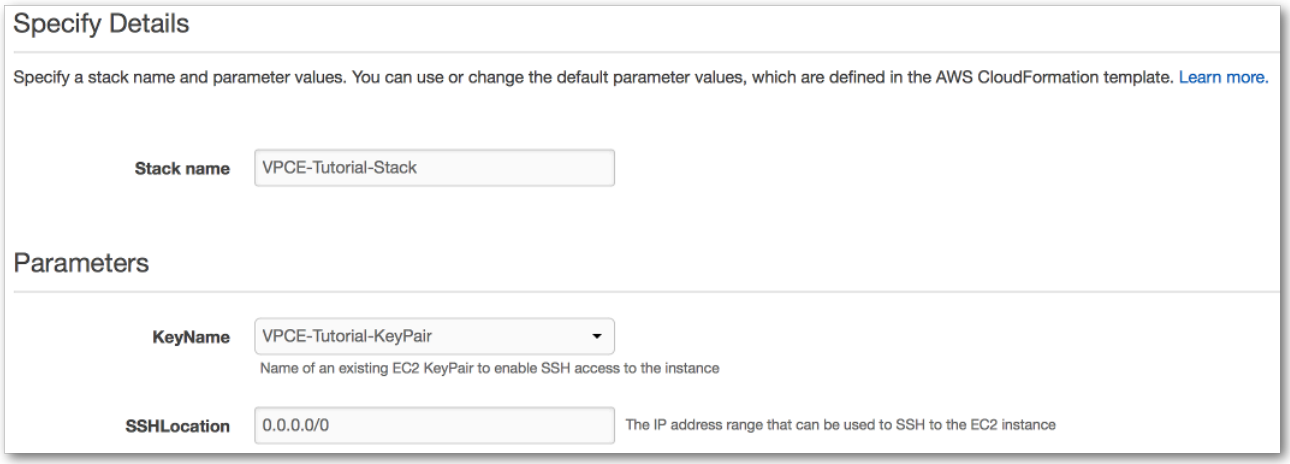
1. Baixe o arquivo de modelo do GitHub site.
2. Faça login no console do AWS CloudFormation.
3. Escolha Create Stack (Criar pilha).

4. Na página Select Template (Selecionar modelo), escolha Upload a template to Amazon S3 (Carregar um modelo no Amazon S3) e, em seguida, o arquivo e Next (Avançar).
5. Na página Especificar detalhes, especifique os nomes de pilha e chave:

   a. Para Nome da pilha, digite **`VPCE-Tutorial-Stack`**.

   b. Para KeyName, escolha VPCE-Tutorial -. KeyPair

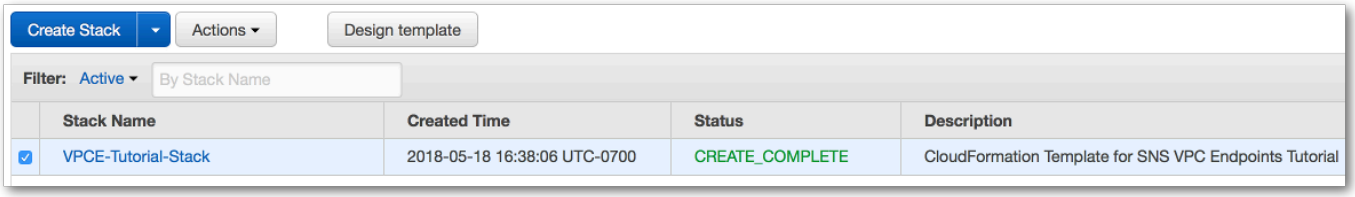
   c. Para SSHLocation, mantenha o valor padrão de**`0.0.0.0/0`**.



   d. Escolha Próximo.

6. Na página Opções, mantenha todos os valores padrão e escolha Próximo.
7. Na página Revisar, verifique os detalhes da pilha.
8. Em Capacidades, reconheça que isso AWS CloudFormation pode criar recursos do IAM com nomes personalizados.
9. Escolha Criar.

   O AWS CloudFormation console abre a página Stacks. O VPCE-Tutorial-Stack tem um status de CREATE_IN_PROGRESS. Em instantes, após a conclusão do processo de criação, o status muda para CREATE_COMPLETE.



> **Tip**
>
> Escolha o botão Atualizar para ver o status mais recente da pilha.

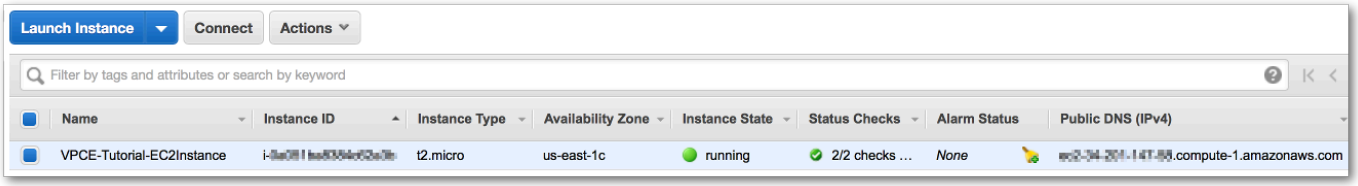
Etapa 3: confirme se sua EC2 instância da Amazon não tem acesso à Internet

A EC2 instância da Amazon que foi lançada em sua VPC na etapa anterior não tem acesso à Internet. Ela não permite o tráfego de saída e não pode publicar mensagens no Amazon SNS. Para verificar isso, faça login na instância. Em seguida, conecte-se a um endpoint público e tente enviar uma mensagem Amazon SNS.

Neste ponto do tutorial, a tentativa de publicação falha. Em uma etapa posterior, depois de criar um VPC endpoint para o Amazon SNS, sua tentativa de publicação será bem-sucedida.

Para se conectar à sua EC2 instância da Amazon

1. Abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. No menu de navegação à esquerda, localize a seção Instâncias. Em seguida, selecione Instâncias.
3. Na lista de instâncias, selecione VPCE-. Tutorial-EC2Instance
4. Copie o nome do host fornecido na coluna DNS público.



5. Abra um terminal. No diretório que contém o key pair, conecte-se à instância usando o comando a seguir, onde *instance-hostname* está o nome do host que você copiou do console da Amazon EC2 :

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

Para verificar se a instância não tem conectividade com a Internet

- No seu terminal, tente se conectar a qualquer endpoint público, como amazon.com:

```
$ ping amazon.com
```

Como a tentativa de conexão falha, você pode cancelar a qualquer momento (Ctrl + C no Windows ou Command + C no macOS).

Para verificar se a instância não tem conectividade com o Amazon SNS

1. Faça login no console do Amazon SNS.
2. No menu de navegação à esquerda, escolha Tópicos.
3. Na página Tópicos, copie o nome do recurso da Amazon (ARN) para o tópico VPCE-Tutorial-Topic.
4. No seu terminal, tente publicar uma mensagem no tópico:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Como a tentativa de publicação falha, você pode cancelar a qualquer momento.

Etapa 4: criar um endpoint da VPC para Amazon SNS

Para conectar a VPC ao Amazon SNS, você define um VPC endpoint de interface. Depois de adicionar o endpoint, você pode fazer login na EC2 instância da Amazon em sua VPC e, a partir daí, usar a API do Amazon SNS. Você pode publicar mensagens no tópico. Elas serão publicadas de maneira privada. Eles permanecem na AWS rede e não navegam pela Internet pública.

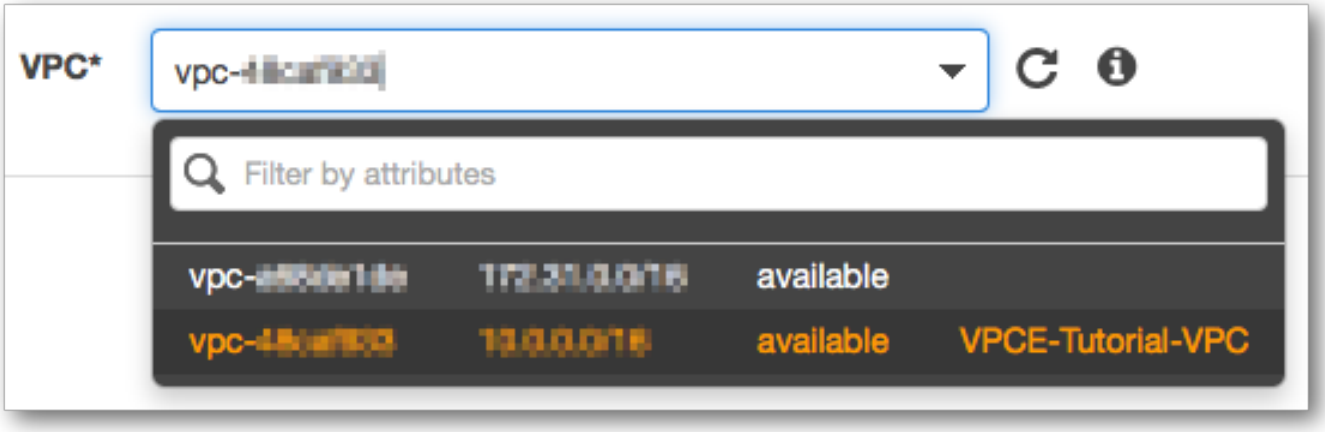
> **Note**
>
> A instância ainda não tem acesso a outros AWS serviços e endpoints na Internet.

Para criar o endpoint

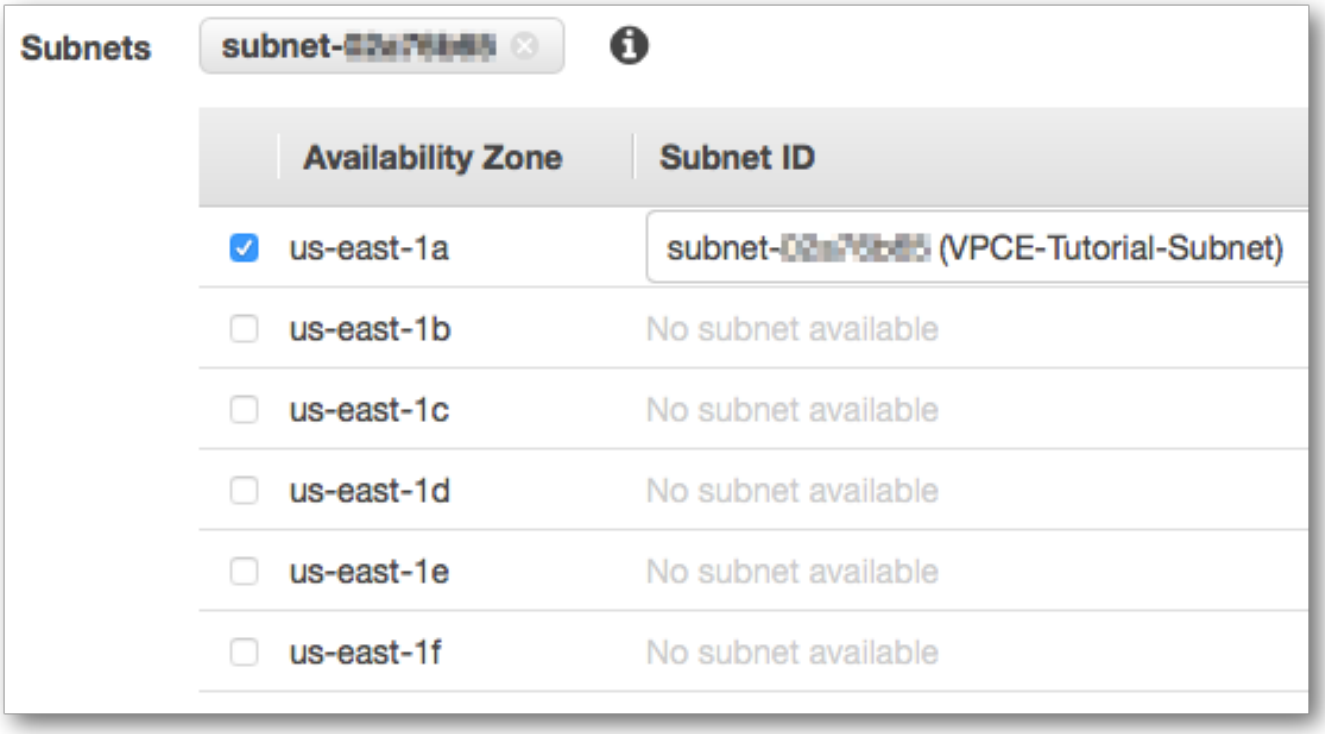
1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No menu de navegação à esquerda, escolha Endpoints.
3. Escolha Criar endpoint.

4. Na página Criar endpoint, em Categoria de serviço, mantenha o valor padrão de Serviços da AWS .
5. Em Service Name (Nome do serviço), escolha o nome do serviço para o Amazon SNS.

   Os nomes de serviços variam de acordo com a região escolhida. Por exemplo, se você escolher Leste dos EUA (Norte da Virgínia), o nome do serviço será com.amazonaws. ***us-east-1***.sns.
6. Para VPC, escolha a VPC com nome VPCE-Tutorial-VPC.

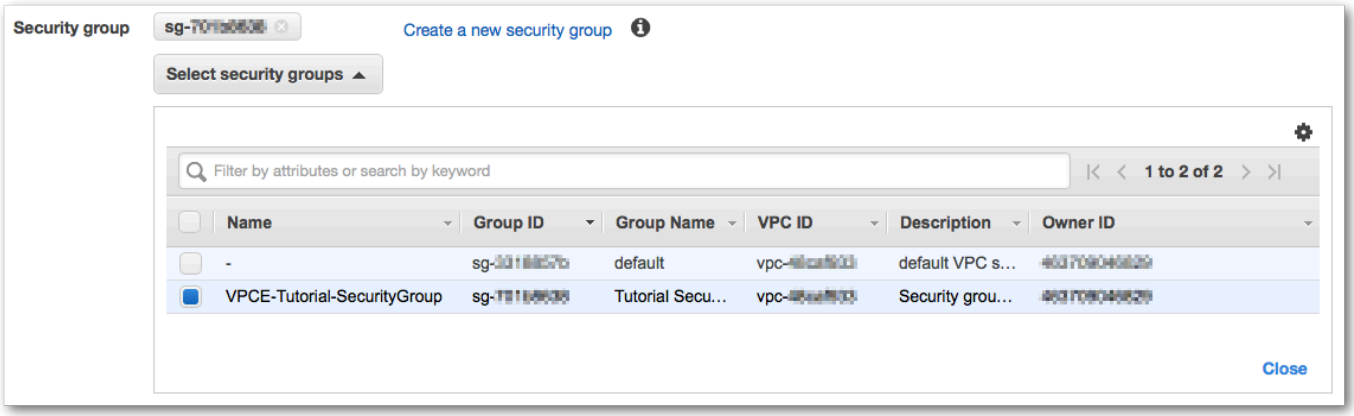


7. Em Sub-redes, selecione aquela que contém VPCE-Tutorial-Subnet no ID de sub-rede.

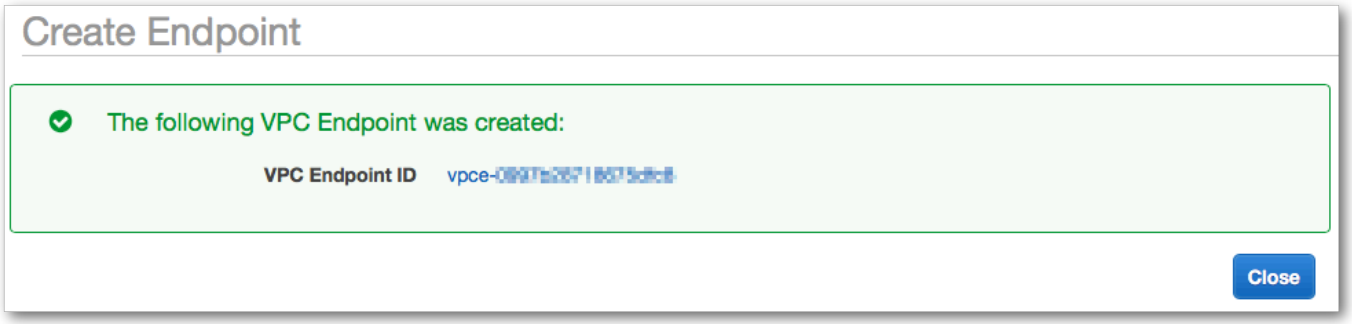


8. Em Ativar nome DNS privado, selecione Ativar para este terminal.

9. Em Grupo de segurança, escolha Selecionar grupo de segurança e escolha VPCE-Tutorial -. SecurityGroup

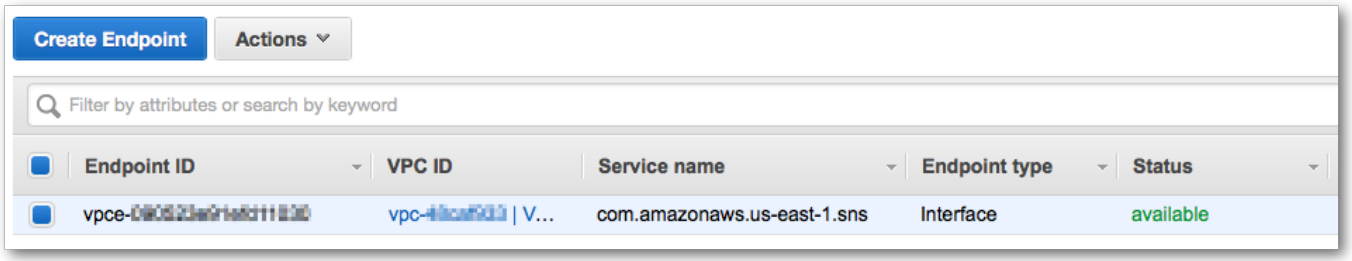


10. Escolha Criar endpoint. O console da Amazon VPC verifica se o endpoint da VPC foi criado.



11. Escolha Fechar.

    O console da Amazon VPC abre a página Endpoints. O novo endpoint tem um status pendente. Em instantes, após a conclusão do processo de criação, o status muda para disponível.



Etapa 5: publicar uma mensagem no seu tópico do Amazon SNS

Agora que sua VPC inclui um endpoint para o Amazon SNS, você pode fazer login na EC2 instância da Amazon e publicar mensagens no tópico.

Para publicar uma mensagem

1. Se o seu terminal não estiver mais conectado à sua EC2 instância da Amazon, conecte-se novamente:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Execute o mesmo comando que você fez anteriormente para publicar uma mensagem no seu tópico do Amazon SNS. Desta vez, a tentativa de publicação será bem-sucedida e o Amazon SNS retornará um ID de mensagem:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"


{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

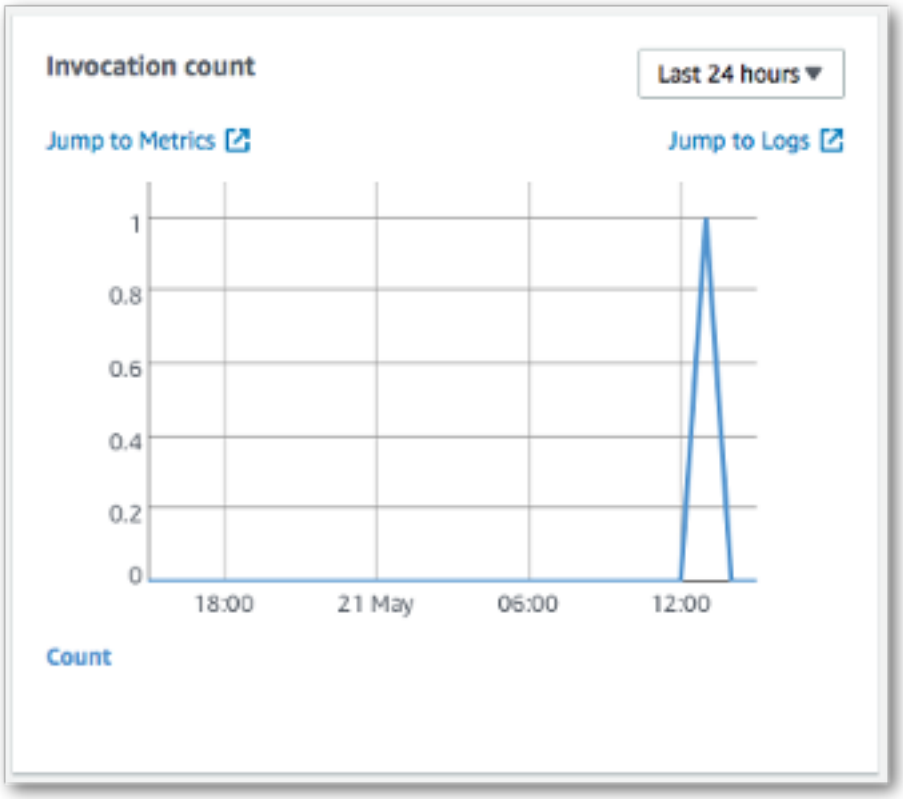
Etapa 6: verificar as entregas de mensagens

Quando o tópico do Amazon SNS receber uma mensagem, ele enviará a mensagem para as duas funções assinantes do Lambda. Quando essas funções recebem a mensagem, elas registram o evento nos CloudWatch registros. Para verificar se a entrega da mensagem foi bem-sucedida, verifique se as funções foram invocadas e se os CloudWatch registros foram atualizados.

Para verificar se as funções do Lambda foram invocadas

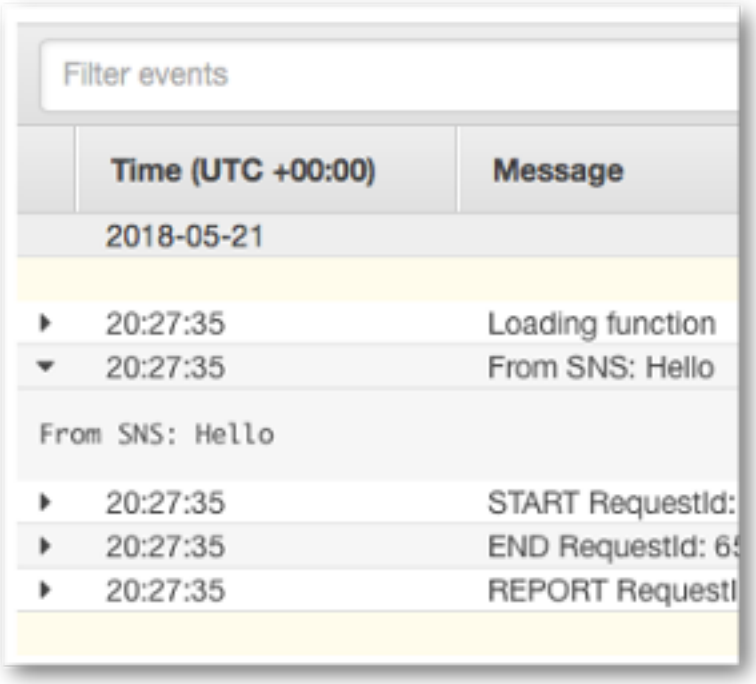
1. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Na página Funções, escolha VPCE-Tutorial-Lambda-1.
3. Escolha Monitoramento.
4. Verifique o gráfico Contagem de invocação. Este gráfico mostra o número de vezes que a função do Lambda foi executada.

   A contagem de invocação corresponde ao número de vezes que você publicou uma mensagem no tópico.



Para verificar se os CloudWatch registros foram atualizados

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No menu de navegação à esquerda, escolha Logs.
3. Verifique os registros que foram escritos pelas funções do Lambda:
   a. Escolha o grupo de registros/aws/lambda/VPCE-Tutorial-Lambda-1/.
   b. Escolha o fluxo de logs.
   c. Verifique se o log inclui a entrada `From SNS: Hello`.



d. Escolha Grupos de logs na parte superior do console para retornar para a página Grupos de logs. Em seguida, repita as etapas anteriores para o grupo de registros the /aws/lambda/ VPCE -Tutorial-Lambda-2/.

Parabéns! Ao adicionar um endpoint para o Amazon SNS a uma VPC, você conseguiu publicar uma mensagem em um tópico de dentro da rede gerenciada pela VPC. A mensagem foi publicada de maneira privada sem ser exposta à Internet pública.

Etapa 7: limpar

A menos que você queira manter os recursos criados, você poderá excluí-los agora. Ao excluir AWS recursos que você não está mais usando, você evita cobranças desnecessárias no seu Conta da AWS.

Primeiro, exclua seu endpoint da VPC usando o console da Amazon VPC. Em seguida, exclua os outros recursos que você criou excluindo a pilha no AWS CloudFormation console. Quando você exclui uma pilha, AWS CloudFormation remove os recursos da pilha do seu. Conta da AWS

Para excluir seu VPC endpoint

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No menu de navegação à esquerda, escolha Endpoints.

3. Selecione o endpoint que você criou.
4. Escolha Ações e escolha Excluir endpoint.
5. Na janela Excluir endpoint, escolha Sim, excluir.

   O status do endpoint muda para excluindo. Quando a exclusão é concluída, o endpoint é removido da página.

Para excluir sua AWS CloudFormation pilha

1. Abra o AWS CloudFormation console em https://console.aws.amazon.com/cloudformation.
2. Selecione a pilha VPCE-Tutorial-Stack.
3. Escolha Ações e, em seguida, escolha Excluir pilha.
4. Na janela Excluir pilha, escolha Sim, excluir.

   O status da pilha muda para DELETE_IN_PROGRESS. Quando a exclusão é concluída, a pilha é removida da página.

Recursos relacionados

Para obter mais informações, consulte os recursos a seguir.

- AWS Blog de segurança: Protegendo mensagens publicadas no Amazon SNS com AWS PrivateLink
- O que é Amazon VPC?
- VPC Endpoints
- O que é a Amazon EC2?
- Conceitos do AWS CloudFormation

Aprimorar a segurança do Amazon SNS com a proteção de dados de mensagens

- Proteção de dados de mensagens é um recurso do Amazon SNS usado para definir regras e políticas próprias para auditar e controlar o conteúdo de dados em movimento, em vez de dados em repouso.

- A proteção de dados de mensagens oferece serviços de governança, conformidade e auditoria para aplicações corporativas centradas em mensagens, para que a entrada e a saída de dados possam ser controladas pelo proprietário do tópico do Amazon SNS e os fluxos de conteúdo possam ser rastreados e registrados em log.
- Você pode criar regras de governança baseadas em carga útil para impedir a entrada de conteúdo não autorizado em seus fluxos de mensagens.
- Você pode conceder diferentes permissões de acesso ao conteúdo para assinantes individuais e auditar todo o processo de fluxo de conteúdo.

Gerenciamento de identidade e acesso no Amazon SNS

O acesso ao Amazon SNS requer credenciais que AWS possam ser usadas para autenticar suas solicitações. Essas credenciais devem ter permissões para acessar AWS recursos, como tópicos e mensagens do Amazon SNS. As seções a seguir fornecem detalhes sobre como é possível usar o AWS Identity and Access Management (IAM) e o Amazon SNS para ajudar a proteger seus recursos controlando quem pode acessá-los.

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para utilizar os recursos do Amazon SNS. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Amazon SNS.

Usuário do serviço: se você usar o serviço do Amazon SNS para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões de que você precisa. À medida que mais recursos do Amazon SNS forem usados para realizar o trabalho, talvez sejam necessárias permissões adicionais. Compreenda como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não puder acessar um recurso no Amazon SNS, consulte Resolução de problemas de identidade e acesso do Amazon Simple Notification Service.

Administrador do serviço: se você for o responsável pelos recursos do Amazon SNS em sua empresa, provavelmente terá acesso total ao Amazon SNS. Cabe a você determinar quais funcionalidades e recursos do Amazon SNS os usuários do seu serviço devem acessar. Envie as

solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como a empresa pode usar o IAM com o Amazon SNS, consulte Como o Amazon SNS funciona com o IAM.

Administrador do IAM: se você é um administrador do IAM, talvez queira saber detalhes sobre como pode escrever políticas para gerenciar o acesso ao Amazon SNS. Para visualizar exemplos de políticas baseadas em identidade do Amazon SNS que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade do Amazon Simple Notification Service.

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como usuário Conta da AWS raiz, usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte Versão 4 do AWS Signature para solicitações de API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser necessário fornecer informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia do usuário do AWS IAM Identity Center e Usar a autenticação multifator da AWS no IAM no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário-raiz para tarefas diárias. Proteja as credenciais do usuário-raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário-raiz, consulte Tarefas que exigem credenciais de usuário-raiz no Guia do Usuário do IAM.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade. Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, é recomendável usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte O que é o Centro de Identidade do IAM? no Guia do Usuário do AWS IAM Identity Center .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, é recomendável contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte Alternar as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários

de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Casos de uso para usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Para assumir temporariamente uma função do IAM no AWS Management Console, você pode alternar de um usuário para uma função do IAM (console). Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para usar perfis, consulte Métodos para assumir um perfil no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para ter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidade de terceiros (federação) no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Guia do Usuário do AWS IAM Identity Center .
- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
  - Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir o perfil para executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados a serviço.
- Aplicativos em execução na Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo solicitações AWS CLI de AWS API. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.

- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada usuário Conta da AWS root. Para obter mais informações sobre Organizations e SCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.
- Políticas de controle de recursos (RCPs) — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo o usuário Conta da AWS raiz, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Controle de acesso

O Amazon SNS tem seu próprio sistema de permissões baseado em recursos que usa políticas escritas na mesma linguagem usada para políticas AWS Identity and Access Management (IAM). Isso significa que você pode atingir objetivos semelhantes com as políticas do IAM e do Amazon SNS.

> **Note**
>
> É importante entender que todos Contas da AWS podem delegar suas permissões aos usuários em suas contas. O acesso entre contas permite compartilhar o acesso a seus recursos da AWS sem a necessidade de gerenciar usuários adicionais. Para obter informações sobre como usar o acesso entre contas, consulte Habilitar o acesso entre contas no Guia do usuário do IAM.

Casos de exemplo para controle de acesso do Amazon SNS

Você tem muita flexibilidade com relação à forma de conceder ou negar acesso a um recurso. No entanto, os casos de uso típicos são bastante simples:

- Você deseja conceder a outro Conta da AWS um tipo específico de ação de tópico (por exemplo, Publicar). Para obter mais informações, consulte Conceder Conta da AWS acesso a um tópico.
- Você deseja limitar as assinaturas do tópico apenas ao protocolo HTTPS. Para obter mais informações, consulte Limitar assinaturas para HTTPS.
- Você deseja permitir que o Amazon SNS publique mensagens para a fila do Amazon SQS. Para obter mais informações, consulte Publicar mensagens em uma fila do Amazon SQS.

Conceitos chave de política de acesso do Amazon SNS

As seções a seguir descrevem os conceitos que você precisa entender para usar a linguagem de políticas de acesso. Eles são apresentados em uma ordem lógica, com os primeiros termos que você precisa saber na parte superior da lista.

Permissão

Permissão refere-se ao conceito de conceder ou negar algum tipo de acesso a um recurso específico. Basicamente, as permissões seguem este formato: “A tem/não tem permissão para realizar B para C, quando D se aplicar”. Por exemplo, Jane (A) tem permissão para publicar (B) em TopicA (C) desde que ela utilize o protocolo HTTP (D). Sempre que Jane fizer uma publicação em TopicA, o serviço verificará se ela tem permissão e se a solicitação está de acordo com as condições definidas na permissão.

Declaração

Declaração é a descrição formal de uma única permissão, criada na linguagem de políticas de acesso. Você sempre cria uma declaração como parte de um documento de contêiner mais abrangente conhecido como política (consulte o próximo conceito).

Política

Política é um documento (escrito na linguagem de políticas de acesso) que funciona como um contêiner para uma ou mais declarações. Por exemplo, uma política poderia ter duas declarações: uma que afirme que Jane pode se inscrever usando o protocolo de e-mail, e outra afirmando que Bob não pode fazer uma publicação no tópico A. Como mostrado na figura a seguir, um cenário equivalente seria ter duas políticas, uma que afirme que Jane pode se inscrever usando o protocolo de e-mail e outra afirmando que Bob não pode publicar no tópico A.

Somente caracteres ASCII são permitidos em documentos de política. Você pode utilizar `aws:SourceAccount` e `aws:SourceOwner` contornar o cenário em que precisa conectar outros AWS serviços ARNs que contenham caracteres não ASCII. Veja a diferença entre `aws:SourceAccount` versus `aws:SourceOwner`.

Emissor

Emissor é a pessoa que cria uma política para conceder permissões a um recurso. O emissor (por definição) é sempre o proprietário do recurso. AWS não permite que os usuários do AWS serviço criem políticas para recursos que não possuem. Se John for o proprietário do recurso, AWS autentica a identidade de John ao enviar a política que ele escreveu para conceder permissões para esse recurso.

Entidade principal

Entidade principal é a pessoa ou as pessoas que recebem a permissão na política. A entidade principal é A na declaração “A tem permissão para realizar B para C, quando D se aplicar”. Em uma política, você pode definir a entidade principal como “ninguém” (ou seja, pode especificar um curinga para representar todas as pessoas). Você poderá fazer isso, por exemplo, se não quiser restringir o acesso com base na identidade real do solicitante, mas em outras características de identificação, como o endereço IP do solicitante.

Ação

Ação é a atividade que a entidade principal tem permissão para executar. A ação é B na declaração “A tem permissão para realizar B para C, quando D se aplicar”. Normalmente, a ação é apenas a operação na solicitação para AWS. Por exemplo, Jane envia uma solicitação ao Amazon SNS com `Action=Subscribe`. Você pode especificar uma ou várias ações em uma política.

Recurso

Recurso é o objeto ao qual a entidade principal está solicitando acesso. O recurso é C na declaração “A tem permissão para realizar B para C, quando D se aplicar”.

Condições e chaves

Condições são todas as restrições ou detalhes sobre a permissão. A condição é D na declaração “A tem permissão para realizar B para C, quando D se aplicar”. A parte da política que especifica as condições pode ser a mais detalhada e complexa de todas as partes. As condições comuns são relacionadas a:

- Data e hora (por exemplo, a solicitação precisa chegar antes de um dia específico)
- Endereço IP (por exemplo, o endereço IP do solicitante precisa ser fazer parte de determinado intervalo CIDR)

Chave é a característica específica que fundamenta a restrição de acesso. Por exemplo, a data e a hora da solicitação.

Você usa condições e chaves em conjunto para expressar a restrição. A maneira mais fácil de entender como você realmente implementa uma restrição é com um exemplo: se você deseja restringir o acesso para antes de 30 de maio de 2010, deve usar a condição chamada `DateLessThan`. Você usa a chave chamada `aws:CurrentTime` e a define com o valor

`2010-05-30T00:00:00Z`. A AWS define as condições e chaves que você pode usar. O AWS serviço em si (por exemplo, Amazon SQS ou Amazon SNS) também pode definir chaves específicas do serviço. Para obter mais informações, consulte Permissões da API do Amazon SNS: referência de ações e recursos.

Solicitante

O solicitante é a pessoa que envia uma solicitação a um AWS serviço e solicita acesso a um recurso específico. O solicitante envia uma solicitação AWS que basicamente diz: “Você vai me permitir fazer B a C onde D se aplica?”

Avaliação

A avaliação é o processo que o AWS serviço usa para determinar se uma solicitação recebida deve ser negada ou permitida com base nas políticas aplicáveis. Para obter informações sobre a lógica de avaliação, consulte Lógica de avaliação.

Efeito

Efeito é o resultado que você deseja que uma declaração de política retorne no tempo da avaliação. Você especifica esse valor ao criar as declarações em uma política, e os valores possíveis são negar e permitir.

Por exemplo, você pode criar uma política que tenha uma declaração que nega todas as solicitações originárias da Antártica (efeito=negar considerando que a solicitação usa um endereço IP alocado para a Antártica). Como alternativa, você pode escrever uma política que tenha uma declaração que permita todas as solicitações que não são originárias da Antártica (effect=allow, considerando que a solicitação não é originária da Antártica). Embora as duas declarações pareçam realizar a mesma ação, na lógica da linguagem de políticas de acesso, elas são diferentes. Para obter mais informações, consulte Lógica de avaliação.

Embora haja apenas dois valores possíveis que você pode especificar para o efeito (permitir ou negar), pode haver três resultados diferentes no tempo de avaliação da política: negação padrão, permitir ou negação explícita. Para obter mais informações, consulte os seguintes conceitos e Lógica de avaliação.

Negação padrão

Negação padrão é o resultado padrão de uma política na ausência de permitir ou negação explícita.

Permitir

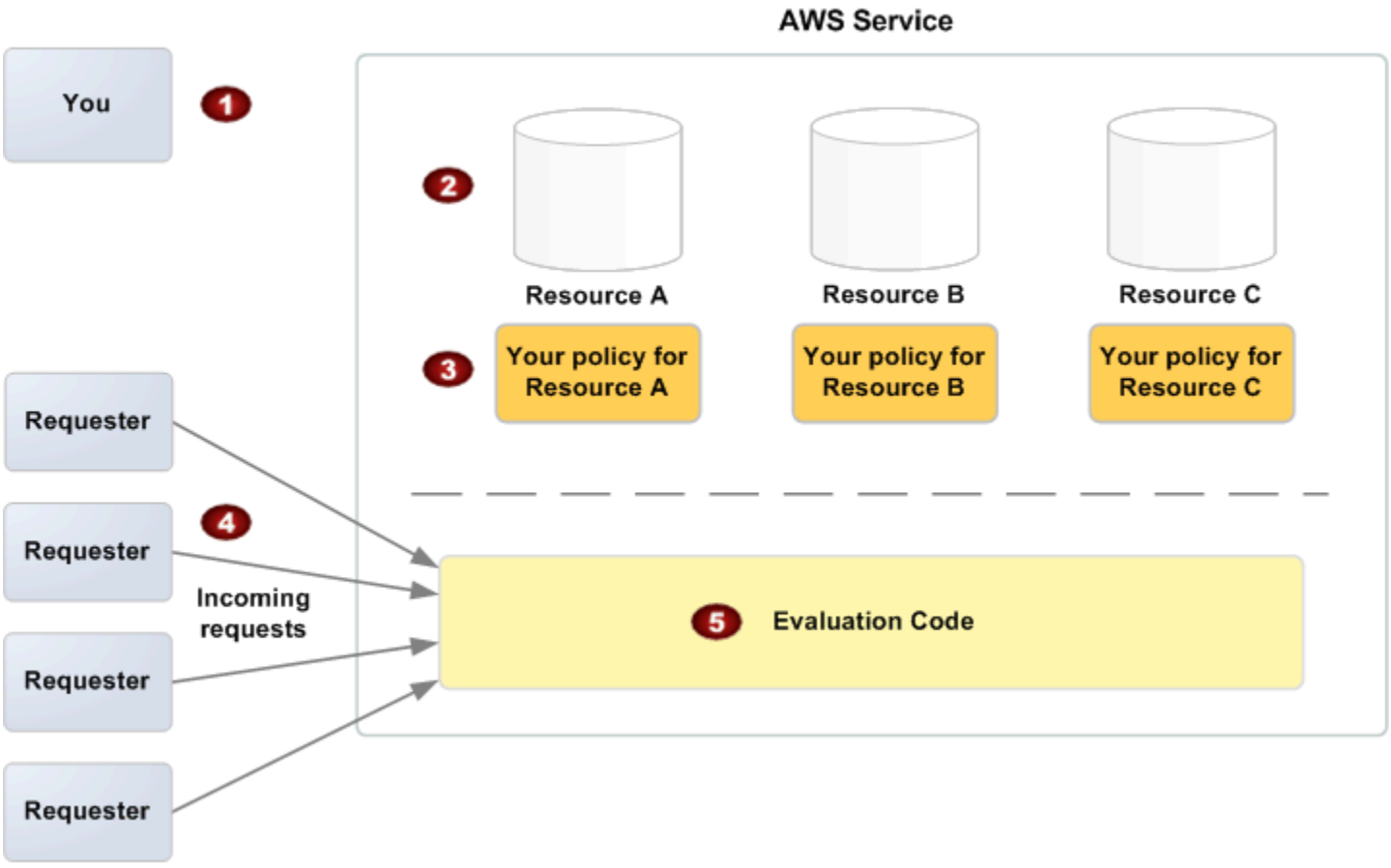
Permitir resulta de uma declaração que tem efeito = permitir, supondo que todas as condições indicadas foram atendidas. Exemplo: permitir solicitações se elas forem recebidas antes de 13h00 em 30 de abril de 2010. Um permitir substitui todas as negações padrão, mas nunca uma negação explícita.

Negação explícita

A negação explícita resulta de uma declaração que tem efeito = negar, supondo que todas as condições indicadas foram atendidas. Exemplo: negar todas as solicitações originárias da Antártica. Todas as solicitações originárias da Antártica sempre serão negadas independentemente do que qualquer outra política possa permitir.

Visão geral da arquitetura do controle de acesso do Amazon SNS

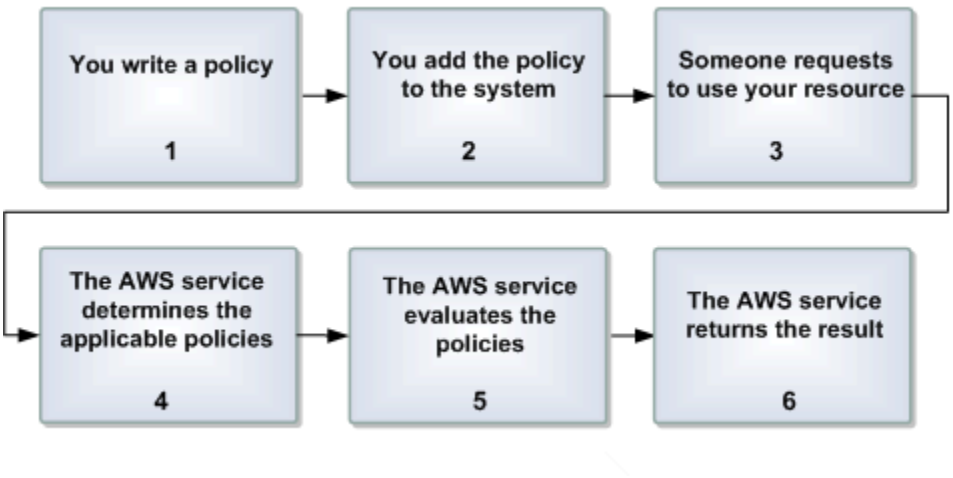
A figura e a tabela a seguir descrevem os principais componentes que interagem para fornecer controle de acesso aos seus recursos.



1	Você, o proprietário do recurso.
2	Seus recursos (contidos no AWS serviço; por exemplo, filas do Amazon SQS).
3	Suas políticas. Normalmente, você tem uma única política por recurso, embora seja possível ter várias. O AWS serviço em si fornece uma API que você usa para carregar e gerenciar suas políticas.
4	Solicitantes e suas solicitações recebidas no serviço. AWS
5	O código de avaliação da linguagem de políticas de acesso. Esse é o conjunto de códigos dentro do AWS serviço que avalia as solicitações recebidas em relação às políticas aplicáveis e determina se o solicitante tem permissão para acessar o recurso. Para obter informações sobre como o serviço toma decisões, consulte Lógica de avaliação.

Utilização da linguagem de políticas de acesso no Amazon SNS

A figura e a tabela a seguir descrevem o processo geral de como o controle de acesso funciona com a linguagem de políticas de acesso.



Processo para usar o controle de acesso com a linguagem de políticas de acesso

1	Você cria uma política para seu recurso.

	Por exemplo, você pode criar uma política para especificar permissões para seus tópicos do Amazon SNS.
2	Você carrega sua política para AWS. O AWS serviço em si fornece uma API que você usa para carregar suas políticas. Por exemplo, você pode usar a ação `SetTopicAttributes` do Amazon SNS com a finalidade de carregar uma política para determinado tópico do Amazon SNS.
3	Alguém envia uma solicitação para usar seu recurso. Por exemplo, um usuário envia uma solicitação ao Amazon SNS para usar um de seus tópicos.
4	O AWS serviço determina quais políticas são aplicáveis à solicitação. Por exemplo, o Amazon SNS verifica todas as políticas disponíveis do Amazon SNS e determina quais são aplicáveis (com base no recurso, no solicitante etc.).
5	O AWS serviço avalia as políticas. Por exemplo, o Amazon SNS avalia as políticas e determina se o solicitante tem permissão para usar o tópico ou não. Para obter informações sobre a lógica de decisão, consulte Lógica de avaliação.
6	O AWS serviço nega a solicitação ou continua a processá-la. Por exemplo, com base no resultado de avaliação da política, o serviço retorna um erro de “acesso negado” para o solicitante ou continua a processar a solicitação.

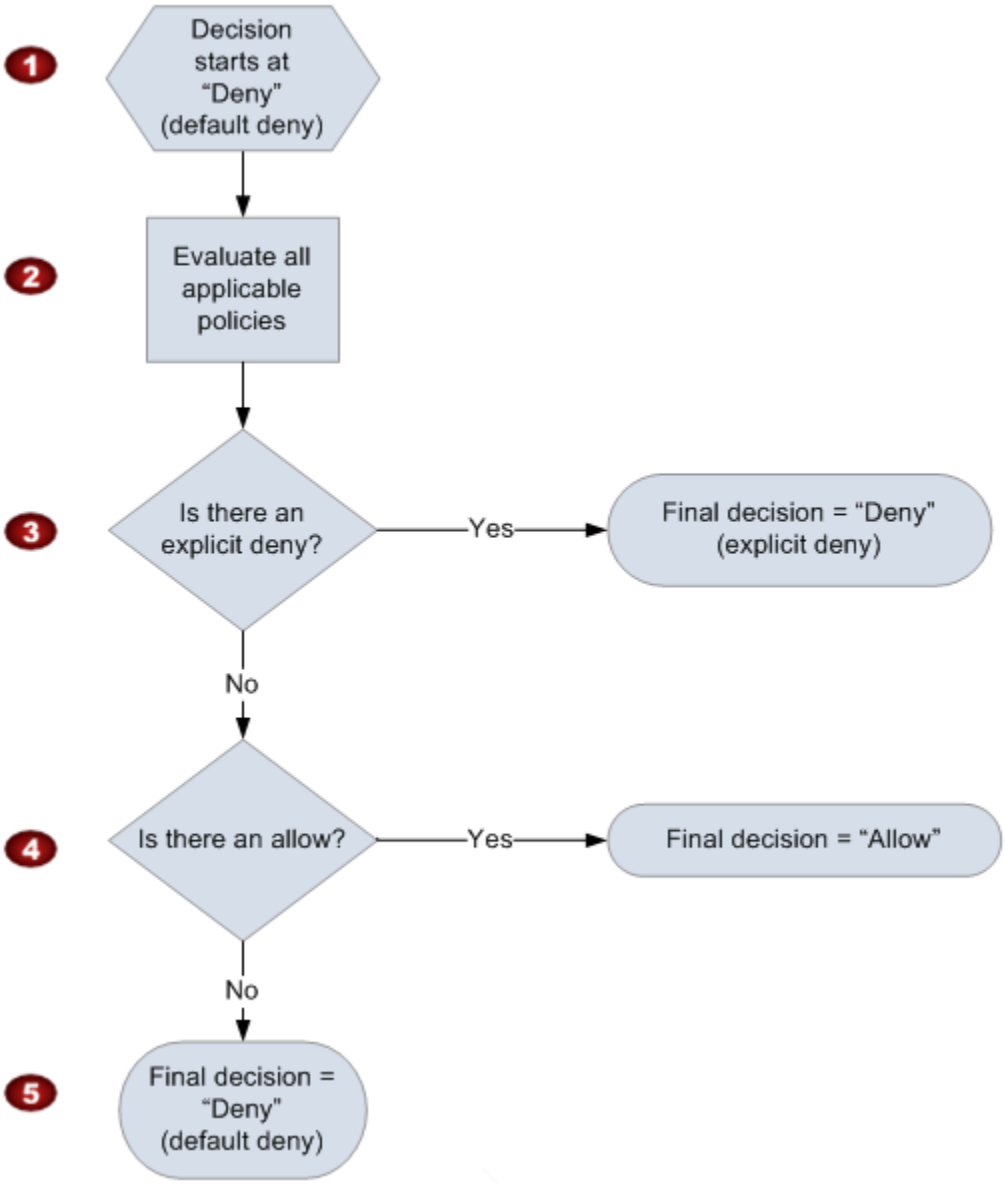
Lógica de avaliação

No momento da avaliação, o objetivo é decidir se uma solicitação de concessão deve ser permitida ou negada. A lógica de avaliação segue várias regras básicas:

- Por padrão, todas as solicitações para usar o recurso que venham de outras pessoas, e não de você, serão negadas.
- Um valor permitir substitui qualquer negação padrão.
- Uma negação explícita substitui todas as permissões.

- A ordem em que as políticas são avaliadas não é importante.

O seguinte fluxograma e discussão descrevem em mais detalhes como a decisão é tomada.



1	A decisão começa com uma negação padrão.
2	O código de aplicação avalia todas as políticas aplicáveis à solicitação (com base no recurso, na entidade principal, na ação e nas condições). A ordem em que o código de aplicação avalia as políticas não é importante.

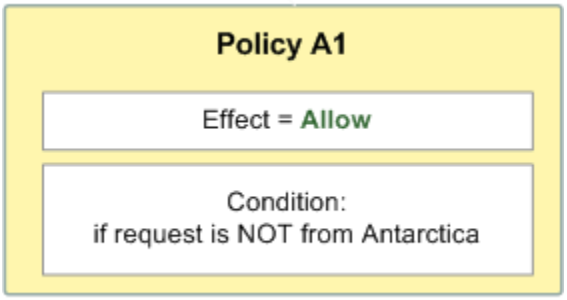
3	Em todas essas políticas, o código de aplicação procura uma instrução de negação explícita que se aplica à solicitação. Se o código de aplicação encontrar uma instrução desse tipo, ele retornará a decisão de “negar”, e o processo será concluído (essa é uma negação explícita; para obter mais informações, consulte Negação explícita).
4	Se nenhuma negação explícita for encontrada, o código de aplicação procurará qualquer instrução “permitir” aplicável à solicitação. Se encontrar uma instrução “permitir”, o código de aplicação retornará a decisão de “permitir”, e o processo será concluído (o serviço continuará a processar a solicitação).
5	Se nenhuma instrução de permitir for encontrada, a decisão final será “negar”, porque não havia negação explícita, ou permitir, porque é considerada uma negação padrão (para obter mais informações, consulte Negação padrão).

Interação entre negações explícitas e padrão

Uma política resulta em uma negação padrão caso não se aplique diretamente à solicitação. Por exemplo, se um usuário solicitar o uso do Amazon SNS, mas a política sobre o tópico não se referir nem um pouco à do Conta da AWS usuário, essa política resultará em uma negação padrão.

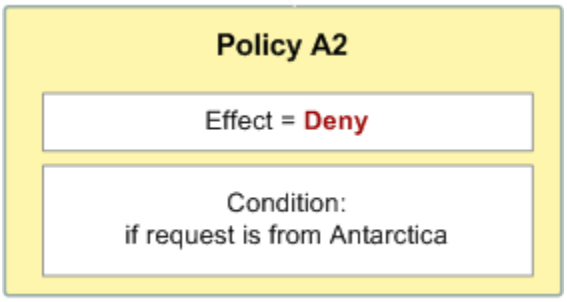
Uma política também resultará em uma negação padrão se uma condição em uma declaração não for atendida. Se todas as condições da declaração forem atendidas, a política resultará em permitir ou negação explícita, com base no valor do elemento efeito na política. As políticas não especificam o que fazer se uma condição não for atendida, e, portanto, o resultado padrão nesse caso é uma negação padrão.

Por exemplo, digamos que você deseja impedir solicitações que venham da Antártica. Você cria uma política (chamada Política A1) que permite uma solicitação somente se ela não for originária da Antártica. O seguinte diagrama ilustra a política.



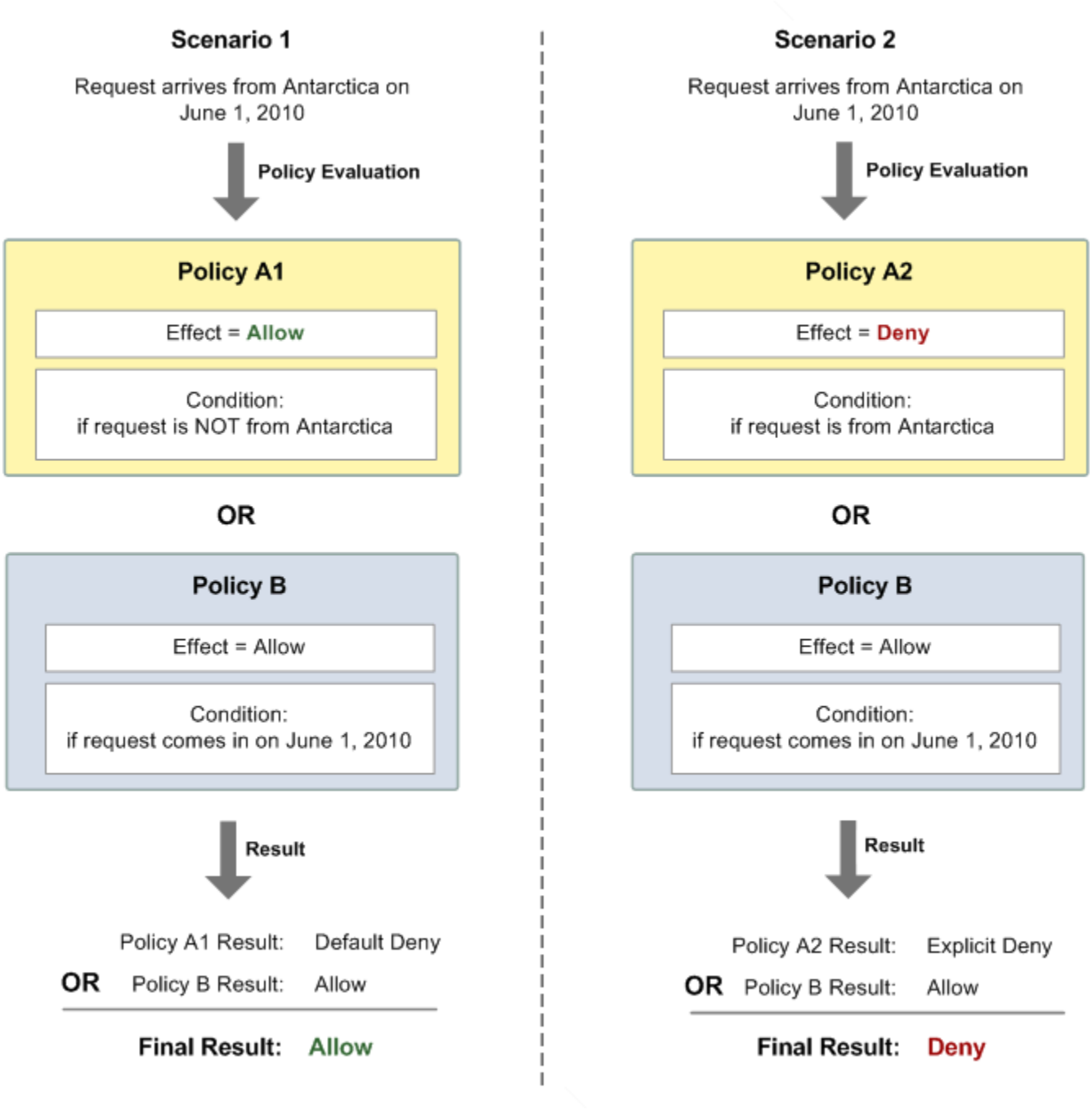
Se alguém enviar uma solicitação dos EUA, a condição será atendida (a solicitação não é originária da Antártica). Portanto, a solicitação será permitida. Porém, se alguém enviar uma solicitação da Antártica, a condição não será atendida, e, assim, o resultado da política será uma negação padrão.

Você pode transformar o resultado em uma negação explícita recriando a política (chamada Política A2), como mostrado no diagrama a seguir. Aqui, a política negará explicitamente uma solicitação se ela for originária da Antártica.



Se alguém enviar uma solicitação da Antártica, a condição será atendida, e o resultado da política será uma negação explícita.

A distinção entre negação padrão e negação explícita é importante porque uma negação padrão pode ser substituída por uma permissão, mas uma negação explícita não pode. Por exemplo, digamos que haja outra política que permita solicitações se elas chegarem em 1.º de junho de 2010. Como essa política afeta o resultado geral quando associada à política que restringe o acesso da Antártica? Compararemos o resultado geral ao associar a política baseada em data (que chamaremos de Política B) com as políticas anteriores A1 e A2. O Cenário 1 associa a Política A1 à Política B, e o Cenário 2 associa a Política A2 à Política B. A figura e a discussão a seguir mostram os resultados quando uma solicitação chega da Antártica em 1.º de junho de 2010.



No Cenário 1, a Política A1 retorna uma negação padrão, conforme descrito anteriormente nesta seção. A Política B retorna um permitir porque a política (por definição) permite solicitações que chegam em 1.º de junho de 2010. O permitir da Política B substitui a negação padrão da Política A1 e, assim, a solicitação é permitida.

No Cenário 2, a Política A2 retorna uma negação explícita, conforme descrito anteriormente nesta seção. Novamente, a Política B retorna um permitir. A negação explícita da Política A2 substitui o permitir da Política B e, assim, a solicitação é negada.

Casos de exemplo para controle de acesso do Amazon SNS

Esta seção descreve alguns exemplos de casos de uso comuns para controle de acesso.

Conceder Conta da AWS acesso a um tópico

Digamos que você tenha um tópico no Amazon SNS e queira permitir que um ou mais Contas da AWS realizem uma ação específica sobre esse tópico, como publicar mensagens. Você pode fazer isso usando a ação da API do Amazon SNS `AddPermission`.

A `AddPermission` ação permite que você especifique um tópico, uma lista de Conta da AWS IDs, uma lista de ações e um rótulo. Em seguida, o Amazon SNS gera e adiciona automaticamente uma nova declaração de política à política de controle de acesso do tópico. Não é necessário escrever a declaração de política por conta própria — o Amazon SNS trata disso para você. Se precisar remover a política posteriormente, você pode fazer isso ligando `RemovePermission` e fornecendo o rótulo que você usou ao adicionar a permissão.

Por exemplo, se você chamar `AddPermission` o tópico arn:aws:sns:us-east- 2:444455556666:, especificar a MyTopic ID 1111-2222-3333, a ação e o rótulo Conta da AWS , o Amazon SNS gerará `Publish` e inserirá a seguinte declaração de política na política de controle de acesso do tópico: `grant-1234-publish`

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Depois que essa declaração for adicionada, o Conta da AWS 1111-2222-3333 terá permissão para publicar mensagens no tópico.

Informações adicionais

- Gerenciamento personalizado de políticas: embora `AddPermission` seja conveniente para conceder permissões, geralmente é útil gerenciar manualmente a política de controle de acesso

do tópico para cenários mais complexos, como adicionar condições ou conceder permissões a serviços ou perfis específicos do IAM. Faça isso usando a API `SetTopicAttributes` para atualizar o atributo de política diretamente.

- Melhores práticas de segurança: tenha cuidado ao conceder permissões para garantir que somente entidades Contas da AWS ou entidades confiáveis tenham acesso aos seus tópicos do Amazon SNS. Analise e audite regularmente as políticas anexadas aos seus tópicos para manter a segurança.
- Limites da política: lembre-se de que há limites para o tamanho e a complexidade das políticas do Amazon SNS. Se você precisar adicionar muitas permissões ou condições complexas, certifique-se de que sua política permaneça dentro desses limites.

Limitar assinaturas para HTTPS

Para restringir o protocolo de entrega de notificações do seu tópico do Amazon SNS a HTTPS, você deve criar uma política personalizada. A ação `AddPermission` no Amazon SNS não permite que você especifique restrições de protocolo ao conceder acesso ao seu tópico. Portanto, você precisa escrever manualmente uma política que imponha essa restrição e, em seguida, usar a ação `SetTopicAttributes` para aplicar a política ao seu tópico.

Veja como você pode criar uma política que limita as assinaturas a HTTPS:

1. Escreva a política. A política deve especificar a ID da Conta da AWS à qual você deseja conceder acesso e impor a condição de que somente assinaturas HTTPS sejam permitidas. Abaixo está um exemplo de política que concede permissão ao Conta da AWS ID 1111-2222-3333 para assinar o tópico, mas somente se o protocolo usado for 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. Aplique a política. Use a ação `SetTopicAttributes` na API do Amazon SNS para aplicar essa política ao seu tópico. Defina o atributo `Policy` do tópico para a política JSON que você criou.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informações adicionais

- Personalizando o controle de acesso. Essa abordagem permite que você imponha controles de acesso mais granulares, como restringir protocolos de assinatura, o que não é possível apenas com a ação `AddPermission`. As políticas personalizadas oferecem flexibilidade para cenários que exigem condições específicas, como aplicação de protocolos ou restrições de endereço IP.
- Práticas recomendadas de segurança. Limitar as assinaturas ao HTTPS aumenta a segurança de suas notificações, garantindo que os dados em trânsito sejam criptografados. Analise regularmente suas políticas de tópicos para garantir que elas atendam aos seus requisitos de segurança e conformidade.
- Teste de políticas. Antes de aplicar a política em um ambiente de produção, teste-a em um ambiente de desenvolvimento para garantir que ela se comporte conforme o esperado. Isso ajuda a evitar problemas de acesso acidental ou restrições não intencionais.

Publicar mensagens em uma fila do Amazon SQS

Para publicar mensagens do seu tópico do Amazon SNS em uma fila do Amazon SQS, você precisa configurar as permissões corretas na fila do Amazon SQS. Embora o Amazon SNS e o Amazon SQS AWS usem a linguagem de política de controle de acesso, você deve definir explicitamente uma política na fila do Amazon SQS para permitir que as mensagens sejam enviadas do tópico do Amazon SNS.

Você pode conseguir isso usando a ação `SetQueueAttributes` para aplicar uma política personalizada à fila do Amazon SQS. Diferentemente do Amazon SNS, o Amazon SQS não suporta

a ação `AddPermission` para criar declarações de política com condições. Portanto, você deve escrever a política manualmente.

Veja a seguir um exemplo de política do Amazon SQS que concede permissão ao Amazon SNS para enviar mensagens para a sua fila. Observe que essa política está associada à fila do Amazon SQS, não ao tópico do Amazon SNS. As ações específicas são ações do Amazon SQS, e o recurso é o nome do recurso da Amazon (ARN) da fila. Você pode recuperar o ARN da fila usando a ação `GetQueueAttributes`.

```
{
  "Statement": [{
    "Sid": "Allow-SNS-SendMessage",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": ["sqs:SendMessage"],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:MyQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:444455556666:MyTopic"
      }
    }
  }]
}
```

Essa política usa a condição `aws:SourceArn` para restringir o acesso à fila com base na origem das mensagens que estão sendo enviadas. Isso garante que somente mensagens originadas do tópico SNS especificado (nesse caso, arn:aws:sns:us-east- 2:444455556666:) possam ser entregues na fila. MyTopic

Informações adicionais

- ARN da fila. Certifique-se de recuperar o ARN correto da sua fila do Amazon SQS usando a ação `GetQueueAttributes`. Esse ARN é essencial para definir as permissões corretas.
- Práticas recomendadas de segurança. Ao configurar políticas, siga sempre o princípio do privilégio mínimo. Conceda somente as permissões necessárias ao tópico do Amazon SNS para interagir com a fila do Amazon SQS e revise regularmente suas políticas para garantir que elas sejam seguras e estejam. up-to-date

- Políticas padrão no Amazon SNS. O Amazon SNS não concede automaticamente uma política padrão que permita que outras pessoas Serviços da AWS ou contas acessem tópicos recém-criados. Por padrão, os tópicos do Amazon SNS são criados sem permissões, o que significa que são privados e só podem ser acessados pela conta que os criou. Para permitir o acesso de outras pessoas Serviços da AWS, contas ou diretores, você deve definir e anexar explicitamente uma política de acesso ao tópico. Isso se alinha ao princípio do menor privilégio, garantindo que nenhum acesso não intencional seja concedido por padrão.
- Testes e validação. Depois de definir a política, teste a integração publicando mensagens no tópico do Amazon SNS e verificando se elas foram entregues com sucesso na fila do Amazon SQS. Isso ajuda a confirmar se a política está configurada corretamente.

Permitir que notificações de eventos do Simple Storage Service (Amazon S3) publiquem em um tópico

Para permitir que um bucket do Amazon S3 de outro Conta da AWS publique notificações de eventos em seu tópico do Amazon SNS, você precisa configurar adequadamente a política de acesso do tópico. Isso envolve escrever uma política personalizada que conceda permissão ao serviço Amazon S3 a partir da Conta da AWS específica e, em seguida, aplicar essa política ao seu tópico do Amazon SNS.

Veja como você pode configurá-lo:

1. Escreva a política. A política deve conceder o serviço Amazon S3 (s3.amazonaws.com) as permissões necessárias para publicar em seu tópico do Amazon SNS. Você usará a `SourceAccount` condição para garantir que somente o especificado Conta da AWS, que possui o bucket do Amazon S3, possa publicar notificações para o seu tópico.

   Veja abaixo um exemplo de política:

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

- Proprietário do tópico — 111122223333 é o ID que Conta da AWS possui o tópico do Amazon SNS.
- Proprietário do bucket do Amazon S3 — 444455556666 é o ID que Conta da AWS possui o bucket do Amazon S3 que envia notificações.

2. Aplique a política. Use a ação `SetTopicAttributes` para definir essa política em seu tópico do Amazon SNS. Isso atualizará o controle de acesso do tópico para incluir as permissões especificadas em sua política personalizada.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informações adicionais

- Usar condição **`SourceAccount`**. A `SourceAccount` condição garante que somente eventos originados do especificado Conta da AWS (444455556666 neste caso) possam acionar o tópico do Amazon SNS. Essa é uma medida de segurança para impedir que contas não autorizadas enviem notificações para seu tópico.
- Outros serviços de suporte **`SourceAccount`**. A condição `SourceAccount` é suportada pelos seguintes serviços. É fundamental usar essa condição quando você quiser restringir o acesso ao seu tópico do Amazon SNS com base na conta de origem.
  - Amazon API Gateway
  - Amazon CloudWatch
  - DevOpsGuru da Amazon
  - Amazon EventBridge
  - GameLift Servidores Amazon
  - API SMS and Voice do Amazon Pinpoint
  - Amazon RDS

- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

- Testes e validação. Depois de aplicar a política, teste a configuração acionando um evento no bucket do Amazon S3 e confirmando que ele foi publicado com sucesso no seu tópico do Amazon SNS. Isso ajudará a garantir que sua política seja configurada corretamente.
- Práticas recomendadas de segurança. Revise e audite regularmente suas políticas de tópicos do Amazon SNS para garantir que estejam em conformidade com seus requisitos de segurança. Limitar o acesso somente a contas e serviços confiáveis é essencial para manter as operações seguras.

Permitir que o Amazon SES publique em um tópico pertencente a outra conta

Você pode permitir que outra pessoa AWS service (Serviço da AWS) publique em um tópico que seja de propriedade de outra pessoa Conta da AWS. Suponha que você fez login na conta 111122223333, abriu o Amazon SES e criou um e-mail. Para publicar notificações sobre esse e-mail em um tópico do Amazon SNS que a conta 444455556666 possui, crie uma política como a apresentada a seguir. Para fazer isso, você precisa fornecer informações sobre a entidade principal (o outro serviço) e a propriedade de cada recurso. A instrução `Resource` fornece o ARN do tópico, que inclui o ID da conta do respectivo proprietário, 444455556666. A instrução `"aws:SourceOwner": "111122223333"` especifica que sua conta é proprietária do e-mail.

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

Ao publicar eventos no Amazon SNS, os seguintes serviços comportam `aws:SourceOwner`:

- Amazon API Gateway
- Amazon CloudWatch
- DevOpsGuru da Amazon
- GameLift Servidores Amazon
- API SMS and Voice do Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` versus `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner` foi desativado e novos serviços podem ser integrados ao Amazon SNS somente por meio de `aws:SourceArn` e `aws:SourceAccount`. O Amazon SNS ainda mantém a compatibilidade com versões anteriores dos serviços que no momento são compatíveis com `aws:SourceOwner`.

Cada uma das chaves de condição `aws:SourceAccount` e `aws:SourceOwner` é definida por alguns Serviços da AWS quando eles publicam em um tópico do Amazon SNS. Quando suportado, o valor será o ID da AWS conta de 12 dígitos em nome do qual o serviço está publicando dados. Alguns serviços aceitam a primeira e alguns aceitam a segunda.

- Veja como Permitir que notificações de eventos do Simple Storage Service (Amazon S3) publiquem em um tópico as notificações do Amazon S3 são usadas `aws:SourceAccount` e uma lista de AWS serviços que suportam essa condição.
- Veja como Permitir que o Amazon SES publique em um tópico pertencente a outra conta o Amazon SES usa `aws:SourceOwner` e uma lista de AWS serviços que suportam essa condição.

Permitir que contas em AWS Organizations uma organização publiquem em um tópico em uma conta diferente

O AWS Organizations serviço ajuda você a gerenciar centralmente o faturamento, controlar o acesso e a segurança e compartilhar recursos em todo o seu. Contas da AWS

Você pode encontrar o ID da organização no console do Organizations. Para obter mais informações, consulte Visualização de detalhes de uma organização na conta de gerenciamento.

Neste exemplo, qualquer Conta da AWS pessoa da organização `myOrgId` pode publicar um tópico `MyTopic` na conta do Amazon SNS. `444455556666` A política verifica o valor do ID da organização usando a chave de condição global `aws:PrincipalOrgID`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
```

```
}
```

Permitir que qualquer CloudWatch alarme seja publicado em um tópico em uma conta diferente

Use as etapas a seguir para invocar um tópico do Amazon SNS com CloudWatch um alarme em diferentes. Contas da AWS Este exemplo usa duas contas:

- A conta A é usada para criar o CloudWatch alarme.
- A conta B é usada para criar um tópico do SNS.

Crie um tópico do SNS na conta B

1. Faça login no console do Amazon SNS.
2. No painel de navegação, selecione Topics (Tópicos) e Create topic (Criar tópico).
3. Escolha Padrão para o tipo de tópico e, em seguida, crie um nome para o tópico.
4. Escolha Criar tópico e, em seguida, copie o ARN do tópico.
5. No painel de navegação, escolha Subscriptions (Assinaturas) e, depois, selecione Create subscription (Criar assinatura).
6. Adicione o ARN do tópico na seção ARN do tópico, escolha E-mail como protocolo e insira um endereço de e-mail.
7. Escolha Criar assinatura e, em seguida, verifique seu e-mail para confirmar a assinatura.

Crie um CloudWatch alarme na conta A

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Alarmes e, em seguida, escolha Criar alarmes.
3. Se você ainda não criou um alarme, crie um agora. Caso contrário, selecione sua métrica e forneça detalhes sobre o limite e os parâmetros de comparação.
4. Em Configurar ações, em Notificações, escolha Usar ARN do tópico para notificar outras contas e, em seguida, insira o tópico ARN da Conta B.
5. Crie um nome para o alarme e escolha Criar alarme.

Atualize a política de acesso do tópico SNS na conta B

1. Faça login no console do Amazon SNS.
2. No painel de navegação, escolha Tópicos e selecione o tópico.
3. Escolha Editar e adicione o seguinte à política:

> **Note**
>
> Substitua os valores de exemplo na política abaixo pelos seus.

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

Teste o alarme

Para testar o alarme, altere o limite do alarme com base nos pontos de dados métricos ou altere manualmente o estado do alarme. Ao alterar o limite ou o estado do alarme, você recebe uma notificação por e-mail.

Solução alternativa para usar um tópico local do Amazon SNS e encaminhar mensagens

Use as etapas a seguir para habilitar notificações CloudWatch entre contas do Amazon SNS para alarmes:

1. Crie um tópico do Amazon SNS na mesma conta do CloudWatchalarme (111122223333).
2. Inscreva uma função Lambda ou uma EventBridgeregra da Amazon para esse tópico do Amazon SNS.
3. A função ou EventBridge regra Lambda pode então publicar a mensagem no tópico do Amazon SNS na conta de destino (444455556666).

Restringir a publicação em um tópico do Amazon SNS somente de um endpoint da VPC específico

Nesse caso, o tópico na conta 444455556666 está autorizado a publicar somente no endpoint da VPC com o ID `vpce-1ab2c34d`.

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

Como o Amazon SNS funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon SNS, entenda que recursos do IAM estão disponíveis para uso com o Amazon SNS.

Recursos do IAM que você pode usar com o Amazon Simple Notification Service

Atributo do IAM	Suporte ao Amazon SNS
Políticas baseadas em identidade	Sim
Políticas baseadas em atributos	Sim
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Parcial
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Não

Para ter uma visão de alto nível de como o Amazon SNS e AWS outros serviços funcionam com a maioria dos recursos do IAM, AWS consulte os serviços que funcionam com o IAM no Guia do usuário do IAM.

AWS políticas gerenciadas para Amazon Simple Notification Service

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque elas estão disponíveis para uso de todos os

AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

AWS política gerenciada: Amazon SNSFull Access

`AmazonSNSFullAccess`fornece acesso total ao Amazon SNS usando o. AWS Management Console Essa política também inclui as seguintes ações de leitura e gravação para AWS End User Messaging SMS quando chamado usando o Amazon SNS. É possível vincular esta política a usuários, grupos ou funções.

Detalhes das permissões

As seguintes permissões se aplicam somente ao usar o Amazon SNS: APIs

- `sns:*`— Permite permissões completas para realizar qualquer ação relacionada ao Amazon SNS. Esse caractere curinga (*) significa que o usuário pode executar todas as ações possíveis do Amazon SNS.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Permite que você recupere uma lista de números de telefone que foram verificados para envio de mensagens SMS dentro da Conta da AWS.
- `sms-voice:CreateVerifiedDestinationNumber`— Permite que você verifique um novo número de telefone para uso com os serviços de mensagens SMS internos AWS.
- `sms-voice:SendDestinationNumberVerificationCode`— Permite que você envie um código de verificação para um número de telefone que está sendo verificado para receber mensagens SMS dentro da AWS.
- `sms-voice:SendTextMessage`— Permite criar uma nova mensagem de texto e enviá-la para o número de telefone do destinatário. `SendTextMessage` envia apenas uma mensagem SMS para um destinatário cada vez que é invocada.

- `sms-voice:DeleteVerifiedDestinationNumber`— Permite que você remova um número de telefone da lista de números verificados dentro do Conta da AWS
- `sms-voice:VerifyDestinationNumber`— Permite iniciar e concluir o processo de verificação de um número de telefone a ser usado nos AWS serviços de mensagens SMS.
- `sms-voice:DescribeAccountAttributes`— Permite que você recupere informações detalhadas sobre os atributos no nível da conta relacionados aos serviços de mensagens SMS contidos na AWS.
- `sms-voice:DescribeSpendLimits`— Permite que você recupere informações sobre os limites de gastos associados aos serviços de mensagens SMS na Conta da AWS
- `sms-voice:DescribePhoneNumbers`— Permite que você recupere informações detalhadas sobre os números de telefone associados aos serviços de mensagens SMS na Conta da AWS
- `sms-voice:SetTextMessageSpendLimitOverride`— Permite que você defina ou substitua o limite de gastos para mensagens de texto SMS dentro do Conta da AWS
- `sms-voice:DescribeOptedOutNumbers`— Permite que você recupere uma lista de números de telefone que cancelaram a opção de recebimento de mensagens SMS em sua conta da AWS .
- `sms-voice:DeleteOptedOutNumber`— Permite que você remova um número de telefone da lista de números excluídos dentro do Conta da AWS

Política de exemplo do **`AmazonSNSFullAccess`**

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

Para ver as permissões dessa política, consulte Amazon SNSFull Access na Referência de políticas AWS gerenciadas.

AWS política gerenciada: Amazon SNSRead OnlyAccess

`AmazonSNSReadOnlyAccess` fornece acesso de leitura ao Amazon SES por meio do AWS Management Console. Essa política também inclui as seguintes ações somente para leitura AWS End User Messaging SMS quando chamado usando o Amazon SNS. É possível vincular esta política a usuários, grupos e funções.

Detalhes das permissões

As seguintes permissões se aplicam somente ao usar o Amazon SNS: APIs

- `sns:GetTopicAttributes`— Permite que você recupere os atributos de um tópico do Amazon SNS. Isso inclui informações como o ARN (Amazon Resource Name) do tópico, a lista de assinantes, políticas de entrega, políticas de controle de acesso e quaisquer outros metadados associados ao tópico.
- `sns:List*`— Permite que você execute qualquer operação que comece com `List` para recursos do Amazon SNS. Isso inclui permissões para listar vários elementos relacionados ao Amazon SNS, como:
  - `sns:ListTopics`— Permite que você recupere uma lista de todos os tópicos do Amazon SNS na Conta da AWS.

  - `sns:ListSubscriptions`— Permite que você recupere uma lista de todas as assinaturas dos tópicos do Amazon SNS.
  - `sns:ListSubscriptionsByTopic`— Permite listar todas as assinaturas para um tópico específico do Amazon SNS.
  - `sns:ListPlatformApplications`— Permite listar todos os aplicativos da plataforma criados para notificações push móveis.
  - `sns:ListEndpointsByPlatformApplication`— Permite listar todos os endpoints associados a um aplicativo de plataforma.
- `sns:CheckIfPhoneNumberIsOptedOut`— Permite verificar se um número de telefone específico cancelou a opção de recebimento de mensagens SMS por meio do Amazon SNS.
- `sns:GetEndpointAttributes`— Permite que você recupere os atributos de um endpoint associado a um aplicativo da plataforma Amazon SNS. Isso pode incluir atributos como o status ativado do endpoint, dados personalizados do usuário e quaisquer outros metadados associados ao endpoint.
- `sns:GetDataProtectionPolicy`— Permite que você recupere a política de proteção de dados associada a um tópico do Amazon SNS.
- `sns:GetPlatformApplicationAttributes`— Permite que você recupere os atributos de um aplicativo da plataforma Amazon SNS. Os aplicativos de plataforma são usados no Amazon SNS para enviar notificações por push para dispositivos móveis por meio de serviços como o Apple Push Notification Service (APNS) ou Firebase Cloud Messaging (FCM).
- `sns:GetSMSAttributes`— Permite que você recupere as configurações padrão de SMS para a Conta da AWS.
- `sns:GetSMSSandboxAccountStatus`— Permite que você recupere o status atual da sandbox de SMS para sua Conta da AWS.
- `sns:GetSubscriptionAttributes`— Permite que você recupere os atributos de uma assinatura específica de um tópico do Amazon SNS.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Permite que você visualize ou recupere uma lista de números de telefone que foram verificados para envio de mensagens SMS dentro do Conta da AWS
- `sms-voice:DescribeAccountAttributes`— Permite que você visualize ou recupere informações sobre os atributos no nível da conta relacionados aos serviços de mensagens SMS contidos na AWS.
- `sms-voice:DescribeSpendLimits`— Permite que você visualize ou recupere informações sobre os limites de gastos associados aos serviços de mensagens SMS em sua Conta da AWS

- `sms-voice:DescribePhoneNumbers`— Permite que você visualize ou recupere informações sobre os números de telefone usados para serviços de mensagens SMS na Conta da AWS
- `sms-voice:DescribeOptedOutNumbers`— Permite que você visualize ou recupere uma lista de números de telefone que cancelaram a opção de recebimento de mensagens SMS da sua Conta da AWS

Política de exemplo do **`AmazonSNSReadOnlyAccess`**

```
{
    "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"
                }
```

```
            }
        }
    ]
}
```

Para ver as permissões dessa política, consulte Amazon SNSFull Access na Referência de políticas AWS gerenciadas.

Atualizações do Amazon SNS para AWS políticas gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Amazon SNS desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nesta página, assine o feed RSS na página Histórico do documento do Amazon SNS.

Alteração	Descrição	Data
AmazonSNSFullAccess: atualizar para uma política existente	O Amazon SNS adicionou novas permissões para permitir acesso total ao Amazon SNS usando o AWS Management Console.	24/09/2024
Amazon SNSRead OnlyAccess — Atualização de uma política existente	O Amazon SNS adicionou novas permissões para permitir acesso somente leitura ao Amazon SNS usando o AWS Management Console.	24/09/2024
O Amazon SNS passou a monitorar alterações	O Amazon SNS começou a monitorar as alterações em suas políticas AWS gerenciadas.	27/08/2024

Ações de políticas para o Amazon SNS

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista das ações do Amazon SNS, consulte Recursos definidos pelo Amazon Simple Notification Service na Referência de autorização do serviço.

As ações de política no Amazon SNS usam o seguinte prefixo antes da ação:

```
sns
```

Para especificar várias ações em uma única declaração, separe-as com vírgulas.

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Para visualizar exemplos de políticas baseadas em identidade do Amazon SNS, consulte Exemplos de políticas baseadas em identidade do Amazon Simple Notification Service.

Recursos de políticas para o Amazon SNS

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode

ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon SNS e seus ARNs, consulte Ações definidas pelo Amazon Simple Notification Service na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Recursos definidos pelo Amazon Simple Notification Service.

Para visualizar exemplos de políticas baseadas em identidade do Amazon SNS, consulte Exemplos de políticas baseadas em identidade do Amazon Simple Notification Service.

Chaves de condição de política para o Amazon SNS

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos de `Condition` em uma declaração ou várias chaves em um único elemento de `Condition`, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um recurso somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos da política do IAM: variáveis e tags no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista de chaves de condição do Amazon SNS, consulte Chaves de condição do Amazon Simple Notification Service na Referência de autorização do serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Recursos definidos pelo Amazon Simple Notification Service.

Para visualizar exemplos de políticas baseadas em identidade do Amazon SNS, consulte Exemplos de políticas baseadas em identidade do Amazon Simple Notification Service.

ACLs no Amazon SNS

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com o Amazon SNS

Compatível com ABAC (tags em políticas): parcial

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define as permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a entidades do IAM (usuários ou funções) e a vários AWS recursos. Marcar de entidades e atributos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela estiver tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

Usar credenciais temporárias com o Amazon SNS

Compatível com credenciais temporárias: sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS "Trabalhe com o IAM" no Guia do usuário do IAM.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar funções, consulte Alternar para um perfil do IAM (console) no Guia do usuário do IAM.

Você pode criar manualmente credenciais temporárias usando a AWS API AWS CLI ou. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias no IAM.

Permissões de entidades principais entre serviços para o Amazon SNS

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado um principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Perfis de serviço para o Amazon SNS

Compatível com perfis de serviço: sim

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> A alteração das permissões de um perfil de serviço pode interromper a funcionalidade do Amazon SNS. Edite perfis de serviço somente quando o Amazon SNS fornecer orientação para isso.

Perfis vinculados ao serviço para o Amazon SNS

Compatível com perfis vinculados ao serviço: Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir o perfil para executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Perfil vinculado ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Exemplos de políticas baseadas em identidade do Amazon Simple Notification Service

Por padrão, os usuários e perfis não têm permissão para criar ou modificar recursos do Amazon SNS. Eles também não podem realizar tarefas usando a AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) ou. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Amazon SNS, incluindo o formato de cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição ARNs para o Amazon Simple Notification Service na Referência de autorização de serviço.

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon SNS em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando

as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon SNS

Para acessar o console do Amazon Simple Notification Service, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon SNS em seu. Conta da AWS Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam à operação de API que estiverem tentando executar.

Para garantir que usuários e funções ainda possam usar o console do Amazon SNS, também anexe o Amazon `ConsoleAccess` SNS `ReadOnly` AWS ou a política gerenciada às entidades. Para obter informações, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS

Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada usuário Conta da AWS root. Para obter mais informações sobre Organizations e SCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.

- Políticas de controle de recursos (RCPs) — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo o usuário Conta da AWS raiz, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Políticas baseadas em identidade do Amazon SNS

Compatível com políticas baseadas em identidade: sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para o Amazon SNS

Para visualizar exemplos de políticas baseadas em identidade do Amazon SNS, consulte Exemplos de políticas baseadas em identidade do Amazon Simple Notification Service.

Políticas baseadas em recursos no Amazon SNS

É compatível com políticas baseadas em atributos	Sim

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um administrador do IAM na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Usar políticas baseadas em identidade com o Amazon SNS

O Amazon Simple Notification Service se integra ao AWS Identity and Access Management (IAM) para que você possa especificar quais ações do Amazon SNS um usuário pode realizar com os Conta da AWS recursos do Amazon SNS. Você pode especificar um tópico específico na política. Por exemplo, você poderá usar variáveis ao criar uma política do IAM que conceda a determinados usuários em sua organização permissão para usar a ação `Publish` com tópicos específicos em sua Conta da AWS. Para obter mais informações, consulte Variáveis de políticas no guia Como usar o IAM.

> **Important**
>
> O uso do Amazon SNS com o IAM não altera a forma como você usa o Amazon SNS. Não há alterações nas ações do Amazon SNS nem novas ações do Amazon SNS relacionadas a controle de acesso e usuários.

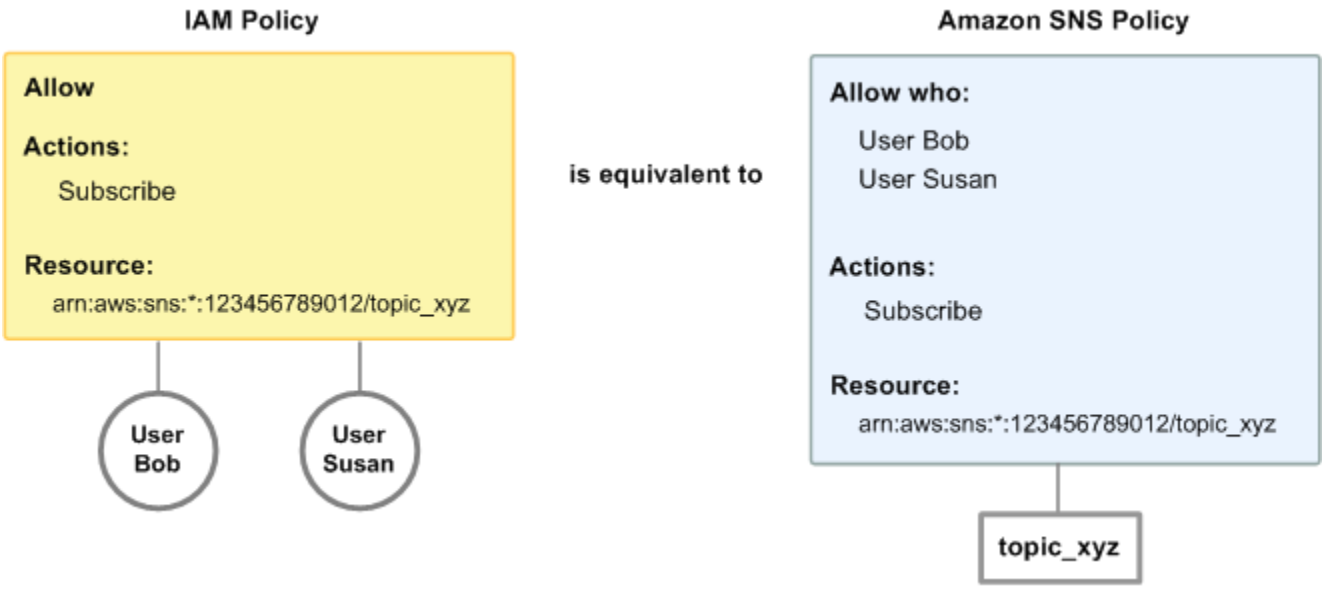
Para obter exemplos de políticas que abrangem ações e recursos do Amazon SNS, consulte Exemplos de políticas do Amazon SNS.

Políticas do IAM e do Amazon SNS juntas

Você usa uma política do IAM para restringir o acesso dos usuários a ações e tópicos do Amazon SNS. Uma política do IAM pode restringir o acesso somente aos usuários da sua AWS conta, não a outros Contas da AWS.

Você usa uma política do Amazon SNS com determinado tópico para restringir quem pode trabalhar com esse tópico (por exemplo, quem pode publicar mensagens nele, quem pode assiná-lo etc.). As políticas do Amazon SNS podem conceder acesso a outras pessoas Contas da AWS ou a usuários dentro da sua. Conta da AWS

Para conceder permissão para seus tópicos do Amazon SNS aos usuários, você pode usar políticas do IAM, políticas do Amazon SNS ou ambas. Na maior parte das vezes, você obtém os mesmos resultados com qualquer uma delas. Por exemplo, o diagrama a seguir mostra uma política do IAM e uma política do Amazon SNS que são equivalentes. A política do IAM permite a `Subscribe` ação do Amazon SNS para o tópico chamado topic_xyz em seu. Conta da AWS A política do IAM é anexada aos usuários Bob e Susan (o que significa que Bob e Susan têm as permissões declaradas na política). A política do Amazon SNS também concede a Pedro e Ana permissão para acessar `Subscribe` para topic_xyz.



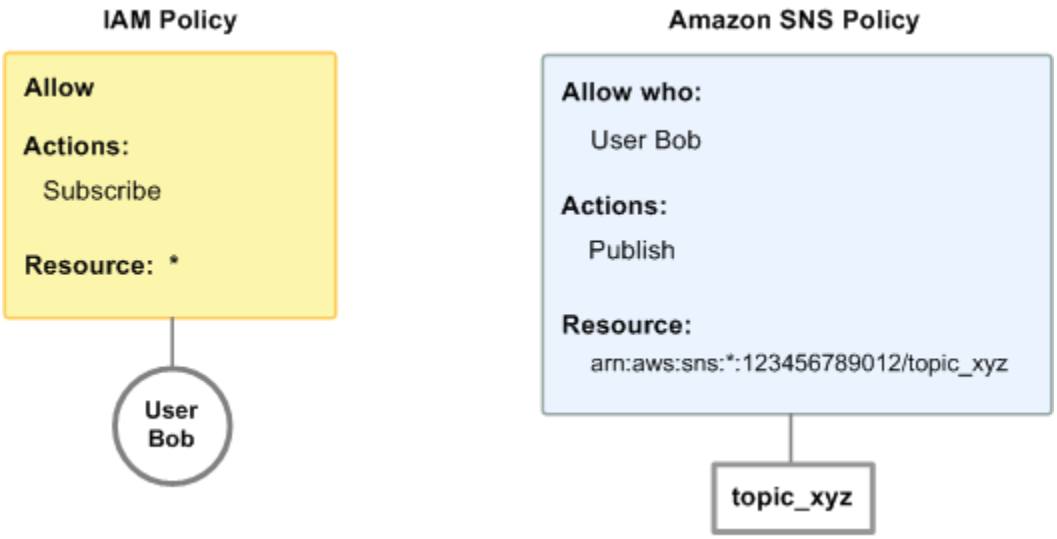
> **Note**
>
> O exemplo anterior mostra políticas simples, sem condições. Você pode especificar determinada condição na política e obter o mesmo resultado.

Há uma diferença entre as políticas AWS do IAM e do Amazon SNS: o sistema de políticas do Amazon SNS permite que você conceda permissão a Contas da AWS outras pessoas, enquanto a política do IAM não.

Você decide como usar ambos os sistemas juntos para gerenciar suas permissões, de acordo com suas necessidades. Os exemplos a seguir mostram como os dois sistemas de política funcionam em conjunto.

Example 1

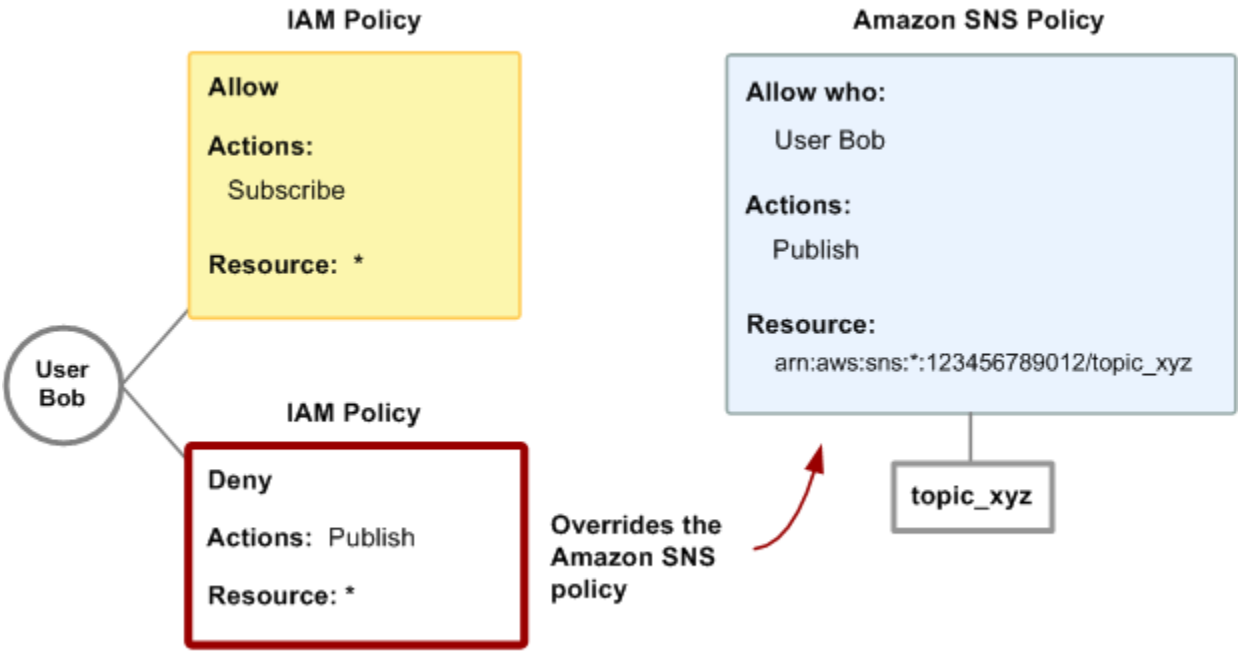
Neste exemplo, uma política do IAM e uma política do Amazon SNS se aplicam a Pedro. A política do IAM concede a ele permissão para `Subscribe` qualquer um dos tópicos, enquanto a política Conta da AWS do Amazon SNS concede a ele permissão para usar `Publish` em um tópico específico (topic_xyz). O seguinte diagrama ilustra o conceito.



Se Bob enviasse uma solicitação para se inscrever em qualquer tópico da AWS conta, a política do IAM permitiria a ação. Se Pedro enviasse uma solicitação para publicar uma mensagem no topic_xyz, a política do Amazon SNS permitiria a ação.

Example 2

Neste exemplo, nós nos baseamos no exemplo 1 (onde há duas políticas que se aplicam a Pedro). Digamos que Pedro publique mensagens no tópico topic_xyz que não deveria ter publicado. Por isso, você deseja remover totalmente a capacidade dele de publicar nos tópicos. O mais fácil a fazer é adicionar uma política do IAM que negue acesso a ele à ação `Publish` em todos os tópicos. Essa terceira política se sobrepõe à política do Amazon SNS que originalmente deu a ele permissão para publicar no topic_xyz, pois uma negação explícita sempre se sobrepõe a uma permissão (para obter mais informações sobre a lógica de avaliação de políticas, consulte Lógica de avaliação). O seguinte diagrama ilustra o conceito.



Para obter exemplos de políticas que abrangem ações e recursos do Amazon SNS, consulte Exemplos de políticas do Amazon SNS.

Formato do ARN de recurso do Amazon SNS

Para o Amazon SNS, os tópicos são o único tipo de recurso que você pode especificar em uma política. Veja a seguir o formato de nome do recurso da Amazon (ARN) para tópicos.

```
arn:aws:sns:region:account_ID:topic_name
```

Para obter mais informações sobre ARNs, consulte o ARNsGuia do usuário do IAM.

Example

A seguir está um ARN para um tópico nomeado MyTopic na região us-east-2, pertencente a 123456789012. Conta da AWS

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

Example

Se você tivesse um tópico nomeado MyTopic em cada uma das diferentes regiões suportadas pelo Amazon SNS, você poderia especificar os tópicos com o seguinte ARN.

```
arn:aws:sns:*:123456789012:MyTopic
```

Você pode usar os caracteres curinga * e ? no nome do tópico. Por exemplo, o seguinte poderia fazer referência a todos os tópicos criados por Pedro nos quais ele inseriu o prefixo `bob_`.

```
arn:aws:sns:*:123456789012:bob_*
```

Por uma questão de conveniência para você, ao criar um tópico, o Amazon SNS retorna o ARN do tópico como resposta.

Ações da API do Amazon SNS

Em uma política do IAM, você pode especificar quaisquer ações que o Amazon SNS oferecer. No entanto, as ações `ConfirmSubscription` e `Unsubscribe` não exigem autenticação, o que significa que, mesmo que você especifique essas ações em uma política, o IAM não restringirá o acesso dos usuários a essas ações.

Cada ação que você especificar em uma política deve ser prefixada com a string `sns:` em letras minúsculas. Por exemplo, para especificar todas as ações do Amazon SNS, você usaria `sns:*`. Para obter uma lista de ações, consulte Amazon Simple Notification Service.

Chaves de política do Amazon SNS

O Amazon SNS implementa as seguintes chaves de política AWS abrangentes, além de algumas chaves específicas do serviço.

Para ver uma lista das chaves de condição suportadas por cada uma delas AWS service (Serviço da AWS), consulte Ações, recursos e chaves de condição Serviços da AWS no Guia do usuário do IAM. Para ver uma lista de chaves de condição que podem ser usadas em várias Serviços da AWS, consulte as chaves de contexto de condição AWS globais no Guia do usuário do IAM.

O Amazon SNS usa as chaves específicas de serviço a seguir. Use essas chaves em políticas que restringem o acesso a solicitações de `Subscribe`.

- sns:endpoint: o URL, o endereço de e-mail ou o ARN de uma solicitação `Subscribe` ou de uma assinatura confirmada anteriormente. Use com condições de string (consulte Exemplos de políticas do Amazon SNS) para restringir o acesso a endpoints específicos (por exemplo, *@yourcompany.com).

- sns:protocol: o valor `protocol` de uma solicitação `Subscribe` ou de uma assinatura confirmada anteriormente. Use com condições de string (consulte Exemplos de políticas do Amazon SNS) para restringir a publicação a protocolos de entrega específicos (por exemplo, https).

Exemplos de políticas do Amazon SNS

Esta seção mostra várias políticas simples para controlar o acesso de usuários ao Amazon SNS.

> **Note**
>
> No futuro, o Amazon SNS pode adicionar novas ações que devem ser logicamente incluídas em uma das seguintes políticas, com base nos objetivos indicados da política.

Example 1: Permitir que um grupo crie e gerencie tópicos

Neste exemplo, criamos uma política que concede acesso a `CreateTopic`, `ListTopics`, `SetTopicAttributes` e `DeleteTopic`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: Permitir que o grupo de TI publique mensagens em determinado tópico

Neste exemplo, criamos um grupo de TI e atribuímos uma política que concede acesso a `Publish` no tópico de interesse específico.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3: Permita que os Conta da AWS usuários se inscrevam em tópicos

Neste exemplo, criamos uma política que concede acesso à ação `Subscribe`, com as condições de correspondência de string para as chaves de política `sns:Protocol` e `sns:Endpoint`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "sns:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: Permitir que um parceiro publique mensagens em determinado tópico

Você pode usar uma política do Amazon SNS ou uma política do IAM para permitir que um parceiro publique em um tópico específico. Se seu parceiro tiver uma Conta da AWS, talvez seja mais fácil usar uma política do Amazon SNS. No entanto, qualquer pessoa da empresa do parceiro que possua as credenciais AWS de segurança pode publicar mensagens sobre o tópico. Este exemplo pressupõe que você deseje limitar o acesso a determinada pessoa (ou aplicação). Para fazer isso, é preciso tratar o parceiro como um usuário dentro de sua própria empresa e usar uma política do IAM em vez de uma política do Amazon SNS.

Neste exemplo, criamos um grupo chamado WidgetCo que representa a empresa parceira; criamos um usuário para a pessoa específica (ou aplicativo) na empresa parceira que precisa de acesso; e então colocamos o usuário no grupo.

Em seguida, anexamos uma política que concede ao grupo `Publish` acesso ao tópico específico nomeado WidgetPartnerTopic.

Também queremos evitar que o WidgetCo grupo faça qualquer outra coisa com tópicos, então adicionamos uma declaração que nega permissão para qualquer ação do Amazon SNS, `Publish` exceto em qualquer outro tópico que não seja. WidgetPartnerTopic Isso será necessário somente se

houver uma política ampla em outro lugar no sistema que conceda aos usuários amplo acesso ao Amazon SNS.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Gerenciar políticas personalizadas do IAM do Amazon SNS

As políticas personalizadas do IAM permitem que você especifique permissões para usuários, grupos ou funções individuais do IAM, concedendo ou restringindo o acesso a AWS recursos e ações específicos. Ao gerenciar os recursos do Amazon SNS, as políticas do IAM personalizadas permitem que você personalize as permissões de acesso de acordo com os requisitos operacionais e de segurança da sua organização.

Use as etapas a seguir para gerenciar políticas do IAM personalizadas para o Amazon SNS:

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas.
3. Para criar uma nova política personalizada do IAM, escolha Criar política e escolha SNS. Para editar uma política existente, selecione a política na lista e escolha Editar política.
4. No editor de políticas, defina as permissões para acessar os recursos do Amazon SNS. Você pode especificar ações, recursos e condições com base em seus requisitos específicos.
5. Para conceder permissões para ações do Amazon SNS, inclua ações relevantes do Amazon SNS como `sns:Publish`, `sns:Subscribe` e `sns:DeleteTopic` e em sua política do IAM. Defina o ARN (Amazon Resource Name) dos tópicos do Amazon SNS aos quais as permissões se aplicam.

6. Especifique os usuários, grupos ou perfis do IAM aos quais a política deve ser anexada. Você pode anexar a política diretamente aos usuários ou grupos do IAM ou associá-la aos perfis do IAM usados pelos Serviços da AWS ou aplicações.
7. Revise a configuração da política do IAM para garantir que ela esteja alinhada aos seus requisitos de controle de acesso. Depois de verificadas, salve as alterações da política.
8. Anexe a política personalizada do IAM aos usuários, grupos ou perfis relevantes do IAM dentro da sua Conta da AWS. Isso concede a eles as permissões definidas na política para gerenciar os recursos do Amazon SNS.

Usar credenciais de segurança temporárias com o Amazon SNS

AWS Identity and Access Management (IAM) permite que você conceda credenciais de segurança temporárias a usuários e aplicativos que precisam acessar seus AWS recursos. Essas credenciais de segurança temporárias são usadas principalmente para perfis do IAM e acesso federado por meio de protocolos padrão do setor, como SAML e OpenID Connect (OIDC).

Para gerenciar com eficácia o acesso aos AWS recursos, é essencial entender os seguintes conceitos-chave:

- Funções do IAM — As funções são usadas para delegar acesso aos AWS recursos. As funções podem ser assumidas por entidades como EC2 instâncias da Amazon, funções Lambda ou usuários de outras. Contas da AWS
- Usuários federados — são usuários autenticados por meio de provedores de identidade externos (IdPs) usando SAML ou OIDC. O acesso federado é recomendado para usuários humanos, enquanto os perfis do IAM devem ser usadas para aplicativos de software.
- Roles Anywhere — Para aplicativos externos que exigem AWS acesso, você pode usar o IAM Roles Anywhere para gerenciar o acesso com segurança sem criar credenciais de longo prazo.

Use credenciais de segurança temporárias para fazer solicitações ao Amazon SNS. As bibliotecas de API SDKs e de API calculam a assinatura necessária usando essas credenciais para autenticar suas solicitações. Solicitações com credenciais expiradas serão negadas pelo Amazon SNS.

Para obter mais informações sobre credenciais de segurança temporárias, consulte Como usar perfis do IAM e Fornecer acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.

Example Exemplo de solicitação de HTTPS

O exemplo a seguir demonstra como autenticar uma solicitação do Amazon SNS usando credenciais de segurança temporárias obtidas AWS Security Token Service do (STS).

```
https://sns.us-east-2.amazonaws.com/
?Action=CreateTopic
&Name=My-Topic
&SignatureVersion=4
&SignatureMethod=AWS4-HMAC-SHA256
&Timestamp=2023-07-05T12:00:00Z
&X-Amz-Security-Token=SecurityTokenValue
&X-Amz-Date=20230705T120000Z
&X-Amz-Credential=<your-access-key-id>/20230705/us-east-2/sns/aws4_request
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Etapas para autenticar a solicitação

1. Obter credenciais de segurança temporárias — Use AWS STS para assumir uma função ou obter credenciais de usuário federado. Isso fornecerá a você um ID de chave de acesso, uma chave de acesso secreta e um token de segurança.
2. Crie a solicitação — Inclua os parâmetros necessários para sua ação do Amazon SNS (por exemplo, CreateTopic) e garanta o uso de HTTPS para comunicação segura.
3. Assine a solicitação — Use o processo de assinatura da AWS versão 4 para assinar sua solicitação. Isso envolve a criação de uma solicitação canônica e, em seguida string-to-sign, o cálculo da assinatura. Para obter mais informações sobre o AWS Signature versão 4, consulte Usar a assinatura do Signature versão 4 no Guia do usuário do Amazon EBS.
4. Envie a solicitação — Inclua o X-Amz-Security-Token no cabeçalho da solicitação para passar as credenciais de segurança temporárias para o Amazon SNS.

Permissões da API do Amazon SNS: referência de ações e recursos

A lista a seguir fornece informações específicas para a implementação de controle de acesso do Amazon SNS:

- Cada política deve abranger apenas um tópico (ao gravar uma política, não inclua declarações que abranjam diferentes tópicos).

- Cada política deve ter uma política exclusiva `Id`.
- Cada declaração em uma política deve ter uma declaração exclusiva `sid`.

Cotas de política

A tabela a seguir lista as cotas máximas para uma declaração de política.

Name	Cota máxima
Bytes	30 kb
Declarações	100
Entidades principais	1 a 200 (0 é inválido.)
Recurso	1 (0 é inválido. O valor deve corresponder ao ARN do tópico da política.)

Ações de política válidas do Amazon SNS

O Amazon SNS é compatível com as ações mostradas na tabela a seguir.

Ação	Descrição
sns: AddPermission	Concede permissão para adicionar permissões à política do tópico.
sns: DeleteTopic	Concede permissão para excluir um tópico.
sns: GetDataProtectionPolicy	Concede permissão para recuperar a política de proteção de dados de um tópico.
sns: GetTopicAttributes	Concede permissão para receber todos os atributos de tópico.
sns: ListSubscriptionsByTopic	Concede permissão para recuperar todas as assinaturas de determinado tópico.

Ação	Descrição
sns: ListTagsForResource	Concede permissão para relacionar todas as etiquetas adicionadas a um tópico específico.
sns:Publish	Concede permissão para publicar individualmente ou em lote em um tópico ou endpoint. Para obter mais informaçõ es, consulte Publish e PublishBatchna Referência da API do Amazon Simple Notification Service.
sns: PutDataProtectionPolicy	Concede permissão para definir a política de proteção de dados de um tópico.
sns: RemovePermission	Concede permissão para remover as permissões na política de tópico.
sns: SetTopicAttributes	Concede permissão para definir os atributos de um tópico.
sns:Subscribe	Concede permissão para assinar um tópico.

Chaves específicas do serviço

O Amazon SNS usa as chaves específicas de serviço a seguir. Você pode usar essas chaves nas políticas que restringem o acesso a solicitações `Subscribe`.

- sns:endpoint: o URL, o endereço de e-mail ou o ARN de uma solicitação `Subscribe` ou de uma assinatura confirmada anteriormente. Use com condições de string (consulte Exemplos de políticas do Amazon SNS) para restringir o acesso a endpoints específicos (por exemplo,*@example.com).
- sns:protocol: o valor `protocol` de uma solicitação `Subscribe` ou de uma assinatura confirmada anteriormente. Use com condições de string (consulte Exemplos de políticas do Amazon SNS) para restringir a publicação a protocolos de entrega específicos (por exemplo, https).

> **Important**
>
> Quando você usar uma política para controlar o acesso por sns:Endpoint, lembre-se de que os problemas de DNS podem afetar a resolução de nomes de endpoints no futuro.

Resolução de problemas de identidade e acesso do Amazon Simple Notification Service

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon SNS e o IAM.

Não tenho autorização para executar uma ação no Amazon SNS

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `sns:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

Nesse caso, a política de Mateo deve ser atualizada para permitir que ele tenha acesso ao recurso *`my-example-widget`* usando a ação `sns:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Caso receba uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, as políticas deverão ser atualizadas para permitir a transmissão de um perfil ao Amazon SNS.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon SNS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas de fora da minha Conta da AWS acessem meus recursos do Amazon SNS

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon SNS é compatível com esses recursos, consulte Como o Amazon SNS funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Registrar em log e monitorar no Amazon SNS

O Amazon SNS permite que você rastreie e monitore a atividade de mensagens registrando chamadas de API CloudTrail e monitorando tópicos com. CloudWatch Essas ferramentas ajudam você a obter informações sobre a entrega de mensagens, solucionar problemas e garantir a integridade de seus fluxos de trabalho de mensagens. Este tópico abrange o seguinte:

- Registrando chamadas AWS de API do SNS usando AWS CloudTrail. Esse registro permite que você acompanhe as ações realizadas em seus tópicos do Amazon SNS, como criação de tópicos,

gerenciamento de assinaturas e publicação de mensagens. Ao analisar CloudTrail os registros, você pode identificar quem fez solicitações específicas de API e quando essas solicitações foram feitas, ajudando você a auditar e solucionar problemas de uso do Amazon SNS.

- Monitorando tópicos do Amazon SNS usando CloudWatch. CloudWatch fornece métricas que permitem observar o desempenho e a integridade de seus tópicos do Amazon SNS em tempo real. Configure alarmes com base nessas métricas, permitindo que você responda prontamente a qualquer anomalia, como falhas na entrega ou alta latência de mensagens. Esse recurso de monitoramento garante que você possa manter a confiabilidade do seu sistema de mensagens baseado em SNS abordando proativamente possíveis problemas.

Registrando chamadas AWS de API do SNS usando AWS CloudTrail

AWS O SNS é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou um AWS service (Serviço da AWS). CloudTrail captura todas as chamadas de API para o SNS como eventos. As chamadas capturadas incluem chamadas do console do SNS e chamadas de código para as operações da API do SNS. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao SNS, o endereço IP do qual a solicitação foi feita, quando foi feita e detalhes adicionais.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

CloudTrail está ativo Conta da AWS quando você cria a conta e você tem acesso automático ao histórico de CloudTrail eventos. O histórico de CloudTrail eventos fornece um registro visível, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento registrados em um. Região da AWS Para obter mais informações, consulte Trabalhando com o histórico de CloudTrail eventos no Guia AWS CloudTrail do usuário. Não há CloudTrail cobrança pela visualização do histórico de eventos.

Para um registro contínuo dos eventos dos Conta da AWS últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do CloudTrailLake.

CloudTrail trilhas

Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Todas as trilhas criadas usando o AWS Management Console são multirregionais. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. É recomendável criar uma trilha multirregional porque você captura todas as atividades Regiões da AWS em sua conta. Ao criar uma trilha de região única, é possível visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte Criar uma trilha para a Conta da AWS e Criar uma trilha para uma organização no Guia do usuário do AWS CloudTrail .

Você pode entregar uma cópia dos seus eventos de gerenciamento contínuos para o bucket do Amazon S3 sem nenhum custo CloudTrail criando uma trilha. No entanto, há cobranças de armazenamento do Amazon S3. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços. Para receber informações sobre a definição de preços do Amazon S3, consulte Definição de preços do Amazon S3.

CloudTrail Armazenamentos de dados de eventos em Lake

CloudTrail O Lake permite que você execute consultas baseadas em SQL em seus eventos. CloudTrail O Lake converte eventos existentes no formato JSON baseado em linhas para o formato Apache ORC. O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de seletores de eventos avançados. Os seletores que aplicados a um armazenamento de dados de eventos controlam quais eventos persistem e estão disponíveis para consulta. Para obter mais informações sobre o CloudTrail Lake, consulte Trabalhando com o AWS CloudTrail Lake no Guia AWS CloudTrail do Usuário.

CloudTrail Os armazenamentos e consultas de dados de eventos em Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a opção de preço que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Eventos de dados do SNS em CloudTrail

Os Eventos de dados fornecem informações sobre as operações de recursos realizadas em um recurso (por exemplo, leitura ou gravação em um objeto do Amazon S3). Também são conhecidas como operações de plano de dados. Eventos de dados geralmente são atividades de alto volume. Por padrão, CloudTrail não registra eventos de dados. O histórico de CloudTrail eventos não registra eventos de dados.

Há cobranças adicionais para eventos de dados. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Você pode registrar eventos de dados para os tipos de recursos do SNS usando o CloudTrail console ou AWS CLI as operações CloudTrail da API. Para obter mais informações sobre como registrar eventos de dados em log, consulte Registrar eventos de dados com o AWS Management Console e Registrar eventos de dados com a AWS Command Line Interface no Guia do usuário do AWS CloudTrail .

A tabela a seguir lista os tipos de recursos do SNS para os quais você pode registrar eventos de dados. A coluna Tipo de evento de dados (console) mostra o valor a ser escolhido na lista Tipo de evento de dados no CloudTrail console. A coluna de valor resources.type mostra o `resources.type` valor, que você especificaria ao configurar seletores de eventos avançados usando o ou. AWS CLI CloudTrail APIs A CloudTrail coluna Dados APIs registrados em mostra as chamadas de API registradas CloudTrail para o tipo de recurso.

Tipo de evento de dados (console)	valor resources.type	Dados APIs registrados em CloudTrail
Tópico do SNS	`AWS::SNS::Topic`	• `Publish` • `PublishBatch`
Endpoint da plataforma SNS	`AWS::SNS::PlatformEndpoint`	• Publish Para obter detalhes adicionais, consulte `AdvancedEventSelector` a Referência AWS CloudTrail da API.

> **Note**
>
> O tipo de recurso SNS não `AWS::SNS::PhoneNumber` está registrado por. CloudTrail

É possível configurar seletores de eventos avançados para filtrar os campos `eventName`, `readOnly` e `resources.ARN` para registrar em log somente os eventos que são importantes para você. Para obter mais informações sobre esses campos, consulte AdvancedFieldSelector, na Referência de APIs do AWS CloudTrail .

Para obter informações sobre como registrar eventos de dados, consulte Registrar eventos de dados com o AWS Management Console e Registrar eventos de dados com o AWS CLI no Guia CloudTrail do usuário.

Eventos de gerenciamento do SNS em CloudTrail

Os eventos de gerenciamento fornecem informações sobre as operações de gerenciamento que são realizadas nos recursos do seu Conta da AWS. Também são conhecidas como operações de ambiente de gerenciamento. Por padrão, CloudTrail registra eventos de gerenciamento.

AWS O SNS registra as seguintes operações do plano de controle do SNS CloudTrail como eventos de gerenciamento.

- `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**
>
> Quando você não estiver conectado à Amazon Web Services (modo não autenticado) e `Unsubscribe`as ações `ConfirmSubscription`ou forem invocadas, elas não serão registradas. CloudTrail Por exemplo, quando você escolhe o link fornecido em uma notificação por e-mail para confirmar uma assinatura pendente em um tópico, a ação `ConfirmSubscription` é chamada no modo não autenticado. Neste exemplo, a `ConfirmSubscription` ação não seria registrada. CloudTrail

Exemplos de eventos do SNS

Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a operação de API solicitada, a data e a hora da operação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, os eventos não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra um CloudTrail evento que demonstra as **`DeleteTopic`** ações **`ListTopicsCreateTopic`**, e.


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

O exemplo a seguir mostra um CloudTrail evento que demonstra a **`Publish`** ação.

```
{
  "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"
  }
}
```

O exemplo a seguir mostra um CloudTrail evento que demonstra a **`PublishBatch`** ação.

```
{
  "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"
  }
}
```


Para obter informações sobre o conteúdo do CloudTrail registro, consulte o conteúdo do CloudTrail registro no Guia AWS CloudTrail do usuário.

Monitorando tópicos do Amazon SNS usando CloudWatch

O Amazon SNS e o Amazon CloudWatch são integrados para que você possa coletar, visualizar e analisar métricas para cada notificação ativa do Amazon SNS. Depois de configurar CloudWatch o Amazon SNS, você pode obter uma melhor visão sobre o desempenho de seus tópicos, notificações push e entregas de SMS do Amazon SNS. Por exemplo, você pode definir um alarme para enviar uma notificação por e-mail se um limite especificado para uma métrica do Amazon SNS for atingido, como `NumberOfNotificationsFailed`. Para obter uma lista de todas as métricas para as quais o Amazon SNS envia CloudWatch, consulte. Métricas do Amazon SNS Para obter mais informações

sobre notificações por push do Amazon SNS, consulte Enviar notificações por push para dispositivos móveis com o Amazon SNS.

> **Note**
>
> As métricas que você configura CloudWatch para seus tópicos do Amazon SNS são coletadas e enviadas automaticamente CloudWatch em intervalos de 1 minuto. Essas métricas são reunidas em todos os tópicos que atendem às CloudWatch diretrizes para ser ativo. Um tópico é considerado ativo CloudWatch por até seis horas a partir da última atividade (ou seja, qualquer chamada de API) no tópico.
> Não há cobrança pelas métricas do Amazon SNS relatadas em CloudWatch; elas são fornecidas como parte do serviço Amazon SNS.

Veja CloudWatch as métricas do Amazon SNS

Você pode monitorar as métricas do Amazon SNS usando o CloudWatch console, a própria interface CloudWatch de linha de comando (CLI) ou usando programaticamente a API. CloudWatch Os procedimentos a seguir mostram como acessar as métricas com o AWS Management Console.

Para visualizar métricas usando o CloudWatch console

1. Faça login no console do CloudWatch.
2. No painel de navegação, selecione Métricas.
3. Na guia Todas as métricas, escolha SNS e uma das seguintes dimensões:
   - País, Tipo de SMS
   - PhoneNumber
   - Métricas de tópico
   - Métricas sem dimensões
4. Para exibir mais detalhes, escolha um item específico. Por exemplo, se você escolher Métricas de tópico e depois escolher NumberOfMessagesPublished, o número médio de mensagens publicadas do Amazon SNS por um período de 1 minuto em todo o intervalo de tempo de 6 horas será exibido.
5. Para visualizar as métricas de uso do Amazon SNS, na guia All metrics (Todas as métricas, escolha Usage (Uso) e selecione a target Amazon SNS usage metric (métrica-alvo de uso do Amazon SNS) (por exemplo, `NumberOfMessagesPublishedPerAccount`).

Defina CloudWatch alarmes para as métricas do Amazon SNS

CloudWatch também permite definir alarmes quando um limite é atingido para uma métrica. Por exemplo, você pode definir um alarme para a métrica, NumberOfNotificationsFailed, para que, quando o número limite especificado for atingido dentro do período de amostragem, uma notificação por e-mail seja enviada para informá-lo sobre o evento.

Para definir alarmes usando o console CloudWatch

1. Faça login no AWS Management Console e abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Selecione Alarmes e clique no botão Criar alarme. Isso inicia o assistente Criar alarme.
3. Role pelas métricas do Amazon SNS para localizar aquela em que você deseja colocar um alarme. Selecione a métrica para criar um alarme e escolha Continue (Continuar).
4. Preencha os valores de Name (Nomes), Description (Descrição), Threshold (Limite) e Time (Tempo) para a métrica e escolha Continue (Continuar).
5. Escolha Alarme como o estado do alarme. Se você quiser CloudWatch enviar um e-mail quando o estado do alarme for atingido, escolha um tópico existente do Amazon SNS ou escolha Criar novo tópico de e-mail. Se você escolher Create New Email Topic (Criar novo tópico de e-mail), poderá definir o nome e os endereços de e-mail para um novo tópico. Esta lista será salva e aparecerá na caixa suspensa para alertas futuros. Escolha Continuar.

   > **Note**
   >
   > Se você escolher Create New Email Topic (Criar novo tópico de e-mail) para criar um novo tópico do Amazon SNS, os endereços de e-mail deverão ser verificados para que possam receber notificações. Os e-mails são enviados somente quando o alerta entra em um estado de alerta. Se essa alteração para estado de alerta ocorrer antes que os endereços de e-mail sejam verificados, a notificação não será recebida.

6. Nesse momento, o assistente Criar alarme lhe oferece uma oportunidade para revisar o alarme que você está prestes a criar. Se você precisar fazer alterações, use os links Editar à direita. Quando estiver satisfeito, escolha Criar alarme.

Para obter mais informações sobre uso CloudWatch e alarmes, consulte a CloudWatchdocumentação.

Métricas do Amazon SNS

O Amazon SNS envia as seguintes métricas para. CloudWatch

Namespace	Métrica	Descrição
AWS/SNS	`NumberOfMessagesPublished`	O número de mensagens publicadas nos tópicos do Amazon SNS. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma
AWS/SNS	`NumberOfNotificationsDelivered`	O número de mensagens entregues com êxito dos tópicos do Amazon SNS para endpoints de assinatura. Para obter uma tentativa de entrega bem-sucedida, o endpoint deve aceitar a assinatura da mensagem. Uma assinatura aceitará uma mensagem se a.) faltar uma política de filtro ou b.) a política de filtro incluir atributos que combinem com os atribuídos à mensagem. Se a assinatura rejeita a mensagem, a tentativa de entrega não é contada para esta métrica. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName

Namespace	Métrica	Descrição
		Estatísticas válidas: soma

Namespace	Métrica	Descrição
AWS/SNS	`NumberOfNotificationsFailed`	O número de mensagens do Amazon SNS com falha na entrega. Para o Amazon SQS, e-mail, SMS ou endpoints de push para dispositivos móveis, a métrica é incrementada em 1 quando o Amazon SNS para de tentar entregar as mensagens. Para endpoints HTTP ou HTTPS, a métrica inclui cada tentativa de entrega com falha, incluindo novas tentativas que seguem a tentativa inicial. Para todos os outros endpoints, a contagem aumenta em 1 quando a mensagem não é entregue (independentemente do número de tentativas). Essa métrica não inclui mensagens que foram rejeitadas pelo filtro de assinatura políticas. Você pode controlar o número de novas tentativas para endpoints HTTP. Para obter mais informações, consulte Novas tentativas de entrega de mensagens do Amazon SNS. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName

Namespace	Métrica	Descrição
		Estatísticas válidas: soma, média
AWS/SNS	`NumberOfNotificationsFilteredOut`	O número de mensagens que foram rejeitadas pelo filtro de assinatura políticas. Uma política de filtro rejeita uma mensagem quando os atributos de mensagem não correspondem aos atributos de política. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	O número de mensagens que foram rejeitadas por políticas de filtro de assinatura para filtragem baseada em atributos. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média

Namespace	Métrica	Descrição
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	O número de mensagens que foram rejeitadas por políticas de filtro de assinatura para filtragem baseada em carga útil. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	O número de mensagens que foram rejeitadas por políticas de filtro de assinatura por conta de atributos de mensagens são inválidos. Por exemplo, devido à formatação incorreta do atributo JSON. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média

Namespace	Métrica	Descrição
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	O número de mensagens que foram rejeitadas pelas políticas de filtro de assinatura por conta de as mensagens não terem atributos. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	O número de mensagens que foram rejeitadas por políticas de filtro de assinatura porque o corpo da mensagem era inválido para filtragem; por exemplo, corpo da mensagem JSON inválido. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média

Namespace	Métrica	Descrição
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	O número de mensagens que foram movidas para uma fila de mensagens mortas. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	O número de mensagens que não puderam ser movidas para uma fila de mensagens mortas. Unidades: contagem Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: soma, média
AWS/SNS	`PublishSize`	O tamanho das mensagens publicadas. Unidade: bytes Dimensões válidas: aplicativo PhoneNumber, plataforma e TopicName Estatísticas válidas: mínimo, máximo, média e contagem

Namespace	Métrica	Descrição
AWS/SNS	`SMSMonthToDateSpentUSD`	As cobranças que você acumulou desde o início do mês atual com o envio de mensagens SMS. Você pode definir um alarme para essa métrica para saber quando suas month-to-date cobranças estão próximas da cota mensal de gastos com SMS da sua conta. Quando o Amazon SNS determina que o envio de uma mensagem SMS pode gerar um custo que excede essa cota, ele interrompe a publicação de mensagens SMS em poucos minutos. Para obter informações sobre como configurar sua cota de gasto mensal de SMS ou para obter informações sobre como solicitar um aumento de cota de gasto com a AWS, consulte Definir preferências de mensagens SMS no Amazon SNS. Unidade: USD Dimensões válidas: nenhuma Estatísticas válidas: soma

Namespace	Métrica	Descrição
AWS/SNS	`SMSSuccessRate`	A taxa de entregas bem-sucedidas de mensagem SMS. Unidades: contagem Dimensões válidas: PhoneNumber Estatísticas válidas: soma, média, amostras de dados

Dimensões para métricas do Amazon SNS

O Amazon Simple Notification Service envia as seguintes dimensões para CloudWatch.

Dimensão	Descrição
`Application`	Filtros em objetos do aplicativo, que representam um aplicativo e um dispositivo registrados em um dos serviços de notificação por push compatíveis, como APNs o FCM.
`Application,Platform`	Filtra objetos do aplicativo e da plataforma, onde os objetos da plataforma são para os serviços de notificação push compatíve is, como APNs o FCM.
`Country`	Os filtros no país ou região de destino de uma mensagem SMS. O país ou região é representado por seu código alfa-2 ISO 3166-1.
`PhoneNumber`	Filtra pelo número de telefone quando o SMS é publicado diretamente em um número de telefone (sem um tópico).
`Platform`	Filtros em objetos da plataforma para os serviços de notificação push, como o APNs FCM.
`TopicName`	Filtra pelos nomes dos tópicos do Amazon SNS.

Dimensão	Descrição
SMSType	Os filtros no tipo da mensagem SMS. Podem ser promocionais ou transacionais.

Métricas de uso do Amazon SNS

O Amazon Simple Notification Service envia as seguintes métricas de uso para CloudWatch.

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
AWS/Usage	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	Recurso	• O número de mensagens publicadas nos tópicos do Amazon SNS em toda a sua AWS conta. • Unidades: nenhuma • Estatísticas válidas: soma
AWS/Usage	SNS	ResourceCount	ApproximateNumberOfTopics	Recurso	• O número aproximado de tópicos em sua AWS conta. • Unidades: nenhuma

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
					• Estatísticas válidas: média, mínimo, máximo, soma
AWS/Usage	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Recurso	• O número aproximado de políticas de filtro em sua conta da AWS . • Unidades: nenhuma • Estatísticas válidas: média, mínimo, máximo, soma

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
AWS/Usage	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Recurso	• O número aproximado de assinaturas pendentes em sua conta. AWS • Unidades: nenhuma • Estatísticas válidas: média, mínimo, máximo, soma

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
AWS/Usage	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber` • `DeleteTopic`	API	• O número de chamadas de API para a API do Amazon SNS seleciona da em sua AWS conta. • O conteúdo não é permitido na seção final. Unidades: nenhuma • Estatísticas válidas: soma

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

Namespace	Serviço	Métrica	Recurso	Tipo	Descrição
			• `Subscribe` • `Unsubscribe` • `UntagResource` • `VerifySMSSandboxPhoneNumber`		

Validação de conformidade para o Amazon SNS

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Governança e conformidade de segurança: esses guias de implementação de solução abordam considerações sobre a arquitetura e fornecem etapas para implantar recursos de segurança e conformidade.
- Referência de serviços qualificados para HIPAA: lista os serviços qualificados para HIPAA. Nem todos Serviços da AWS são elegíveis para a HIPAA.
- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da

AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).

- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon SNS

A resiliência no Amazon SNS é garantida por meio da utilização AWS da infraestrutura global, que gira em torno Regiões da AWS de zonas de disponibilidade. Regiões da AWS oferecem zonas de disponibilidade fisicamente separadas e isoladas conectadas por redes de baixa latência, alto rendimento e altamente redundantes. Essa arquitetura permite um failover contínuo entre as zonas de disponibilidade sem interrupção, tornando os aplicativos e bancos de dados inerentemente mais tolerantes a falhas e escaláveis em comparação com as infraestruturas tradicionais de data center. Ao usar zonas de disponibilidade, os assinantes do Amazon SNS se beneficiam de maior disponibilidade e confiabilidade, garantindo a entrega de mensagens apesar de possíveis interrupções. Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além disso, as assinaturas de tópicos do Amazon SNS podem ser configuradas com novas tentativas de entrega e filas de mensagens não entregues, permitindo o tratamento automático

de falhas transitórias e garantindo que as mensagens cheguem de forma confiável aos destinos pretendidos.

O Amazon SNS também suporta filtragem de mensagens e atributos de mensagens, o que permite adaptar estratégias de resiliência a seus casos de uso específicos, aprimorando a robustez geral de seus aplicativos.

Segurança da infraestrutura no Amazon SNS

Como um serviço gerenciado, o Amazon SNS é protegido pelos procedimentos AWS globais de segurança de rede encontrados na documentação de melhores práticas de segurança, identidade e conformidade.

Use ações de AWS API para acessar o Amazon SNS pela rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.2 ou posterior. Os clientes também devem ter suporte a pacotes de criptografia com sigilo de encaminhamento perfeito (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE).

Você deve assinar as solicitações usando um ID de chave de acesso e uma chave de acesso secreta associados a uma entidade principal do IAM. Como alternativa, você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Você pode chamar essas ações de API de qualquer local da rede, mas o Amazon SNS oferece suporte a políticas de acesso com base em recursos, que podem incluir restrições com base no endereço IP de origem. Você também pode usar as políticas do Amazon SNS para controlar o acesso de endpoints específicos ou específicos da Amazon VPC. VPCs Isso isola efetivamente o acesso à rede de determinado tópico do Amazon SNS apenas a partir da VPC específica dentro da rede da AWS . Para obter mais informações, consulte Restringir a publicação em um tópico do Amazon SNS somente de um endpoint da VPC específico.

Práticas recomendadas de segurança para o Amazon SNS

AWS fornece muitos recursos de segurança para o Amazon SNS. Analise esses recursos de segurança no contexto de sua própria política de segurança.

> **Note**
>
> As orientações desses recursos de segurança se aplicam a casos de uso e implementações comuns. Recomendamos que você examine as melhores práticas no contexto do seu caso de uso, arquitetura e modelo de ameaças específicos.

Melhores práticas preventivas

Veja a seguir as práticas recomendadas de segurança preventiva para o Amazon SNS.

Tópicos

- Verifique se os tópicos não estão acessíveis ao público geral
- Implemente o privilégio de acesso mínimo
- Use funções do IAM para aplicativos e AWS serviços que exigem acesso ao Amazon SNS
- Implemente a criptografia no lado do servidor
- Aplicar a criptografia de dados em trânsito
- Considerar o uso de endpoints da VPC para acessar o Amazon SNS
- Certificar-se de que as assinaturas não estejam configuradas para entregar a endpoints http brutos

Verifique se os tópicos não estão acessíveis ao público geral

A menos que você exija explicitamente que qualquer pessoa na Internet possa ler ou escrever em seu tópico do Amazon SNS, você deve garantir que seu tópico não seja acessível ao público (acessível por qualquer pessoa no mundo ou por qualquer usuário AWS autenticado).

- Evite criar políticas com o `Principal` definido como `""`.
- Evite usar um curinga (`*`). Em vez disso, nomeie um usuário ou usuários específicos.

Implemente o privilégio de acesso mínimo

Quando concede permissões, você decide quem as recebe, para quais tópicos as permissões se aplicam e as ações específicas de API que você deseja permitir para esses tópicos. A implementação do princípio de privilégio mínimo é importante para reduzir os riscos de segurança. Também ajuda a reduzir o efeito negativo de erros ou intenção maliciosa.

Siga o aviso de segurança padrão de concessão de privilégios mínimos. Ou seja, conceda apenas as permissões necessárias para executar uma tarefa específica. Você pode implementar o privilégio mínimo usando uma combinação de políticas de segurança pertencentes ao acesso do usuário.

O Amazon SNS usa o modelo publicador-assinante, exigindo três tipos de acesso à conta de usuário:

- Administradores: acesso para criar, modificar e excluir tópicos. Os administradores também controlam as políticas de tópicos.
- Editores: acesso ao envio de mensagens para tópicos.
- Assinantes: acesso à assinatura de tópicos.

Para obter mais informações, consulte as seções a seguir:

- Gerenciamento de identidade e acesso no Amazon SNS
- Permissões da API do Amazon SNS: referência de ações e recursos

Use funções do IAM para aplicativos e AWS serviços que exigem acesso ao Amazon SNS

Para que aplicativos ou AWS serviços, como a Amazon EC2, acessem os tópicos do Amazon SNS, eles devem usar AWS credenciais válidas em suas AWS solicitações de API. Como essas credenciais não são alternadas automaticamente, você não deve armazená-las AWS diretamente no aplicativo ou na instância. EC2

Use uma função do IAM para gerenciar credenciais temporárias para aplicações ou serviços que precisam acessar o Amazon SNS. Ao usar uma função, você não precisa distribuir credenciais de longo prazo (como nome de usuário, senha e chaves de acesso) para uma EC2 instância ou AWS serviço, como AWS Lambda. Em vez disso, a função fornece permissões temporárias que os aplicativos podem usar quando fazem chamadas para outros AWS recursos.

Para obter mais informações, consulte IAM Roles (Funções do IAM) e Common Scenarios for Roles: Users, Applications, and Services (Cenários comuns para funções: usuários, aplicações e produtos) no Guia do usuário do IAM.

Implemente a criptografia no lado do servidor

Para atenuar problemas de vazamento de dados, utilize a criptografia em repouso para criptografar as mensagens usando uma chave armazenada em um local diferente do local de armazenamento das suas mensagens. A criptografia no lado do servidor (SSE) fornece criptografia dos dados em repouso. O Amazon SNS criptografa os dados no nível da mensagem ao armazená-los e descriptografa as mensagens para você, quando você as acessa. SSE usa chaves gerenciadas em AWS Key Management Service. Quando você autentica sua solicitação e tem as permissões de acesso, não há diferença de acesso entre as filas criptografadas e não criptografadas.

Para ter mais informações, consulte Segurança dos dados do Amazon SNS com a criptografia do lado do servidor e Gerenciar chaves e custos de criptografia do Amazon SNS.

Aplicar a criptografia de dados em trânsito

É possível, mas não recomendado, publicar mensagens que não estejam criptografadas durante o trânsito usando HTTP. No entanto, quando um tópico é criptografado em repouso usando AWS KMS, é necessário usar HTTPS para publicar mensagens para garantir a criptografia em repouso e em trânsito. Embora o tópico não rejeite automaticamente mensagens HTTP, o uso de HTTPS é necessário para manter os padrões de segurança.

AWS recomenda que você use HTTPS em vez de HTTP. Ao usar HTTPS, as mensagens são criptografadas automaticamente durante o trânsito, mesmo que o tópico do SNS em si não esteja criptografado. Sem HTTPS, um invasor baseado em rede pode espionar o tráfego da rede ou manipulá-lo usando um ataque como. man-in-the-middle

Para impor somente conexões criptografadas por HTTPS, adicione a condição `aws:SecureTransport` à política do IAM anexada aos tópicos do SNS não criptografados. Isso força os editores de mensagens a usar HTTPS em vez de HTTP. Você pode usar o seguinte exemplo de política como um guia:

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

Considerar o uso de endpoints da VPC para acessar o Amazon SNS

Se você tiver tópicos com os quais é preciso interagir, mas que não devem de forma alguma ficar expostos à Internet, use VPC endpoints para limitar o acesso ao tópico apenas aos hosts dentro de uma VPC específica. Você pode usar políticas de tópicos para controlar o acesso a tópicos de endpoints específicos da Amazon VPC ou de pontos específicos. VPCs

Os endpoints da VPC do Amazon SNS fornecem duas maneiras de controlar o acesso às suas mensagens:

- É possível controlar as solicitações, os usuários ou os grupos permitidos por um VPC endpoint específico.
- Você pode controlar quais endpoints VPCs ou VPC têm acesso ao seu tópico usando uma política de tópicos.

Para ter mais informações, consulte Criar o endpoint e Criar uma política de endpoint da Amazon VPC para o Amazon SNS.

Certificar-se de que as assinaturas não estejam configuradas para entregar a endpoints http brutos

Evite configurar assinaturas para entregar a endpoints http brutos. Sempre tenha assinaturas entregando para um nome de domínio de endpoint. Por exemplo, uma assinatura configurada para entregar a um endpoint, `http://1.2.3.4/my-path`, deve ser alterada para `http://my.domain.name/my-path`.

Solução de problemas de tópicos do Amazon SNS usando AWS X-Ray

AWS X-Ray coleta dados sobre as solicitações que seu aplicativo atende e permite que você visualize e filtre dados para identificar possíveis problemas e oportunidades de otimização. Para qualquer solicitação rastreada para seu aplicativo, você pode ver informações detalhadas sobre a solicitação, a resposta e as chamadas que seu aplicativo faz para AWS recursos downstream, microsserviços, bancos de dados e web HTTP. APIs

É possível usar o X-Ray com o Amazon SNS para rastrear e analisar as mensagens que passam pela aplicação. Você pode usar o AWS Management Console para visualizar o mapa de conexões entre o Amazon SNS e outros serviços que seu aplicativo usa. Também é possível usar o console para visualizar métricas como a latência média e as taxas de falha. Para obter mais informações, consulte Amazon SNS e AWS X-Rayno Guia do desenvolvedor do AWS X-Ray .

Rastreamento ativo no Amazon SNS

Use AWS X-Ray para rastrear e analisar solicitações de usuários à medida que elas passam por seus tópicos do Amazon SNS para assinaturas de endpoint do Amazon Data Firehose, Amazon AWS LambdaSQS e HTTP/S.

Com o X-Ray, você tem uma end-to-end visão de cada solicitação, permitindo que você:

- Identifique o que está chamando seu tópico do Amazon SNS e quais serviços estão a jusante de suas assinaturas.
- Analise latências, como:
  - Tempo gasto no tópico do Amazon SNS antes do processamento.
  - Prazos de entrega para cada endpoint inscrito.

> **Important**
>
> Os tópicos do Amazon SNS com várias assinaturas podem atingir um limite de tamanho e não ser totalmente rastreados. Para obter informações sobre os limites de tamanho do documento de rastreamento, consulte as cotas do serviço de raio-X na Referência AWS geral.

Se você chamar uma API do Amazon SNS de um serviço que já foi rastreado, o Amazon SNS enviará o rastreamento adiante, mesmo que o rastreamento do X-Ray não esteja habilitado na API.

O Amazon SNS permite o rastreamento do X-Ray para tópicos comuns e FIFO. Você pode ativar o X-Ray para um tópico do Amazon SNS usando o console do Amazon SNS, a API `SetTopicAttributes` do Amazon SNS, a referência de CLI do Amazon Simple Notification Service ou o AWS CloudFormation.

Para saber mais sobre como usar o Amazon SNS com o X-Ray, consulte Amazon SNS e AWS X-Ray no Guia do desenvolvedor do AWS X-Ray .

Permissões de rastreamento ativo

Ao usar o console do Amazon SNS, o Amazon SNS tenta criar as permissões necessárias para que o tópico do Amazon SNS chame o X-Ray. A tentativa poderá ser rejeitada se você não tiver permissões suficientes para usar o console do Amazon SNS. Para ter mais informações, consulte Gerenciamento de identidade e acesso no Amazon SNS e Casos de exemplo para controle de acesso do Amazon SNS.

Ao usar a CLI, você deve configurar manualmente as permissões. Essas permissões são configuradas usando políticas de recursos. Para saber mais sobre como usar as permissões necessárias no X-Ray, consulte Amazon SNS e AWS X-Ray.

Habilitando o rastreamento ativo em um tópico do Amazon SNS usando o console AWS

Quando o rastreamento ativo é habilitado em um tópico do Amazon SNS, ele lê o ID de rastreamento, envia os dados para o cliente com base no ID de rastreamento e propaga o ID de rastreamento para serviços downstream.

1. Faça login no console do Amazon SNS.
2. Selecione um tópico ou crie um. Para obter mais detalhes sobre como criar tópicos, consulte Criar um tópico do Amazon SNS.
3. Na página Criar tópico, na seção Detalhes, selecione um tipo de tópico: FIFO ou Padrão.
    a. Insira um Nome para o tópico.
    b. (Opcional) Insira um Nome de exibição para o tópico.
4. Expanda Active tracing (Rastreamento ativo) e escolha Use active tracing (Usar rastreamento ativo).

Depois de habilitar o X-Ray para seu tópico do Amazon SNS, você pode usar o mapa do serviço X-Ray para visualizar os end-to-end rastreamentos e mapas de serviço do tópico.

Habilitando o rastreamento ativo em um tópico do Amazon SNS usando o SDK AWS

O exemplo de código a seguir mostra como habilitar o rastreamento ativo em um tópico do Amazon SNS usando AWS o SDK for Java.

```
public static void enableActiveTracing(SnsClient snsClient, String topicArn) {

        try {

            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName("TracingConfig")
                .attributeValue("Active")
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
 request.attributeValue());

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

Habilitando o rastreamento ativo em um tópico do Amazon SNS usando a CLI AWS

O exemplo de código a seguir mostra como habilitar o rastreamento ativo em um tópico do Amazon SNS usando a AWS CLI.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Habilitando o rastreamento ativo em um tópico do Amazon SNS usando AWS CloudFormation

A AWS CloudFormation pilha a seguir mostra como habilitar o rastreamento ativo em um tópico do Amazon SNS.

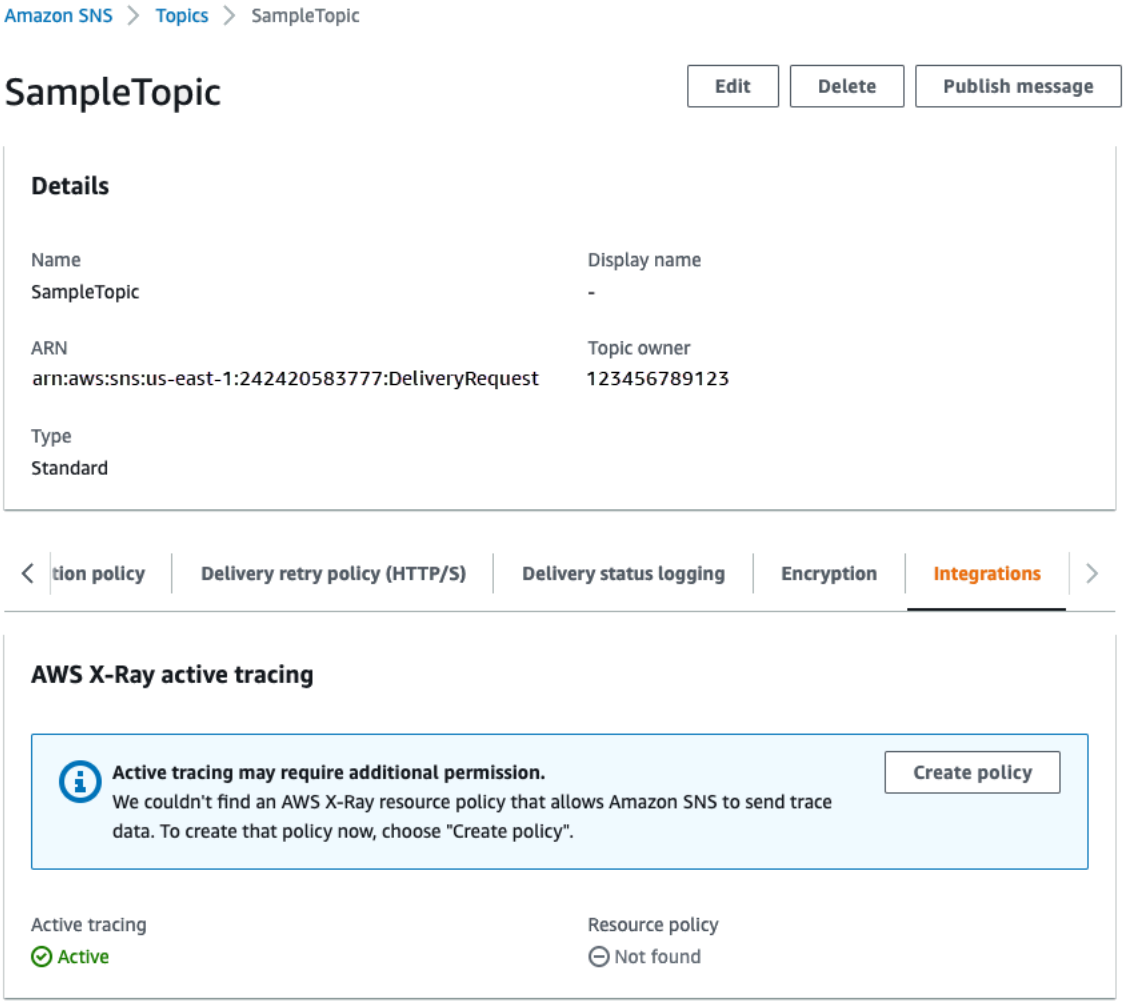
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Verificar se o rastreamento ativo está habilitado para o tópico

Você pode usar o console do Amazon SNS para verificar se o rastreamento ativo está habilitado para o tópico ou para verificar quando a política de recursos não foi adicionada.

1. Faça login no console do Amazon SNS.
2. No painel de navegação à esquerda, selecione Tópicos.
3. Na página Topics (Tópicos), escolha um tópico.
4. Escolha a guia Integrações.

   Quando o rastreamento ativo está habilitado, um ícone Active (Ativo) é exibido.
5. Se você habilitou o rastreamento ativo e não vê que a política de recursos foi adicionada, escolha Criar política para adicionar as outras permissões necessárias.



Testar rastreamento ativo

1. Faça login no console do Amazon SNS.
2. Criar um tópico do Amazon SNS. Para obter detalhes sobre como fazer isso, consulte Para criar um tópico usando o AWS Management Console.
3. Expanda Active tracing (Rastreamento ativo) e escolha Use active tracing (Usar rastreamento ativo).
4. Publique uma mensagem no tópico do Amazon SNS. Para obter detalhes sobre como fazer isso, consulte Para publicar mensagens nos tópicos do Amazon SNS usando o AWS Management Console.
5. Use o mapa do serviço X-Ray para visualizar os end-to-end rastreamentos e os mapas de serviço do tópico.


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

Histórico de documentação do Amazon SNS

A tabela a seguir descreve alterações recentes no Guia do desenvolvedor do Amazon Simple Notification Service.

Às vezes, os recursos do serviço são lançados de forma incremental nas AWS regiões em que um serviço está disponível. Atualizamos esta documentação apenas para a primeira versão. Não fornecemos informações sobre a disponibilidade da região nem anunciamos lançamentos subsequentes da região. Para obter informações sobre a disponibilidade de recursos de serviço na região e para assinar notificações sobre atualizações, consulte O que há de novo em AWS? .

Alteração	Descrição	Data
Foi adicionado suporte para endpoints de pilha dupla (IPv4 e) IPv6	O Amazon SNS agora suporta solicitações IPv6 de API, permitindo que os clientes se conectem a endpoints públicos usando IPv4 IPv6, ou pilha dupla.	3 de abril de 2025
O Amazon SNS adicionou o `FifoThroughputScope` atributo para os tópicos FIFO do Amazon SNS	O Amazon SNS suporta o FifoThroughputScope atributo, que especifica a cota de taxa de transferência e o comportamento de desduplicação a ser aplicado ao tópico FIFO. Os valores válidos são `Topic` ou `MessageGroup` .	21 de janeiro de 2025
Atualizações nas políticas gerenciadas `AmazonSNSFullAccess` e `AmazonSNSReadOnlyAccess`	O Amazon SNS adicionou novas permissões ao AmazonSNSFullAccess and AmazonSNSReadOnlyAccess políticas gerenciadas, que permitem acesso adicional ao Amazon SNS por meio do. AWS Management Console	24 de setembro de 2024

Integração com o Amazon SNS AWS End User Messaging SMS para entrega de mensagens SMS	O Amazon SNS oferece suporte a novos recursos, como gerenciamento de recursos de SMS, mensagens bidirecionais, permissões granulares de recursos, regras de bloqueio de países e cobrança centralizada para todas as mensagens AWS SMS sem fazer alterações nas configurações ou na rede AWS global de SMS usada pelo Amazon SNS.	24 de setembro de 2024
Suporte Oeste do Canadá (Calgary) para tópicos do FIFO	O Amazon SNS oferece suporte ao tópico FIFO no Oeste do Canadá (Calgary).	28 de março de 2024
Suporte por SMS do Amazon SNS em cinco novas regiões	Foi adicionado suporte às seguintes: Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Melbourne), Oriente Médio (Emirados Árabes Unidos), Europa (Zurique) e Europa (Espanha).	8 de fevereiro de 2024
Compatível para HTTP v1 do Firebase Cloud Messaging (FCM)	O Amazon SNS oferece suporte às credenciais do FCM v1.	18 de janeiro de 2024
Amazon SNS oferece suporte a SMS na região Ásia-Pacífico (Jacarta)	O Amazon SNS oferece suporte a mensagens SMS na região Ásia-Pacífico (Jacarta).	14 de dezembro de 2023

AWS CloudFormation suporte para configuração de tópicos do `DeliveryStatusLogging` Amazon SNS	AWS CloudFormation o suporte está disponível para configuração `DeliveryStatusLogging` durante a criação ou atualização de tópicos do Amazon SNS.	7 de dezembro de 2023
Novos operadores de filtragem de mensagens adicionados	Agora você pode usar os operadores de correspondência de sufixos, igual-ignorar-capitalização e OR ao filtrar mensagens do Amazon SNS.	16 de novembro de 2023
Suporte adicionado para arquivamento e reprodução de mensagens.	Os proprietários de tópicos podem arquivar mensagens em um tópico por até 365 dias. Os assinantes do tópico podem reproduzir as mensagens arquivadas em um endpoint inscrito com o objetivo de recuperar mensagens em virtude de uma falha em uma aplicação subsequente ou para replicar o estado de uma aplicação existente.	26 de outubro de 2023
Suporte adicionado para inscrever uma fila padrão em um tópico FIFO	Você pode inscrever uma fila FIFO ou uma fila padrão do Amazon SQS em um tópico FIFO do Amazon SNS. Somente as filas FIFO do Amazon SQS garantem que as mensagens sejam recebidas em ordem e sem duplicatas.	14 de setembro de 2023

Suporte de SMS adicionado para Israel (Tel Aviv)	O Amazon SNS SMS agora é compatível na região de Israel (Tel Aviv).	28 de agosto de 2023
Suporte ao rastreamento ativo do X-Ray adicionado a tópicos FIFO	Anteriormente suportado apenas com tópicos padrão do Amazon SNS, AWS X-Ray agora rastreia e analisa as solicitações dos usuários à medida que elas percorrem seus tópicos de FIFO até suas assinaturas de endpoint Amazon Data Firehose, AWS Lambda Amazon SQS e HTTP/S.	31 de maio de 2023
Suporte avançado para o cabeçalho Content-Type	É possível definir o cabeçalho Content-Type na política de solicitação para especificar o tipo de mídia da notificação.	23 de março de 2023
Adição de suporte ao rastreamento ativo	AWS X-Ray rastreia e analisa as solicitações dos usuários à medida que elas percorrem seus tópicos padrão do Amazon SNS até suas assinaturas de endpoint do Amazon Data Firehose, AWS Lambda Amazon SQS e HTTP/S.	8 de fevereiro de 2023
Registro de ID de remetente de Singapura	Instruções adicionadas para registrar o remetente em IDs Cingapura.	10 de janeiro de 2023

Filtragem de mensagens baseada em carga útil	A filtragem baseada em carga útil permite filtrar mensagens com base na carga útil da mensagem e evitar os custos associados ao processamento de dados indesejados.	22 de novembro de 2022
SHA256 algoritmo de hash adicionado para assinatura de mensagens do Amazon SNS	Support adicionado para algoritmo de SHA256 hash ao usar a assinatura de mensagens do Amazon SNS.	15 de setembro de 2022
Adição de mais regiões às mensagens SMS	O Amazon SNS oferece suporte a mensagens SMS nas seguintes regiões: África (Cidade do Cabo), Ásia-Pacífico (Osaka), Europa (Milão) e AWS GovCloud (Leste dos EUA).	9 de setembro de 2022
Adição de compatibilidade com a proteção de dados de mensagens	A proteção de dados de mensagens protege os dados publicados em seus tópicos do Amazon SNS usando políticas de proteção de dados para auditar e bloquear as informações confidenciais que se movem entre aplicativos AWS ou serviços.	8 de setembro de 2022
Novo processo de registro de números de chamada gratuita	Suporte adicionado para envio de mensagens do Amazon SNS usando números de telefone de chamada gratuita (TFN) para destinatários dos Estados Unidos.	1º de agosto de 2022

Suporte para controle de acesso baseado em atributos (ABAC)	Suporte adicionado para controle de acesso baseado em atributos (ABAC) para ações de API, incluindo `Publish` e `PublishBatch` . O ABAC é uma estratégia de autorização que define permissões de acesso com base em tags que podem ser anexadas a recursos do IAM, como usuários e funções do IAM, e a AWS recursos, como tópicos do Amazon SNS, para simplificar o gerenciamento de permissões.	10 de janeiro de 2022
Suporte para autenticação baseada em token da Apple para notificações push	É possível autorizar o Amazon SNS a enviar notificações por push para sua aplicação iOS ou macOS fornecendo informações que identifiquem você como desenvolvedor da aplicação.	28 de outubro de 2021
Novos remetentes de mensagens SMS são colocados na sandbox de SMS	A sandbox de SMS existe para ajudar a evitar fraudes e abusos e para ajudar a proteger sua reputação como remetente. Enquanto sua AWS conta estiver na sandbox de SMS, você poderá enviar mensagens SMS somente para números de telefone de destino verificados.	1º de junho de 2021

Novos remetentes de mensagens SMS são colocados na sandbox de SMS	A sandbox de SMS existe para ajudar a evitar fraudes e abusos e para ajudar a proteger sua reputação como remetente. Enquanto sua AWS conta estiver na sandbox de SMS, você poderá enviar mensagens SMS somente para números de telefone de destino verificados.	1º de junho de 2021
Novos atributos para enviar mensagens SMS para destinatários na Índia	Dois novos atributos,ID da entidadeeID de modeloAgora são necessários para enviar mensagens SMS para destinatários na Índia.	22 de abril de 2021
Atualizações para os operadores de filtragem	Um novo operador,`cidr`, está disponível para correspon dência de endereços IP de origem de mensagem e sub-redes. Agora você também pode verificar a ausência de uma chave de atributo, e usar um prefixo com o`anything-but` operador para correspon dência de string de atributo.	7 de abril de 2021

Finalizando o suporte para códigos longos P2P para destinos dos EUA	A partir de 1º de junho de 2021, os provedores de telecomunicações dos EUA não oferecem mais suporte ao uso de códigos longos person-to-person (P2P) para comunicações application-to-person (A2P) com destinos nos EUA. Em vez disso, você pode usar códigos curtos, números de ligação gratuita ou um novo tipo de número de origem chamado10DLC.	16 de fevereiro de 2021
Support para métricas de 1 minuto da Amazon CloudWatch	A CloudWatch métrica de 1 minuto para o Amazon SNS agora está disponível em AWS todas as regiões.	28 de janeiro de 2021
Suporte para endpoints do Amazon Data Firehose	Você pode assinar fluxos de entrega do Firehose para tópicos do SNS. Isso permite que você envie notificações para endpoints de arquivamento e análise, como buckets do Amazon Simple Storage Service (Amazon S3), tabelas do Amazon Redshift, Amazon Service (Service) e muito mais. OpenSearch OpenSearch	12 de janeiro de 2021
Os números de origem estão disponíveis	Você pode usar números de origem ao enviar mensagens de texto (SMS).	23 de outubro de 2020

Suporte para tópicos FIFO do Amazon SNS	Para integrar aplicativos distribuídos que exigem consistência de dados quase em tempo real, você pode usar tópicos FIFO (first in, first out) do Amazon SNS com filas FIFO do Amazon SQS.	22 de outubro de 2020
A biblioteca do cliente em versão ampliada do Amazon SNS para Java está disponível	Você pode usar essa bibliotec a para publicar mensagens grandes do Amazon SNS.	25 de agosto de 2020
O SSE está disponível nas regiões da China	A criptografia no lado do servidor (SSE) para o Amazon SNS está disponível nas regiões da China.	20 de janeiro de 2020
Support para usar DLQs para capturar mensagens não entregues	É possível usar uma fila de mensagens mortas (DLQ) do Amazon SQS com uma assinatura do Amazon SNS para capturar mensagens que não podem ser entregues.	14 de novembro de 2019
Support para especificar valores de APNs cabeçalho personalizados	Você pode especificar um valor de APNs cabeçalho personalizado.	18 de outubro de 2019
Support para o campo de cabeçalho `apns-push-type` " para APNs	Você pode usar o campo de `apns-push-type` cabeçalho para enviar notificações móveis APNs.	10 de setembro de 2019
Support para solução de problemas de tópicos usando AWS X-Ray	É possível usar o X-Ray para solucionar problemas com mensagens que passam pelos tópicos do SNS.	24 de julho de 2019

Suporte para correspondência de chave de atributo usando o operador `exists`	É possível usar o operador `exists` para verificar se uma mensagem recebida tem um atributo cuja chave está listada na política de filtro.	5 de julho de 2019
Suporte para correspondência “tudo, exceto” de vários valores numéricos	Além de várias strings, o Amazon SNS permite a correspondência “tudo, exceto” de vários valores numéricos.	5 de julho de 2019
As notas de release do Amazon SNS estão disponíveis como um feed RSS	Seguindo o título desta página (Histórico de documentação), escolhaRSS.	22 de junho de 2019

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.