aws

Guía para desarrolladores

Amazon Simple Notification Service

Amazon Simple Notification Service: Guía para desarrolladores

Table of Contents

¿Qué es Amazon SNS?

Amazon Simple Notification Service (Amazon SNS) es un servicio totalmente gestionado que proporciona la entrega de mensajes desde los editores (productores) a los suscriptores (consumidores). Los publicadores se comunican de forma asíncrona con los suscriptores mediante el envío mensajes a un tema, que es un punto de acceso lógico y un canal de comunicación.

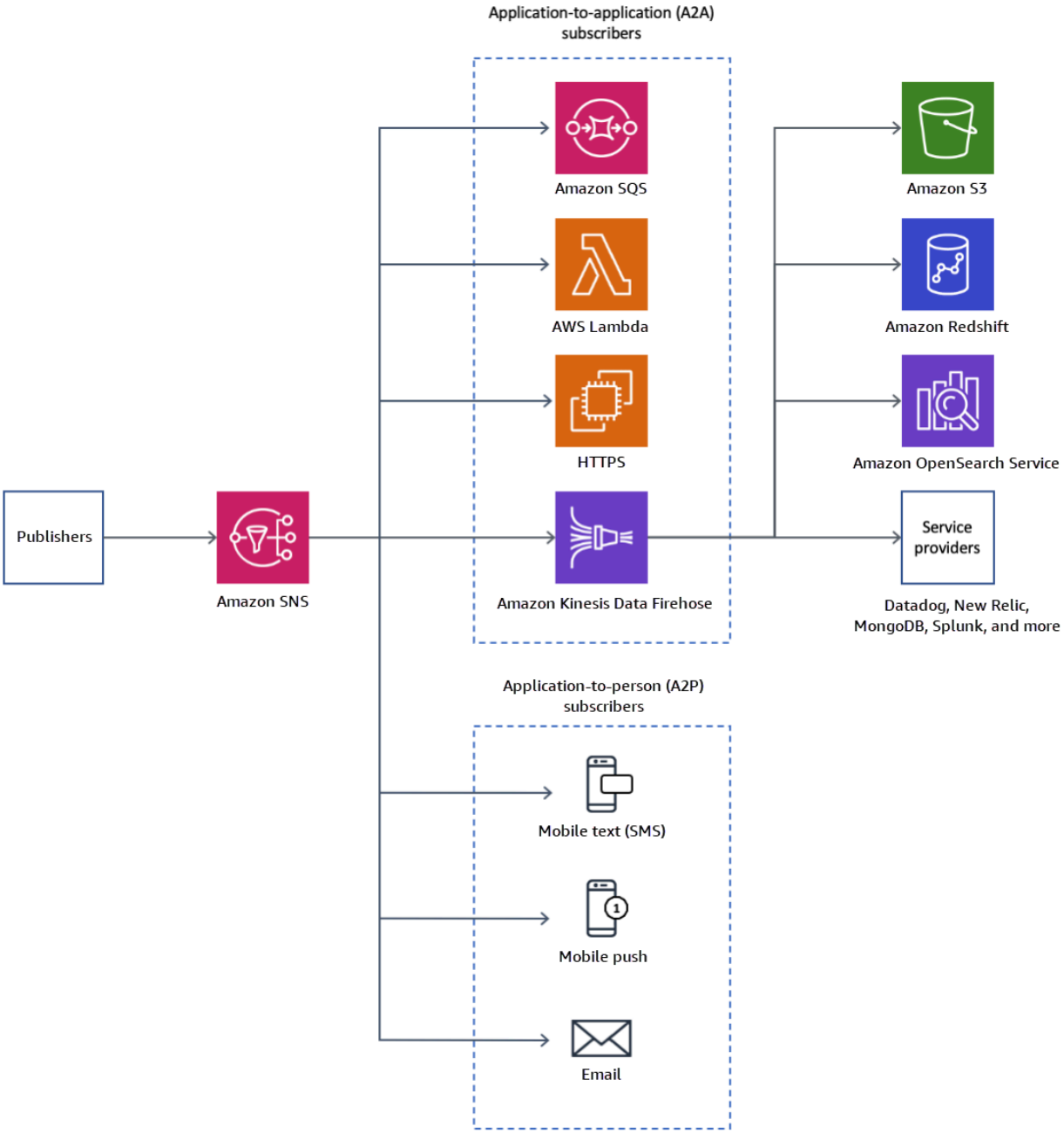
Funcionamiento

En las redes sociales, los editores envían mensajes a un tema, que actúa como canal de comunicación. El tema actúa como un punto de acceso lógico, lo que garantiza que los mensajes se entreguen a varios suscriptores en diferentes plataformas.

Los suscriptores de un tema de SNS pueden recibir mensajes a través de distintos puntos de conexión, según su caso de uso, por ejemplo:

- Amazon SQS
- Lambda
- Puntos de enlace HTTP (S)
- Correo electrónico
- Notificaciones push en móviles
- Mensajes de texto (SMS) móviles
- Amazon Data Firehose
- Proveedores de servicios (por ejemplo, Datadog, MongoDB, Splunk)

SNS admite la mensajería Application-to-Application (A2A) y Application-to-Person (A2P), lo que brinda flexibilidad para enviar mensajes entre diferentes aplicaciones o directamente a teléfonos móviles, direcciones de correo electrónico y más.



Acceso a Amazon SNS

Puede acceder a Amazon SNS y gestionarlo a través de la consola o AWS CLI AWS SDKs, según el método de interacción que prefiera. La consola ofrece una interfaz gráfica para tareas básicas,

mientras que SDKs proporciona capacidades avanzadas de configuración y automatización para casos de uso más complejos. AWS CLI

- En la consola de Amazon SNS, se ofrece una interfaz de usuario conveniente para crear temas y suscripciones, enviar y recibir mensajes, y monitorear eventos y registros.
- El AWS Command Line Interface (AWS CLI) le proporciona acceso directo a la API de Amazon SNS para casos de uso avanzados de configuración y automatización. Para obtener más información, consulte Uso de Amazon SNS con la AWS CLI.
- AWS se proporciona SDKs en varios idiomas. Para obtener más información, consulte SDKs los kits de herramientas.

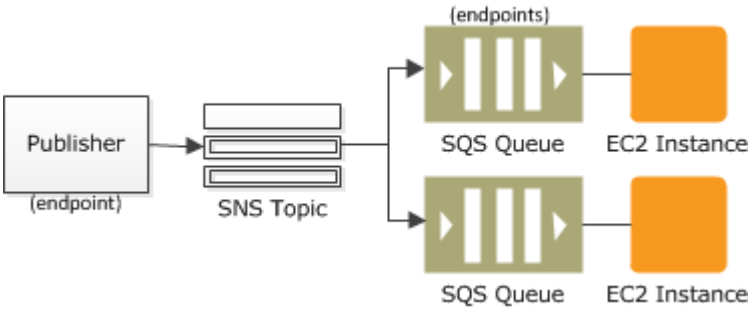
Situaciones comunes de Amazon SNS

Utilice estos escenarios habituales de Amazon SNS para implementar arquitecturas escalables basadas en eventos y garantizar una comunicación fiable en tiempo real entre las aplicaciones y los usuarios.

Integración de aplicaciones

El escenario de distribución ramificada se produce cuando un mensaje publicado en un tema de SNS se replica y se envía a varios puntos de conexión, como flujos de entrega de Firehose, colas de Amazon SQS, puntos de conexión HTTP(S) y funciones de Lambda. De este modo se permite un procesamiento asíncrono paralelo.

Por ejemplo, puede desarrollar una aplicación que publique un mensaje en un tema del SNS cada vez que se realice un pedido de un producto. Después, las colas de SQS que están suscritas a este tema de SNS reciben notificaciones idénticas del nuevo pedido. Una instancia de servidor de Amazon Elastic Compute Cloud (Amazon EC2) conectada a una de las colas de SQS puede gestionar el procesamiento o la tramitación del pedido. Además, puedes adjuntar otra instancia de EC2 servidor de Amazon a un almacén de datos para analizar todos los pedidos recibidos.



También puede utilizar la distribución ramificada para replicar los datos enviados a su entorno de producción con su entorno de prueba. Si profundizamos en el ejemplo anterior, puede suscribir otra cola de SQS al mismo tema de SNS para los nuevos pedidos que entren. A continuación, si asocia esta nueva cola de SQS a su entorno de prueba, puede seguir mejorando y probando su aplicación utilizando los datos recibidos desde su entorno de producción.

> **Important**
>
> Tenga en cuenta la privacidad y la seguridad de los datos de producción antes de enviarlos a su entorno de prueba.

Para obtener más información, consulte los siguientes recursos:

- Distribución ramificada a los flujos de entrega de Firehose
- Distribución ramificada de las notificaciones de Amazon SNS a las funciones de Lambda para su procesamiento automatizado
- Distribución ramificada de notificaciones de Amazon SNS a colas de Amazon SQS para su procesamiento asíncrono
- Distribución ramificada de notificaciones de Amazon SNS a puntos de conexión HTTPS
- Informática basada en eventos con Amazon SNS AWS y servicios de cómputo, almacenamiento, bases de datos y redes

Alertas de aplicación

Las alertas de la aplicación y del sistema son notificaciones que se desencadenan mediante umbrales predeterminados. Amazon SNS puede enviar estas notificaciones a usuarios especificados a través de SMS y correo electrónico. Por ejemplo, puede recibir una notificación inmediata cuando se produzca un evento, como un cambio específico en su grupo de Amazon EC2 Auto Scaling, la carga de un nuevo archivo en un bucket de Amazon S3 o el incumplimiento de un umbral métrico en Amazon CloudWatch. Para obtener más información, consulte Configuración de las notificaciones de Amazon SNS en la Guía CloudWatch del usuario de Amazon.

Notificaciones de usuario

Amazon SNS puede enviar mensajes de correo electrónico push y mensajes de texto (mensajes SMS) a personas o grupos. Por ejemplo, puede enviar confirmaciones de pedidos de comercio

electrónico como notificaciones de usuario. Para obtener más información sobre el uso de Amazon SNS para enviar mensajes SMS, consulte Mensajería de texto móvil con Amazon SNS.

Notificaciones push en móviles

Las notificaciones de inserción en móviles le permiten enviar mensajes directamente a aplicaciones móviles. Por ejemplo, puede usar Amazon SNS para enviar notificaciones de actualización a una aplicación. El mensaje de notificación puede incluir un enlace para descargar e instalar la actualización. Para obtener más información sobre el uso de Amazon SNS para enviar mensajes de notificaciones push, consulte Envío de notificaciones push para móvil con Amazon SNS.

Precios de Amazon SNS

Amazon SNS no tiene costos iniciales. El pago se basa en la cantidad de mensajes que publique, la cantidad de notificaciones que envíe y cualquier llamada de API adicional para administrar temas y suscripciones. Los precios de entrega varían según el tipo de punto de enlace. Puede comenzar sin costo con el nivel gratuito de Amazon SNS. Para obtener información, consulte Worldwide SMS Pricing.

Características y funciones de Amazon SNS

Amazon SNS ofrece un conjunto integral de características diseñadas para mejorar la mensajería entre las aplicaciones y los usuarios. Estas características permiten una comunicación fluida, una entrega segura de los mensajes y una sólida administración de los mensajes, lo que garantiza una alta disponibilidad, durabilidad y flexibilidad para una amplia variedad de casos de uso de mensajería.

Application-to-application mensajería

> Una pplication-to-application mensajería admite suscriptores como las transmisiones de entrega de Amazon Data Firehose, las funciones de Lambda, las colas de Amazon SQS, HTTP/S los puntos de conexión y las canalizaciones de Event Fork. AWS Esto permite una entrega de mensajes eficaz en arquitecturas basadas en eventos.

Application-to-person notifications

> Una pplication-to-person notificación proporciona notificaciones de usuario a los suscriptores, como aplicaciones móviles, números de teléfono móvil y direcciones de correo electrónico.

Temas estándar y FIFO

Los temas FIFO garantizan una ordenación, agrupación y desduplicación estrictas de los mensajes, lo que permite la suscripción de colas FIFO y estándar para el procesamiento de mensajes. Los temas estándar se utilizan cuando la ordenación y la posible duplicación de los mensajes no fundamentales, y son compatibles con todos los protocolos de entrega para casos de uso más amplios.

Durabilidad de los mensajes

En Amazon SNS, se utiliza una serie de estrategias que funcionan en conjunto para proporcionar durabilidad a los mensajes:

- Los mensajes publicados se almacenan en varios servidores y centros de datos separados por zona geográfica.
- Si no se dispone de un punto de enlace suscrito, Amazon SNS ejecuta una política de reintentos de entrega.
- Para conservar los mensajes que no se entreguen antes de que finalice la política de reintento de entrega, puede crear una cola de mensajes fallidos.

Archivo, reproducción y análisis de mensajes

Puede archivar mensajes con Amazon SNS de distintas maneras como suscribir flujos de entrega de Firehose a temas de SNS, lo que le permite enviar notificaciones a puntos de conexión de análisis como buckets de Amazon Simple Storage Service (Amazon S3), tablas de Amazon Redshift, etc. Además, los temas FIFO de Amazon SNS admiten el archivo y la reproducción de mensajes como un archivo de mensajes local y sin código que permite a los propietarios de los temas almacenar (o archivar) los mensajes dentro del tema. Los suscriptores de los temas pueden recuperar (o reproducir) los mensajes archivados devueltos a un punto de conexión suscrito. Para obtener más información, consulte Archivo y reproducción de mensajes de Amazon SNS para temas FIFO.

Atributos de mensajes

Atributos de mensajes de Amazon SNS le permite proporcionar cualquier metadato arbitrario sobre el mensaje.

Filtrado de mensajes

De forma predeterminada, cada suscriptor recibe todos los mensajes publicados en el tema. Para recibir un subconjunto de los mensajes, un suscriptor debe asignar una política de filtro a la suscripción del tema. Un suscriptor también puede definir el alcance de la política de filtrado para

habilitar el filtrado basado en cargas o en atributos. El valor predeterminado para el alcance de la política de filtrado es `MessageAttributes`. Cuando los atributos del mensaje entrante coinciden con los atributos de la política de filtro, el mensaje se entrega al punto de enlace suscrito. De lo contrario, el mensaje se filtra. Cuando el alcance de la política de filtrado es `MessageBody`, los atributos de la política de filtrado se comparan con la carga. Para obtener más información, consulte Filtrado de mensajes.

Seguridad de mensajes

El cifrado del lado del servidor protege el contenido de los mensajes que se almacenan en los temas de Amazon SNS mediante las claves de cifrado proporcionadas por. AWS KMS Para obtener más información, consulte También the section called “Protección de los datos con cifrado del servidor” puede establecer una conexión privada entre Amazon SNS y su nube privada virtual (VPC). para obtener más información, consulte. the section called “Protección del tráfico con puntos de conexión de VPC”

AWS servicios que se utilizan habitualmente con Amazon SNS

Integre Amazon SNS con varios Nube de AWS servicios para impulsar la gestión de mensajes, mejorar el control de acceso, permitir el procesamiento basado en eventos y automatizar los recursos. Esta integración optimiza el rendimiento, refuerza la seguridad y agiliza las operaciones.

Amazon CloudWatch

Amazon CloudWatch proporciona supervisión y observabilidad para Amazon SNS, lo que le ayuda a realizar un seguimiento de la entrega de mensajes, detectar anomalías y solucionar problemas. Con CloudWatch puede:

- Supervise las métricas de Amazon SNS, como la cantidad de mensajes publicados, entregados o fallidos en todos los temas y suscripciones.
- Configure CloudWatch alarmas para activar acciones automatizadas cuando las métricas de Amazon SNS superen los umbrales predefinidos, como por ejemplo una gran cantidad de errores de entrega o limitaciones.
- Utilice CloudWatch los registros para capturar el estado de entrega de Amazon SNS de los mensajes enviados a los puntos de enlace HTTP/S, Lambda y Amazon SQS para su depuración y auditoría.

Para obtener más información, consulte Supervisión de temas de Amazon SNS mediante CloudWatch.

Amazon SQS

Amazon SQS es un servicio de cola de mensajes totalmente gestionado que permite una comunicación segura, duradera y escalable entre componentes de software distribuidos. Ayuda a disociar la arquitectura de las aplicaciones al almacenar los mensajes en búfer, garantizar una entrega fiable y evitar los fallos del sistema debidos a la pérdida de mensajes. Amazon SQS se integra con Amazon SNS de las siguientes maneras:

- Colas de cartas sin salida: Amazon SNS puede redirigir los mensajes que no se pueden entregar a una cola de Amazon SQS para solucionar problemas y volver a procesarlos.
- Suscripciones temáticas: puede suscribir una cola de Amazon SQS a un tema de Amazon SNS, lo que permite a Amazon SNS distribuir los mensajes a varios consumidores mediante Amazon SQS.
- Compatibilidad con colas FIFO: las colas FIFO de Amazon SQS se pueden suscribir a los temas de FIFO de Amazon SNS, lo que garantiza un orden estricto de los mensajes y un procesamiento único. Las colas estándar de Amazon SQS también pueden suscribirse a temas de Amazon SNS, pero no garantizan la entrega de los mensajes ordenados ni la deduplicación.

AWS CloudFormation

AWS CloudFormation automatiza el aprovisionamiento y la administración de AWS los recursos, incluidos los temas y las suscripciones de Amazon SNS, mediante la infraestructura como código (IaC). Con AWS CloudFormation, puede:

- Defina los temas, las suscripciones y los permisos de Amazon SNS en una plantilla reutilizable con control de versiones.
- Garantice un despliegue uniforme de los recursos de Amazon SNS en varias Cuentas de AWS regiones.
- Actualice o modifique las configuraciones de Amazon SNS mediante conjuntos de cambios sin intervención manual.

Para obtener más información, consulte la Guía del usuario de AWS CloudFormation.

AWS CloudTrail

CloudTrail proporciona visibilidad de la actividad de las API para Amazon SNS, lo que le ayuda a supervisar y auditar el acceso a los temas, suscripciones y mensajes de Amazon SNS. Con CloudTrail, puede:

- Realice un seguimiento de las llamadas a la API realizadas a Amazon SNS, incluidas las personas que accedieron a los temas, las suscripciones y los permisos o los modificaron.

- Detecte actividades no autorizadas o inesperadas mediante el análisis de los registros con fines de seguridad y conformidad.
- Intégrelo con Amazon CloudWatch o AWS Security Hub cree alertas basadas en acciones inusuales de Amazon SNS.

Para obtener más información, consulte la Registro de llamadas a la API de AWS SNS mediante AWS CloudTrail.

AWS Lambda

AWS Lambda es un servicio de computación sin servidor que ejecuta automáticamente el código en respuesta a eventos, lo que elimina la necesidad de aprovisionar o administrar servidores. Le permite crear aplicaciones basadas en eventos que se escalan automáticamente y se ejecutan en un entorno informático de alta disponibilidad.

Amazon SNS se integra con Lambda al permitirle suscribir una función de Lambda a un tema de Amazon SNS. Cuando un tema de Amazon SNS recibe un mensaje, puede activar la función Lambda, lo que permite el procesamiento en tiempo real, la automatización y la ejecución de la lógica de la aplicación. Esta integración se suele utilizar para:

- Procesamiento basado en eventos: active automáticamente funciones en respuesta a los mensajes de Amazon SNS.
- Transformación de datos: modifique o filtre los mensajes de Amazon SNS antes de reenviarlos a otros servicios.
- Flujos de trabajo automatizados: procese las notificaciones para alertas de aplicaciones, monitoreo del sistema u organización de eventos.

AWS Identity and Access Management (IAM)

IAM proporciona un control de acceso seguro a los AWS recursos, lo que le permite administrar quién puede acceder a sus temas de Amazon SNS, qué acciones pueden realizar y en qué condiciones. Con IAM, puede:

- Autentique a los usuarios y los servicios antes de que puedan interactuar con los temas de Amazon SNS.
- Defina permisos detallados para especificar los temas de Amazon SNS en los que los usuarios o los roles pueden publicar, suscribirse o administrar.
- Utilice políticas basadas en la identidad para aplicar las mejores prácticas de seguridad, como restringir el acceso a direcciones IP o condiciones específicas Cuentas de AWS.

Para obtener más información, consulte Uso de políticas basadas en identidades con Amazon SNS.

AWS Key Management Service (AWS KMS)

AWS KMS mejora la seguridad de Amazon SNS al habilitar el cifrado del lado del servidor (SSE) para garantizar la confidencialidad de los mensajes. Con AWS KMS, puede:

- Cifre los mensajes de Amazon SNS en reposo AWS mediante claves de cifrado administradas por el cliente o administradas por el cliente (). CMKs
- Controle el acceso a los temas de Amazon SNS definiendo políticas clave detalladas que restrinjan quién puede publicar o suscribirse.
- Garantice el cumplimiento de los requisitos normativos y de seguridad mediante una auditoría exhaustiva del uso de las claves. AWS CloudTrail

Para obtener más información, consulte Administración de las claves de cifrado y los costos de Amazon SNS.

AWS X-Ray

X-Ray proporciona rastreo para Amazon SNS, lo que le ayuda a analizar y depurar el flujo de mensajes a través de su arquitectura basada en eventos. Con X-Ray, puede:

- Rastree la entrega de mensajes de Amazon SNS en varios puntos de Servicios de AWS enlace, como Lambda, Amazon SQS y HTTP/S.
- Identifique los cuellos de botella de la latencia visualizando el tiempo que tardan los mensajes en publicarse, entregarse y procesarse.
- Detecta errores y reintentos en los flujos de mensajes de Amazon SNS para solucionar problemas de entregas fallidas o tiempos de procesamiento lentos.

Para obtener más información, consulte Rastreo activo en Amazon SNS.

Uso de Amazon SNS con un SDK AWS

AWS Los kits de desarrollo de software (SDKs) están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ ejemplos de código
AWS CLI	AWS CLI ejemplos de código
AWS SDK para Go	AWS SDK para Go ejemplos de código
AWS SDK para Java	AWS SDK para Java ejemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK para .NET	AWS SDK para .NET ejemplos de código
AWS SDK para PHP	AWS SDK para PHP ejemplos de código
Herramientas de AWS para PowerShell	Herramientas de AWS para PowerShell ejemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) ejemplos de código
AWS SDK para Ruby	AWS SDK para Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Para obtener ejemplos específicos de Amazon SNS, consulte Ejemplos de código para Amazon SNS mediante AWS SDKs.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

Creación de un tema de Amazon SNS y publicación de mensajes

En este tema se describen los pasos básicos para administrar los recursos de Amazon SNS, con especial hincapié en los temas, las suscripciones y la publicación de mensajes. En primer lugar, configurará los permisos de acceso necesarios para Amazon SNS, asegurándose de que dispone de los permisos correctos para crear y administrar recursos de Amazon SNS. A continuación, creará un nuevo tema de Amazon SNS, que servirá como centro neurálgico para administrar y entregar los mensajes a los suscriptores. Tras crear el tema, procederá a crear una suscripción a este tema, lo que permitirá que puntos de conexión específicos reciban los mensajes publicados en él.

Una vez que el tema y la suscripción estén listos, publicará un mensaje en el tema y observará cómo Amazon SNS entrega el mensaje de manera eficaz a todos los puntos de conexión suscritos. Por último, aprenderá a eliminar tanto la suscripción como el tema, completando así el ciclo de vida de los recursos de Amazon SNS que ha administrado. Este enfoque le proporciona una comprensión clara de las operaciones básicas de Amazon SNS, así como las habilidades prácticas necesarias para administrar los flujos de trabajo de mensajería mediante la consola de Amazon SNS.

Configuración del acceso para Amazon SNS

Para poder usar Amazon SNS por primera vez, debe completar los pasos siguientes.

Cree un Cuenta de AWS usuario de IAM

Para acceder a cualquier AWS servicio, primero debe crear un Cuenta de AWS. Puede usarlo Cuenta de AWS para ver sus informes de actividad y uso y para administrar la autenticación y el acceso.

Inscríbase en una Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abrir https://portal.aws.amazon.com/billing/registro.
2. Siga las instrucciones que se le indiquen.

Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

Cuando te registras en un Cuenta de AWS, se crea un usuario Cuenta de AWS root. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. En cualquier momento, puede ver la actividad de su cuenta actual y administrarla accediendo a https://aws.amazon.com/y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de registrarte en un usuario Cuenta de AWS raíz Cuenta de AWS, protege tu usuario raíz AWS IAM Identity Center, habilita y crea un usuario administrativo para que no lo utilices en las tareas diarias.

Proteja a su usuario Cuenta de AWS root

1. Inicia sesión AWS Management Consolecomo propietario de la cuenta; para ello, selecciona el usuario root e introduce tu dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In .
2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la Guía del AWS IAM Identity Center usuario.

Inicio de sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, use la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte Iniciar sesión en el portal de AWS acceso en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .
2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center .

Pasos a seguir a continuación

Ahora que está preparado para trabajar con Amazon SNS, empiece realizando las siguientes tareas:

1. Creación de un tema de Amazon SNS
2. Creación de una suscripción a un tema de Amazon SNS
3. Publicación de un mensaje de Amazon SNS
4. Eliminación de un tema y una suscripción de Amazon SNS

Creación de un tema de Amazon SNS

Un tema de Amazon SNS es un punto de acceso lógico que actúa como un canal de comunicación. Un tema le permite agrupar varios puntos de enlace (como Amazon SQS AWS Lambda, HTTP/S o una dirección de correo electrónico).

Para difundir los mensajes de un sistema productor de mensajes (por ejemplo, un sitio web de comercio electrónico) que trabaja con otros servicios que requieren sus mensajes (por ejemplo, sistemas de pago y tramitación), puede crear un tema para su sistema productor.

La primera tarea, y la más habitual, en Amazon SNS es la creación de un tema. En esta página se muestra cómo puede utilizar los AWS Management Console AWS SDK para Java, los y los AWS SDK para .NET para crear un tema.

Durante la creación, elige un tipo de tema (estándar o FIFO) y asigna un nombre al tema. Después de un tema, no podrá modificar el tipo o el nombre del tema. Todas las demás opciones de configuración son opcionales durante la creación del tema y puede editarlas más adelante.

> **Important**
>
> No agregue información de identificación personal (PII) ni ninguna otra información confidencial o sensible en los nombres de los temas. Otros Amazon Web Services, incluidos los CloudWatch registros, pueden acceder a los nombres de los temas. Los nombres de los temas no están diseñados para contener información privada o confidencial.

Para crear un tema mediante el AWS Management Console

La creación de un tema en Amazon SNS sienta las bases para la distribución de mensajes, ya que le permite publicar mensajes que se pueden distribuir de forma ramificada entre varios suscriptores. Este paso es esencial para configurar el tipo de tema, las opciones de cifrado y las políticas de acceso, a fin de garantizar que el tema cumpla los requisitos operativos, de conformidad y de seguridad de la organización.

1. Inicie sesión en la consola de Amazon SNS.
2. Realice una de las siguientes acciones:

   - Si nunca se ha creado ningún tema bajo tu Cuenta de AWS dirección, lee la descripción de Amazon SNS en la página de inicio.

- Si los temas se han creado Cuenta de AWS anteriormente bajo su dirección, en el panel de navegación, elija Temas.

3. En la página Temas, elija Crear tema.
4. En la página Crear tema, en la sección Detalles, haga lo siguiente:

   a. Para Tipo, elija un tipo de tema (estándar o FIFO).

   b. Ingrese un nombre para el nuevo tema. En el caso de un tema FIFO, agregue .fifo al final del nombre.

   c. (Opcional) Ingrese un nombre para mostrar para el tema.

   > **Important**
   >
   > Cuando se suscriba a un punto de conexión de correo electrónico, el recuento combinado de caracteres del nombre mostrado del tema de Amazon SNS y de la dirección de correo electrónico de envío (por ejemplo, no-reply@sns.amazonaws.com) no debe superar los 320 caracteres UTF-8. Puede utilizar una herramienta de codificación de terceros para verificar la longitud de la dirección de envío antes de configurar un nombre para mostrar para su tema de Amazon SNS.

   d. (Opcional) En el caso de un tema FIFO, puede elegir Desduplicación de mensajes basada en el contenido para habilitar la desduplicación de mensajes predeterminada. Para obtener más información, consulte Desduplicación de mensajes de Amazon SNS para temas FIFO.

5. (Opcional) Expanda la sección Encryption (Cifrado) y haga lo siguiente. Para obtener más información, consulte Protección de los datos de Amazon SNS con cifrado del servidor.

   a. Elija Habilitar el cifrado.

   b. Especifique la AWS KMS clave. Para obtener más información, consulte Términos clave.

   Se muestran los valores de Description (Descripción), Account (Cuenta) y KMS ARN (ARN de KMS) de cada tipo de KMS.

> **Important**
>
> Si no es el propietario de la KMS o si ha iniciado sesión con una cuenta que no tiene los permisos `kms:ListAliases` y `kms:DescribeKey`, no podrá ver la información sobre la KMS en la consola de Amazon SNS.
> Pida al propietario de la KMS que le conceda estos permisos. Para obtener más información, consulte Permisos API de AWS KMS : referencia de recursos y acciones en la Guía para desarrolladores de AWS Key Management Service .

- El KMS AWS gestionado para Amazon SNS (predeterminado) alias/aws/sns está seleccionado de forma predeterminada.

  > **Note**
  >
  > Tenga en cuenta lo siguiente:
  >
  > - La primera vez que utilice AWS Management Console para especificar el KMS AWS administrado para Amazon SNS para un tema, AWS KMS crea el KMS AWS administrado para Amazon SNS.
  > - Como alternativa, la primera vez que utilice la `Publish` acción en un tema con SSE habilitado, AWS KMS creará el KMS AWS administrado para Amazon SNS.

- Para usar un KMS personalizado de su AWS cuenta, elija el campo clave de KMS y, a continuación, elija el KMS personalizado de la lista.

  > **Note**
  >
  > Para obtener instrucciones sobre cómo crear claves personalizadas KMSs, consulte Creación de claves en la Guía para AWS Key Management Service desarrolladores

- Para usar un ARN de KMS personalizado de su AWS cuenta o de otra AWS cuenta, introdúzcalo en el campo de clave de KMS.

6. (Opcional) De forma predeterminada, solo el propietario del tema puede publicar en el tema o suscribirse a este. Para configurar permisos de acceso adicionales, expanda la sección

Access policy (Política de acceso). Para obtener más información, consulte Identity and Access Management en Amazon SNS y Ejemplos de casos de control de acceso con Amazon SNS.

> **Note**
>
> Cuando se crea un tema a través de la consola, la política predeterminada utiliza la clave de condición `aws:SourceOwner`. Esta clave es similar a `aws:SourceAccount`.

7. (Opcional) Para configurar la forma en que Amazon SNS reintenta los intentos de entrega de mensajes con error, expanda la sección Política de reintentos de entrega (HTTP/S). Para obtener más información, consulte Reintento de entrega de mensajes de Amazon SNS.
8. (Opcional) Para configurar la forma en que Amazon SNS registra la entrega de mensajes CloudWatch, amplíe la sección Registro del estado de entrega. Para obtener más información, consulte Estado de entrega de mensajes de Amazon SNS.
9. (Opcional) Para añadir etiquetas de metadatos al tema, expanda la sección Tags (Etiquetas), escriba un valor en Key (Clave) y en Value (Valor) (opcional) y elija Add tag (Añadir etiqueta). Para obtener más información, consulte Etiquetado de temas de Amazon SNS.
10. Seleccione Crear tema.

    Se crea el tema y se muestra la ***MyTopic***página.

    El nombre del tema, el ARN, el nombre para mostrar (opcional) y el ID de AWS cuenta del propietario del tema se muestran en la sección Detalles.
11. Copie el ARN del tema en el portapapeles, por ejemplo:

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

Para crear un tema mediante un SDK AWS

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

Los siguientes ejemplos de código muestran cómo utilizar `CreateTopic`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree un tema con un nombre 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;
        }
    }
```

Cree un tema nuevo con un nombre y atributos específicos de FIFO y desduplicación.

```
    /// <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 obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Creación de un tema de SNS

En el siguiente ejemplo de `create-topic` se crea un tema de SNS denominado `my-topic`.

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

Salida:

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

Para obtener más información, consulte Uso de la interfaz de línea de AWS comandos con Amazon SQS y Amazon SNS en la Guía del usuario de AWS la interfaz de línea de comandos.

- Para obtener más información sobre la API, consulte la Referencia CreateTopicde AWS CLI comandos.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

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

```
        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

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

- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulta CreateTopicla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- Para obtener más información sobre la API, consulta CreateTopicla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Para obtener más información sobre la API, consulta CreateTopicla referencia sobre la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte CreateTopicla referencia sobre la API ABAP del AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta CreateTopicla referencia sobre la API de AWS SDK for Swift.

Creación de una suscripción a un tema de Amazon SNS

Para recibir los mensajes publicados en un tema, tiene que suscribirse a un punto de enlace en el tema. Cuando suscriba un punto de enlace a un tema, el punto de enlace comenzará a recibir todos los mensajes publicados en el tema asociado.

> **Note**
>
> Los puntos de enlace HTTP (S), las direcciones de correo electrónico y los AWS recursos, entre otros, Cuentas de AWS requieren la confirmación de la suscripción antes de poder recibir mensajes.

Para suscribir un punto de enlace a un tema de Amazon SNS, siga estos pasos:

La suscripción de un punto de conexión a un tema de Amazon SNS permite la entrega de mensajes al punto de conexión especificado, lo que garantiza que los sistemas o usuarios correctos reciban notificaciones cuando se publique un mensaje en el tema. Este paso es esencial para vincular el tema con los consumidores, ya sean aplicaciones, destinatarios de correo electrónico u otros servicios, lo que permite una comunicación fluida entre los sistemas.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación izquierdo, elija Suscripciones.
3. En la página Subscriptions (Suscripciones), elija Create subscription (Crear suscripción).
4. En la página Crear suscripción, en la sección Detalles, haga lo siguiente:

a. En ARN de tema, elija el nombre de recurso de Amazon (ARN) de un tema. Este valor es el AWS ARN que se generó al crear el tema Amazon SNS, por ejemplo. `arn:aws:sns:us-east-2:123456789012:your_topic`

b. En Protocolo, elija un tipo de punto de enlace. Los tipos de puntos de enlace disponibles son:

- HTTP/HTTPS
- Correo electrónico/Correo electrónico JSON
- Amazon Data Firehose
- Amazon SQS

> **Note**
>
> Para suscribirse a un tema de SNS FIFO, elija esta opción.

- AWS Lambda
- Punto de conexión de aplicación de plataforma
- SMS

c. En Punto de enlace, ingrese el valor del punto de enlace, como una dirección de correo electrónico o el ARN de una cola de Amazon SQS.

d. Solo para los puntos de conexión de Firehose: en ARN del rol de suscripción, especifique el ARN del rol de IAM que creó para escribir en flujos de entrega de Firehose. Para obtener más información, consulte Requisitos previos para suscribir flujos de entrega de Firehose a temas de Amazon SNS.

e. (Opcional) Para los puntos de conexión de Firehose, Amazon SQS y HTTP/S, también puede habilitar la entrega de mensajes sin procesar. Para obtener más información, consulte Entrega de mensajes sin procesar de Amazon SNS.

f. (Opcional) Para configurar una política de filtro, expanda la sección Política de filtro de suscripción. Para obtener más información, consulte Políticas de filtro de suscripciones de Amazon SNS.

g. (Opcional) Para habilitar el filtrado basado en cargas, configure `Filter Policy Scope` en `MessageBody`. Para obtener más información, consulte Alcance de políticas de filtrado de suscripciones de Amazon SNS.

h. (Opcional) Para configurar una cola de mensajes fallidos en la suscripción, expanda la sección Política de reconducción (cola de mensajes fallidos). Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS.

i. Elija Crear una suscripción.

   En la consola se crea la suscripción y se abre la página Detalles de la suscripción.

Publicación de un mensaje de Amazon SNS

Después de crear un tema de Amazon SNS y suscribir un punto de enlace a él, puede publicar mensajes en un tema. Cuando se publica un mensaje, Amazon SNS intenta entregar el mensaje al punto de enlace suscrito.

Para publicar mensajes en temas de Amazon SNS mediante la AWS Management Console, siga estos pasos:

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación izquierdo, elija Topics (Temas).
3. En la página Temas, seleccione un tema y, a continuación, elija Publicar en tema.

   En la consola, se abre la página Publicar mensaje en un tema.
4. En la sección Detalles básicos, haga lo siguiente:

   a. (Opcional) Ingrese un asunto para el mensaje.

   b. En el caso de un tema FIFO, ingrese un ID de grupo de mensajes. Los mensajes del mismo grupo de mensajes se entregan en el orden en que se publican.

   c. En el caso de un tema FIFO, escriba un ID de desduplicación de mensajes. Este ID es opcional si ha habilitado la configuración Desduplicación de mensajes en función del contenido para el tema.

   d. (Opcional) En el caso de las notificaciones push en móviles, ingrese un valor de período de vida (TTL) en segundos. Esta es la cantidad de tiempo que un servicio de notificaciones push, como Apple Push Notification Service (APNs) o Firebase Cloud Messaging (FCM), tiene para entregar el mensaje al punto final.
5. En la sección Message body (Cuerpo del mensaje), realice alguna de las siguientes acciones:

a. Elija Carga idéntica para todos los protocolos de entrega y, a continuación, ingrese el mensaje.

b. Elija Carga personalizada para cada protocolo de entrega y, a continuación, ingrese un objeto JSON para definir el mensaje que se envía a cada protocolo.

Para obtener más información, consulte Publicación de notificaciones de Amazon SNS con cargas útiles específicas de la plataforma.

6. En la sección Atributos del mensaje, agregue cualquier atributo que quiera que Amazon SNS haga coincidir con el atributo `FilterPolicy` de la suscripción y, de esta manera, poder decidir si el punto de enlace suscrito tiene interés en el mensaje publicado.

a. En Tipo, elija un tipo de atributo, como Matriz.Cadena.

> **Note**
>
> Para el tipo de atributo Matriz.Cadena, incluya la matriz entre corchetes (`[]`). Dentro de la matriz, delimite los valores de cadena con comillas dobles. No es necesario usar comillas con los números ni con las palabras clave `true`, `false` y `null`.

b. Ingrese un nombre para el atributo, como, por ejemplo, `customer_interests`.

c. Ingrese un valor para el atributo, como, por ejemplo, `["soccer", "rugby", "hockey"]`.

Si el tipo de atributo es String (Cadena), String.Array (Matriz.Cadena) o Number (Número), Amazon SNS evalúa el atributo del mensaje con la filter policy (política de filtrado) de una suscripción (si existe), antes de enviar el mensaje a la suscripción cuyo ámbito de políticas de filtrado proporcionado no esté establecido explícitamente en `MessageBody`.

Para obtener más información, consulte Atributos de mensajes de Amazon SNS.

7. Elija Publish message (Publicar mensaje).

El mensaje se publica en el tema. Además, en la consola, se abre la página Detalles.

Para publicar un mensaje en un tema mediante la AWS , siga estos pasos:

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

Los siguientes ejemplos de código muestran cómo utilizar `Publish`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Publique un mensaje en un tema

```
    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 un mensaje en un tema con opciones de grupo, duplicación y 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 las selecciones del usuario a la acción de publicación.

```
    /// <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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

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

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

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

    return outcome.IsSuccess();
}
```

Publique un mensaje con un 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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para C++ .

CLI

AWS CLI

Ejemplo 1: Publicar un mensaje en un tema

En el siguiente ejemplo de `publish` se publica el mensaje especificado en el tema de SNS especificado. El mensaje proviene de un archivo de texto que le permite incluir saltos de línea.

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

Contenido de `message.txt`:

```
Hello World
Second Line
```

Salida:

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

Ejemplo 2: publicar un mensaje SMS en un número de teléfono

En el siguiente ejemplo de `publish`, se publica el mensaje `Hello world!` en el número de teléfono `+1-555-555-0100`.

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

Salida:

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia del comando de la AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();
```

```
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 un mensaje en un tema con opciones de grupo, duplicación y 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

```
        println("${result.messageId} message sent.")
    }
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para PHP .

PowerShell

Herramientas para la PowerShell V4

Ejemplo 1: En este ejemplo se muestra la publicación de un mensaje con una sola línea MessageAttribute declarada.

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

Ejemplo 2: En este ejemplo se muestra la publicación de un mensaje con varios MessageAttributes declarados de antemano.

```
$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 obtener información sobre la API, consulte Publish in Herramientas de AWS para PowerShell Cmdlet Reference (V4).

Herramientas para la versión 5 PowerShell

Ejemplo 1: En este ejemplo se muestra la publicación de un mensaje con una sola MessageAttribute línea declarada.

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

Ejemplo 2: En este ejemplo se muestra la publicación de un mensaje con varios MessageAttributes declarados de antemano.

```
$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 obtener información sobre la API, consulte Publish in Herramientas de AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información sobre. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Publique un mensaje con atributos para que una suscripción pueda filtrar en función de los 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 un mensaje que toma diferentes formas en función del protocolo del suscriptor.

```
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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener información sobre la API, consulte Publicar en la Referencia de la API de AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;
```

```
    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta Publicar en el AWS SDK para obtener información sobre la API de Swift.

Publicación de mensajes grandes con Amazon SNS y Amazon S3

Para publicar mensajes grandes de Amazon SNS, puede utilizar la biblioteca de clientes ampliada de Amazon SNS para Java o la biblioteca de clientes ampliada de Amazon SNS para Python. Estas bibliotecas son útiles para los mensajes que superan el máximo actual de 256 KB, con un máximo de 2 GB. Ambas bibliotecas guardan la carga útil real en un bucket de Amazon S3 y publican la referencia del objeto de Amazon S3 almacenado en el tema de Amazon SNS. Las colas de Amazon

SQS suscritas pueden utilizar la biblioteca de clientes extendidos de Amazon SQS para Java con el fin de eliminar la referencia y recuperar cargas de Amazon S3. Otros puntos de conexión, como Lambda, pueden utilizar la descarga de carga de la biblioteca común de Java para AWS con el fin de eliminar la referencia y recuperar la carga.

> **Note**
>
> Las bibliotecas de clientes ampliadas de Amazon SNS son compatibles con temas estándar y FIFO.

Biblioteca de clientes ampliada de Amazon SNS para Java

Requisitos previos

A continuación, se enumeran los requisitos previos para utilizar la biblioteca de clientes ampliada de Amazon SNS para Java:

- Un AWS SDK. El ejemplo de esta página usa el SDK de AWS Java. Para instalar y configurar el SDK, consulte Configurar el AWS SDK para Java en la Guía para AWS SDK para Java desarrolladores.
- Y Cuenta de AWS con las credenciales adecuadas. Para crear una Cuenta de AWS, vaya a la página de AWS inicio y, a continuación, seleccione Crear una AWS cuenta. Siga las instrucciones.

  Para obtener información sobre las credenciales, consulte Configurar AWS las credenciales y la región para el desarrollo en la Guía para AWS SDK para Java desarrolladores.
- Java 8 o superior.
- La biblioteca de clientes ampliada de Amazon SNS para Java (también disponible en Maven).

Configuración del almacenamiento de mensajes

La biblioteca Amazon SNS Extended Client utiliza la biblioteca común de Java Payload Offloading AWS para almacenar y recuperar mensajes. Puede configurar las siguientes opciones de almacenamiento de mensajes de Amazon S3:

- Umbral de tamaño de los mensajes personalizados: los mensajes con cargas útiles y atributos que superen este tamaño se almacenan automáticamente en Amazon S3.

- **`alwaysThroughS3`**flag: defina este valor `true` para forzar que todas las cargas útiles de mensajes se almacenen en Amazon S3. Por ejemplo:

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

- Clave KMS personalizada: la clave que se debe usar para el cifrado del lado del servidor en su bucket de Amazon S3.
- Nombre del depósito: el nombre del depósito de Amazon S3 para almacenar las cargas útiles de los mensajes.

Ejemplo: Publicación de mensajes en Amazon SNS con carga almacenada en Amazon S3

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema y una cola de ejemplo.
- Suscriba la cola para recibir mensajes del tema.
- Publique un mensaje de prueba.

La carga del mensaje se almacena en Amazon S3 y se publica la referencia a ella. El cliente extendido de Amazon SQS se utiliza para recibir el mensaje.

SDK para Java 1.x

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Para publicar mensajes grandes, utilice la biblioteca de clientes extendidos de Amazon SNS para Java. El mensaje que envía hace referencia a un objeto de Amazon S3 en el que se incluye el contenido real del mensaje.

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

Otros protocolos de puntos de enlace

Tanto las bibliotecas de Amazon SNS como Amazon SQS utilizan la descarga de carga de la biblioteca común de Java para AWS con el fin de almacenar y recuperar cargas de mensajes con Amazon S3. Cualquier punto de enlace habilitado para Java (por ejemplo, un punto de enlace HTTPS implementado en Java) puede usar la misma biblioteca para eliminar la referencia al contenido del mensaje.

Los puntos de enlace que no puedan utilizar la biblioteca común de Java para la descarga de cargas útiles AWS pueden seguir publicando mensajes con cargas útiles almacenadas en Amazon S3. A continuación, mostramos un ejemplo de una referencia de Amazon S3 que se publica mediante el ejemplo de código anterior:

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

Biblioteca de clientes ampliada de Amazon SNS para Python

Requisitos previos

A continuación, se indican los requisitos previos para utilizar la biblioteca de clientes ampliada de Amazon SNS para Python:

- Un AWS SDK. El ejemplo de esta página usa Boto3 del SDK de AWS Python. Para instalar y configurar el SDK, consulte la documentación del SDK de AWS para Python.
- Y Cuenta de AWS con las credenciales adecuadas. Para crear una Cuenta de AWS, vaya a la página de AWS inicio y, a continuación, seleccione Crear una AWS cuenta. Siga las instrucciones.

  Para obtener información sobre las credenciales, consulte Credenciales en la Guía para desarrolladores del SDK de AWS para Python.
- Python 3.x (o posterior) y pip.
- La biblioteca de clientes ampliada de Amazon SNS para Python (también disponible en PyPI).

Configuración del almacenamiento de mensajes

Los siguientes atributos están disponibles en el cliente, tema PlatformEndpointy objetos de Amazon SNS de Boto3 para configurar las opciones de almacenamiento de mensajes de Amazon S3.

- **`large_payload_support`**— El nombre del bucket de Amazon S3 que almacenará los mensajes de gran tamaño.
- **`use_legacy_attribute`**— Si`True`, entonces todos los mensajes publicados utilizan el atributo de mensaje reservado heredado (`SQSLargePayloadSize`) en lugar del atributo de mensaje reservado actual (`ExtendedPayloadSize`).
- **`message_size_threshold`**: el umbral para almacenar el mensaje en el bucket de mensajes de gran tamaño. No puede ser menor `0` ni mayor que`262144`. El valor predeterminado es `262144`.
- **`always_through_s3`**: si `True`, todos los mensajes se almacenan en Amazon S3. El valor predeterminado es `False`.
- **`s3_client`**— El `client` objeto Amazon S3 de Boto3 que se utilizará para almacenar objetos en Amazon S3. Úselo si quiere controlar el cliente Amazon S3 (por ejemplo, credenciales o configuraciones personalizadas de Amazon S3). El valor predeterminado es cuando `boto3.client("s3")` se usa por primera vez si no se ha establecido previamente.

Ejemplo: publicación de mensajes en Amazon SNS con carga almacenada en Amazon S3

En el siguiente ejemplo de código, se muestra cómo:

- Cree un tema de Amazon SNS y una cola de Amazon SQS de ejemplo.
- Adjunta la política a la cola de Amazon SQS para recibir el mensaje del tema Amazon SNS.

- Suscriba la cola para recibir mensajes del tema.
- Publique un mensaje de prueba mediante el cliente extendido, el recurso temático y PlatformEndpoint el recurso de Amazon SNS.
- La carga del mensaje se almacena en Amazon S3 y se publica la referencia a ella.
- Imprima el mensaje publicado de la cola junto con el mensaje original recuperado de Amazon S3.

Para publicar mensajes grandes, utilice la biblioteca de clientes ampliada de Amazon SNS para Python. El mensaje que envía hace referencia a un objeto de Amazon S3 en el que se incluye el contenido real del mensaje.

```
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)
```

Salida

```
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 mensajes de Amazon SNS

Amazon SNS admite los atributos de entrega de mensajes, con los que se pueden ofrecer elementos de metadatos estructurados (como marcas temporales, datos geoespaciales, firmas

e identificadores) relacionados con el mensaje. En el caso de las suscripciones SQS, se puede enviar un máximo de 10 atributos de mensaje cuando se activa Raw Message Delivery (Entrega de mensajes sin procesar). Para enviar más de 10 atributos de mensaje, debe estar desactivada la entrega de mensajes sin procesar. Los mensajes con más de 10 atributos de mensaje dirigidos a las suscripciones de Amazon SQS habilitadas para la entrega de mensajes sin procesar se descartarán como errores del cliente.

Los atributos de los mensajes son opcionales y están separados del cuerpo de los mensajes, pero se envían junto a él. El receptor puede utilizar esta información para decidir cómo gestionar el mensaje sin tener que procesar el cuerpo de este en primer lugar.

Para obtener información sobre el envío de mensajes con atributos mediante el AWS Management Console o el AWS SDK para Java, consulte el Para publicar mensajes en temas de Amazon SNS mediante la AWS Management Console, siga estos pasos: tutorial.

> **Note**
>
> Los atributos de los mensajes se envían únicamente cuando la estructura de los mensajes es String, no JSON.

También puede utilizar los atributos del mensaje como ayuda para estructurar el mensaje de notificación push en los puntos de enlace móviles. En este caso, los atributos del mensaje solo se usan para ayudar a estructurar el mensaje de notificación push. Los atributos no se entregan al punto de enlace, como se entregan cuando se envían mensajes con atributos a puntos de enlace de Amazon SQS.

También puede utilizar atributos de mensajes para que sus mensajes se puedan filtrar mediante políticas de filtro de suscripciones. Puede aplicar políticas de filtro a las suscripciones de temas. Cuando se aplica una política de filtrado con el alcance de la política de filtrado establecido en `MessageAttributes` (predeterminado), una suscripción recibe solo los mensajes que tienen atributos aceptados por la política. Para obtener más información, consulte Filtrado de mensajes en Amazon SNS.

> **Note**
>
> Cuando se utilizan atributos de mensaje para el filtrado, el valor debe ser una cadena JSON válida. De este modo, se garantiza que el mensaje se entrega a una suscripción con el filtrado de atributos de mensajes activado.

Elementos y validación de los atributos de los mensajes

Cada atributo de mensaje consta de los siguientes elementos:

- Nombre: el nombre del atributo de mensaje puede contener los siguientes caracteres: A-Z, a-z, 0-9, subrayado (_), guion (-) y punto (.). El nombre no debe comenzar ni finalizar con un punto y no debe tener dos puntos sucesivos. El nombre distingue entre mayúsculas y minúsculas y debe ser único entre todos los nombres de atributo del mensaje. El nombre puede tener una longitud de hasta 256 caracteres. El nombre no puede comenzar con `AWS.` o `Amazon.` (ni ninguna variación en el uso de mayúsculas y minúsculas), ya que estos prefijos están reservados para el uso de Amazon Web Services.
- Tipo: los tipos de datos admitidos para el atributo de mensaje son `String`, `String.Array`, `Number` y `Binary`. El tipo de datos tiene las mismas restricciones en lo que respecta al contenido que el cuerpo del mensaje. Para obtener más información, consulte la sección Tipos de datos y validación de los atributos de los mensajes.
- Valor: el valor del atributo de mensaje especificado por el usuario. Para los tipos de datos de cadena, el atributo value debe seguir las mismas restricciones de contenido que el cuerpo del mensaje. Sin embargo, si el atributo message se utiliza para filtrar, el valor debe ser una cadena JSON válida para garantizar la compatibilidad con las políticas de filtrado de suscripciones de Amazon SNS. Para obtener más información, consulte la acción Publicar en la Referencia de la API de Amazon Simple Notification Service.

El nombre, el tipo y el valor no deben estar vacíos ni ser null. Además, el cuerpo del mensaje no debe estar vacío ni ser null. Todas las partes del atributo de mensaje, incluido el nombre, el tipo y el valor, están incluidas en la restricción de tamaño del mensaje, que actualmente es de 256 KB.

Tipos de datos y validación de los atributos de los mensajes

Los tipos de datos de los atributos de los mensajes identifican la forma en que Amazon SNS gestiona los valores de los atributos de los mensajes. Por ejemplo, si el tipo es un número, Amazon SNS valida que es un número.

Amazon SNS admite los siguientes tipos de datos lógicos para todos los puntos de enlace, excepto según se indica:

- Cadena: las cadenas son Unicode con codificación binaria UTF-8. Para obtener una lista de los valores de código, consulte http://en.wikipedia. org/wiki/ASCII#ASCII_printable_characters.

  > **Note**
  >
  > No se admiten valores sustitutos en los atributos de los mensajes. Por ejemplo, si se utiliza un valor sustituto para representar un emoji, se producirá el siguiente error: `Invalid attribute value was passed in for message attribute`.

- Cadena.matriz: una matriz, con formato de cadena, que puede contener varios valores. Los valores pueden ser cadenas, números o las palabras clave `true`, `false` y `null`. Un String.Array de tipo numérico o booleano no requiere comillas. Los distintos valores de String.Array están separados por comas.

  Este tipo de datos no se admite en las AWS Lambda suscripciones. Si especifica este tipo de datos para los puntos de enlace de Lambda, se pasa como el tipo de datos `String` en la carga útil JSON que Amazon SNS entrega a Lambda.

- Número: los números son enteros positivos o negativos o números de coma flotante. Tienen una precisión y un rango adecuados para abarcar la mayoría de los posibles valores que los tipos integer, float y double admiten normalmente. Un número puede tener un valor comprendido entre -10^9 y 10^9, con una precisión de 5 dígitos tras el separador decimal. Los ceros iniciales y finales se recortan.

  Este tipo de datos no se admite en las AWS Lambda suscripciones. Si especifica este tipo de datos para los puntos de enlace de Lambda, se pasa como el tipo de datos `String` en la carga útil JSON que Amazon SNS entrega a Lambda.

- Binario: con los atributos de tipo binarios, se puede almacenar datos binarios de cualquier índole, tales como datos comprimidos, datos cifrados o imágenes.

Atributos de mensaje reservados para notificaciones de inserción en móviles

En la siguiente tabla se muestran los atributos de mensaje reservados para los servicios de notificación push en móviles que puede utilizar para estructurar su mensaje de notificación push:

Servicio de notificaciones de inserción	Atributo de mensaje reservado
ADM	`AWS.SNS.MOBILE.ADM.TTL`
APNs[1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`
	`AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS.COLLAPSE_ID`
	`AWS.SNS.MOBILE.APNS.PRIORITY`
	`AWS.SNS.MOBILE.APNS.PUSH_TYPE`
	`AWS.SNS.MOBILE.APNS.TOPIC`
	`AWS.SNS.MOBILE.APNS.TTL`
Baidu	`AWS.SNS.MOBILE.BAIDU.DeployStatus`
	`AWS.SNS.MOBILE.BAIDU.MessageKey`
	`AWS.SNS.MOBILE.BAIDU.MessageType`
	`AWS.SNS.MOBILE.BAIDU.TTL`

Servicio de notificaciones de inserción	Atributo de mensaje reservado
FCM	`AWS.SNS.MOBILE.FCM.TTL`
	`AWS.SNS.MOBILE.GCM.TTL`
macOS	`AWS.SNS.MOBILE.MACOS_SANDBOX.TTL`
	`AWS.SNS.MOBILE.MACOS.TTL`
MPNS	`AWS.SNS.MOBILE.MPNS.NotificationClass`
	`AWS.SNS.MOBILE.MPNS.TTL`
	`AWS.SNS.MOBILE.MPNS.Type`
WNS	`AWS.SNS.MOBILE.WNS.CachePolicy`
	`AWS.SNS.MOBILE.WNS.Group`
	`AWS.SNS.MOBILE.WNS.Match`
	`AWS.SNS.MOBILE.WNS.SuppressPopup`
	`AWS.SNS.MOBILE.WNS.Tag`
	`AWS.SNS.MOBILE.WNS.TTL`
	`AWS.SNS.MOBILE.WNS.Type`

[1] Apple rechazará las notificaciones de Amazon SNS si los atributos de los mensajes no cumplen sus requisitos. Para obtener más información, consulta Cómo enviar solicitudes de notificación al APNs sitio web para desarrolladores de Apple.

Agrupación en lotes de mensajes de Amazon SNS

¿Qué es la agrupación en lotes de mensajes?

Una alternativa a la publicación de mensajes en temas estándar o FIFO mediante solicitudes de API `Publish` individuales consiste en utilizar la API `PublishBatch` de Amazon SNS para publicar hasta 10 mensajes en una única solicitud de API. Enviar los mensajes por lotes puede contribuir a reducir los costes asociados a la conexión de aplicaciones distribuidas (mensajería A2A) o al envío de notificaciones a personas (mensajería A2P) con Amazon SNS hasta 10 veces. Amazon SNS tiene cuotas en cuanto al número de mensajes que se pueden publicar en un tema por segundo en función de la región en la que se opere. Consulte la página Cuotas y puntos de conexión de Amazon SNS de la guía de Referencia general de AWS para obtener más información sobre las cuotas de API.

> **Note**
>
> El tamaño total total de todos los mensajes que envíes en una sola solicitud de `PublishBatch` API no puede superar los 262.144 bytes (256 KiB).
> La API `PublishBatch` utiliza la misma acción de API `Publish` para las políticas de IAM.

¿Cómo funciona la agrupación en lotes de mensajes?

Publicar mensajes con la API `PublishBatch` es similar a hacerlo con la API `Publish`. La principal diferencia es que a cada mensaje de una solicitud de API `PublishBatch` se le debe asignar un ID de lote único (hasta 80 caracteres). De esta forma, Amazon SNS puede devolver respuestas de API individuales para cada mensaje de un lote, con objeto de confirmar que el mensaje se ha publicado, o bien que se ha producido un error. En el caso de los mensajes que se publiquen en temas FIFO, además de asignar un ID de lote único, se debe incluir un `MessageDeduplicationID` y un `MessageGroupId` en cada mensaje individual.

Ejemplos

Publicación de un lote de 10 mensajes en un tema estándar

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

Publicación de un lote de 10 mensajes en un tema 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());
    }
}
```

Eliminación de un tema y una suscripción de Amazon SNS

Cuando se elimina un tema, las suscripciones asociadas se eliminan de forma asíncrona. Aunque los clientes pueden seguir accediendo a estas suscripciones, estas ya no están asociadas al tema, aunque se vuelva a crear el tema con el mismo nombre. Si un publicador intenta publicar un mensaje en el tema eliminado, recibirá un mensaje de error que indica que el tema no existe. Del mismo modo, cualquier intento de suscribirse al tema eliminado también generará un mensaje de error. No puede eliminar una suscripción que esté pendiente de confirmación. Amazon SNS elimina de forma automática las suscripciones no confirmadas después de 48 horas.

Para eliminar un tema o una suscripción de Amazon SNS mediante AWS Management Console

Eliminar un tema o una suscripción de Amazon SNS garantiza una administración eficaz de los recursos, ya que evita el uso de recursos innecesarios y mantiene organizada la consola de Amazon SNS. Este paso ayuda a evitar los posibles costos derivados de los recursos inactivos y optimiza la administración al eliminar los temas o las suscripciones que ya no son necesarios.

Para eliminar un tema mediante el AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación izquierdo, elija Topics (Temas).
3. En la página Temas, seleccione un tema y, a continuación, elija Eliminar.
4. En el cuadro de diálogo Eliminar tema, ingrese `delete me` y, a continuación, elija Eliminar.

   La consola elimina el tema.

Para eliminar una suscripción mediante el AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación izquierdo, elija Suscripciones.
3. En la página Suscripciones, seleccione una suscripción con el estado Confirmado y, a continuación, elija Eliminar.
4. En el cuadro de diálogo Eliminar suscripción, elija Eliminar.

   La consola elimina la suscripción.

Para eliminar una suscripción y un tema mediante la SDK de AWS , siga estos pasos:

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

Los siguientes ejemplos de código muestran cómo utilizar `DeleteTopic`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Elimine un tema por su ARN de tema.

```
/// <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 obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Eliminación de un tema de SNS

El siguiente ejemplo de `delete-topic` elimina el tema de SNS especificado.

```
aws sns delete-topic \
```

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

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia de AWS CLI comandos.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>
```

```
                Where:
                   topicArn - The ARN of the topic to delete.
                """;

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

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

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulta DeleteTopicla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Para obtener más información sobre la API, consulta DeleteTopicla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte DeleteTopicla referencia sobre la API ABAP del AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import AWSSNS

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

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

- Para obtener más información sobre la API, consulta DeleteTopicla referencia sobre la API de AWS SDK for Swift.

Pasos a seguir a continuación

Ahora que ha creado un tema con una suscripción y enviado mensajes a este, es posible que desee probar lo siguiente:

- Explore el Centro de desarrolladores de AWS.
- Obtenga información sobre la protección de los datos en la sección Seguridad.
- Habilite el cifrado en el servidor para un tema.
- Habilite el cifrado en el lado de servidor para un tema con una cola suscrita de Amazon Simple Queue Service (Amazon SQS) cifrada.
- Suscriba Canalizaciones de bifurcación de eventos de AWS a un tema.

Ordenación y desduplicación de mensajes mediante temas FIFO de Amazon SNS

En este tema se proporciona información sobre las características y funcionalidades de los temas FIFO (First-In-First-Out) y cómo se integran con las colas FIFO de Amazon SQS. Aprenderá a utilizar estos servicios juntos para garantizar la ordenación y desduplicación estrictas de los mensajes, algo esencial para las aplicaciones que requieren coherencia de datos. Este contenido describe los casos de uso específicos en los que los temas FIFO de Amazon SNS son beneficiosos y proporciona información sobre situaciones en las que el orden y la exclusividad de los mensajes son fundamentales.

También conocerá los detalles técnicos de la ordenación y agrupación de los mensajes, y cómo afectan a la entrega de los mensajes. En el tema sobre la desduplicación de mensajes se explican los mecanismos que impiden la duplicación de mensajes, lo que garantiza que cada mensaje se procese una sola vez. Además, aprenderá sobre el filtrado, la seguridad y la durabilidad de los mensajes, aspectos importantes para mantener la integridad y la fiabilidad de su sistema de mensajería. Este contenido también incluye información sobre el archivado y la reproducción de mensajes, y ofrece estrategias para administrar el historial de los mensajes. También se proporcionan ejemplos de código prácticos para ayudarlo a implementar estas características en sus propias aplicaciones, con lo que obtendrá experiencia práctica con los temas FIFO de Amazon SNS y su integración con las colas FIFO de Amazon SQS.

Temas de FIFO de alto rendimiento en Amazon SNS

Los temas de FIFO de alto rendimiento en Amazon SNS gestionan de manera eficiente el alto rendimiento de los mensajes y, al mismo tiempo, mantienen un orden estricto de los mensajes, lo que garantiza la fiabilidad y la escalabilidad de las aplicaciones que procesan numerosos mensajes. Esta solución es ideal para situaciones que exigen tanto un alto rendimiento como una entrega de mensajes ordenada. Para mejorar el rendimiento de los mensajes mediante temas de FIFO de alto rendimiento, se recomienda aumentar el número de grupos de mensajes. Para obtener más información sobre las cuotas de mensajes de alto rendimiento, consulte las cuotas de servicio de Amazon SNS en. Referencia general de Amazon Web Services

Casos de uso para temas de FIFO de Amazon SNS de alto rendimiento

Los siguientes casos de uso destacan las diversas aplicaciones de los temas de FIFO de alto rendimiento y muestran su eficacia en todos los sectores y escenarios:

- Procesamiento de datos en tiempo real: las aplicaciones que se ocupan de flujos de datos en tiempo real, como el procesamiento de eventos o la ingesta de datos telemétricos, pueden beneficiarse de los temas de FIFO de alto rendimiento para gestionar la afluencia continua de mensajes y, al mismo tiempo, conservar su orden para un análisis preciso.
- Procesamiento de pedidos de comercio electrónico: en las plataformas de comercio electrónico en las que es fundamental mantener el orden de las transacciones de los clientes, los dispositivos FIFO, de alto rendimiento, garantizan que los pedidos se entreguen de forma secuencial y sin demoras, incluso durante las temporadas altas de compras.
- Servicios financieros: las instituciones financieras que gestionan datos comerciales o transaccionales de alta frecuencia se basan en temas de FIFO de alto rendimiento para procesar los datos de mercado y las transacciones con una latencia mínima y, al mismo tiempo, cumplen con los estrictos requisitos reglamentarios en materia de pedidos de mensajes.
- Transmisión multimedia: las plataformas de transmisión y los servicios de distribución multimedia utilizan temas FIFO de alto rendimiento para gestionar la entrega de archivos multimedia y contenido en streaming, lo que garantiza una experiencia de reproducción fluida para los usuarios y, al mismo tiempo, mantiene el orden correcto de entrega del contenido

Particiones y distribución de datos para un alto rendimiento en temas de FIFO de Amazon SNS

Con temas de alto rendimiento, Amazon SNS distribuye los datos de temas FIFO entre las particiones. Una partición es una asignación de capacidad para un tema que se replica automáticamente en varias zonas de disponibilidad dentro de un. Región de AWS Usted no administra las particiones. En su lugar, Amazon SNS administra automáticamente las particiones en su nombre, en función de la tasa de entrada.

Para los temas de FIFO, Amazon SNS modifica el número de particiones de un tema en las siguientes situaciones:

- Si la tasa de publicación actual se acerca o supera la que pueden admitir las particiones existentes, se asignan particiones adicionales hasta que el tema alcance la cuota regional.

Para obtener información sobre las cuotas, consulte las cuotas de servicio de Amazon SNS en. Referencia general de Amazon Web Services

- Si las particiones actuales se utilizan poco, es posible que se reduzca el número de particiones.

La administración de las particiones tiene lugar automáticamente en segundo plano y es transparente para las aplicaciones. Su tema y sus mensajes están disponibles en todo momento.

> **Note**
>
> Si aumentas de forma repentina y significativa el tráfico a tu tema y envías varias veces el volumen habitual, es posible que la API de publicación se vea limitada temporalmente. Esta limitación puede durar hasta que dure el período de deduplicación, mientras que el tema se amplía para adaptarse al aumento del tráfico.

Distribuir los datos por grupo de mensajes IDs

Al publicar un mensaje en un tema FIFO, Amazon SNS utiliza el valor del ID del grupo de mensajes de cada mensaje como entrada para una función hash interna. El valor de salida de la función hash determina qué partición procesa el mensaje. Una partición determinada IDs puede gestionar uno o más grupos de mensajes.

> **Note**
>
> Amazon SNS está optimizado para una distribución uniforme de los elementos en las particiones de un tema FIFO, independientemente del número de particiones. AWS recomienda utilizar un grupo de mensajes IDs que pueda tener un gran número de valores distintos.

Habilite un alto rendimiento en su tema FIFO de Amazon SNS

De forma predeterminada, los temas FIFO de Amazon SNS están configurados para la deduplicación a nivel de tema, esto se controla mediante el atributo del tema `FifoThroughputScope`establecido en `Topic` y tienen cuotas de rendimiento más restringidas; consulte las cuotas de servicio de Amazon SNS en. Referencia general de Amazon Web Services

Para habilitar un alto rendimiento en su tema FIFO de Amazon SNS, actualice `FifoThroughputScope` el atributo a. `MessageGroup` Este cambio se puede realizar a través de la consola o mediante el SDK AWS CLI y, además, se puede configurar durante la creación del tema, algo que Amazon SNS recomienda para ofrecer la mejor experiencia al cliente y reducir las posibilidades de que el tema se vea limitado.

> **Important**
>
> Una vez que haya activado el contenido de `FifoThroughputScope` un tema`MessageGroup`, no podrá volver al rendimiento. `Topic`

Habilite el modo de alto rendimiento para cualquier cola FIFO de Amazon SQS suscrita

Cuando publique en su tema de FIFO de Amazon SNS con el alto rendimiento activado y esté suscrito a una o más colas de FIFO de Amazon SQS, se recomienda que habilite el alto rendimiento en sus colas de FIFO de Amazon SQS para que su tema de FIFO de Amazon SNS se entregue sin problemas. Para obtener más información, consulte Rendimiento alto de las colas FIFO en la Guía para desarrolladores de Amazon Simple Queue Service.

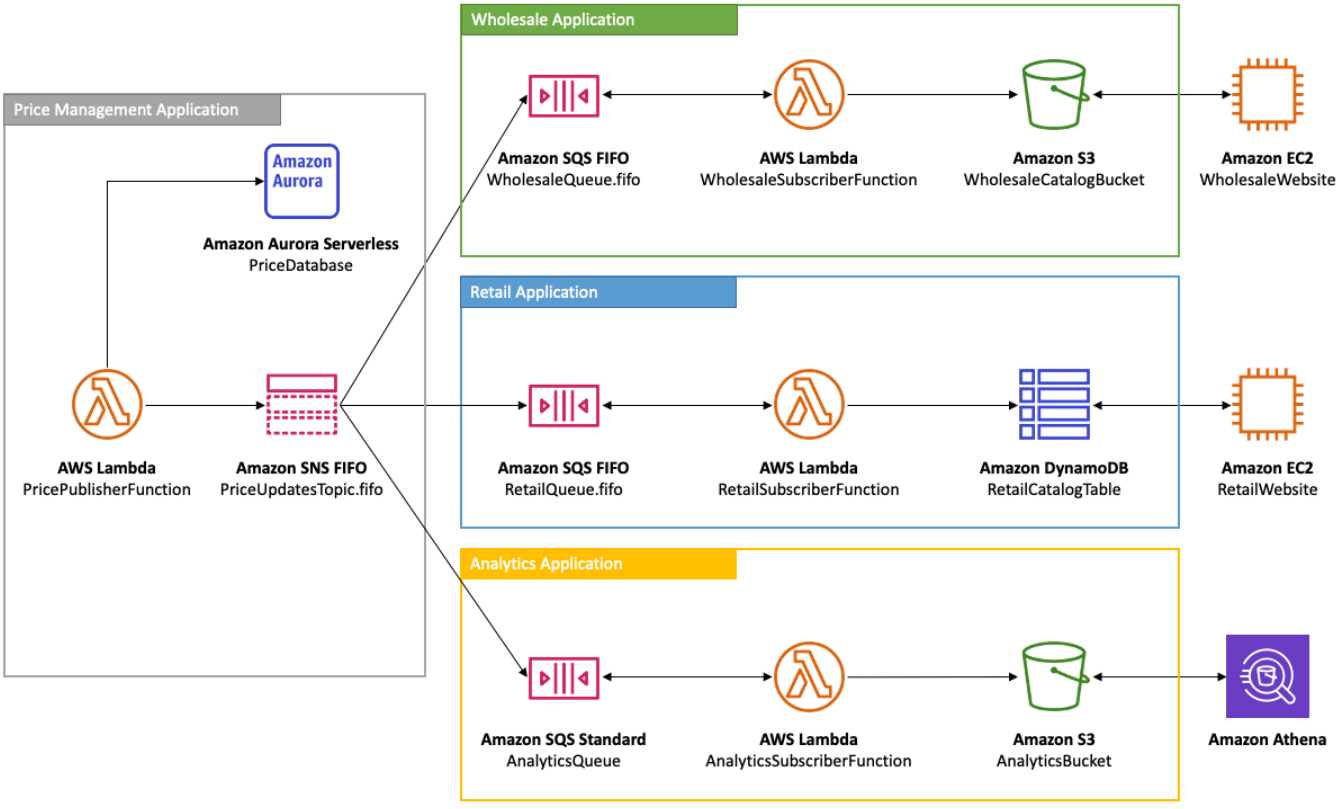
Ejemplo de caso de uso de temas FIFO de Amazon SNS

En el siguiente ejemplo, se describe una plataforma de comercio electrónico creada por un fabricante de partes de automóviles con temas FIFO de Amazon SNS y colas de Amazon SQS. La plataforma se compone de cuatro aplicaciones sin servidor:

- Los administradores de inventarios utilizan una aplicación de administración de precios para establecer el precio de cada elemento en stock. En esta empresa, los precios de los productos pueden cambiar en función de la fluctuación del cambio de divisas, la demanda del mercado y los cambios en la estrategia de ventas. La aplicación de administración de precios utiliza una función AWS Lambda que publica actualizaciones de precios en un tema FIFO de Amazon SNS cada vez que cambian los precios.
- Con una aplicación mayorista, se proporciona el backend para un sitio web en el que los talleres de carrocería de automóviles y fabricantes de automóviles pueden comprar partes de automóviles de la compañía a granel. Para obtener notificaciones de cambio de precio, la aplicación mayorista

suscribe su cola FIFO de Amazon SQS al tema FIFO de Amazon SNS de la aplicación de administración de precios.

- Con una aplicación minorista, se proporciona el backend para otro sitio web en el que los propietarios de automóviles y entusiastas de ajuste de automóviles pueden comprar partes de automóviles individuales para sus vehículos. Para obtener notificaciones de cambio de precio, la aplicación minorista también suscribe su cola FIFO de Amazon SQS al tema FIFO de Amazon SNS de la aplicación de administración de precios.
- Una aplicación de análisis que agrega actualizaciones de precios y las almacena en un bucket de Amazon S3, lo que permite a Amazon Athena consultar el bucket con fines de inteligencia empresarial (BI). Para obtener notificaciones de cambio de precio, la aplicación de análisis también suscribe su cola estándar de SQS al tema FIFO de SNS de la aplicación de administración de precios. A diferencia de las demás aplicaciones, la de análisis no requiere que las actualizaciones de precios estén ordenadas de forma estricta.

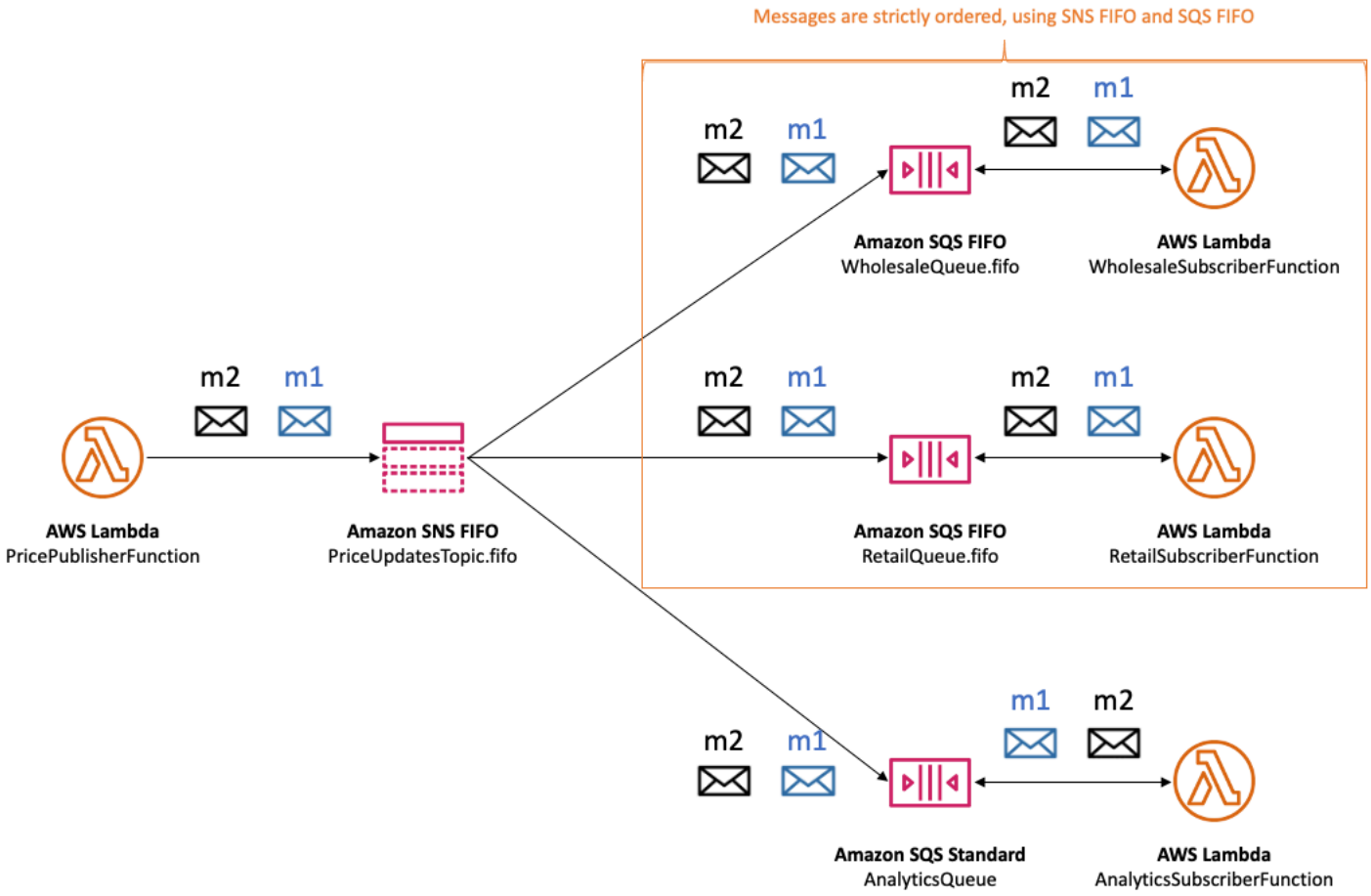


Para que las aplicaciones mayoristas y minoristas reciban actualizaciones de precios en el orden correcto, en la aplicación de administración de precios se debe utilizar un sistema de distribución de mensajes estrictamente ordenado. Si utiliza los temas FIFO de Amazon SNS y las colas FIFO de Amazon SQS, podrá procesar los mensajes en orden y sin duplicados. Para obtener más

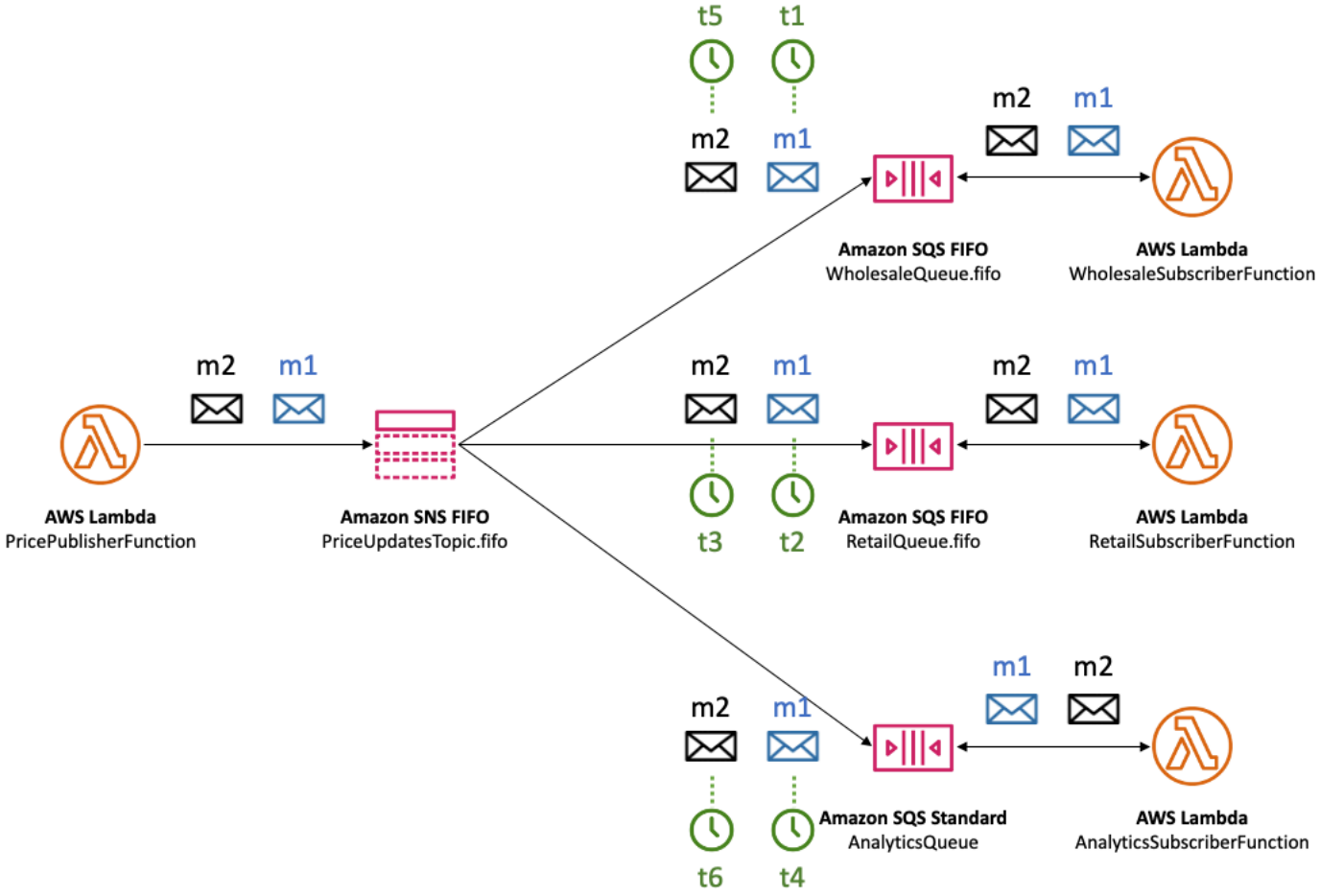
información, consulte Detalles de ordenación de mensajes de Amazon SNS para temas FIFO. Para ver los fragmentos de código que implementan este caso de uso, consulte Ejemplos de código de Amazon SNS para temas FIFO.

Detalles de ordenación de mensajes de Amazon SNS para temas FIFO

Un tema FIFO de Amazon SNS siempre entrega mensajes a las colas de Amazon SQS suscritas en el orden exacto en que los mensajes se publican en el tema, y solo una vez. Con una cola FIFO de Amazon SQS suscrita, el consumidor de la cola recibe los mensajes en el orden exacto en que los mensajes se entregan a la cola, y sin duplicados. Sin embargo, con una cola estándar de SQS suscrita, el consumidor de la cola puede recibir mensajes desordenados y varias veces. Esto permite desvincular aún más a los suscriptores de los publicadores, lo que proporciona a los suscriptores más flexibilidad en cuanto al consumo de mensajes y la optimización de costos, como se muestra en el siguiente diagrama, basado en Ejemplo de caso de uso de temas FIFO de Amazon SNS.

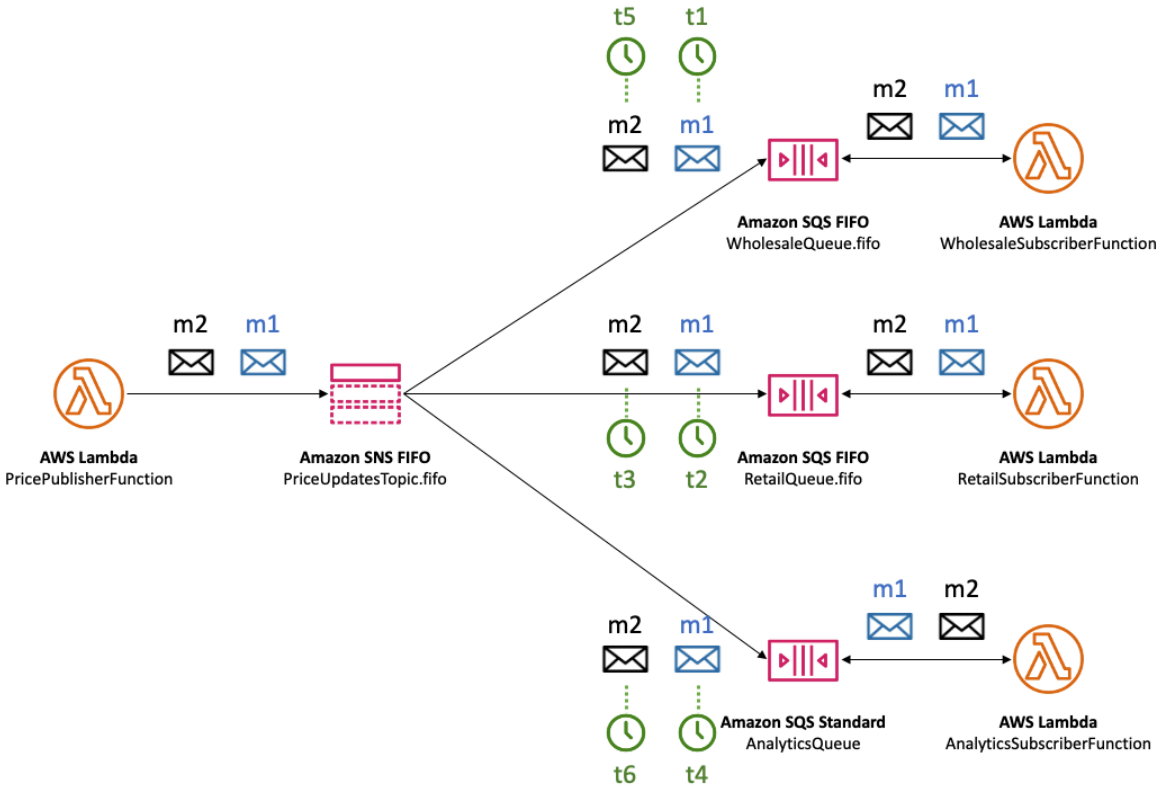


Tenga en cuenta que no hay pedidos implícitos de los suscriptores. En el ejemplo siguiente, se muestra que el mensaje m1 se entrega primero al suscriptor mayorista, después al suscriptor minorista y, a continuación, al suscriptor de análisis. El mensaje m2 se entrega primero al suscriptor minorista, después al suscriptor mayorista y, a continuación, al suscriptor de análisis. Si bien los dos mensajes se entregan a los suscriptores en un orden diferente, el orden de mensajes se conserva para cada suscriptor. Cada suscriptor se percibe de forma aislada de cualquier otro suscriptor.



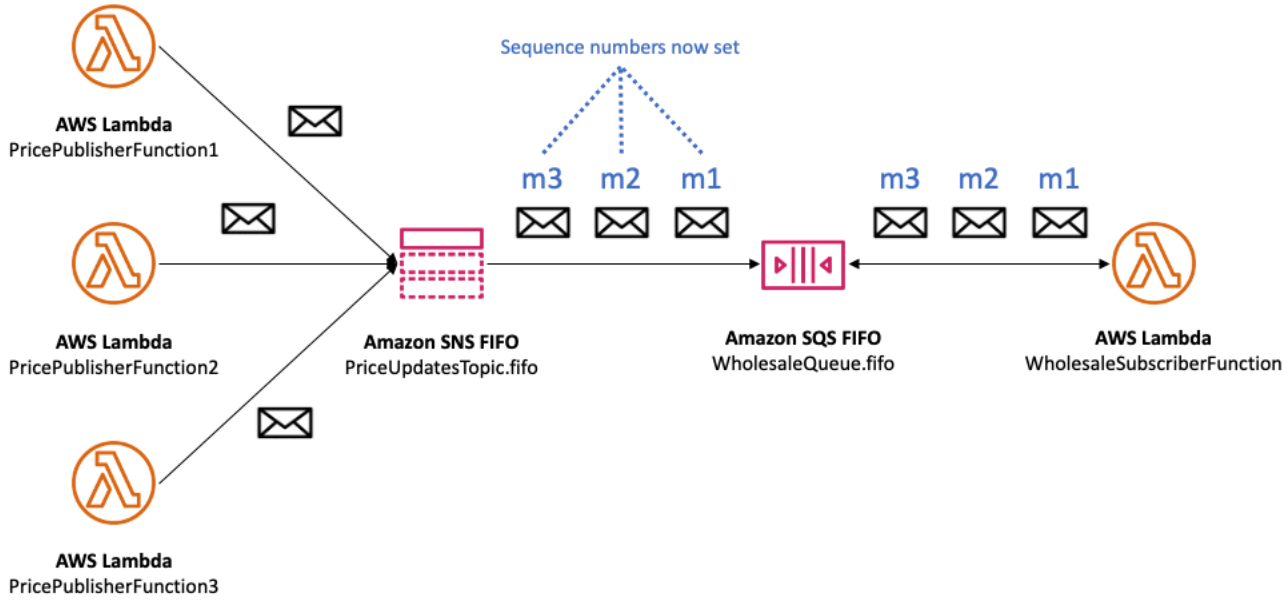
Si un suscriptor de cola FIFO de Amazon SQS se vuelve inaccesible, puede dejar de estar sincronizado. Por ejemplo, supongamos que el propietario de la cola de aplicaciones mayorista cambia por error la política de colas de Amazon SQS de forma que se impida que la entidad principal de servicio de Amazon SNS entregue mensajes a la cola. En este caso, se producen errores en los envíos de actualizaciones de precios a la cola de mayoristas, mientras que los de las colas de minoristas y analistas se realizan con éxito, lo que provoca que los suscriptores no estén sincronizados. Cuando el propietario de la cola de aplicaciones mayoristas corrige su política de colas, Amazon SNS reanuda la entrega de mensajes a la cola suscrita. Se descartan todos los

mensajes publicados en el tema que tengan como destino la cola configurada incorrectamente, a menos que la suscripción correspondiente tenga configurada una cola de mensajes fallidos.



Puede tener varias aplicaciones (o varios subprocesos dentro de la misma aplicación) que publiquen mensajes en un tema FIFO SNS en paralelo. Al hacerlo, delega con eficacia la secuenciación de mensajes en el servicio Amazon SNS. Para determinar la secuencia establecida de mensajes, puede verificar el número de secuencia.

El número de secuencia es un número grande no consecutivo que Amazon SNS asigna a cada mensaje. La longitud del número de secuencia es de 128 bits y sigue aumentando para cada grupo de mensajes. El número de secuencia se pasa a las colas FIFO de Amazon SQS suscritas como parte del cuerpo del mensaje. Sin embargo, si habilita la entrega de mensajes sin procesar, el mensaje que se entrega a la cola FIFO de Amazon SQS no incluye el número de secuencia ni ningún otro metadato de mensajes de Amazon SNS.



Los temas FIFO de Amazon SNS definen el pedido en el contexto de un grupo de mensajes. Para obtener más información, consulte Agrupación de mensajes de Amazon SNS para temas FIFO.

Agrupación de mensajes de Amazon SNS para temas FIFO

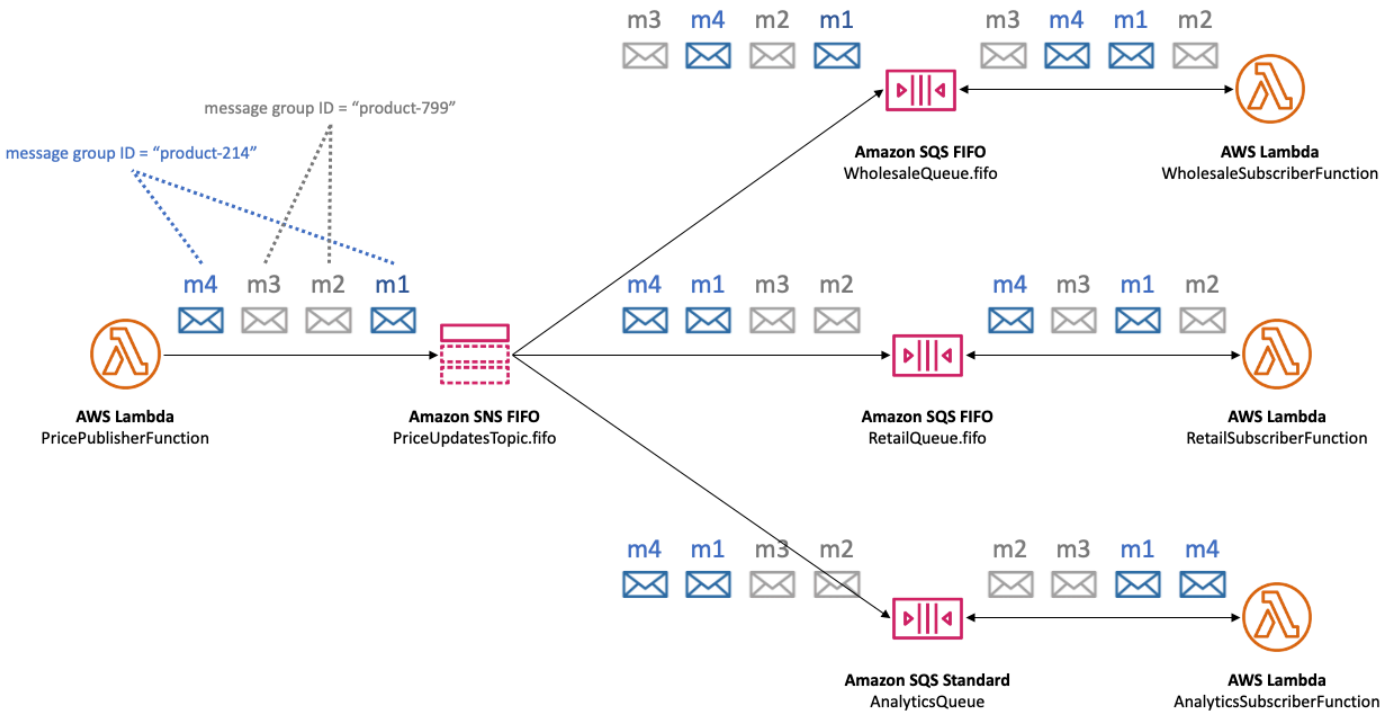
Los mensajes que pertenecen al mismo grupo se procesan uno por uno, en un orden estricto relativo al grupo.

Al publicar mensajes en un tema de FIFO de Amazon SNS, debe establecerse el ID del grupo de mensajes. El ID de grupo es un token obligatorio con el que se especifica que un mensaje pertenece a un grupo de mensajes específico. El tema de SNS FIFO pasa el ID de grupo a las colas FIFO de Amazon SQS suscritas. No hay límite en cuanto al número de grupos IDs en los temas de FIFO de SNS o en las colas de FIFO de SQS. El ID del grupo de mensajes no se transfiere a las colas estándar de Amazon SQS.

No hay afinidad entre un grupo de mensajes y una suscripción. Por lo tanto, los mensajes que se publican en cualquier grupo de mensajes se entregan a todas las colas suscritas, sujeto a las políticas de filtro adjuntas a las suscripciones. Para obtener más información, consulte Entrega de mensajes de Amazon SNS para temas FIFO y Filtrado de mensajes de Amazon SNS para temas FIFO.

En el caso de uso de ejemplo de administración de precios de partes de automóviles, hay un grupo de mensajes dedicado para cada producto vendido en la plataforma. El mismo tema FIFO de Amazon SNS se utiliza para procesar todas las actualizaciones de precios. La secuencia de

actualizaciones de precios se conserva en el contexto de un solo producto de partes de automóviles, pero no en varios productos. En el siguiente diagrama, se muestra cómo funciona. Tenga en cuenta que, en el caso del producto cuyo ID de grupo de mensajes es product-214, el mensaje m1 se procesa antes que el m4. Esta secuencia se conserva a lo largo de los flujos de trabajo, desde FIFO de Amazon SNS a FIFO de Amazon SQS. Del mismo modo, para el producto cuyo ID de grupo de mensajes es product-799, el mensaje m2 se procesa antes que m3. Sin embargo, cuando se utilizan colas estándar de Amazon SQS, el orden de los mensajes ya no está garantizado y los grupos de mensajes no existen. Los grupos de mensajes product-214 y product-799 son independientes entre sí, por lo que no hay relación entre cómo se secuencian sus mensajes.



Distribuir los datos por grupo IDs de mensajes para mejorar el rendimiento

Para optimizar el rendimiento de la entrega, los temas FIFO de Amazon SNS entregan mensajes de diferentes grupos de mensajes en paralelo, mientras que el orden de los mensajes se mantiene estrictamente dentro de cada grupo de mensajes. Cada grupo de mensajes individual puede entregar un máximo de 300 mensajes por segundo. Por lo tanto, para lograr un alto rendimiento para un solo tema, utilice una gran cantidad de grupos IDs de mensajes distintos. Al utilizar un conjunto diverso de grupos de mensajes, los temas FIFO de Amazon SNS distribuyen automáticamente los mensajes entre un mayor número de particiones paralelas.

> **Note**
>
> Los temas FIFO de Amazon SNS están optimizados para una distribución uniforme de los mensajes entre los grupos de mensajes IDs, independientemente del número de grupos. AWS recomienda utilizar un gran número de grupos de mensajes distintos IDs para optimizar el rendimiento.

Al publicar en el tema FIFO de Amazon SNS con un rendimiento alto y una o más colas FIFO de Amazon SQS estén suscritas, se recomienda que habilite el rendimiento alto en las colas. Para obtener más información, consulte Rendimiento alto de las colas FIFO en la Guía para desarrolladores de Amazon Simple Queue Service.

Entrega de mensajes de Amazon SNS para temas FIFO

Los temas FIFO (primero en entrar, primero en salir) de Amazon SNS admiten la entrega a colas tanto estándar como FIFO de Amazon SQS para ofrecer a los clientes flexibilidad y control a la hora de integrar aplicaciones distribuidas que requieren coherencia de datos casi en tiempo real.

Para las cargas de trabajo que necesitan preservar un orden estricto de los mensajes o la desduplicación, la combinación de temas FIFO de Amazon SNS con colas FIFO de Amazon SQS suscritas como punto de conexión de entrega proporciona una mejora de la mensajería entre aplicaciones cuando el orden de las operaciones y los eventos es crítico, o cuando no se pueden tolerar los duplicados.

Para las cargas de trabajo que toleran los pedidos y las at-least-once entregas con el máximo esfuerzo, suscribir las colas estándar de Amazon SQS a los temas de FIFO de Amazon SNS permite reducir los costos, además de compartir las colas entre las cargas de trabajo que no utilizan FIFO.

> **Note**
>
> Para distribuir los mensajes de los temas de FIFO de Amazon SNS a AWS Lambda las funciones, se requieren pasos adicionales. En primer lugar, suscriba las colas FIFO o estándar de Amazon SQS al tema. A continuación, configure las colas para desencadenar las funciones. Para obtener más información, consulte SQS FIFO como fuente de eventos en el blog de informática de AWS .

Los temas de SNS FIFO no pueden entregar mensajes a los puntos de enlace administrados por el cliente, como direcciones de correo electrónico, aplicaciones móviles, números de teléfono para mensajería de texto (SMS) o puntos de enlace HTTP (S). No se garantiza que estos tipos de puntos de enlace conserven un orden estricto de mensajes. Los intentos de suscribir puntos de enlace administrados por el cliente a temas de SNS FIFO provocan errores.

Los temas FIFO de SNS admiten las mismas capacidades de filtrado de mensajes que los temas estándar. Para obtener más información, consulte Filtrado de mensajes de Amazon SNS para temas FIFO y la publicación Simplifique su mensajería de publicación o suscripción con el filtrado de mensajes de Amazon SNS en el blog informático de AWS .

Filtrado de mensajes de Amazon SNS para temas FIFO

Los temas FIFO de Amazon SNS admiten el filtrado de mensajes. Si utiliza el filtrado de mensajes, se simplifica la arquitectura al descargar la lógica de enrutamiento de mensajes de los sistemas de publicadores y la lógica de filtrado de mensajes de los sistemas de suscriptores.

Cuando suscribe una cola FIFO o estándar de Amazon SQS a un tema FIFO de SNS, puede utilizar el filtrado de mensajes para especificar que el suscriptor recibe un subconjunto de mensajes, en lugar de todos ellos. Cada suscriptor puede establecer su propia política de filtrado como atributos de suscripción. En función de su alcance, la política de filtrado se compara con los atributos o el cuerpo del mensaje entrante. Si la política de filtrado coincide, el tema ofrece una copia del mensaje al suscriptor. Si no hay coincidencia, el tema no entrega una copia del mensaje.

En el caso de uso de ejemplo de administración de precios de partes de automóviles, suponga que se establecen las siguientes políticas de filtrado de Amazon SNS y el alcance de la política de filtrado es `MessageBody`:

- Para la cola mayorista, la política de filtrado `{"business":["wholesale"]}` coincide con cada mensaje que contiene una clave llamada `business` y con `wholesale` en el conjunto de valores. En el siguiente diagrama, una de las claves del mensaje m1 es `business` con el valor de `wholesale`. Una de las claves del mensaje m3 es `business` con un valor de `["wholesale,retail"]`. Por lo tanto, m1 y m3 coinciden con los criterios de la política de filtro y ambos mensajes se entregan a la cola mayorista.
- Para la cola minorista, la política de filtrado `{"business":["retail"]}` coincide con cada mensaje que contiene una clave llamada `business` y con `retail` en el conjunto de valores. En el diagrama, una de las claves del mensaje m2 es `business` con el valor de `retail`. Una de

las claves del mensaje m3 es `business` con el valor de `["wholesale,retail"]`. Por lo tanto, m2 y m3 coinciden con los criterios de la política de filtro y ambos mensajes se entregan a la cola minorista.

- Para la cola de análisis, deseamos que Amazon Athena reciba todos los registros, por lo que no se aplica ninguna política de filtrado.

Los temas FIFO de SNS admiten una variedad de operadores coincidentes, incluidos valores de cadena de atributos, valores numéricos de atributo y claves de atributo. Para obtener más información, consulte Filtrado de mensajes en Amazon SNS.

Los temas FIFO de SNS no entregan mensajes duplicados a los puntos de enlace suscritos. Para obtener más información, consulte Desduplicación de mensajes de Amazon SNS para temas FIFO.

Desduplicación de mensajes de Amazon SNS para temas FIFO

Los temas FIFO de Amazon SNS y las colas FIFO de Amazon SQS admiten la desduplicación de mensajes, que proporciona una entrega y procesamiento de mensajes exactamente una vez, siempre que se cumplan las siguientes condiciones:

- La cola FIFO de Amazon SQS suscrita existe y tiene permisos para que la entidad principal de servicio de Amazon SNS pueda entregar mensajes a la cola.
- El consumidor de cola FIFO de Amazon SQS procesa el mensaje y lo elimina de la cola antes de que venza el tiempo de espera de visibilidad.
- El tema de suscripción a Amazon SNS no tiene filtrado de mensajes Al configurar el filtrado de mensajes, los temas FIFO de Amazon SNS admiten la at-most-once entrega, ya que los mensajes se pueden filtrar en función de las políticas de filtrado de la suscripción.
- No hay interrupciones en la red que impidan el reconocimiento de la entrega del mensaje.

> **Note**
>
> La deduplicación de mensajes se aplica a todo un tema FIFO de Amazon SNS cuando el `FifoThroughputScope` atributo del tema está establecido en. `Topic` Cuando el atributo del tema `FifoThroughputScope` está establecido en`MessageGroup`, la deduplicación de mensajes se aplica a cada grupo de mensajes individual.

Al publicar un mensaje en un tema FIFO de Amazon SNS, el mensaje debe incluir un ID de desduplicación. Este ID se incluye en el mensaje que el tema FIFO de Amazon SNS entrega a las colas FIFO de Amazon SQS suscritas.

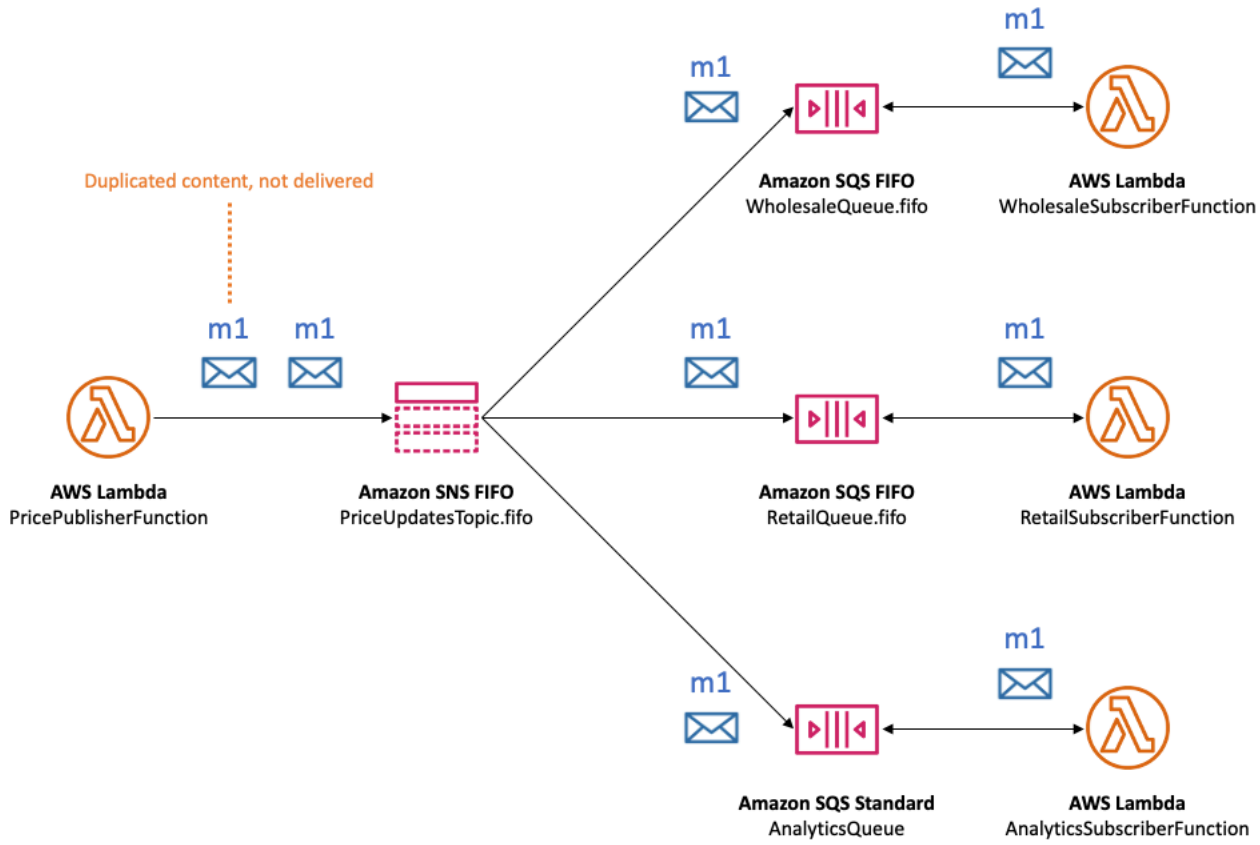
Si un mensaje con un ID de deduplicación concreto se publica correctamente en un tema FIFO de Amazon SNS, cualquier mensaje publicado con el mismo ID de deduplicación, dentro del intervalo de deduplicación de cinco minutos, se acepta pero no se entrega. El tema FIFO de Amazon SNS sigue rastreando el ID de deduplicación de mensajes, en el ámbito de deduplicación configurado por el atributo del tema`FifoThroughputScope`, incluso después de que el mensaje se entregue a los puntos de enlace suscritos.

Si se garantiza que el cuerpo del mensaje es único para cada mensaje publicado, puede habilitar la desduplicación basada en contenido para un tema FIFO de Amazon SNS y las colas FIFO de Amazon SQS suscritas. Amazon SNS utiliza el cuerpo del mensaje para generar un valor hash único que se utilizará como ID de desduplicación para cada mensaje, por lo que no es necesario establecer uno cuando envíe cada mensaje.

> **Note**
>
> Los atributos de mensaje no se incluyen en el cálculo hash.

Cuando la desduplicación basada en contenido está habilitada para un tema FIFO de Amazon SNS y se publica un mensaje con un identificador de desduplicación, el identificador de desduplicación publicado invalida el identificador de desduplicación basado en contenido generado.

En el caso de uso de ejemplo de la administración de precios de partes de automóviles, la empresa debe establecer un ID de desduplicación universalmente único para cada actualización de precios. Esto se debe a que el cuerpo del mensaje puede ser idéntico incluso cuando el atributo del mensaje es diferente para mayoristas y minoristas. Sin embargo, si la empresa agregó el tipo de negocio (mayorista y minorista) al cuerpo del mensaje junto con el ID del producto y el precio del producto, podrían habilitar la duplicación basada en contenido en el tema FIFO de Amazon SNS y en las colas FIFO de Amazon SQS suscritas.



Además del orden y la deduplicación de mensajes, los temas de FIFO de Amazon SNS incluyen el cifrado del lado del servidor de mensajes (SSE) AWS KMS con claves y la privacidad de los mensajes a través de puntos de enlace de VPC con. AWS PrivateLink Para obtener más información, consulte Seguridad de los mensajes de Amazon SNS para temas FIFO.

Seguridad de los mensajes de Amazon SNS para temas FIFO

Puede habilitar el cifrado para los temas de FIFO de Amazon SNS y las colas de FIFO de Amazon SQS mediante AWS Key Management Service () claves maestras del cliente ()AWS KMS. CMKs

- Puede crear nuevos temas y colas de FIFO cifrados o habilitar el cifrado de los ya existentes.
- Solo se cifra el cuerpo del mensaje. Los atributos de los mensajes, los metadatos de los recursos y las métricas de los recursos permanecen sin cifrar.

> **Note**
>
> Al agregar la encriptación a un tema o cola FIFO existente, no se encriptan los mensajes atrasados, y al eliminar la encriptación de un tema o cola, se mantienen encriptados los mensajes atrasados.

En los temas FIFO de SNS, se descifran los mensajes de inmediato antes de entregarlos a los puntos de enlace suscritos. En las colas FIFO de SQS, se descifra el mensaje justo antes de devolverlos a la aplicación del consumidor. Para obtener más información, consulte Cifrado de datos de Amazon SNS y la publicación Cifrado de mensajes publicados en Amazon SNS con AWS KMS en el blog informático de AWS .

Además, los temas FIFO de SNS y las colas FIFO de SQS admiten la privacidad de los mensajes con puntos de enlace de la VPC de interfaz impulsados por AWS PrivateLink. Mediante los puntos de enlace de interfaz, puede enviar mensajes desde subredes de Amazon Virtual Private Cloud (Amazon VPC) a temas y colas de FIFO sin atravesar Internet público. Este modelo mantiene los mensajes dentro de la AWS infraestructura y la red, lo que mejora la seguridad general de la aplicación. Cuando lo usa AWS PrivateLink, no necesita configurar una puerta de enlace a Internet, una traducción de direcciones de red (NAT) o una red privada virtual (VPN). Para obtener más información, consulte Protección del tráfico de Amazon SNS con puntos de conexión de VPC y la publicación Asegurar los mensajes publicados en Amazon SNS con AWS PrivateLink en el blog informático de AWS .

Los temas FIFO de SNS también admiten colas de mensajes fallidos y almacenamiento de mensajes en todas las zonas de disponibilidad. Para obtener más información, consulte Durabilidad de los mensajes de Amazon SNS para temas FIFO.

Durabilidad de los mensajes de Amazon SNS para temas FIFO

Los temas FIFO de Amazon SNS y las colas de Amazon SQS son duraderos. Ambos tipos de recursos almacenan los mensajes de forma redundante en varias zonas de disponibilidad y proporcionan colas de mensajes fallidos para manejar casos excepcionales.

En Amazon SNS, la entrega de mensajes falla cuando el tema de Amazon SNS no puede obtener acceso a una cola de Amazon SQS suscrita por un error en el cliente o en el servidor:

- Si los errores del cliente se producen cuando el tema FIFO de Amazon SNS tiene metadatos obsoletos de la suscripción. Dos causas comunes de errores del cliente se producen cuando el propietario de la cola de Amazon SQS realiza una de las siguientes acciones:
  - Elimina la cola.
  - Cambia la política de cola de forma que impida que la entidad principal de servicio de Amazon SNS le entregue mensajes.

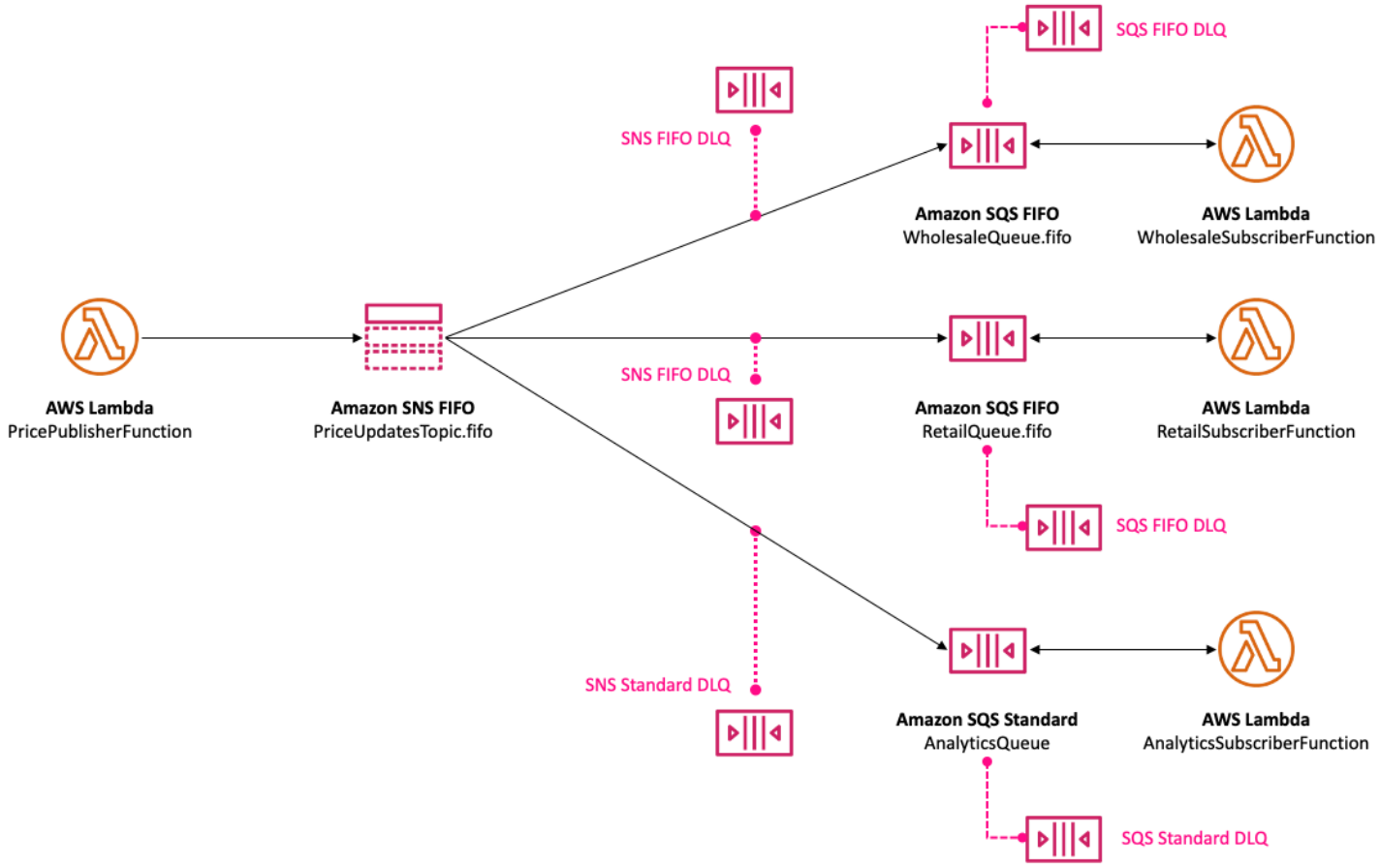
  Amazon SNS no vuelve a intentar entregar mensajes que han fallado debido a errores del cliente.
- Pueden producirse errores en el lado del servidor en estas situaciones:
  - El servicio Amazon SQS no está disponible.
  - Amazon SQS no puede procesar una solicitud válida del servicio Amazon SNS.

  Cuando se producen errores en el servidor, los temas FIFO de Amazon SNS vuelven a intentar entregarlos un máximo de 100 015 veces durante 23 días. Para obtener más información, consulte Reintento de entrega de mensajes de Amazon SNS.

Para cualquier tipo de error, Amazon SNS puede dejar de lado los mensajes a las colas de mensajes fallidos de Amazon SQS para que no se pierdan los datos.

En Amazon SQS, el procesamiento de mensajes falla cuando la aplicación de consumidor no recibe el mensaje, lo procesa y lo elimina de la cola. Cuando se produce un error en la cantidad máxima de solicitudes de recepción, Amazon SQS puede dejar de lado los mensajes a las colas de mensajes fallidos para que no se pierdan los datos.

En el caso de uso de ejemplo de administración de precios de piezas de automóviles, la empresa puede asignar una cola de mensajes fallidos (DLQ) de Amazon SQS a cada suscripción de tema FIFO de Amazon SNS, así como a cada cola de Amazon SQS suscrita. De esta manera, se protege a la empresa de cualquier pérdida de actualización de precios.



La cola de mensajes fallidos asociada a una suscripción a Amazon SNS debe ser una cola de Amazon SQS del mismo tipo que la cola suscriptora. Por ejemplo, la suscripción FIFO de Amazon SNS para una cola FIFO de Amazon SQS debe tener una cola FIFO de Amazon SQS como cola de mensajes fallidos. Del mismo modo, la suscripción FIFO de Amazon SNS para una cola estándar de Amazon SQS debe tener una cola estándar de Amazon SQS como cola de mensajes fallidos. Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS y la publicación Diseñar aplicaciones duraderas sin servidor DLQs para Amazon SNS, Amazon SQS AWS Lambda, en el blog de informática.AWS

Para prolongar la durabilidad y facilitar la recuperación de errores posteriores, los propietarios de los temas también pueden utilizar los temas FIFO para archivar los mensajes durante un máximo de 365 días. Los suscriptores de un tema pueden reproducir esos mensajes a un punto de conexión suscrito para recuperar los mensajes perdidos debido a un error en una aplicación posterior o para

replicar el estado de una aplicación existente. Para obtener más información, consulte Archivo y reproducción de mensajes de Amazon SNS para temas FIFO.

Archivo y reproducción de mensajes de Amazon SNS para temas FIFO

¿Qué es el archivo y reproducción de mensajes?

Amazon SNS ofrece una característica de archivo y reproducción de mensajes sin código, diseñada específicamente para temas FIFO (First-In-First-Out). Esta característica permite a los propietarios de los temas almacenar los mensajes directamente en el archivo de temas durante un máximo de 365 días y permite a los suscriptores reproducir estos mensajes cuando sea necesario. El archivo y la reproducción de los mensajes son esenciales para recuperar los mensajes perdidos y sincronizar las aplicaciones entre regiones o sistemas mediante la replicación de los estados.

Se puede acceder a esta funcionalidad a través de la AWS API AWS CloudFormation, el SDK y AWS Management Console.

Casos de uso clave

- Recuperación de mensajes: recupere los mensajes perdidos debido a errores en las aplicaciones posteriores reproduciéndolos en el punto de conexión del suscriptor.
- Replicación de estado: replique el estado de un sistema existente en un entorno nuevo reproduciendo los mensajes desde una marca de tiempo específica.
- Corrección de errores: reenvíe los mensajes perdidos durante las interrupciones para garantizar que todos los eventos se procesen correctamente.

Componentes del archivo y reproducción de mensajes

Administre el archivado y la reproducción de mensajes para los temas de FIFO de Amazon SNS, lo que incluye establecer períodos de retención, monitorear el uso de los mensajes archivados, iniciar las reproducciones CloudWatch mediante los atributos de suscripción y comprender los permisos necesarios para modificar e iniciar las repeticiones.

Archivo de mensajes

- El propietario del tema habilita la característica de archivo y establece el período de retención de los mensajes, que puede ser de hasta 365 días. Para obtener más información, consulte Archivo de mensajes de Amazon SNS para propietarios de temas FIFO.
- CloudWatch las métricas ayudan a monitorear los mensajes archivados.

Reproducción de mensajes

- Un suscriptor inicia una reproducción y selecciona el intervalo de tiempo para que los mensajes se vuelvan a procesar en el punto de conexión suscrito. Para obtener más información, consulte Reproducción de mensajes de Amazon SNS para los suscriptores de temas FIFO.
- La reproducción se administra mediante los atributos de suscripción con la característica `ReplayPolicy`.

Permisos adecuados

- **`SetSubscriptionAttributes`**: necesario para configurar o modificar la configuración de reproducción mediante el atributo `ReplayPolicy` de una suscripción.
- **`Subscribe`**: necesario para adjuntar una nueva suscripción e iniciar las reproducciones.
- **`GetTopicAttributes`**: permite ver las propiedades del tema, pero el inicio de la reproducción gira principalmente en torno a la administración de la suscripción.

Archivo de mensajes de Amazon SNS para propietarios de temas FIFO

El archivo de mensajes proporciona la posibilidad de archivar una sola copia de todos los mensajes publicados en el tema. Puede almacenar los mensajes publicados en el tema habilitando la política de archivo de mensajes correspondiente en el tema, que permite archivar los mensajes de todas las suscripciones vinculadas a ese tema. Los mensajes se pueden archivar durante un mínimo de un día y un máximo de 365 días.

Se aplican cargos adicionales al establecer una política de archivo. Para obtener información acerca de los precios, consulte Precios de Amazon SNS.

Crea una política de archivado de mensajes mediante el AWS Management Console

Utilice esta opción para crear una nueva política de archivo de mensajes con la AWS Management Console.

1. Inicie sesión en la consola de Amazon SNS.
2. Elija un tema o cree uno nuevo. Para obtener más información acerca de la creación de temas, consulte Creación de un tema de Amazon SNS.

> **Note**
>
> El archivado y la reproducción de mensajes de Amazon SNS solo están disponibles para temas de FIFO application-to-application (A2A).

3. En la página Editar tema, expanda la sección Política de archivo.
4. Habilite la característica de política de archivo e ingrese el número de días durante los que desea archivar los mensajes en el tema.
5. Elija Guardar cambios.

Para consultar, editar y desactivar una política de temas de archivo de mensajes

- En la página de Detalles del tema, la Política de retención muestra el estado de la política de archivo, incluida la cantidad de días para los que está configurada. Seleccione la pestaña Política de archivo para ver los siguientes detalles del archivo de mensajes:
  - Estado: el estado de archivo y reproducción aparece como activo cuando se aplica una política de archivo. El estado de archivo y reproducción aparece como inactivo cuando la política de archivo se establece en un objeto JSON vacío.
  - Periodo de retención de mensajes: el número de días especificado para la retención de mensajes.
  - Fecha de inicio del archivo: fecha a partir de la cual los suscriptores pueden reproducir los mensajes.
  - Vista previa en JSON: la vista previa en JSON de la política de archivo.
- (Opcional) Para editar una política de archivo, vaya a la página de resumen del tema y elija Editar.
- (Opcional) Para desactivar una política de archivo, vaya a la página de resumen del tema y elija Editar. Desactive la política de archivo y elija Guardar cambios.
- (Opcional) Para eliminar un tema con una política de archivo, primero debe desactivar la política de archivo tal y como se describió anteriormente.

> **Important**
>
> Para evitar la eliminación accidental de mensajes, no puede eliminar un tema con una política de archivo de mensajes activa. La política de archivo de mensajes del tema se debe desactivar antes de poder eliminar el tema. Al desactivar una política de archivo de mensajes, Amazon SNS elimina todos los mensajes de archivo. Al eliminar un tema, se eliminan las suscripciones y es posible que no se entreguen los mensajes en tránsito.

Crear una política de archivo de mensajes con la API

Para crear una política de archivo de mensajes mediante la API, debe agregar el atributo `ArchivePolicy` al tema. Puede configurar `ArchivePolicy` con las acciones de la API `CreateTopic` y `SetTopicAttributes`. `ArchivePolicy` tiene un valor único, `MessageRetentionPeriod`, que representa el número de días que Amazon SNS retiene los mensajes. Para activar el archivo de mensajes para el tema, establezca `MessageRetentionPeriod` en un valor entero mayor que cero. Por ejemplo, para retener mensajes en el archivo durante 30 días, establezca `ArchivePolicy` en:

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

Para desactivar el archivo de mensajes para el tema y borrar el archivo, desactive `ArchivePolicy`, de la siguiente manera:

```
{}
```

Crear una política de archivo de mensajes mediante el SDK

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte `credentials`Archivos `config` y compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

En el siguiente ejemplo de código, se muestra cómo configurar `ArchivePolicy` para que un tema de Amazon SNS retenga todos los mensajes publicados en ese tema durante 30 días.

```
// 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);
```

Cree una política de archivado de mensajes mediante AWS CloudFormation

Para crear una política de archivado utilizando, AWS CloudFormation consulte `AWS::SNS::Topic`la Guía del AWS CloudFormation usuario.

Conceder acceso a un archivo cifrado

Antes de que un suscriptor pueda empezar a reproducir mensajes de un tema cifrado, debe completar los siguientes pasos. Como los mensajes anteriores se reproducen, Amazon SNS debe disponer de acceso de `Decrypt` a la clave de KMS que se utilizó para cifrar los mensajes del archivo.

1. Cuando cifra los mensajes con una clave de KMS y los almacena en el tema, debe conceder a Amazon SNS la capacidad de descifrar estos mensajes mediante una política de claves. Para obtener más información, consulte Conceder permisos de descifrado a Amazon SNS.
2. Habilitar AWS KMS para Amazon SNS. Para obtener más información, consulte Configuración de AWS KMS permisos.

> **Important**
>
> Cuando agregue las secciones nuevas a su política de clave de KMS, no cambie las secciones existentes en la política. Si el cifrado está habilitado en un tema y la clave de KMS está desactivada o eliminada o la política de claves de KMS no está configurada

correctamente para Amazon SNS, Amazon SNS no podrá reproducir los mensajes a los suscriptores.

Conceder permisos de descifrado a Amazon SNS

Para que Amazon SNS acceda a los mensajes cifrados desde el archivo del tema y los reproduzca en los puntos de conexión suscritos, debe habilitar el principio del servicio de Amazon SNS para descifrar estos mensajes.

A continuación, se muestra una política de ejemplo que se requiere para permitir que la entidad principal del servicio de Amazon SNS descifre los mensajes almacenados durante una reproducción del historial de mensajes dentro del tema.

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

Supervisa las métricas del archivo de mensajes con Amazon CloudWatch

Puedes monitorizar los mensajes archivados con Amazon CloudWatch con las siguientes métricas. Para recibir notificaciones de anomalías en tus cargas de trabajo y evitar que se vean afectadas, puedes configurar las CloudWatch alarmas de Amazon en función de estas métricas. Para obtener más información, consulta Registro y monitoreo en Amazon SNS.

Métrica	Descripción
ApproximateNumberOfMessagesArchived	Proporciona al propietario del tema el número total de mensajes archivados en el archivo de temas, en una resolución de 60 minutos.

Métrica	Descripción
ApproximateNumberOfBytesArchived	Proporciona al propietario del tema el número total de bytes archivados en todos los mensajes del archivo de temas, con una resolución de 60 minutos.
NumberOfMessagesArchiveProcessing	Proporciona al propietario del tema el número de mensajes guardados en el archivo de temas durante el intervalo en una resolución de 1 minuto.
NumberOfBytesArchiveProcessing	Proporciona al propietario del tema el número total de bytes guardados en el archivo de temas durante el intervalo en una resolución de 1 minuto.

La API `GetTopicAttributes` tiene una propiedad `BeginningArchiveTime` que representa la marca temporal más antigua en la que un suscriptor puede iniciar una reproducción. A continuación, se muestra una respuesta de ejemplo para esta acción de la API:

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

Reproducción de mensajes de Amazon SNS para los suscriptores de temas FIFO

La reproducción de Amazon SNS permite a los suscriptores de los temas recuperar y volver a entregar los mensajes archivados del almacén de datos de temas a un punto final suscrito.

- Los mensajes se pueden reproducir inmediatamente después de crear la suscripción.
- Un mensaje reproducido conserva el mismo contenido y `MessageId` que `Timestamp` el original.

- El mensaje incluye un `Replayed` atributo para indicar que se trata de un mensaje reproducido.
- Para reproducir solo mensajes específicos, aplique una política de filtrado a su suscripción.

Para obtener más información sobre el filtrado de mensajes, consulte Filtrar los mensajes reproducidos.

Cree una política de reproducción de mensajes mediante el AWS Management Console

Utilice esta opción para crear una nueva política de reproducción con la AWS Management Console.

1. Inicie sesión en la consola de Amazon SNS.
2. Elija una suscripción a un tema o cree uno nuevo. Para obtener más información acerca de la creación de suscripciones, consulte Creación de una suscripción a un tema de Amazon SNS.
3. Para iniciar la reproducción del mensaje, vaya al menú desplegable Reproducción y elija Iniciar reproducción.
4. Desde el modal de periodo de reproducción, seleccione las siguientes opciones:
   a. Elija la fecha y la hora de inicio de la reproducción: elija la fecha (YYYY/MM/DDformato) y la hora (formato hh:mm:ss de 24 horas) a partir de las que quiere empezar a reproducir los mensajes archivados. La hora de inicio debe ser posterior al inicio de la hora aproximada de archivo.
   b. (Opcional) Elige la fecha y hora de finalización de la reproducción: elige la fecha (YYYY/MM/DDformato) y la hora (formato hh:mm:ss de 24 horas) en las que quieres dejar de reproducir los mensajes archivados.
   c. Elija Iniciar la reproducción.
5. (Opcional) Para detener la reproducción de un mensaje, vaya a la página de detalles de la suscripción y elija Detener la reproducción en el menú desplegable de reproducción.
6. (Opcional) Para supervisar las métricas de reproducción de mensajes desde este flujo de trabajo mediante, consulte. CloudWatch Supervisa las métricas de reproducción de mensajes con Amazon CloudWatch

Para ver y editar una política de reproducción de mensajes

Desde la página de detalles de suscripción, podrá realizar las acciones siguientes:

- Para ver el estado de la reproducción del mensaje, el campo de estado de la reproducción muestra los siguientes valores:
  - Completada: la reproducción ha vuelto a entregar correctamente todos los mensajes y ahora muestra los mensajes recién publicados.
  - En curso: la reproducción está reproduciendo actualmente los mensajes seleccionados.
  - Con error: la reproducción no se ha podido completar.
  - Pendiente: el estado predeterminado cuando se inicia la reproducción.
- (Opcional) Para modificar una política de reproducción de un mensaje, vaya a la página de detalles de la suscripción y elija Iniciar la reproducción en el menú desplegable de Reproducción. Al iniciar una repetición, se sustituirá la reproducción existente.

Agregar una política de reproducción a la suscripción mediante la API

Para reproducir los mensajes archivados, utilice el atributo `ReplayPolicy`. `ReplayPolicy` se puede usar con las acciones de la API `Subscribe` y `SetSubscriptionAttributes`. Esta política tiene los siguientes valores:

- **`StartingPoint`** (obligatorio): indica desde dónde empezar a reproducir los mensajes.
- **`EndingPoint`** (opcional): indica cuándo dejar de reproducir los mensajes. Si `EndingPoint` se omite, la reproducción continuará hasta que se ajuste a la hora actual.
- **`PointType`** (obligatorio): establece el tipo de puntos de inicio y final. Actualmente, el valor admitido `PointType` es `Timestamp`.

Por ejemplo, para recuperarse de un error posterior y volver a enviar todos los mensajes durante un periodo de dos horas el 1 de octubre de 2023, use la acción de la API `SetSubscriptionAttributes` para configurar una `ReplayPolicy` de la siguiente manera:

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

Para reproducir todos los mensajes enviados al tema a partir del 1 de octubre de 2023 y seguir recibiendo todos los mensajes recién publicados sobre el tema, use la acción de la API

`SetSubscriptionAttributes` para configurar una `ReplayPolicy` en la suscripción de la siguiente manera:

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

Para comprobar que un mensaje se ha reproducido, se agrega el atributo booleano `Replayed` a cada mensaje reproducido.

Agregar una política de reproducción a la suscripción mediante el SDK

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte `credentials`Archivos `config` y compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

El siguiente ejemplo de código muestra cómo configurar `ReplayPolicy` en una suscripción para volver a entregar mensajes del archivo del tema FIFO de Amazon SNS durante un periodo de 2 horas el 1 de octubre 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);
```

Entendiendo el EndingPoint

Cuando se aplica un atributo `ReplayPolicy` a una suscripción de Amazon SNS, el valor `EndingPoint` es opcional. Si no se proporciona `EndingPoint`, la reproducción empezará a partir del `StartingPoint` especificado y continuará hasta alcanzar la hora actual, incluido el procesamiento de los mensajes recién publicados. Una vez actualizada, la suscripción funcionará como una suscripción normal y recibirá los mensajes nuevos a medida que se publiquen.

Si se especifica un `EndingPoint`, el servicio reproducirá los mensajes desde el `StartingPoint` hasta el `EndingPoint` y, a continuación, se detendrá. Esta acción pone en pausa la suscripción. Mientras la suscripción esté en pausa, los mensajes recién publicados no se entregarán al punto de conexión suscrito.

Para reanudar la entrega de mensajes, aplique un nuevo atributo `ReplayPolicy` sin proporcionar un `EndingPoint` y establezca el `StartingPoint` en el punto en el tiempo en el que desee seguir recibiendo mensajes. Por ejemplo, para reanudar una suscripción desde el momento en que finalizó una reproducción anterior, establezca el nuevo `StartingPoint` en el `EndingPoint` proporcionado anteriormente.

Filtrar los mensajes reproducidos

El filtrado de mensajes de Amazon SNS le permite controlar los mensajes reproducidos que Amazon SNS reproduce en el punto de conexión del suscriptor. Cuando el filtrado y el archivo de mensajes están habilitados, Amazon SNS primero recupera el mensaje del almacén de datos del tema y, a continuación, lo aplica a la `FilterPolicy` de la suscripción. El mensaje se entrega al punto de conexión suscrito cuando hay una coincidencia; de lo contrario, el mensaje se filtra. Para obtener más información, consulte Políticas de filtro de suscripciones de Amazon SNS.

Supervisa las métricas de reproducción de mensajes con Amazon CloudWatch

Puedes monitorizar los mensajes reproducidos con Amazon CloudWatch con las siguientes métricas. Para recibir notificaciones de anomalías en tus cargas de trabajo y evitar que se vean afectadas, puedes configurar las CloudWatch alarmas de Amazon en función de estas métricas. Para obtener más información, consulta Registro y monitoreo en Amazon SNS.

Métrica	Descripción
NumberOfReplayedNotificationsDelivered	Proporciona al suscriptor el número total de mensajes reproducidos del archivo de temas, con una resolución de 1 minuto.
NumberOfReplayedNotificationsFailed	Proporciona al suscriptor el número total de mensajes reproducidos que han producido un error al entregar desde el archivo de temas, en una resolución de 1 minuto.

Ejemplos de código de Amazon SNS para temas FIFO

Utilice los siguientes ejemplos de código para integrar el ejemplo de caso práctico de administración de precios de autopartes con un tema FIFO de Amazon SNS y una cola FIFO de Amazon SQS o una cola estándar.

Uso de un SDK AWS

Con un AWS SDK, puede crear un tema FIFO de Amazon SNS estableciendo su `FifoTopic` atributo en. **`true`** Puede crear una cola FIFO de Amazon SQS al establecer su atributo `FifoQueue` a **`true`**. Además, debe agregar el sufijo **`.fifo`** al nombre de cada recurso FIFO. Después de crear un tema o una cola FIFO, no puede convertirlo en un tema o cola estándar.

En el ejemplo de código siguiente, se crean estos recursos de cola FIFO y estándar:

- El tema FIFO de Amazon SNS que distribuye las actualizaciones de precios
- Las colas FIFO de Amazon SQS que proporcionan estas actualizaciones a las aplicaciones mayoristas y minoristas
- La cola estándar de Amazon SQS para la aplicación de análisis que almacena registros, que pueden consultarse para inteligencia empresarial (BI)
- Las suscripciones FIFO de Amazon SNS que conectan las tres colas al tema

En este ejemplo, se establecen las Políticas de filtrado en las suscripciones. Si prueba el ejemplo al publicar un mensaje en el tema, asegúrese de publicar el mensaje con el atributo de `business`. Especifique `retail` o `wholesale` en el valor de atributo. De lo contrario, el mensaje se filtra y no

se entrega a las colas suscritas. Para obtener más información, consulte Filtrado de mensajes de Amazon SNS para temas FIFO.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este ejemplo

- crea un tema FIFO de Amazon SNS, dos colas FIFO de Amazon SQS y una cola estándar.
- suscribe las colas al tema y publica un mensaje en el tema.

La prueba verifica la recepción del mensaje en cada cola. El ejemplo completo también muestra la incorporación de políticas de acceso y, al final, elimina los recursos.

```
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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
  - CreateTopic
  - Publish
  - Subscribe

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crea un tema FIFO de Amazon SNS, suscribe una cola FIFO de Amazon SQS al tema y publica un mensaje en el tema.

```
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 obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
  - CreateTopic
  - Publish
  - Subscribe

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crea un tema de FIFO, suscribe una cola FIFO de Amazon SQS al tema y publica un mensaje en un tema de 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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para SAP ABAP.
  - CreateTopic
  - Publish
  - Subscribe

Recepción de mensajes de suscripciones FIFO

Ahora puede recibir actualizaciones de precios en las tres aplicaciones suscritas. Como se muestra enthe section called “Caso de uso de temas FIFO”, el punto de entrada de cada aplicación de consumo es la cola Amazon SQS, que su AWS Lambda función correspondiente puede sondear automáticamente. Cuando una cola de Amazon SQS es un origen de eventos para una función de Lambda, Lambda escala su flota de sondeadores según sea necesario para consumir mensajes de forma eficiente.

Para obtener más información, consulte Uso AWS Lambda con Amazon SQS en la Guía para AWS Lambda desarrolladores. Para obtener información sobre cómo escribir sus propios encuestadores de colas, consulte Recomendaciones para las colas estándar y FIFO de Amazon SQS en la Guía para desarrolladores de Amazon Simple Queue Service y ReceiveMessageen la Referencia de la API de Amazon Simple Queue Service.

¿Usando AWS CloudFormation

AWS CloudFormation le permite utilizar un archivo de plantilla para crear y configurar un conjunto de AWS recursos juntos como una sola unidad. En esta sección, se incluye una plantilla de ejemplo que crea lo siguiente:

- El tema FIFO de Amazon SNS que distribuye las actualizaciones de precios
- Las colas FIFO de Amazon SQS que proporcionan estas actualizaciones a las aplicaciones mayoristas y minoristas
- La cola estándar de Amazon SQS para la aplicación de análisis que almacena registros, que pueden consultarse para inteligencia empresarial (BI)
- Las suscripciones FIFO de Amazon SNS que conectan las tres colas al tema
- Una política de filtro en la que se especifica que las aplicaciones de suscriptor reciben solo las actualizaciones de precios que necesitan.

> **Note**
>
> Si prueba este código de ejemplo al publicar un mensaje en el tema, asegúrese de publicar el mensaje con el atributo de `business`. Especifique `retail` o `wholesale` en el valor de atributo. De lo contrario, el mensaje se filtra y no se entrega a las colas suscritas.

```
{
  "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 obtener más información sobre la implementación de AWS recursos mediante una AWS CloudFormation plantilla, consulte Primeros pasos en la Guía del AWS CloudFormation usuario.

Filtrado de mensajes en Amazon SNS

De forma predeterminada, un suscriptor de un tema de Amazon SNS recibe todos los mensajes que se publican en el tema. Para recibir solo un subconjunto de los mensajes, un suscriptor debe asignar una política de filtrado a la suscripción del tema.

Una política de filtrado es un objeto JSON que contiene propiedades que definen qué mensajes recibe el suscriptor. Amazon SNS admite políticas que actúan en los atributos del mensaje o en el cuerpo del mensaje, de acuerdo con el alcance de la política de filtrado que haya establecido para la suscripción. Las políticas de filtrado del cuerpo del mensaje asumen que la carga útil del mensaje es un objeto JSON bien formado.

Si una suscripción no tiene una política de filtrado, el suscriptor recibe todos los mensajes publicados en el tema. Cuando publica un mensaje en un tema con una política de filtrado, Amazon SNS compara los atributos del mensaje o el cuerpo del mensaje con las propiedades de la política de filtrado para cada una de las suscripciones del tema. Si todos los atributos del mensaje o propiedades del cuerpo del mensaje satisfacen las condiciones especificadas en la política de filtrado, Amazon SNS envía el mensaje al suscriptor. De lo contrario, Amazon SNS no envía el mensaje a ese suscriptor.

Para obtener más información, consulte Filtrar mensajes publicados en temas.

Alcance de políticas de filtrado de suscripciones de Amazon SNS

El atributo de `FilterPolicyScope` suscripción permite definir el ámbito de filtrado estableciendo uno de los siguientes valores:

- `MessageAttributes`— Aplica la política de filtrado a los atributos de los mensajes (configuración predeterminada).
- `MessageBody`— Aplica la política de filtrado al cuerpo del mensaje.

> **Note**
>
> Si no se define el alcance de la política de filtrado para una política de filtrado existente, el alcance se establece de forma predeterminada en `MessageAttributes`.

Políticas de filtro de suscripciones de Amazon SNS

Una política de filtrado de suscripciones le permite especificar nombres de propiedad y asignar una lista de valores a cada nombre de propiedad. Para obtener más información, consulte Filtrado de mensajes en Amazon SNS.

Cuando Amazon SNS evalúa los atributos del mensaje o las propiedades de cuerpo del mensaje con la política de filtrado de suscripciones, omite los que no están especificados en la política.

> **Important**
>
> AWS los servicios como IAM y Amazon SNS utilizan un modelo de computación distribuida denominado consistencia eventual. Los añadidos o cambios a una política de filtro de suscripción pueden tardar hasta 15 minutos en tener efecto.

Una suscripción acepta un mensaje con las siguientes condiciones:

- Cuando el alcance de la política de filtrado se establece en `MessageAttributes`, cada nombre de propiedad de la política de filtrado coincide con el nombre de un atributo de mensaje. Para cada nombre de propiedad coincidente de la política de filtrado, al menos un valor de propiedad coincide con el valor del atributo del mensaje.
- Cuando el alcance de la política de filtrado se establece en `MessageBody`, cada nombre de propiedad de la política de filtrado coincide con el nombre de una propiedad de cuerpo de mensaje. Para cada nombre de propiedad coincidente de la política de filtrado, al menos un valor de propiedad coincide con el valor de la propiedad del cuerpo del mensaje.

Amazon SNS admite actualmente los siguientes operadores de filtro:

- Lógica AND
- Lógica OR
- Operador OR
- Coincidencia de claves
- Coincidencia exacta de valores numéricos
- El valor numérico es cualquier cosa menos coincidente
- Coincidencia de rango de valor numérico

- Coincidencia exacta de valor de cadena
- El valor de cadena es cualquier cosa menos coincidente
- Coincidencia de cadenas con un prefijo con cualquier cosa menos el operador
- El valor de la cadena es igual a ignorar mayúsculas y minúsculas
- Coincidencia de la dirección IP con el valor de cadena
- Coincidencia de prefijo de valor de cadena
- Coincidencia de sufijo de valor de cadena

Políticas de filtrado de ejemplo de Amazon SNS

En el siguiente ejemplo, se muestra una carga de mensaje entregada por un tema de Amazon SNS que procesa transacciones de los clientes.

El primer ejemplo incluye el campo `MessageAttributes` con atributos que describen la transacción:

- Intereses del cliente
- Nombre del almacén
- Estado del evento
- Precio de compra en USD

Como este mensaje incluye el campo `MessageAttributes`, cualquier suscripción a un tema que establezca una `FilterPolicy` puede aceptar o rechazar el mensaje de forma selectiva, siempre y cuando `FilterPolicyScope` esté configurado en `MessageAttributes` en la suscripción. Para obtener información sobre cómo aplicar atributos a un mensaje, consulte Atributos de mensajes de 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"
      }
   }
}
```

En el siguiente ejemplo se muestran los mismos atributos incluidos en el campo `Message`, también denominado message payload (carga del mensaje) o message body (cuerpo del mensaje). La suscripción al tema que incluye una `FilterPolicy` puede aceptar o rechazar el mensaje de forma selectiva, siempre y cuando `FilterPolicyScope` esté configurado en `MessageBody` en la suscripción.

```
{
"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"
}
```

Las siguientes políticas de filtro aceptan o rechazan mensajes en función de los nombres de propiedad y valores.

Política que acepta el mensaje de ejemplo

Las propiedades de la siguiente política de filtrado de suscripciones coinciden con los atributos asignados en el mensaje de ejemplo. Tenga en cuenta que la misma política de filtrado funciona para un `FilterPolicyScope` si está configurado en `MessageAttributes` o `MessageBody`. Cada suscriptor elige el alcance de filtrado en función de la composición de los mensajes que recibe del tema.

Si la propiedad de esta política no coincide con un atributo asignado en el mensaje, la política rechaza el mensaje.

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

Política que rechaza el mensaje de ejemplo

La siguiente política de filtrado de suscripciones tiene varias discrepancias entre las propiedades y los atributos asignados en el mensaje de ejemplo. Por ejemplo, como el nombre de propiedad `encrypted` no aparece en los atributos del mensaje, esta propiedad de la política provoca que se rechace el mensaje, con independencia del valor que tenga asignado.

Si se producen discrepancias, la política rechaza el mensaje.

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

```
    ]
}
```

Restricciones de las políticas de filtrado de Amazon SNS

Al configurar políticas de filtrado en Amazon SNS, hay algunas reglas importantes que debe tener en cuenta. Estas reglas ayudan a garantizar la aplicación efectiva de las políticas de filtrado y, al mismo tiempo, a mantener el rendimiento y la compatibilidad del sistema.

Restricciones de política comunes

Al configurar las políticas de filtrado en Amazon SNS, siga estas reglas importantes para garantizar que funcionen de manera eficaz y, al mismo tiempo, mantengan el rendimiento y la compatibilidad del sistema:

- Coincidencia de cadenas: en el caso de la coincidencia de cadenas en la política de filtrado, la comparación distingue entre mayúsculas y minúsculas.
- Coincidencia numérica: en el caso de las coincidencias numéricas, el valor puede oscilar entre -10^9 y 10^9 (entre 1000 millones y 1000 millones), con cinco dígitos de precisión después de la coma decimal.
- Complejidad de la política de filtrado: la combinación total de valores de una política de filtrado no debe superar los 150. Para calcular la combinación total, multiplique el número de valores de cada matriz en la política de filtrado.
- Número límite de claves: una política de filtrado puede tener un máximo de cinco claves.

Consideraciones adicionales

- El código JSON de la política de filtro puede contener lo siguiente:
  - Cadenas entre comillas
  - Números
  - Las palabras clave `true`, `false` y `null`, sin comillas

- Cuando utilice la API de Amazon SNS, debe pasar el JSON de la política de filtrado como una cadena UTF-8 válida.

- El tamaño máximo de una política de filtrado es de 256 KB.

- De forma predeterminada, puede tener hasta 200 políticas de filtrado por tema y 10 000 políticas de filtrado por AWS cuenta.

  Este límite de política no impedirá que las suscripciones en cola de Amazon SQS se creen con la API. `Subscribe` Sin embargo, se producirá un error al asociar la política de filtro a la llamada a la API `Subscribe` (o a la llamada a la API `SetSubscriptionAttributes`).

  Para aumentar esta cuota, puede usar AWS Service Quotas.

Restricciones de la política para el filtrado basado en atributos

El filtrado basado en atributos es la opción predeterminada. `FilterPolicyScope` está configurado en `MessageAttributes` en la suscripción.

- Amazon SNS no acepta una política de filtrado anidado para el filtrado basado en atributos.
- Amazon SNS compara las propiedades de la política solo con los atributos de mensaje que tienen los siguientes tipos de datos:
  - `String`
  - `String.Array`

> **Important**
>
> Cuando utilice el filtrado basado en atributos en Amazon SNS, debe evitar dos veces algunos caracteres especiales, en concreto:
>
> - Comillas dobles («)
> - Barras invertidas ()
>
> Si no se eliminan dos veces estos caracteres, la política de filtrado no coincidirá con los atributos de un mensaje publicado y la notificación no se entregará.

Consideraciones adicionales

- No se recomienda pasar objetos a matrices, ya que puede producir resultados inesperados debido al anidamiento, que no es compatible con el filtrado basado en atributos. Use el filtrado basado en carga para políticas anidadas.
- `Numberse` admite para valores de atributos numéricos.
- Amazon SNS ignora los atributos de los mensajes con el tipo de datos binario.

Ejemplo de política de complejidad:

En el siguiente ejemplo de política, la primera clave tiene tres operadores de coincidencia, la segunda tiene un operador de coincidencia y la tercera tiene dos operadores de coincidencia.

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

La combinación total se calcula como el producto del número de operadores de coincidencia de cada clave de la política de filtrado:

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

Restricciones de la política para el filtrado basado en carga

Para cambiar del filtrado basado en atributos (predeterminado) al filtrado basado en cargas, debe configurar `FilterPolicyScope` en `MessageBody` en la suscripción.

- Amazon SNS acepta una política de filtrado anidado para el filtrado basado en carga.
- En el caso de una política anidada, solo las claves de hoja se tienen en cuenta para el límite de cinco teclas.

Ejemplo de política para el límite de claves:

En el siguiente ejemplo de política:

- Hay dos teclas de hoja: `key_c` y`key_e`.
- `key_c`tiene cuatro operadores de coincidencia con un nivel anidado de tres y **`key_e`** tres operadores de coincidencia con un nivel anidado de dos.

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

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

La combinación total se calcula como el producto del número de operadores de coincidencia y el nivel anidado de cada clave de la política de filtrado:

```
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 AND/OR

Utilice la lógica ANDOR en las políticas de filtrado para hacer coincidir los atributos de los mensajes o las propiedades del cuerpo del mensaje en Amazon SNS. Esto permite un filtrado de mensajes más preciso y flexible.

Lógica AND

Puede aplicar la lógica AND utilizando varios nombres de propiedad.

Considere la siguiente política:

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

Coincide con cualquier atributo de mensaje o propiedad de cuerpo de mensaje con el valor de `customer_interests` establecido en `rugby` y el valor de `price_usd` establecido en un número mayor de 100.

> **Note**
>
> No puede aplicar la lógica AND a los valores del mismo atributo.

Lógica OR

Puede aplicar la lógica OR asignando varios valores a un nombre de propiedad.

Considere la siguiente política:

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

Coincide con cualquier atributo de mensaje o propiedad de cuerpo de mensaje con el valor de `customer_interests` establecido en `rugby`, `football` o `baseball`.

Operador OR

Puede utilizar el operador `"$or"` para definir de forma explícita una política de filtrado que exprese la relación OR entre varios atributos de la política.

Amazon SNS solo reconoce una relación `"$or"` cuando la política ha cumplido todas las condiciones siguientes. Cuando no se cumplen todas estas condiciones, `"$or"` se trata como un nombre de atributo normal, igual que cualquier otra cadena de la política.

- Hay un atributo de campo `"$or"` en la regla seguido de una matriz, por ejemplo `“$or” : []`.
- Hay al menos 2 objetos en la matriz de `"$or"`: `"$or": [{}, {}]`.
- Ninguno de los objetos de la matriz de `"$or"` tiene nombres de campo que sean palabras clave reservadas.

De lo contrario, `"$or"` se trata como un nombre de atributo normal, igual que otras cadenas de la política.

La siguiente política no se analiza como una relación OR porque el número y el prefijo son palabras clave reservadas.

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

Ejemplos de operadores **`OR`**

0R estándar:

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

La lógica de filtrado de esta política es:

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

Coincide con cualquiera de los siguientes conjuntos de atributos de mensaje:

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

o

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

También coincide con los siguientes cuerpos de mensaje:

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

o

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

Restricciones de políticas que incluyen relaciones **OR**

Considere la siguiente política:

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

La lógica de esta política también se puede simplificar de la siguiente manera:

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

El cálculo de la complejidad de las políticas con relaciones OR se puede simplificar como la suma de las complejidades combinadas de cada declaración OR.

La combinación total se calcula del siguiente modo:

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

`source` tiene un valor, `metricName` tiene dos valores, `metricType` tiene un valor, `metricId` tiene dos valores y `spaceId` tiene tres valores.

Tenga en cuenta la siguiente política de filtrado anidado:

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

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

La lógica de esta política se puede simplificar de la siguiente manera:

```
("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")
```

El cálculo para las combinaciones totales es el mismo para las políticas no anidadas, excepto que debemos tener en cuenta el nivel de anidación de una clave.

La combinación total se calcula del siguiente modo:

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

`metricName` tiene dos valores, `namespace` tiene dos valores, `scope` es una clave anidada de dos niveles con un valor, `source` es una clave anidada de dos niveles con un valor y `type` es una clave anidada de dos niveles con un valor.

Coincidencia de claves

Utilice el `exists` operador en una política de filtrado para hacer coincidir los mensajes entrantes en función de si una propiedad específica está presente o ausente.

- `exists`solo funciona en los nodos de hoja (atributos finales de la estructura).
- No se aplica a los nodos intermedios dentro de una estructura JSON anidada.

- Utilice `"exists": true` para hacer coincidir los mensajes entrantes que incluyen la propiedad especificada. La clave debe tener un valor no nulo y no vacío.

Por ejemplo, la siguiente propiedad de política utiliza el operador `exists` con un valor de `true`:

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

Coincide con la lista de atributos de mensaje que contenga la clave de atributo `store`, como el siguiente:

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

También coincide con el siguiente cuerpo de mensaje:

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

Sin embargo, no coincide con la lista de atributos del mensaje sin la clave de atributo `store`, como el siguiente:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

- Utilice `"exists": false` para hacer coincidir los mensajes entrantes que no incluyan la propiedad especificada.

> **Note**
>
> `"exists": false` solo coincide si hay al menos un atributo. Si el conjunto de atributos está vacío, el filtro no coincide.

Por ejemplo, la siguiente propiedad de política utiliza el operador `exists` con un valor de `false`:

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

No coincide con la lista de atributos de mensaje que contenga la clave de atributo `store`, como el siguiente:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Sin embargo, coincide con la lista de atributos del mensaje sin la clave de atributo `store`, como el siguiente:

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

También coincide con el siguiente cuerpo de mensaje:

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

Coincidencia de valor numérico

Filtre los mensajes haciendo coincidir los valores numéricos con los valores de los atributos del mensaje o con los valores de las propiedades del cuerpo del mensaje. Los valores numéricos no están entre comillas en la política JSON. Puede utilizar las siguientes operaciones numéricas para filtrar.

> **Note**
>
> Los prefijos solo se admiten para la coincidencia de cadena.

Coincidencia exacta

Cuando un valor de propiedad de política incluye la palabra clave `numeric` y el operador `=`, coincide con cualquier atributo de mensajes o valores de propiedad de cuerpo de mensajes que tenga el mismo nombre y un valor numérico igual.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos de mensaje:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Coincidencia "anything-but"

Cuando el valor de una propiedad de política incluye la palabra clave `anything-but`, coincide con cualquier valor de atributo o propiedad del cuerpo del mensaje que no incluya ninguno de los valores de las propiedades de la política.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos de mensaje:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Además, coincide con el siguiente atributo de mensaje (porque contiene un valor que no es `100` ni `500`):

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

También coincide con el siguiente cuerpo de mensaje (porque contiene un valor que no es `100` ni `500`):

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

Sin embargo, no coincide con el siguiente atributo de mensaje:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

```
}
```

Coincidencia de intervalo de valores

Además del operador =, una propiedad de política numérica puede incluir los siguientes operadores: <, <=, > y >=.

Considere la siguiente propiedad de política:

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

Coincide con cualquier atributo de mensaje o propiedad de cuerpo de mensaje con valores numéricos negativos.

Considere otro atributo de mensaje:

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

Coincide con cualquier atributo de mensaje o propiedad de cuerpo de mensaje con números positivos hasta el 150 inclusive.

Coincidencia de valor de cadena

Filtre los mensajes haciendo coincidir los valores de cadena con los valores de los atributos del mensaje o los valores de las propiedades del cuerpo del mensaje. Los valores de cadena están entre comillas en la política JSON. Puede utilizar las siguientes operaciones de cadena para hacer coincidir los atributos o las propiedades del cuerpo del mensaje:

Coincidencia exacta

La coincidencia exacta se produce cuando un valor de propiedad de la política coincide con uno o varios valores de atributo del mensaje. En el `String.Array` caso de los atributos de tipo, cada elemento de la matriz se trata como una cadena independiente con fines de coincidencia.

Considere la siguiente propiedad de política:

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

Coincide con los siguientes atributos de mensaje:

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

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Sin embargo, no coincide con los siguientes atributos del mensaje:

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

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Coincidencia "anything-but"

Cuando el valor de una propiedad de política incluye la palabra clave `anything-but`, coincide con cualquier atributo de mensaje o valor del cuerpo del mensaje que no incluya ninguno de los valores de las propiedades de la política. `anything-but` se puede combinar con `"exists": false`. En el caso `String.Array` de los atributos de tipo, coincide si ninguno de los elementos de la matriz aparece en la propiedad de la política.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos del mensaje:

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

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Además, coincide con el siguiente atributo de mensaje (porque contiene un valor que no es `rugby` ni `tennis`):

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

Y también coincide con el siguiente cuerpo de mensaje (porque contiene un valor que no es `rugby` ni `tennis`):

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

Sin embargo, no coincide con los siguientes atributos del mensaje:

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

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Uso de un prefijo con el operador **anything-but**

Para la coincidencia de la cadena, también puede usar un prefijo con `anything-but` con el operador. Por ejemplo, la propiedad de política siguiente deniega el prefijo `order-`:

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

Coincide con cualquiera de los siguientes atributos:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Sin embargo, no coincide con el siguiente atributo de mensaje:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Equals-ignore-case coincidente

Cuando una propiedad de política incluye la palabra clave `equals-ignore-case`, realizará una coincidencia que no distingue entre mayúsculas y minúsculas con el atributo de mensaje o valor de propiedad del cuerpo.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos de mensaje:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Coincidencia de direcciones IP

Puede utilizar el operador `cidr` para verificar si un mensaje entrante se origina desde una dirección IP o subred específica.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos de mensaje:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Sin embargo, no coincide con el siguiente atributo de mensaje:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Coincidencia de prefijos

Cuando una propiedad de política incluye la palabra clave `prefix`, coincide con cualquier valor de atributo de mensaje o valor de propiedad de cuerpo que empiece por los caracteres especificados.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos de mensaje:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Sin embargo, no coincide con el siguiente atributo de mensaje:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Coincidencia de sufijos

Cuando una propiedad de política incluye la palabra clave `suffix`, coincide con cualquier valor de atributo de mensaje o valor de propiedad de cuerpo que termine por los caracteres especificados.

Considere la siguiente propiedad de política:

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

Coincide con cualquiera de los siguientes atributos de mensaje:

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

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

También coincide con los siguientes cuerpos de mensaje:

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

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

Sin embargo, no coincide con el siguiente atributo de mensaje:

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

Tampoco coincide con el siguiente cuerpo del mensaje:

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

Aplicación de una política de filtrado de suscripciones en Amazon SNS

El filtrado de mensajes en Amazon SNS le permite entregar mensajes a los suscriptores de forma selectiva en función de las políticas de filtrado. Estas políticas definen las condiciones que deben cumplir los mensajes para poder entregarse a una suscripción. Si bien la entrega de mensajes sin procesar es una opción que puede afectar al procesamiento de los mensajes, no es necesaria para que los filtros de suscripción funcionen.

Puede aplicar una política de filtro a una suscripción de Amazon SNS mediante la consola de Amazon SNS. O bien, para aplicar las políticas mediante programación, puede utilizar la API de Amazon SNS, AWS Command Line Interface el AWS CLI() o AWS cualquier SDK compatible con Amazon SNS. También puede usar. AWS CloudFormation

Habilitación de la entrega de mensajes sin procesar

La entrega de mensajes sin procesar garantiza que las cargas útiles de los mensajes se entreguen tal cual a los suscriptores, sin necesidad de codificación ni transformación adicionales. Esto puede resultar útil cuando los suscriptores necesitan el formato de mensaje original para su procesamiento. Sin embargo, la entrega de mensajes sin procesar no está directamente relacionada con la funcionalidad de los filtros de suscripción.

Aplicación de filtros de suscripciones

Para aplicar filtros de mensajes a una suscripción, debe definir una política de filtrado mediante la sintaxis JSON. Esta política especifica las condiciones que debe cumplir un mensaje para entregarse a la suscripción. Los filtros se pueden basar en atributos del mensaje, como los atributos del mensaje, la estructura del mensaje o incluso el contenido del mensaje.

Relación entre la entrega de mensajes sin procesar y los filtros de suscripción

Aunque habilitar la entrega de mensajes sin procesar puede afectar a la forma en que los suscriptores entregan y procesan los mensajes, no es un requisito previo para usar filtros de suscripción. Sin embargo, en situaciones en las que los suscriptores requieren el formato de mensaje original sin ninguna modificación, habilitar la entrega de mensajes sin procesar podría resultar beneficioso junto con los filtros de suscripción.

Consideraciones para un filtrado eficaz

Al implementar el filtrado de mensajes, tenga en cuenta los requisitos específicos de su aplicación y de sus suscriptores. Defina políticas de filtrado que coincidan con precisión con los criterios de entrega de mensajes para garantizar una distribución eficiente y específica de los mensajes.

> **Important**
>
> AWS los servicios como IAM y Amazon SNS utilizan un modelo de computación distribuida denominado consistencia eventual. Los añadidos o cambios a una política de filtro de suscripción pueden tardar hasta 15 minutos en tener efecto.

AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, seleccione Subscriptions (Suscripciones).
3. Seleccione una suscripción y, a continuación, elija Edit (Editar).
4. En la página Edit (Editar), amplíe la sección Subscription filter policy (Política de filtro de suscripción).
5. Elija entre el filtrado basado en atributos o el filtrado basado en cargas.
6. En el campo JSON editor (Editor JSON), proporcione el cuerpo JSON de su política de filtrado.
7. Elija Guardar cambios.

   Amazon SNS aplica la política de filtro a la suscripción.

AWS CLI

Para aplicar una política de filtrado con AWS Command Line Interface (AWS CLI), utilice el `set-subscription-attributes`comando, como se muestra en el siguiente ejemplo. Para la opción `--attribute-name`, especifique `FilterPolicy`. Para `--attribute-value`, especifique la política JSON.

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

Para proporcionar un objeto JSON válido para su política, incluya los nombres de atributos y valores entre comillas dobles. Incluya también todo el argumento de la política entre comillas. Para evitar que las comillas se interpreten como caracteres de escape, puede utilizar comillas simples para delimitar la política y comillas dobles para delimitar los nombres y los valores JSON, tal y como se muestra en el ejemplo anterior.

Si quieres cambiar de un filtrado de mensajes basado en atributos (predeterminado) a uno basado en cargas útiles, también puedes usar el set-subscription-attributescomando. Para la opción `--attribute-name`, especifique `FilterPolicyScope`. En `--attribute-value`, especifique `MessageBody`.

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

Para verificar que su política de filtro se ha aplicado, utilice el comando `get-subscription-attributes`. Los atributos del resultado deben mostrar la política de filtro para la clave `FilterPolicy`, tal y como se muestra en el ejemplo siguiente:

```
$ 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

En los siguientes ejemplos de código, se muestra cómo utilizar `SetSubscriptionAttributes`.

> **Important**
>
> Si utiliza el ejemplo de SDK for Java 2.x, la clase `SNSMessageFilterPolicy` no está disponible para usar. Para obtener instrucciones sobre cómo instalar esta clase, consulta el ejemplo del sitio web. GitHub

CLI

AWS CLI

Establecimiento de los atributos de suscripción

En el siguiente ejemplo de `set-subscription-attributes`, se establece el atributo `RawMessageDelivery` en una suscripción de 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 no genera ninguna salida.

En el siguiente ejemplo de `set-subscription-attributes`, se establece un atributo `FilterPolicy` en una suscripción de 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 no genera ninguna salida.

En el siguiente ejemplo de `set-subscription-attributes`, se elimina el atributo `FilterPolicy` de una suscripción de 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 no genera ninguna salida.

- Para obtener más información sobre la API, consulte SetSubscriptionAttributesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta SetSubscriptionAttributesla Referencia AWS SDK for Java 2.x de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta SetSubscriptionAttributesla AWS Referencia de API de SDK for Python (Boto3).

API de Amazon SNS

Para aplicar una política de filtro con la API de Amazon SNS, realice una solicitud a la acción `SetSubscriptionAttributes`. Establezca el parámetro `AttributeName` en `FilterPolicy` y el parámetro `AttributeValue` en el objeto JSON de la política de filtro.

Si quiere cambiar de filtrado de mensajes basado en atributos (predeterminado) a filtrado de mensajes basado en cargas, puede usar también la acción `SetSubscriptionAttributes`.

Establezca el parámetro `AttributeName` en `FilterPolicyScope` y el parámetro `AttributeValue` en `MessageBody`.

AWS CloudFormation

Para aplicar una política de filtrado AWS CloudFormation, utilice una plantilla JSON o YAML para crear una AWS CloudFormation pila. Para obtener más información, consulta la `FilterPolicy`propiedad del `AWS::SNS::Subscription` recurso en la Guía del AWS CloudFormation usuario y en la AWS CloudFormation plantilla de ejemplo.

1. Inicie sesión en la consola de AWS CloudFormation.
2. Elija Crear pila.
3. En la página Select Template (Seleccionar plantilla), elija Upload a template to Amazon S3 (Cargar una plantilla en Amazon S3), elija el archivo y, a continuación, elija Next (Siguiente).
4. En la página Specify Details (Especificar detalles), haga lo siguiente:

   a. Para Stack Name (Nombre de la pila), escriba `MyFilterPolicyStack`.

   b. Para myHttpEndpoint, escriba el punto final HTTP al que se va a suscribir al tema.

   > **Tip**
   >
   > Si no dispone de un punto de enlace HTTP, cree uno.

5. En la página Opciones, seleccione Siguiente.
6. En la página Review (Revisar), elija Create (Crear).

Eliminación de una política de filtrado de suscripciones en Amazon SNS

Para dejar de filtrar los mensajes que se envían a una suscripción, quite la política de filtro de la suscripción sobrescribiéndola con un cuerpo JSON vacío. Después de quitar la política, la suscripción acepta todos los mensajes que se han publicado en ella.

Uso del AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.

2. En el panel de navegación, seleccione Subscriptions (Suscripciones).
3. Seleccione una suscripción y, a continuación, elija Edit (Editar).
4. En la *EXAMPLE1-23bc-4567-d890-ef12g3hij456* página de edición, expanda la sección Política de filtro de suscripciones.
5. En el campo JSON editor (Editor de JSON), proporcione un cuerpo JSON vacío de su política de filtro: `{}`.
6. Elija Guardar cambios.

   Amazon SNS aplica la política de filtro a la suscripción.

Usando el AWS CLI

Para eliminar una política de filtrado con AWS CLI, usa el `set-subscription-attributes`comando y proporciona un cuerpo JSON vacío para el `--attribute-value` argumento:

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

Uso de la API de Amazon SNS

Para quitar una política de filtro con la API de Amazon SNS, realice una solicitud a la acción `SetSubscriptionAttributes`. Establezca el parámetro `AttributeName` en `FilterPolicy` y proporcione un cuerpo JSON vacío para el parámetro `AttributeValue`.

Protección de datos de mensajes en Amazon SNS

¿Qué es la protección de datos de mensajes?

La protección de datos de los mensajes protege los datos que se publican en sus temas de Amazon SNS mediante el uso de políticas de protección de datos para auditar, enmascarar, redactar o bloquear la información confidencial que se mueve entre aplicaciones o servicios. AWS

La función de protección de datos de mensajes escanea los datos en movimiento para detectar información de identificación personal (PII) e información médica protegida (PHI) mediante identificadores de datos. Tiene la opción de elegir usar identificadores de datos predefinidos (o administrados por Amazon SNS) (por ejemplo, nombres, direcciones, números de tarjetas de crédito y códigos de medicamentos con receta) o puede crear sus propios identificadores de datos personalizados, específicos para su caso de uso empresarial. Con la información analizada, la función de protección de datos de mensajes proporciona registros de auditoría detallados y le permite tomar medidas para proteger esos datos.

La función de protección de datos de mensajes permite realizar las siguientes acciones para ayudar a proteger la información confidencial del cliente:

- Auditar: audite hasta el 99 % de los datos que se publican en un tema de Amazon SNS. A continuación, puede optar por enviar los resultados a Amazon CloudWatch, Amazon S3 o Amazon Data Firehose.
- Anonimizar: enmascara o elimina información confidencial sin interrumpir la publicación o entrega de mensajes.
- Denegar: bloquea la transmisión de datos entre aplicaciones y AWS recursos si hay datos confidenciales en la carga útil.

> **Note**
>
> Amazon SNS admite la función de protección de datos de mensajes únicamente para los temas estándar de Amazon SNS.

¿Por qué debo utilizar la función de protección de datos de mensajes?

Si introduce la función de protección de datos de mensajes en sus programas de control, gestión de riesgos y conformidad, puede implementar políticas de protección de datos que le ayuden a identificar y evitar la fuga de datos. Esto proporciona a sus equipos herramientas que pueden ayudar a reducir los riesgos financieros, legales y reglamentarios al cumplir normas de privacidad, como la HIPAA, el RGPD, la PCI y la FedRAMP. También libera a los desarrolladores de la sobrecarga operativa asociada a la creación y administración de sus propias herramientas para proteger los datos confidenciales.

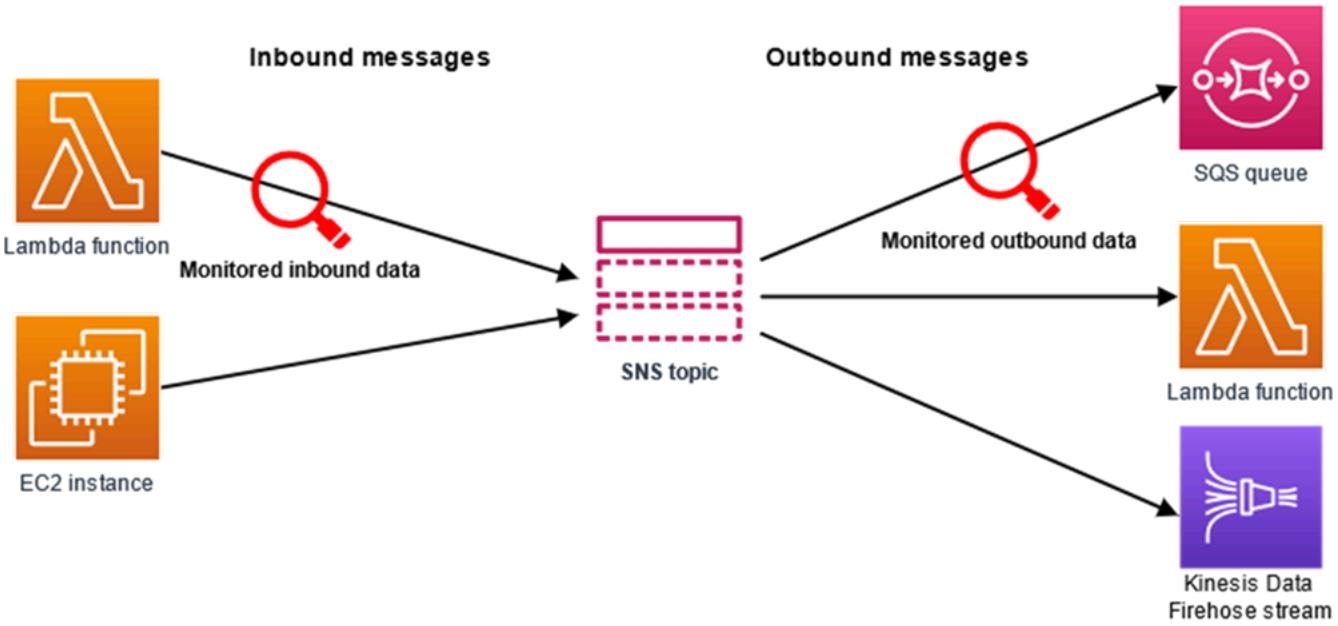
Por ejemplo, puede utilizar la función de protección de datos de mensajes para crear una política de auditoría que determine si alguno de sus sistemas envía o recibe datos confidenciales sin darse cuenta. Si los resultados de la auditoría indican que los sistemas están enviando información de tarjetas de crédito a sistemas que no la requieren, puede utilizar una política de bloqueo para impedir que se entreguen esos datos.

> **Note**
>
> Amazon SNS admite la función de protección de datos de mensajes únicamente para los temas estándar de Amazon SNS.

Descripción de las políticas de protección de datos de Amazon SNS

¿Qué son las políticas de protección de datos?

Amazon SNS utiliza políticas de protección de datos para seleccionar los datos confidenciales que desea analizar y las acciones que desea realizar para evitar que en sus temas de Amazon SNS se intercambien esos datos. Para seleccionar los datos confidenciales que le interesan, debe utilizar identificadores de datos. A continuación, la función de protección de datos de mensajes de Amazon SNS detecta los datos confidenciales mediante machine learning y la coincidencia de patrones. En respuesta a los identificadores de datos encontrados, puede definir una operación de auditoría, anonimización o denegación. Estas operaciones permiten registrar la información confidencial que se encuentra (o no se encuentra), enmascarar o eliminar información confidencial o denegar la entrega de mensajes.

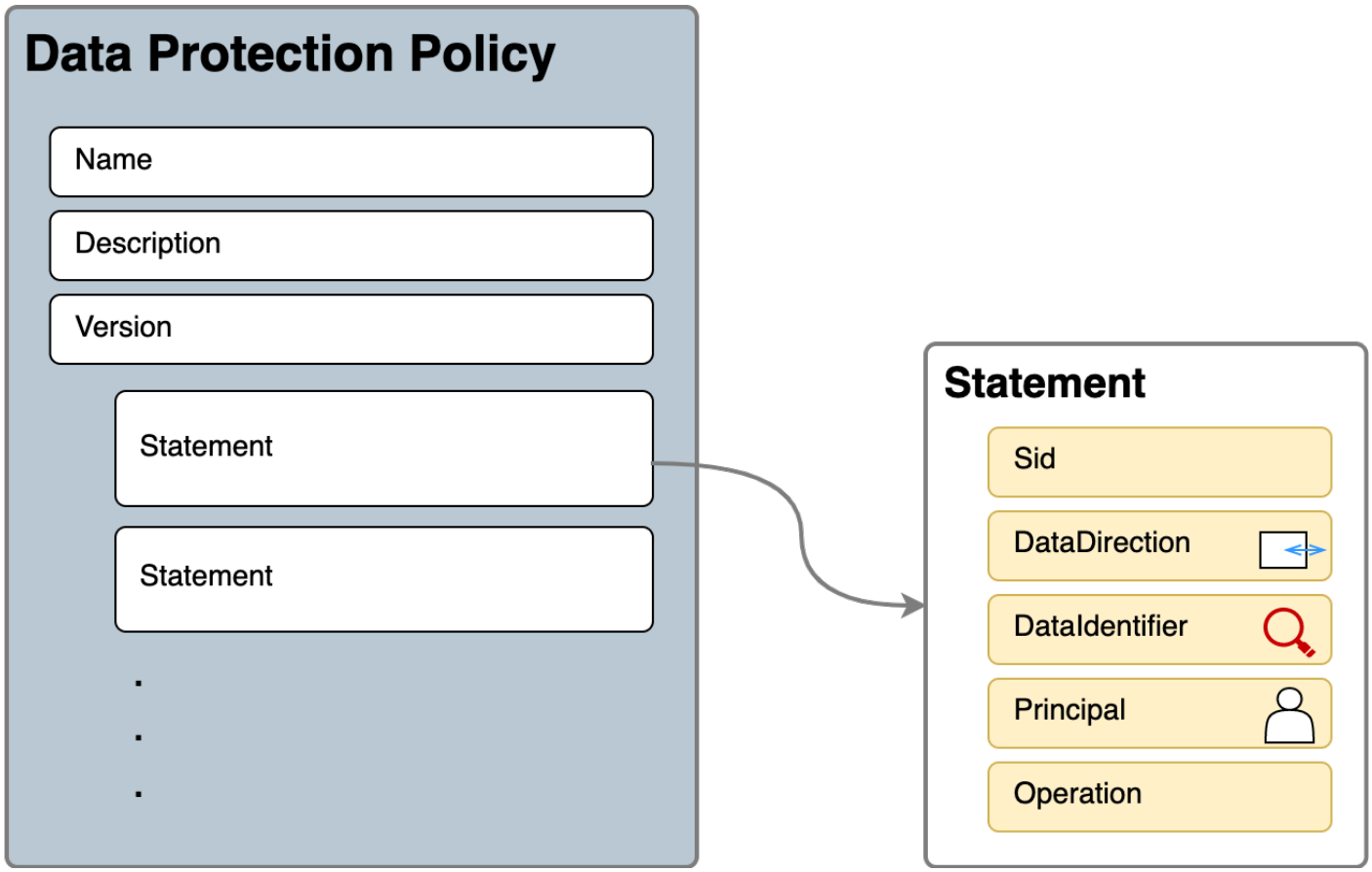


¿Cómo está estructurada la política de protección de datos?

Tal y como se muestra en la siguiente figura, un documento de la política de protección de datos incluye los siguientes elementos:

- Información opcional aplicable a toda la política en la parte superior del documento
- Una o varias instrucciones individuales

Cada instrucción incluye información sobre un único permiso.



Solo se puede definir una política de protección de datos por cada tema de Amazon SNS. La política de protección de datos puede incluir una o varias instrucciones de denegación o anonimización, pero solo una instrucción de auditoría.

Propiedades JSON para la política de protección de datos

Una política de protección de datos requiere la siguiente información básica para su identificación:

- Name: el nombre de la política.
- Description (opcional): la descripción de la política.
- Version: la versión del idioma de la política. La versión actual es 2021-06-01.
- Statement: una lista de instrucciones en la que se especifican las acciones de la política de protección de datos.

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

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

Propiedades JSON de una instrucción de política

Una instrucción de política establece el contexto de detección de la operación de protección de datos.

- Sid (opcional): el identificador de la instrucción.
- DataDirection— Entrante (para solicitudes de API de publicación) o saliente (para entregas de notificaciones) con respecto al tema Amazon SNS.
- DataIdentifier— Los datos confidenciales que se deben buscar en el tema de Amazon SNS. Por ejemplo, nombre, dirección o número de teléfono.
- Entidad principal: la entidad principal de IAM que publicó en el tema o la entidad principal de IAM que se suscribió al tema.
- Operation (Operación): la acción de respuesta, ya sea Audit (Auditar), De-identify (Anonimizar) (enmascarar o eliminar) o Deny (Denegar) (bloquear), que el tema de Amazon SNS ejecuta una vez que encuentra información confidencial.

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

Propiedades JSON de una operación de instrucción de política

Una instrucción de política establece una de las siguientes operaciones de protección de datos.

- Audit: emite métricas y registros de búsqueda sin interrumpir la publicación o entrega de mensajes.
- De-identify: enmascara o elimina información confidencial sin interrumpir la publicación de mensajes.
- Deny: bloquea la solicitud de publicación de Amazon SNS o no entrega el mensaje.

¿Cómo determino las entidades principales de IAM para mi política de protección de datos?

La función de protección de datos de mensajes utiliza dos entidades principales de IAM que interactúan con Amazon SNS.

1. Entidad principal de API de publicación (entrante): la entidad principal de IAM autenticada que llama a la API `Publish` de Amazon SNS.
2. Entidad principal de suscripción (saliente): la entidad principal de IAM autenticada que llamó a la API `Subscribe` durante la creación de la suscripción.

`SubscriptionPrincipal` es una propiedad de suscripción a Amazon SNS disponible públicamente que se puede recuperar de la 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"
  }
}
```

Operaciones de la política de protección de datos de Amazon SNS

Estos son ejemplos de políticas de protección de datos que puede utilizar para auditar y denegar datos confidenciales. Para ver un tutorial completo que incluye una aplicación de ejemplo, consulte

la publicación del blog Introducing message data protection for Amazon SNS (Introducción a la protección de datos de mensajes para Amazon SNS).

Operación Audit (Auditar)

La operación de auditoría toma muestras de los mensajes entrantes relacionados con el tema y registra los datos confidenciales encontrados en un AWS destino. La frecuencia de muestreo puede ser un número entero comprendido entre 0 y 99. Esta operación requiere uno de los siguientes tipos de destinos de registro:

1. FindingsDestination— El destino del registro cuando el tema de Amazon SNS encuentra datos confidenciales en la carga útil.
2. NoFindingsDestination— El destino del registro cuando el tema de Amazon SNS no encuentra datos confidenciales en la carga útil.

Puede usar lo siguiente Servicios de AWS en cada uno de los siguientes tipos de destinos de registro:

- Amazon CloudWatch Logs (opcional): `LogGroup` debe estar en la región del tema y el nombre debe empezar por /aws/vendedlogs/.
- Amazon Data Firehose (opcional): el `DeliveryStream` debe estar en la región del tema y tener Direct PUT como origen del flujo de entrega. Para obtener más información, consulte Source, Destination, and Name en la Guía para desarrolladores de Amazon Data Firehose.
- Amazon S3 (opcional): un nombre de bucket de Amazon S3. Se requieren acciones adicionales para utilizar el bucket de Amazon S3 con el cifrado SSE-KMS activado.

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

Permisos necesarios al especificar los destinos de registro

Al especificar los destinos de registro en la política de protección de datos, debe añadir los siguientes permisos a la política de identidad de IAM de la entidad principal de IAM que llama a la API `PutDataProtectionPolicy` de Amazon SNS o la API `CreateTopic` con el parámetro `--data-protection-policy`.

Destino de auditoría	Permiso de IAM
Predeterminado/a	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups

Destino de auditoría	Permiso de IAM
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Se requieren acciones adicionales para utilizar el bucket de Amazon S3 con el cifrado SSE-KMS activado.

```
{
  "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 clave requerida para el uso con SSE-KMS

Si utiliza un bucket de Amazon S3 como destino de registro, puede proteger los datos de su bucket habilitando el cifrado del lado del servidor con claves gestionadas por Amazon S3 (SSE-S3) o el cifrado del lado del servidor con (SSE-KMS). AWS KMS keys Para obtener más información, consulte Protección de datos mediante cifrado del lado del servidor en la Guía del usuario de Amazon S3.

Si elija SSE-S3, no se requiere ninguna configuración adicional. Amazon S3 se encarga de la clave de cifrado.

Si elija SSE-KMS, debe utilizar una clave administrada por el cliente. Debe actualizar la política de clave para la clave administrada por el cliente, de modo que la cuenta de entrega de registros pueda escribir en el bucket de S3. Para obtener más información sobre la política de claves necesaria para su uso con SSE-KMS, consulte el cifrado del lado del servidor de bucket de Amazon S3 en la Guía del usuario de Amazon CloudWatch Logs.

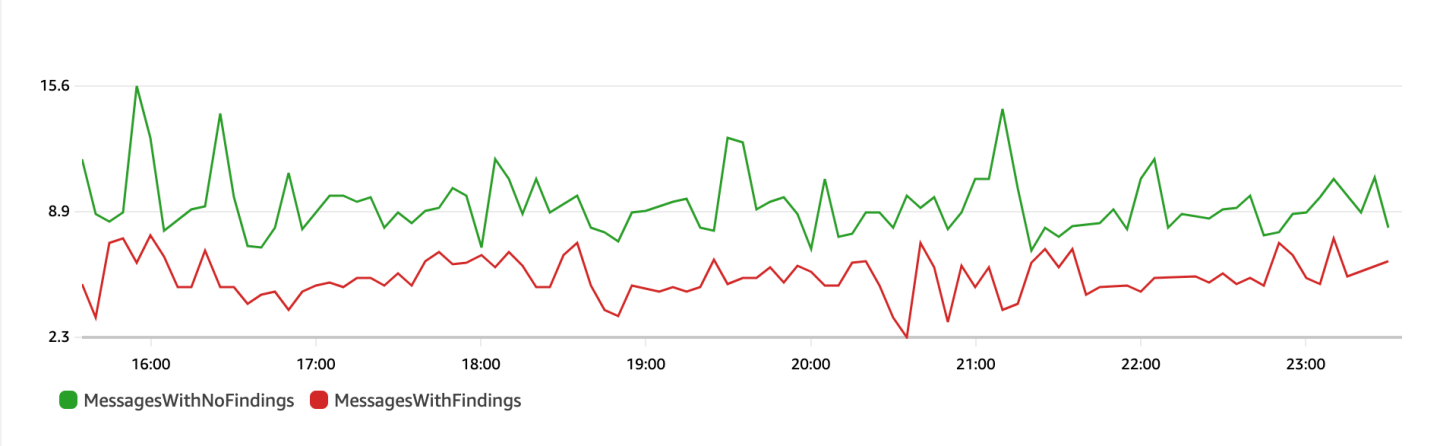
Ejemplo de registro de destino de auditoría

En el siguiente ejemplo, `callerPrincipal` se usa para identificar el origen del contenido confidencial y utilizar `messageID` como referencia para compararla con la respuesta de la 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 la operación Audit (Auditar)

Cuando una operación de auditoría ha especificado la propiedad `FindingsDestination` o la `NoFindingsDestination` propiedad, los propietarios del tema también reciben CloudWatch `MessagesWithFindings` las `MessagesWithNoFindings` métricas.



Operación de anonimización

La operación de Desidentificación enmascara o elimina información confidencial de los mensajes publicados o entregados. Esta operación está disponible para los mensajes entrantes y salientes, y requiere uno de los siguientes tipos de configuraciones:

- MaskConfig— Enmascarar con un carácter compatible de la siguiente tabla. Por ejemplo, ssn: `123-45-6789` se convierte en ssn: `###########`.

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

Carácter de máscara compatible	Nombre
*	Asterisco
A-Z, a-z y 0-9	Alfanumérico

Carácter de máscara compatible	Nombre
	Espacio
!	Signo de exclamación
$	Símbolo del dólar
%	Signo de porcentaje
&	Ampersand
()	Paréntesis
+	Signo más
,	Coma
-	Guion
.	Periodo
/\	Barra, barra diagonal invertida
#	Signo numérico
:	Dos puntos
;	Punto y coma
=, <>	Es igual a, menor o mayor que
@	Arroba
[]	Corchetes
^	Símbolo de intercalación
_	Guion bajo
`	Acento grave

Carácter de máscara compatible	Nombre
|	Barra vertical
~	Símbolo de tilde

- RedactConfig— Redacte eliminando los datos por completo. Por ejemplo, ssn: `123-45-6789` se convierte en ssn: .

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

En un mensaje entrante, la información confidencial se anonimiza después de la operación de auditoría y quien llama a la API de `SNS:Publish` recibe el siguiente error de parámetro no válido cuando todo el mensaje es confidencial.

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

Operación Deny (Denegar)

La operación Deny (Denegar) interrumpe la solicitud de la API `Publish` o la entrega del mensaje si este contiene datos confidenciales. El objeto de la operación Deny (Denegar) está vacío, ya que no requiere ninguna configuración adicional.

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

En un mensaje entrante, el intermediario de la API `SNS:Publish` recibe un error de autorización.

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

En un mensaje saliente, el tema de Amazon SNS no entrega el mensaje a la suscripción. Para realizar un seguimiento de las entregas no autorizadas, active el registro de estado de entrega del tema. A continuación, se muestra un ejemplo de un registro de estado 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"
}
```

Ejemplos de políticas de protección de datos de Amazon SNS

Estos ejemplos son políticas de protección de datos que puede utilizar para auditar y denegar datos confidenciales. Para ver un tutorial completo que incluye una aplicación de ejemplo, consulte la publicación del blog Introducing message data protection for Amazon SNS (Introducción a la protección de datos de mensajes para Amazon SNS).

Ejemplo de política para realizar una auditoría

Las políticas de auditoría le permiten auditar hasta el 99% de los mensajes entrantes y enviar los resultados a Amazon CloudWatch, Amazon Data Firehose y Amazon S3.

Por ejemplo, puede crear una política de auditoría para evaluar si alguno de sus sistemas envía o recibe datos confidenciales sin darse cuenta. Si los resultados de la auditoría muestran que los sistemas envían información de tarjetas de crédito a sistemas que no la requieren, puede implementar una política de protección de datos para bloquear la entrega de los datos.

El siguiente ejemplo audita el 99% de los mensajes que circulan por el tema buscando números de tarjetas de crédito y enviando los resultados a CloudWatch Logs, Firehose y Amazon S3.

Política de protección de datos:

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

Ejemplo de formato de resultados de auditoría:

```
{
    "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"
}
```

Política de ejemplo con instrucción de máscara de desidentificación entrante

En el siguiente ejemplo, se impide que un usuario publique un mensaje en un tema con `CreditCardNumber` al enmascarar la información confidencial en el contenido del mensaje.

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

Ejemplo de resultados entrantes de máscara de desidentificación:

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

// delivered message
```

```
My credit card number is ################
```

Política de ejemplo con instrucción de eliminación de desidentificación entrante

En el siguiente ejemplo se impide que un usuario publique un mensaje en un tema con `CreditCardNumber` al eliminar la información confidencial del contenido del mensaje.

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

Ejemplo de resultados entrantes de anonimización y eliminación:

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

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

Política de ejemplo con instrucción de máscara de desidentificación saliente

En el siguiente ejemplo, se impide que un usuario reciba un mensaje con `CreditCardNumber` al enmascarar la información confidencial en el contenido del mensaje.

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

Ejemplo de resultados salientes de máscara de desidentificación:

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

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

Política de ejemplo con instrucción de eliminación de desidentificación entrante

En el siguiente ejemplo se impide que un usuario publique un mensaje en un tema con `CreditCardNumber` al eliminar la información confidencial del contenido del mensaje.

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

Ejemplo de resultados salientes de desidentificación y eliminación:

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

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

Ejemplo de política con instrucción de denegación de mensaje entrante

En el siguiente ejemplo se impide que un usuario publique un mensaje en un tema con `CreditCardNumber` en el contenido de dicho mensaje. Las cargas útiles denegadas en la respuesta de la API tienen el código de estado»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": {}
      }
    }
  ]
}
```

Ejemplo de política con instrucción de denegación de mensaje saliente

El siguiente ejemplo bloquea una AWS cuenta para que no reciba mensajes que contenganCreditCardNumber.

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

Ejemplo de resultados de denegación salientes registrados en 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"
}
```

Creación de políticas de protección de datos en Amazon SNS

Las políticas de protección de datos le ayudan a proteger los datos publicados en los temas de Amazon SNS auditando, anonimizando (enmascarando o eliminando) y denegando (bloqueando) la información confidencial que se mueve entre aplicaciones o Servicios de AWS. Puede usar la AWS API, AWS CLI AWS CloudFormation, o AWS Management Console para crear políticas de protección de datos en Amazon SNS. Solo se puede definir una política por cada tema de Amazon SNS. Cada política de protección de datos puede incluir una o varias instrucciones de anonimización y denegación, pero solo una instrucción de auditoría.

Temas

- Creación de políticas de protección de datos en Amazon SNS mediante la API
- Creación de políticas de protección de datos en Amazon SNS mediante la CLI
- Creación de políticas de protección de datos en Amazon SNS mediante CloudFormation
- Creación de políticas de protección de datos en Amazon SNS mediante la consola
- Creación de políticas de protección de datos de Amazon SNS para proteger los datos de los mensajes con el SDK.

Creación de políticas de protección de datos en Amazon SNS mediante la API

La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Amazon Simple Notification Service endpoints and quotas (Limitación y cuotas de Amazon Simple Notification Service).

Creación de una política de protección de datos mediante la API

Cree una política de protección de datos de Amazon SNS mediante la AWS API.

Para crear una política de protección de datos junto con un tema de Amazon SNS (API de AWS)

Utilice la propiedad `DataProtectionPolicy` de un tema estándar de Amazon SNS:

- `CreateTopic`

Para recuperar o crear una política de protección de datos para un tema de Amazon SNS existente (API de AWS)

Llame a una de las siguientes operaciones:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Creación de políticas de protección de datos en Amazon SNS mediante la CLI

La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Amazon Simple Notification Service endpoints and quotas (Limitación y cuotas de Amazon Simple Notification Service).

Creación de políticas de protección de datos mediante la AWS CLI

Cree una política de protección de datos de Amazon SNS mediante. AWS Command Line Interface

Para crear una política de protección de datos junto con un tema de Amazon SNS (AWS CLI)

Utilice esta opción para crear una nueva política de protección de datos junto con un tema estándar de Amazon SNS:

- create-topic

Para crear o recuperar una política de protección de datos para un tema de Amazon SNS existente (AWS CLI)

Llame a una de las siguientes operaciones:

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

Creación de políticas de protección de datos en Amazon SNS mediante CloudFormation

La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Amazon Simple Notification Service endpoints and quotas (Limitación y cuotas de Amazon Simple Notification Service).

Creación de políticas de protección de datos (CloudFormation)

Cree una política de protección de datos de Amazon SNS utilizando. AWS CloudFormation

Para crear una política de protección de datos junto con un tema de Amazon SNS (CloudFormation)

Utilice esta opción para crear una nueva política de protección de datos junto con un tema estándar de Amazon SNS:

- AWS::SNS::Topic

Creación de políticas de protección de datos en Amazon SNS mediante la consola

La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Amazon Simple Notification Service endpoints and quotas (Limitación y cuotas de Amazon Simple Notification Service).

Para crear una política de protección de datos junto con un tema de Amazon SNS (consola)

Utilice esta opción para crear una nueva política de protección de datos junto con un tema estándar de Amazon SNS.

1. Inicie sesión en la consola de Amazon SNS.
2. Elija un tema o cree uno nuevo. Para obtener más información acerca de la creación de temas, consulte Creación de un tema de Amazon SNS.
3. En la página Create topic (Crear tema), en la sección Details (Detalles), elija Standard (Estándar).
    a. Ingrese un nombre para el nuevo tema.
    b. (Opcional) Ingrese un nombre para mostrar para el tema.
4. Expanda Data protection policy (Política de protección de datos).

5. Elija un Configuration mode (Modo de configuración):

   - Basic (Básico): se define una política de protección de datos mediante un sencillo menú.

   - Advanced (Avanzado): se define una política de protección de datos personalizada mediante JSON.

6. (Opcional) Para crear el identificador de datos personalizado propio, expanda la sección Configuración del identificador de datos personalizado y haga lo siguiente:

   a. Ingrese un nombre único para el identificador de datos personalizado. Los nombres de los identificadores de datos personalizados admiten caracteres alfanuméricos, de guion bajo (_) y guion (-). Se admiten hasta 128 caracteres. No se puede compartir el mismo nombre que un identificador de datos administrado. Para ver una lista completa de las limitaciones de los identificadores de datos personalizados, consulte Restricciones de identificadores de datos personalizados.

   b. Introduzca una expresión regular (RegEx) para el identificador de datos personalizado. RegExadmite caracteres alfanuméricos, caracteres RegEx reservados y símbolos. RegEx tiene una longitud máxima de 200 caracteres. Si RegEx es demasiado complicado, Amazon SNS fallará en la llamada a la API. Para obtener una lista completa de RegEx limitaciones, consulteRestricciones de identificadores de datos personalizados.

   c. (Opcional) Elija Agregar identificador de datos personalizado para agregar identificadores de datos adicionales según sea necesario. Cada política de protección de datos admite un máximo de 10 identificadores de datos personalizados.

7. Elija las instrucciones que le gustaría añadir a su política de protección de datos. Puede agregar tipos de instrucciones de audit (auditoría), de-identify (anonimización) (enmascarar o eliminar) y deny (denegar) (bloquear) a la misma política de protección de datos.

   a. Add audit statement (Añadir instrucción de auditoría): configure qué datos confidenciales auditar, qué porcentaje de mensajes desea auditar para esos datos y dónde enviar los registros de auditoría.

   > **Note**
   >
   > Solo se permite una instrucción de auditoría por tema o política de protección de datos.

i. Seleccione los identificadores de datos para definir los datos confidenciales que desea auditar.

ii. En Audit sample rate (Frecuencia de muestreo de auditoría), introduzca el porcentaje de mensajes que desea auditar en busca de información confidencial, hasta un máximo del 99 %.

iii. En el caso del destino de la auditoría, seleccione el destino Servicios de AWS al que desea enviar los resultados de la auditoría e introduzca un nombre de destino para cada uno de los Servicio de AWS que utilice. Puede seleccionar uno de los siguientes Amazon Web Services:

- Amazon CloudWatch — CloudWatch Logs es la solución de registro AWS estándar. Con CloudWatch Logs, puede realizar análisis de registros con Logs Insights (consulte los ejemplos aquí) y crear métricas y alarmas. CloudWatch Muchos servicios publican los registros en los registros, lo que facilita la agregación de todos los registros con una sola solución. Para obtener información sobre Amazon CloudWatch, consulta la Guía del CloudWatch usuario de Amazon.
- Amazon Data Firehose: Firehose satisface las demandas de transmisión en tiempo real a Splunk y OpenSearch Amazon Redshift para realizar más análisis de registros. Para obtener más información acerca de Amazon Data Firehose, consulte la Guía del usuario de Amazon Data Firehose.
- Amazon Simple Storage Service: Amazon S3 es un destino de registro económico para fines de archivado. Es posible que deba conservar los registros durante años. En tal caso puede colocar registros en Amazon S3 para ahorrar costes. Para obtener información sobre Amazon Simple Storage Service, consulte la Guía del usuario de Amazon Simple Storage.

b. Add a de-identify statement (Agregar una instrucción de anonimización): configure los datos confidenciales que desea anonimizar en el mensaje, ya sea que desee enmascararlos o eliminarlos y las cuentas para detener la entrega de esos datos.

i. Para Data identifiers (Identificadores de datos), seleccione la información confidencial que desea anonimizar.

ii. En Definir esta declaración de desidentificación, seleccione las AWS cuentas o entidades principales de IAM a las que se aplica esta declaración de desidentificación. Puede aplicarla a todas las AWS cuentas o a cuentas o entidades de IAM específicas

(raíces de AWS cuentas, roles o usuarios) que usen una cuenta o entidad de IAM. IDs ARNs Separe varios IDs o ARNs utilice una coma (,).

Estas son las claves de entidades principales de IAM que se admiten:

- IAM account principals (Entidades principales de cuentas de IAM): por ejemplo, `arn:aws:iam::AWS-account-ID:root`.
- IAM role principals (Entidades principales de roles de IAM): por ejemplo, `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM user principals (Entidades principales de usuarios de IAM): por ejemplo, `arn:aws:iam::AWS-account-ID:user/user-name`.

iii. Para De-identify Option (Opción de anonimización), seleccione cómo desea anonimizar los datos confidenciales. Las siguientes opciones son compatibles:

- Redact (Eliminar): elimina los datos por completo. Por ejemplo, el correo electrónico: `classified@amazon.com` pasa a ser el correo electrónico: .
- Mask (Enmascarar): sustituye los datos por caracteres individuales. Por ejemplo, el correo electrónico: `classified@amazon.com` pasa a ser el correo electrónico: `**********`.

iv. (Opcional) Siga agregando instrucciones de anonimización según sea necesario.

c. Add deny statement (Añadir instrucción de denegación): configure qué datos confidenciales desea evitar que se utilicen en su tema y qué entidades principales evitar que entreguen esos datos.

i. Para data direction (dirección de los datos), elija la dirección de los mensajes de la instrucción de denegación:

- Inbound messages (Mensajes entrantes): aplique esta instrucción de denegación a los mensajes que se envían al tema.
- Outbound messages (Mensajes salientes): aplique esta instrucción de denegación a los mensajes que el tema envía a los puntos de conexión de la suscripción.

ii. Elija los data identifiers (identificadores de datos) para definir la información confidencial que desea denegar.

iii. Elija las entidades principales de IAM a las que se aplica esta instrucción de denegación. Puede aplicarlo a todas las AWS cuentas, a entidades específicas Cuentas de AWS o de IAM (por ejemplo, raíces de cuentas, roles o usuarios) que usen

una cuenta IDs o entidad de IAM. ARNs Separe varios IDs o ARNs utilice una coma (,). Estas son las claves de entidades principales de IAM que se admiten:

- IAM account principals (Entidades principales de cuentas de IAM): por ejemplo, `arn:aws:iam::AWS-account-ID:root`.
- IAM role principals (Entidades principales de roles de IAM): por ejemplo, `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM user principals (Entidades principales de usuarios de IAM): por ejemplo, `arn:aws:iam::AWS-account-ID:user/user-name`.

iv. (Opcional) Siga añadiendo instrucciones de denegación según sea necesario.

Creación de políticas de protección de datos de Amazon SNS para proteger los datos de los mensajes con el SDK.

La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Amazon Simple Notification Service endpoints and quotas (Limitación y cuotas de Amazon Simple Notification Service).

Creación de políticas de protección de datos mediante el SDK de AWS

Cree una política de protección de datos de Amazon SNS mediante el AWS SDK.

Para crear una política de protección de datos junto con un tema de Amazon SNS (SDK de AWS)

Utilice las siguientes opciones para crear una nueva política de protección de datos junto con un tema estándar de 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();
```

Para crear o recuperar una política de protección de datos para un tema de Amazon SNS existente (SDK de AWS)

Utilice las siguientes opciones para crear o recuperar una nueva política de protección de datos junto con un tema estándar de 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();
```

Eliminación de políticas de protección de datos de Amazon SNS

Puede eliminar las políticas de protección de datos de Amazon SNS mediante la AWS API, AWS CLI AWS CloudFormation, o. AWS Management Console

Para obtener información general sobre las políticas de protección de datos de Amazon SNS, consulte Descripción de las políticas de protección de datos de Amazon SNS.

El número y el tamaño de recursos de la política de protección de datos de Amazon SNS en una cuenta de AWS son limitados. Para obtener más información, consulte Limitación de la API de Amazon SNS en la Referencia general de AWS.

Eliminación de políticas de protección de datos mediante la consola

Eliminación de una política de protección de datos administrada mediante la consola

1. Inicie sesión en la consola de Amazon SNS.
2. Elija el tema que contiene la política de protección de datos que desea eliminar.
3. Elija Editar.
4. Expanda la sección Data protection policy (Política de protección de datos).
5. Elija Remove (Eliminar) junto a la política de protección de los datos que desea eliminar.
6. Elija Guardar cambios.

Eliminación de una política de protección de datos mediante una cadena JSON vacía

Puede eliminar una política de protección de datos actualizándola a una cadena JSON vacía.

Eliminar una política de protección de datos mediante el AWS CLI

Puede eliminar una política de protección de datos mediante la AWS CLI.

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

Identificadores de datos de Amazon SNS

Amazon SNS utiliza una combinación de criterios y técnicas, como machine learning y la coincidencia de patrones, para detectar datos confidenciales. Estos criterios y técnicas, que se

denominan en su conjunto identificadores de datos, pueden detectar una lista extensa y creciente de tipos de datos confidenciales en muchos países y regiones. Los identificadores de datos administrados de Amazon SNS ofrecen tipos de datos preconfigurados para proteger los datos financieros, la información médica personal (PHI) y la información de identificación personal (PII). También puede utilizar identificadores de datos personalizados para crear sus propios identificadores de datos adaptados a su caso de uso específico.

Using managed data identifiers in Amazon SNS

¿Qué son los identificadores de datos administrados?

Los identificadores de datos gestionados por Amazon SNS están diseñados para detectar un tipo específico de datos confidenciales, como números de tarjetas de crédito, claves de acceso AWS secretas o números de pasaporte de un país o región determinados. Al crear una política de protección de datos, puede configurar Amazon SNS para que utilice estos identificadores para analizar los mensajes que pasan por el tema y tomar medidas cuando se detecten.

Amazon SNS puede detectar las siguientes categorías de datos confidenciales mediante identificadores de datos administrados:

- Credenciales, como claves privadas o claves de acceso AWS secretas
- Identificadores de dispositivos, como la dirección IP o la dirección MAC
- Información financiera, como números de tarjetas de crédito
- Información médica, para PHI, como números de seguro médico o identificación médica
- Información personal, para PII, como permisos de conducir o números de la Seguridad Social

Dentro de cada categoría, Amazon SNS puede detectar varios tipos de datos confidenciales. En los temas de esta sección, se enumeran y describen cada tipo y los requisitos pertinentes para detectarlos. Para cada tipo, también se indica el identificador único (ID) del identificador de datos administrados que está diseñado para detectar los datos. Al crear una política de protección de datos, puede usar este ID para incluir el identificador de datos administrados para que la función de protección de datos de mensajes lo detecte.

Requisitos de palabras clave

Para detectar ciertos tipos de datos confidenciales, Amazon SNS busca palabras clave en las proximidades de los datos. Si es así para un tipo concreto de datos, en un tema posterior de esta sección se indican los requisitos de palabras clave específicos para esos datos.

Las palabras clave no distinguen entre mayúsculas y minúsculas. Además, si una palabra clave contiene un espacio, Amazon SNS busca automáticamente las variaciones de palabras clave que no contienen el espacio o que contienen un guion bajo (_) o un guion (-) en lugar del espacio. En ciertos casos, Amazon SNS también expande o abrevia una palabra clave para tener en cuenta las variaciones comunes de esa palabra clave.

Identificadores de datos administrados de Amazon SNS para tipos de datos confidenciales

En la siguiente tabla se enumeran y describen los tipos de información de credenciales, dispositivos, financiera, médica y sanitaria protegida (PHI) que Amazon SNS puede detectar mediante identificadores de datos administrados. Estos datos se suman a ciertos tipos de datos que también podrían considerarse información de identificación personal (PII).

Los identificadores de datos dependientes de la región requieren el nombre del identificador con un guion y los códigos de dos letras (ISO 3166-1 alpha-2). Por ejemplo, -US. DriversLicense

Identificador	Categoría	Países/idiomas
BankAccountNumber	Datos financieros	DE, ES, FR, GB, IT
CepCode	Personal	BR
Cnpj	Personal	BR
CpfCode	Personal	BR
DriversLicense	Personal	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	Estado	EE. UU.
ElectoralRollNumber	Personal	GB
HealthInsuranceCardNumber	Estado	UE
HealthInsuranceClaimNumber	Estado	EE. UU.

Identificador	Categoría	Países/idiomas
HealthInsuranceNumber	Estado	FR
HealthcareProcedureCode	Estado	EE. UU.
IndividualTaxIdentification Number	Personal	EE. UU.
InseeCode	Personal	FR
MedicareBeneficiaryNumber	Estado	EE. UU.
NationalDrugCode	Estado	EE. UU.
NationalIdentificationNumber	Personal	DE, ES, IT
NationalInsuranceNumber	Personal	GB
NationalProviderId	Estado	EE. UU.
NhsNumber	Estado	GB
NieNumber	Personal	ES
NifNumber	Personal	ES
PassportNumber	Personal	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	Personal	CA
PersonalHealthNumber	Estado	CA
PhoneNumber	Personal	BR, DE, ES, FR, GB, IT, US
PostalCode	Personal	CA
RgNumber	Personal	BR
SocialInsuranceNumber	Personal	CA
Ssn	Personal	ES, US

Identificador	Categoría	Países/idiomas
TaxId	Personal	DE, ES, FR, GB
ZipCode	Personal	EE. UU.

Identificadores compatibles que son independientes del idioma o la región

Identificador	Categoría
Dirección	Personal
AwsSecretKey	Credenciales
CreditCardExpiration	Datos financieros
CreditCardNumber	Datos financieros
CreditCardSecurityCode	Datos financieros
EmailAddress	Personal
IpAddress	Personal
LatLong	Personal
Nombre	Personal
OpenSshPrivateKey	Credenciales
PgpPrivateKey	Credenciales
PkcsPrivateKey	Credenciales
PuttyPrivateKey	Credenciales
VehicleIdentificationNumber	Personal

Tipos de datos confidenciales de Amazon SNS: credenciales

En la siguiente tabla se enumeran y describen los tipos de credenciales que Amazon SNS puede detectar mediante identificadores de datos administrados.

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Países y regiones
AWS clave de acceso secreta	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Cualquiera
Clave privada de OpenSSH	OpenSshPrivateKey	No	Cualquiera
Clave privada de PGP	PgpPrivateKey	No	Cualquiera
Clave privada del estándar de criptografía de clave pública (PKCS)	PkcsPrivateKey	No	Cualquiera
Clave privada PuTTY	PuttyPrivateKey	No	Cualquiera

Identificador de datos ARNs para los tipos de datos de credenciales

A continuación, se enumeran los nombres de recursos de Amazon (ARNs) para los identificadores de datos que puede añadir a sus políticas de protección de datos.

Identificador de datos de credenciales ARNs
arn:aws:data protection: :aws:data-identifier/ AwsSecretKey
arn:aws:protección de datos: :aws:data-identifier/ OpenSshPrivateKey
arn:aws:protección de datos: :aws:data-identifier/ PgpPrivateKey

Identificador de datos de credenciales ARNs
arn:aws:protección de datos: :aws:data-identifier/ PkcsPrivateKey
arn:aws:protección de datos: :aws:data-identifier/ PuttyPrivateKey

Tipos de datos confidenciales de Amazon SNS: dispositivos

En la siguiente tabla se enumeran y describen los tipos de identificadores de dispositivos que Amazon SNS puede detectar mediante identificadores de datos administrados.

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Países y regiones
Dirección IP	IpAddress	No	Cualquiera

Identificador de datos para los tipos de ARNs datos del dispositivo

A continuación, se enumeran los nombres de recursos de Amazon (ARNs) para los identificadores de datos que puede añadir a sus políticas de protección de datos.

ARN de identificador de datos de dispositivos
arn:aws:dataprotection: :aws:data-identifier/ IpAddress

Tipos de datos confidenciales de Amazon SNS: financieros

Obtenga información se enumeran y describen los tipos de información financiera que Amazon SNS puede detectar mediante identificadores de datos administrados.

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de cuenta bancaria	BankAccountNumber	Sí, consulte Palabras clave	Esto incluye: números	Alemania, España, Francia,

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
	BankAccountNumber-US	para números de cuentas bancarias.	de cuentas bancarias internacionales (IBANs) que constan de hasta 34 caracteres alfanuméricos, incluidos elementos como el código de país.	Italia, Reino Unido
Fecha de caducidad de la tarjeta	CreditCardExpiration	exp d, exp m, exp y, expiration, expiry	–	Cualquiera
Datos de banda magnética de tarjetas de crédito	CreditCardMagneticStripe	Sí, por ejemplo: card data, iso7813, mag, magstripe, stripe, swipe	Esto incluye las pistas 1 y 2.	Cualquiera

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de tarjetas 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	La detección requiere que los datos sean una secuencia de 13 a 19 dígitos que se ajuste a la fórmula de cheques de Luhn y utilice un prefijo de número de tarjeta estándar para cualquiera de los siguientes tipos de tarjetas de crédito: American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card Bureau (JCB) UnionPay, Mastercard y Visa (enlace de superíndice a continuación).	Cualquiera

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Código de verificación de tarjeta de crédito	CreditCardSecurityCode	card id, card identification code, card identification number, card security code, card validation code, card validation number, card verification data, card verification value, cvc, cvc2, cvv, cvv2, elo verification code	–	Cualquiera

1. Amazon SNS no informa de las siguientes secuencias, que los emisores de tarjetas de crédito se reservan para las pruebas públicas:

   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 y 76009244561.

Palabras clave para números de cuentas bancarias

Utilice las siguientes palabras clave para detectar números de cuentas bancarias internacionales (IBANs) que consten de un máximo de 34 caracteres alfanuméricos, incluidos elementos como el código de país.

País o región	Palabras clave			
Francia	account code, account number, accountno #, accountnumber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
Alemania	account code, account number, accountno #, accountnumber#, bankleitzahl, bban, customer account id, customer account number,			

País o región	Palabras clave			
	customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
Italia	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 o región	Palabras clave			
España	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 o región	Palabras clave			
EE. UU.	bank account, bank acct, checking account, checking acct, deposit account, deposit acct, savings account, savings acct, chequing account, chequing acct			

Identificador de datos ARNs para los tipos de datos financieros

A continuación, se enumeran los nombres de recursos de Amazon (ARNs) para los identificadores de datos que puede añadir a sus políticas de protección de datos.

Identificador de datos financieros ARNs
arn:aws:data protection: :aws:data-identifier/ -DE BankAccountNumber
arn:aws:protección de datos: :aws:data-identifier/ BankAccountNumber -ES
arn:aws:protección de datos: :aws:data-identifier/ -FR BankAccountNumber
arn:aws:protección de datos: :aws:data-identifier/ -GB BankAccountNumber
arn:aws:protección de datos: :aws:data-identifier/ -IT BankAccountNumber
arn:aws:protección de datos: :aws:data-identifier/ -US BankAccountNumber
arn:aws:protección de datos: :aws:data-identifier/ CreditCardExpiration
arn:aws:protección de datos: :aws:data-identifier/ CreditCardNumber

Identificador de datos financieros ARNs
arn:aws:protección de datos: :aws:data-identifier/ CreditCardSecurityCode

Tipos de datos confidenciales de Amazon SNS: información médica protegida (PHI)

En la siguiente tabla se enumeran y describen los tipos de información médica protegida (PHI) que Amazon SNS puede detectar mediante identificadores de datos administrados.

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Países y regiones
Número de registro de la Administración para el Control de Drogas (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	EE. UU.
Número de tarjeta de seguro médico (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical	UE

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Países y regiones
		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	
Número de reclamación del seguro médico (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	EE. UU.
Número de seguro médico o identificación médica	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Países y regiones
Código del sistema de codificación de procedimientos comunes de atención médica (HCPCS)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	EE. UU.
Número de beneficiario de Medicare (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	EE. UU.
Código nacional de medicamento (NDC)	NationalDrugCode	national drug code, ndc	EE. UU.
Identificador nacional de proveedores (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	EE. UU.
Número del Servicio Nacional de Salud (NHS)	NhsNumber	national health service, NHS	GB
Número médico personal (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Palabras clave para números de seguro médico e identificación médica

Para detectar distintos tipos de números de seguro médico e identificación médica, Amazon SNS requiere que una palabra clave esté cerca de los números. Esto incluye números de tarjetas de seguro médico europeas (UE, Finlandia), números de seguro médico (Francia), identificadores de beneficiarios de Medicare (EE. UU.), números de seguro nacional (Reino Unido), números del NHS (Reino Unido) y números médicos personales (Canadá).

En la siguiente tabla se enumeran las palabras clave que Amazon SNS reconoce para países y regiones específicos.

País o región	Palabras clave
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
Finlandia	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 o región	Palabras clave
	t, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
Francia	carte d'assuré social, carte vitale, insurance card
Reino Unido	national health service, NHS
EE. UU.	mbi, medicare beneficiary

Identificador de datos para los tipos de datos de información de salud protegida (PHI) ARNs

A continuación se muestra el identificador de datos Amazon Resource Names (ARNs) que se puede utilizar en las políticas de protección de datos de PHI.

Identificador de datos PHI ARNs
arn:aws:data protection: :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:data protection: :aws:data-identifier/ -US HealthcareProcedureCode
arn:aws:protección de datos: :aws:data-identifier/ -EU HealthInsuranceCardNumber
arn:aws:data protection: :aws:data-identifier/ -US HealthInsuranceClaimNumber
arn:aws:protección de datos: :aws:data-identifier/ -FR HealthInsuranceNumber
arn:aws:data protection: :aws:data-identifier/ -US MedicareBeneficiaryNumber
arn:aws:data protection: :aws:data-identifier/ -US NationalDrugCode
arn:aws:protección de datos: :aws:data-identifier/ -GB NationalInsuranceNumber
arn:aws:data protection: :aws:data-identifier/ -US NationalProviderId
arn:aws:protección de datos: :aws:data-identifier/ -GB NhsNumber
arn:aws:protección de datos: :aws:data-identifier/ -CA PersonalHealthNumber

Tipos de datos confidenciales de Amazon SNS: información de identificación personal (PII)

En la siguiente tabla se enumeran y describen los tipos de información de identificación personal (PII) que Amazon SNS puede detectar mediante identificadores de datos administrados.

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Fecha de nacimiento	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	La mayoría de los formatos de fecha están admitidos, como todos los dígitos y combinaciones de dígitos y nombres de meses. Los componentes de fecha se pueden separar mediante espacios, barras (/) o guiones (-).	Cualquiera
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	–	Brasil

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
		pessoa juridica, cnpj		
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	Brasil

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de identificación del permiso de conducir	DriversLicense	Sí, consulte Palabras clave de números de identificación del permiso de conducir.	–	Alemania, Australia, Austria, Bélgica, Bulgaria, Canadá, Chipre, Croacia, Dinamarca, EE. UU., Eslovaquia, Eslovenia, España, Estonia, Finlandia, Francia, Grecia, Hungría, Irlanda, Italia, Letonia, Lituania, Luxemburgo, Malta, Países Bajos, Polonia, Portugal, Rumanía, Reino Unido, República Checa, Suecia

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de registro electoral	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	Reino Unido
Identificación individual del contribuyente	IndividualTaxIdentificationNumber	Sí, consulte Palabras clave para números de identificación y referencia del contribuyente.	–	EE. UU.
Instituto Nacional de Estadística y Estudios Económicos (INSEE)	InseeCode	Sí, consulte Palabras clave para números de documentos nacionales de identificación.	–	Francia

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de identificación nacional	NationalIdentificationNumber	Sí, consulte Palabras clave para números de documentos nacionales de identificación.	Esto incluye los identificadores del documento nacional de identidad (DNI) (España), los códigos del Codice Fiscale (Italia) y los números del documento nacional de identidad (Alemania).	Alemania, España, Italia
Número de 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	Sí, consulte Palabras clave para números de identificación y referencia del contribuyente.	–	España

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de identificación fiscal (NIF)	NifNumber	Sí, consulte Palabras clave para números de identificación y referencia del contribuyente.	–	España
Número de pasaporte	PassportNumber	Sí, consulte Palabras clave para números de pasaporte.	–	Alemania, Canadá, España, Estados Unidos, Francia, Italia, Reino Unido

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de residencia 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 detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de teléfono	PhoneNumber	Brasil: las palabras clave también incluyen cel, celular, fone, móvel, número residencial, numero residencial, telefone Otras: cell, contact, fax, fax number, mobile, phone, phone number, tel, telephone, telephone number	Esto incluye los números gratuitos de Estados Unidos y números de fax. Si una palabra clave está cerca de los datos, no es necesario que el número incluya un código de país. Si una palabra clave no está cerca de los datos, el número debe incluir un código de país.	Alemania, Brasil, Canadá, España, Estados Unidos, Francia, Italia, Reino Unido
Postal Code (Código postal)	PostalCode	No	–	Canadá
Registro Geral (RG)	RgNumber	Sí, consulte Palabras clave para números de documentos nacionales de identificación.	–	Brasil

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Número de Seguro Social (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Canadá
Número de la Seguridad Social (SSN)	Ssn	España: número de la seguridad social, social security no., social security no. número de la seguridad social, social security number, socialsecurityno#, ssn, ssn# EE. UU.: social security, ss#, ssn	–	España, Estados Unidos
Número de identificación o referencia del contribuyente	TaxId	Sí, consulte Palabras clave para números de identificación y referencia del contribuyente.	Esto incluye TIN (Francia), Steueridentifikationsnummer (Alemania), CIF (España) y TRN, UTR (Reino Unido).	Alemania, España, Francia, Reino Unido

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Código postal de Estados Unidos	ZipCode	zip code, zip+4	–	EE. UU.
Dirección postal	Address	No	Aunque no se requiere una palabra clave, para la detección es necesario que la dirección incluya el nombre de una ciudad o lugar y un código postal.	Alemania, Australia, Canadá, España, Estados Unidos, Francia, Italia, Reino Unido
Dirección de correo electrónico	EmailAddress	correo electrónico, dirección de correo electrónico, correo electrónico, dirección de correo electrónico	–	Cualquiera

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Coordenadas del sistema de posicionamiento global (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS puede detectar las coordenadas GPS si las coordenadas de latitud y longitud se almacenan como un par y están en formato de grados decimales (DD), por ejemplo, 41.948614 ,-87.655311. No es compatible con coordenadas en formato de grados y minutos decimales (DDM), por ejemplo 41°56.9168'N 87°39.3187'O, o en formato de grados, minutos y segundos (DMS), por ejemplo 41°56'55.0104"N 87°39'19.1196"O.	Cualquiera

Tipo de detección	Identificador de datos administrados	Palabra clave necesaria	Información adicional	Países y regiones
Nombre completo	Name	No	Amazon SNS solo puede detectar nombres completos. La compatibilidad se limita a los conjuntos de caracteres latinos.	Cualquiera
Número de identificación de vehículo (VIN)	VehicleIdentificationNumber	Fahrgestellnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	Amazon SNS puede detectar si VINs se compone de una secuencia de 17 caracteres y cumple con las normas ISO 3779 y 3780. Estos estándares fueron diseñados para su uso en todo el mundo.	Cualquiera

Palabras clave de números de identificación del permiso de conducir

Para detectar distintos tipos de números de identificación de permisos de conducir, Amazon SNS requiere que una palabra clave esté cerca de los números. En la siguiente tabla se enumeran las palabras clave que Amazon SNS reconoce para países y regiones específicos.

País o región	Palabras clave
Australia	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
Austria	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Bélgica	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgaria	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
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 o región	Palabras clave
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
Croacia	vozačka dozvola
Chipre	άδεια οδήγησης
República Checa	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
Estonia	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finlandia	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
Francia	permis de conduire
Alemania	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Grecia	δεια οδήγησης, adeia odigisis
Hungría	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Irlanda	ceadúnas tiomána
Italia	patente di guida, patente di guida numero, patente guida, patente guida numero

País o región	Palabras clave
Letonia	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Lituania	vairuotojo pažymėjimas
Luxemburgo	fahrerlaubnis, führerschäin
Malta	liċenzja tas-sewqan
Países Bajos	permis de conduire, rijbewijs, rijbewijsnummer
Polonia	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
Rumanía	numărul permisului de conducere, permis de conducere
Eslovaquia	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Eslovenia	vozniško dovoljenje

País o región	Palabras clave
España	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
Suecia	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
EE. UU.	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

Palabras clave para números de documentos nacionales de identificación

Para detectar distintos tipos de números de documentos nacionales de identificación, Amazon SNS requiere que una palabra clave esté cerca de los números. Esto incluye los identificadores del

documento nacional de identidad (DNI) (España), los códigos del Instituto Nacional de Estadística y Estudios Económicos (INSEE) de Francia, los números del documento nacional de identidad alemán y los números del Registro Geral (RG) (Brasil).

En la siguiente tabla se enumeran las palabras clave que Amazon SNS reconoce para países y regiones específicos.

País o región	Palabras clave
Brasil	registro geral, rg
Francia	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#
Alemania	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italia	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
España	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#

Palabras clave para números de pasaporte

Para detectar distintos tipos de números de pasaporte, Amazon SNS requiere que una palabra clave esté cerca de los números. En la siguiente tabla se enumeran las palabras clave que Amazon SNS reconoce para países y regiones específicos.

País o región	Palabras clave
Canadá	passeport, passeport#, passport, passport#, passportno, passportno#
Francia	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
Alemania	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Italia	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
España	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
EE. UU.	passport, travel document

Palabras clave para números de identificación y referencia del contribuyente

Para detectar distintos tipos de números de identificación y referencia del contribuyente, Amazon SNS requiere que haya una palabra clave cerca de los números. En la siguiente tabla se enumeran las palabras clave que Amazon SNS reconoce para países y regiones específicos.

País o región	Palabras clave
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
Francia	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Alemania	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
España	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
EE. UU.	número de identificación tributaria individual (ITIN)

Identificador de datos ARNs para la información de identificación personal (PII)

En la siguiente tabla se enumeran los nombres de recursos de Amazon (ARNs) para los identificadores de datos que puede añadir a sus políticas de protección de datos.

Identificador de datos PII ARNs
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:data protection: :aws:data-identifier/ -BR CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:protección de datos: :aws:data-identifier/ -BR CpfCode
arn:aws:protección de datos: :aws:data-identifier/ DateOfBirth
arn:aws:protección de datos: :aws:data-identifier/ DriversLicense -AT
arn:aws:protección de datos: :aws:data-identifier/ -AU DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -BE DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -BG DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -CA DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ DriversLicense -CY
arn:aws:protección de datos: :aws:data-identifier/ -CZ DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ DriversLicense -DE
arn:aws:protección de datos: :aws:data-identifier/ -DK DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -EE DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ DriversLicense -ES
arn:aws:protección de datos: :aws:data-identifier/ -FI DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -FR DriversLicense

Identificador de datos PII ARNs
arn:aws:protección de datos: :aws:data-identifier/ -GB DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -GR DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -HR DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -HU DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -IE DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -IT DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -LT DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -LU DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -LV DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -MT DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -NL DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -PL DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -PT DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -RO DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ DriversLicense -SE
arn:aws:protección de datos: :aws:data-identifier/ -SI DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -SK DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -US DriversLicense
arn:aws:protección de datos: :aws:data-identifier/ -GB ElectoralRollNumber
arn:aws:protección de datos: :aws:data-identifier/ EmailAddress

Identificador de datos PII ARNs
arn:aws:protección de datos: :aws:data-identifier/ -US IndividualTaxIdentificationNumber
arn:aws:protección de datos: :aws:data-identifier/ -FR InseeCode
arn:aws:protección de datos: :aws:data-identifier/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:protección de datos: :aws:data-identifier/ NationalIdentificationNumber -DE
arn:aws:protección de datos: :aws:data-identifier/ NationalIdentificationNumber -ES
arn:aws:protección de datos: :aws:data-identifier/ -IT NationalIdentificationNumber
arn:aws:protección de datos: :aws:data-identifier/ NieNumber -ES
arn:aws:protección de datos: :aws:data-identifier/ NifNumber -ES
arn:aws:protección de datos: :aws:data-identifier/ -CA PassportNumber
arn:aws:protección de datos: :aws:data-identifier/ PassportNumber -DE
arn:aws:protección de datos: :aws:data-identifier/ PassportNumber -ES
arn:aws:protección de datos: :aws:data-identifier/ -FR PassportNumber
arn:aws:protección de datos: :aws:data-identifier/ -GB PassportNumber
arn:aws:protección de datos: :aws:data-identifier/ -IT PassportNumber
arn:aws:protección de datos: :aws:data-identifier/ -US PassportNumber
arn:aws:protección de datos: :aws:data-identifier/ -CA PermanentResidenceNumber
arn:aws:protección de datos: :aws:data-identifier/ -BR PhoneNumber
arn:aws:protección de datos: :aws:data-identifier/ PhoneNumber -DE
arn:aws:protección de datos: :aws:data-identifier/ PhoneNumber -ES

Identificador de datos PII ARNs
arn:aws:protección de datos: :aws:data-identifier/ -FR PhoneNumber
arn:aws:protección de datos: :aws:data-identifier/ -GB PhoneNumber
arn:aws:protección de datos: :aws:data-identifier/ -IT PhoneNumber
arn:aws:protección de datos: :aws:data-identifier/ -US PhoneNumber
arn:aws:protección de datos: :aws:data-identifier/ -CA PostalCode
arn:aws:protección de datos: :aws:data-identifier/ -BR RgNumber
arn:aws:protección de datos: :aws:data-identifier/ -CA SocialInsuranceNumber
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:protección de datos: :aws:data-identifier/ TaxId -DE
arn:aws:protección de datos: :aws:data-identifier/ TaxId -ES
arn:aws:protección de datos: :aws:data-identifier/ -FR TaxId
arn:aws:protección de datos: :aws:data-identifier/ -GB TaxId
arn:aws:protección de datos: :aws:data-identifier/ VehicleIdentificationNumber
arn:aws:protección de datos: :aws:data-identifier/ -US ZipCode

Uso de identificadores de datos personalizados en Amazon SNS

Los identificadores de datos personalizados (CDIs) le permiten definir sus propias expresiones regulares personalizadas que se pueden utilizar en su política de protección de datos. Con los identificadores de datos personalizados, puede centrarse en los casos de uso de la información de identificación personal (PII) específica de la empresa que los identificadores de datos administrados no pueden proporcionar. Por ejemplo, puede usar un identificador de datos personalizado para

buscar un empleado específico de la empresa. IDs Los identificadores de datos personalizados se pueden utilizar junto con los identificadores de datos administrados.

¿Qué son los identificadores de datos personalizados?

Los identificadores de datos personalizados (CDIs) le permiten definir sus propias expresiones regulares personalizadas que se pueden utilizar en su política de protección de datos. Con los identificadores de datos personalizados, puede centrarse en los casos de uso de la información de identificación personal (PII) específica de la empresa que los identificadores de datos administrados no pueden proporcionar. Por ejemplo, puede usar un identificador de datos personalizado para buscar un empleado específico de la empresa. IDs Los identificadores de datos personalizados se pueden utilizar junto con los identificadores de datos administrados.

Uso de identificadores de datos personalizados en la política de protección de datos

La siguiente política de protección de datos indica al tema Amazon SNS que detecte las cargas útiles que transporten a un IDs empleado específico de la empresa y, a continuación, IDs las oculte con el símbolo hash (#).

1. Cree un bloque de `Configuration` en la política de protección de datos.
2. Ingrese un `Name` para el identificador de datos personalizado. Por ejemplo, **`EmployeeId`**.
3. Ingrese un `Regex` para el identificador de datos personalizado. Por ejemplo, **`EID-\d{9}-US`**.
4. Consulte el siguiente identificador de datos personalizado en una instrucción de la 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) Siga agregando identificadores de datos personalizados adicionales al bloque de `Configuration` según sea necesario. Las políticas de protección de datos admiten actualmente un máximo de 10 identificadores de datos personalizados.

Restricciones de identificadores de datos personalizados

Los identificadores de datos personalizados de Amazon SNS tienen las siguientes limitaciones:

- Cada política de protección de datos admite un máximo de 10 identificadores de datos personalizados.
- Los nombres de identificadores de datos personalizados tienen una longitud máxima de 128 caracteres. Se admiten los siguientes caracteres:
  - Alfanumérico: (a-zA-Z0-9)
  - Símbolos: (“_” | “-”)
- RegEx tiene una longitud máxima de 200 caracteres. Se admiten los siguientes caracteres:
  - Alfanumérico: (a-zA-Z0-9)
  - Símbolos: (“_” | “#” | “=” | “@” | / | “;” | “,” | “-” |)
  - RegEx caracteres reservados: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '\\' | '*' | '+' | '.')
- Los identificadores de datos personalizados no pueden compartir el mismo nombre que un identificador de datos administrado.
- Los identificadores de datos personalizados se deben especificar en todas las políticas de protección de datos de cada tema de Amazon SNS.

Entrega de mensajes de Amazon SNS

En este tema se describe cómo Amazon SNS gestiona la entrega de mensajes en varios escenarios. Obtendrá información sobre la entrega de mensajes sin procesar, en la que Amazon SNS entrega los mensajes en su formato original sin modificar el formato en el punto de conexión. También descubrirá cómo enviar mensajes desde un tema de Amazon SNS a una cola de Amazon SQS en Cuenta de AWS otro lugar, lo que le proporcionará información sobre la mensajería entre cuentas.

En este tema se proporciona información sobre la entrega de mensajes de Amazon SNS a una cola de Amazon SQS o a una función de Lambda Regiones de AWS en diferentes aspectos, cómo funciona la entrega entre regiones y las consideraciones implicadas.

Además, aprenderá a supervisar e interpretar el estado de entrega de los mensajes, que proporciona información básica sobre si los mensajes se consiguieron entregar o si se produjeron problemas. En los casos en los que se produzca un error en la entrega de mensajes, conocerá el proceso de reintento de entrega de mensajes, incluida la forma en que Amazon SNS intenta volver a entregar los mensajes automáticamente para garantizar que lleguen a sus destinos previstos. En este tema también se analiza el uso de colas de mensajes fallidos para capturar los mensajes que no se pudieron entregar tras varios intentos, lo que le permite analizar y solucionar estos errores de forma eficaz.

Entrega de mensajes sin procesar de Amazon SNS

Para evitar que los puntos de conexión de Amazon Data Firehose, Amazon SQS y HTTP/S procesen el formato JSON de los mensajes, Amazon SNS permite la entrega de mensajes sin procesar:

- Cuando se habilita la entrega de mensajes sin procesar para los puntos de conexión de Amazon Data Firehose o Amazon SQS, los metadatos de Amazon SNS se eliminan del mensaje publicado y el mensaje se envía tal cual.
- Cuando habilita la entrega de mensajes sin formato para los puntos de enlace HTTP/S, el encabezado HTTP `x-amz-sns-rawdelivery` con su valor establecido en `true` se agrega al mensaje, lo que indica que el mensaje se ha publicado sin formato JSON.
- Cuando habilita la entrega de mensajes sin procesar para los puntos de conexión HTTP/S, se entregan el cuerpo del mensaje, la IP del cliente y los encabezados necesarios. Cuando especifica atributos de mensaje, no se enviará.
- Cuando habilita la entrega de mensajes sin procesar para los puntos de conexión de Firehose, se entrega el cuerpo del mensaje. Cuando especifica atributos de mensaje, no se enviará.

Para habilitar la entrega de mensajes sin procesar mediante un AWS SDK, debe usar la acción de la `SetSubscriptionAttribute` API y establecer el valor del `RawMessageDelivery` atributo en. `true`

Habilitación de la entrega de mensajes sin procesar mediante la AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. En la página Temas, elija un tema suscrito a un punto de conexión de Firehose, Amazon SQS o HTTP/S.
4. En la ***MyTopic*** página, en la sección Suscripción, elija una suscripción y elija Editar.
5. En la *EXAMPLE1-23bc-4567-d890-ef12g3hij456* página de edición, en la sección Detalles, selecciona Habilitar la entrega de mensajes sin procesar.
6. Elija Guardar cambios.

Ejemplos de formato de mensajes

En los siguientes ejemplos, el mismo mensaje se envía dos veces a la misma cola de Amazon SQS. La única diferencia es que la entrega de mensajes sin procesar está desactivada para el primer mensaje y habilitada para el segundo.

- La entrega de mensajes sin procesar está desactivada

```
{
  "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"
}
```

- La entrega de mensajes sin procesar está habilitada

```
This is a test message.
```

Atributos de los mensajes y entrega de mensajes sin procesar para las suscripciones de Amazon SQS

Amazon SNS admite los atributos de entrega de mensajes, con los que se pueden ofrecer elementos de metadatos estructurados (como marcas de tiempo, datos geoespaciales, firmas e identificadores) relacionados con el mensaje. En el caso de las suscripciones de Amazon SQS con la característica Entrega de mensajes sin procesar habilitada, se puede enviar un máximo de 10 mensajes. Para enviar más de 10 atributos de mensaje, debe deshabilitar la entrega de mensajes sin procesar. Sin embargo, Amazon SNS descarta los mensajes con más de 10 atributos de mensaje dirigidos a las suscripciones de Amazon SQS con la entrega de mensajes sin procesar habilitada y los trata como errores del cliente.

Envío de mensajes de Amazon SNS a una cola de Amazon SQS de otra cuenta

Este documento describe cómo publicar una notificación en un tema de Amazon SNS con una o varias suscripciones a colas de Amazon SQS de otra cuenta. Configure el tema y las colas igual que lo haría si estuviesen en la misma cuenta (consulte Distribución ramificada de notificaciones de Amazon SNS a colas de Amazon SQS para su procesamiento asíncrono). La principal diferencia radica en cómo controla la confirmación de suscripción y eso depende de cómo suscribe la cola al tema.

Es recomendable seguir los pasos a los que se hace referencia en la sección El propietario de la cola crea la suscripción cuando sea posible, porque la confirmación es automática cuando el propietario de la cola crea la suscripción.

> **Note**
>
> Si la cola de Amazon SQS tiene un gran volumen de mensajes, recomendamos que el propietario de la cola cree la suscripción.

El propietario de la cola crea la suscripción

La cuenta que creó la cola de Amazon SQS es el propietario de la cola. Cuando el propietario de la cola crea una suscripción, esta no necesita una confirmación. La cola comienza a recibir notificaciones desde el tema tan pronto como la acción `Subscribe` se completa. Para dejar que el propietario de la cola se suscriba al tema del propietario del tema, el propietario del tema debe conceder un permiso de cuenta al propietario de la cola para llamar a la acción `Subscribe` en el tema.

Paso 1: Establecer la política del tema mediante la AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. Seleccione un tema y, a continuación, seleccione Edit (Editar).
4. En la *`MyTopic`* página de edición, expanda la sección Política de acceso.
5. Escriba la siguiente política:

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

```
}
```

Esta política concede permiso a la cuenta `111122223333` para llamar a `sns:Subscribe` en `MyTopic` en la cuenta `123456789012`.

Un usuario con las credenciales de la cuenta `111122223333` puede suscribirse a `MyTopic`. Este permiso permite al ID de la cuenta delegar el permiso a su rol o usuario de IAM. Solo la cuenta raíz o los usuarios administradores podrán llamar a `sns:Subscribe`. El usuario o rol de IAM también debe tener `sns:subscribe` para permitir que su cola se suscriba.

6. Elija Guardar cambios.

Un usuario con las credenciales de la cuenta `111122223333` puede suscribirse MyTopic.

Paso 2: Para añadir una suscripción a Amazon SQS Queue a un tema de otro mediante el Cuenta de AWSAWS Management Console

Antes de empezar, asegúrese de que dispone del tema y ARNs de la cola y de que ha dado permiso al tema para enviar mensajes a la cola.

1. Inicie sesión en la consola de Amazon SQS.
2. En el panel de navegación, elija Queues (Colas).
3. En la lista de colas, elija la cola para suscribirse al tema de Amazon SNS.
4. Elija Subscribe to Amazon SNS topic (Suscribirse al tema de Amazon SNS).
5. Desde Specify an Amazon SNS topic available for this queue menu (Especificar un tema de Amazon SNS disponible para este menú de cola), elija el Amazon SNS topic (Tema de Amazon SNS) para la cola.
6. Elija Enter Amazon SNS topic ARN (Ingresar ARN de tema de Amazon SNS) y, a continuación, ingrese el Amazon Resource Name (ARN) (Nombre de recurso de Amazon [ARN]) del tema.
7. Seleccione Guardar.

> **Note**
>
> - Para poder comunicarse con el servicio, la cola debe tener permisos para Amazon SNS.
> - Puesto que usted es el propietario de la cola, no tiene que confirmar la suscripción.

Un usuario que no es el propietario de la cola crea una suscripción

Cualquier usuario que crea una suscripción y no es el propietario de la cola tiene que confirmar la suscripción.

Cuando utiliza la acción `Subscribe`, Amazon SNS envía una confirmación de suscripción a la cola. La suscripción aparece en la consola de Amazon SNS, con su ID de suscripción establecido en Confirmación pendiente.

Para confirmar la suscripción, un usuario con permiso para leer los mensajes de la cola debe recuperar la URL de confirmación de la suscripción y el propietario de la suscripción debe confirmar la suscripción mediante la URL de confirmación de la suscripción. Hasta que no se confirme la suscripción, no se enviarán a la cola las notificaciones publicadas en el tema. Para confirmar la suscripción, puede utilizar la consola de Amazon SQS o la acción `ReceiveMessage`.

> **Note**
>
> Antes de suscribir un punto de enlace al tema, asegúrese de que la cola pueda recibir mensajes desde el tema mediante la configuración del permiso `sqs:SendMessage` para la cola. Para obtener más información, consulte Paso 2: conceder permiso al tema de Amazon SNS y enviar mensajes a la cola de Amazon SQS.

Paso 1: Para añadir una suscripción a Amazon SQS Queue a un tema de otro mediante el Cuenta de AWSAWS Management Console

Antes de empezar, asegúrese de tener el tema y la cola y de haber dado permiso al tema ARNs para enviar mensajes a la cola.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, seleccione Subscriptions (Suscripciones).
3. En la página Subscriptions (Suscripciones), elija Create subscription (Crear suscripción).
4. En la página Crear suscripción, en la sección Detalles, haga lo siguiente:
   a. En Topic ARN (ARN del tema), introduzca el ARN del tema.
   b. En Protocolo, elija Amazon SQS.
   c. En Endpoint (Punto de enlace), introduzca el ARN de la cola.
   d. Elija Crear una suscripción.

> **Note**
>
> - Para poder comunicarse con el servicio, la cola debe tener permisos para Amazon SNS.

A continuación se muestra una declaración de política de ejemplo que permite al tema de Amazon SNS enviar un mensaje a la cola de 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"
      }
   }
}
```

Paso 2: Para confirmar una suscripción mediante el AWS Management Console

1. Inicie sesión en la consola de Amazon SQS.
2. Seleccione la cola que tenga una suscripción pendiente con el tema.
3. Elija Send and receive messages (Enviar y recibir mensajes) y, a continuación, elija Poll for messages (Sondear en busca de mensajes).

   Se recibe un mensaje con la confirmación de la suscripción en la cola.
4. En la columna Body (Cuerpo) , realice las siguientes acciones:

   a. Seleccione More Details (Más detalles).

   b. En el cuadro de diálogo Message Details (Detalles de mensajes), busque y anote el valor de SubscribeURL. Este es el enlace de suscripción (el ejemplo se muestra a continuación). Para obtener más información sobre la validación de tokens de API, consulte `ConfirmSubscription` en la Referencia de la API de Amazon SNS.

```
https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

c. Anote los valores del enlace de confirmación de la suscripción. La URL se debe pasar del propietario de la cola al propietario de la suscripción. El propietario de la suscripción debe ingresar la URL en la Consola de Amazon SNS.

5. Inicie sesión como propietario de la suscripción en la Consola de Amazon SNS El propietario de la suscripción realiza la confirmación.
6. Elija el tema correspondiente.
7. Elija la suscripción correspondiente en la tabla de listas de suscripciones del tema. Se etiqueta como "Pendiente de confirmación".
8. Elija Confirm subscription (Confirmar suscripción).
9. Aparece un modal que solicita el enlace de confirmación de la suscripción. Pegue los valores del enlace de confirmación de la suscripción.
10. Seleccione Confirm subscription (Confirmar suscripción) en el modal.

    Se muestra una respuesta XML, por ejemplo:

```
<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>
```

    La cola suscrita está lista para recibir mensajes del tema.
11. (Opcional) Si ve la suscripción del tema en la consola de Amazon SNS, puede ver que el mensaje Confirmación pendiente se ha sustituido por el ARN de suscripción en la columna ID de suscripción.

¿Cómo obligo a una suscripción a requerir autenticación en las solicitudes de cancelación de suscripción?

El propietario de la suscripción debe configurar la marca `AuthenticateOnUnsubscribe` como true en la confirmación de la suscripción.

- `AuthenticateOnUnsubscribe` se establece automáticamente en true cuando el propietario de la cola crea la suscripción.
- `AuthenticateOnUnsubscribe` no se puede establecer en true cuando se navega por el enlace de confirmación de la suscripción sin autenticación.

Envío de mensajes de Amazon SNS a una cola de Amazon SQS o a una función AWS Lambda en una región distinta

Amazon SNS admite entregas entre regiones, tanto para regiones habilitadas de forma predeterminada como para las regiones registradas. Si desea conocer la lista actual de las regiones de AWS compatibles con Amazon SNS, incluidas las regiones registradas, consulte Puntos de conexión y cuotas de Amazon Simple Notification Service en la Referencia general de Amazon Web Services.

Amazon SNS admite la entrega entre regiones de notificaciones a colas de Amazon SQS y a funciones AWS Lambda . Cuando una de las regiones es una región registrada, debe especificar una entidad principal de servicio de Amazon SNS diferente en la política del recurso suscrito.

El comando de suscripción de Amazon SNS debe ejecutarse en la región donde está alojado Amazon SNS. Por ejemplo, si Amazon SNS está en la cuenta “A” de la región us-east-1 y la función de Lambda está en la cuenta “B” de la región us-east-2, el comando de la CLI de suscripción debe ejecutarse en la cuenta “A” de la región us-east-1.

Regiones registradas

Amazon SNS admite las siguientes regiones registradas:

Nombre de la región	Región
Región África (Ciudad del Cabo)	af-south-1

Nombre de la región	Región
Región de Asia-Pacífico (Hong Kong)	ap-east-1
Región de Asia Pacífico (Hyderabad)	ap-south-2
Región Asia-Pacífico (Yakarta)	ap-southeast-3
Región de Asia-Pacífico (Melbourne)	ap-southeast-4
Región Europa (Milán)	eu-south-1
Región Europa (España)	eu-south-2
Región Europa (Zúrich)	eu-central-2
Región Israel (Tel Aviv)	il-central-1
Región Medio Oriente (Baréin)	me-south-1
Región Medio Oriente (EAU)	me-central-1

Para obtener información sobre cómo habilitar una región de suscripción voluntaria, consulte Gestión de AWS regiones en.Referencia general de Amazon Web Services

Cuando se utiliza Amazon SNS para entregar mensajes de las regiones registradas a regiones que están habilitadas de forma predeterminada, debe modificar la política de recursos creada para la cola. Sustituya la entidad principal `sns.amazonaws.com` por `sns.<opt-in-region>.amazonaws.com`. Por ejemplo:

- Si desea suscribir una cola de Amazon SQS en Este de EE. UU. (Norte de Virginia) a un tema de Amazon SNS en Asia-Pacífico (Hong Kong), cambie la entidad principal en la política de cola a `sns.ap-east-1.amazonaws.com`. Las regiones registradas incluyen cualquier región lanzada después del 20 de marzo de 2019, que incluye Asia-Pacífico (Hong Kong), Asia-Pacífico (Yakarta), Medio Oriente (Baréin), Europa (Milán) y África (Ciudad del Cabo). Las regiones lanzadas antes del 20 de marzo de 2019 están habilitadas de forma predeterminada.

Soporte de entrega entre regiones a Amazon SQS

Tipo de entrega entre regiones	Admitido/No admitido	
Región habilitada de forma predeterminada a región de suscripción	Compatible con `sns.<opt-in-region>.amazonaws.com` en la entidad principal de servicio de la cola	
Región de suscripción a región habilitada de forma predeterminada	Compatible con `sns.<opt-in-region>.amazonaws.com` en la entidad principal de servicio de la cola	
Región de suscripción a región de suscripción	No compatible	

El siguiente es un ejemplo de una declaración de política de acceso que permite que un tema de Amazon SNS de una región optativa (af-south-1) realice envíos a una cola de Amazon SQS de una región (us-east-1). enabled-by-default Contiene la configuración de entidad principal de servicio regionalizada necesaria en la ruta `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 suscribir una AWS Lambda función en EE. UU. Este (Virginia del Norte) a un tema de Amazon SNS en Asia Pacífico (Hong Kong), cambie el principio de la política de AWS Lambda funciones a. `sns.ap-east-1.amazonaws.com` Las regiones registradas incluyen cualquier región lanzada después del 20 de marzo de 2019, que incluye Asia-Pacífico (Hong Kong), Asia-Pacífico (Yakarta), Medio Oriente (Baréin), Europa (Milán) y África (Ciudad del Cabo). Las regiones lanzadas antes del 20 de marzo de 2019 están habilitadas de forma predeterminada.

  Soporte de entrega transregional a AWS Lambda

Tipo de entrega entre regiones	Admitido/No admitido
Región habilitada de forma predeterminada a región de suscripción	No compatible
Región de suscripción a región habilitada de forma predeterminada	Compatible con `sns.<opt-in-region>.amazonaws.com` en la entidad principal de servicio de la función Lambda
Región de suscripción a región de suscripción	No compatible

Estado de entrega de mensajes de Amazon SNS

Amazon SNS permite registrar el estado de entrega de los mensajes de notificación enviados a los temas con los siguientes puntos de enlace de Amazon SNS:

- Amazon Data Firehose
- Amazon Simple Queue Service
- AWS Lambda
- HTTPS
- Punto de conexión de aplicación de plataforma

Los registros del estado de entrega se envían a Amazon CloudWatch Logs y proporcionan información sobre las operaciones de entrega de mensajes. Estos registros le ayudan a:

- Determine si un mensaje se entregó correctamente a un punto final.
- Identifique la respuesta del punto final a Amazon SNS.
- Mida el tiempo de permanencia del mensaje (el tiempo transcurrido entre la fecha y hora de publicación y su entrega al punto final).

Puede configurar el registro del estado de la entrega mediante la API AWS Management Console AWS SDKs, Query o. AWS CloudFormation

Requisitos previos para el registro del estado de la entrega

En este tema se describen los permisos de IAM necesarios para permitir que Amazon SNS escriba registros de entrega y se explica la convención de CloudWatch nomenclatura de los grupos de registros predeterminada. Esto garantiza que tiene la configuración y el acceso correctos para monitorear y analizar los registros de entrega de mensajes contenidos en CloudWatch los registros.

Permisos de IAM necesarios

La función de IAM asociada al registro del estado de la entrega debe incluir los siguientes permisos para que Amazon SNS pueda escribir CloudWatch en los registros. Puede utilizar un rol existente con estos permisos o crear uno nuevo durante la configuración.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

Convención de nomenclatura de grupos de registros

De forma predeterminada, Amazon SNS crea grupos de registros para los CloudWatch registros de estado de entrega mediante la siguiente convención de nomenclatura. Los flujos de registro de este grupo corresponden a los protocolos de punto final (por ejemplo, Lambda o Amazon SQS). Asegúrese de tener permisos para ver estos registros en la consola de CloudWatch Logs.

```
sns/<region>/<account-id>/<topic-name>
```

Configuración del registro del estado de entrega mediante la AWS Management Console

En este tema se explica cómo habilitar el registro del estado de entrega de mensajes para los temas de Amazon SNS, incluida la configuración de los ajustes de registro, la asignación de funciones de IAM y la verificación de que CloudWatch Logs capture los registros de entrega para su supervisión y solución de problemas.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. Seleccione el tema deseado y, a continuación, elija Editar.
4. Amplíe la sección de registro del estado de la entrega.
5. Elija el protocolo para el que quiere habilitar el registro (por ejemplo, HTTP, Lambda, Amazon SQS).
6. Introduzca la frecuencia de muestreo de éxito, que es el porcentaje de mensajes satisfactorios de los que desea recibir CloudWatch registros.

7. En la sección Funciones de IAM, debe configurar las funciones para registrar tanto los éxitos como los errores:

   - Utilice un rol de servicio existente: seleccione un rol de IAM existente que tenga los permisos necesarios para que Amazon SNS escriba registros. CloudWatch

   - Crear un nuevo rol de servicio: seleccione Crear nuevos roles para definir los roles de IAM para las entregas correctas y fallidas en la consola de IAM. Para obtener información sobre los permisos, consulte. Requisitos previos para el registro del estado de la entrega
8. Seleccione Save changes (Guardar cambios).

   Tras activar el registro, puede ver y analizar los CloudWatch registros que contienen el estado de entrega del mensaje. Para obtener más información sobre su uso CloudWatch, consulte la CloudWatchdocumentación.

Verificar la configuración del registro

1. Inicie sesión en la consola de CloudWatch Logs.
2. Localice el grupo de registros denominado`sns/<region>/<account-id>/<topic-name>`.
3. Asegúrese de que existan flujos de registro para el protocolo de punto final configurado.
4. Envíe un mensaje de prueba a su tema y confirme que aparecen las entradas de registro que indican que las entregas se han realizado correctamente o han fallado.

Configurar el registro del estado de la entrega mediante el AWS SDKs

Se AWS SDKs proporcionan APIs en varios idiomas para establecer los atributos de los temas para el registro del estado de entrega de los mensajes. Por ejemplo, utilice la SetTopicAttributesAPI para configurar:

- `LambdaSuccessFeedbackRoleArn`— Función de IAM para la entrega exitosa de mensajes a los puntos finales de Lambda.
- `LambdaSuccessFeedbackSampleRate`— Frecuencia de muestreo de los mensajes enviados correctamente a los puntos finales Lambda.
- `LambdaFailureFeedbackRoleArn`— Función de IAM para la entrega fallida de mensajes a los puntos finales de Lambda.

Ejemplo de comando AWS CLI

```
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 los temas

Utilice los siguientes valores de nombre de atributo de tema para el estado de entrega de los mensajes:

HTTP

- `HTTPSuccessFeedbackRoleArn`— Estado de entrega correcto del mensaje para un tema de Amazon SNS que está suscrito a un punto de enlace HTTP.
- `HTTPSuccessFeedbackSampleRate`— Porcentaje de mensajes exitosos que se deben muestrear para un tema de Amazon SNS suscrito a un punto de enlace HTTP.
- `HTTPFailureFeedbackRoleArn`— Estado de entrega de mensajes fallido para un tema de Amazon SNS que está suscrito a un punto de enlace HTTP.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— Estado de entrega correcto del mensaje para un tema de Amazon SNS que está suscrito a un punto de conexión Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate`— Porcentaje de mensajes correctos que se deben muestrear para un tema de Amazon SNS que esté suscrito a un punto de enlace Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn`— Estado de entrega de mensajes fallido para un tema de Amazon SNS que está suscrito a un punto final de Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— El estado de entrega del mensaje se ha realizado correctamente para un tema de Amazon SNS que está suscrito a un punto final de Lambda.
- `LambdaSuccessFeedbackSampleRate`— Porcentaje de mensajes correctos que se deben muestrear para un tema de Amazon SNS suscrito a un punto final de Lambda.

- `LambdaFailureFeedbackRoleArn`— Estado de entrega de mensajes fallido para un tema de Amazon SNS que está suscrito a un punto final de Lambda.

Puntos finales de aplicación de plataforma

- `ApplicationSuccessFeedbackRoleArn`— Estado de entrega correcto del mensaje para un tema de Amazon SNS que está suscrito a un AWS punto final de aplicación.
- `ApplicationSuccessFeedbackSampleRate`— Porcentaje de mensajes correctos que se deben muestrear para un tema de Amazon SNS suscrito a un AWS punto final de aplicación.
- `ApplicationFailureFeedbackRoleArn`— Estado de entrega de mensajes fallido para un tema de Amazon SNS que está suscrito a un AWS punto final de aplicación.

> **Note**
>
> Además, puede configurar los atributos de la aplicación para registrar el estado de entrega directamente en los servicios de notificaciones push. Para obtener más información, consulte Uso de los atributos de la aplicaciones de Amazon SNS para el estado de entrega de los mensajes.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— Estado de entrega correcto del mensaje para un tema de Amazon SNS que está suscrito a un punto final de Amazon SQS.
- `SQSSuccessFeedbackSampleRate`— Porcentaje de mensajes correctos que se deben muestrear para un tema de Amazon SNS suscrito a un punto de conexión de Amazon SQS.
- `SQSFailureFeedbackRoleArn`— Estado de entrega de mensajes fallido para un tema de Amazon SNS que está suscrito a un punto final de Amazon SQS.

Los registros de los puntos de enlace de las aplicaciones de plataforma se escriben en el mismo grupo de CloudWatch registros que los demás puntos de enlace.

> **Note**
>
> Los `<ENDPOINT>FailureFeedbackRoleArn` atributos `<ENDPOINT>SuccessFeedbackRoleArn` y se utilizan para conceder a Amazon

SNS acceso de escritura para usar CloudWatch Logs en su nombre. El atributo `<ENDPOINT>SuccessFeedbackSampleRate` permite especificar el porcentaje de la frecuencia de muestreo (0-100) de los mensajes entregados correctamente. Tras configurar el `<ENDPOINT>FailureFeedbackRoleArn` atributo, todas las entregas de mensajes fallidas generarán CloudWatch registros.

AWS Ejemplos de SDK para configurar los atributos de los temas

Los siguientes ejemplos de código muestran cómo utilizar `SetTopicAttributes`.

CLI

AWS CLI

Establecimiento de un atributo para un tema

En el ejemplo de `set-topic-attributes` siguiente, se establece el atributo `DisplayName` del tema especificado.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte SetTopicAttributesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

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

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

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
```

```
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

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

- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Para obtener más información sobre la API, consulta SetTopicAttributesla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK para PHP de la API.

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK para Ruby de la API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obtener más información sobre la API, consulte SetTopicAttributesla referencia sobre la API ABAP del AWS SDK para SAP.

Configuración del registro del estado de entrega mediante AWS CloudFormation

Para configurar `DeliveryStatusLogging` el uso AWS CloudFormation, usa una plantilla JSON o YAML para crear una AWS CloudFormation pila. Para obtener más información, consulta la `DeliveryStatusLogging` propiedad del `AWS::SNS::Topic` recurso en la Guía del AWS CloudFormation usuario. A continuación, se muestran ejemplos de AWS CloudFormation

plantillas en JSON y YAML para crear un tema nuevo o actualizar un tema existente con todos los `DeliveryStatusLogging` atributos del protocolo Amazon SQS.

Asegúrese de que las funciones de IAM a las que se hace referencia en el `SuccessFeedbackRoleArn` documento `FailureFeedbackRoleArn` tengan los permisos de registro necesarios. CloudWatch

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
```

Reintento de entrega de mensajes de Amazon SNS

Amazon SNS define una política de entrega para cada protocolo de entrega. En la política de entrega, se define cómo Amazon SNS reintenta la entrega de mensajes cuando se producen errores en el servidor (cuando el sistema que aloja el punto de enlace suscrito deja de estar disponible). Cuando se agota la política de entrega, Amazon SNS deja de intentar la entrega y descarta el mensaje, a menos que se adjunte una cola de mensajes fallidos a la suscripción. Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS.

Protocolos y políticas de entrega

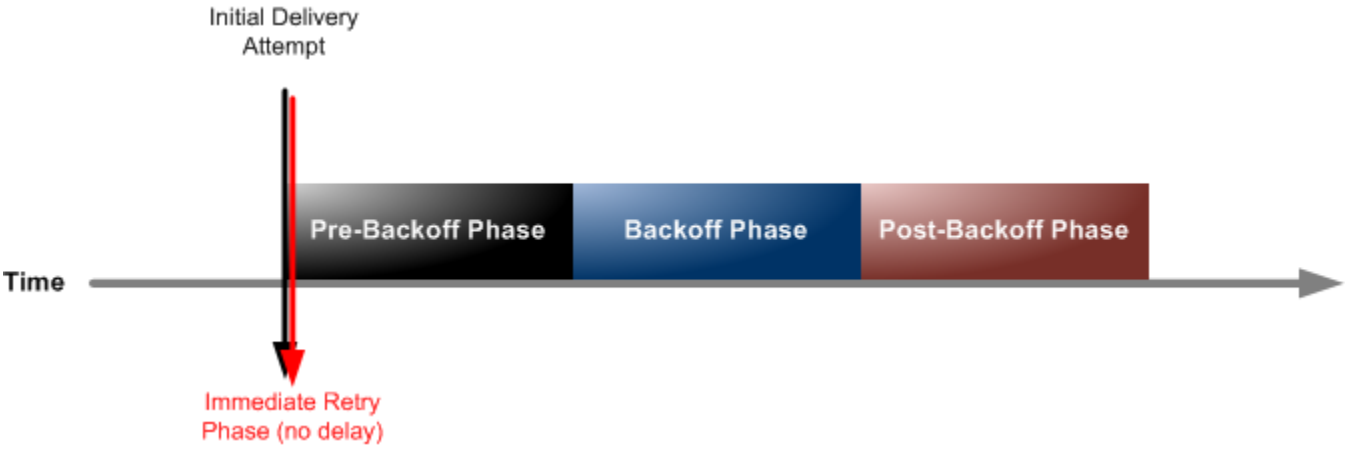
> **Note**
>
> - Con la excepción de las políticas personalizadas, admite políticas personalizadas. HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S Consulte Creación de una política de entrega HTTP/S.
> - Amazon SNS aplica la fluctuación de retardo a los reintentos de entrega. Para obtener más información, consulte la publicación Retroceso exponencial y fluctuación de retardo del blog de arquitectura de AWS .
> - El tiempo total de reintentos de la política para un punto de conexión HTTP/S no puede ser superior a 3600 segundos. Se trata de un límite codificado y no se puede aumentar.

Tipo de punto de conexión	Protocolos de entrega	Fase de reintento inmediato (sin retraso)	Fase previa a la regresión	Fase de regresión	Fase posterior a la regresión	Total de intentos
AWS puntos finales gestionados	Amazon Data Firehose[1]	3 veces, sin retraso	2 veces, 1 segundo de diferencia	10 veces, con retardo exponencial, de 1 segundo a 20 segundos	100 000 veces, con 20 segundos de diferencia	100 015 veces, durante 23 días
	AWS Lambda					
	Amazon SQS					
Puntos de enlace administrados por el cliente	SMTP	0 veces, sin demora	2 veces, 10 segundos de diferencia	10 veces, con retardo exponencial, de 10 segundos a 600 segundos (10 minutos)	38 veces, 600 segundos (10 minutos) de diferencia	50 intentos, durante 6 horas
	SMS					
	Inserción en móvil					

[1] Para los errores de limitación controlada con el protocolo de Firehose, Amazon SNS utiliza la misma política de entrega que para los puntos de conexión administrados por el cliente.

Fases de la política de entrega

En el siguiente diagrama, se muestran las fases de una política de entrega.



Cada política de entrega se compone de cuatro fases.

1. Fase de reintento inmediato (sin demora): esta fase se produce inmediatamente después del intento de entrega inicial. No hay un plazo de tiempo entre los reintentos de esta fase.
2. Fase previa al retraso: esta fase sigue a la fase de reintento inmediato. Amazon SNS utiliza esta fase para intentar realizar un conjunto de reintentos antes de aplicar una función de respaldo. En esta fase se especifica el número de reintentos y el tiempo de retraso entre ellos.
3. Fase de espera: esta fase controla el retraso entre los reintentos mediante la función de retroceso. En esta fase, se establece el retraso mínimo, el retraso máximo y la función retry-backoff, que define con qué rapidez van a ir aumentando los retrasos desde el valor mínimo hasta alcanzar el retraso máximo. La función de retardo puede ser aritmética, exponencial, geométrica o lineal.
4. Fase posterior al retroceso: esta fase sigue a la fase de retroceso. Especifica un número de reintentos y el tiempo de retraso entre ellos. Esta es la fase final.

Creación de una política de entrega HTTP/S

Puede definir la forma en que Amazon SNS reintenta la entrega de mensajes a los puntos de enlace HTTP/S mediante una política de entrega con cuatro fases: sin demoras, previa a la demora, retrasada y posterior a la demora. Esta política le permite anular la configuración de reintentos predeterminada y personalizarla para que se adapte a la capacidad de su servidor HTTP.

Puedes definir tu política de entrega de HTTP/S como un objeto JSON a nivel de tema o de suscripción:

- Política a nivel de tema: se aplica a todas las suscripciones a HTTP/S vinculadas al tema. Usa la acción `CreateTopic`o la acción de la `SetTopicAttributes`API para configurar esta política.

- Política de nivel de suscripción: se aplica solo a una suscripción específica. Usa la acción `Subscribe`o la `SetSubscriptionAttributes`API para configurar esta política.

Como alternativa, también puedes usar el AWS::SNS::Subscriptionrecurso en tus AWS CloudFormation plantillas.

Debe personalizar su política de entrega en función de la capacidad de su servidor HTTP/S:

- Un solo servidor para todas las suscripciones: si todas las suscripciones a HTTP/S de un tema utilizan el mismo servidor, defina la política de entrega como un atributo del tema para garantizar la coherencia en todas las suscripciones.
- Diferentes servidores para las suscripciones: si las suscripciones se dirigen a servidores diferentes, cree una política de entrega única para cada suscripción, adaptada a la capacidad del servidor específico.

También puedes configurar el `Content-Type` encabezado en la política de solicitudes para especificar el tipo de medio de la notificación. De forma predeterminada, Amazon SNS envía todas las notificaciones a los puntos de enlace HTTP/S con el tipo de contenido establecido en. `text/plain; charset=UTF-8` Sin embargo, puede anular este valor predeterminado mediante el `headerContentType`campo de la política de solicitudes.

El siguiente objeto JSON define una política de entrega con reintentos estructurados en cuatro fases:

1. Fase sin demoras: vuelve a intentarlo 3 veces de forma inmediata.
2. Fase previa al retraso: vuelva a intentarlo 2 veces con un intervalo de 1 segundo.
3. Fase de espera: vuelva a intentarlo 10 veces con retrasos exponenciales que oscilan entre 1 y 60 segundos.
4. Fase posterior al retraso: vuelva a intentarlo 35 veces con un intervalo fijo de 60 segundos.

Amazon SNS realiza un total de 50 intentos para entregar un mensaje antes de descartarlo. Para conservar los mensajes que no se puedan entregar después de todos los intentos, configure su suscripción para transferir los mensajes que no se puedan entregar a una cola de mensajes sin entregar (DLQ). Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS.

Amazon SNS considera que se pueden volver a intentar todos los errores 5XX y 429 (se enviaron demasiadas solicitudes). Estos errores están sujetos a la política de entrega. Todos los demás errores se consideran fallos permanentes y no se intentará volver a intentarlo.

> **Note**
>
> Esta política de entrega utiliza la `maxReceivesPerSecond` propiedad para limitar el tráfico de entrega a una media de 10 mensajes por segundo por suscripción. Si bien este mecanismo ayuda a evitar que su terminal HTTP/S se vea abrumado por el alto tráfico, está diseñado para mantener una tasa de entrega media y no impone un límite estricto. De vez en cuando, se pueden producir picos de tráfico de entrega por encima del límite especificado, especialmente si tu tasa de publicación es significativamente superior al límite límite. Cuando el tráfico de publicación (entrante) supera la velocidad de entrega (saliente), puede provocar una acumulación de mensajes pendientes y una mayor latencia de entrega. Para evitar estos problemas, asegúrese de que el `maxReceivesPerSecond` valor se ajuste a los requisitos de capacidad y carga de trabajo de su servidor HTTP/S.

El siguiente ejemplo de política de entrega anula el tipo de contenido predeterminado para la notificación de HTTP/S a. `application/json`

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

La política de entrega se compone de una política de reintentos, una política de aceleración y una política de solicitudes. En total, hay 9 atributos en una política de entrega.

Política	Descripción	Constraint
`minDelayTarget`	Retraso mínimo de un reintento. Unidad: segundos	Entre 1 y el retraso máximo Valor predeterminado: 20
`maxDelayTarget`	Retraso máximo de un reintento. Unidad: segundos	Entre el retraso mínimo y 3600 Valor predeterminado: 20
`numRetries`	Número total de reintentos, incluidos los reintentos inmediatos, los reintentos previos al retardo y los reintentos posteriores al retardo.	Entre 0 y 100 Predeterminado: 3
`numNoDelayRetries`	Número de reintentos que se van a realizar inmediatamente, sin retraso entre ellos.	0 o más Predeterminado: 0
`numMinDelayRetries`	Número de reintentos en la fase previa al retardo, con el retraso mínimo especificado entre ellos.	0 o más Predeterminado: 0
`numMaxDelayRetries`	Número de reintentos en la fase posterior al retardo, con el retraso máximo entre ellos.	0 o más Predeterminado: 0
`backoffFunction`	Modelo de retardo entre reintentos.	Una de las cuatro opciones: • aritmética • exponencial

Política	Descripción	Constraint
		• geométrica • lineal Valor predeterminado: lineal
`maxReceivesPerSecond`	El número medio máximo de envíos de mensajes por segundo y por suscripción.	1 o más Predeterminado: sin limitaciones (sin límite en la tasa de entrega)

Política	Descripción	Constraint
`headerContentType`	El tipo de contenido de la notificación que se envía a los puntos de conexión HTTP/S.	Si no está definida la política de solicitudes, el tipo de contenido será `text/plain; charset=UTF-8` de forma predeterminada. Cuando la entrega de mensajes sin procesar está desactivada para una suscripción (valor predeterminado), o cuando la política de entrega está definida en el nivel de tema, los tipos de contenido de encabezado admitidos son `application/json` y `text/plain` . Cuando se activa la entrega de mensajes sin procesar para una suscripción, se admiten los siguientes tipos de contenido: • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • aplicación/ x-www-form-urlencoded

Política	Descripción	Constraint
		• application/xhtml+xml • application/xml

Amazon SNS utiliza la siguiente fórmula para calcular la cantidad de reintentos en la fase de retardo:

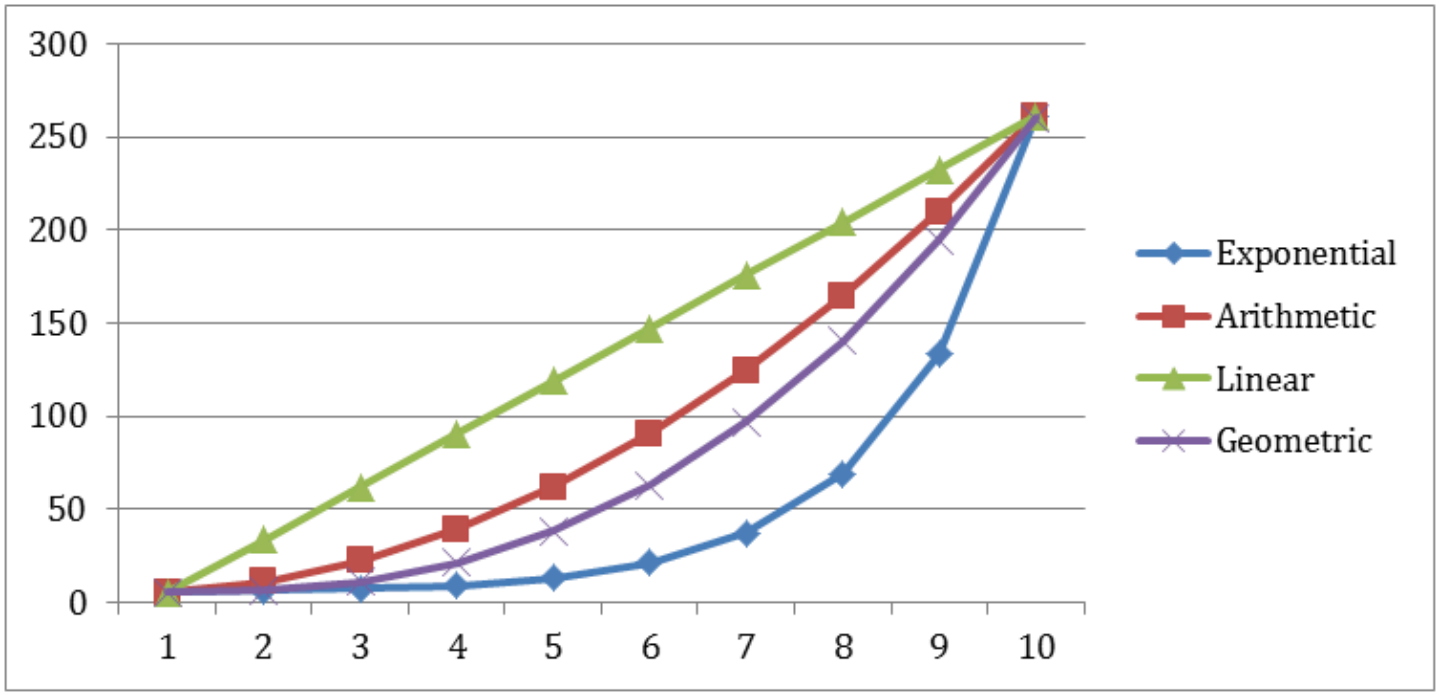
```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

Puede controlar la frecuencia de los reintentos durante la fase de espera mediante tres parámetros:

- **`minDelayTarget`**— Establece el retraso del primer reintento de la fase de espera.
- **`maxDelayTarget`**— Establece el retraso del último reintento en la fase de espera.
- **`backoffFunction`**— Determina el algoritmo que Amazon SNS utiliza para calcular los retrasos de todos los reintentos entre el primer y el último intento. Puede elegir entre cuatro funciones de retroceso disponibles.

El siguiente diagrama ilustra cómo las diferentes funciones de retraso de los reintentos afectan a los retrasos entre los reintentos durante la fase de retraso. La política de entrega utilizada en este ejemplo incluye la siguiente configuración: 10 reintentos en total, un retraso mínimo de 5 segundos y un retraso máximo de 260 segundos.

- El eje vertical muestra el retraso (en segundos) de cada reintento.
- El eje horizontal representa la secuencia de reintentos, que va desde el primer intento hasta el décimo.



Colas de mensajes fallidos de Amazon SNS

Una cola de mensajes fallidos es una cola de Amazon SQS a la que una suscripción de Amazon SNS puede enviar mensajes que no se pueden entregar de manera correcta a los suscriptores. Los mensajes que no se pueden entregar debido a errores del cliente o errores del servidor se mantienen en la cola de mensajes fallidos para su posterior análisis o reprocesamiento. Para obtener más información, consulte Configuración de una cola de mensajes fallidos de Amazon SNS para una suscripción y Reintento de entrega de mensajes de Amazon SNS.

Note

- La suscripción a Amazon SNS y la cola de Amazon SQS deben estar en la AWS misma cuenta y región.
- En un tema FIFO, puede usar una cola FIFO de Amazon SQS como una cola de mensajes fallidos para la suscripción de Amazon SNS. Las suscripciones a temas FIFO utilizan colas FIFO y las suscripciones a temas estándar utilizan colas estándar.
- Para utilizar una cola de Amazon SQS cifrada como cola de letra muerta, debe utilizar un KMS personalizado con una política de claves que conceda al servicio Amazon SNS el acceso principal a las acciones de la API. AWS KMS Para obtener más información, consulte Protección de los datos de Amazon SNS con cifrado del servidor en esta guía

y Protección de datos de Amazon SQS con el cifrado del lado del servidor (SSE) y AWS KMS en la Guía para desarrolladores de Amazon Simple Queue Service.

¿Por qué no se pueden entregar los mensajes?

En general, la entrega de mensajes falla cuando Amazon SNS no puede obtener acceso a un punto de enlace suscrito por un error en el lado del cliente o del servidor. Cuando Amazon SNS recibe un error del lado del cliente o continúa recibiendo un error en el lado del servidor de un mensaje que supera la cantidad de reintentos especificada por la política de reintentos correspondiente, descarta el mensaje, a menos que se adjunte una cola de mensajes fallidos a la suscripción. Las entregas fallidas no cambian el estado de las suscripciones. Para obtener más información, consulte Reintento de entrega de mensajes de Amazon SNS.

Errores del cliente

Los errores del lado del cliente pueden producirse cuando Amazon SNS tiene metadatos obsoletos de la suscripción. Estos errores suelen producirse cuando un propietario elimina el punto de enlace (por ejemplo, una función Lambda suscrita a un tema de Amazon SNS) o cuando un propietario cambia la política asociada al punto de enlace suscrito de forma que impide que Amazon SNS entregue mensajes al punto de enlace. Amazon SNS no vuelve a intentar la entrega del mensaje que falla como resultado de un error del lado del cliente.

Errores del servidor

Los errores del servidor pueden producirse cuando el sistema responsable del punto de enlace suscrito deja de estar disponible o devuelve una excepción que indica que no puede procesar una solicitud válida procedente de Amazon SNS. Cuando se producen errores en el lado del servidor, Amazon SNS reintenta las entregas fallidas mediante una función de retroceso exponencial o lineal. En el caso de los errores del servidor causados por puntos de enlace AWS gestionados respaldados por Amazon SQS o Amazon SNS, Amazon AWS Lambda SNS vuelve a intentar la entrega hasta 100 015 veces, en un plazo de 23 días.

Los puntos de enlace administrados por el cliente (como HTTP, SMTP, SMS o inserción móvil) también pueden causar errores en el lado del servidor. Amazon SNS reintenta también la entrega a estos tipos de puntos de enlace. Aunque los puntos de enlace HTTP son compatibles con las políticas de reintentos definidas por el cliente, Amazon SNS establece una política interna de 50 reintentos de entrega durante 6 horas para los puntos de enlace SMTP, SMS e inserción móvil.

¿Cómo funcionan las colas de mensajes fallidos?

Como las entregas de mensajes se producen en el nivel de la suscripción, se adjunta una cola de mensajes fallidos a la suscripción de Amazon SNS (y no al tema). De este modo, puede identificar más fácilmente el punto de enlace de destino original de cada mensaje.

Las colas de mensajes fallidos que se adjuntan a las suscripciones de Amazon SNS son colas de Amazon SQS normales. Para obtener más información acerca del período de retención de mensajes, consulte Cuotas relacionadas con los mensajes en la Guía para desarrolladores de Amazon Simple Queue Service. Puede cambiar el período de retención de los mensajes con la acción `SetQueueAttributes` de la API de Amazon SQS. Para que las aplicaciones sean más resistentes, le recomendamos que establezca el período máximo de retención de las colas de mensajes fallidos en 14 días.

¿Cómo se transfieren los mensajes a una cola de mensajes fallidos?

Los mensajes se transfieren a la cola de mensajes fallidos a través de una política de redireccionamiento. Una política de redireccionamiento es un objeto JSON que hace referencia al ARN de la cola de mensajes fallidos. El atributo `deadLetterTargetArn` especifica el ARN. El ARN debe apuntar a una cola de Amazon SQS en la Cuenta de AWS misma región y región que su suscripción a Amazon SNS. Para obtener más información, consulte Configuración de una cola de mensajes fallidos de Amazon SNS para una suscripción.

El siguiente objeto JSON es un ejemplo de una política de redireccionamiento asociada a una suscripción de SNS.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

¿Cómo puedo sacar los mensajes de una cola de mensajes fallidos?

Los mensajes se pueden sacar de la cola de mensajes fallidos de dos formas:

- Evitar escribir lógica de consumo de Amazon SQS: establezca la cola de mensajes fallidos como fuente de eventos en la función Lambda para drenar dicha cola.
- Escriba la lógica de consumo de Amazon SQS: utilice la API AWS o el SDK de Amazon SQS AWS CLI o escriba una lógica de consumo personalizada para sondear, procesar y eliminar los mensajes de la cola de letra muerta.

¿Cómo puedo monitorizar y registrar las colas de mensajes fallidos?

Puedes usar CloudWatch las métricas de Amazon para monitorear las colas de cartas muertas asociadas a tus suscripciones de Amazon SNS. Todas las colas de Amazon SQS emiten CloudWatch métricas a intervalos de un minuto. Para obtener más información, consulte CloudWatch las métricas disponibles para Amazon SQS en la Guía para desarrolladores de Amazon Simple Queue Service. Todas las suscripciones a Amazon SNS con colas de letra muerta también emiten métricas. CloudWatch Para obtener más información, consulte Supervisión de temas de Amazon SNS mediante CloudWatch.

Para recibir notificaciones sobre la actividad en sus colas de espera, puede utilizar métricas y alarmas. CloudWatch No es adecuado configurar una alarma para la métrica `NumberOfMessagesSent`, porque esta métrica no captura los mensajes enviados a una DLQ como resultado de intentos de procesamiento fallidos. En su lugar, utilice la métrica `ApproximateNumberOfMessagesVisible`, que captura todos los mensajes actualmente disponibles en la DLQ, incluidos los que se han movido debido a errores de procesamiento.

Ejemplo de configuración de alarmas CloudWatch

1. Cree una CloudWatchalarma para la **`ApproximateNumberOfMessagesVisible`** métrica.
2. Establezca el umbral de alarma en 1 (u otro valor adecuado en función de sus expectativas y del tráfico de la DLQ).
3. Especifique el tema de Amazon SNS al que se enviará la notificación cuando la alarma se desactive. Este tema de Amazon SNS puede enviar la notificación de alarma a cualquier tipo de punto de enlace final (como una dirección de correo electrónico, un número de teléfono o una aplicación de localización móvil).

Puedes usar CloudWatch los registros para investigar las excepciones que provocan el error en las entregas de Amazon SNS y para que los mensajes se envíen a colas de letra muerta. Amazon SNS puede registrar tanto las entregas correctas como las fallidas. CloudWatch Para obtener más información, consulte Atributos de aplicaciones móviles de Amazon SNS.

Configuración de una cola de mensajes fallidos de Amazon SNS para una suscripción

Una cola de mensajes fallidos es una cola de Amazon SQS a la que una suscripción de Amazon SNS puede enviar mensajes que no se pueden entregar de manera correcta a los suscriptores.

Los mensajes que no se pueden entregar debido a errores del cliente o errores del servidor se mantienen en la cola de mensajes fallidos para su posterior análisis o reprocesamiento. Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS y Reintento de entrega de mensajes de Amazon SNS.

En esta página, se muestra cómo puede utilizar el AWS Management Console, un AWS SDK AWS CLI, y AWS CloudFormation para configurar una cola de cartas muertas para una suscripción a Amazon SNS.

> **Note**
>
> En un tema FIFO, puede usar una cola FIFO de Amazon SQS como una cola de mensajes fallidos para la suscripción de Amazon SNS. Las suscripciones a temas FIFO utilizan colas FIFO y las suscripciones a temas estándar utilizan colas estándar.

Requisitos previos

Antes de configurar una cola de mensajes fallidos, complete los siguientes requisitos previos:

1. Cree un tema de Amazon SNS llamado `MyTopic`.
2. Cree una cola de Amazon SQS llamada `MyEndpoint` con el fin de utilizarla como punto de enlace para la suscripción de Amazon SNS.
3. (Omitir para AWS CloudFormation) Suscriba la cola al tema.
4. Cree otra cola de Amazon SQS llamada `MyDeadLetterQueue` con el fin de utilizarla como cola de mensajes fallidos para la suscripción de Amazon SNS.
5. Para conceder a la entidad principal de Amazon SNS acceso a la acción de la API de Amazon SQS, establezca la siguiente política de cola 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 una cola de espera para una suscripción a Amazon SNS mediante el AWS Management Console

Asegúrese de completar los requisitos previos antes de comenzar con este tutorial.

1. Inicie sesión en la consola de Amazon SQS.
2. Cree una cola de Amazon SQS o utilice una cola existente, y anote su ARN en la pestaña Detalles de la cola; por ejemplo:

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

3. Inicie sesión en la consola de Amazon SNS.
4. En el panel de navegación, seleccione Subscriptions (Suscripciones).
5. En la página Subscriptions (Suscripciones), seleccione una suscripción existente y haga clic en Edit (Editar).
6. En la *1234a567-bc89-012d-3e45-6fg7h890123i* página de edición, amplíe la sección Política de Redrive (cola de cartas no válidas) y, a continuación, haga lo siguiente:

   a. Elija Enabled (Habilitado).

   b. Especifique el ARN de una cola de Amazon SQS.

7. Elija Guardar cambios.

   Su suscripción está configurada para usar una cola de mensajes fallidos.

Para configurar una cola de espera para una suscripción a Amazon SNS mediante un SDK AWS

Asegúrese de completar los requisitos previos antes de ejecutar este ejemplo.

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

En el siguiente ejemplo de código, se muestra cómo utilizar `SetSubscriptionAttributesRedrivePolicy`.

Java

SDK para Java 1.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 una cola de espera para una suscripción a Amazon SNS mediante el AWS CLI

Asegúrese de completar los requisitos previos antes de comenzar con este tutorial.

1. Instalar y configurar la AWS CLI. Para obtener más información, consulte la Guía del usuario de AWS Command Line Interface .
2. Use el siguiente 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 una cola de espera para una suscripción a Amazon SNS mediante AWS CloudFormation

Asegúrese de completar los requisitos previos antes de comenzar con este tutorial.

1. Copie el siguiente código JSON a un archivo 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. Inicie sesión en la consola de AWS CloudFormation.
3. En la página Select Template (Seleccionar plantilla), elija Upload a template to Amazon S3 (Cargar una plantilla en Amazon S3), seleccione el archivo `MyDeadLetterQueue.json` y haga clic en Next (Siguiente).
4. En la página Specify Details (Especificar detalles), escriba `MyDeadLetterQueue` en Stack Name (Nombre de pila) y haga clic en Next (Siguiente).
5. En la página Opciones, seleccione Siguiente.
6. En la página Review (Revisar), elija Create (Crear).

   AWS CloudFormation comienza a crear la **`MyDeadLetterQueue`** pila y muestra el estado CREATE_IN_PROGRESS. Cuando se completa el proceso, muestra el estado CREATE_COMPLETE. AWS CloudFormation

Archivo, reproducción y análisis de mensajes de Amazon SNS

Los temas estándar de Amazon SNS admiten el archivo de mensajes a través de Amazon Data Firehose. Puede distribución mediante ramificación las notificaciones a los flujos de entrega de Firehose para enviar notificaciones a destinos de almacenamiento y análisis compatibles con Firehose, incluidos Amazon Simple Storage Service (Amazon S3) y Amazon Redshift, entre otros.

Los temas FIFO de Amazon SNS admiten un archivo de mensajes local y sin código que permite a los propietarios de los temas almacenar (o archivar) los mensajes publicados en un tema durante un máximo de 365 días. En el caso de los temas con una `ArchivePolicy` activa, los suscriptores pueden crear una `ReplayPolicy` para recuperar (o reproducir) los mensajes archivados devueltos a un punto de conexión suscrito. Para obtener más información sobre esta característica, consulte Archivo y reproducción de mensajes de Amazon SNS para temas FIFO.

Características	Temas estándar	Temas FIFO
Archivo de mensajes	Distribución ramificada a los flujos de entrega de Firehose	Archivo de mensajes de Amazon SNS para propietarios de temas FIFO
Reproducción de mensajes	La reproducción de temas estándar no es una característica integrada. Muchos clientes crean los suyos propios a partir de su archivo de mensajes.	Reproducción de mensajes de Amazon SNS para los suscriptores de temas FIFO

Administración y optimización de recursos en Amazon SNS

En este tema se ofrece orientación sobre cómo aprovechar todo el potencial de Amazon SNS garantizando un rendimiento óptimo, reduciendo los costos innecesarios y manteniendo los recursos bien organizados.

Temas

Etiquetado de temas de Amazon SNS

Amazon SNS admite el etiquetado de temas de Amazon SNS. Esto puede ayudarlo a realizar un seguimiento y administrar los costos asociados a sus temas, proporcionar una mayor seguridad en sus políticas de AWS Identity and Access Management (IAM) y permitirle buscar o filtrar fácilmente miles de temas. El etiquetado le permite gestionar sus temas de Amazon SNS mediante Resource AWS Groups. Para obtener más información sobre Resource Groups, consulte la Guía del usuario de AWS Resource Groups.

Etiquetado para asignación de costos

Para organizar e identificar los recursos de Amazon SNS para asignación de costos, puede agregar etiquetas que identifiquen el propósito de un tema. Esto es útil especialmente cuando dispone de muchos temas. Puede utilizar etiquetas de asignación de costes para organizar su AWS factura y reflejar su propia estructura de costes. Para ello, regístrese para que la factura de su AWS cuenta incluya las claves y los valores de las etiquetas. Para obtener más información, consulte Configuración de un informe de asignación de costos mensual en la Guía del usuario de Administración y facturación y costos de AWS.

Por ejemplo, podría agregar etiquetas que representen el centro de costos y el objetivo de sus temas de Amazon SNS, como se indica a continuación:

Recurso	Clave	Valor
Tema 1	Centro de costos	43289
	Aplicación	Procesamiento de pedidos

Recurso	Clave	Valor
Tema 2	Centro de costos	43289
	Aplicación	Procesamiento de pagos
Tema 3	Centro de costos	76585
	Aplicación	Archivado

Este plan de etiquetado le permite agrupar dos temas relacionados con el rendimiento de las máquinas de estado en el mismo centro de costos, mientras etiqueta una actividad no relacionada con una etiqueta de asignación de costos distinta.

Etiquetado para el control de acceso

AWS Identity and Access Management permite controlar el acceso a los recursos en función de las etiquetas. Después de etiquetar sus recursos, proporcione información sobre las etiquetas de recurso en el elemento de condición de una política de IAM para administrar el acceso basado en etiquetas. Para obtener información sobre cómo etiquetar los recursos mediante la consola de Amazon SNS o el SDK de AWS, consulte Configuración de etiquetas.

Puede restringir el acceso a una identidad de IAM. Por ejemplo, puede restringir a `Publish` y `PublishBatch` el acceso a todos los temas de Amazon SNS que incluyan una etiqueta con la clave `environment` y el valor `production`, al mismo tiempo que permite el acceso a todos los demás temas de Amazon SNS. En el ejemplo siguiente, la política restringe la capacidad de publicar mensajes en temas etiquetados con `production`, al tiempo que permite que los mensajes se publiquen en temas etiquetados con `development`. Para obtener más información, consulte Control del acceso mediante etiquetas en la Guía del usuario de IAM.

> **Note**
>
> La configuración del permiso de IAM para `Publish` establece permisos para `Publish` y `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"
      }
    }
  }]
}
```

Etiquetado para búsqueda y filtrado de recursos

Una AWS cuenta puede tener decenas de miles de temas de Amazon SNS (consulte Cuotas de Amazon SNS para obtener más información). Al etiquetar los temas, puede simplificar el proceso de búsqueda o filtrado de temas.

Por ejemplo, puede tener cientos de temas asociados a su entorno de producción. En lugar de tener que buscar manualmente estos temas, puede consultarlos todos con una etiqueta determinada:

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

Configuración de etiquetas para un tema de Amazon SNS

En este tema se explica cómo configurar las etiquetas de un tema de Amazon SNS mediante el AWS Management Console, un AWS SDK o el. AWS CLI

> **Important**
>
> No agregue información de identificación personal (PII) ni otra información confidencial en las etiquetas. Las etiquetas son accesibles para otros servicios de Amazon Web Services, incluida la facturación. Las etiquetas no se han diseñado para usarse con información privada o confidencial.

Publicar, añadir y eliminar etiquetas de un tema de Amazon SNS mediante el AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.

3. En la página Topics (Temas) seleccione un tema y Delete (Eliminar).
4. Expanda la sección Etiquetas.

   Se enumeran las etiquetas añadidas al tema.
5. Modifique las etiquetas del tema:
   - Para agregar una etiqueta, elija Add tag (Agregar etiqueta) y, opcionalmente, ingrese las opciones Key (Clave) y Value (Valor).
   - Para eliminar una etiqueta, elija Remove tag (Quitar etiqueta) junto a un par clave-valor.
6. Elija Guardar cambios.

Agregar etiquetas a un tema mediante un SDK de AWS

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

En los siguientes ejemplos de código, se muestra cómo utilizar `TagResource`.

CLI

AWS CLI

Añadir una etiqueta a un tema

El siguiente ejemplo de `tag-resource` añade una etiqueta de metadatos al tema de Amazon SNS especificado.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte TagResourcela Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta TagResourcela Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta TagResourcela referencia sobre el AWS SDK para la API de Kotlin.

Administración de las etiquetas con las acciones de la API de Amazon SNS

Para administrar etiquetas mediante la API de Amazon SNS, utilice las siguientes acciones de la API:

- ListTagsForResource
- TagResource
- UntagResource

Acciones de API compatibles con ABAC

A continuación, se muestra una lista de acciones de API que compatibles con el control de acceso basado en atributos (ABAC). Para obtener más información sobre ABAC, consulte ¿Para qué sirve ABAC? AWS en la Guía del usuario de IAM.

- AddPermission
- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe

- TagResource
- Unsubscribe
- UntagResource

Orígenes y destinos de eventos de Amazon SNS

Amazon SNS conecta Servicios de AWS los sistemas externos mediante el enrutamiento de las notificaciones basadas en eventos. Amazon SNS recibe eventos de varios tipos Servicios de AWS, como actualizaciones de canalización de datos, acciones de EC2 escalado de Amazon o alertas de seguridad, y publica estos eventos en los temas de Amazon SNS. A continuación, estos temas envían notificaciones a los destinos designados.

Amazon SNS admite dos tipos principales de destinos: Application-to-Application (A2A) y Application-to-Person (A2P). En la mensajería A2A, Amazon SNS puede enviar eventos a Lambda para activar una lógica empresarial personalizada, a Amazon SQS para poner los mensajes en cola y a Amazon Kinesis Data Firehose para transmitir datos a los servicios de almacenamiento y análisis. Para la mensajería A2P, Amazon SNS puede enviar notificaciones por SMS, correo electrónico y notificaciones push a dispositivos móviles, lo que garantiza que los usuarios o los equipos reciban alertas oportunas.

Al actuar como un centro central, Amazon SNS dirige las notificaciones a los lugares correctos, lo que le ayuda a automatizar y administrar su AWS infraestructura de manera más eficaz. Esta configuración permite una integración perfecta entre los servicios y una comunicación fiable con los usuarios y los sistemas.

Fuentes de eventos de Amazon SNS

Amazon SNS se integra con una amplia gama de Servicios de AWS categorías, lo que permite a estos servicios publicar eventos sobre temas de Amazon SNS. Esta integración proporciona notificaciones en tiempo real sobre eventos clave, como los cambios en la infraestructura, el rendimiento de las aplicaciones y la administración de costos.

> **Note**
>
> Amazon SNS presentó temas FIFO en octubre de 2020. Actualmente, la mayoría de AWS los servicios admiten el envío de eventos únicamente sobre temas estándar.

Servicios de análisis

En la siguiente tabla se describe cómo Amazon SNS se integra con servicios de AWS análisis como Athena y AWS Data Pipeline Amazon Redshift para proporcionar notificaciones en tiempo real sobre

eventos clave, incluidas las infracciones de los límites de control, las actualizaciones del estado de las canalizaciones y las actividades de almacenamiento de datos.

Puede utilizar estas integraciones para automatizar las respuestas y mantener una supervisión eficaz de sus operaciones de datos.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Athena: con este, puede analizar los datos en Amazon S3 con SQL estándar.	Reciba notificaciones cuando se superen los límites de control. Para obtener más informaci ón, consulte Establecimiento de los límites de control de uso de datos en la Guía del usuario de Amazon Athena.
AWS Data Pipeline: describe cómo automatizar el movimiento y la transformación de los datos.	Reciba notificaciones sobre el estado de los componentes de canalización. Para obtener más información, consulte SnsAlarm en la Guía para desarrolladores de AWS Data Pipeline .
Amazon Redshift: administra todo el trabajo de configuración, operación y escalado del almacenamiento de datos.	Reciba notificaciones de eventos de Amazon Redshift. Para obtener más información, consulte Notificaciones de eventos de Amazon Redshift en la Guía de administración de Amazon Redshift.

Servicios de integración de la aplicación

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de integración de aplicaciones, por ejemplo AWS Step Functions, EventBridge y permite el enrutamiento de datos y las notificaciones en tiempo real para las aplicaciones críticas para la empresa.

Puede aprovechar estas integraciones para recibir alertas de EventBridge eventos y organizar los flujos de trabajo mediante Step Functions, lo que mejora la automatización y la capacidad de respuesta de sus aplicaciones.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon EventBridge: ofrece un flujo de datos en tiempo real desde sus propias aplicacio nes, aplicaciones software-as-a-service (SaaS) y AWS servicios, y dirige esos datos a los objetivos, incluido Amazon SNS. EventBridge anteriormente se llamaba Events CloudWatch .	Reciba notificaciones de EventBridge eventos. Para obtener más información, consulta EventBridgelos objetivos de Amazon en la Guía del EventBridge usuario de Amazon.
AWS Step Functions— Le permite combinar AWS Lambda funciones y otros AWS servicios para crear aplicaciones críticas para la empresa.	Reciba notificaciones de eventos de Step Functions. Para obtener más información, consulte Llamar a Amazon SNS con Step Functions en la Guía para desarrolladores de AWS Step Functions .

Servicios de administración de costos y facturación

En la siguiente tabla se describe cómo Administración de facturación y costos de AWS se integra con Amazon SNS para proporcionar notificaciones sobre presupuestos, cambios de precios y anomalías de costes.

Puede aprovechar esta integración para configurar los temas de Amazon SNS para recibir alertas en tiempo real sobre AWS sus gastos, lo que le ayudará a controlar los costos y responder a los cargos inesperados de manera eficiente.

Servicio de AWS	Beneficio del uso de Amazon SNS
Administración de facturación y costos de AWS: proporciona funciones con las que puede monitorear sus costos y pagar su factura.	Reciba notificaciones de presupuesto, notificac iones de cambio de precio y alertas de anomalías. Para obtener más información, consulte las páginas siguientes en la Guía del usuario de AWS Billing : • Creación de un tema de Amazon SNS para las notificaciones del presupuesto •

Servicio de AWS	Beneficio del uso de Amazon SNS
	Configuración de notificaciones • Detecta gastos inusuales con AWS Cost Anomaly Detection

Servicios de aplicaciones empresariales

En la siguiente tabla se describe cómo Amazon Chime se integra con Amazon SNS para enviar notificaciones sobre eventos importantes de reuniones, lo que le permite mantenerse informado sobre sus comunicaciones y su programación.

Puede utilizar esta integración para usar las notificaciones de eventos de Amazon Chime SDK para mejorar sus herramientas de colaboración dentro y fuera de su organización.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Chime: con este, puede conocer, conversar y realizar llamadas de empresa dentro y fuera de la organización.	Reciba notificaciones importantes de eventos de reuniones. Para obtener más información, consulte Notificaciones de eventos de Amazon Chime SDK en la Guía para desarrolladores de Amazon Chime.

Servicios de computación

En la siguiente tabla se describe cómo Amazon SNS se integra con varios servicios AWS informáticos, lo que le permite recibir notificaciones de eventos clave, como las acciones de Auto Scaling, las finalizaciones de EC2 Image Builder, los cambios en el entorno de Elastic Beanstalk, los resultados de las funciones de Lambda y los umbrales de métricas de Lightsail.

Puede utilizar estas integraciones para administrar de manera eficaz sus aplicaciones y recursos manteniéndose informado sobre las actualizaciones y acciones criticas en Servicios de AWS.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon EC2 Auto Scaling: le ayuda a tener el número correcto de instancias de Amazon Elastic Compute Cloud (Amazon EC2) disponibles para gestionar la carga de su aplicación.	Reciba notificaciones cuando Auto Scaling lance o finalice EC2 instancias de Amazon en su grupo de Auto Scaling. Para obtener más información, consulte Cómo recibir notificaciones de Amazon SNS cuando su grupo de Auto Scaling se amplía en la Guía del usuario de Amazon EC2 Auto Scaling.
EC2 Image Builder: ayuda a automatizar la creación, la administración y el despliegue de imágenes personalizadas, seguras y de up-to-date servidor que vienen preinstaladas y preconfiguradas con software y ajustes para cumplir con estándares de TI específicos.	Reciba notificaciones cuando se completen las creaciones. Para obtener más información, consulta el artículo Seguimiento de las imágenes de servidor más recientes en las canalizaciones de EC2 Image Builder en el blog de AWS informática.
AWS Elastic Beanstalk: se utiliza para administrar los detalles del aprovisionamiento de capacidad, el equilibrio de carga y el escalado de su aplicación, y para proporcionar el monitoreo del estado de la aplicación.	Reciba notificaciones de eventos importantes que afecten su aplicación. Para obtener más información, consulte Notificaciones de entorno de Elastic Beanstalk con Amazon SNS en la Guía para desarrolladores de AWS Elastic Beanstalk .
AWS Lambda: con este, puede ejecutar código sin aprovisionar ni administrar servidores.	Reciba los datos de salida de la función mediante el establecimiento de un tema de SNS como una cola de mensajes fallidos de Lambda o un destino Lambda. Para obtener más información, consulte Invocación asincrónica en la Guía para desarrolladores de AWS Lambda .
Amazon Lightsail: ayuda a los desarrolladores a empezar a AWS usarlo para crear sitios web o aplicaciones web.	Reciba notificaciones cuando una métrica de una de las instancias, bases de datos o balanceadores de carga cruza un umbral especificado. Para obtener más información, consulte Agregar contactos de notificación en

Servicio de AWS	Beneficio del uso de Amazon SNS
	Amazon Lightsail en la Guía para desarrolladores de Amazon Lightsail.

Servicios de contenedores

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de AWS contenedores, como Amazon EKS Distro y Amazon ECS, lo que le permite realizar un seguimiento de las actualizaciones y los parches de seguridad de los clústeres de Amazon EKS y recibir notificaciones de las nuevas versiones de AMI optimizadas para ECS.

Puede utilizar estas integraciones para mantener la seguridad y la eficiencia de sus implementaciones de contenedores manteniéndose informado sobre las actualizaciones y los cambios importantes.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon EKS Distro: con este, puede crear clústeres de confianza y seguros dondequiera que se implementen sus aplicaciones.	Realice el seguimiento de actualizaciones y parches de seguridad para clústeres creados con la distribución de Amazon EKS. Para obtener más información, consulte Presentación de Amazon EKS Distro: una distribución de código abierto de Kubernetes utilizada por Amazon EKS.
Amazon Elastic Container Service (Amazon ECS): con este, puede ejecutar, detener y administrar contenedores en un clúster.	Reciba notificaciones cuando esté disponible una nueva AMI optimizada para Amazon ECS. Para obtener más información, consulte Suscripción a las notificaciones de actualización de AMI optimizadas para Amazon ECS en la Guía para desarrolladores de Amazon Elastic Container Service.

Servicios de interacción con los clientes

En la siguiente tabla se describe cómo Amazon SNS mejora los servicios de fidelización de clientes mediante la integración con Amazon Connect y Amazon Simple Email Service (SES), lo que le permite recibir alertas y validaciones, configurar la mensajería SMS bidireccional y supervisar las notificaciones por correo electrónico para detectar rebotes, quejas y entregas. AWS End User Messaging SMS

Estas integraciones le ayudan a administrar las comunicaciones con los clientes a través de varios canales.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Connect: con este, puede configura r un centro de contacto en la nube omnicanal para captar clientes.	Reciba alertas y validaciones. Para obtener más información, consulte El poder de AWS Amazon Connect en la Guía del administrador de Amazon Connect.
AWS End User Messaging SMS: le ayuda a interactuar con sus clientes enviándoles correo electrónico, mensajes SMS y de voz y notificac iones push.	Configure los SMS bidireccionales, que le permiten recibir mensajes de sus clientes. Para obtener más información, consulte Two-way SMS messaging en la Guía del usuario de AWS End User Messaging SMS .
Amazon Simple Email Service (Amazon SES): se utiliza para proporcionar una forma rentable de enviar y recibir correos electrónicos a través de sus propios dominios y direcciones de correo electrónico.	Reciba notificaciones de rebotes, reclamos y entregas. Para obtener más información, consulte Configuración de las notificaciones de Amazon SNS para Amazon SES en la Guía para desarrolladores de Amazon Simple Notification Service.

Servicios de bases de datos

En la siguiente tabla se describe cómo Amazon SNS se integra con servicios de AWS bases de datos como AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon, Amazon Neptune, Amazon Redshift y ElastiCache Amazon Relational Database Service (RDS) para enviar

notificaciones sobre eventos importantes, como migraciones de datos, actividades de mantenimiento, actualizaciones de caché y cambios en la base de datos.

Estas integraciones le ayudan a supervisar y administrar sus entornos de bases de datos de manera más eficaz al proporcionar alertas puntuales sobre los principales eventos operativos.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS Database Migration Service: migra datos de bases de datos en las instalaciones a la Nube de AWS.	Reciba notificaciones cuando se produzcan AWS DMS eventos; por ejemplo, cuando se cree o elimine una instancia de replicación. Para obtener más información, consulte Trabajo con eventos y notificaciones en AWS Database Migration Service en la Guía del usuario de AWS Database Migration Service .
Amazon DynamoDB: se utiliza para ofrecer un rendimiento rápido y predecible con una escalabilidad perfecta en este servicio de bases de datos NoSQL completamente administrado.	Reciba notificaciones cuando se produzcan eventos de mantenimiento. Para obtener más información, consulte Personalización de la configuración del clúster DAX en laGuía para desarrolladores de Amazon DynamoDB.
Amazon ElastiCache: proporciona una caché en memoria rentable, redimensionable y de alto rendimiento, a la vez que elimina la complejidad asociada a la implementación y la administración de un entorno de caché distribuida.	Reciba notificaciones cuando se produzcan eventos significativos. Para obtener más información, consulte Notificaciones de eventos y Amazon SNS en la Guía del usuario de Amazon ElastiCache (Memcached).
Amazon Neptune: con este, se puede crear y ejecutar aplicaciones que funcionan con conjuntos de datos altamente conectados.	Reciba notificaciones cuando se produce un evento Neptune. Para obtener más información, consulte Uso de notificación de eventos de Neptune en la Guía del usuario de Neptune.
Amazon Redshift: administra todo el trabajo de configuración, operación y escalado del almacenamiento de datos.	Reciba notificaciones de eventos de Amazon Redshift. Para obtener más información, consulte Notificaciones de eventos de Amazon Redshift en la Guía de administración de Amazon Redshift.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Relational Database Service: facilita la configuración, el funcionamiento y el escalado de una base de datos relacional en AWS la nube.	Reciba notificaciones de eventos de Amazon RDS. Para obtener más información, consulte Uso de las notificaciones de eventos de Amazon RDS en la Guía del usuario de Amazon RDS.

Servicios de herramientas para desarrolladores

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de herramientas para AWS desarrolladores, como AWS CodeBuild,, AWS CodeDeploy Amazon AWS CodeCommit, y AWS CodePipeline cómo proporciona notificaciones sobre eventos críticos CodeGuru, como cambios en el estado de la compilación, actualizaciones de repositorios, progreso de la implementación, anomalías de rendimiento y acciones de canalización.

Estas integraciones le ayudan a supervisar y administrar de manera eficaz sus flujos de trabajo de desarrollo de software al recibir alertas puntuales sobre eventos importantes.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS CodeBuild: se utiliza para compilar el código fuente, ejecuta pruebas unitarias y produce artefactos listos para implementarse.	Reciba notificaciones cuando las creacione s tienen éxito, fallan o se mueven de una fase de compilación a otra. Para obtener más información, consulte el ejemplo de creación de notificaciones CodeBuild en la Guía del AWS CodeBuild usuario.
AWS CodeCommit: se utiliza para proporcio nar control de versiones para almacenar y administrar activos de forma privada en la nube.	Reciba notificaciones sobre los eventos CodeCommit del repositorio. Para obtener más información, consulte Ejemplo: creación de un AWS CodeCommit disparador para un tema de Amazon SNS en la Guía del AWS CodeCommi t usuario.
AWS CodeDeploy— Automatiza las implement aciones de aplicaciones en instancias de	Reciba notificaciones sobre despliegues o eventos de CodeDeploy instancias. Para

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon, EC2 instancias locales, funciones Lambda sin servidor o servicios de Amazon ECS.	obtener más información, consulte Crear un desencadenante para un CodeDeploy evento en la Guía del AWS CodeDeploy usuario.
Amazon CodeGuru: recopila datos de rendimiento en tiempo de ejecución de sus aplicaciones activas y proporciona recomenda ciones que pueden ayudarlo a ajustar el rendimiento de sus aplicaciones.	Reciba notificaciones cuando se produzcan anomalías. Para obtener más información, consulta Cómo trabajar con informes de anomalías y recomendaciones en la Guía del CodeGuru usuario de Amazon.
AWS CodePipeline: se utiliza para automatiz ar los pasos necesarios y liberar de manera continua los cambios de software.	Reciba notificaciones sobre las acciones de aprobación. Para obtener más información, consulte Gestionar las acciones de aprobació n CodePipeline en la Guía del AWS CodePipel ine usuario.

Servicios web y móviles front-end

En la siguiente tabla se describe cómo Amazon SNS se integra AWS End User Messaging SMS para mejorar la interacción con los clientes mediante el envío de correos electrónicos, SMS, mensajes de voz y notificaciones push, incluida la posibilidad de configurar SMS bidireccionales para recibir los mensajes de los clientes.

Esta integración le permite interactuar de forma más eficaz con sus clientes a través de varios canales de comunicación.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS End User Messaging SMS: le ayuda a interactuar con sus clientes enviándoles correo electrónico, mensajes SMS y de voz y notificac iones push.	Configure los SMS bidireccionales, que le permiten recibir mensajes de sus clientes. Para obtener más información, consulte Two-way SMS messaging en la Guía del usuario de AWS End User Messaging SMS .

Servicios de desarrollo de juegos

En la siguiente tabla se describe cómo Amazon SNS se integra con los GameLift servidores de Amazon para proporcionar notificaciones sobre eventos de emparejamiento y colas en servidores de juegos multijugador basados en sesiones.

Esta integración ayuda a los desarrolladores de juegos a automatizar y supervisar la implementación, el funcionamiento y el escalado de sus servidores de juegos, lo que garantiza una experiencia de juego fluida.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon GameLift Servers: proporciona soluciones para alojar servidores de juegos multijugador basados en sesiones en la nube, incluido un servicio totalmente gestionado para implementar, operar y escalar servidores de juegos.	Reciba notificaciones de eventos de emparejamiento y cola. Para obtener más información, consulte las páginas siguientes: • Para ver las notificaciones de emparejamiento, consulta Cómo configurar la notificación de FlexMatch eventos en la Guía para FlexMatch desarrolladores de Amazon GameLift Servers. • Para ver las notificaciones de cola, consulta Configurar la notificación de eventos para la ubicación de las sesiones de juego en la Guía para desarrolladores de Amazon GameLift Servers.

Servicios del Internet de las cosas

En la siguiente tabla se describe cómo Amazon SNS se integra AWS IoT con servicios AWS IoT Core como,, AWS IoT Greengrass y AWS IoT Device Defender AWS IoT Events, para proporcionar notificaciones de eventos y alertas de IoT.

Estas integraciones le permiten supervisar de manera eficaz el comportamiento de los dispositivos, recibir alertas de actividades anómalas y administrar los dispositivos de IoT con actualizaciones y acciones en tiempo real.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS IoT Core— Proporciona los servicios en la nube que conectan sus dispositivos de IoT a otros dispositivos y Nube de AWS servicios.	Reciba notificaciones de AWS IoT Core eventos. Para obtener más información, consulte Creación de una regla de Amazon SNS en la Guía para desarrolladores de AWS IoT .
AWS IoT Device Defender: con este, puede auditar la configuración de los dispositivos, monitorear los dispositivos conectados para detectar un comportamiento anormal y mitigar los riesgos de seguridad.	Recibe alarmas cuando un dispositivo infringe un comportamiento. Para obtener más información, consulta Cómo usar la AWS IoT Device Defender detección en la Guía para AWS IoT desarrolladores.
AWS IoT Events: con este, puede monitorea r sus flotas de equipos y dispositivos para saber si se producen errores o cambios en su operación, y para activar acciones cuando se produzcan estos eventos.	Reciba notificaciones de AWS IoT Events eventos. Para obtener más información, consulte Amazon Simple Notification Service en la Guía para desarrolladores de AWS IoT Events .
AWS IoT Greengrass— Se extiende AWS a los dispositivos físicos para que puedan actuar localmente en función de los datos que generan y, al mismo tiempo, utilizar la nube para la gestión, el análisis y el almacenamiento duradero.	Reciba notificaciones de AWS IoT Greengras s eventos. Para obtener información, consulte Conector de SNS en la Guía para desarroll adores de AWS IoT Greengrass Version 1 .

Servicios de Machine Learning

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de aprendizaje AWS automático, como Amazon, CodeGuru Amazon DevOps Guru, Amazon Lookout for Metrics, Amazon Rekognition y SageMaker Amazon AI, para proporcionar notificaciones de anomalías, información operativa y actividades de etiquetado de datos.

Estas integraciones le permiten supervisar el rendimiento de las aplicaciones, recibir alertas sobre irregularidades en los datos y optimizar la implementación de modelos de machine learning con actualizaciones en tiempo real.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon CodeGuru: recopila datos de rendimiento en tiempo de ejecución de sus aplicaciones activas y proporciona recomenda ciones que pueden ayudarlo a ajustar el rendimiento de sus aplicaciones.	Reciba notificaciones cuando se produzcan anomalías. Para obtener más información, consulta Cómo trabajar con informes de anomalías y recomendaciones en la Guía del CodeGuru usuario de Amazon.
Amazon DevOps Guru: genera información operativa mediante el aprendizaje automático para ayudarlo a mejorar el rendimiento de sus aplicaciones operativas.	Reenvíe información y confirmaciones. Para obtener más información, consulte Entregue información operativa basada en ML a sus equipos de guardia mediante PagerDuty Amazon DevOps Guru en el blog AWS Manag ement & Governance.
Amazon Lookout for Metrics: se utiliza para buscar anomalías en los datos, determina sus causas raíz y tomar medidas con rapidez.	Reciba notificaciones de anomalías. Para obtener más información, consulte Uso de Amazon SNS con Lookout for Metrics en la Guía para desarrolladores de Amazon Lookout for Metrics.
Amazon Rekognition: con este, puede agregar con facilidad el análisis de imagen y video a sus aplicaciones	Reciba notificaciones de los resultados de la solicitud. Para obtener más información, consulte Referencia: notificación de resultado s de análisis de video en la Guía para desarroll adores de Amazon Rekognition.
Amazon SageMaker AI: permite a los científic os de datos y desarrolladores crear y entrenar modelos de aprendizaje automático y, a continuación, implementarlos directamente en un entorno hospedado listo para la producción.	Reciba notificaciones cuando un objeto de datos está etiquetado. Para obtener más información, consulte Creación de un trabajo de etiquetado de streaming en la Guía para desarrolladores de Amazon SageMaker AI.

Servicios de administración y gobernanza

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de AWS administración y gobierno, como Amazon Q Developer en las aplicaciones de chat,,,,,,, y,,,,,,,,,,,,,,,

AWS CloudFormation,, CloudTrail,,, CloudWatch,, AWS Config,, AWS Control Tower,, AWS License Manager,,, AWS Service Catalog,,, AWS Systems Manager,,,, y,,,,,,,,,,,,,,,,,,,,,,,,,,,,

Estas integraciones lo ayudan a monitorear y administrar sus AWS entornos de manera eficiente al enviar alertas y actualizaciones oportunas a los equipos y sistemas pertinentes.

Servicio de AWS	Beneficio del uso de Amazon SNS
Desarrollador de aplicaciones de chat de Amazon Q: permite DevOps a los equipos de desarrollo de software utilizar las salas de chat de Amazon Chime y Slack para monitorear y responder a los eventos operativos en el. Nube de AWS	Entregue notificaciones a salas de conversaciones. Para obtener más información, consulte Configuración de Amazon Q Developer en aplicaciones de chat en la Guía del administrador de Amazon Q Developer en aplicaciones de chat.
AWS CloudFormation— Le permite crear y aprovisionar despliegues de AWS infraestructura de forma predecible y repetitiva.	Reciba notificaciones cuando se crean y actualizan las pilas. Para obtener más información, consulte Configuración de las opciones de pila de AWS CloudFormation en la Guía del usuario de AWS CloudFormation .
AWS CloudTrail: se utiliza para ofrecer el historial de eventos de su actividad de Cuenta de AWS .	Reciba notificaciones cuando CloudTrail publique nuevos archivos de registro en su bucket de Amazon S3. Para obtener más información, consulte Configuración de las notificaciones de Amazon SNS CloudTrail en la Guía del AWS CloudTrail usuario.
Amazon CloudWatch: supervisa sus AWS recursos y las aplicaciones en las que se ejecuta AWS en tiempo real.	Reciba notificaciones cuando las alarmas cambian de estado. Para obtener más información, consulta Uso de CloudWatch alarmas de Amazon en la Guía del CloudWatch usuario de Amazon.
AWS Config— Proporciona una vista detallada de la configuración de AWS los recursos de su Cuenta de AWS.	Reciba notificaciones cuando se actualicen los recursos o cuando AWS Config evalúe las reglas personalizadas o administradas con respecto a sus recursos. Para obtener

Servicio de AWS	Beneficio del uso de Amazon SNS
	más información, consulte el tema Notificaciones que se AWS Config envían a un SNS y Ejemplos de notificaciones de cambio de elementos de configuración en la Guía para AWS Config desarrolladores.
AWS Control Tower— Le permite configurar y gestionar un entorno seguro, compatible y con múltiples cuentas AWS .	Utilice alertas para evitar la desviación dentro de su zona de destino y reciba notificaciones de conformidad. Para obtener más información, consulte Amazon Simple Notification Service en la Guía del usuario de AWS Control Tower .
AWS License Manager— Le ayuda a gestionar las licencias de software de los proveedores de software de forma centralizada en todos sus AWS entornos locales y locales.	Reciba notificaciones y alertas de License Manager. Para obtener más información, consulte Configuración de License Manager en la Guía del usuario de License Manager y Creación de ServiceNow incidentes para AWS License Manager notificaciones en el blog AWS Management & Governance.
AWS Service Catalog: con este, los administradores de TI pueden crear, administrar y distribuir carteras de productos aprobados para que los usuarios finales puedan obtener acceso a ellos a través de un portal personalizado.	Reciba notificaciones sobre eventos de pila. Para obtener más información, consulte AWS Service Catalog notification constraints (Restricciones de notificación de Service Catalog) en la Guía del administrador de Service Catalog.
AWS Systems Manager— Le permite ver y controlar su infraestructura AWS.	Reciba notificaciones sobre el estado de los comandos. Para obtener más información, consulte Monitoreo de los cambios de estado de Systems Manager mediante notificaciones de Amazon SNS en la Guía del usuario de AWS Systems Manager .

Servicios multimedia

En la siguiente tabla se describe cómo Amazon SNS se integra con Amazon Elastic Transcoder para enviar notificaciones cuando los trabajos de transcodificación multimedia cambian de estado, lo que le permite supervisar y administrar de forma eficaz la conversión de los archivos multimedia almacenados en Amazon S3 a formatos adecuados para los dispositivos de reproducción de los consumidores.

Esta integración le ayuda a optimizar los flujos de trabajo de procesamiento multimedia al proporcionar alertas en tiempo real sobre el estado del trabajo.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Elastic Transcoder: con este, puede convertir archivos de medios que haya guardado en Amazon S3 en archivos de medios en los formatos compatibles con los reproductores de consumo.	Reciba notificaciones cuando los trabajos cambian de estado. Para obtener más información, consulte Notificaciones del estado de trabajo en la Guía para desarrolladores de Amazon Elastic Transcoder.

Servicios de migración y transferencia

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de AWS migración y transferencia, como AWS Database Migration Service (DMS) AWS Application Discovery Service, y cómo proporciona notificaciones para eventos como la recopilación de datos del servidor AWS Snowball Edge, las actividades de migración de bases de datos y los trabajos de transferencia de datos.

Estas integraciones le ayudan a administrar y supervisar de forma eficaz sus procesos de migración a la nube al ofrecer alertas y actualizaciones en tiempo real sobre tareas de migración críticas.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS Application Discovery Service— Le ayuda a planificar la migración a Internet Nube de AWS mediante la recopilación de datos de uso y configuración sobre sus servidores locales.	Reciba notificaciones de eventos a través AWS CloudTrail de. Para obtener más informaci ón, consulte Registro de llamadas a la API de Application Discovery Service con AWS

Servicio de AWS	Beneficio del uso de Amazon SNS
	CloudTrail en la Guía del usuario Application Discovery Service.
AWS Database Migration Service: migra datos de bases de datos en las instalaciones a la Nube de AWS.	Reciba notificaciones cuando se produzcan AWS DMS eventos; por ejemplo, cuando se cree o elimine una instancia de replicaci ón. Para obtener más información, consulte Trabajo con eventos y notificaciones en AWS Database Migration Service en la Guía del usuario de AWS Database Migration Service .
AWS Snowball Edge— Utiliza dispositivos de almacenamiento físico para transferir grandes cantidades de datos entre Amazon S3 y su ubicación de almacenamiento de datos in situ a gran faster-than-internet velocidad.	Reciba notificaciones de trabajos de Snowball Edge. Para obtener más información, consulte Notifications for Snow Family devices en la Guía del usuario de AWS Snowball Edge .

Servicios de redes y entrega de contenido

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de AWS redes y entrega de contenido, como Amazon API Gateway, Amazon, Elastic Load Balancing CloudFront AWS Direct Connect, Amazon Route 53 y Amazon VPC, para enviar notificaciones sobre eventos como mensajes de API, alarmas de CloudFront métricas, cambios en el estado de la conexión, eventos del equilibrador de carga, estados de comprobación de estado y actividades de punto final de VPC.

Estas integraciones le ayudan a supervisar y administrar sus operaciones de red y entrega de contenido al proporcionar alertas y actualizaciones puntuales.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon API Gateway: le permite crear e implementar su propio REST WebSocket APIs a cualquier escala.	Reciba mensajes publicados en un punto de enlace de API Gateway. Para obtener más información, consulte el tutorial: Creación de una API REST de API Gateway con AWS

Servicio de AWS	Beneficio del uso de Amazon SNS
	integración en la Guía para desarrolladores de API Gateway.
Amazon CloudFront: acelera la distribución de su contenido web estático y dinámico, como .html, .css, .php, imágenes y archivos multimedia.	Reciba notificaciones cuando se produzcan alarmas basadas en métricas específicas CloudFront . Para obtener más información, consulta Cómo configurar alarmas para recibir notificaciones en la Guía para CloudFront desarrolladores de Amazon.
AWS Direct Connect— Conecta su red interna a una AWS Direct Connect ubicación a través de un cable Ethernet de fibra óptica estándar.	Reciba notificaciones cuando las alarmas monitorean el estado de un estado de cambio de conexión de AWS Direct Connect . Para obtener más información, consulte Creación de CloudWatch alarmas para supervisar las AWS Direct Connect conexiones en la Guía del AWS Direct Connect usuario.
Elastic Load Balancing: distribuye automátic amente el tráfico entrante entre varios destinos, como EC2 instancias de Amazon, contenedo res y direcciones IP, en más o más zonas de disponibilidad.	Reciba notificaciones de alarmas que haya creado para eventos de balanceador de carga. Para obtener más información, consulta Cómo crear CloudWatch alarmas para tu balancead or de cargas en la Guía del usuario de los balanceadores de carga clásicos.
Amazon Route 53: se utiliza para proporcionar registro de dominio, enrutamiento de DNS y comprobación de estado.	Reciba notificaciones cuando una comprobac ión de estado es incorrecta. Para obtener más información, consulte Recibir una notificación de Amazon SNS cuando una comprobación de estado es incorrecta (consola) en la Guía para desarrolladores de Amazon Route 53.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Virtual Private Cloud (Amazon VPC): le permite lanzar AWS recursos en una red virtual que haya definido.	Reciba notificaciones cuando se produzcan eventos específicos en los puntos de enlace de interfaz. Para obtener más información, consulte Cree y administre una notificación para un servicio de punto de enlace en la Guía del usuario de Amazon VPC.

Servicios de seguridad, identidad y conformidad

En la siguiente tabla se describe cómo Amazon SNS se integra con los servicios de AWS seguridad, identidad y conformidad, como AWS Directory Service Amazon, Amazon Inspector y GuardDuty AWS Security Hub, para proporcionar notificaciones sobre cambios de estado del directorio, hallazgos de seguridad, eventos del Inspector y anuncios de centros de seguridad.

Estas integraciones le ayudan a mantener buenas prácticas de seguridad al ofrecer alertas y actualizaciones puntuales sobre los eventos de seguridad y conformidad.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS Directory Service— Proporciona varias formas de utilizar Microsoft Active Directory (AD) con otros Servicios de AWS.	Reciba correos electrónicos o mensajes de texto (SMS) cuando cambie el estado de su directorio. Para obtener más información, consulte Configure notificaciones de estado de directorio en la Guía de administración de AWS Directory Service .
Amazon GuardDuty: proporciona una supervisión de seguridad continua para ayudar a identificar actividades inesperadas y potencialmente no autorizadas o maliciosas en su AWS entorno.	Reciba notificaciones sobre nuevos tipos de resultados publicados, actualizaciones para los tipos de resultados existentes u otros cambios en funcionalidades. Para obtener más información, consulte el tema Suscripción a GuardDuty anuncios (SNS) en la Guía GuardDuty del usuario de Amazon.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon Inspector: comprueba la accesibilidad a la red de sus EC2 instancias de Amazon y el estado de seguridad de las aplicaciones que se ejecutan en esas instancias.	Reciba notificaciones de eventos de Amazon Inspector. Para obtener más información, consulte Configuración de un tema de SNS para las notificaciones de Amazon Inspector en la Guía del usuario de Amazon Inspector.
AWS Security Hub: automatiza los controles de seguridad de AWS y centraliza las alertas de seguridad.	Reciba notificaciones sobre AWS Security Hub anuncios, incluidas notificaciones sobre AWS Security Hub controles o estándares que se hayan agregado, editado o retirado. Para obtener más información, consulte Suscribir se a AWS Security Hub anuncios con Amazon SNS.

Servicios sin servidores

En la siguiente tabla se describe cómo Amazon SNS se integra con servicios como Amazon DynamoDB EventBridge, Amazon y Lambda para enviar notificaciones sobre eventos clave, como actualizaciones de mantenimiento, flujos de datos en tiempo real y resultados de funciones de Lambda.

Estas integraciones le ayudan a supervisar y administrar sus aplicaciones de manera eficaz al recibir alertas puntuales sobre las operaciones críticas y automatizar las respuestas a estos eventos.

Servicio de AWS	Beneficio del uso de Amazon SNS
Amazon DynamoDB: se utiliza para ofrecer un rendimiento rápido y predecible con una escalabilidad perfecta en este servicio de bases de datos NoSQL completamente administrado.	Reciba notificaciones cuando se produzcan eventos de mantenimiento. Para obtener más información, consulte Personalización de la configuración del clúster DAX en laGuía para desarrolladores de Amazon DynamoDB.
Amazon EventBridge: ofrece un flujo de datos en tiempo real desde sus propias aplicacio nes software-as-a-service (SaaS) Servicios	Reciba notificaciones de EventBridge eventos. Para obtener más información, consulta

Servicio de AWS	Beneficio del uso de Amazon SNS
de AWS y dirige esos datos a los objetivos, incluido Amazon SNS. EventBridge anteriorm ente se llamaba Events CloudWatch .	EventBridgelos objetivos de Amazon en la Guía del EventBridge usuario de Amazon.
AWS Lambda: con este, puede ejecutar código sin aprovisionar ni administrar servidores.	Reciba los datos de salida de la función mediante el establecimiento de un tema de SNS como una cola de mensajes fallidos de Lambda o un destino Lambda. Para obtener más información, consulte Invocación asincróni ca en la Guía para desarrolladores de AWS Lambda .

Servicios de almacenamiento

En la siguiente tabla se describe cómo Amazon SNS se integra con servicios de AWS almacenamiento como AWS Backup Amazon Elastic File System (EFS), Amazon S3 Glacier y Amazon S3, y AWS Snowball Edge cómo proporciona notificaciones sobre diversos eventos, como actividades de respaldo, alarmas del sistema de archivos, trabajos de recuperación de datos, cambios de bucket y operaciones de transferencia de datos.

Estas integraciones le ayudan a supervisar y administrar sus soluciones de almacenamiento de manera eficaz al recibir alertas puntuales sobre eventos de almacenamiento críticos.

Servicio de AWS	Beneficio del uso de Amazon SNS
AWS Backup: le ayuda a centralizar y automatizar las copias de seguridad de datos en Servicios de AWS en la nube y en las instalaciones.	Reciba notificaciones de eventos. AWS Backup Para obtener más información, consulte Uso de Amazon SNS para realizar un seguimiento de AWS Backup eventos en la Guía para AWS Backup desarrolladores.
Amazon Elastic File System: proporciona almacenamiento de archivos para sus EC2 instancias de Amazon.	Reciba notificaciones de alarmas que haya creado para eventos de Amazon EFS. Para obtener más información, consulte Herramien

Servicio de AWS	Beneficio del uso de Amazon SNS
	tas de monitoreo automatizado en la Guía del usuario de Amazon Elastic File System.
Amazon S3 Glacier: se utiliza con el fin de ofrecer almacenamiento para los datos que se utilizan con poca frecuencia.	Configure las notificaciones de un almacén para que, cuando se complete un trabajo, se envíe un mensaje a un tema de SNS. Para obtener más información, consulte Configuración de notificaciones de almacén en Amazon S3 Glacier en la Guía para desarrolladores de Amazon S3 Glacier.
Amazon Simple Storage Service (Amazon S3): se utiliza para ofrecer almacenamiento de objetos.	Reciba notificaciones cuando se producen cambios en un bucket de Amazon S3 o en una rara instancia cuando los objetos no se replican en su región de destino. Para obtener más información, consulte la Explicación: configuración de un bucket para notificaciones (tema de SNS o cola de SQS) y Monitoreo del progreso con métricas de replicación y notificaciones de eventos de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.
AWS Snowball Edge— Utiliza dispositivos de almacenamiento físico para transferir grandes cantidades de datos entre Amazon S3 y su ubicación de almacenamiento de datos in situ a gran faster-than-internet velocidad.	Reciba notificaciones de trabajos de Snowball Edge. Para obtener más información, consulte Notifications for Snow Family devices en la Guía del usuario de AWS Snowball Edge .

Orígenes de fuentes adicionales

En la siguiente tabla se describe cómo se puede utilizar Amazon SNS para recibir notificaciones puntuales sobre las actualizaciones AWS diarias de las funciones y los cambios en los intervalos de direcciones AWS IP, lo que garantiza que esté informado sobre las últimas versiones de AWS servicio, los tipos de instancias, los puntos de enlace de la VPC y los cambios en las direcciones IP públicas.

Estas integraciones le ayudan a adaptarse a los cambios en la AWS infraestructura y up-to-date a gestionar sus recursos de forma eficaz.

Origen	Beneficio del uso de Amazon SNS
AWS Actualizaciones diarias de funciones	Reciba puntualmente información detallada sobre lanzamientos y actualizaciones de AWS a través de un tema de Amazon SNS. Estas versiones incluyen Regiones de AWS puntos de enlace de Amazon VPC Servicios de AWS integrados con AWS Service Quotas, tipos de instancias de Amazon, tipos de EC2 instancia s de Amazon SageMaker AI, tipos de instancia s de Amazon Nimble Studio, versiones del motor de base de datos de Amazon RDS y versiones de Apache Kafka de Amazon MSK. Servicios de AWS Para obtener más informaci ón, consulte Suscríbase a las actualizaciones AWS diarias de funciones a través de Amazon SNS en el blog de AWS noticias.
AWS Intervalos de direcciones IP	Reciba notificaciones de cambios en los rangos de AWS IP a través de un tema de Amazon SNS. Para obtener más información, consulte las notificaciones de rangos de direccion es AWS IP en y Suscríbase a los cambios de direcciones IP AWS públicas a través de Amazon SNS en el blog de AWS noticias. Referencia general de Amazon Web Services

Para obtener más información acerca de la informática basada en eventos, consulte las siguientes fuentes:

- ¿Qué es una arquitectura basada en eventos?
- Informática basada en eventos con Amazon SNS AWS y servicios de cómputo, almacenamiento, bases de datos y redes en el blog Compute AWS

- Cómo enriquecer las arquitecturas basadas en eventos con AWS Event Fork Pipelines en el blog de Compute AWS

Destinos de eventos de Amazon SNS

En este tema se enumeran todos los destinos de los eventos, organizados por mensajería application-to-application (A2A) y notificaciones (A2P). application-to-person

> **Note**
>
> Amazon SNS presentó temas FIFO en octubre de 2020. Actualmente, la mayoría de los AWS servicios solo admiten la recepción de eventos de temas estándar de SNS. Amazon SQS admite la recepción de eventos de temas estándar y FIFO de SNS.

Destinos de A2A

En la siguiente tabla se describe cómo Amazon SNS puede entregar eventos a varios destinos application-to-application (A2A), como Amazon Data Firehose, Lambda, Amazon SQS, Event Fork Pipelines y puntos de enlace HTTP/S. AWS

Estas integraciones le permiten archivar y analizar datos, activar una lógica empresarial personalizada, facilitar la integración de las aplicaciones y dirigir los eventos a webhooks externos, lo que mejora la eficiencia y la flexibilidad de las arquitecturas basadas en eventos.

Destino de eventos	Beneficio del uso de Amazon SNS
Amazon Data Firehose	Entregue eventos a flujos de entrega con fines de archivado y análisis. A través de las transmisiones de entrega, puede entregar eventos a AWS destinos como Amazon Simple Storage Service (Amazon S3), Amazon Redshift y OpenSearch Amazon OpenSearc h Service (Service), o a destinos de terceros, como Datadog, New Relic, MongoDB y Splunk. Para obtener más información, consulte

Destino de eventos	Beneficio del uso de Amazon SNS
	Distribución ramificada a los flujos de entrega de Firehose.
AWS Lambda	Entregue eventos a funciones para desencadenar la ejecución de la lógica empresarial personalizada. Para obtener más información, consulte Distribución ramificada de las notificaciones de Amazon SNS a las funciones de Lambda para su procesamiento automatizado.
Amazon SQS	Entregue eventos a colas con fines de integración de aplicaciones. Para obtener más información, consulte Distribución ramificada de notificaciones de Amazon SNS a colas de Amazon SQS para su procesamiento asíncrono.
AWS Canalizaciones de Event Fork	Entregue eventos a copias de seguridad y almacenamiento de eventos, búsqueda y análisis de eventos, o canalizaciones de repetición de eventos. Para obtener más información, consulte Distribución ramificada de eventos de Amazon SNS a AWS Event Fork Pipelines.
HTTP/S	Entregue eventos a webhooks externos. Para obtener más información, consulte Distribución ramificada de notificaciones de Amazon SNS a puntos de conexión HTTPS.

Destinos A2P

En la siguiente tabla se describe cómo Amazon SNS envía notificaciones application-to-person (A2P) a varios destinos, incluidos teléfonos móviles mediante SMS y notificaciones push nativas, bandejas

de entrada de correo electrónico, salas de chat de Amazon Chime, canales de Slack e información operativa a los equipos de guardia mediante. PagerDuty

Estas integraciones mejoran la comunicación y la eficiencia operativa al permitir alertas y actualizaciones en tiempo real en múltiples plataformas y canales de comunicación.

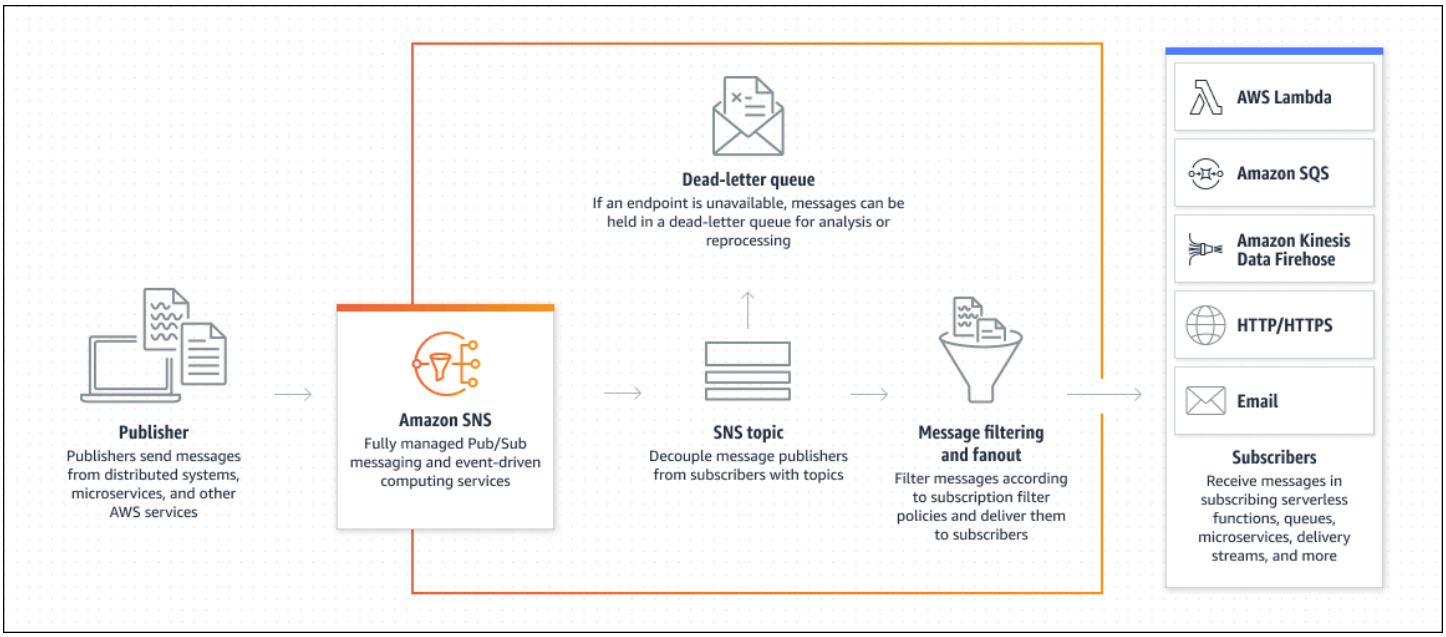
Destino de eventos	Beneficio del uso de Amazon SNS
SMS	Entregue eventos a teléfonos móviles como mensajes de texto. Para obtener más información, consulte Mensajería de texto móvil con Amazon SNS.
Correo electrónico	Entregue eventos a las bandejas de entrada como mensajes de correo electrónico. Para obtener más información, consulte Configuración y administración de suscripciones de correo electrónico de Amazon SNS.
Punto de enlace de plataforma	Entregue eventos a teléfonos móviles como notificaciones push nativas. Para obtener más información, consulte Envío de notificaciones push para móvil con Amazon SNS.
Amazon Q Developer en aplicaciones de chat	Entregue eventos a las salas de chat de Amazon Chime o a los canales de Slack. Para obtener más información, consulta las siguientes páginas de la Guía del administrador de aplicaciones de chat para desarrolladores de Amazon Q: • Configuración de Amazon Q Developer en aplicaciones de chat con Amazon Chime • Configuración de Amazon Q Developer en aplicaciones de chat con Slack •

Destino de eventos	Beneficio del uso de Amazon SNS
	Uso de Amazon Q Developer en aplicaciones de chat con otros AWS servicios
PagerDuty	Proporcione información operativa a los equipos de guardia. Para obtener más información, consulte Entregue información operativa basada en ML a sus equipos de guardia a través de PagerDuty Amazon DevOps Guru en el blog AWS Management & Governance.

> **Note**
>
> Puede ofrecer AWS eventos nativos y eventos personalizados a las aplicaciones de chat:
>
> - AWS Eventos nativos: puede utilizar Amazon Q Developer en aplicaciones de chat para enviar AWS eventos nativos, a través de temas de Amazon SNS, a Amazon Chime y Slack. El conjunto de AWS eventos nativos compatibles incluye eventos de Administración de facturación y costos de AWS AWS Health, AWS CloudFormation CloudWatch, Amazon y más. Para obtener más información, consulte Uso de Amazon Q Developer en aplicaciones de chat con otros servicios en la Guía del administrador de Amazon Q Developer en aplicaciones de chat.
> - Eventos personalizados: también puede enviar sus eventos personalizados, a través de temas de Amazon SNS, a Amazon Chime, Slack y Microsoft Teams. Para ello, publica eventos personalizados en un tema de SNS, que entrega los eventos a una función Lambda suscrita. La función Lambda utiliza el webhook de la aplicación de conversación para entregar los eventos a los destinatarios. Para obtener más información, consulte ¿Cómo uso los webhooks para publicar mensajes de Amazon SNS en Amazon Chime, Slack o Microsoft Teams?

Uso de Amazon SNS para la mensajería application-to-application

Amazon SNS simplifica la mensajería application-to-application (A2A) al separar a los editores de los suscriptores, lo que admite microservicios, sistemas distribuidos y aplicaciones sin servidor. Los mensajes se envían a los temas de Amazon SNS, donde se pueden filtrar y entregar a suscriptores como Lambda, Amazon SQS o puntos de conexión HTTP. Si se produce un error en la entrega, los mensajes se almacenan en una cola de mensajes fallidos para su posterior análisis o para volverlos a procesar.



Distribución ramificada a los flujos de entrega de Firehose

Puede suscribir las transmisiones de entrega de Amazon Data Firehose a los temas de Amazon SNS, lo que le permite enviar notificaciones a puntos de enlace de almacenamiento y análisis adicionales. Los mensajes publicados en un tema de Amazon SNS se envían al flujo de entrega de Firehose suscrito y se entregan al destino tal y como están configurados en Firehose. El propietario de una suscripción puede suscribir hasta cinco flujos de entrega de Firehose a un tema de Amazon SNS. Cada flujo de entrega de Firehose tiene una cuota predeterminada para solicitudes y rendimiento por segundo. Este límite podría dar lugar a más mensajes publicados (tráfico entrante) que entregados (tráfico saliente). Cuando hay más tráfico entrante que saliente, la suscripción puede acumular una gran cantidad de mensajes atrasados, lo que provoca una latencia de entrega de

mensajes elevada. Puede solicitar un aumento de cuota en función de la tasa de publicación para evitar un impacto adverso en la carga de trabajo.

A través de las transmisiones de entrega de Firehose, puede distribuir las notificaciones de Amazon SNS a Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Service (OpenSearch Service) y a proveedores de servicios de terceros, OpenSearch como Datadog, New Relic, MongoDB y Splunk.

Por ejemplo, puede utilizar esta funcionalidad para almacenar de forma permanente los mensajes enviados a un tema en un bucket de Amazon S3 con fines de conformidad, archivado u otros. Para ello, cree una transmisión de entrega de Firehose con un destino de bucket de Amazon S3 y suscriba esa transmisión de entrega al tema Amazon SNS. Como otro ejemplo, para analizar los mensajes enviados a un tema de Amazon SNS, cree una transmisión de entrega con un destino de índice de OpenSearch servicios. A continuación, puede suscribir el flujo de entrega de Firehose al tema de Amazon SNS.

Amazon SNS también admite el registro de estado de entrega de mensajes para las notificaciones enviadas a los puntos de conexión de Firehose. Para obtener más información, consulte Estado de entrega de mensajes de Amazon SNS.

Requisitos previos para suscribir flujos de entrega de Firehose a temas de Amazon SNS

Para suscribir una transmisión de entrega de Amazon Data Firehose a un tema de SNS, debe tener: Cuenta de AWS

- Un tema de SNS estándar. Para obtener más información, consulte Creación de un tema de Amazon SNS.
- Un flujo de entrega de Firehose. Para obtener más información, consulte Creating an Amazon Data Firehose Delivery Stream y Grant Your Application Access to Your Firehose Resources en la Guía para desarrolladores de Amazon Data Firehose.
- Un rol AWS Identity and Access Management (IAM) que confía en el director del servicio Amazon SNS y tiene permiso para escribir en el flujo de entrega. Ingresará el nombre de recurso de Amazon (ARN) de este rol como `SubscriptionRoleARN` cuando cree la suscripción. Amazon SNS asume este rol y, gracias a esto, Amazon SNS puede colocar registros en el flujo de entrega de Firehose.

  En el siguiente ejemplo de política, se muestran los permisos recomendados:

```
{
  "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 conceder todos los permisos necesarios para usar Firehose, también puedes usar la política AWS gestionada. `AmazonKinesisFirehoseFullAccess` O bien, para proporcionar permisos más estrictos para usar Firehose, puede crear su propia política. Como mínimo, en la política se debe proporcionar permiso para ejecutar la operación `PutRecord` en un flujo de entrega específico.

En todos los casos, también debe editar la relación de confianza para incluir el principal de servicio de Amazon SNS. Por ejemplo:

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

```
}
```

Para obtener más información sobre la creación de funciones, consulte Crear una función para delegar permisos a un AWS servicio en la Guía del usuario de IAM.

Una vez que haya completado estos requisitos, puede suscribir el flujo de entrega al tema de SNS.

Suscripción de un flujo de entrega de Firehose a un tema de Amazon SNS

Para enviar notificaciones de Amazon SNS a los flujos de entrega de Amazon Data Firehose, primero asegúrese de haber cumplido todos los requisitos previos. Para obtener una lista de los puntos de conexión compatibles, consulte Puntos de conexión y cuotas de Amazon Data Firehose en la Referencia general de Amazon Web Services.

Suscripción de un flujo de entrega de Firehose a un tema

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, seleccione Suscripciones.
3. En la página Subscriptions (Suscripciones), elija Create subscription (Crear suscripción).
4. En la página Crear suscripción, en la sección Detalles, haga lo siguiente:
   a. En ARN de tema, elija el nombre de recurso de Amazon (ARN) de un tema estándar.
   b. En Protocolo, elija Firehose.
   c. En Punto de conexión, elija el ARN de un flujo de entrega de Firehose en el que se puedan recibir notificaciones de Amazon SNS.
   d. En ARN del rol de suscripción, especifique el ARN rol de AWS Identity and Access Management (IAM) que creó para escribir en flujos de entrega de Firehose. Para obtener más información, consulte Requisitos previos para suscribir flujos de entrega de Firehose a temas de Amazon SNS.
   e. (Opcional) Para eliminar cualquier metadato de Amazon SNS de los mensajes publicados, elija Habilitar la entrega de mensajes. Para obtener más información, consulte Entrega de mensajes sin procesar de Amazon SNS.
5. (Opcional) Para configurar una política de filtro, expanda la sección Política de filtro de suscripción. Para obtener más información, consulte Políticas de filtro de suscripciones de Amazon SNS.

6. (Opcional) Para configurar una cola de mensajes fallidos en la suscripción, expanda la sección Política de reconducción (cola de mensajes fallidos). Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS.
7. Elija Crear una suscripción.

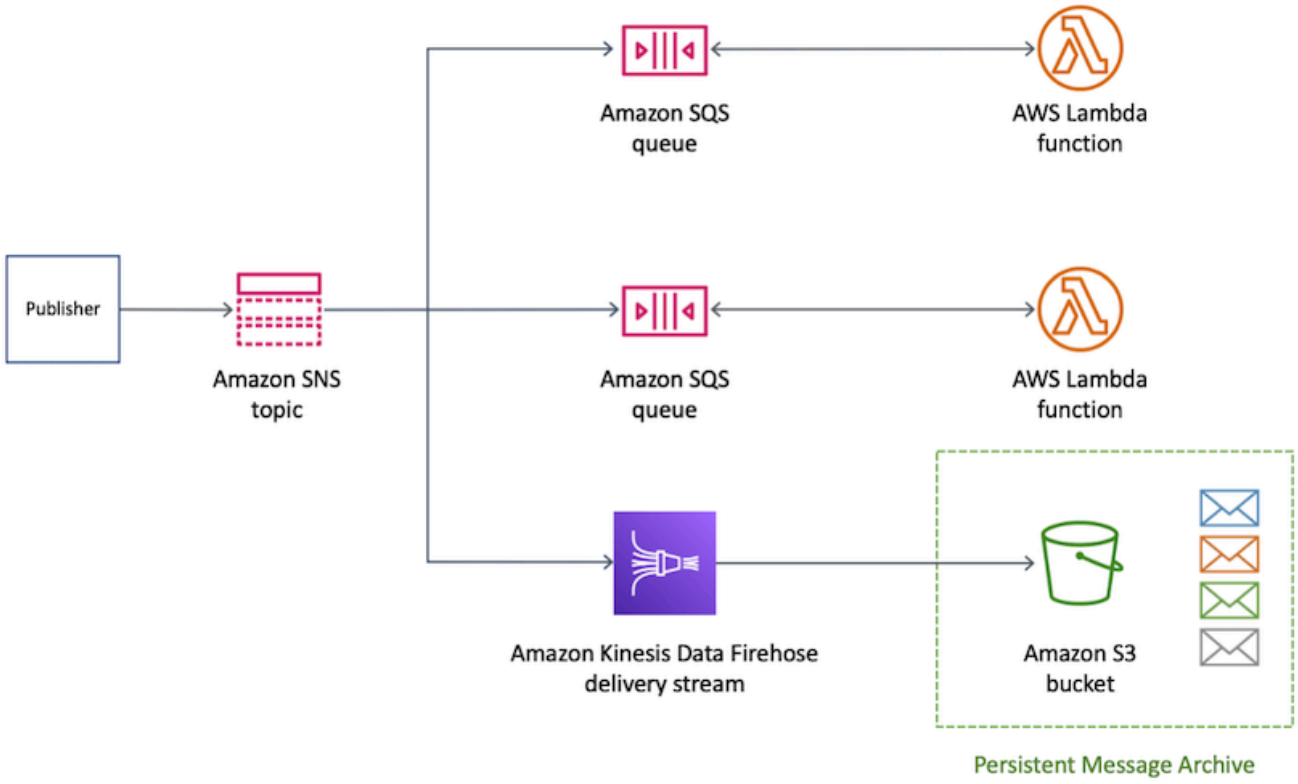
En la consola se crea la suscripción y se abre la página Detalles de la suscripción.

Administración de mensajes de Amazon SNS en varios destinos de flujo de entrega

Los flujos de entrega de Amazon Data Firehose le permiten gestionar los mensajes de Amazon SNS en varios destinos, lo que permite la integración con Amazon S3, Amazon Service, OpenSearch Amazon Redshift y los puntos de enlace HTTP para el almacenamiento, la indexación y el análisis. Al configurar correctamente el formato y la entrega de los mensajes, puede almacenar las notificaciones de Amazon SNS en Amazon S3 para su posterior procesamiento, analizar los datos estructurados de los mensajes con Amazon Athena, indexar los mensajes para buscarlos y visualizarlos OpenSearch en tiempo real y estructurar los archivos en Amazon Redshift para realizar consultas avanzadas.

Almacenamiento y análisis de mensajes de Amazon SNS en destinos de Amazon S3

En este tema se explica cómo las transmisiones de entrega de Amazon Data Firehose publican datos en Amazon Simple Storage Service (Amazon S3).



Temas

- Formato de las notificaciones de Amazon SNS para su almacenamiento en destinos de Amazon S3
- Análisis de los mensajes de Amazon SNS almacenados en Amazon S3 con Athena

Formato de las notificaciones de Amazon SNS para su almacenamiento en destinos de Amazon S3

El siguiente ejemplo muestra una notificación de Amazon SNS enviada a un bucket de Amazon Simple Storage Service (Amazon S3), con una sangría para facilitar la lectura.

> **Note**
>
> En este ejemplo, se desactivó la entrega de mensajes sin formato en el mensaje publicado. Si se desactiva la entrega de mensajes sin formato, Amazon SNS agrega metadatos JSON al mensaje, incluidas las siguientes propiedades:
>
> - `Type`
> - `MessageId`

- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

Para obtener más información acerca de la entrega sin procesar, consulte Entrega de mensajes sin procesar de 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"
        }
    }
 }
```

En el siguiente ejemplo, se muestran tres mensajes de SNS enviados a través de un flujo de entrega de Amazon Data Firehose al mismo bucket de Amazon S3. Se aplica el almacenamiento en búfer y los saltos de línea separan cada mensaje.

```
{"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"}
```


Análisis de los mensajes de Amazon SNS almacenados en Amazon S3 con Athena

En esta página se explica cómo analizar los mensajes de Amazon SNS que se envían a través de las transmisiones de entrega de Amazon Data Firehose a los destinos del Amazon Simple Storage Service (Amazon S3).

Análisis de mensajes SNS enviados a través de flujos de entrega de Firehose a destinos de Amazon S3

1. Configure sus recursos de Amazon S3. Para recibir instrucciones, consulte Creación de un bucket en la Guía del usuario de Amazon Simple Storage Service y Trabajar con buckets de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.
2. Configure el flujo de entrega. Para obtener más información, consulte Choose Amazon S3 for Your Destination en la Guía para desarrolladores de Amazon Data Firehose.
3. Utilice Amazon Athena para consultar los objetos de Amazon S3 mediante SQL estándar. Para obtener más información, consulte Introducción en la Guía del usuario de Amazon Athena.

Consulta de ejemplo

En esta consulta, suponga lo siguiente:

- Los mensajes se almacenan en la tabla `notifications` del esquema `default`.

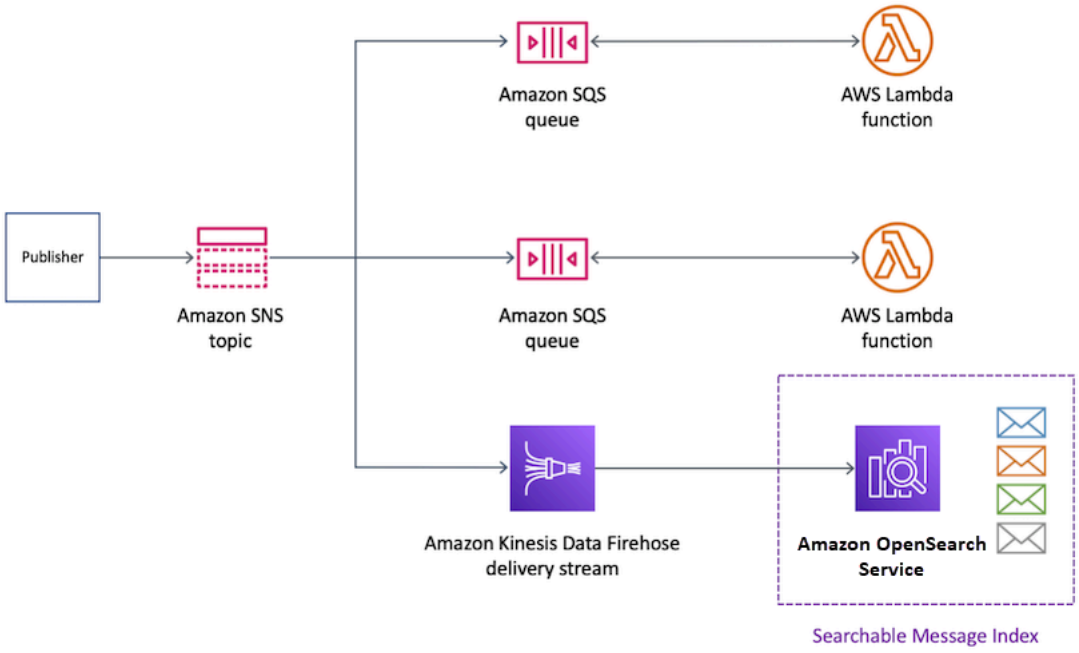
- En la tabla `notifications`, se incluye una columna `timestamp` con un tipo de `string`.

Con la siguiente consulta, se devuelven todos los mensajes SNS recibidos en el intervalo de fechas 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';
```

Integración de los mensajes de Amazon SNS con los destinos de Amazon Service OpenSearch

En esta sección se explica cómo las transmisiones de entrega de Amazon Data Firehose publican los datos en Amazon OpenSearch Service (OpenSearch Servicio).



Temas

- Almacenamiento y formato de las notificaciones de Amazon SNS en OpenSearch índices de servicios
- Análisis de los mensajes de Amazon SNS para OpenSearch los destinos del servicio

Almacenamiento y formato de las notificaciones de Amazon SNS en OpenSearch índices de servicios

En el siguiente ejemplo se muestra una notificación de Amazon SNS enviada a un índice de Amazon OpenSearch Service (OpenSearch Service) llamado. `my-index` Este índice tiene un campo de filtro de tiempo en el campo `Timestamp`. La notificación SNS se coloca en la propiedad `_source` de la carga.

> **Note**
>
> En este ejemplo, se desactivó la entrega de mensajes sin formato en el mensaje publicado. Si se desactiva la entrega de mensajes sin formato, Amazon SNS agrega metadatos JSON al mensaje, incluidas las siguientes propiedades:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Para obtener más información acerca de la entrega sin procesar, consulte Entrega de mensajes sin procesar de 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
  ]
}
```

Análisis de los mensajes de Amazon SNS para OpenSearch los destinos del servicio

En este tema se explica cómo analizar los mensajes de Amazon SNS enviados a través de las transmisiones de entrega de Amazon Data Firehose a los destinos de Amazon OpenSearch Service (Service)OpenSearch .

Para analizar los mensajes SNS enviados a través de los flujos OpenSearch de entrega de Firehose a los destinos del servicio

1. Configure sus recursos de OpenSearch servicio. Para obtener instrucciones, consulta Cómo empezar a usar Amazon OpenSearch Service en la Guía para desarrolladores OpenSearch de Amazon Service.
2. Configure el flujo de entrega. Para obtener instrucciones, consulte Choose OpenSearch Service for Your Destination en la Guía para desarrolladores de Amazon Data Firehose.
3. Ejecute una consulta con OpenSearch Service Queries y Kibana. Para obtener más información, consulta el Paso 3: Buscar documentos en un dominio de OpenSearch servicio y Kibana en la Guía para desarrolladores de Amazon OpenSearch Service.

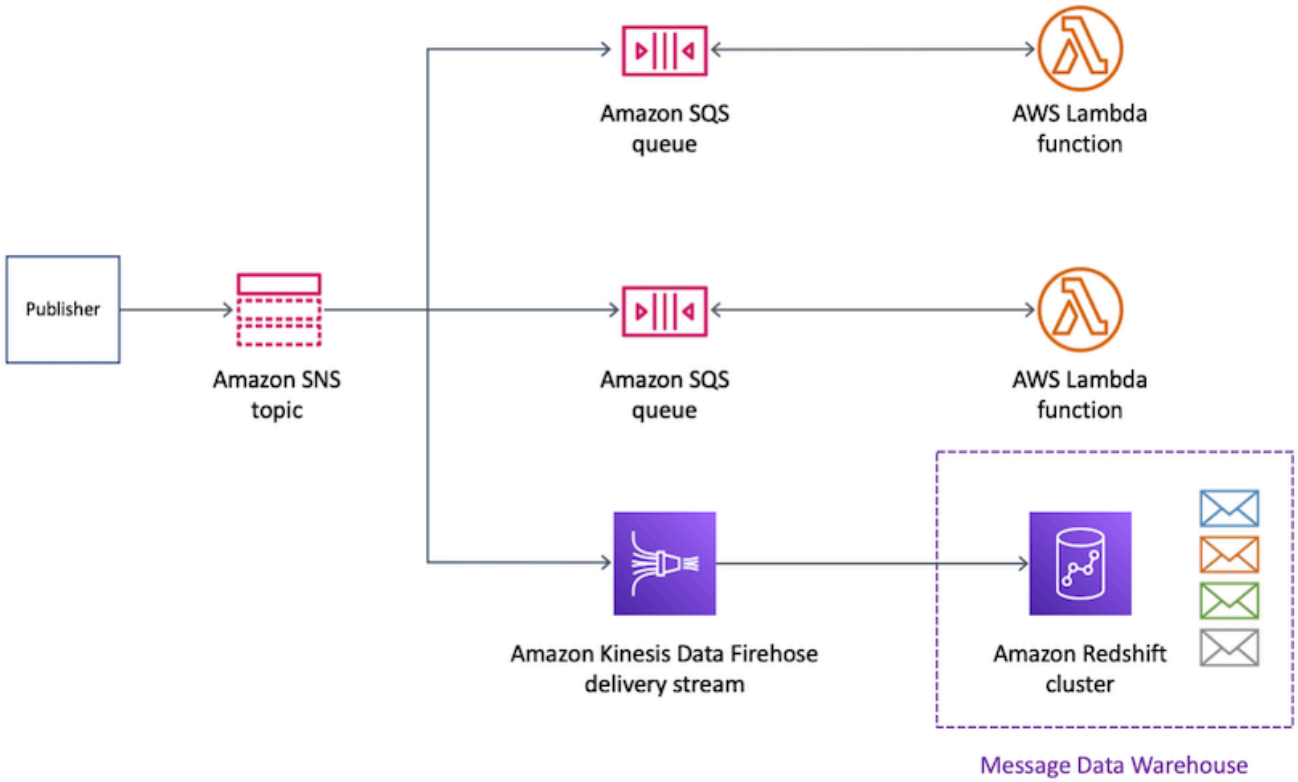
Consulta de ejemplo

En el siguiente ejemplo, se consulta el índice `my-index` de todos los mensajes SNS recibidos en el intervalo de fechas 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"
            }
          }
        }
      ]
    }
  }
}
```

Configuración de la entrega y el análisis de mensajes de Amazon SNS en destinos de Amazon Redshift

En este tema se explica cómo distribuir las notificaciones de Amazon SNS a una transmisión de entrega de Amazon Data Firehose, que luego publica los datos en Amazon Redshift. Con esta configuración, puede conectarse a la base de datos de Amazon Redshift y utilizar una herramienta de consulta SQL para recuperar los mensajes de Amazon SNS que coincidan con criterios específicos.



Temas

- Estructuración de los archivos de mensajes de Amazon SNS en tablas de Amazon Redshift
- Análisis de los mensajes de Amazon SNS almacenados en destinos de Amazon Redshift

Estructuración de los archivos de mensajes de Amazon SNS en tablas de Amazon Redshift

En el caso de los puntos de enlace de Amazon Redshift, los mensajes de Amazon SNS se archivan como filas de una tabla. A continuación, se muestra un ejemplo de cómo se almacenan los datos:

> **Note**
>
> En este ejemplo, se desactivó la entrega de mensajes sin formato en el mensaje publicado. Si se desactiva la entrega de mensajes sin formato, Amazon SNS agrega metadatos JSON al mensaje, incluidas las siguientes propiedades:
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- `Subject`
- `Message`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Para obtener más información acerca de la entrega sin procesar, consulte Entrega de mensajes sin procesar de Amazon SNS.
Si bien Amazon SNS agrega propiedades al mensaje con las mayúsculas que se muestran en esta lista, los nombres de columna de las tablas de Amazon Redshift aparecen en minúsculas. Para transformar los metadatos JSON en el punto de enlace de Amazon Redshift, puede utilizar el comando `COPY` SQL. Para obtener más información, consulte los ejemplos Copiar desde JSON y Cargar desde datos JSON con la opción 'auto ignorecase' en la Guía para desarrolladores de bases de datos de Amazon Redshift.

type	messageId	topicarn	subject	message	marca de tiempo	unsubscribeurl	messageattributes
Notificación	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Asunto de muestra	Mensaje de muestra	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Darse de baja y=arn:aws:sns:us-east-1:1111:my-topic:326deeeb-cb	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageId	topicarn	subject	message	marca de tiempo	unsubscribeurl	messageattributes
						f4-45da-b92b-ca77a247813b SubscriptionArn	
Notificación	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Asunto de muestra 2	Mensaje de muestra 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Darse de baja y=arn:aws:sns:us-east-1:1111:my-topic:326deeeb-cb f4-45da-b92b-ca77a247813b SubscriptionArn	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageId	topicarn	subject	message	marca de tiempo	unsubscribeurl	messageattributes
Notificación	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Asunto de muestra 3	Mensaje de muestra 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Darse de baja y=arn:aws:sns:us-east-1:1111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813bSubscriptionArn	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Para obtener más información sobre la distribución de notificaciones a los puntos de enlace de Amazon Redshift, consulte Configuración de la entrega y el análisis de mensajes de Amazon SNS en destinos de Amazon Redshift.

Análisis de los mensajes de Amazon SNS almacenados en destinos de Amazon Redshift

En este tema se describe cómo analizar los mensajes de Amazon SNS que se envían a través de las transmisiones de entrega de Amazon Data Firehose a los destinos de Amazon Redshift.

Análisis de los mensajes SNS enviados a través de flujos de entrega de a destinos de Amazon Redshift

1. Configure sus recursos de Amazon Redshift. Para obtener instrucciones, consulte Introducción a Amazon Redshift en la Guía de introducción a Amazon Redshift.
2. Configure el flujo de entrega. Para obtener más información, consulte Choose Amazon Redshift for Your Destination en la Guía para desarrolladores de Amazon Data Firehose.
3. Ejecute una consulta. Para obtener más información, consulte Consulta de una base de datos mediante el editor de consultas en la Guía de administración de Amazon Redshift.

Consulta de ejemplo

En esta consulta, suponga lo siguiente:

- Los mensajes se almacenan en la tabla `notifications` del esquema `public` predeterminado.
- La propiedad `Timestamp` del mensaje SNS se almacena en la columna `timestamp` de la tabla con un tipo de datos de columna de `timestamptz`.

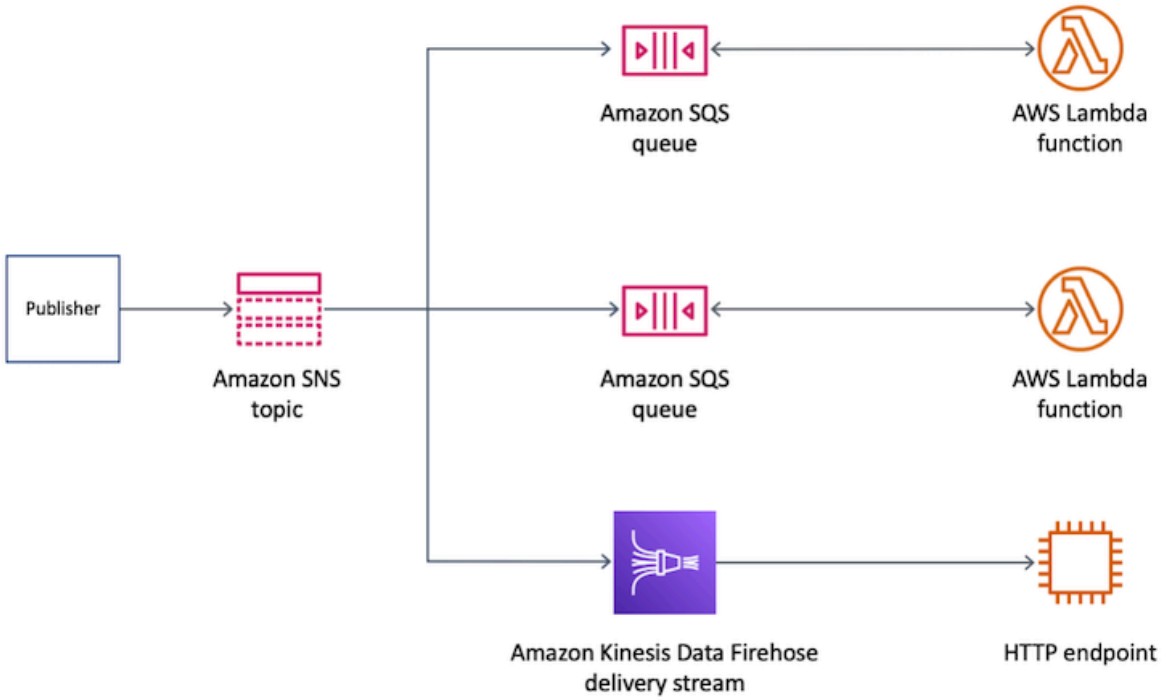
> **Note**
>
> Para transformar los metadatos JSON en el punto de enlace de Amazon Redshift, puede utilizar el comando `COPY` SQL. Para obtener más información, consulte los ejemplos Copiar desde JSON y Cargar desde datos JSON con la opción 'auto ignorecase' en la Guía para desarrolladores de bases de datos de Amazon Redshift.

Con la siguiente consulta, se devuelven todos los mensajes SNS recibidos en el intervalo de fechas especificado:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

Configuración de la entrega de mensajes de Amazon SNS a destinos HTTP mediante Amazon Data Firehose

En este tema se explica cómo las transmisiones de entrega de Amazon Data Firehose publican datos en puntos de enlace HTTP.



Temas

- Formato de notificaciones de Amazon SNS para su entrega a destinos HTTP

Formato de notificaciones de Amazon SNS para su entrega a destinos HTTP

A continuación, se muestra un ejemplo del cuerpo de una solicitud HTTP POST de Amazon SNS, enviada a través de una transmisión de entrega de Amazon Data Firehose a un punto final HTTP. La notificación de Amazon SNS está codificada como una carga útil en base64 dentro de la propiedad records.

> **Note**
>
> En este ejemplo, se desactivó la entrega de mensajes sin formato en el mensaje publicado. Para obtener más información acerca de la entrega sin procesar, consulte Entrega de mensajes sin procesar de Amazon SNS.


```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
        "data":
 "eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```

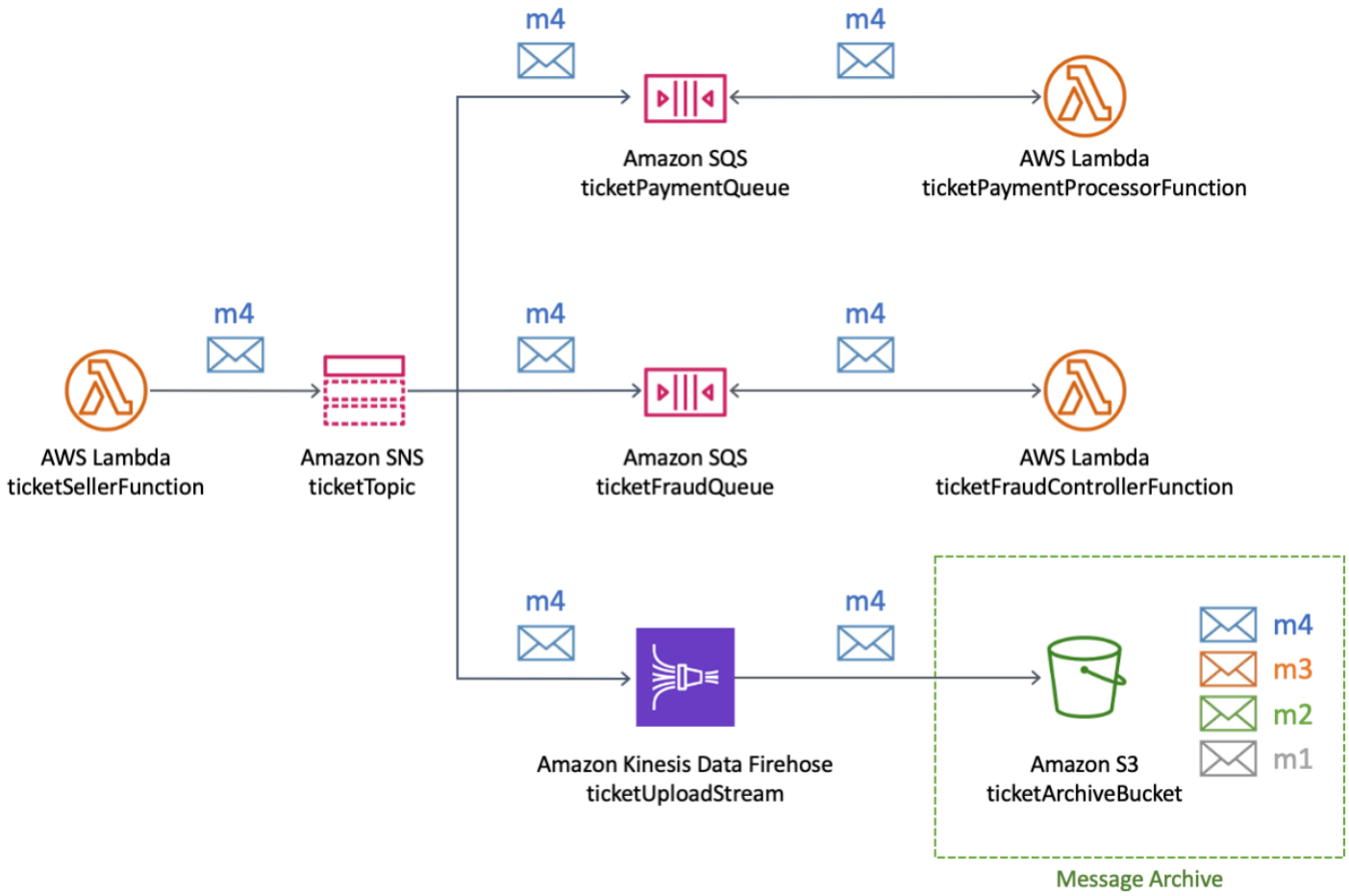

Archivo y análisis de mensajes de Amazon SNS: un caso de uso de ejemplo para las plataformas de venta de billetes de avión

En este tema se proporciona un tutorial para un caso de uso común de archivado y análisis de mensajes de Amazon SNS.

La configuración de este caso de uso es una plataforma de emisión de billetes de avión que opera en un entorno regulado.

1. La plataforma está sujeta a un marco de conformidad que exige que la empresa archive todas las ventas de entradas durante un mínimo de cinco años.
2. Para cumplir el objetivo de cumplimiento en materia de retención de datos, la empresa suscribe una transmisión de entrega de Amazon Data Firehose a un tema existente de Amazon SNS.
3. El destino del flujo de entrega es un bucket de Amazon Simple Storage Service (Amazon S3). Con esta configuración, todos los eventos publicados en el tema de SNS se archivan en el bucket de Amazon S3.

En el siguiente diagrama, se ilustra la arquitectura de esta configuración.



Para ejecutar análisis y obtener información sobre la venta de entradas, la empresa ejecuta consultas SQL con Amazon Athena. Por ejemplo, la empresa puede consultar para conocer los destinos más populares y los viajeros más frecuentes.

Para crear los AWS recursos para este caso de uso, puede utilizar la plantilla AWS Management Console o una plantilla. AWS CloudFormation

Temas

- Configuración de los recursos de AWS iniciales para el archivo y análisis de mensajes de Amazon SNS
- Configuración de un flujo de Firehose para el archivo de mensajes de Amazon SNS
- Suscripción del flujo de entrega de Firehose al tema de Amazon SNS
- Pruebas y consulta de una configuración de Amazon SNS para una administración de datos eficaz
- Automatización del archivo de mensajes de Amazon SNS con una plantilla de AWS CloudFormation

Configuración de los recursos de AWS iniciales para el archivo y análisis de mensajes de Amazon SNS

En este tema se describe cómo crear los recursos necesarios para el caso práctico de archivado y análisis de mensajes:

- Un bucket de Amazon Simple Storage Service (Amazon S3)
- Dos colas de Amazon Simple Queue Service (Amazon SQS)
- Un tema de Amazon SNS
- Dos suscripciones de Amazon SQS al tema de Amazon SNS

Para crear los recursos iniciales, siga estos pasos:

1. Cree un bucket de Amazon S3:

   a. Abra la consola de Amazon S3.

   b. Elija Crear bucket.

   c. En Nombre del bucket, ingrese un nombre único. Mantenga los otros campos como valores predeterminados.

   d. Elija Crear bucket.

   Para obtener más información sobre los buckets de Amazon S3, consulte Creación de un bucket en la Guía del usuario de Amazon Simple Storage Service y Trabajar con buckets de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

2. Cree las dos colas de Amazon SQS:

   a. Abra la consola de Amazon SQS.

   b. Elige Crear cola.

   c. En Tipo, seleccione Estándar.

   d. En Nombre, escriba **`ticketPaymentQueue`**.

   e. En Política de acceso, en Elegir método, elija Avanzado.

   f. En el cuadro Política de JSON, pegue la siguiente 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"
        }
      }
    }
  ]
}
```

En esta política de acceso, sustituya el Cuenta de AWS número (*123456789012*) por el suyo propio y cambie la AWS región (*us-east-1*) en consecuencia.

g. Elige Crear cola.

h. Repita estos pasos para crear una segunda cola SQS llamada **ticketFraudQueue**.

Para obtener más información sobre la creación de colas SQS, consulte Creación de una cola de Amazon SQS (consola) en la Guía para desarrolladores de Amazon Simple Queue Service.

3. Cree el tema de SNS:

a. Abra la página Topics (Temas) en la consola de Amazon SNS.

b. Seleccione Crear tema.

c. En Detalles, en Tipo, elija Estándar.

d. En Nombre, escriba **ticketTopic**.

e. Seleccione Crear tema.

Para obtener más información sobre la creación de temas de SNS, consulte Creación de un tema de Amazon SNS.

4. Suscriba las colas de SQS al tema de SNS:

a. En la consola de Amazon SNS, en la página de detalles del tema TicketTopic, elija Crear suscripción.

b. En Detalles, en Protocolo, elija Amazon SQS.

c. Para Endpoint, elija el nombre de recurso de Amazon (ARN) de la ticketPaymentQueuecola.

d. Seleccione Crear subscripción.

e. Repita estos pasos para crear una segunda suscripción con el ARN de la ticketFraudQueuecola.

Para obtener más información sobre la suscripción a los temas de SNS, consulte Creación de una suscripción a un tema de Amazon SNS. También puede suscribir colas de SQS a temas de SNS desde la consola de Amazon SQS. Para obtener más información, consulte Suscripción de una cola de Amazon SQS a un tema de Amazon SNS (consola) en la Guía para desarrolladores de Amazon Simple Queue Service.

Ha creado los recursos iniciales para este caso de uso de ejemplo. Para continuar, consulte Configuración de un flujo de Firehose para el archivo de mensajes de Amazon SNS.

Configuración de un flujo de Firehose para el archivo de mensajes de Amazon SNS

En este tema se explica cómo crear el flujo de entrega de Amazon Data Firehose para el caso práctico de análisis y archivado de mensajes.

Creación de un flujo de entrega de Firehose

1. Abra la consola de servicios de Amazon Kinesis.
2. Seleccione Firehose y, a continuación, elija Crear un flujo de entrega.
3. En la página Nuevo flujo de entrega, en Nombre del flujo de entrega, ingrese **`ticketUploadStream`** y, a continuación, elija Siguiente.
4. En la página Procesar registros, elija Siguiente.
5. En la página Elegir un destino, haga lo siguiente:

   a. En Destino, elija Amazon S3.

   b. En Destino S3, en bucket de S3, elija el bucket de S3 que creó en un principio.

   c. Elija Next (Siguiente).

6. En la página Ajustar configuración, en Condiciones del búfer S3, realice una de las siguientes operaciones:

   - En Tamaño del búfer, ingrese **1**.
   - En Intervalo del búfer, ingrese **60**.

   Con el uso de estos valores para el búfer de Amazon S3, puede probar con rapidez la configuración. La primera condición que se cumple desencadena la entrega de datos al bucket de S3.
7. En la página Configurar ajustes, en Permisos, elija crear un rol AWS Identity and Access Management (IAM) con los permisos necesarios asignados automáticamente. A continuación, elija Siguiente.
8. En la página Revisar, elija Crear un flujo de entrega.
9. En la página de transmisiones de entrega de Kinesis Data Firehose, elija la transmisión de entrega que acaba ticketUploadStreamde crear (). En la pestaña Detalles, anote el nombre de recurso de Amazon (ARN) del flujo para después.

Para obtener más información sobre la creación de flujos de entrega, consulte Creación de un flujo de entrega de Amazon Data Firehose en la Guía para desarrolladores de Amazon Data Firehose. Para obtener más información sobre la creación de funciones de IAM, consulte Crear una función para delegar permisos a un AWS servicio en la Guía del usuario de IAM.

Ha creado el flujo de entrega de Firehose con los permisos necesarios. Para continuar, consulte Suscripción del flujo de entrega de Firehose al tema de Amazon SNS.

Suscripción del flujo de entrega de Firehose al tema de Amazon SNS

En este tema se explica cómo crear los siguientes recursos para el ejemplo de uso del archivado y el análisis de mensajes:

- La función AWS Identity and Access Management (IAM) que permite a la suscripción a Amazon SNS colocar registros en el flujo de entrega de Amazon Data Firehose.
- La suscripción a la transmisión de entrega de Firehose al tema Amazon SNS.

Con el fin de crear el rol de IAM para la suscripción a Amazon SNS, siga estos pasos:

1. Abra la página Roles en la consola de IAM.

2. Elija Create role.
3. En Seleccionar el tipo de entidad de confianza, elija Servicio de AWS .
4. En Elegir un caso de uso, elija SNS. A continuación, elija Siguiente: permisos.
5. Elija Siguiente: Etiquetas.
6. Elija Siguiente: Revisar.
7. En la página Revisión, en Nombre del rol, ingrese **`ticketUploadStreamSubscriptionRole`**. A continuación, elija Crear rol.
8. Cuando se cree el rol, elija su nombre () ticketUploadStreamSubscriptionRole.
9. En la página Resumen, elija Agregar política en línea.
10. En la página Crear política, elija la pestaña JSON y, a continuación, pegue la siguiente política en el cuadro:

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

    En esta política, sustituya el Cuenta de AWS número (*123456789012*) por el suyo propio y cambie la AWS región (*us-east-1*) en consecuencia.
11. Elija Revisar política.
12. En la página Crear política, en Nombre, ingrese **`FirehoseSnsPolicy`**. A continuación, seleccione Create policy (Crear política).

13. En la página Resumen del rol, tenga en cuenta el ARN de rol para después.

Para obtener más información sobre la creación de funciones de IAM, consulte Creación de una función para delegar permisos a un AWS servicio en la Guía del usuario de IAM.

Suscripción del flujo de entrega de Firehose al tema de SNS

1. Abra la página Topics (Temas) en la consola de Amazon SNS.
2. En la pestaña Suscripciones, elija Crear suscripción.
3. En Detalles, en Protocolo, elija Amazon Data Firehose.
4. Para Endpoint, introduzca el nombre de recurso de Amazon (ARN) de la transmisión de ticketUploadStreamentrega que creó anteriormente. Por ejemplo, escriba **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. En el ARN del rol de suscripción, introduzca el ARN del rol de ticketUploadStreamSubscriptionRoleIAM que creó anteriormente. Por ejemplo, escriba **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Seleccione el cuadro de verificación Habilitar la entrega de mensajes sin procesar.
7. Elija Crear una suscripción.

Ha creado el rol de IAM y la suscripción al tema de SNS. Para continuar, consulte Pruebas y consulta de una configuración de Amazon SNS para una administración de datos eficaz.

Pruebas y consulta de una configuración de Amazon SNS para una administración de datos eficaz

En este tema se explica cómo probar un ejemplo de caso práctico de archivado y análisis de mensajes publicando un mensaje en el tema Amazon SNS. Entre las instrucciones se incluye una consulta de ejemplo que puede ejecutar y adaptar a sus propias necesidades.

Para probar la configuración

1. Abra la página Topics (Temas) en la consola de Amazon SNS.
2. Elija el tema **`ticketTopic`**.
3. Elija Publish message (Publicar mensaje).

4. En la página Publicar mensaje en tema, ingrese lo siguiente en el cuerpo del mensaje. Agregue un carácter de nueva línea al final del mensaje.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

   Mantenga todas las demás opciones en sus valores predeterminados.

5. Elija Publish message (Publicar mensaje).

   Para obtener más información sobre la publicación de mensajes, consulte Publicación de un mensaje de Amazon SNS.

6. Después del intervalo de flujo de entrega de 60 segundos, abra la consola de Amazon Simple Storage Service (Amazon S3) y elija el bucket de Amazon S3 que creó en un principio.

   El mensaje publicado aparece en el bucket.

Para consultar los datos, siga estos pasos:

1. Abra la consola de Amazon Athena.
2. Ejecute una consulta.

   Por ejemplo, supongamos que en la tabla `notifications` del esquema `default` se incluyen los siguientes datos:

```
{"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 buscar el destino principal, ejecute la siguiente consulta:

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
```

```
LIMIT 1;
```

Para consultar los tickets vendidos durante un intervalo de fecha y hora específico, ejecute una consulta como la siguiente:

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

Puede adaptar ambas consultas de muestra según sus propias necesidades. Si desea obtener más información sobre el uso de Athena para ejecutar consultas, consulte Introducción en la Guía del usuario de Amazon Athena.

Limpieza

Para evitar incurrir en cargos de uso después de haber terminado la prueba, elimine los siguientes recursos que creó durante el tutorial:

- Suscripciones a Amazon SNS
- Tema de Amazon SNS
- Colas de Amazon Simple Queue Service (Amazon SQS)
- Bucket de Amazon S3
- Flujo de entrega de Amazon Data Firehose
- AWS Identity and Access Management Funciones y políticas (IAM)

Automatización del archivo de mensajes de Amazon SNS con una plantilla de AWS CloudFormation

Para automatizar la implementación de caso de uso de ejemplo de archivado y análisis de mensajes de Amazon SNS, puede usar la siguiente plantilla 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
```

Distribución ramificada de las notificaciones de Amazon SNS a las funciones de Lambda para su procesamiento automatizado

Amazon SNS se integra con AWS Lambda, lo que le permite activar funciones de Lambda en respuesta a las notificaciones de Amazon SNS. Cuando se publica un mensaje en un tema de SNS que tiene una función de Lambda suscrita, la función de Lambda se invoca con la carga útil del mensaje publicado. La función Lambda recibe la carga útil del mensaje como parámetro de entrada y puede manipular la información del mensaje, publicar el mensaje en otros temas de SNS o enviar el mensaje a otros servicios. AWS

Además, Amazon SNS también admite los atributos de estado de entrega de los mensajes para las notificaciones enviadas a los puntos de conexión de Lambda. Para obtener más información, consulte Estado de entrega de mensajes de Amazon SNS.

Temas

Requisitos previos para integrar Amazon SNS con las funciones de Lambda en las distintas regiones

Para invocar funciones de Lambda mediante notificaciones de Amazon SNS, necesita lo siguiente:

- Una función Lambda
- Un tema de Amazon SNS

Para obtener información acerca de cómo crear una función de Lambda para utilizarla con Amazon SNS, consulte Uso de Lambda con Amazon SNS. Para obtener más información acerca de la creación de un tema de Amazon SNS, consulte Crear un tema.

Cuando se utiliza Amazon SNS para enviar mensajes desde regiones registradas a regiones habilitadas de forma predeterminada, debe modificar la política creada en la función de AWS Lambda reemplazando la entidad principal `sns.amazonaws.com` por `sns.<opt-in-region>.amazonaws.com`.

Por ejemplo, si desea suscribir una función de Lambda en EE. UU. Este (Norte de Virginia) a un tema de SNS en Asia-Pacífico (Hong Kong), cambie la entidad principal en la política de funciones de AWS Lambda a `sns.ap-east-1.amazonaws.com`. Las regiones registradas incluyen cualquier región lanzada después del 20 de marzo de 2019, que incluye Asia-Pacífico (Hong Kong), Oriente Medio (Baréin), UE (Milán) y África (Ciudad del Cabo). Las regiones lanzadas antes del 20 de marzo de 2019 están habilitadas de forma predeterminada.

> **Note**
>
> AWS no admite la entrega entre regiones a Lambda desde una región que esté habilitada de forma predeterminada a una región de suscripción voluntaria. Además, no se admite el reenvío entre regiones de mensajes SNS desde regiones registradas a otras regiones registradas.

Suscripción de una función de Lambda a un tema de Amazon SNS

En este tema se explica cómo suscribir una función de Lambda a un tema de Amazon SNS, lo que permite que los mensajes publicados activen la función.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. Elija un tema en la página Temas.
4. En la sección Suscripciones, elija Crear suscripción.
5. En la página Crear suscripción, en la sección Detalles, haga lo siguiente:
    a. Compruebe el ARN de tema elegido.
    b. En Protocolo, elija. AWS Lambda
    c. En Punto de conexión, escriba el ARN de una función.
    d. Elija Crear suscripción.

Cuando se publica un mensaje en un tema de SNS que tiene una función de Lambda suscrita, la función de Lambda se invoca con la carga útil del mensaje publicado. Para obtener información sobre cómo usarlo AWS Lambda con Amazon SNS, incluido un tutorial, consulte Uso AWS Lambda con Amazon SNS.

Distribución ramificada de notificaciones de Amazon SNS a colas de Amazon SQS para su procesamiento asíncrono

Amazon SNS funciona conjuntamente con Amazon Simple Queue Service (Amazon SQS). Estos servicios ofrecen a los desarrolladores diferentes beneficios. Con Amazon SNS, las aplicaciones pueden enviar mensajes en los que el tiempo es esencial a varios suscriptores a través del mecanismo “push”, lo que elimina la necesidad de comprobar o “sondear” de forma periódica en busca de actualizaciones. Amazon SQS es un servicio de cola de mensajes que emplean las aplicaciones distribuidas para intercambiar mensajes mediante un modelo de sondeo, y puede utilizarse para desacoplar los componentes de envío y los de recepción, sin necesitar que cada componente esté disponible de forma simultánea. Al utilizar Amazon SNS y Amazon SQS de forma conjunta, los mensajes pueden entregarse a aquellas aplicaciones que requieran la notificación inmediata de un evento y también pueden conservarse en una cola de Amazon SQS para que otras aplicaciones los procesen posteriormente.

Cuando suscribe una cola de Amazon SQS a un tema de Amazon SNS, puede publicar un mensaje en el tema y Amazon SNS envía un mensaje Amazon SQS a la cola suscrita. En el mensaje de Amazon SQS se incluye el tema y el mensaje que se publicaron en el tema junto con los metadatos del mensaje en un documento JSON. El mensaje de Amazon SQS tendrá un aspecto similar al documento JSON siguiente.

```
{
   "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"
}
```

Suscripción de una cola de Amazon SQS a un tema de Amazon SNS

Para permitir que un tema de Amazon SNS envíe mensajes a una cola de Amazon SQS, elija una de las siguientes opciones:

- Utilice la consola de Amazon SQS, lo que simplifica el proceso. Para obtener más información, consulte Suscripción de una cola de Amazon SQS a un tema de Amazon SNS en la Guía para desarrolladores de Amazon Simple Queue Service.
- Utilice los siguientes pasos:
  1. Obtenga el nombre de recurso de Amazon (ARN) de la cola a la que desea enviar mensajes y del tema al que desea suscribir la cola.
  2. Conceda el permiso `sqs:SendMessage` al tema de Amazon SNS para que pueda enviar mensajes a la cola.
  3. Suscriba la cola al tema de Amazon SNS.
  4. Conceda a los usuarios de IAM o a Cuentas de AWS los permisos adecuados para publicar en el tema de Amazon SNS y leer los mensajes de la cola de Amazon SQS.

5. Pruebe el procedimiento publicando un mensaje en el tema y leyendo el mensaje de la cola.

Si desea saber cómo configurar un tema para enviar mensajes a una cola que está en otra cuenta de AWS, consulte Envío de mensajes de Amazon SNS a una cola de Amazon SQS de otra cuenta.

Para ver una AWS CloudFormation plantilla que cree un tema que envíe mensajes a dos colas, consulte. Automatización de la mensajería de Amazon SNS a Amazon SQS con AWS CloudFormation

Paso 1: obtener el ARN de la cola y el del tema

Cuando suscriba una cola a un tema, necesitará una copia del ARN de la cola. Del mismo modo, cuando conceda permiso para que el tema envíe mensajes a la cola, necesitará una copia del ARN del tema.

Para obtener el ARN de la cola, puede utilizar la consola Amazon SQS o la acción de la API. GetQueueAttributes

Para obtener el ARN de la cola de la consola de Amazon SQS, siga estos pasos:

1. Inicie sesión en la consola de Amazon SQS AWS Management Console y ábrala en. https://console.aws.amazon.com/sqs/
2. Seleccione la casilla de la cola cuyo ARN quiere obtener.
3. En la sección Detalles, copie el valor del ARN de forma que pueda utilizarlo para suscribirse al tema de Amazon SNS.

Para obtener el ARN del tema, puede utilizar la consola de Amazon SNS, el comando `sns-get-topic-attributes` o la acción `GetQueueAttributes` de la API.

Para obtener el ARN del tema desde la consola de Amazon SNS, siga estos pasos:

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, seleccione el tema cuyo ARN desea obtener.
3. En la sección Detalles, copie el valor ARN del tema de forma que pueda utilizarlo para conceder permiso al tema de Amazon SNS y enviar mensajes a la cola.

Paso 2: conceder permiso al tema de Amazon SNS y enviar mensajes a la cola de Amazon SQS

Para que un tema de Amazon SNS pueda enviar mensajes a una cola, debe establecer una política en la cola que permita que el tema de Amazon SNS pueda ejecutar la acción `sqs:SendMessage`.

Antes de suscribir una cola a un tema, necesita un tema y una cola. Si aún no ha creado un tema o una cola, créelos ahora. Para obtener más información, consulte Creación de un tema y Creación de una cola en la Guía para desarrolladores de Amazon Simple Queue Service.

Para establecer una política en una cola, puede utilizar la consola Amazon SQS o la acción de SetQueueAttributesla API. Antes de comenzar, asegúrese de que dispone del ARN del tema al que desea conceder permiso para enviar mensajes a la cola. Si está suscrito a una cola para varios temas, la política debe contener un elemento `Statement` para cada tema.

Para establecer una SendMessage política en una cola mediante la consola Amazon SQS

1. Inicie sesión en la consola de Amazon SQS AWS Management Console y ábrala en. https://console.aws.amazon.com/sqs/
2. Seleccione la casilla de la cola cuya política desea establecer, elija la pestaña Política de acceso y, a continuación, elija Editar.
3. En la sección Política de acceso, defina quién puede acceder a la cola.

   - Añada una condición que permita la acción para el tema.
   - Establezca `Principal` para que sea el servicio de Amazon SNS, como se muestra en el ejemplo siguiente.
   - Utilice las claves de condición global `aws:SourceArn` o `aws:SourceAccount` para protegerse del escenario de suplente confuso. Para usar estas claves de condición, establezca el valor en el ARN de su tema. Si su cola está suscrita a varios temas, puede utilizar `aws:SourceAccount` en su lugar.

   Por ejemplo, la siguiente política permite MyTopic enviar mensajes a 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"
        }
      }
    }
  ]
}
```

Paso 3: suscribir la cola al tema de Amazon SNS

Para enviar mensajes a una cola a través de un tema, debe suscribir la cola al tema de Amazon SNS. La cola se especifica con el ARN. Para suscribirse a un tema, puede utilizar la consola de Amazon SNS, el comando `sns-subscribe` de la CLI o la acción de API `Subscribe`. Antes de comenzar, asegúrese de que tiene el ARN de la cola que desea suscribir.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. Elija un tema en la página Temas.
4. En la ***MyTopic*** página, en la página Suscripciones, selecciona Crear suscripción.
5. En la página Crear suscripción, en la sección Detalles, haga lo siguiente:

   a. Verifique el ARN del tema.

   b. En Protocolo, elija Amazon SQS.

   c. En Punto de conexión, indique el ARN de una cola de Amazon SQS.

   d. Elija Crear suscripción.

   Cuando se confirme la suscripción, el campo ID de suscripción de la nueva suscripción mostrará su ID de suscripción. Si el propietario de la cola crea la suscripción, esta se confirmará de forma automática y se activará casi de inmediato.

   Normalmente, suscribirá su propia cola a su propio tema en su propia cuenta. Sin embargo, también puede suscribir una cola de otra cuenta a su tema. Si el usuario que crea la suscripción

no es el propietario de la cola (por ejemplo, si un usuario de una cuenta A suscribe una cola de una cuenta B a un tema de la cuenta A), la suscripción deberá confirmarse. Para obtener más información sobre cómo suscribir una cola desde otra cuenta y confirmar la suscripción, consulte Envío de mensajes de Amazon SNS a una cola de Amazon SQS de otra cuenta.

Paso 4: conceder a los usuarios permisos para las acciones adecuadas del tema y la cola

Debe usar AWS Identity and Access Management (IAM) para permitir que solo los usuarios adecuados publiquen en el tema de Amazon SNS y lean o eliminen mensajes de la cola de Amazon SQS. Para obtener más información sobre el control de las acciones que pueden realizar los usuarios de IAM en los temas y las colas, consulte Uso de políticas basadas en identidades con Amazon SNS e Identity and Access Management en Amazon SQS en la Guía para desarrolladores de Amazon Simple Queue Service.

Hay dos formas de controlar el acceso a un tema o una cola:

- Añada una política a un usuario o un grupo de IAM. La forma más sencilla de conceder a los usuarios permisos para temas o colas consiste en crear un grupo, añadir la política adecuada al grupo y, a continuación, añadir usuarios a dicho grupo. Es mucho más fácil añadir y eliminar usuarios de un grupo que mantener un seguimiento de las políticas que se han configurado para los distintos usuarios.
- Añada una política a un tema o una cola. Si quiere conceder permisos sobre un tema o una cola a otra AWS cuenta, la única forma de hacerlo es añadiendo una política que tenga como principal la parte a la Cuenta de AWS que desea conceder permisos.

Debe utilizar el primer método para la mayoría de los casos (aplicar políticas a grupos y administrar los permisos de los usuarios añadiendo o eliminando los usuarios a los grupos). Si necesita conceder permisos a un usuario de otra cuenta, debe utilizar el segundo método.

Cómo añadir una política a un usuario o un grupo de IAM

Si ha añadido la siguiente política a un usuario o grupo de IAM, debería conceder a ese usuario o a los miembros de ese grupo permiso para realizar la `sns:Publish` acción sobre el tema MyTopic.

```
{
  "Statement": [
```

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

Si añadiera la siguiente política a un usuario o grupo de IAM, le daría permiso a ese usuario o a los miembros de ese grupo para realizar las `sqs:DeleteMessage` acciones `sqs:ReceiveMessage` y acciones de las colas MyQueue 1 y 2. MyQueue

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

Cómo añadir una política a un tema o una cola

Las siguientes políticas de ejemplo muestran cómo conceder a otra cuenta permisos sobre un tema y una cola.

> **Note**
>
> Cuando concedes a otra persona el Cuenta de AWS acceso a un recurso de tu cuenta, también concedes permisos a ese recurso a los usuarios de IAM que tienen acceso de nivel de administrador (acceso comodín). Al resto de los usuarios de IAM de la otra cuenta se les deniega de manera automática el acceso al recurso. Si desea conceder a usuarios específicos de IAM en esa Cuenta de AWS acceso a su recurso, la cuenta o el usuario de IAM con acceso de nivel de administrador debe delegar permisos para el recurso a esos

usuarios de IAM. Para obtener más información acerca de la delegación entre cuentas, consulte Cómo habilitar el acceso entre cuentas en la Guía del usuario de IAM.

Si agregas la siguiente política a un tema MyTopic de la cuenta 123456789012, darías permiso a la cuenta 111122223333 para realizar la acción en ese tema. `sns:Publish`

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

Si agregas la siguiente política a una cola MyQueue en la cuenta 123456789012, darías permiso a la cuenta 111122223333 para realizar las acciones y acciones de esa cola. `sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Paso 5: probar las suscripciones de un tema a una cola

Puede probar las suscripciones de un tema a una cola publicando en el tema y viendo el mensaje que el tema envía a la cola.

Para publicar en un tema mediante la consola de Amazon SNS, siga estos pasos:

1. Con las credenciales del usuario de IAM Cuenta de AWS o del usuario de IAM con permiso para publicar en el tema, inicie sesión en la consola de Amazon SNS AWS Management Console y ábrala en. https://console.aws.amazon.com/sns/
2. En el panel de navegación, seleccione el tema y elija Publicar en tema.
3. En el cuadro Asunto, escriba un asunto (por ejemplo, **`Testing publish to queue`**), en el cuadro Mensaje, introduzca algún texto (por ejemplo, **`Hello world!`**) y, por último, elija Publicar mensaje. Aparecerá el siguiente mensaje: Your message has been successfully published.

Para ver el mensaje del tema mediante la consola de Amazon SQS, siga estos pasos:

1. Con las credenciales del usuario de IAM Cuenta de AWS o del usuario de IAM con permiso para ver los mensajes de la cola, inicie sesión en la consola de Amazon SQS AWS Management Console y ábrala en. https://console.aws.amazon.com/sqs/
2. Elija una cola que esté suscrita al tema.
3. Elija Enviar y recibir mensajes y, a continuación, elija Sondeo de mensajes. Aparecerá un mensaje con el tipo Notificación.
4. En la columna Cuerpo, elija Más detalles. El cuadro Detalles del mensaje contiene un documento JSON con el tema y el mensaje que ha publicado en el tema. El mensaje tiene un aspecto similar al documento JSON siguiente.

```
{
  "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. Elija Cerrar. Ha publicado correctamente en un tema que envía mensajes de notificación a una cola.

Automatización de la mensajería de Amazon SNS a Amazon SQS con AWS CloudFormation

AWS CloudFormation le permite utilizar un archivo de plantilla para crear y configurar un conjunto de AWS recursos juntos como una sola unidad. Esta sección tiene una plantilla de ejemplo que facilita la implementación de temas que publiquen en colas. Las plantillas se hacen cargo de los pasos de configuración en su lugar, creando dos colas y un tema con suscripciones a las colas, añadiendo una política a las colas, de modo que el tema pueda enviar mensajes a las colas y creando usuarios y grupos de IAM para controlar el acceso a dichos recursos.

Para obtener más información sobre la implementación de AWS recursos mediante una AWS CloudFormation plantilla, consulte Primeros pasos en la Guía del AWS CloudFormation usuario.

Uso de una AWS CloudFormation plantilla para configurar temas y colas dentro de un Cuenta de AWS

Con la plantilla de ejemplo, se crea un tema de Amazon SNS con el que se puede enviar mensajes a dos colas de Amazon SQS con los permisos adecuados para que los miembros de un grupo de IAM puedan publicar en el tema y los de otro leer mensajes de las colas. Además, con la plantilla, se crean usuarios de IAM que se agregan a cada grupo.

Copie el contenido de la plantilla en un archivo. También puede descargar la plantilla desde la página de AWS CloudFormation plantillas. En la página de plantillas, selecciona Buscar plantillas de muestra por AWS servicio y, a continuación, elige Amazon Simple Queue Service.

My SNSTopic está configurado para publicar en dos puntos de enlace suscritos, que son dos colas de Amazon SQS MyQueue (1 y 2). MyQueue MyPublishTopicGroup es un grupo de IAM cuyos miembros tienen permiso para publicar en My SNSTopic mediante la acción Publish API o el comando sns-publish. La plantilla crea los usuarios de IAM y les proporciona los perfiles

de inicio de sesión MyPublishUser MyQueueUser y las claves de acceso. El usuario que crea una pila con esta plantilla especifica las contraseñas para los perfiles de inicio de sesión como parámetros de entrada. La plantilla crea claves de acceso para los dos usuarios de IAM con MyPublishUserKey y. MyQueueUserKey AddUserToMyPublishTopicGroup se agrega MyPublishUser a la MyPublishTopicGroup para que el usuario tenga los permisos asignados al grupo.

My RDMessage QueueGroup es un grupo de IAM cuyos miembros tienen permiso para leer y eliminar mensajes de las dos colas de Amazon SQS mediante ReceiveMessagelas DeleteMessageacciones y API. AddUserToMyQueueGroup se agrega MyQueueUser a My RDMessage QueueGroup para que el usuario tenga los permisos asignados al grupo. MyQueuePolicy asigna permiso SNSTopic a My para publicar sus notificaciones en las dos colas.

En la siguiente lista se muestra el contenido de la AWS CloudFormation plantilla.

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

Distribución ramificada de notificaciones de Amazon SNS a puntos de conexión HTTPS

Puede utilizar Amazon SNS para enviar mensajes de notificación a uno o varios puntos de conexión HTTP o HTTPS. Cuando suscribe un punto de conexión a un tema, puede publicar una notificación en el tema y Amazon SNS enviará una solicitud HTTP POST al entregar el contenido de la notificación al punto de conexión suscrito. Cuando suscribe el punto de conexión, indica si Amazon SNS utiliza HTTP o HTTPS para enviar la solicitud POST al punto de conexión. Si utiliza HTTPS, puede aprovechar la compatibilidad con Amazon SNS para lo siguiente:

- Indicación de nombre de servidor (SNI): esto permite que Amazon SNS admita puntos de conexión HTTPS que requieren SNI, como un servidor que solicita varios certificados para alojar varios dominios. Para obtener más información sobre SNI, consulte Server Name Indication.

- Autenticación de acceso básica y abreviada: esto permite especificar un nombre de usuario y contraseña en la URL HTTPS para la solicitud HTTP POST, como `https://user:password@domain.com` o `https://user@domain.com`. El nombre de usuario y la contraseña se cifran a través de la conexión SSL establecida al utilizar HTTPS. Solo el nombre de dominio se envía en texto sin cifrar. Para obtener más información sobre la autenticación de acceso básica y abreviada, consulte RFC-2617.

  Important

  Amazon SNS no admite actualmente puntos de conexión HTTP(S) privados. URLs Los HTTPS solo se pueden recuperar de la acción de la API de Amazon `GetSubscriptionAttributes` SNS para los directores a los que haya concedido acceso a la API.

  Note

  El servicio cliente debe admitir el encabezado de respuesta `HTTP/1.1 401 Unauthorized`

La solicitud contiene el asunto y el mensaje que se publicaron en el tema junto con los metadatos de la notificación en un documento JSON. La solicitud tendrá un aspecto similar a la siguiente solicitud HTTP POST. Para obtener más información sobre el encabezado HTTP y el formato JSON del cuerpo de la solicitud, consulte Encabezados de HTTP/HTTPS y Formato JSON de notificación HTTP/HTTPS.

Note

Amazon SNS considera que se pueden volver a intentar todos los errores 5XX y 429 (se enviaron demasiadas solicitudes). Estos errores están sujetos a la política de entrega. Todos los demás errores se consideran fallos permanentes y no se intentará volver a intentarlo.

```
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"
}
```

Suscripción de un punto de conexión HTTP/S en un tema de Amazon SNS

En este tema se explica cómo suscribir puntos de enlace HTTP/S a los temas de Amazon SNS.

Temas

- Paso 1: Asegúrese de que el punto de enlace está listo para procesar mensajes de Amazon SNS.
- Paso 2: Suscribir el punto de enlace HTTP/HTTPS al tema de Amazon SNS
- Paso 3: confirme la suscripción de Amazon SNS
- Paso 4: defina la política de entrega para la suscripción de Amazon SNS (opcional)
- Paso 5: conceda a los usuarios permisos para publicar en el tema de Amazon SNS (opcional)
- Paso 6: envíe mensajes de Amazon SNS al punto de conexión HTTP/HTTPS

Paso 1: Asegúrese de que el punto de enlace está listo para procesar mensajes de Amazon SNS.

Antes de suscribir su punto de enlace HTTP o HTTPS a un tema, debe asegurarse de que el punto de enlace HTTP o HTTPS tiene la capacidad de administrar las solicitudes HTTP POST que Amazon SNS utiliza para enviar la confirmación de suscripción y los mensajes de notificación. Por lo general, esto implica crear e implementar una aplicación web (por ejemplo, un servlet Java si el host del punto de enlace ejecuta Linux con Apache y Tomcat) que procese las solicitudes HTTP de Amazon SNS. Cuando suscribe un punto de enlace HTTP, Amazon SNS envía una solicitud de confirmación de la suscripción. El punto de enlace debe estar preparado para recibir y procesar esta solicitud cuando cree la suscripción, porque Amazon SNS envía esta solicitud en ese momento. Amazon SNS no enviará notificaciones al punto de enlace hasta que se confirme la suscripción. Una vez confirmada la suscripción, Amazon SNS enviará notificaciones al punto de enlace cuando se ejecute una acción de publicación en el tema suscrito.

Para configurar el punto de enlace para que procese los mensajes de confirmación de la suscripción y de notificación

1. El código debe leer los encabezados HTTP de las solicitudes HTTP POST que Amazon SNS envía a su punto de enlace. El código debe examinar el campo de encabezado `x-amz-sns-message-type`, en el que se indica el tipo de mensaje que Amazon SNS ha enviado. En este encabezado, puede determinar el tipo de mensaje sin tener que analizar el cuerpo de la solicitud HTTP. Hay dos tipos que debe administrar: `SubscriptionConfirmation` y `Notification`. El mensaje `UnsubscribeConfirmation` se utiliza únicamente cuando la suscripción se elimina del tema.

   Para obtener información detallada sobre el encabezado HTTP, consulte Encabezados de HTTP/HTTPS. La siguiente solicitud HTTP POST es un ejemplo de un mensaje de confirmación de la suscripción.

```
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. El código debe analizar el documento JSON del cuerpo de la solicitud HTTP POST y el texto sin formato de tipo de contenido para leer los pares de nombre-valor que conforman el mensaje de Amazon SNS. Utilice un analizador JSON que se encargue de convertir la representación en forma de secuencias de escape de los caracteres de control en sus valores de caracteres ASCII (por ejemplo, convertir \n en un carácter de nueva línea). Puede utilizar un analizador JSON existente como Jackson JSON Processor o crear el suyo propio. Para poder enviar el texto del asunto y los campos de los mensajes en formato JSON válido, Amazon SNS debe convertir algunos caracteres de control en secuencias de escape que se puedan incluir en el documento JSON. Cuando reciba el documento JSON en el cuerpo de la solicitud POST enviada a su punto de enlace, debe convertir los caracteres incluidos en secuencias de escape en sus valores de caracteres originales si desea una representación exacta del asunto original y de los mensajes publicados en el tema. Esto es fundamental si desea verificar la firma de una notificación, porque la firma utiliza el mensaje y el asunto en sus formatos originales como parte de la cadena para firmar.
3. El código debe verificar la autenticidad de una notificación, la confirmación de la suscripción o la cancelación del mensaje de confirmación enviado por Amazon SNS. Mediante la información incluida en el mensaje de Amazon SNS, el punto de enlace puede volver a crear la firma para que se pueda verificar el contenido del mensaje cotejando la firma propia con la firma que Amazon SNS envió con el mensaje. Para obtener más información acerca de la verificación de la firma de un mensaje, consulte Verificación de la firmas de mensajes de Amazon SNS.

4. Según el tipo especificado por el campo de encabezado `x-amz-sns-message-type`, el código debe leer el documento JSON incluido en el cuerpo de la solicitud HTTP y procesar el mensaje. Estas son las directrices para administrar los dos tipos principales de mensajes:

   SubscriptionConfirmation

   Lea el valor de `SubscribeURL` y visite esa URL. Para confirmar la suscripción y empezar a recibir notificaciones en el punto de enlace, debe visitar la URL `SubscribeURL` (por ejemplo, enviando una solicitud HTTP GET a la URL). Consulte el ejemplo de la solicitud HTTP del paso anterior para ver cómo es esa URL `SubscribeURL`. Para obtener más información sobre el formato del mensaje `SubscriptionConfirmation`, consulte Formato JSON de confirmación de suscripción HTTP/HTTPS. Cuando visite la dirección URL, recibirá una respuesta similar al siguiente documento XML. El documento devuelve el ARN de suscripción del punto de enlace en el 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 a visitar el`SubscribeURL`, puede confirmar la suscripción mediante la ConfirmSubscriptionacción cuyo valor correspondiente aparece en el `Token` mensaje. `SubscriptionConfirmation` Si desea permitir únicamente al propietario del tema y al propietario de la suscripción que cancelen la suscripción del punto de enlace, puede llamar a la acción `ConfirmSubscription` con una firma de AWS .

   Notificación

   Lea los valores de `Subject` y `Message` para obtener la información de la notificación que se publicó en el tema.

   Para obtener más información sobre el formato del mensaje `Notification`, consulte Encabezados de HTTP/HTTPS. La siguiente solicitud HTTP POST es un ejemplo de un mensaje de notificación enviado al punto de enlace 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. Asegúrese de que su punto de enlace responde al mensaje HTTP POST de Amazon SNS con el código de estado adecuado. El tiempo de espera de la conexión se agotará en 15 segundos aproximadamente. Si el punto de conexión no responde antes de que se agote el tiempo de espera de la conexión, o si devuelve un código de estado fuera del intervalo 200-4xx, Amazon SNS considerará la entrega del mensaje un intento fallido.
6. Asegúrese de que el código puede administrar los reintentos de entrega de mensajes de Amazon SNS. Si Amazon SNS no recibe una respuesta correcta del punto de enlace, intenta entregar de nuevo el mensaje. Esto se aplica a todos los mensajes, incluido el mensaje de confirmación de la suscripción. De forma predeterminada, si la entrega inicial del mensaje da un error, Amazon SNS realiza tres reintentos con un intervalo entre los intentos fallidos establecido en 20 segundos.

> **Note**
>
> El tiempo de espera de la solicitud de mensajes se agota tras 15 segundos aproximadamente. Esto significa que si no se pudo entregar el mensaje porque se agotó el tiempo de espera, Amazon SNS lo volverá a intentar aproximadamente 35 segundos después del intento de entrega anterior. Puede establecer una política de entrega diferente para el punto de enlace.

Amazon SNS usa el campo de encabezado `x-amz-sns-message-id` para identificar de forma única cada mensaje publicado en un tema de Amazon SNS. Al comparar IDs los mensajes que ha procesado con los mensajes entrantes, puede determinar si se trata de un intento de reintento.

7. Si suscribe un punto de enlace HTTPS, asegúrese de que el punto de enlace tiene un certificado de servidor de una entidad de certificación (CA) de confianza. Amazon SNS solo enviará mensajes a puntos de enlace HTTPS que tengan un certificado de servidor de una CA en la que confíe Amazon SNS.
8. Implemente el código que ha creado para recibir mensajes de Amazon SNS. Cuando suscriba el punto de enlace, este debe estar preparado para recibir al menos el mensaje de confirmación de la suscripción.

Paso 2: Suscribir el punto de enlace HTTP/HTTPS al tema de Amazon SNS

Para enviar mensajes a un punto de enlace HTTP o HTTPS a través de un tema, debe suscribir el punto de enlace al tema de Amazon SNS. El punto de enlace se especifica por medio de su URL. Para suscribir a un tema, puede utilizar la consola de Amazon SNS, el comando sns-subscribe o la acción de API Suscribir. Antes de empezar, asegúrese de que tiene la dirección URL del punto de enlace que desea suscribir y de que el punto de enlace está preparado para recibir los mensajes de configuración y notificación, tal como se describe en el paso 1.

Para suscribir un punto de enlace HTTP o HTTPS a un tema mediante la consola de Amazon SNS, siga estos pasos:

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, seleccione Subscriptions (Suscripciones).
3. Elija Create subscription (Crear suscripción).

4. En la lista desplegable Protocol (Protocolo), seleccione HTTP o HTTPS.
5. En el cuadro Endpoint (Punto de enlace), pegue la dirección URL del punto de enlace al que desea que el tema envíe los mensajes y, a continuación, elija Create subscription (Crear suscripción).
6. Se muestra el mensaje de confirmación. Seleccione Cerrar.

   Aparece PendingConfirmation el identificador de suscripción de tu nueva suscripción. Cuando confirme la suscripción, Subscription ID (ID de suscripción) mostrará el ID de suscripción.

Paso 3: confirme la suscripción de Amazon SNS

Para confirmar tu suscripción a Amazon SNS, sigue estos pasos para asegurarte de que tu punto de conexión pueda recibir correctamente los mensajes. Este proceso implica configurar el dispositivo de punto final para gestionar los mensajes de confirmación entrantes, recuperar la URL de confirmación y confirmar la suscripción. Puedes confirmar la suscripción de forma automática o manual, en función de tu configuración.

1. Tras suscribirse a un tema de Amazon SNS, Amazon SNS envía un mensaje de confirmación a su punto de conexión. Este mensaje contiene un mensaje `SubscribeURL` que debe utilizar para confirmar la suscripción.
2. El punto de conexión debe estar configurado para escuchar los mensajes entrantes de Amazon SNS. Cuando llegue el mensaje de confirmación, **`SubscribeURL`**extráigalo del mensaje.
3. Una vez que lo tengas`SubscribeURL`, puedes confirmar la suscripción de dos maneras:

   - Confirmación automática: su terminal puede confirmar automáticamente la suscripción enviando una solicitud HTTP GET al`SubscribeURL`.

     Este método no requiere intervención manual.
   - Confirmación manual: si la confirmación automática no está configurada, copie el **`SubscribeURL`**fragmento del mensaje de confirmación y péguelo en la barra de direcciones del navegador.

     Esto confirmará la suscripción manualmente.
4. También puede verificar el estado de la suscripción mediante la consola de Amazon SNS:

   a. Inicie sesión en la consola de Amazon SNS.

   b. En el panel de navegación, seleccione Suscripciones.

c. Busque su suscripción en la lista.

- Si se confirma, se `SubscriptionArn` mostrará.
- Si sigue sin confirmarse, se mostrará como`PendingConfirmation`.

Paso 4: defina la política de entrega para la suscripción de Amazon SNS (opcional)

De forma predeterminada, si la entrega inicial del mensaje da un error, Amazon SNS realiza tres reintentos con un intervalo entre los intentos fallidos establecido en 20 segundos. Como se ha explicado en el paso 1, el punto de enlace debe tener código que pueda administrar los reintentos de entrega de mensajes. Mediante la configuración de la política de entrega en un tema o suscripción, puede controlar la frecuencia y el intervalo con los que Amazon SNS intenta entregar de nuevo los mensajes fallidos. También puede especificar el tipo de contenido de sus notificaciones HTTP/S en `DeliveryPolicy`. Para obtener más información, consulte Creación de una política de entrega HTTP/S.

Paso 5: conceda a los usuarios permisos para publicar en el tema de Amazon SNS (opcional)

De forma predeterminada, el propietario del tema tiene permisos para publicar en el tema. Para permitir que otros usuarios o aplicaciones publiquen en el tema, debe usar AWS Identity and Access Management (IAM) para conceder permisos de publicación al tema. Si desea obtener más información sobre cómo conceder permisos para las acciones de Amazon SNS a los usuarios de IAM, consulte Uso de políticas basadas en identidades con Amazon SNS.

Hay dos formas de controlar el acceso a un tema:

- Agregue una política a un usuario o un grupo de IAM. La forma más sencilla de conceder a los usuarios permisos para temas consiste en crear un grupo, añadir la política adecuada al grupo y, a continuación, añadir usuarios a dicho grupo. Es mucho más fácil añadir y eliminar usuarios de un grupo que mantener un seguimiento de las políticas que se han configurado para los distintos usuarios.
- Añadiendo una política al tema. Si desea conceder permisos para un tema a otra cuenta de AWS , el único modo de hacerlo es agregando una política que tenga como entidad principal la Cuenta de AWS a la que desee conceder los permisos.

Debe utilizar el primer método para la mayoría de los casos (aplicar políticas a grupos y administrar los permisos de los usuarios añadiendo o eliminando los usuarios a los grupos). Si necesita conceder permisos a un usuario de otra cuenta, utilice el segundo método.

Si agregas la siguiente política a un usuario o grupo de IAM, le darías permiso a ese usuario o miembros de ese grupo para realizar la `sns:Publish` acción en el tema. MyTopic

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

La siguiente política de ejemplo muestra cómo conceder a otra cuenta permisos sobre un tema.

> **Note**
>
> Cuando concedes a otra persona el Cuenta de AWS acceso a un recurso de tu cuenta, también concedes permisos a ese recurso a los usuarios de IAM que tienen acceso de nivel de administrador (acceso comodín). Al resto de los usuarios de IAM de la otra cuenta se les deniega de manera automática el acceso al recurso. Si quieres conceder a usuarios de IAM específicos de ese tipo Cuenta de AWS acceso a tu recurso, la cuenta o un usuario de IAM con acceso de nivel de administrador debe delegar los permisos del recurso a esos usuarios de IAM. Para obtener más información acerca de la delegación entre cuentas, consulte Cómo habilitar el acceso entre cuentas en la Guía del usuario de IAM.

Si agregaste la siguiente política a un tema MyTopic de la cuenta 123456789012, darías permiso a la cuenta 111122223333 para realizar la acción sobre ese tema. `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"
  }]
}
```

Paso 6: envíe mensajes de Amazon SNS al punto de conexión HTTP/HTTPS

Puede enviar un mensaje a las suscripciones de un tema mediante su publicación en el tema. Para publicar en un tema, puede utilizar la consola de Amazon SNS, el comando `sns-publish` de la CLI o la API `Publish`.

Si ha seguido el paso 1, el código que ha implementado en el punto de enlace debería procesar la notificación.

Para publicar en un tema mediante la consola de Amazon SNS, siga estos pasos:

1. Con las credenciales del usuario de IAM Cuenta de AWS o del usuario de IAM con permiso para publicar en el tema, inicie sesión en la consola de Amazon SNS AWS Management Console y ábrala en. https://console.aws.amazon.com/sns/
2. En el panel de navegación izquierdo, elija Topics (Temas) y, a continuación, seleccione un tema.
3. Seleccione el botón Publish message (Publicar mensaje).
4. En el cuadro Subject (Asunto), introduzca el asunto (por ejemplo, **`Testing publish to my endpoint`**).
5. En el cuadro Message (Mensaje), introduzca algún texto (por ejemplo, **`Hello world!`**) y elija Publish message (Publicar mensaje).

   Aparecerá el siguiente mensaje: Your message has been successfully published.

Verificación de la firmas de mensajes de Amazon SNS

Amazon SNS utiliza las firmas de los mensajes para confirmar la autenticidad de los mensajes que se envían a su punto de enlace HTTP. Para garantizar la integridad de los mensajes y evitar la suplantación de identidad, debe verificar la firma antes de procesar cualquier mensaje de Amazon SNS.

¿Cuándo debe verificar las firmas de Amazon SNS?

Debe verificar las firmas de los mensajes de Amazon SNS en los siguientes escenarios:

- Cuando Amazon SNS envía un mensaje de notificación a su punto de enlace HTTP (S).
- Cuando Amazon SNS envía un mensaje de confirmación a su punto de conexión tras una `Subscribe`llamada a la `Unsubscribe`API.

Amazon SNS admite dos versiones de firma:

- SignatureVersion1 — Utiliza un SHA1 hash del mensaje.
- SignatureVersion2 — Utiliza un SHA256 hash del mensaje. Esto proporciona una mayor seguridad y es la opción recomendada.

Para verificar correctamente las firmas de los mensajes de SNS, siga estas prácticas recomendadas:

- Recupere siempre el certificado de firma mediante HTTPS para evitar ataques de interceptación no autorizados.
- Compruebe que el certificado lo ha emitido Amazon SNS.
- Confirme que la cadena de confianza del certificado sea válida.
- El certificado debe provenir de una URL firmada por el SNS.
- No confíe en ningún certificado proporcionado en el mensaje sin validación.
- Rechaza cualquier mensaje que contenga un mensaje inesperado `TopicArn` para evitar la suplantación de identidad.
- Los AWS SDKs para Amazon SNS incluyen una lógica de validación integrada, lo que reduce el riesgo de errores de implementación.

Temas sobre la configuración de la versión de firma de mensajes en Amazon SNS

La configuración de la versión de firma de mensajes en los temas de Amazon SNS le permite mejorar la seguridad y la compatibilidad del proceso de verificación de mensajes.

Seleccione entre `SignatureVersion` 1 (SHA1) y `SignatureVersion` 2 (SHA256) para controlar el algoritmo de hash utilizado para firmar los mensajes. Los temas de Amazon SNS son 1 de forma predeterminada. **`SignatureVersion`** Puede configurar este ajuste mediante la acción de la `SetTopicAttributes`API.

Usa el siguiente ejemplo para configurar el atributo topic `SignatureVersion` mediante AWS CLI:

```
aws sns set-topic-attributes \
```

```
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

Verificar la firma de un mensaje de Amazon SNS cuando se utilizan solicitudes basadas en consultas HTTP

La verificación de la firma de un mensaje de Amazon SNS cuando se utilizan solicitudes basadas en consultas HTTP garantiza la autenticidad e integridad del mensaje. Este proceso confirma que el mensaje proviene de Amazon SNS y no ha sido manipulado durante el tránsito. Al analizar el mensaje, crear la cadena correcta para firmarlo y validar la firma con una clave pública de confianza, se protege el sistema contra la suplantación de identidad y las alteraciones no autorizadas de los mensajes.

1. Extraiga los pares clave-valor del documento JSON del cuerpo de la solicitud HTTP POST enviada por Amazon SNS. Estos campos son necesarios para construir la cadena que se va a firmar.

   - `Message`
   - `Subject` (si está presente)
   - `MessageId`
   - `Timestamp`
   - `TopicArn`
   - `Type`

   Por ejemplo:

   ```
   MESSAGE_FILE="message.json"
   FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
   ```

   > **Note**
   >
   > Si algún campo contiene caracteres de escape (por ejemplo,`\n`), conviértalos a su formato original para garantizar una coincidencia exacta.

2. Localice el `SigningCertURL` campo en el mensaje de Amazon SNS. Este certificado contiene la clave pública necesaria para verificar la firma del mensaje. Por ejemplo:

```
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
```

3. Asegúrese de `SigningCertURL` que proviene de un AWS dominio de confianza (por ejemplo, https://sns.us-east-1.amazonaws.com). Rechaza cualquier AWS dominio URLs externo por motivos de seguridad.
4. Descargue el certificado X.509 desde la URL proporcionada. Por ejemplo:

```
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
```

5. Extraiga la clave pública del certificado X.509 descargado. La clave pública le permite descifrar la firma del mensaje y comprobar su integridad. Por ejemplo:

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. Los distintos tipos de mensajes requieren distintos pares clave-valor en la cadena para poder firmarlos. Identifique el tipo de mensaje (`Type`campo del mensaje de Amazon SNS) para determinar qué pares clave-valor incluir:
   - Mensaje de notificación: incluye `Message``MessageId`, `Subject` (si está presente),`Timestamp`, `TopicArn` y. `Type`
   - SubscriptionConfirmationo UnsubscribeConfirmation mensaje: incluye `MessageMessageId`,`SubscribeURL`, `TimestampToken`,`TopicArn`, y`Type`.
7. Amazon SNS requiere que la cadena se firme para seguir un orden de campos estricto y fijo para la verificación. Solo se deben incluir los campos explícitamente obligatorios; no se pueden añadir campos adicionales. Los campos opcionales, por ejemplo`Subject`, deben incluirse solo si están presentes en el mensaje y deben aparecer en la posición exacta definida por el orden de los campos obligatorios. Por ejemplo:

```
KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
```

> **Important**
>
> No añada un carácter de nueva línea al final de la cadena.

8. Organice los pares clave-valor en orden de bytes (alfabéticamente por nombre de clave).
9. Construye la cadena para firmarla con el siguiente ejemplo 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
```

Ejemplo de mensaje de notificación:

```
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 ejemplo:

```
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. El `Signature` campo del mensaje está codificado en Base64. Debe decodificarlo para comparar su forma binaria sin procesar con el hash derivado. Por ejemplo:

    ```
    SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
    echo "$SIGNATURE" | base64 -d > signature.bin
    ```

11. Usa el `SignatureVersion` campo para seleccionar el algoritmo de hash:

    - Para `SignatureVersion` 1, utilice SHA1(por ejemplo,`-sha1`).
    - Para `SignatureVersion` 2, utilice SHA256(por ejemplo,`-sha256`).

12. Para confirmar la autenticidad del mensaje de Amazon SNS, genere un hash de la cadena construida y verifique la firma con la clave pública.

    ```
    openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
     "$STRING_TO_SIGN"
    ```

    Si la firma es válida, el resultado lo es`Verified OK`. De lo contrario, la salida es`Verification Failure`.

Ejemplo de script con gestión de errores

El siguiente script de ejemplo automatiza el proceso de verificación:

```
#!/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)
```

Análisis de los formatos de mensajes de Amazon SNS

Cuando Amazon SNS envía mensajes a puntos de enlace HTTP/HTTPS, contienen encabezados HTTP y un cuerpo de mensaje JSON. Estos mensajes siguen un formato estructurado que incluye metadatos como el tipo de mensaje, el ARN del tema, las marcas de tiempo y las firmas digitales. Al analizar correctamente los mensajes de Amazon SNS, puede determinar si un mensaje es una confirmación de suscripción, una notificación o una confirmación de cancelación de suscripción, extraer los datos relevantes y verificar la autenticidad mediante la validación de firmas.

Encabezados de HTTP/HTTPS

Cuando Amazon SNS envía una confirmación de suscripción, una notificación o un mensaje de confirmación de anulación de la suscripción a los puntos de enlace HTTP/HTTPS, envía un mensaje POST con una serie de valores de encabezado específicos de Amazon SNS. Puede utilizar los valores del encabezado para tareas como identificar el tipo de mensaje sin tener que analizar el cuerpo del mensaje JSON para leer el valor de `Type`. De forma predeterminada, Amazon SNS envía todas las notificaciones a puntos de conexión HTTP/S con `Content-Type` establecido a `text/plain; charset=UTF-8`. Para elegir un `Content-Type` distinto de text/plain (valor predeterminado), consulte `headerContentType` en Creación de una política de entrega HTTP/S.

x-amz-sns-message-type

Tipo de mensaje. Los valores posibles son `SubscriptionConfirmation`, `Notification` y `UnsubscribeConfirmation`.

x-amz-sns-message-id

Un identificador único universal (UUID), único para cada mensaje publicado. En las notificaciones que Amazon SNS reenvía durante un reintento, se usa el ID de mensaje original.

x-amz-sns-topic-arn

Nombre de recurso de Amazon (ARN) del tema en el que se publicó el mensaje.

x-amz-sns-subscription-arn

ARN de la suscripción a este punto de enlace.

El siguiente encabezado HTTP POST es un ejemplo de encabezado para un mensaje `Notification` a un punto de conexión 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 confirmación de suscripción HTTP/HTTPS

Después de suscribir un HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS punto final. Este mensaje contiene un valor `SubscribeURL` que debe visitar para confirmar la suscripción (o bien, puede utilizar el valor `Token` con `ConfirmSubscription`).

> **Note**
>
> Amazon SNS no envía notificaciones a este punto de conexión hasta que se confirma la suscripción

El mensaje de confirmación de la suscripción es un mensaje POST con un cuerpo que contiene un documento JSON con los siguientes pares de nombre-valor.

`Type`

Tipo de mensaje. Para obtener una confirmación de suscripción, el tipo es `SubscriptionConfirmation`.

`MessageId`

Un identificador único universal (UUID), único para cada mensaje publicado. En los mensajes que Amazon SNS reenvía durante un reintento, se usa el ID de mensaje original.

`Token`

Un valor que puede utilizar con la acción `ConfirmSubscription` para confirmar la suscripción. También puede visitar simplemente `SubscribeURL`.

`TopicArn`

Nombre de recurso de Amazon (ARN) del tema al que está suscrito este punto de enlace.

`Message`

Cadena que describe el mensaje. Para la confirmación de suscripción, esta cadena tiene el aspecto siguiente:

```
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`

Dirección URL que debe visitar para confirmar la suscripción. O bien, puede utilizar `Token` con la acción `ConfirmSubscription` para confirmar la suscripción.

`Timestamp`

Hora (GMT) de envío de la confirmación de suscripción.

`SignatureVersion`

Versión de la firma de Amazon SNS utilizada.

- Si `SignatureVersion` es 1, `Signature` es una firma `SHA1withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Type`, `Timestamp` y `TopicArn`.
- Si `SignatureVersion` es 2, `Signature` es una firma `SHA256withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Type`, `Timestamp` y `TopicArn`.

`Signature`

Firma de `SHA1withRSA` o `SHA256withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Type`, `Timestamp` y `TopicArn`.

`SigningCertURL`

Dirección URL del certificado que se utilizó para firmar el mensaje.

El mensaje HTTP POST siguiente es un ejemplo de un mensaje de `SubscriptionConfirmation` a un punto de conexión 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 notificación HTTP/HTTPS

Cuando Amazon SNS envía una notificación a un punto de enlace HTTP o HTTPS suscrito, el cuerpo del mensaje POST enviado al punto de enlace contiene un documento JSON con los siguientes pares de nombre-valor.

`Type`

Tipo de mensaje. Para una notificación, el tipo es `Notification`.

`MessageId`

Un identificador único universal (UUID), único para cada mensaje publicado. En las notificaciones que Amazon SNS reenvía durante un reintento, se usa el ID de mensaje original.

`TopicArn`

Nombre de recurso de Amazon (ARN) del tema en el que se publicó el mensaje.

`Subject`

Parámetro `Subject` especificado cuando se publicó la notificación en el tema.

> **Note**
>
> Se trata de un parámetro opcional. Si no se especifica `Subject`, el par de nombre y valor no aparecerá en este documento JSON.

`Message`

Valor `Message` especificado cuando se publicó la notificación en el tema.

`Timestamp`

Hora (GMT) de publicación de la notificación.

`SignatureVersion`

Versión de la firma de Amazon SNS utilizada.

- Si `SignatureVersion` es 1, `Signature` es una firma `SHA1withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Subject` (si está presente), `Type`, `Timestamp` y `TopicArn`.
- Si `SignatureVersion` es 2, `Signature` es una firma `SHA256withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Subject` (si está presente), `Type`, `Timestamp` y `TopicArn`.

`Signature`

Firma de `SHA1withRSA` o `SHA256withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Subject` (si está presente), `Type`, `Timestamp` y `TopicArn`.

`SigningCertURL`

Dirección URL del certificado que se utilizó para firmar el mensaje.

`UnsubscribeURL`

Dirección URL que puede utilizar para cancelar la suscripción del punto de enlace a este tema. Si visita esta URL, Amazon SNS cancela la suscripción del punto de enlace y deja de enviarle notificaciones.

El mensaje HTTP POST siguiente es un ejemplo de un mensaje de `Notification` a un punto de conexión 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 confirmación de cancelación de suscripción HTTP/HTTPS

Después de cancelar la suscripción de un punto de enlace HTTP/HTTPS a un tema, Amazon SNS envía un mensaje de confirmación de cancelación de la suscripción al punto de enlace.

El mensaje de cancelación de la suscripción es un mensaje POST con un cuerpo que contiene un documento JSON con los siguientes pares de nombre-valor.

`Type`

Tipo de mensaje. Para obtener una confirmación de la cancelación de suscripción, el tipo es `UnsubscribeConfirmation`.

`MessageId`

Un identificador único universal (UUID), único para cada mensaje publicado. En los mensajes que Amazon SNS reenvía durante un reintento, se usa el ID de mensaje original.

`Token`

Valor que puede utilizar con la acción `ConfirmSubscription` para volver a confirmar la suscripción. También puede visitar simplemente `SubscribeURL`.

`TopicArn`

Nombre de recurso de Amazon (ARN) del tema del que el punto de enlace ha cancelado su suscripción.

`Message`

Cadena que describe el mensaje. Para la confirmación de la cancelación de suscripción, esta cadena tiene el aspecto siguiente:

```
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`

Dirección URL que debe visitar para volver a confirmar la suscripción. O bien, puede utilizar `Token` con la acción `ConfirmSubscription` para volver a confirmar la suscripción.

`Timestamp`

Hora (GMT) de envío de la cancelación de la suscripción.

`SignatureVersion`

Versión de la firma de Amazon SNS utilizada.

- Si `SignatureVersion` es 1, `Signature` es una firma `SHA1withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Type`, `Timestamp` y `TopicArn`.
- Si `SignatureVersion` es 2, `Signature` es una firma `SHA256withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Type`, `Timestamp` y `TopicArn`.

`Signature`

Firma de `SHA1withRSA` o `SHA256withRSA` codificada en Base64 de los valores `Message`, `MessageId`, `Type`, `Timestamp` y `TopicArn`.

`SigningCertURL`

Dirección URL del certificado que se utilizó para firmar el mensaje.

El mensaje HTTP POST siguiente es un ejemplo de un mensaje de `UnsubscribeConfirmation` a un punto de conexión 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"
}
```

SetSubscriptionAttributespolítica de entrega (formato JSON)

Si envía una solicitud a la acción `SetSubscriptionAttributes` y establece el parámetro `AttributeName` al valor `DeliveryPolicy`, el valor del parámetro `AttributeValue` válido debe ser un objeto JSON válido. Por ejemplo, el siguiente ejemplo establece la política de entrega en 5 reintentos en total.

```
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}}
...
```

Utilice el siguiente formato JSON para el valor del parámetro `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 obtener más información sobre la `SetSubscriptionAttribute` acción, consulta la referencia SetSubscriptionAttributesde la API de Amazon Simple Notification Service. Para obtener más información sobre los encabezados content-type de HTTP compatibles, consulte Creación de una política de entrega HTTP/S.

SetTopicAttributes política de entrega (formato JSON)

Si envía una solicitud a la acción `SetTopicAttributes` y establece el parámetro `AttributeName` al valor `DeliveryPolicy`, el valor del parámetro `AttributeValue` válido debe ser un objeto JSON válido. Por ejemplo, el siguiente ejemplo establece la política de entrega en 5 reintentos en total.

```
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}}}
...
```

Utilice el siguiente formato JSON para el valor del parámetro `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 obtener más información sobre la `SetTopicAttribute` acción, consulta la referencia SetTopicAttributesde la API de Amazon Simple Notification Service. Para obtener más información sobre los encabezados content-type de HTTP compatibles, consulte Creación de una política de entrega HTTP/S.

Distribución ramificada de eventos de Amazon SNS a AWS Event Fork Pipelines

Para el archivado y el análisis de eventos, Amazon SNS recomienda ahora utilizar su integración nativa con Amazon Data Firehose. Puede suscribir las transmisiones de entrega de Firehose a temas de SNS, lo que le permite enviar notificaciones a puntos de enlace de archivado y análisis, como depósitos de Amazon Simple Storage Service (Amazon S3), tablas de Amazon Redshift, Amazon Service (Service) y más. OpenSearch OpenSearch El uso de Amazon SNS con las transmisiones de entrega de Firehose es una solución totalmente gestionada y sin código que no requiere el uso de funciones. AWS Lambda Para obtener más información, consulte Distribución ramificada a los flujos de entrega de Firehose.

Puede usar Amazon SNS para crear aplicaciones basadas en eventos que utilicen los servicios de suscriptor para realizar trabajos de manera automática en respuesta a eventos desencadenados por los servicios de publicador. Este patrón arquitectónico puede hacer que los servicios sean más reutilizables, interoperables y escalables. Sin embargo, puede ser muy laborioso bifurcar el procesamiento de eventos a canalizaciones que cumplan los requisitos comunes de administración de eventos, como el almacenamiento, la copia de seguridad, la búsqueda, el análisis y la reproducción de eventos.

Para acelerar el desarrollo de sus aplicaciones basadas en eventos, puede suscribir canales de gestión de eventos (impulsados por Event Fork AWS Pipelines) a los temas de Amazon SNS. AWS Event Fork Pipelines es un conjunto de aplicaciones anidadas de código abierto, basadas

en el modelo de aplicaciones AWS sin servidor (AWS SAM), que puede implementar directamente desde el paquete AWS Event Fork Pipelines (elija Mostrar aplicaciones que creen funciones de IAM personalizadas o políticas de recursos) en su cuenta. AWS

Para ver un AWS caso de uso de Event Fork Pipelines, consulte. Implementación y prueba de la aplicación de ejemplo de canalizaciones de bifurcación de eventos de Amazon SNS

Temas

- Cómo funciona AWS Event Fork Pipelines
- Implementación de Event Fork Pipelines AWS
- Implementación y prueba de la aplicación de ejemplo de canalizaciones de bifurcación de eventos de Amazon SNS
- Suscripción de AWS Event Fork Pipelines a un tema de Amazon SNS

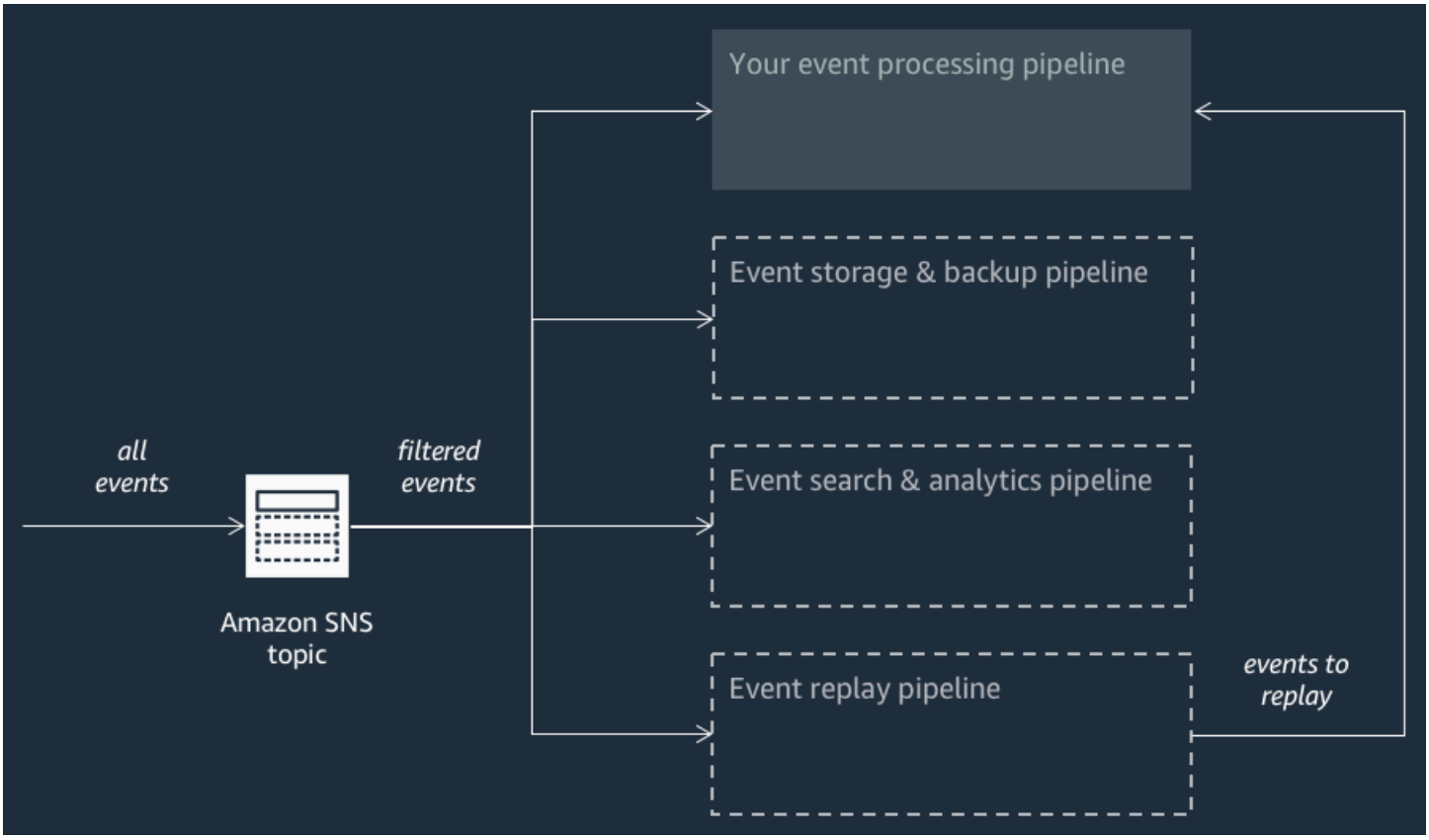
Cómo funciona AWS Event Fork Pipelines

AWS Event Fork Pipelines es un patrón de diseño sin servidor. Sin embargo, también es un conjunto de aplicaciones anidadas sin servidor basadas en AWS SAM (que puede implementar directamente desde el AWS Serverless Application Repository (AWS SAR) al suyo para enriquecer sus Cuenta de AWS plataformas basadas en eventos). Puede implementar estas aplicaciones anidadas de forma individual, según lo requiera su arquitectura.

Temas

- Canalización de almacenamiento y copia de seguridad de eventos
- Canalización de búsqueda y análisis de eventos
- Canalización de reproducción de eventos

El siguiente diagrama muestra una aplicación AWS Event Fork Pipelines complementada con tres aplicaciones anidadas. Puede implementar cualquiera de las canalizaciones del paquete AWS Event Fork Pipelines en el AWS SAR de forma independiente, según lo requiera su arquitectura.



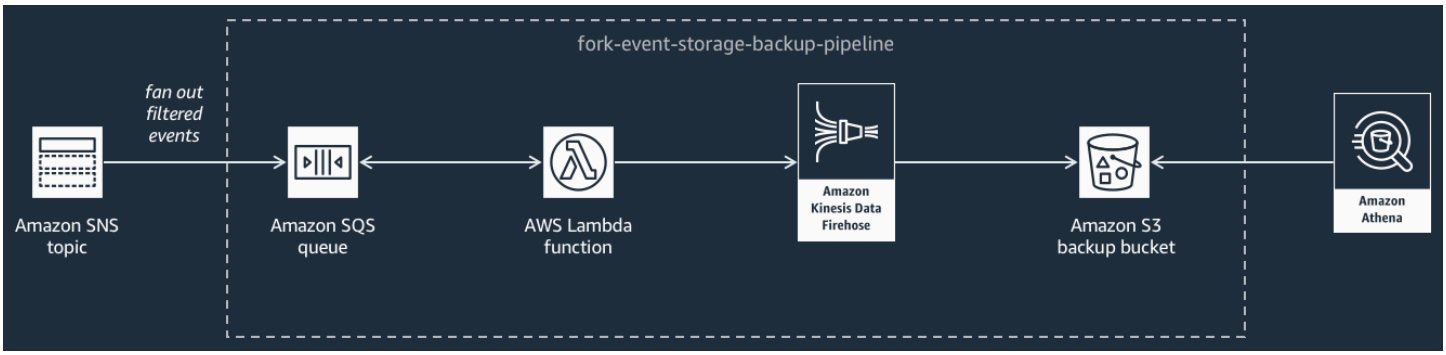
Cada canalización está suscrita al mismo tema de Amazon SNS, lo que le permite procesar eventos en paralelo a medida que se publican en el tema. Cada canalización es independiente y puede establecer su propia política de filtros de suscripción. De este modo, una canalización puede procesar solo un subconjunto de los eventos que le interesan (en lugar de todos los eventos publicados en el tema).

> **Note**
>
> Como colocas las tres canalizaciones de AWS Event Fork junto a las canalizaciones de procesamiento de eventos habituales (es posible que ya estés suscrito a tu tema de Amazon SNS), no necesitas cambiar ninguna parte de tu editor de mensajes actual para aprovechar las canalizaciones de AWS Event Fork en tus cargas de trabajo actuales.

Canalización de almacenamiento y copia de seguridad de eventos

En el siguiente diagrama se muestra la canalización de almacenamiento y copia de seguridad de eventos. Puede suscribir esta canalización a su tema de Amazon SNS para hacer una copia de seguridad automática de los eventos que pasan por su sistema.

Esta canalización se compone de una cola de Amazon SQS que almacena en búfer los eventos publicados por el tema de Amazon SNS, una AWS Lambda función que sondea automáticamente estos eventos de la cola y los envía a una transmisión de Amazon Data Firehose y un depósito de Amazon S3 que realiza copias de seguridad duraderas de los eventos cargados por la transmisión.

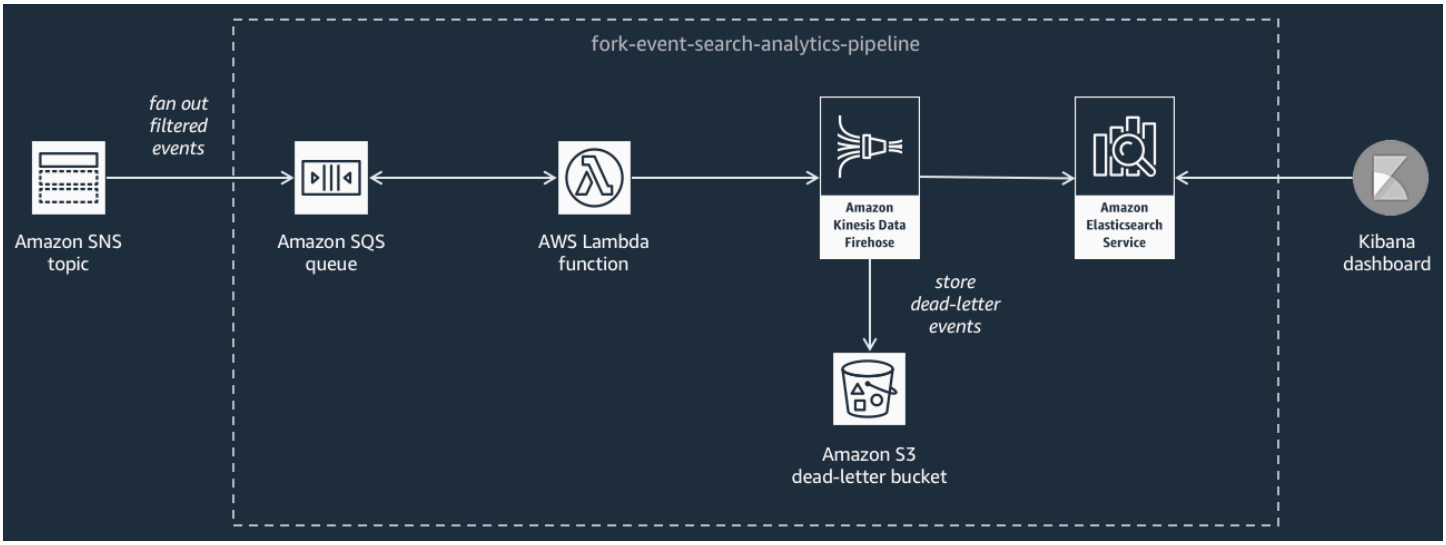


Para optimizar el comportamiento del flujo de Firehose, puede configurarlo para que almacene en búfer, transforme y comprima los eventos antes de cargarlos en el bucket. A medida que se carguen los eventos, puede utilizar Amazon Athena para consultar el bucket mediante consultas de SQL estándar. También puede configurar la canalización para reutilizar un bucket de Amazon S3 existente o crear uno nuevo.

Canalización de búsqueda y análisis de eventos

En el siguiente diagrama se muestra la canalización de búsqueda y análisis de eventos. Puede suscribir esta canalización a su tema de Amazon SNS para indexar los eventos que pasan por su sistema en un dominio de búsqueda y, a continuación, ejecutar análisis en ellos.

Esta canalización se compone de una cola de Amazon SQS que almacena en búfer los eventos publicados por el tema de Amazon SNS, una AWS Lambda función que sondea los eventos de la cola y los envía a una transmisión de Amazon Data Firehose, un OpenSearch dominio de Amazon Service que indexa los eventos cargados por la transmisión de Firehose y un depósito de Amazon S3 que almacena los eventos con letra muerta que no se pueden indexar en el dominio de búsqueda.



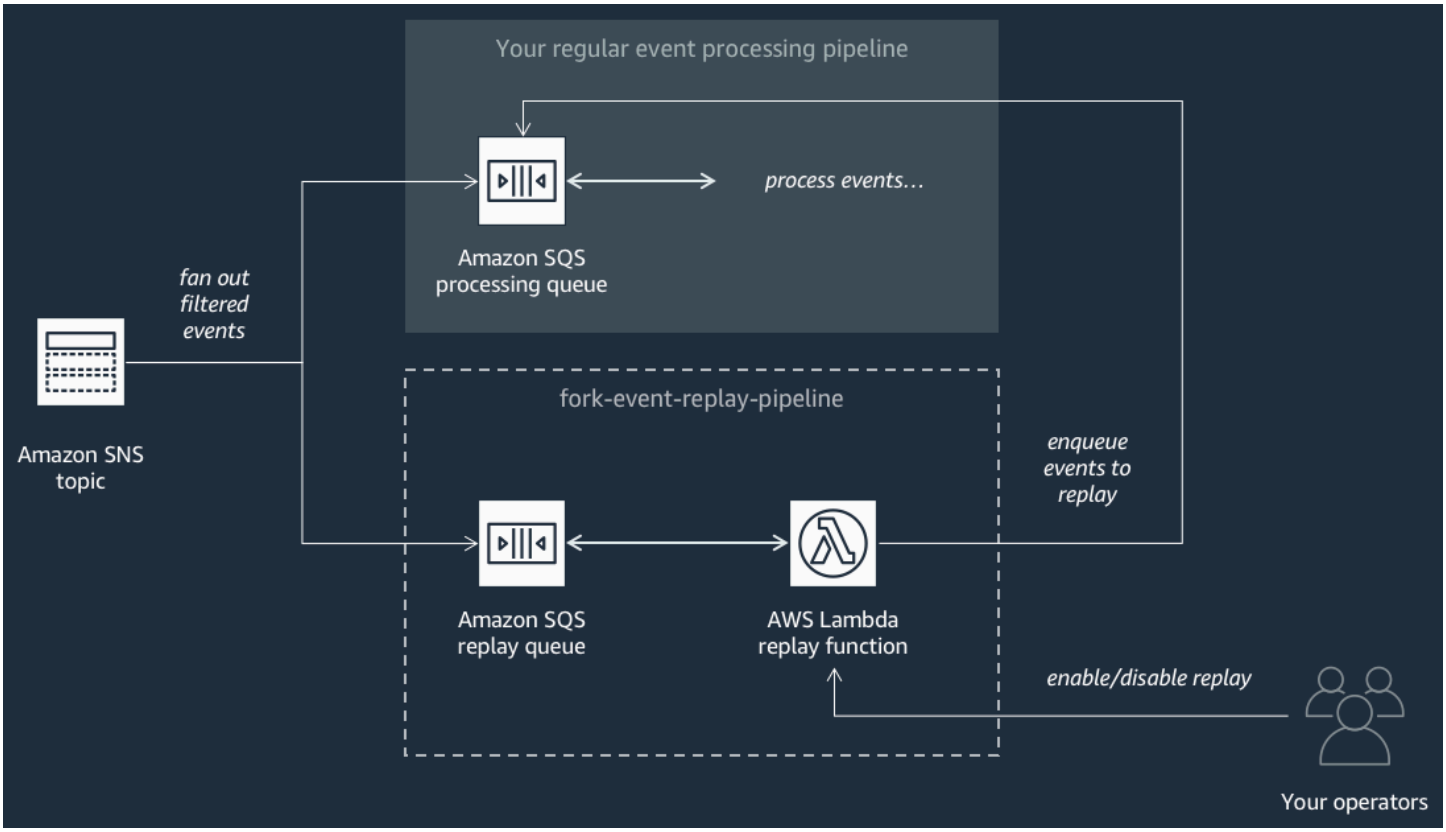
Si desea ajustar el flujo de Firehose en cuanto al almacenamiento en búfer, la transformación y la compresión de eventos, puede configurar esta canalización.

También puedes configurar si la canalización debe reutilizar un OpenSearch dominio existente en el tuyo Cuenta de AWS o crear uno nuevo para ti. A medida que los eventos se indexan en el dominio de búsqueda, puede utilizar Kibana para ejecutar análisis de sus eventos y actualizar los paneles visuales en tiempo real.

Canalización de reproducción de eventos

En el siguiente diagrama se muestra la canalización de reproducción de eventos. Para registrar los eventos que ha procesado el sistema en los últimos 14 días (por ejemplo, cuando su plataforma necesita recuperarse de un error), puede suscribir esta canalización a su tema de Amazon SNS y, a continuación, volver a procesar los eventos.

Esta canalización está compuesta por una cola de Amazon SQS que almacena en búfer los eventos publicados por el tema de Amazon SNS y una AWS Lambda función que sondea los eventos de la cola y los redirige a su canalización de procesamiento de eventos normal, que también está suscrita a su tema.



> **Note**
>
> De forma predeterminada, la característica de reproducción está deshabilitada, por lo que los eventos no se redireccionan. Si necesita volver a procesar los eventos, debe habilitar la cola de reproducción de Amazon SQS como fuente de eventos para la característica de reproducción de AWS Lambda .

Implementación de Event Fork Pipelines AWS

La suite AWS Event Fork Pipelines (elija Mostrar aplicaciones que creen funciones de IAM o políticas de recursos personalizadas) está disponible como un grupo de aplicaciones públicas en el AWS Serverless Application Repository, desde donde puede desplegarlas y probarlas manualmente mediante la AWS Lambda consola. Para obtener información sobre la implementación de canalizaciones mediante la AWS Lambda consola, consulte. Suscripción de AWS Event Fork Pipelines a un tema de Amazon SNS

En un escenario de producción, recomendamos incrustar AWS Event Fork Pipelines en la plantilla SAM general de AWS la aplicación. La función de aplicación anidada le permite hacerlo añadiendo

el recurso `AWS::Serverless::Application` a su plantilla de AWS SAM, haciendo referencia al AWS SAR `ApplicationId` y al de la aplicación anidada. `SemanticVersion`

Por ejemplo, puedes usar Event Storage and Backup Pipeline como una aplicación anidada añadiendo el siguiente fragmento de código YAML a la `Resources` sección de tu plantilla 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
```

Al especificar los valores de los parámetros, puede utilizar funciones AWS CloudFormation intrínsecas para hacer referencia a otros recursos de la plantilla. Por ejemplo, en el fragmento de código YAML anterior, el `TopicArn` parámetro hace referencia al `AWS::SNS::Topic` recurso`MySNSTopic`, definido en otra parte de la plantilla. AWS SAM Para obtener más información, consulte la Referencia de funciones intrínsecas en la Guía del usuario de AWS CloudFormation .

> **Note**
>
> La página de la AWS Lambda consola de la aplicación AWS SAR incluye el botón Copiar como recurso SAM, que copia en el portapapeles el YAML necesario para anidar una aplicación AWS SAR.

Implementación y prueba de la aplicación de ejemplo de canalizaciones de bifurcación de eventos de Amazon SNS

Para acelerar el desarrollo de sus aplicaciones basadas en eventos, puede suscribir canales de gestión de eventos (impulsados por Event Fork AWS Pipelines) a temas de Amazon SNS. AWS Event Fork Pipelines es un conjunto de aplicaciones anidadas de código abierto, basadas en el modelo de aplicaciones AWS sin servidor (AWS SAM), que puede implementar directamente desde

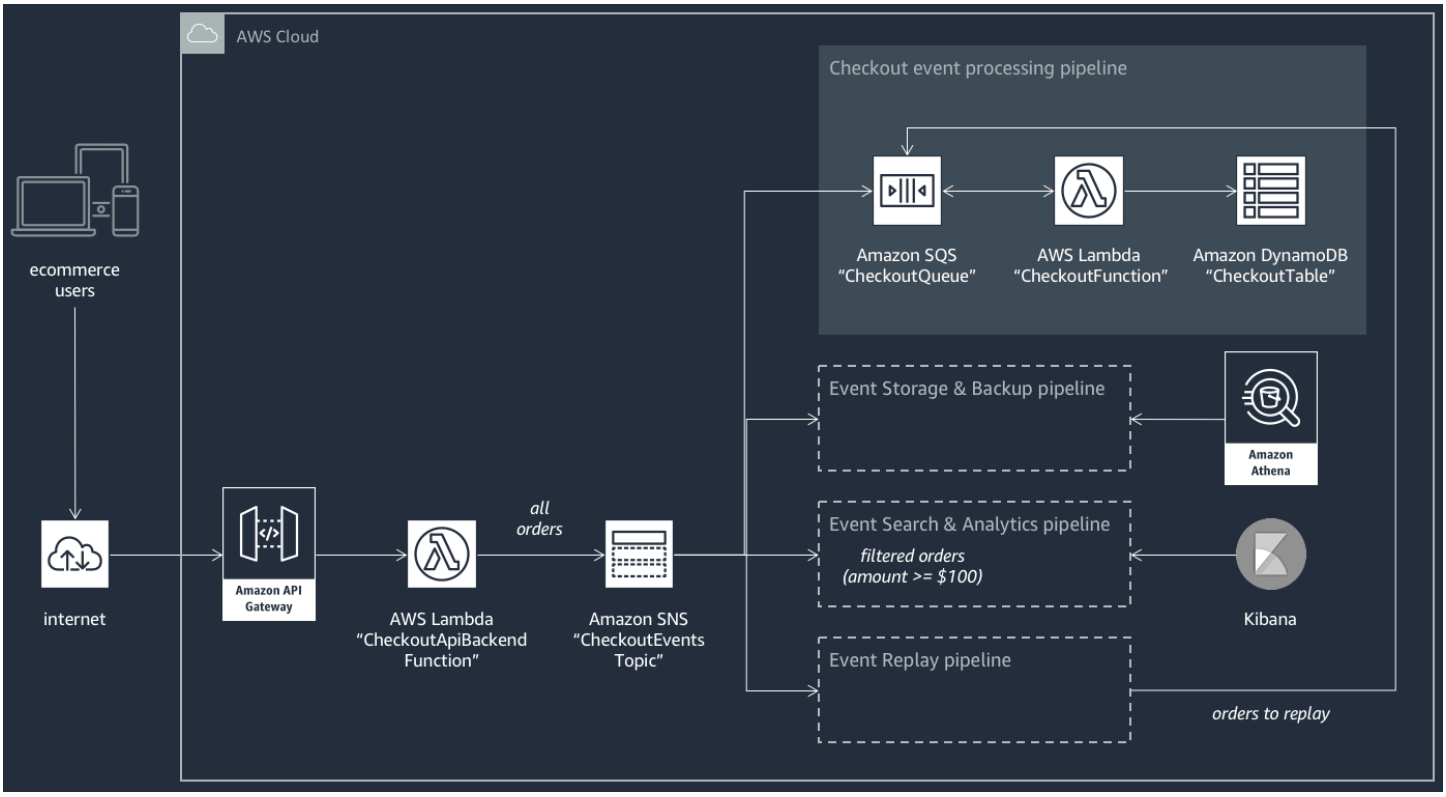
el paquete AWS Event Fork Pipelines (elija Mostrar aplicaciones que creen funciones de IAM personalizadas o políticas de recursos) en su cuenta. AWS Para obtener más información, consulte Cómo funciona AWS Event Fork Pipelines.

En esta página, se muestra cómo puede utilizarla AWS Management Console para implementar y probar la aplicación de ejemplo Event Fork Pipelines. AWS

> **Important**
>
> Para evitar incurrir en costes no deseados una vez que termine de implementar la aplicación de ejemplo AWS Event Fork Pipelines, elimine su pila. AWS CloudFormation Para obtener más información, consulte Eliminación de una pila en la consola de AWS CloudFormation en la Guía del usuario de AWS CloudFormation .

AWS Ejemplo de caso de uso de Event Fork Pipelines

El siguiente escenario describe una aplicación de comercio electrónico sin servidor y basada en eventos que utiliza AWS Event Fork Pipelines. Puede utilizar este ejemplo de aplicación de comercio electrónico en la consola AWS Serverless Application Repository y, a continuación, implementarla en la suya Cuenta de AWS mediante la AWS Lambda consola, donde podrá probarla y examinar su código fuente. GitHub



Esta aplicación de comercio electrónico recibe los pedidos de los compradores a través de una RESTful API alojada en API Gateway y respaldada por la AWS Lambda función`CheckoutApiBackendFunction`. Con esta función, se publican todos los pedidos recibidos en un tema de Amazon SNS llamado `CheckoutEventsTopic` que, a su vez, distribuye los pedidos a cuatro canalizaciones diferentes.

La primera canalización es la canalización normal de procesamiento de pago que ha diseñado e implementado el propietario de la aplicación de comercio electrónico. Esta canalización incluye la cola Amazon SQS `CheckoutQueue` que almacena en búfer todos los pedidos recibidos, una AWS Lambda función denominada `CheckoutFunction` que sondea la cola para procesar estos pedidos y la tabla DynamoDB que guarda de forma segura todos los pedidos realizados. `CheckoutTable`

Aplicando Event Fork AWS Pipelines

Los componentes de la aplicación de comercio electrónico controlan la lógica de negocio central. Sin embargo, el propietario de la aplicación de comercio electrónico también debe tener en cuenta lo siguiente:

- Conformidad: copias de seguridad seguras y comprimidas, encriptadas en reposo, y saneamiento de la información confidencial.

- Resistencia: reproducción de los pedidos más recientes en caso de que se interrumpa el proceso de gestión logística.
- Capacidad de búsqueda: ejecutar análisis y generar métricas en pedidos realizados.

En lugar de implementar esta lógica de procesamiento de eventos, el propietario de la aplicación puede suscribir AWS Event Fork Pipelines al tema `CheckoutEventsTopic` Amazon SNS.

- Canalización de almacenamiento y copia de seguridad de eventos se ha configurado para transformar los datos a fin de eliminar los detalles de la tarjeta de crédito, almacenar en búfer los datos durante 60 segundos, comprimirlos mediante GZIP y cifrarlos mediante la clave administrada por el cliente predeterminada para Amazon S3. Esta clave es administrada AWS y alimentada por AWS Key Management Service ()AWS KMS.

  Para obtener más información, consulte Choose Amazon S3 For Your Destination, Amazon Data Firehose Data Transformation y Configure Settings en la Guía para desarrolladores de Amazon Data Firehose.
- Canalización de búsqueda y análisis de eventos se ha configurado con una duración de reintento de índice de 30 segundos, un bucket para almacenar los pedidos que no se han indexado en el dominio de búsqueda y una política de filtro para restringir el conjunto de pedidos indexados.

  Para obtener más información, consulte Elija un OpenSearch servicio para su destino en la Guía para desarrolladores de Amazon Data Firehose.
- Canalización de reproducción de eventos se ha configurado con la parte de cola de Amazon SQS de la canalización normal de procesamiento de pedidos que ha diseñado e implementado el propietario de la aplicación de comercio electrónico.

  Para obtener más información, consulte Nombre y URL de la cola en la Guía para desarrolladores de Amazon Simple Queue Service.

La siguiente política de filtro de JSON se establece en la configuración para la canalización de búsqueda y análisis de eventos. Solo coincide con los pedidos entrantes en los que el importe total es de 100 USD o superior. Para obtener más información, consulte Filtrado de mensajes en Amazon SNS.

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

Al utilizar el patrón AWS Event Fork Pipelines, el propietario de la aplicación de comercio electrónico puede evitar la sobrecarga de desarrollo que suele implicar la programación de una lógica indiferenciada para la gestión de eventos. En su lugar, puede implementar AWS Event Fork Pipelines directamente desde dentro de ella. AWS Serverless Application Repository Cuenta de AWS

Paso 1: implementar la aplicación de ejemplo de Amazon SNS

1. Inicie sesión en la consola de AWS Lambda.
2. En el panel de navegación, elija Functions (Funciones) y, a continuación, Create function (Crear función).
3. En la página Create function (Crear función), proceda del modo siguiente:
   a. Elija Examinar el repositorio de aplicaciones sin servidor, Aplicaciones públicas, Mostrar aplicaciones que crean roles de IAM personalizados o políticas de recursos.
   b. Busque `fork-example-ecommerce-checkout-api` y, a continuación, elija la aplicación.
4. En la página fork-example-ecommerce-checkout-api, haga lo siguiente:
   a. En la sección Application settings (Configuración de la aplicación), escriba el valor de Application name (Nombre de aplicación) (por ejemplo, `fork-example-ecommerce-my-app`).

      > **Note**
      >
      > - Para encontrar fácilmente sus recursos más tarde, mantenga el prefijo `fork-example-ecommerce`.
      > - Para cada implementación, el nombre de la aplicación debe ser único. Si reutilizas el nombre de una aplicación, la implementación solo actualizará la AWS CloudFormation pila implementada anteriormente (en lugar de crear una nueva).

   b. (Opcional) Introduzca una de las siguientes LogLevelconfiguraciones para la ejecución de la función Lambda de la aplicación:
      - `DEBUG`
      - `ERROR`
      - `INFO` (predeterminado)
      - `WARNING`

5. Elija I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Confirmo que esta aplicación crea políticas de recursos o roles de IAM personalizados e implementa aplicaciones anidadas) y, a continuación, en la parte inferior de la página, elija Deploy (Implementar).

En la *my-app* página Estado de despliegue de fork-example-ecommerce -, Lambda muestra el estado Su aplicación se está desplegando.

En la sección Recursos, AWS CloudFormation comienza a crear la pila y muestra el estado CREATE_IN_PROGRESS de cada recurso. Cuando se completa el proceso, muestra el estado CREATE_COMPLETE. AWS CloudFormation

> **Note**
>
> La implementación de todos los recursos puede tardar entre 20 y 30 minutos.

Cuando se haya completado la implementación, Lambda muestra el estado La aplicación se ha implementado.

Paso 2: ejecutar la aplicación de ejemplo vinculada a SNS

1. En la AWS Lambda consola, en el panel de navegación, selecciona Aplicaciones.
2. En la página Applications (Aplicaciones), en el campo de búsqueda, busque `serverlessrepo-fork-example-ecommerce-`*`my-app`* y, a continuación, seleccione la aplicación.
3. En la sección Resources (Recursos), haga lo siguiente:

   a. Para buscar el recurso cuyo tipo es ApiGatewayRestApi, por ejemplo, ordene los recursos por tipo y`ServerlessRestApi`, a continuación, amplíe el recurso.

   b. Se muestran dos recursos anidados, de los tipos ApiGatewayDeployment y ApiGatewayStage.

   c. Copie el enlace Prod API endpoint (Punto de enlace de la API de Prod) y añádale `/checkout`, por ejemplo:

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

4. Copie el siguiente código JSON a un archivo 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 una solicitud HTTPS a su punto de enlace de la API, pase la carga de evento de muestra como entrada mediante la ejecución de un comando `curl`, por ejemplo:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

La API devuelve la siguiente respuesta vacía, lo que indica que la ejecución es correcta:

```
{ }
```

Paso 3: verificar el rendimiento de la aplicación y las canalizaciones de Amazon SNS

Paso 1: verificar la ejecución de la canalización de pago de ejemplo

1. Inicie sesión en la consola de Amazon DynamoDB.
2. En el panel de navegación, elija Tables (Tablas).
3. Busque `serverlessrepo-fork-example` y elija `CheckoutTable`.
4. En la página de detalles de tabla, elija Items (Elementos) y, a continuación, el elemento creado.

   Se muestran los atributos almacenados.

Paso 2: verificar la ejecución de la canalización de almacenamiento y copia de seguridad de eventos

1. Inicie sesión en la consola de Amazon S3.
2. En el panel de navegación, elija Buckets.

3. Busque `serverlessrepo-fork-example` y, a continuación, elija `CheckoutBucket`.
4. Navegue por la jerarquía de directorios hasta que encuentre un archivo con la extensión `.gz`.
5. Para descargar el archivo, elija Actions (Acciones), Open (Abrir).
6. La canalización se configura con una función Lambda con la que se sanea la información de la tarjeta de crédito por razones de conformidad.

   Para verificar que la carga de JSON almacenada no contiene información de tarjeta de crédito, descomprima el archivo.

Paso 3: verificar la ejecución de la canalización de búsqueda y análisis de eventos

1. Inicie sesión en la consola OpenSearch de servicio.
2. En el panel de navegación, en My domains (Mis dominios), elija su dominio con el prefijo `serverl-analyt`.
3. La canalización se configura con una política de filtro de suscripciones de Amazon SNS con la que se establece una condición de coincidencia numérica.

   Para comprobar que el evento está indexado porque se refiere a una orden cuyo valor es superior a 100 USD, en la ***abcdefgh1ijk*** página serverl-analyt-, seleccione Indices, checkout_events.

Paso 4: verificar la ejecución de la canalización de reproducción de eventos

1. Inicie sesión en la consola de Amazon SQS.
2. En la lista de colas, busque `serverlessrepo-fork-example` y elija `ReplayQueue`.
3. Seleccione Enviar y recibir mensajes.
4. En el cuadro de ***123ABCD4E5F6*** diálogo Enviar y recibir mensajes en fork-example-ecommerce -***my-app***... ReplayP- ReplayQueue -, seleccione Sondear los mensajes.
5. Para verificar que el evento está en cola, seleccione More Details (Más detalles) junto al mensaje que aparece en la cola.

Paso 4: simular un problema y reproducir eventos para la recuperación

Paso 1: habilitar el problema simulado y enviar una segunda solicitud de API

1. Inicie sesión en la consola de AWS Lambda.

2. En el panel de navegación, elija Functions (Funciones).
3. Busque `serverlessrepo-fork-example` y elija `CheckoutFunction`.
4. En el fork-example-ecommerce- - ***my-app*** -... CheckoutFunction ***ABCDEF*** página, en la sección Variables de entorno, defina la variable BUG_ENABLED en true y, a continuación, seleccione Guardar.
5. Copie el siguiente código JSON a un archivo 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 una solicitud HTTPS a su punto de enlace de la API, pase la carga de evento de muestra como entrada mediante la ejecución de un comando `curl`, por ejemplo:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

La API devuelve la siguiente respuesta vacía, lo que indica que la ejecución es correcta:

```
{ }
```

Paso 2: verificar la corrupción de datos simulada

1. Inicie sesión en la consola de Amazon DynamoDB.
2. En el panel de navegación, elija Tables (Tablas).
3. Busque `serverlessrepo-fork-example` y elija `CheckoutTable`.
4. En la página de detalles de tabla, elija Items (Elementos) y, a continuación, el elemento creado.

   Se muestran los atributos almacenados, algunos marcados como CORRUPTED! (Dañados).

Paso 3: deshabilitar el problema simulado

1. Inicie sesión en la consola de AWS Lambda.
2. En el panel de navegación, elija Functions (Funciones).
3. Busque `serverlessrepo-fork-example` y elija `CheckoutFunction`.
4. En el fork-example-ecommerce- - ***`my-app`*** -... CheckoutFunction ***`ABCDEF`*** página, en la sección Variables de entorno, defina la variable BUG_ENABLED en false y, a continuación, seleccione Guardar.

Paso 4: habilitar la reproducción para la recuperación del problema

1. En la AWS Lambda consola, en el panel de navegación, selecciona Funciones.
2. Busque `serverlessrepo-fork-example` y elija `ReplayFunction`.
3. Expanda la sección Designer (Diseñador), elija el mosaico SQS y, a continuación, en la sección SQS, elija Enabled (Habilitado).

> **Note**
>
> El desencadenador de fuentes de eventos de Amazon SQS tarda aproximadamente un minuto en habilitarse.

4. Seleccione Guardar.
5. Para ver los atributos recuperados, vuelva a la consola de Amazon DynamoDB.
6. Para deshabilitar la reproducción, regrese a la AWS Lambda consola y desactive el activador de la fuente de eventos de Amazon SQS para. `ReplayFunction`

Suscripción de AWS Event Fork Pipelines a un tema de Amazon SNS

Para acelerar el desarrollo de sus aplicaciones basadas en eventos, puede suscribir canales de gestión de eventos (impulsados por Event Fork AWS Pipelines) a temas de Amazon SNS. AWS Event Fork Pipelines es un conjunto de aplicaciones anidadas de código abierto, basadas en el modelo de aplicaciones AWS sin servidor (AWS SAM), que puede implementar directamente desde el paquete AWS Event Fork Pipelines (elija Mostrar aplicaciones que creen funciones de IAM personalizadas o políticas de recursos) en su cuenta. AWS Para obtener más información, consulte Cómo funciona AWS Event Fork Pipelines.

En esta sección se muestra cómo se puede utilizar AWS Management Console para implementar una canalización y, a continuación, suscribir AWS Event Fork Pipelines a un tema de Amazon SNS. Antes de comenzar, cree un tema de Amazon SNS.

Para eliminar los recursos que componen una canalización, busque la canalización en la página de aplicaciones de la AWS Lambda consola, expanda la sección de plantillas de SAM, elija CloudFormationpila y, a continuación, elija Otras acciones, eliminar pila.

Implementación y suscripción de la canalización de almacenamiento y copia de seguridad de eventos en Amazon SNS

> Para el archivado y el análisis de eventos, Amazon SNS recomienda ahora utilizar su integración nativa con Amazon Data Firehose. Puede suscribir las transmisiones de entrega de Firehose a temas de SNS, lo que le permite enviar notificaciones a puntos de enlace de archivado y análisis, como depósitos de Amazon Simple Storage Service (Amazon S3), tablas de Amazon Redshift, Amazon Service (Service) y más. OpenSearch OpenSearch El uso de Amazon SNS con las transmisiones de entrega de Firehose es una solución totalmente gestionada y sin código que no requiere el uso de funciones. AWS Lambda Para obtener más información, consulte Distribución ramificada a los flujos de entrega de Firehose.

En este tutorial, se muestra cómo implementar la canalización de almacenamiento y copia de seguridad de eventos y suscribirla a un tema de Amazon SNS. Este proceso convierte automáticamente la AWS SAM plantilla asociada a la canalización en una AWS CloudFormation pila y, a continuación, implementa la pila en la suya. Cuenta de AWS Este proceso también crea y configura el conjunto de recursos que componen la canalización de almacenamiento y copia de seguridad de eventos, incluidos los siguientes:

- Cola de Amazon SQS
- Función de Lambda
- Flujo de entrega de Firehose
- Bucket de copia de seguridad de Amazon S3

Para obtener más información sobre cómo configurar una transmisión con un bucket de Amazon S3 como destino, consulte la `S3DestinationConfiguration` referencia de la API Amazon Data Firehose.

Para obtener más información sobre la transformación de eventos y la configuración del almacenamiento en búfer, la compresión y el cifrado de eventos, consulte Creating an Amazon Data Firehose Delivery Stream en la Guía para desarrolladores de Amazon Data Firehose.

Para obtener más información sobre el filtrado de eventos, consulte Políticas de filtro de suscripciones de Amazon SNS en esta guía.

1. Inicie sesión en la consola de AWS Lambda.
2. En el panel de navegación, elija Functions (Funciones) y, a continuación, Create function (Crear función).
3. En la página Create function (Crear función), proceda del modo siguiente:

   a. Elija Examinar el repositorio de aplicaciones sin servidor, Aplicaciones públicas, Mostrar aplicaciones que crean roles de IAM personalizados o políticas de recursos.

   b. Busque `fork-event-storage-backup-pipeline` y, a continuación, elija la aplicación.
4. En la página fork-event-storage-backup-pipeline, haga lo siguiente:

   a. En la sección Application settings (Configuración de la aplicación), escriba el valor de Application name (Nombre de aplicación) (por ejemplo, `my-app-backup`).

   > **Note**
   >
   > - Para cada implementación, el nombre de la aplicación debe ser único. Si reutilizas el nombre de una aplicación, la implementación solo actualizará la AWS CloudFormation pila implementada anteriormente (en lugar de crear una nueva).

   b. (Opcional) Para BucketArn, introduzca el ARN del bucket de Amazon S3 en el que se cargan los eventos entrantes. Si no introduce ningún valor, se crea un nuevo bucket de Amazon S3 en su AWS cuenta.

   c. (Opcional) Para DataTransformationFunctionArn, introduzca el ARN de la función Lambda a través de la cual se transforman los eventos entrantes. Si no escribe un valor, se deshabilita la transformación de datos.

   d. (Opcional) Introduzca una de las siguientes LogLevelconfiguraciones para la ejecución de la función Lambda de la aplicación:

      - `DEBUG`
      - `ERROR`

- `INFO` (predeterminado)
- `WARNING`

e. Para TopicArn, introduzca el ARN del tema de Amazon SNS al que se va a suscribir esta instancia de la canalización de bifurcación.

f. (Opcional) Para StreamBufferingIntervalInSecondse StreamBufferingSizeInMBs, introduzca los valores para configurar el almacenamiento en búfer de los eventos entrantes. Si no escribe ningún valor, se utilizan 300 segundos y 5 MB.

g. (Opcional) Introduzca uno de los siguientes StreamCompressionFormatajustes para comprimir los eventos entrantes:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (predeterminado)
- `ZIP`

h. (Opcional) Para StreamPrefix, introduzca el prefijo de cadena para nombrar los archivos almacenados en el bucket de copias de seguridad de Amazon S3. Si no escribe un valor, no se usa ningún prefijo.

i. (Opcional) Para SubscriptionFilterPolicy, introduzca la política de filtrado de suscripciones de Amazon SNS, en formato JSON, que se utilizará para filtrar los eventos entrantes. La política de filtrado decide qué eventos se indexan en el índice de OpenSearch servicios. Si no escribe ningún valor, no se utiliza el filtrado (se indexan todos los eventos).

j. (Opcional) Para SubscriptionFilterPolicyScope, introduzca la cadena `MessageBody` o `MessageAttributes` para habilitar el filtrado de mensajes basado en la carga útil o en los atributos.

k. Elija I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Confirmo que esta aplicación crea políticas de recursos o roles de IAM personalizados e implementa aplicaciones anidadas) y, a continuación, elija Deploy (Implementar).

En la *`my-app`* página Estado de despliegue de, Lambda muestra el estado Su aplicación se está desplegando.

En la sección Recursos, AWS CloudFormation comienza a crear la pila y muestra el estado CREATE_IN_PROGRESS de cada recurso. Cuando se completa el proceso, muestra el estado CREATE_COMPLETE. AWS CloudFormation

Cuando se haya completado la implementación, Lambda muestra el estado La aplicación se ha implementado.

Los mensajes publicados en su tema de Amazon SNS se almacenan automáticamente en el bucket de backup de Amazon S3 aprovisionado por Event Storage and Backup Pipeline.

Implementación y suscripción de la canalización de búsqueda y análisis de eventos en Amazon SNS

Para el archivado y el análisis de eventos, Amazon SNS recomienda ahora utilizar su integración nativa con Amazon Data Firehose. Puede suscribir las transmisiones de entrega de Firehose a temas de SNS, lo que le permite enviar notificaciones a puntos de enlace de archivado y análisis, como depósitos de Amazon Simple Storage Service (Amazon S3), tablas de Amazon Redshift, Amazon Service (Service) y más. OpenSearch OpenSearch El uso de Amazon SNS con las transmisiones de entrega de Firehose es una solución totalmente gestionada y sin código que no requiere el uso de funciones. AWS Lambda Para obtener más información, consulte Distribución ramificada a los flujos de entrega de Firehose.

En este tutorial, se muestra cómo implementar la canalización de búsqueda y análisis de eventos y suscribirla a un tema de Amazon SNS. Este proceso convierte automáticamente la AWS SAM plantilla asociada a la canalización en una AWS CloudFormation pila y, a continuación, implementa la pila en la suya. Cuenta de AWS Este proceso también crea y configura el conjunto de recursos que componen la canalización de búsqueda y análisis de eventos, incluidos los siguientes:

- Cola de Amazon SQS
- Función de Lambda
- Flujo de entrega de Firehose
- Dominio OpenSearch de Amazon Service
- Bucket de Amazon S3 de mensajes fallidos

Para obtener más información sobre la configuración de un flujo con un índice como destino, consulte `ElasticsearchDestinationConfiguration` en la Referencia de la API de Amazon Data Firehose.

Para obtener más información sobre la transformación de eventos y la configuración del almacenamiento en búfer, la compresión y el cifrado de eventos, consulte Creating an Amazon Data Firehose Delivery Stream en la Guía para desarrolladores de Amazon Data Firehose.

Para obtener más información sobre el filtrado de eventos, consulte Políticas de filtro de suscripciones de Amazon SNS en esta guía.

1. Inicie sesión en la consola de AWS Lambda.
2. En el panel de navegación, elija Functions (Funciones) y, a continuación, Create function (Crear función).
3. En la página Create function (Crear función), proceda del modo siguiente:

   a. Elija Examinar el repositorio de aplicaciones sin servidor, Aplicaciones públicas, Mostrar aplicaciones que crean roles de IAM personalizados o políticas de recursos.

   b. Busque `fork-event-search-analytics-pipeline` y, a continuación, elija la aplicación.
4. En la página fork-event-search-analytics-pipeline, haga lo siguiente:

   a. En la sección Application settings (Configuración de la aplicación), escriba el valor de Application name (Nombre de aplicación) (por ejemplo, `my-app-search`).

   > **Note**
   >
   > Para cada implementación, el nombre de la aplicación debe ser único. Si reutilizas el nombre de una aplicación, la implementación solo actualizará la AWS CloudFormation pila implementada anteriormente (en lugar de crear una nueva).

   b. (Opcional) Para DataTransformationFunctionArn, introduzca el ARN de la función Lambda utilizada para transformar los eventos entrantes. Si no escribe un valor, se deshabilita la transformación de datos.

   c. (Opcional) Introduzca una de las siguientes LogLevelconfiguraciones para la ejecución de la función Lambda de la aplicación:

      - `DEBUG`

- `ERROR`
- `INFO` (predeterminado)
- `WARNING`

d. (Opcional) Para SearchDomainArn, introduzca el ARN del dominio del OpenSearch servicio, un clúster que configura la funcionalidad de procesamiento y almacenamiento necesaria. Si no escribe ningún valor, se creará un nuevo dominio con la configuración predeterminada.

e. Para TopicArn, introduzca el ARN del tema de Amazon SNS al que se va a suscribir esta instancia de la canalización de bifurcación.

f. Para SearchIndexName, introduzca el nombre del índice de OpenSearch servicios para la búsqueda y el análisis de eventos.

> **Note**
>
> Las siguientes cuotas se aplican a los nombres de índice:
>
> - No pueden incluir letras mayúsculas
> - No pueden incluir los siguientes caracteres: `\ / * ? " < > | ` , #`
> - No pueden comenzar por los siguientes caracteres: `- + _`
> - No pueden ser los siguientes: `. ..`
> - No pueden tener más de 80 caracteres
> - No pueden tener más de 255 bytes
> - No puede contener dos puntos (de OpenSearch Service 7.0)

g. (Opcional) Introduzca una de las siguientes SearchIndexRotationPeriodconfiguraciones para el período de rotación del índice de OpenSearch servicios:

- `NoRotation` (predeterminado)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

La rotación de índice agrega una marca temporal al nombre del índice, lo que facilita el vencimiento de los datos antiguos.

h. Para SearchTypeName, introduzca el nombre del tipo de OpenSearch servicio para organizar los eventos en un índice.

> **Note**
>
> - OpenSearch Los nombres de los tipos de servicio pueden contener cualquier carácter (excepto bytes nulos), pero no pueden empezar por él_.
> - En el OpenSearch caso de Service 6.x, solo puede haber un tipo por índice. Si especifica un tipo nuevo para un índice existente que ya tiene otro tipo, Firehose devuelve un error de tiempo de ejecución.

i. (Opcional) Para StreamBufferingIntervalInSecondsy StreamBufferingSizeInMBs, introduzca los valores para configurar el almacenamiento en búfer de los eventos entrantes. Si no escribe ningún valor, se utilizan 300 segundos y 5 MB.

j. (Opcional) Introduzca uno de los siguientes StreamCompressionFormatajustes para comprimir los eventos entrantes:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (predeterminado)
- `ZIP`

k. (Opcional) Para StreamPrefix, introduzca el prefijo de cadena para nombrar los archivos almacenados en el depósito de letras muertas de Amazon S3. Si no escribe un valor, no se usa ningún prefijo.

l. (Opcional) Para StreamRetryDurationInSecons, introduzca la duración del reintento en los casos en que Firehose no pueda indexar los eventos en OpenSearch el índice de servicios. Si no escribe un valor, se usan 300 segundos.

m. (Opcional) Para SubscriptionFilterPolicy, introduzca la política de filtrado de suscripciones de Amazon SNS, en formato JSON, que se utilizará para filtrar los eventos entrantes. La política de filtrado decide qué eventos se indexan en el índice de OpenSearch servicios. Si no escribe ningún valor, no se utiliza el filtrado (se indexan todos los eventos).

n. Elija I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Confirmo que esta aplicación crea políticas de recursos o roles de IAM personalizados e implementa aplicaciones anidadas) y, a continuación, elija Deploy (Implementar).

En la *my-app-search* página Estado de despliegue de, Lambda muestra el estado Su aplicación se está desplegando.

En la sección Recursos, AWS CloudFormation comienza a crear la pila y muestra el estado CREATE_IN_PROGRESS de cada recurso. Cuando se completa el proceso, muestra el estado CREATE_COMPLETE. AWS CloudFormation

Cuando se haya completado la implementación, Lambda muestra el estado La aplicación se ha implementado.

Los mensajes publicados en tu tema de Amazon SNS se indexan automáticamente en el índice de OpenSearch servicios proporcionado por la canalización de búsqueda y análisis de eventos. Si la canalización no puede indexar un evento, lo almacena en un depósito de papel muerto de Amazon S3.

Implementación de la canalización de reproducción de eventos con la integración de Amazon SNS

En este tutorial, se muestra cómo implementar la canalización de reproducción de eventos y suscribirla a un tema de Amazon SNS. Este proceso convierte automáticamente la AWS SAM plantilla asociada a la canalización en una AWS CloudFormation pila y, a continuación, despliega la pila en la suya Cuenta de AWS. Con este proceso, también se crea y configura el conjunto de recursos que componen la canalización de reproducción de eventos, incluida una cola de Amazon SQS y una función Lambda.

Para obtener más información sobre el filtrado de eventos, consulte Políticas de filtro de suscripciones de Amazon SNS en esta guía.

1. Inicie sesión en la consola de AWS Lambda.
2. En el panel de navegación, elija Functions (Funciones) y, a continuación, Create function (Crear función).
3. En la página Create function (Crear función), proceda del modo siguiente:
   a. Elija Examinar el repositorio de aplicaciones sin servidor, Aplicaciones públicas, Mostrar aplicaciones que crean roles de IAM personalizados o políticas de recursos.
   b. Busque `fork-event-replay-pipeline` y, a continuación, elija la aplicación.
4. En la página fork-event-replay-pipeline, haga lo siguiente:

a. En la sección Application settings (Configuración de la aplicación), escriba el valor de Application name (Nombre de aplicación) (por ejemplo, `my-app-replay`).

> **Note**
>
> Para cada implementación, el nombre de la aplicación debe ser único. Si reutilizas el nombre de una aplicación, la implementación solo actualizará la AWS CloudFormation pila implementada anteriormente (en lugar de crear una nueva).

b. (Opcional) Introduzca una de las siguientes LogLevelconfiguraciones para la ejecución de la función Lambda de la aplicación:

- `DEBUG`
- `ERROR`
- `INFO` (predeterminado)
- `WARNING`

c. (Opcional) Para ReplayQueueRetentionPeriodInSeconds, introduzca el tiempo, en segundos, durante el que la cola de reproducción de Amazon SQS guarda el mensaje. Si no escribe un valor, se usan 1 209 600 segundos (14 días).

d. Para TopicArn, introduzca el ARN del tema de Amazon SNS al que se va a suscribir esta instancia de la canalización de bifurcación.

e. Para DestinationQueueName, introduzca el nombre de la cola de Amazon SQS a la que la función de reproducción de Lambda reenvía los mensajes.

f. (Opcional) Para SubscriptionFilterPolicy, introduzca la política de filtrado de suscripciones de Amazon SNS, en formato JSON, que se utilizará para filtrar los eventos entrantes. La política de filtro decide qué eventos se almacenan en búfer para la reproducción. Si no escribe ningún valor, no se utiliza el filtrado (todos los eventos se almacenan en búfer para la reproducción).

g. Elija I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Confirmo que esta aplicación crea políticas de recursos o roles de IAM personalizados e implementa aplicaciones anidadas) y, a continuación, elija Deploy (Implementar).

En la *`my-app-replay`* página Estado de despliegue de, Lambda muestra el estado Su aplicación se está desplegando.

En la sección Recursos, AWS CloudFormation comienza a crear la pila y muestra el estado CREATE_IN_PROGRESS de cada recurso. Cuando se completa el proceso, muestra el estado CREATE_COMPLETE. AWS CloudFormation

Cuando se haya completado la implementación, Lambda muestra el estado La aplicación se ha implementado.

Los mensajes publicados en su tema de Amazon SNS se almacenan en búfer para reproducirlos en la cola de Amazon SQS aprovisionada de manera automática por la canalización de reproducción de eventos.

> **Note**
>
> De forma predeterminada, la reproducción está deshabilitada. Para habilitar la reproducción, vaya a la página de la función en la consola de Lambda, expanda la sección Diseñador, seleccione el mosaico SQS y, a continuación, en la sección SQS, elija Habilitado.

Uso de Amazon EventBridge Scheduler con Amazon SNS

Amazon EventBridge Scheduler es un programador sin servidor que le permite crear, ejecutar y gestionar tareas desde un servicio gestionado centralizado. Con EventBridge Scheduler, puede crear planificaciones mediante expresiones de Cron y Rate para patrones recurrentes, o configurar invocaciones únicas. Puede configurar intervalos de tiempo flexibles para la entrega, definir límites de reintentos y establecer el tiempo máximo de retención para las invocaciones de la API.

En esta página se explica cómo usar EventBridge Scheduler para publicar un mensaje de un tema de Amazon SNS de forma programada.

Configuración del rol de ejecución

Al crear una nueva programación, EventBridge Scheduler debe tener permiso para invocar la operación de la API de destino en su nombre. Estos permisos se otorgan a EventBridge Scheduler mediante una función de ejecución. La política de permisos que adjunta al rol de ejecución de su programación define los permisos necesarios. Estos permisos dependen de la API de destino que desee que invoque EventBridge Scheduler.

Cuando utilizas la consola del EventBridge Scheduler para crear una programación, como en el siguiente procedimiento, EventBridge Scheduler configura automáticamente una función de

ejecución en función del objetivo seleccionado. Si desea crear un cronograma utilizando uno de los EventBridge planificadores SDKs, debe tener un rol de ejecución existente que otorgue los permisos que el EventBridge programador requiere para invocar un destino. AWS CLI AWS CloudFormationPara obtener más información sobre la configuración manual de una función de ejecución para su programación, consulte Configuración de una función de ejecución en la Guía del usuario del Scheduler. EventBridge

Creación de una programación

Creación de una programación con la consola

1. Abre la consola de Amazon EventBridge Scheduler en https://console.aws.amazon.com/scheduler/casa.
2. En la página Programaciones, elija Crear programación.
3. En la página Especificar los detalles de la programación, en la sección Nombre y descripción de la programación, proceda del modo siguiente:

   a. En Nombre de la programación, escriba un nombre para la programación. Por ejemplo, **`MyTestSchedule`**.

   b. (Opcional) En Descripción, escriba una descripción para su programación. Por ejemplo, **`My first schedule`**.

   c. En Grupo de programaciones, elija un grupo de programaciones de la lista desplegable. Si no tiene un grupo, elija predeterminado. Para crear un grupo de programaciones, elija crear mi propia programación.

   Los grupos de programaciones se utilizan para añadir etiquetas a grupos de programaciones.

4. • Elija sus opciones de programación.

Ocurrencia	Haga lo siguiente...	
Programación única Una programación única invoca solo una vez un destino en la fecha y hora que especifique.	En Fecha y hora, realice lo siguiente: • Introduzca una fecha válida con el formato `YYYY/MM/DD` .	

Ocurrencia	Haga lo siguiente...	
	• Introduzca una marca de tiempo con el formato `hh:mm` de 24 horas. • En Zona horaria, elija la zona horaria.	

Ocurrencia	Haga lo siguiente...	
Programación recurrente Una programación recurrente invoca un destino a la velocidad que especifique mediante una expresión cron o de frecuencia.	a. En Tipo de programación, realice una de las siguientes acciones: • Si quiere utilizar una expresión Cron para definir la programación, elija Programación basada en Cron e introduzca la expresión Cron. • Si quiere utilizar una expresión de frecuencia para definir la programación, elija Programación basada en la frecuencia e introduzca la expresión de frecuencia. Para obtener más información sobre las expresiones cron y rate, consulte Tipos de programación en EventBridge Scheduler en la Guía del usuario de Amazon EventBridge Scheduler. b. En Intervalo de tiempo flexible, elija Desactivado para desactivar la opción o elegir	

Ocurrencia	Haga lo siguiente...	
	uno de los periodos de tiempo predefini dos. Por ejemplo, si elige 15 minutos y establece una programación recurrent e para invocar su destino una vez cada hora, la programación se ejecuta 15 minutos después del inicio de cada hora.	

5. (Opcional) Si elige Programación recurrente en el paso anterior, en la sección de Periodo de tiempo, realice lo siguiente:

   a. En Zona horaria, elija una zona horaria.

   b. En Fecha y hora de inicio, introduzca una fecha válida con el formato `YYYY/MM/DD` y, a continuación, especifique una marca de tiempo con el formato `hh:mm` de 24 horas.

   c. En Fecha y hora de finalización, introduzca una fecha válida con el formato `YYYY/MM/DD` y, a continuación, especifique una marca de tiempo con el formato `hh:mm` de 24 horas.

6. Elija Next (Siguiente).

7. En la página Seleccione el destino, elija la operación de AWS API que EventBridge Scheduler invoca:

   a. Elija Publicar Amazon SNS.

   b. En la sección Publicar, selecciona una SNS tema o selecciona Crear nuevo SNS tema.

   c. (Opcional) Introduzca un JSON carga útil. Si no ingresas una carga útil, EventBridge Scheduler usa un evento vacío para invocar la función.

8. Elija Next (Siguiente).

9. En la página Configuración, haga lo siguiente:

   a. Para activar la programación, en Estado de la programación, cambie a Habilitar programación.

b. Para configurar una política de reintentos para su programación, en Política de reintento y cola de mensajes fallidos (DLQ), realice lo siguiente:

   - Cambie a Reintentar.
   - En Duración máxima del evento, introduce el número máximo de horas y minutos que EventBridge Scheduler debe mantener sin procesar un evento.
   - El tiempo máximo es de 24 horas.
   - En Máximo número de reintentos, introduce el número máximo de veces que EventBridge Scheduler reintenta la programación si el objetivo devuelve un error.

     El valor máximo es 185 reintentos.

   Con las políticas de reintentos, si un programa no puede invocar su objetivo, Scheduler vuelve a ejecutar el programa. EventBridge Si se encuentra configurado, debe establecer el tiempo máximo de retención y los reintentos máximos para la programación.

c. Elija dónde guarda el EventBridge Scheduler los eventos no entregados.

Opción Cola de mensajes fallidos (DLQ)	Haga lo siguiente...
No almacenar	Seleccione Ninguno.
Guarda el evento en el mismo Cuenta de AWS lugar en el que estás creando la programación	a. Seleccione Seleccione una cola de Amazon SQS en my Cuenta de AWS as a DLQ. b. Elija el Nombre de recurso de Amazon (ARN) para la cola de Amazon SQS.

Opción Cola de mensajes fallidos (DLQ)	Haga lo siguiente...	
Guarde el evento en un lugar Cuenta de AWS diferente al lugar en el que está creando la programación	a. Elija Especificar una cola de Amazon SQS en otra Cuentas de AWS como DLQ. b. Ingrese el Nombre de recurso de Amazon (ARN) para la cola de Amazon SQS.	

d. Para utilizar una clave administrada por el cliente a fin de cifrar la entrada de destino, en Cifrado, elija Personalizar la configuración de cifrado (avanzado).

   Si elige esta opción, ingrese un ARN de clave de KMS existente o elija Crear una AWS KMS key para navegar hasta la consola de AWS KMS . Para obtener más información sobre cómo EventBridge Scheduler cifra los datos en reposo, consulte Cifrado en reposo en la Guía del usuario de Amazon EventBridge Scheduler.

e. Para que EventBridge Scheduler cree un nuevo rol de ejecución para usted, elija Crear un nuevo rol para este programa. A continuación, ingrese un nombre para el Nombre de rol. Si eliges esta opción, EventBridge Scheduler adjunta al rol los permisos necesarios para el objetivo creado con la plantilla.

10. Elija Next (Siguiente).
11. En la página de Revisar y crear una programación, revise los detalles de su programación. En cada sección, elija Editar para volver a ese paso y editar sus detalles.
12. Elija Crear programación.

    Puede ver una lista de sus programaciones nuevas y existentes en la página Programaciones. En la columna Estado, verifique que su programación nueva se encuentre Habilitada.

Recursos relacionados

Para obtener más información sobre EventBridge Scheduler, consulte lo siguiente:

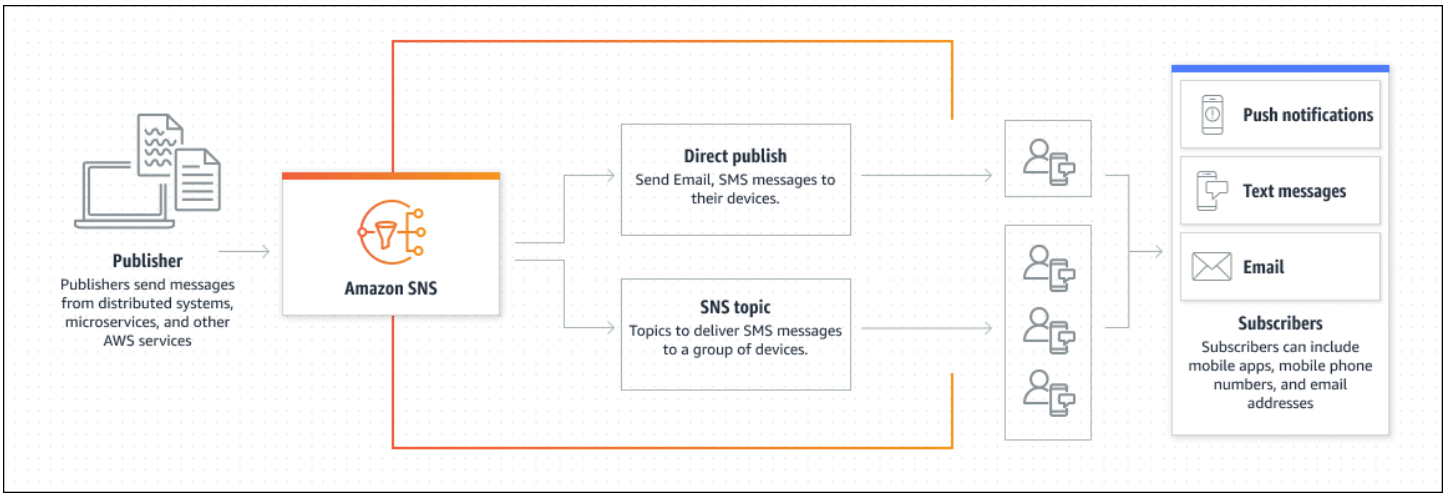
- EventBridge Guía del usuario de Scheduler

- EventBridge Referencia de la API de Scheduler
- EventBridge Precios de Scheduler

Uso de Amazon SNS para la mensajería application-to-person

La mensajería Amazon SNS application-to-person (A2P) le permite enviar notificaciones y alertas directamente a los dispositivos móviles de sus clientes a través de SMS (servicio de mensajes cortos). Con esta característica, puede enviar notificaciones push a aplicaciones móviles, mensajes de texto a números de teléfonos móviles y correos electrónicos de texto sin formato a direcciones de correo electrónico. Además, tiene la flexibilidad de distribuir los mensajes mediante temas para llegar a varios destinatarios o publicar los mensajes directamente en puntos de conexión móviles individuales para una comunicación personalizada.

En este tema se explica cómo usar Amazon SNS para las notificaciones de los usuarios con suscriptores, como aplicaciones móviles, números de teléfono móvil y direcciones de correo electrónico.



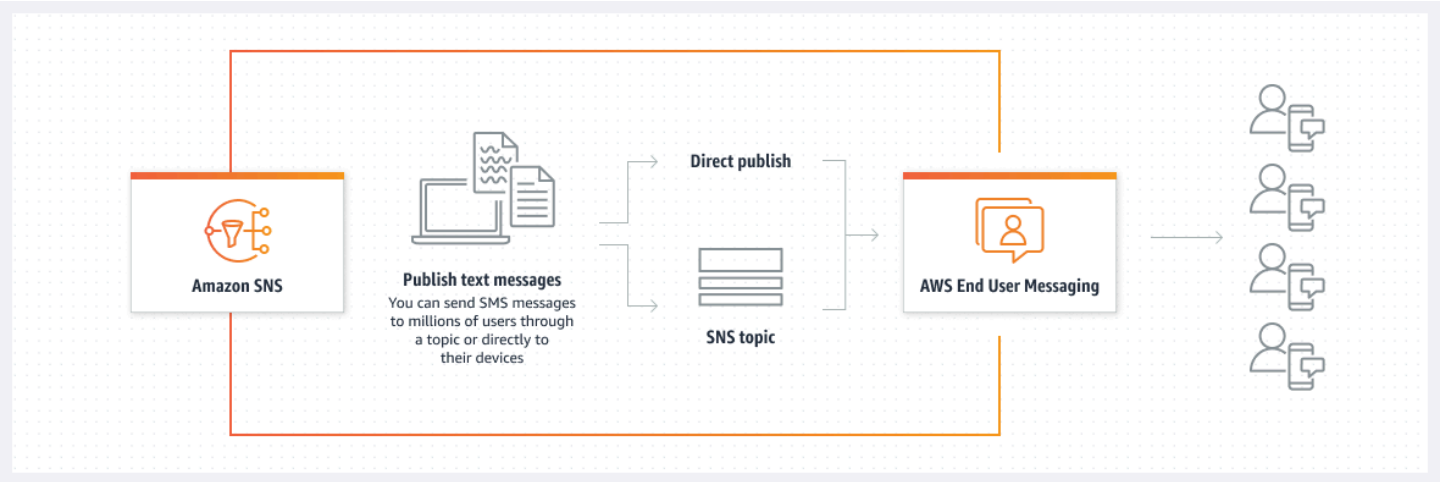
Mensajería de texto móvil con Amazon SNS

> **Important**
>
> Se ha actualizado la Guía para desarrolladores de SMS de Amazon SNS. Amazon SNS se ha integrado con AWS End User Messaging SMS para la entrega de mensajes SMS. Esta guía contiene la información más reciente sobre cómo crear, configurar y administrar los mensajes SMS de Amazon SNS.

La mensajería de texto móvil (SMS) de Amazon SNS se ha diseñado para facilitar la entrega de mensajes a diversas plataformas, como aplicaciones web, móviles y empresariales que admitan SMS. Los usuarios pueden enviar mensajes a uno o varios números de teléfono suscribiéndolos a un tema, lo que simplifica el proceso de distribución.

Los mensajes de Amazon SNS se entregan mediante AWS End User Messaging SMS, lo que garantiza una transmisión fiable de los mensajes. En Amazon SNS APIs, puede configurar varias propiedades, como los tipos de mensajes (promocionales o transaccionales), los límites de gastos mensuales, las listas de exclusión y la optimización de la entrega de mensajes.

AWS End User Messaging SMS gestiona la transmisión de mensajes al número de teléfono de destino a través de su red global de suministro de SMS. Administra el enrutamiento, el estado de la entrega y cualquier requisito exigido por las normativas regionales. Para acceder a características adicionales de SMS, como los permisos detallados, los grupos de teléfonos, los conjuntos de configuraciones, el simulador de SMS y las normas nacionales, consulte la Guía del usuario de AWS End User Messaging SMS.



Las siguientes características clave le ayudan a enviar mensajes SMS de Amazon SNS de forma escalable y extensible:

Personalice las preferencias de los mensajes

Personalice las entregas de SMS para Cuenta de AWS usted configurando las preferencias de SMS en función de su presupuesto y caso de uso. Por ejemplo, puede elegir si los mensajes deben optimizarse en función de la rentabilidad o la fiabilidad de la entrega.

Establezca cuotas de gasto

Adapte sus entregas de SMS especificando cuotas de gasto de las entregas de mensajes individuales y las cuotas de gasto mensuales de su Cuenta de AWS. Cuando lo exijan las leyes y reglamentos locales (como los de EE. UU. y Canadá), los destinatarios de los SMS pueden optar por no recibirlos, lo que significa que deciden dejar de Cuenta de AWS recibir sus mensajes SMS. Cuando un destinatario se da de baja en la recepción de mensajes, usted puede, dentro de unos límites, volver a dar de alta el número de teléfono para reanudar el envío de mensajes.

Envíe mensajes SMS a todo el mundo

Amazon SNS admite la mensajería SMS en varias regiones, lo que le permite enviar mensajes a más de 240 países y regiones.

¿Cómo entrega Amazon SNS mis mensajes SMS?

Cuando solicita a Amazon SNS que envíe SMS en su nombre, los mensajes se envían mediante AWS End User Messaging SMS. La integración entre Amazon SNS y Amazon AWS End User Messaging SMS ofrece las siguientes ventajas:

Políticas de IAM

Puede aprovechar las políticas de IAM y de recursos para controlar y distribuir el acceso a sus recursos de SMS entre otros AWS servicios y regiones.

Configuraciones de AWS End User Messaging SMS

Se utilizan todas las configuraciones relacionadas con el identificador de origen (creación, actualización de la configuración, aprovisionamiento de un nuevo origen IDs, cambio de plantillas de registro). AWS End User Messaging SMS

Facturación de AWS End User Messaging SMS

Sin embargo, toda la facturación por SMS está lista. AWS End User Messaging SMS Puede consolidar sus AWS gastos en sus cargas de trabajo de SMS y, al mismo tiempo, adquirir y gestionar sus recursos de SMS en una ubicación central.

Introducción a los SMS con Amazon SNS

> **Important**
>
> Se ha actualizado la Guía para desarrolladores de SMS de Amazon SNS. Amazon SNS se ha integrado con AWS End User Messaging SMS para la entrega de mensajes SMS. Esta guía contiene la información más reciente sobre cómo crear, configurar y administrar los mensajes SMS de Amazon SNS.

Este tema lo guía a través de la administración de su entorno limitado de SMS y la configuración de IAM y políticas basadas en recursos para conceder a Amazon SNS los permisos necesarios para acceder y utilizar el. AWS End User Messaging SMS APIs

Requisitos previos

Amazon SNS recomienda que actualice su política de IAM para incluir las siguientes acciones a fin de garantizar un control y una visibilidad exhaustivos de sus recursos de Amazon SNS:

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

Uso del entorno de pruebas de SMS de Amazon SNS

Las cuentas SMS de Amazon SNS recién creadas se colocan automáticamente en el entorno limitado de SMS para garantizar la seguridad tanto de AWS los clientes como de los destinatarios al mitigar el riesgo de fraude y abuso. Este entorno sirve como un espacio seguro para fines de prueba y desarrollo. Mientras trabaje dentro del entorno de pruebas de SMS, tendrá acceso a todas las características de Amazon SNS, pero se aplicarán determinadas limitaciones:

- Solo puede enviar mensajes SMS a números de teléfono de destino verificados.
- Puede tener hasta 10 números de teléfono de destino verificados.
- Solo puede eliminar números de teléfono de destino después de que hayan transcurrido 24 horas o más desde la verificación o el último intento de verificación.

Cuando su cuenta se mueve fuera del entorno de pruebas, estas restricciones se eliminan y puede enviar mensajes SMS a cualquier destinatario.

Primeros pasos

Las nuevas cuentas de SMS de Amazon SNS se colocan en un entorno de pruebas de SMS. Siga los siguientes pasos para crear y administrar números de teléfono en su entorno limitado, crear los números de origen y el remitente IDs, y registrar su empresa.

1. Añada un número de teléfono de destino al entorno de pruebas de SMS. Para obtener más información sobre cómo añadir, administrar y mover números de teléfono fuera del entorno de pruebas de SMS de Amazon SNS, consulte Cómo añadir y verificar números de teléfono en el entorno de pruebas de SMS de Amazon SNS.
2. Cree una identidad de origen que los destinatarios vean en los dispositivos cuando les envíe un mensaje SMS. Para obtener más información sobre las identidades de origen, incluidos los distintos tipos que puede utilizar, consulte la documentación de Identidades de origen de los mensajes SMS en Amazon SNS.
3. Registre la empresa. Algunos países requieren que registres la identidad de tu empresa para poder comprar números de teléfono o remitentes IDs y revisar los mensajes que envías a los destinatarios de su país. Para obtener más información sobre los países que requieren registro, consulte Regiones y países admitidos con AWS End User Messaging SMS en la Guía del usuario de AWS End User Messaging SMS .
4. Envíe sus mensajes a un tema o a un teléfono móvil. Para obtener más información, consulte Envío de mensajes SMS con Amazon SNS.

Cómo añadir y verificar números de teléfono en el entorno de pruebas de SMS de Amazon SNS

Para poder empezar a enviar mensajes SMS desde tu Cuenta de AWS entorno limitado de SMS, debes completar los siguientes pasos de configuración. Esto garantiza que tu cuenta esté lista para recibir mensajes SMS y que los números de teléfono de destino estén debidamente verificados.

1. Crea un identificador de origen. Al igual que ocurre con las cuentas fuera del entorno limitado de SMS, es necesario disponer de un identificador de origen para poder enviar mensajes SMS a destinatarios de algunos países o regiones.
2. Añade los números de teléfono de destino a los que quieres enviar los mensajes dentro del entorno limitado de SMS.
3. Compruebe los números de teléfono para asegurarse de que los números de teléfono de destino son válidos para su uso en los mensajes SMS.

Agrega y verifica los números de teléfono de destino

1. Inicie sesión en la consola de Amazon SNS.
2. En el menú de la consola, elija una región que admita la mensajería SMS.
3. En el panel de navegación, elija Text messaging (SMS) (Mensajería de texto (SMS)).
4. En la sección de números de teléfono de destino de Sandbox, selecciona Agregar número de teléfono.
5. En Detalles del destino, proporciona la siguiente información y, a continuación, selecciona Agregar número de teléfono:
   - Código de país y número de teléfono del destino.
   - El idioma en el que quieres que se envíe el mensaje de verificación.
6. Tras añadir el número de teléfono, Amazon SNS enviará una OTP al número de teléfono de destino proporcionado. Esta OTP es necesaria para la verificación.
7. Recibirás la OTP como un mensaje SMS estándar en el número de teléfono de destino que proporcionaste.
   - Si no recibes la OTP en 15 minutos, selecciona Reenviar el código de verificación en la consola de Amazon SNS.
   - Puedes volver a enviar la OTP hasta cinco veces en un período de 24 horas.
8. Cuando recibas la OTP, introdúcela en el cuadro del código de verificación y selecciona Verificar número de teléfono.
9. Comprueba el estado de la verificación.
   - Tras verificar correctamente el número de teléfono, el número de teléfono y su estado de verificación aparecerán en la sección de números de teléfono de destino de Sandbox.
   - Si el estado es Pendiente, la verificación no se realizó correctamente. Esto puede suceder si, por ejemplo, no has introducido el código de país correctamente.
   - Solo puedes eliminar los números de teléfono pendientes o verificados después de que hayan transcurrido 24 horas o más desde el último intento de verificación.
10. Si deseas usar el mismo número de teléfono de destino en otras regiones, repite los pasos anteriores para cada región en la que pretendas usarlo.

Solución de problemas de no recepción de un texto OTP

Solucione los problemas comunes que pueden impedir que un número de teléfono reciba textos OTP.

- Límite de gasto de SMS de Amazon SNS: si su Cuenta de AWS ha superado el límite de gasto para enviar mensajes SMS, es posible que no se entreguen más mensajes, incluidos los textos OTP, hasta que se aumente el límite o se resuelva el problema de facturación.
- Número de teléfono no dado de alta para recibir notificaciones por SMS: en algunos países o regiones, los destinatarios deben darse de alta para recibir mensajes SMS de códigos cortos, que suelen utilizarse en los textos OTP. Si el número de teléfono del destinatario no se ha dado de alta, no recibirá el texto OTP.
- Restricciones o filtros del operador: algunos operadores de telefonía móvil pueden tener restricciones o mecanismos de filtrado que impidan la entrega de determinados tipos de mensajes SMS, incluidos los textos OTP. Esto podría deberse a las políticas de seguridad o a las medidas antispam implementadas por el operador.
- Número de teléfono no válido o incorrecto: si el número de teléfono proporcionado por el destinatario es incorrecto o no es válido, no se entregará el texto OTP.
- Problemas de red: los problemas o interrupciones temporales de la red podrían impedir la entrega de mensajes SMS, incluidos los textos OTP, al teléfono del destinatario.
- Retraso en la entrega: en algunos casos, los mensajes SMS pueden sufrir retrasos en la entrega debido a la congestión de la red u otros factores. Es posible que el texto OTP se acabe entregando, pero podría retrasarse más allá del plazo previsto.
- Suspensión o cancelación de la cuenta: si hay problemas con su cuenta Cuenta de AWS, como la falta de pago o el incumplimiento de las AWS condiciones del servicio, las capacidades de mensajería de Amazon SNS, incluidos los mensajes de texto OTP, pueden suspenderse o cancelarse.

Eliminación de números de teléfono del entorno de pruebas de SMS de Amazon SNS

Puede eliminar números de teléfono de destino pendientes o verificados desde el entorno de pruebas de SMS.

> **Important**
>
> Solo puede eliminar un número de teléfono 24 horas después de verificarlo o 24 horas después de su último intento de verificación.

Para eliminar números de teléfono de destino del entorno de pruebas de SMS, siga estos pasos:

1. Inicie sesión en la consola de Amazon SNS.
2. En el menú de la consola, elija una región que admita la mensajería SMS en la que añadió un número de teléfono de destino.
3. En el panel de navegación, elija Mensajería de texto (SMS).
4. En la página Mensajería de texto móvil (SMS), vaya a la sección Números de teléfono de destino del entorno de pruebas.
5. Elija el número de teléfono específico que desea eliminar y, a continuación, seleccione Eliminar número de teléfono.
6. Para confirmar que desea eliminar el número de teléfono, ingrese **`delete me`** y, a continuación, elija Eliminar.

   Asegúrese de que hayan transcurrido 24 horas o más desde que verificó o intentó verificar el número de teléfono de destino antes de continuar con la eliminación.
7. Repita estos pasos en cada región en la que haya agregado el número de teléfono de destino y que ya no tenga previsto utilizar.

Cómo salir del entorno de pruebas de SMS de Amazon SNS

Para Cuenta de AWS salir del entorno limitado de los SMS, primero tendrá que añadir, verificar y probar los números de teléfono de destino. Después de hacer esto, crea un caso con AWS Support.

Para solicitar que tu AWS cuenta se retire del entorno limitado de SMS

1. Verificar números de teléfono
   a. Mientras Cuenta de AWS se encuentre en el entorno limitado de SMS, abra la consola de Amazon SNS.
   b. En el panel de navegación, bajo Móvil, elija Mensajería de texto (SMS).

c. En la sección de números de teléfono de destino del entorno de pruebas, añada y verifique uno o varios números de teléfono de destino. Esta verificación garantiza que pueda enviar y recibir mensajes correctamente.

2. Probar la publicación de SMS

   - Confirme que puede enviar y recibir mensajes en al menos un número de teléfono de destino verificado. Para obtener instrucciones más detalladas sobre cómo publicar mensajes SMS, consulte Publicación de mensajes SMS en un teléfono móvil mediante Amazon SNS.

3. Iniciar la edición del entorno de pruebas

   - En la página Mensajería de texto móvil (SMS) de la consola de Amazon SNS, en Información de la cuenta, elija Salir de entorno de pruebas de SMS. Esta acción le redirige al Centro de soporte de Amazon y crea automáticamente un caso de soporte con la opción Aumento de la cuota de servicio seleccionada.

4. Rellenar el formulario

   - En el formulario de soporte, bajo Aumento de la cuota de servicio, haga lo siguiente:

     i. Elija Mensajería de texto SNS como servicio.

     ii. Proporcione la URL del sitio web o el nombre de la aplicación desde la que desea enviar los mensajes SMS.

     iii. Elija el tipo de mensajes que quiere enviar: Contraseña de un solo uso, Promocional o Transaccional.

     iv. Elija la Región de AWS desde donde va a enviar los mensajes SMS.

     v. Especifique los países o las regiones donde piensa enviar mensajes SMS.

     vi. Describa cómo sus clientesse dan de alta para recibir mensajes.

     vii. Incluya cualquier plantilla de mensaje que desee utilizar.

5. Especificar la cuota y la región

   - En Requests (Solicitudes), realice el siguiente procedimiento:

     i. Elija el Región de AWSlugar al que quiere mover su. Cuenta de AWS

     ii. Elija Límites generales en Tipo de recurso.

     iii. En Cuota, elija Salir del entorno de pruebas de SMS.

iv. (Opcional) Para solicitar aumentos adicionales u otros ajustes, elija Agregar otra solicitud y especifique los detalles necesarios.

v. En Nuevo valor de cuota, especifique el límite en USD que va a solicitar.

6. Detalles adicionales

   a. En Descripción del caso, proporcione cualquier detalle adicional relevante para su solicitud.

   b. En Opciones de contacto, elija su idioma de contacto preferido.

7. Enviar la solicitud

   - Elija Enviar para enviar la solicitud a Soporte.

El Soporte equipo proporcionará una respuesta inicial a su solicitud en un plazo de 24 horas.

Para evitar que nuestros sistemas se utilicen para enviar contenido no solicitado o malicioso, consideramos cada solicitud con detenimiento. Si podemos, accederemos a su solicitud dentro de ese plazo de 24 horas. Sin embargo, si necesitamos que nos brinde más información, puede que la solicitud tarde más tiempo en concederse.

Si su caso de uso no se ajusta a nuestras políticas, es posible que no podamos atender su solicitud.

Identidades de origen de los mensajes SMS en Amazon SNS

> **Important**
>
> Se ha actualizado la Guía para desarrolladores de SMS de Amazon SNS. Amazon SNS se ha integrado con AWS End User Messaging SMS para la entrega de mensajes SMS. Esta guía contiene la información más reciente sobre cómo crear, configurar y administrar los mensajes SMS de Amazon SNS.

Las identidades de origen de los mensajes SMS son identificadores que se utilizan para representar al remitente de un mensaje SMS. Puede identificarse ante sus destinatarios mediante los siguientes tipos de identidades de origen:

Números de origen

Cadena numérica que identifica el número de teléfono del remitente de un mensaje SMS. Existen varios tipos de números de origen, como los códigos largos (números de teléfono estándar

que suelen tener diez o más dígitos), los códigos largos de diez dígitos (10DLC), los números gratuitos (TFN) y los códigos cortos (números de teléfono que contienen entre cuatro y siete dígitos).

La compatibilidad con los números de origen no está disponible en países donde las leyes locales exigen el uso del remitente IDs en lugar de los números de origen. Cuando envía un mensaje SMS con un número de origen, el dispositivo del destinatario muestra el número de origen como el número de teléfono del remitente. Puede especificar diferentes números de origen en función del caso de uso.

Para obtener más información, consulte Números de teléfono en la Guía del usuario de AWS End User Messaging SMS .

> **Tip**
>
> Para ver una lista de todos los números de origen existentes en su AWS cuenta, en el panel de navegación de la consola de Amazon SNS, seleccione Números de origen.

Remitente IDs

Nombre alfabético que identifica al remitente de un mensaje SMS. Cuando se envía un mensaje SMS mediante un ID de remitente y el destinatario se encuentra en una zona donde se admite la autenticación mediante ID de remitente, este último aparece en el dispositivo del destinatario en lugar del número de teléfono. El ID de remitente proporciona al destinatario del SMS más información sobre el remitente que un número de teléfono, un código largo o un código corto.

IDs Los remitentes son compatibles con varios países y regiones de todo el mundo. En algunos lugares, si una empresa envía mensajes SMS a clientes individuales, es imprescindible usar un ID de remitente previamente registrado ante un organismo de control o un grupo del sector. Para obtener una lista completa de los países y regiones que admiten o requieren el envío de mensajes SMS IDs, consulte los países y regiones compatibles con la mensajería SMS AWS End User Messaging SMS en la Guía del AWS End User Messaging SMS usuario.

El uso del remitente no conlleva ningún cargo adicional IDs. Sin embargo, la asistencia y los requisitos de autenticación del ID de remitente varían según el país. Varios mercados importantes (incluidos Canadá, China y Estados Unidos) no admiten el uso IDs del remitente. En algunas zonas, se exige que las empresas que envían mensajes SMS a clientes individuales utilicen un ID de remitente previamente registrado en un organismo de control o un grupo del sector.

Para obtener información adicional, consulta la sección Remitente IDs en la Guía AWS End User Messaging SMS del usuario.

Configuración de mensajes SMS en Amazon SNS

> **Important**
>
> Se ha actualizado la Guía para desarrolladores de SMS de Amazon SNS. Amazon SNS se ha integrado con AWS End User Messaging SMS para la entrega de mensajes SMS. Esta guía contiene la información más reciente sobre cómo crear, configurar y administrar los mensajes SMS de Amazon SNS.

Puede utilizar las configuraciones de SMS en Amazon SNS para configurar las preferencias de SMS de forma que se adapten a sus necesidades, como ajustar las cuotas de gasto y configurar el registro del estado de entrega. En este tema también se proporcionan detalles sobre cómo publicar mensajes SMS en los temas mediante la consola y el AWS SDK de Amazon SNS, gestionar las cuotas de forma eficiente y obtener estadísticas detalladas sobre la actividad de los SMS.

Envío de mensajes SMS con Amazon SNS

En esta sección se describe cómo enviar mensajes SMS con Amazon SNS, incluida la publicación en un tema, la suscripción de números de teléfono a los temas, la configuración de los atributos de los mensajes y la publicación directa en teléfonos móviles.

Publicación de mensajes SMS en un tema de Amazon SNS

Puede publicar un único mensaje SMS en muchos números de teléfono a la vez mediante la suscripción de dichos números de teléfono a un tema de Amazon SNS. Un tema de SNS es un canal de comunicación al que puede agregar suscriptores y publicar mensajes para todos ellos. El suscriptor recibe todos los mensajes publicados sobre el tema hasta que canceles la suscripción o hasta que el suscriptor opte por no recibir mensajes SMS de tu AWS cuenta.

Envío de un mensaje a un tema mediante la consola de AWS

Creación de un tema

Ejecute los pasos siguientes si todavía no tiene un tema al que quiera enviar mensajes SMS.

1. Inicie sesión en la consola de Amazon SNS.

2. En el menú de la consola, elija una región que admita la mensajería SMS.
3. En el panel de navegación, elija Temas.
4. En la página Temas, elija Crear tema.
5. En la página Crear tema, en Detalles, haga lo siguiente:

   a. En Tipo, seleccione Estándar.

   b. En Nombre, ingrese un nombre para el tema.

   c. (Opcional) En Nombre de visualización, ingrese un prefijo personalizado para los mensajes SMS. Cuando envía un mensaje al tema, Amazon SNS anexa delante el nombre de visualización seguido de un corchete angular de cierre (>) y un espacio. Los nombres de visualización no distinguen entre mayúsculas y minúsculas, y Amazon SNS convierte los nombres de visualización en caracteres en mayúsculas. Por ejemplo, si el nombre de visualización de un tema es `MyTopic` y el mensaje es `Hello World!`, el mensaje aparecerá de la siguiente manera:

   ```
   MYTOPIC> Hello World!
   ```

6. Seleccione Crear tema. El nombre del tema y el nombre de recurso de Amazon (ARN) aparecen en la página Temas.

Para crear una suscripción de SMS, siga estos pasos:

Puede utilizar las suscripciones para enviar un mensaje SMS a varios destinatarios al publicar el mensaje una sola vez en su tema.

> **Note**
>
> Cuando empiece a utilizar Amazon SNS para enviar mensajes SMS, su AWS cuenta estará en el entorno limitado de SMS. El entorno de pruebas de SMS proporciona un entorno seguro para que pruebe las características de Amazon SNS sin arriesgar su reputación como remitente de SMS. Mientras su cuenta se encuentre en el entorno de pruebas de SMS, puede utilizar todas las características de Amazon SNS, pero solo puede enviar mensajes SMS a números de teléfono de destino verificados. Para obtener más información, consulte Uso del entorno de pruebas de SMS de Amazon SNS.

1. Inicie sesión en la consola de Amazon SNS.

2. En el panel de navegación, seleccione Suscripciones.
3. En la página Subscriptions (Suscripciones), elija Create subscription (Crear suscripción).
4. En la página Crear suscripción, en Detalles, haga lo siguiente:

   a. En ARN de tema, ingrese o elija el nombre de recurso de Amazon (ARN) del tema al que desea enviar mensajes SMS.

   b. En Protocolo, elija SMS.

   c. En Punto de enlace, ingrese el número de teléfono al que desea suscribirse al tema.
5. Seleccione Crear subscripción. La información de la suscripción aparece en la página Suscripciones.

   Para agregar más números de teléfono, repita estos pasos. También puede agregar otros tipos de suscripciones, como el correo electrónico.

Cómo enviar un mensaje

Cuando publica un mensaje en un tema, Amazon SNS intenta entregar dicho mensaje a todos los números de teléfono que están suscritos al tema.

1. En Consola de Amazon SNS, en la página Temas, elija el nombre del tema al que desea enviar mensajes SMS.
2. En la página de detalles del tema, seleccione Publish message (Publicar mensaje).
3. En la página Publicar mensaje en el tema, en Detalles del mensaje, haga lo siguiente:

   a. En Asunto, deje el campo en blanco a menos que el tema contenga suscripciones de correo electrónico y quiera publicar tanto en las suscripciones de correo electrónico como en las de SMS. Amazon SNS utiliza el asunto que ingresa como línea de asunto del correo electrónico.

   b. (Opcional) En Período de vida (TLL), ingrese un número de segundos que Amazon SNS tiene para enviar su mensaje SMS a los suscriptores de terminales de aplicaciones móviles.
4. En Cuerpo del mensaje, haga lo siguiente:

   a. En Estructura del mensaje, elija Carga idéntica para todos los protocolos de entrega para enviar el mismo mensaje a todos los tipos de protocolo suscritos al tema. O bien, elija Carga personalizada para cada protocolo de entrega para personalizar el mensaje para suscriptores de diferentes tipos de protocolo. Por ejemplo, puede escribir un mensaje

predeterminado para los suscriptores de números de teléfono y un mensaje personalizado para los suscriptores de correo electrónico.

b. En Cuerpo del mensaje para enviar al punto de enlace, ingrese su mensaje o sus mensajes personalizados por protocolo de entrega.

   Si su tema tiene un nombre de visualización, Amazon SNS lo agrega al mensaje, lo que aumenta la longitud del mensaje. La longitud del nombre de visualización es el número de caracteres del nombre más dos caracteres para el corchete angular de cierre (>) y el espacio que Amazon SNS agrega.

   Para obtener información acerca de las cuotas de tamaño de los mensajes SMS, consulte Publicación de mensajes SMS en un teléfono móvil mediante Amazon SNS.

5. (Opcional) En el caso de los atributos de los mensajes, añada metadatos del mensaje, como marcas de tiempo, firmas y. IDs
6. Elija Publish message (Publicar mensaje). Amazon SNS envía el mensaje SMS y muestra un mensaje de confirmación.

Enviar un mensaje a un tema mediante el AWS SDKs

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema de Amazon SNS
- Suscribir números de teléfono al tema
- Publique mensajes SMS en el tema para que todos los números de teléfono suscritos reciban el mensaje a la vez.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree un tema y devuelva su 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 "";
    }
}
```

Suscriba un punto de enlace a un tema.

```
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);
        }
    }
}
```

Establezca atributos en el mensaje, como el ID del remitente, el precio máximo y su tipo. Los atributos de mensaje son opcionales.

```
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 un mensaje en un tema. El mensaje se envía a cada suscriptor.

```
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);
        }
    }
}
```

Publicación de mensajes SMS en un teléfono móvil mediante Amazon SNS

Puede utilizar Amazon SNS para enviar mensajes SMS de forma directa a un teléfono móvil sin suscribir el número de teléfono a un tema de Amazon SNS.

> **Note**
>
> Suscribir números de teléfono a un tema es útil si quiere enviar un mensaje a varios números de teléfono a la vez. Para obtener instrucciones sobre cómo publicar un mensaje SMS en un tema, consulte Publicación de mensajes SMS en un tema de Amazon SNS.

Cuando se envía un mensaje, se puede controlar si el mensaje se optimiza en función del costo o de la fiabilidad de la entrega. También puede especificar un ID de remitente o número de origen. Si envía el mensaje mediante programación mediante la API de Amazon SNS o AWS SDKs la, puede especificar un precio máximo para la entrega del mensaje.

Cada mensaje SMS puede contener hasta 140 bytes y la cuota de caracteres depende del esquema de codificación. Por ejemplo, un mensaje SMS puede contener:

- 160 caracteres GSM
- 140 caracteres ASCII
- 70 caracteres UCS-2

Si publica un mensaje que exceda la cuota de tamaño, Amazon SNS lo envía como varios mensajes, cada uno de los cuales respetará la cuota. Los mensajes no se dividen en mitad de una palabra, sino en el espacio entre palabras. La cuota de tamaño total de una acción de publicación SMS es 1600 bytes.

Al enviar un mensaje SMS, especifique el número de teléfono mediante el formato E.164, una estructura de numeración de teléfono estándar utilizada para las telecomunicaciones internacionales. Los números de teléfono que siguen este formato pueden tener un máximo de 15 dígitos junto con el prefijo de un signo más (+) y el código del país. Por ejemplo, un número de teléfono estadounidense en formato E.164 aparece como +1 0100. XXX555

Envío de un mensaje (consola)

1. Inicie sesión en la consola de Amazon SNS.

2. En el menú de la consola, elija una región que admita la mensajería SMS.
3. En el panel de navegación, elija Text messaging (SMS) (Mensajería de texto (SMS)).
4. En la página Mensajería de texto móvil (SMS), elija Publicar mensaje de texto.
5. En la página Publicar mensaje SMS, para el tipo de mensaje., elija una de las siguientes opciones:

   - Promotional: mensajes que no son de importancia, como mensajes de marketing.
   - Transactional: mensajes de importancia que admiten transacciones del cliente, como claves de acceso de un solo uso para la autenticación multifactor.

   > **Note**
   >
   > Esta opción de nivel de mensaje anula el tipo de mensaje predeterminado de nivel de cuenta. Puede establecer un tipo de mensaje predeterminado a nivel de cuenta desde la sección Preferencias de mensajes de texto de la página Mensajería de texto móvil (SMS).

   Para obtener información sobre los mensajes promocionales y transaccionales, consulte Precios de SMS en todo el mundo.
6. En Número de teléfono de destino, ingrese el número de teléfono al que desea enviar el mensaje.
7. En Mensaje, ingrese el mensaje que va a enviar.
8. (Opcional) En Identidades de origen, especifique cómo identificarse ante sus destinatarios:

   - Para especificar un ID de remitente, ingrese un ID personalizado que contenga 3 a 11 caracteres alfanuméricos, incluida al menos una letra y sin espacios. El ID de remitente se muestra como el remitente del mensaje en el dispositivo receptor. Por ejemplo, puede utilizar la marca de su negocio para facilitar el reconocimiento del origen del mensaje.

     El soporte para el remitente IDs varía según el país o la región. Por ejemplo, los mensajes que se entregan a números de teléfono de los EE. UU. no mostrarán el ID de remitente. Para conocer los países y regiones que admiten el envío de mensajes SMS IDs, consulta la sección Países y regiones compatibles con la mensajería SMS AWS End User Messaging SMS en la Guía del AWS End User Messaging SMS usuario.

Si no especifica un ID de remitente, se mostrará una de las siguientes características como identidad de origen:

- En los países que admiten códigos largos, se muestra el código largo.
- En los países en los que solo IDs se admiten remitentes, se muestra NOTICE.

Este ID de remitente de nivel de mensaje anula el ID de remitente predeterminado, que se establece en la página Text messaging preferences (Preferencias de la mensajería de texto).

- Para especificar un número de origen, ingrese una cadena de 5 a 14 números que se mostrará como número de teléfono del remitente en el dispositivo del receptor. Esta cadena debe coincidir con un número de origen que esté configurado en su página Cuenta de AWS para el país de destino. El número de origen puede ser un número de 10 DLC, un número gratuito, un código person-to-person largo o un código corto. Para obtener más información, consulte Identidades de origen de los mensajes SMS en Amazon SNS.

  Si no especifica un número de origen, Amazon SNS selecciona un número de origen que se utilizará para el mensaje de texto SMS, en función de la configuración de su Cuenta de AWS .

9. Si envía mensajes SMS a destinatarios en India, expanda Atributos específicos del país y especifique los atributos siguientes:

   - ID de la identidad: ID entidad o ID de entidad principal (PE) para enviar mensajes SMS a destinatarios en la India. Este ID es una cadena única de 1 a 50 caracteres que la Autoridad Reguladora de las Telecomunicaciones de la India (TRAI) proporciona para identificar la entidad que ha registrado en la TRAI.
   - ID de plantilla: ID de plantilla para enviar mensajes SMS a destinatarios en la India. Este ID es una cadena única de 1 a 50 caracteres que proporciona la TRAI para identificar la plantilla que registró en la TRAI. El ID de plantilla debe estar asociado al ID de remitente que especificó para el mensaje.

   Para obtener más información sobre cómo enviar mensajes SMS a destinatarios en la India, consulte India sender ID registration process en la Guía del usuario de AWS End User Messaging SMS .

10. Elija Publish message (Publicar mensaje).

> **Tip**
>
> Para enviar mensajes SMS desde un número de origen, también puede elegir números de origen en el panel de navegación de la consola de Amazon SNS. Elija un número de origen que incluya SMS en la columna Capacidades y, a continuación, elija Publicar mensajes de texto.

Enviar un mensaje ()AWS SDKs

Para enviar un mensaje SMS mediante una de las AWS SDKs, utilice la operación de API de ese SDK que corresponda a la `Publish` solicitud de la API de Amazon SNS. Con esta solicitud, puede enviar un mensaje SMS directamente a un número de teléfono. También puede utilizar el parámetro `MessageAttributes` para establecer valores para los siguientes nombres de atributo:

`AWS.SNS.SMS.SenderID`

Un ID personalizado que contiene entre 3 y 11 caracteres alfanuméricos o guiones (-), entre ellos al menos una letra, y ningún espacio. El ID de remitente aparece como el remitente del mensaje en el dispositivo receptor. Por ejemplo, puede utilizar la marca de su negocio para facilitar el reconocimiento del origen del mensaje.

El soporte para el remitente IDs varía según el país o la región. Por ejemplo, los mensajes que se entregan a números de teléfono de los EE. UU. no muestra el ID de remitente. Para ver una lista de los países o regiones que admiten remitentes IDs, consulta los países y regiones compatibles con la mensajería SMS AWS End User Messaging SMS en la Guía del AWS End User Messaging SMS usuario.

Si no especifica un ID de remitente, aparece un código largo como ID de remitente en los países o regiones admitidos. Para los países o regiones que requieren un ID de remitente alfabético, aparece AVISO como ID de remitente.

Este atributo de nivel de mensaje anula el atributo de nivel de cuenta `DefaultSenderID`, que se puede establecer mediante la solicitud `SetSMSAttributes`.

`AWS.MM.SMS.OriginationNumber`

Una cadena personalizada de 5 a 14 números, que puede incluir un signo más inicial opcional (+). Esta cadena de números aparece como el número de teléfono del remitente en el dispositivo receptor. La cadena debe coincidir con un número de origen que esté configurado en tu AWS

cuenta para el país de destino. El número de origen puede ser un número de 10 DLC, un número gratuito, un código largo person-to-person (P2P) o un código corto. Para obtener más información, consulte Phone numbers en la Guía del usuario de AWS End User Messaging SMS .

Si no especificas un número de origen, Amazon SNS elegirá un número de origen en AWS función de la configuración de tu cuenta.

`AWS.SNS.SMS.MaxPrice`

El precio máximo en USD que estás dispuesto a gastar para enviar el mensaje SMS. Si Amazon SNS determina que el envío del mensaje supondría un costo superior a su precio máximo, no lo envía.

Este atributo no tiene efecto si los costes de los month-to-date SMS ya han superado la cuota establecida para el atributo. `MonthlySpendLimit` Puede establecer el atributo `MonthlySpendLimit` con la solicitud `SetSMSAttributes`.

Si envía el mensaje a un tema de Amazon SNS, el precio máximo se aplica a cada entrega de mensaje a cada número de teléfono que esté suscrito al tema.

`AWS.SNS.SMS.SMSType`

El tipo de mensaje que envía:

- **`Promotional`** (predeterminado): mensajes que no son de importancia, como mensajes de marketing.
- **`Transactional`**: mensajes de importancia que admiten transacciones del cliente, como claves de acceso de un solo uso para la autenticación multifactor.

Este atributo de nivel de mensaje anula el atributo de nivel de cuenta `DefaultSMSType`, que se puede establecer mediante la solicitud `SetSMSAttributes`.

`AWS.MM.SMS.EntityId`

Este atributo solo es necesario para enviar mensajes SMS a destinatarios en la India.

Se trata del ID entidad o ID de entidad principal (PE) para enviar mensajes SMS a destinatarios en la India. Este ID es una cadena única de 1 a 50 caracteres que la Autoridad Reguladora de las Telecomunicaciones de la India (TRAI) proporciona para identificar la entidad que ha registrado en la TRAI.

`AWS.MM.SMS.TemplateId`

Este atributo solo es necesario para enviar mensajes SMS a destinatarios en la India.

Se trata de la plantilla para enviar mensajes SMS a destinatarios en la India. Este ID es una cadena única de 1 a 50 caracteres que proporciona la TRAI para identificar la plantilla que registró en la TRAI. El ID de plantilla debe estar asociado al ID de remitente que especificó para el mensaje.

Envío de un mensaje

En los siguientes ejemplos de código, se muestra cómo publicar mensajes SMS mediante Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
```

```
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

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

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

    return outcome.IsSuccess();
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para C++ .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
```

```
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }
```

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener información sobre la API, consulte Publicar en la Referencia de la API de AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

```
    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Python (Boto3).

Configuración de las preferencias de mensajería SMS en Amazon SNS

Utilice Amazon SNS para especificar las preferencias de mensajería SMS. Por ejemplo, puede especificar si desea optimizar las entregas por costo o fiabilidad, el límite de gasto mensual, cómo se registran las entregas y si desea suscribirse a informes de uso de SMS diarios.

Estas preferencias se aplican en todos los mensajes SMS que envía desde su cuenta, pero puede anular algunas de ellas cuando envía un mensaje individual. Para obtener más información, consulte Publicación de mensajes SMS en un teléfono móvil mediante Amazon SNS.

Configuración de las preferencias de mensajería SMS mediante la AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. Elija una región que admita la mensajería SMS.
3. En el panel de navegación, elija Móvil y después Mensajería de texto (SMS).
4. En la página Mensajería de texto a través del móvil (SMS), en la sección Preferencias de mensajería de texto, elija Editar.
5. En la página Editar preferencias de mensajería de texto, en la sección Detalles, haga lo siguiente:

   a. En Default message type (Tipo predeterminado de mensaje), seleccione uno de los siguientes:

      - Promocional: mensajes no importantes (por ejemplo, marketing). Amazon SNS optimiza la entrega de mensajes para conseguir el costo más bajo.
      - Transaccional (predeterminado): mensajes de importancia que admiten transacciones del cliente, como claves de acceso de un solo uso para la autenticación multifactor. Amazon SNS optimiza el envío de mensajes para conseguir la máxima reputación.

      Para obtener información sobre los precios de los mensajes promocionales y transaccionales, consulte la página relacionada con los precios globales de SMS.

   b. (Opcional) En Account spend limit (Límite de gasto de la cuenta), escriba el importe (en USD) que desea gastar en mensajes SMS cada mes natural.

      > ⚠ Important
      >
      > - De forma predeterminada, la cuota de gasto se establece en 1,00 USD. Si desea aumentar la cuota de servicio, envíe una solicitud.
      > - Si el importe establecido en la consola supera la cuota del servicio, Amazon SNS deja de publicar mensajes SMS.
      > - Como Amazon SNS es un sistema distribuido, deja de enviar mensajes SMS en cuestión de minutos en cuanto se ha excedido la cuota de gasto. Durante este intervalo, si sigue enviando mensajes SMS, podría incurrir en costos que superen la cuota.

6. (Opcional) En Default sender ID (ID de remitente predeterminado), escriba un ID personalizado, como la marca de su negocio, que se muestra como el remitente en el dispositivo receptor.

> **Note**
>
> El soporte para el remitente IDs varía según el país.

7. (Opcional) Ingrese el nombre del nombre del bucket de Amazon S3 para informes de uso.

> **Note**
>
> La política de buckets de Amazon S3 debe conceder acceso de escritura a Amazon SNS.

8. Elija Guardar cambios.

Configuración de preferencias ()AWS SDKs

Para configurar sus preferencias de SMS mediante una de las AWS SDKs, utilice la acción de ese SDK que corresponda a la `SetSMSAttributes` solicitud en la API de Amazon SNS. Con esta solicitud, debe asignar valores a los diferentes atributos de SMS, como la cuota de gasto mensual o el tipo de SMS predeterminado (transaccional o promocional). Para ver todos los atributos de SMS, consulte Establecer SMSAttributes en la referencia de la API de Amazon Simple Notification Service.

En los siguientes ejemplos de código, se muestra cómo utilizar `SetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cómo utilizar Amazon SNS para establecer el atributo predeterminadoSMSType .

```
//! 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 obtener más información sobre la API, consulte Establecer SMSAttributes en la referencia de AWS SDK para C++ la API.

CLI

AWS CLI

Establecimiento de los atributos de los mensajes SMS

En el siguiente ejemplo de `set-sms-attributes`, se establece el ID de remitente predeterminado para los mensajes SMS a `MyName`.

```
aws sns set-sms-attributes \
```

```
    --attributes DefaultSenderID=MyName
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte Set SMSAttributes in AWS CLI Command Reference.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta Establecer SMSAttributes en la referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.

- Para obtener más información sobre la API, consulta Establecer SMSAttributes en la referencia AWS SDK para JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta Establecer SMSAttributes en la referencia AWS SDK para PHP de la API.

Configuración de las preferencias de mensajería SMS para la entrega en un país específico

Puede administrar y controlar su tráfico de SMS enviando mensajes solo a países de destino específicos. Esto garantiza que sus mensajes se envíen solo a los países aprobados, lo que evita cargos por SMS no deseados. En las siguientes instrucciones se utiliza la configuración Proteger de Amazon Pinpoint para especificar los países que desea permitir o bloquear.

1. Abra la AWS SMS consola en https://console.aws.amazon.com/sms-voice/.
2. En el panel de navegación, en Información general, en la sección de Inicio rápido, elija Crear una configuración de protección.
3. En los detalles de configuración de Protect, introduzca un nombre comercial para su configuración de protección (por ejemplo, Allow-Only-AU).
4. En Reglas de país de SMS, activa la casilla de verificación Región/País para bloquear el envío de mensajes a todos los países admitidos.
5. Desactive las casillas de los países a los que desea enviar mensajes. Por ejemplo, para permitir que los mensajes solo lleguen a Australia, desactive la casilla de Australia.
6. En la sección Asociaciones de configuraciones de protección, en Tipo de asociación, seleccione Cuenta predeterminada. Esto garantizará que la configuración de AWS End User Messaging SMS Protect afecte a todos los mensajes enviados a través de Amazon SNS, Amazon Cognito y la llamada a la API Amazon Pinpoint. `SendMessages`
7. Elija Crear configuración de protección para guardar la configuración.

   Se muestra el siguiente mensaje de confirmación.

   ```
   Success Protect configuration protect-abc0123456789 has been created.
   ```

8. Inicie sesión en la consola de Amazon SNS.
9. Publicación de un mensaje en uno de los países bloqueados, como India.

   El mensaje no se entregará. Puede verificarlo en los registros de errores de entrega mediante. CloudWatch Busque un grupo de registros sns/region/AccountID/DirectPublishToPhoneNumber/ Failurepara obtener una respuesta similar a la del siguiente ejemplo:

   ```
   {
   "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"
}
```


Administración de números de teléfono y suscripciones en Amazon SMS

Con Amazon SNS, se ofrecen varias opciones para administrar quién recibe mensajes SMS desde su cuenta. Con una frecuencia limitada, puede reactivar números de teléfono en los que se ha desactivado la recepción de mensajes SMS desde su cuenta. Para dejar de enviar mensajes a suscripciones a SMS, puede eliminar las suscripciones o los temas que se publican en ellos.

Desactivación de la recepción de mensajes SMS

Cuando la legislación y la normativa locales vigentes así lo exijan (como, por ejemplo, en los EE. UU. y Canadá), los destinatarios de SMS podrán utilizar sus dispositivos para cancelar su suscripción respondiendo al mensaje con cualquiera de las palabras siguientes:

- ARRET (francés)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

Para cancelar la suscripción, el destinatario debe responder al mismo número de origen que Amazon SNS utilizó para entregar el mensaje. Tras darse de baja, el destinatario dejará de recibir

los mensajes SMS enviados por usted, Cuenta de AWS a menos que usted indique su número de teléfono.

Si el número de teléfono está suscrito a un tema de Amazon SNS, la cancelación la suscripción no la eliminará, pero los mensajes SMS no se entregarán a dicha suscripción a menos que dé de alta el número de teléfono.

Administración de números de teléfono y suscripciones mediante la consola de Amazon SMS

Puede utilizar la consola de Amazon SNS para controlar qué números de teléfono recibirán mensajes SMS desde su cuenta.

Reactivación de un número de teléfono que se ha dado de baja en la consola de Amazon SNS

Puede ver qué números de teléfono se han dado de baja de la recepción de mensajes SMS desde su cuenta y puede reactivarlos para reanudar el envío de mensajes.

Puede dar de alta un número de teléfono solo una vez cada 30 días.

1. Inicie sesión en la consola de Amazon SNS.
2. En el menú de la consola, establezca el selector de regiones en una región que admita la mensajería SMS.
3. En el panel de navegación, elija Text messaging (SMS) (Mensajería de texto (SMS)).
4. En la página Mensajería de texto a través de móvil (SMS), en la sección Números de teléfono desactivados, se muestran los números de teléfono que se han dado de baja.
5. Seleccione la casilla del número de teléfono que desea volver a dar de alta y, a continuación, elija Activar. El número de teléfono ya no estará dado de baja y recibirá los mensajes SMS que le envíe.

Eliminación de una suscripción a SMS mediante la consola de Amazon SNS

Elimine una suscripción a SMS para detener el envío de mensajes SMS a ese número de teléfono cuando publique en sus temas.

1. En el panel de navegación, seleccione Subscriptions (Suscripciones).
2. Seleccione las casillas de verificación correspondientes a las suscripciones que desee eliminar. A continuación, elija Actions (Acciones) y después Delete Subscriptions (Eliminar suscripciones).
3. En la ventana Delete (Eliminar), elija Delete (Eliminar). Amazon SNS elimina la suscripción y muestra un mensaje de confirmación.

Eliminación de un tema mediante la consola de Amazon SNS.

Elimine un tema cuando ya no quiera publicar mensajes en sus puntos de enlace suscritos.

1. En el panel de navegación, elija Temas.
2. Seleccione las casillas de verificación correspondientes a los temas que desee eliminar. A continuación, elija Actions (Acciones) y después Delete Topics (Eliminar temas).
3. En la ventana Delete (Eliminar), elija Delete (Eliminar). Amazon SNS elimina el tema y muestra un mensaje de confirmación.

Administración de números de teléfono y suscripciones mediante el SDK de AWS

Puede usarlo AWS SDKs para realizar solicitudes programáticas a Amazon SNS y administrar los números de teléfono que pueden recibir mensajes SMS de su cuenta.

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

Visualización de todos los números de teléfono que se han excluido mediante el SDK AWS

Para ver todos los números de teléfono que se han dado de baja, envíe una solicitud `ListPhoneNumbersOptedOut` a través de la API de Amazon SNS.

Los siguientes ejemplos de código muestran cómo utilizar `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Mostrar exclusiones de mensajes SMS

El siguiente ejemplo de `list-phone-numbers-opted-out` muestra los números de teléfono excluidos de la recepción de mensajes SMS.

```
aws sns list-phone-numbers-opted-out
```

Salida:

```
{
```

```
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Para obtener más información sobre la API, consulte ListPhoneNumbersOptedOutla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
```

```
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

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

- Para obtener más información sobre la API, consulta ListPhoneNumbersOptedOutla Referencia AWS SDK for Java 2.x de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta ListPhoneNumbersOptedOutla Referencia AWS SDK para PHP de la API.

Comprobar si un número de teléfono está excluido mediante el SDK AWS

Para verificar si un número de teléfono se ha dado de baja, envíe una solicitud `CheckIfPhoneNumberIsOptedOut` con la API de Amazon SNS.

Los siguientes ejemplos de código muestran cómo utilizar `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
```

```
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Comprobar que se ha cancelado la recepción de mensajes SMS en un número de teléfono

En el siguiente `check-if-phone-number-is-opted-out` ejemplo, se comprueba si el número de teléfono especificado está excluido de la recepción de mensajes SMS de la AWS cuenta corriente.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Salida:

```
{
    "isOptedOut": false
}
```

- Para obtener más información sobre la API, consulte CheckIfPhoneNumberIsOptedOutla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

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

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

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK para JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK para PHP de la API.

Reactivación de un número de teléfono que se ha dado de baja con la API de Amazon SNS

Para dar de alta un número de teléfono, envíe una solicitud `OptInPhoneNumber` con la API de Amazon SNS.

Puede dar de alta un número de teléfono solo una vez cada 30 días.

Eliminar AWS una suscripción por SMS mediante el SDK

Para eliminar una suscripción a SMS desde un tema de Amazon SNS, obtenga el ARN de la suscripción al presentar una solicitud `ListSubscriptions` con la API de Amazon SNS. A continuación, pase el ARN a una solicitud `Unsubscribe`.

Los siguientes ejemplos de código muestran cómo utilizar `Unsubscribe`.

.NET

SDK para .NET

> **Note**
>
> Hay más información sobre. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Darse de baja de un tema mediante un ARN de suscripción

```
/// <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 obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para C++ .

CLI

AWS CLI

Cancelación de la suscripción a un tema

En el siguiente ejemplo de `unsubscribe`, se elimina la suscripción especificada de un tema.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Este comando no genera ninguna salida.

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de comandos de la AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

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

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

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

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

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

```
    }
}
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener información sobre la API, consulte Cancelar suscripción en la Referencia de la API de AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
```

```
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
subscription.arn)
            raise
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   TRY.
       lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
       MESSAGE 'Subscription deleted.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Subscription does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snsinvalidparameterex.
       MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
   ENDTRY.
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta la sección Cancelar suscripción en el AWS SDK para ver la referencia sobre la API de Swift.

Eliminación de un tema utilizando el SDK de AWS

Para eliminar un tema y todas sus suscripciones, obtenga el ARN del tema al presentar una solicitud `ListTopics` con la API de Amazon SNS y, a continuación, pase el ARN a la solicitud `DeleteTopic`.

Los siguientes ejemplos de código muestran cómo utilizar `DeleteTopic`.

.NET

SDK para .NET

> **Note**
>
> Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Elimine un tema por su ARN de tema.

```
/// <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 obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Eliminación de un tema de SNS

El siguiente ejemplo de `delete-topic` elimina el tema de SNS especificado.

```
aws sns delete-topic \
```

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

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia de AWS CLI comandos.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>
```

```
                Where:
                   topicArn - The ARN of the topic to delete.
                """;

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

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

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulta DeleteTopicla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Para obtener más información sobre la API, consulta DeleteTopicla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte DeleteTopicla referencia sobre la API ABAP del AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import AWSSNS

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

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

- Para obtener más información sobre la API, consulta DeleteTopicla referencia sobre la API de AWS SDK for Swift.

Supervisión de la actividad de SMS en Amazon SNS

Mediante la supervisión de la actividad de SMS, puede realizar un seguimiento de los números de teléfono de destino, las entregas realizadas correctamente o que han producido un error, los motivos del error, los costos y otra información. Amazon SNS le ayuda resumiendo las estadísticas en la consola, enviando información a Amazon CloudWatch y enviando informes de uso diario de SMS a un bucket de Amazon S3 que especifique.

Consulta de las estadísticas de entrega de SMS en Amazon SNS

Puede utilizar la consola de Amazon SNS para ver estadísticas de sus entregas de SMS recientes.

1. Inicie sesión en la consola de Amazon SNS.
2. En el menú de la consola, establezca el selector de regiones en una región que admita la mensajería SMS.
3. En el panel de navegación, elija Text messaging (SMS) (Mensajería de texto (SMS)).
4. En la página Text messaging (SMS) (Mensajería de texto (SMS)), en la sección Account stats (Estadísticas de la cuenta), vea los gráficos de las entregas de mensajes SMS transaccionales y promocionales. Cada gráfico muestra los siguientes datos de los últimos 15 días:

   - Tasa de entrega (porcentaje de entregas correctas)
   - Envíos (número de intentos de entrega)
   - Errores (número de entregas erróneas)

En esta página, también puede seleccionar el botón Uso para ir al bucket de Amazon S3 en el que almacena los informes de uso diarios. Para obtener más información, consulte Suscripción a informes de uso diario de SMS en Amazon SNS.

Supervisión de la entrega de SMS de Amazon SNS con CloudWatch métricas y registros de Amazon

Puedes usar Amazon CloudWatch y Amazon CloudWatch Logs para supervisar la entrega de tus mensajes SMS.

Ver las CloudWatch métricas de Amazon

Amazon SNS recopila automáticamente las estadísticas sobre la entrega de sus mensajes SMS y las envía a Amazon. CloudWatch Puede utilizarlas CloudWatch para supervisar estas métricas y crear alarmas que le avisen cuando una métrica supere un umbral. Por ejemplo, puedes monitorear CloudWatch las métricas para conocer tu tarifa de envío de SMS y tus cargos por month-to-date SMS.

Para obtener información sobre la supervisión de CloudWatch las métricas, la configuración de CloudWatch alarmas y los tipos de métricas disponibles, consulteSupervisión de temas de Amazon SNS mediante CloudWatch.

Visualización CloudWatch de registros

Puede recopilar información sobre las entregas de mensajes SMS realizadas correctamente y sin éxito si permite que Amazon SNS escriba en Amazon CloudWatch Logs. Por cada mensaje SMS que envíe, Amazon SNS escribirá un registro en el que se incluya el precio del mensaje, su estado (correcto o error), el motivo del error (si el mensaje generó un error), el tiempo de permanencia del mensaje y otra información.

Para habilitar y ver CloudWatch los registros de sus mensajes SMS

1. Inicie sesión en la consola de Amazon SNS.
2. En el menú de la consola, establezca el selector de regiones en una región que admita la mensajería SMS.
3. En el panel de navegación, elija Text messaging (SMS) (Mensajería de texto (SMS)).
4. En la página Mensajería de texto a través del móvil (SMS), en la sección Preferencias de mensajería de texto, elija Editar.
5. En la siguiente página, expanda la sección Registro de estado de entrega.
6. En Frecuencia de muestreo correcta, especifique el porcentaje de envíos de SMS correctos para los que Amazon SNS escribirá registros en CloudWatch los registros. Por ejemplo:

   - Por ejemplo, para escribir registros únicamente para las entregas erróneas, establezca este valor en 0.
   - Para escribir logs para el 10% de las entregas de correctas, establézcalo en 10.

   Si no especifica ningún porcentaje, Amazon SNS escribirá registros para todas las entregas correctas.

7. Para proporcionar los permisos obligatorios, realice una de las siguientes acciones:

   - Para crear un nuevo rol de servicio, elija Crear nueva función de servicio y, a continuación, Crear nuevos roles. En la página siguiente, elija Permitir para dar acceso de escritura a Amazon SNS a los recursos de su cuenta.
   - Para utilizar una función de servicio existente, haga clic en Usar función de servicio existente y, a continuación, pegue el nombre de ARN en el cuadro Rol de IAM para entregas exitosas y fallidas.

     Mediante la función de servicio que especifique, se debe permitir el acceso de escritura a los recursos de su cuenta. Para obtener más información sobre la creación de funciones de IAM, consulte Creación de una función para un AWS servicio en la Guía del usuario de IAM.
8. Elija Guardar cambios.
9. De vuelta en la página Mensajería de texto móvil (SMS)), vaya a la sección Registros de estado de entrega para ver los registros disponibles.

> **Note**
>
> Según el operador del número de teléfono de destino, los registros de entrega pueden tardar hasta 72 horas en aparecer en la consola de Amazon SNS.

Registro de ejemplo para una entrega de SMS correcta

El log de estado de una entrega de SMS correcta será similar al ejemplo siguiente:

```
{
    "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"
}
```

Registro de ejemplo para una entrega de SMS errónea

El log de estado de una entrega de SMS errónea será similar al ejemplo siguiente:

```
{
    "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 de error de entrega de SMS

El motivo de un error se proporciona con el atributo `providerResponse`. Es posible que los mensajes SMS no se puedan entregar por los motivos siguientes:

- El operador de telefonía lo bloquea por considerarlo spam.
- El destino está en una lista bloqueada
- Número de teléfono no válido.
- Cuerpo de mensaje no válido.
- El operador de telefonía ha bloqueado este mensaje.
- El operador de telefonía no está disponible o no es posible ponerse en contacto con él.

- El teléfono ha bloqueado los SMS.
- El teléfono está en una lista bloqueada
- El teléfono no está disponible o no es posible ponerse en contacto con él.
- Se ha cancelado la suscripción del número de teléfono.
- Esta entrega superaría el precio máximo.
- Error desconocido al intentar ponerse en contacto con el teléfono

Suscripción a informes de uso diario de SMS en Amazon SNS

Puede monitorear sus entregas de SMS si se suscribe a los informes de uso diarios desde Amazon SNS. Todos los días que envía, como mínimo, un SMS, Amazon SNS entrega un informe de uso en formato CSV al bucket de Amazon S3 especificado. El informe de uso de SMS tarda 24 horas en estar disponible en el bucket de Amazon S3.

Información del informe de uso diario

El informe de uso contiene la siguiente información de cada mensaje SMS que envíe desde su cuenta.

Tenga en cuenta que el informe no incluye los mensajes que se envían a los destinatarios que han desactivado la recepción de mensajes.

- Hora de publicación del mensaje (en UTC)
- Message ID
- Número de teléfono de destino
- Tipo de mensaje
- Estado de entrega.
- Precio del mensaje (en USD).
- Número de parte (un mensaje se divide en varias partes si es demasiado largo para un único mensaje).
- Número total de partes.

> **Note**
>
> Si Amazon SNS no recibió el número de parte, establecemos su valor en cero.

Suscripción a informes de uso diario

Para suscribirse a informes de uso diario, debe crear un bucket de Amazon S3 con los permisos pertinentes.

Si quiere crear un bucket de Amazon S3 para sus informes de uso diario, siga estos pasos:

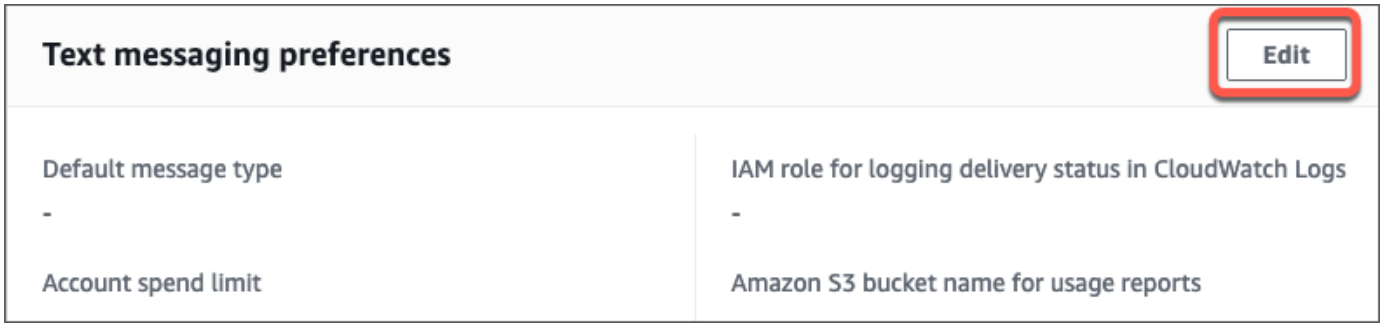
1. Desde la Cuenta de AWS que envía los mensajes SMS, inicie sesión en la consola de Amazon S3.
2. Seleccione la opción Crear bucket.
3. En Bucket Name (Nombre del bucket), le recomendamos que escriba un nombre único para su cuenta y su organización. Por ejemplo, use el patrón `<my-bucket-prefix>-<account_id>-<org-id>`.

   Para obtener información sobre las convenciones y restricciones de los nombres de bucket, consulte Reglas para la nomenclatura de bucket en la Guía del usuario de Amazon Simple Storage Service.
4. Seleccione Crear.
5. En la tabla Todos los buckets, elija el bucket.
6. En la pestaña Permisos, elija Política de bucket.
7. En la ventana Editor de políticas de bucket, indique una política que permita al principal del servicio de Amazon SNS escribir en el bucket. Para ver un ejemplo, consulta Ejemplo de política de bucket.

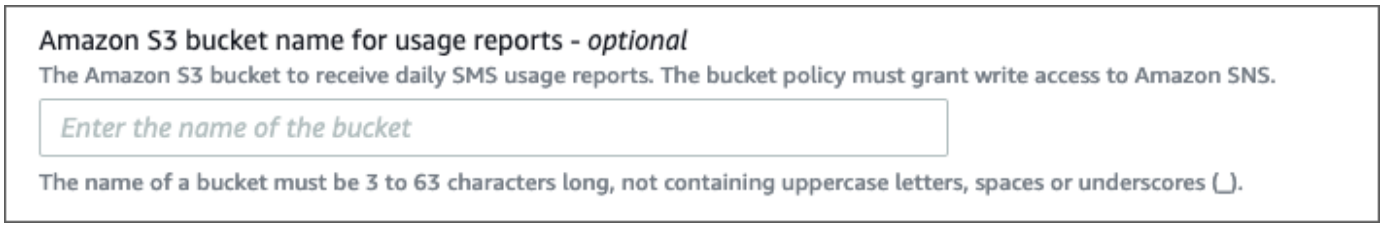
   Si utilizas la política de ejemplo, recuerda *`my-s3-bucket`* sustituirla por el nombre del bucket que elegiste en el paso 3.
8. Seleccione Save.

Para suscribirse a los informes de uso diario

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Text messaging (SMS) (Mensajería de texto (SMS)).
3. En la página Mensajería de texto (SMS), en la sección Preferencias de mensajería de texto, elija Editar.



4. En la página Edit text messaging preferences (Editar preferencias de mensajería de texto), en la sección Details (Detalles), especifique el Amazon S3 bucket name for usage reports (Nombre del bucket de Amazon S3 para los informes de uso).



5. Seleccione Save changes (Guardar cambios).

Ejemplo de política de bucket

Con la siguiente política, la entidad principal del servicio de Amazon SNS puede ejecutar las acciones `s3:PutObject`, `s3:GetBucketLocation` y `s3:ListBucket`.

AWS proporciona herramientas para todos los servicios con los directores de servicio a los que se les ha dado acceso a los recursos de su cuenta. Cuándo la entidad principal de una instrucción de política de bucket de Amazon S3 es un problema de suplente confuso. Para limitar la región y la cuenta desde las que el bucket puede recibir informes de uso diarios, utilice `aws:SourceArn` tal como se muestra en el siguiente ejemplo. Si no desea limitar las regiones que pueden generar estos informes, utilice `aws:SourceAccount` para establecer límites en función de qué cuenta esté generando los informes. Si no conoce el ARN del recurso, utilice `aws:SourceAccount`.

Utilice el siguiente ejemplo que incluye protección contra suplente confuso cuando cree un bucket de Amazon S3 para recibir informes de uso diario de SMS desde 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**
>
> Puede publicar informes de uso en buckets de Amazon S3 que sean propiedad de la Cuenta de AWS que se especifica en el elemento `Condition` en la política de Amazon S3. Para publicar informes de uso en un bucket de Amazon S3 que Cuenta de AWS pertenece a otro, consulte ¿Cómo puedo copiar objetos de Amazon S3 de otro Cuenta de AWS? .

Ejemplo de informe de uso diario

Después de suscribirse a informes de uso diario, cada día Amazon SNS pone un archivo CSV con datos de uso en la siguiente ubicación:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Cada archivo puede contener hasta 50 000 registros. Si los registros de un día superan esta cuota, Amazon SNS agregará varios archivos. A continuación se muestra un informe de ejemplo:

```
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
```

Solicitud de asistencia para mensajería SMS en Amazon SNS

> **Important**
>
> Se ha actualizado la guía para desarrolladores de SMS de Amazon SNS. Amazon SNS se ha integrado con AWS End User Messaging SMS para la entrega de mensajes SMS. Esta guía contiene la información más reciente sobre cómo crear, configurar y administrar los mensajes SMS de Amazon SNS.

Algunas opciones de SMS de Amazon SNS no están disponibles para tu AWS cuenta hasta que te pongas en contacto con nosotros. Soporte Abra un caso en el centro de AWS Support para realizar cualquiera de las siguientes solicitudes:

- Un aumento en su umbral de gasto mensual de SMS

  De forma predeterminada, el umbral de gasto mensual es de 1,00 USD. El umbral de gasto determina el volumen de mensajes que puede enviar con Amazon SNS. Puede solicitar un umbral de gasto con el que se obtenga el volumen mensual de mensajes esperado para su caso de uso de SMS.
- Salir del entorno de pruebas de SMS para que pueda enviar mensajes SMS sin restricciones. Para obtener más información, consulte Cómo salir del entorno de pruebas de SMS de Amazon SNS.
- Un número de origen dedicado
- Un ID de remitente dedicado Un ID de remitente es un ID personalizado que se muestra como remitente en el dispositivo del destinatario. Por ejemplo, puede utilizar la marca de su negocio para facilitar el reconocimiento del origen del mensaje. El soporte para el remitente IDs varía según el país o la región. Para obtener más información, consulte el tema Regiones y países admitidos con AWS End User Messaging SMS en la Guía del usuario de AWS End User Messaging SMS .

Solicitud de aumento de la cuota de gasto mensual de SMS de Amazon SNS

Amazon SNS proporciona cuotas de gastos para ayudarlo a administrar el costo máximo por mes en el que se incurre al enviar SMS a través de su cuenta. La cuota de gasto limita el riesgo en caso de un ataque malicioso e impide que la aplicación ascendente envíe más mensajes de los esperados. Puede configurar Amazon SNS para que deje de publicar mensajes SMS cuando determine que el envío de un mensaje SMS generará un costo que superará la cuota de gasto del mes actual.

Para garantizar que sus operaciones no se vean afectadas, le recomendamos que solicite una cuota de gasto lo suficientemente alta como para admitir sus cargas de trabajo de producción. Para obtener más información, consulte Paso 1: Abrir un caso de SMS de Amazon SNS. Una vez que haya recibido la cuota, podrá administrar su riesgo aplicando la cuota completa o un valor menor, como se describe en Paso 2: Actualizar la configuración de SMS. Si aplica un valor menor, podrá controlar el gasto mensual con la opción de aumentarlo si resultara necesario.

> **Important**
>
> Como Amazon SNS es un sistema distribuido, deja de enviar mensajes SMS en cuestión de minutos si se supera la cuota de gasto. Durante este período, si sigue enviando mensajes SMS, podría incurrir en costos que superen su cuota.

Establecemos la cuota de gasto para todas las cuentas nuevas en 1,00 USD al mes. Esta cuota está pensada para probar las funcionaliades de envío de mensajes de Amazon SNS. Para solicitar un aumento de la cuota de gastos de SMS de su cuenta, abra un caso de aumento de cuota en el AWS Support Center.

Temas

- Paso 1: Abrir un caso de SMS de Amazon SNS
- Paso 2: Actualizar la configuración de SMS en la consola de Amazon SNS

Paso 1: Abrir un caso de SMS de Amazon SNS

Puede solicitar un aumento de su cuota de gasto mensual abriendo un caso de aumento de cuota en el AWS Support Center.

> **Note**
>
> Algunos de los campos en el formulario de solicitud están marcados como opcionales. Sin embargo, Soporte requiere toda la información mencionada en los pasos siguientes con el fin de procesar su solicitud. Si no proporciona toda la información necesaria, puede experimentar retrasos en el procesamiento de su solicitud.

1. Inicie sesión AWS Management Console en https://console.aws.amazon.com/.

2. En el menú Soporte, elija Centro de soporte.
3. En la pestaña Casos de soporte abiertos, elija Crear caso.
4. Elija el enlace ¿Busca aumentos en el límite de servicio? y, a continuación, complete lo siguiente:

   - En el tipo de límite, selecciona SNS Mensajería de texto.
   - (Opcional) En Proporcionar un enlace al sitio o aplicación que enviará los mensajes SMS, proporcione información sobre el sitio web, la aplicación o el servicio que enviará los mensajes SMS.
   - (Opcional) En Tipo de mensaje que tiene previsto enviar, elija el tipo de mensaje que tiene previsto enviar con el código largo:
     - Contraseña de un solo uso: mensajes que proporcionan contraseñas que sus clientes utilizan para autenticarse con su sitio web o aplicación.
     - Promocional: mensajes no importantes que promocionan su empresa o servicio, tales como anuncios u ofertas especiales.
     - Transaccional: mensajes informativos importantes que admiten transacciones del cliente, tales como confirmaciones de pedido o alertas de transacción. Los mensajes transaccionales no pueden incluir contenido promocional ni de marketing.
   - (Opcional) Para saber desde qué AWS región vas a enviar los mensajes, selecciona la región desde la que vas a enviar los mensajes.
   - (Opcional) En Países a los que tiene previsto enviar mensajes, introduzca el país o la región en el que quiere comprar códigos cortos.
   - (Opcional) En Cómo deciden sus clientes recibir mensajes suyos, facilite detalles sobre su proceso de suscripción.
   - (Opcional) En el campo Indique la plantilla de mensajes que piensa utilizar para enviar mensajes a sus clientes, incluya la plantilla que vaya a utilizar.
5. En Solicitudes, complete las secciones siguientes:

   - En Región, elija la región desde la que va a enviar los mensajes.

   > **Note**
   >
   > La región es obligatoria en la sección Solicitudes. Incluso si proporcionó esta información en la sección Detalles del caso, también debe incluirla aquí.

- En Tipo de recurso, elija Límites generales.
- En Límite, elija Aumento del umbral de gasto de cuenta.

6. En Nuevo valor de límite, escriba el importe máximo (en USD) que desea gastar en SMS cada mes natural.
7. En Descripción del caso, en Descripción del caso de uso, proporcione la siguiente información:

   - El sitio web o la aplicación de la empresa o servicio que envía mensajes SMS.
   - El servicio que ofrece su sitio web o aplicación y cómo los mensajes SMS contribuyen a dicho servicio.
   - Cómo se inscriben los usuarios para recibir voluntariamente sus mensajes SMS en su sitio web, aplicación u otra ubicación.

   Si la cuota de gasto solicitada (el valor especificado para Nuevo valor de cuota) es superior a 10 000 USD, proporcione los siguientes datos adicionales para cada país al que envíe mensajes:

   - Si utiliza un ID de remitente o un código corto. Si utiliza un ID de remitente, proporcione:
     - El ID de remitente.
     - Si el ID de remitente está registrado con operadores inalámbricos en el país.
   - El máximo esperado transactions-per-second (TPS) para tus mensajes.
   - El tamaño medio del mensaje.
   - La plantilla para los mensajes que envía en el país.
   - (Opcional) Necesidades de codificación de caracteres, si procede.
8. (Opcional) Si desea enviar más solicitudes, elija Agregar otra solicitud. Si incluye varias solicitudes, proporcione la información necesaria para cada una de ellas. Para la información requerida, consulte las demás secciones en Solicitud de asistencia para mensajería SMS en Amazon SNS.
9. En Opciones de contacto, elija en Idioma de contacto preferido si prefiere recibir las comunicaciones de este caso.
10. Cuando haya terminado, elija Enviar.

El Soporte equipo proporcionará una respuesta inicial a tu solicitud en un plazo de 24 horas.

Para evitar que nuestros sistemas se utilicen para enviar contenido no solicitado o malicioso, consideramos cada solicitud con detenimiento. Si podemos, accederemos a su solicitud dentro de ese plazo de 24 horas. Sin embargo, si necesitamos que nos brinde más información, puede que la solicitud tarde más tiempo en concederse.

Si su caso de uso no se ajusta a nuestras políticas, es posible que no podamos atender su solicitud.

Paso 2: Actualizar la configuración de SMS en la consola de Amazon SNS

Cuando le avisemos del aumento de la cuota de gasto mensual, tendrá que ajustar la cuota de gasto de su cuenta en la consola de Amazon SNS.

> **Important**
>
> Debe completar los siguientes pasos; de lo contrario, no se aumentará su límite de gasto en SMS.

Para ajustar la cuota de gasto en la consola

1. Inicie sesión en la consola de Amazon SNS.
2. Abra el menú de navegación de la izquierda, expanda Móvil y, a continuación, elija Mensajería de texto (SMS).
3. En la página Mensajería de texto a través del móvil (SMS), en la sección Preferencias de mensajería de texto, elija Editar.
4. En la página Editar preferencias de mensajería de texto, en la sección Detalles, introduzca su nuevo límite de gasto de SMS en el campo Límite de gasto de la cuenta.

   > **Note**
   >
   > Es posible que reciba una advertencia de que el valor especificado es mayor que el límite de gasto predeterminado. Puede omitirla.

5. Elija Guardar cambios.

> **Note**
>
> Si aparece un error de “parámetro no válido”, compruebe el contacto del servicio de asistencia de AWS y confirme que ha especificado el nuevo límite de gasto de SMS correcto. Si sigue teniendo problemas, abra un caso en el AWS Support Center.

Cuando crees tu caso en el Soporte Centro, asegúrate de incluir toda la información necesaria para el tipo de solicitud que vayas a enviar. En caso contrario, Soporte debe ponerse en contacto con usted para obtener esta información antes de continuar. Al enviar un caso detallado, contribuye a garantizar que su caso se gestione sin retrasos. Para conocer los detalles para determinados tipos de solicitudes de SMS, consulte los siguientes temas.

Para obtener más información sobre el remitente IDs, consulte la siguiente documentación en la Guía del AWS End User Messaging SMS usuario:

AWS End User Messaging SMS Tema	Descripción
Solicitud de un aumento de la cuota de gasto	Tu cuota de gastos determina cuánto dinero puedes gastar en el envío de mensajes SMS AWS End User Messaging SMS cada mes.
Abrir un caso en el centro de asistencia para obtener un ID de remitente	Si tienes pensado enviar mensajes a destinata rios de un país en el que IDs se requiera el remitente, puedes solicitar un identificador de remitente creando un nuevo caso en el Soporte Centro.

Prácticas recomendadas para la mensajería SMS de Amazon SNS

> **Important**
>
> Se ha actualizado la Guía para desarrolladores de SMS de Amazon SNS. Amazon SNS se ha integrado con AWS End User Messaging SMS para la entrega de mensajes SMS. Esta guía contiene la información más reciente sobre cómo crear, configurar y administrar los mensajes SMS de Amazon SNS.

Los usuarios de teléfonos móviles suelen tener una tolerancia muy baja a los mensajes SMS no solicitados. Las tasas de respuesta de las campañas de mensajes SMS no solicitados casi siempre serán bajas y, por tanto, obtendrá una baja rentabilidad de su inversión.

Además, los operadores de telefonía móvil auditan de forma continua a los remitentes de mensajes SMS masivos. Limitan o bloquean los mensajes de números que determinan que están enviando mensajes no solicitados.

El envío de contenido no solicitado también es una infracción de la política de uso aceptable de AWS. El equipo de Amazon SNS audita con regularidad las campañas de SMS y podría limitar o bloquear su capacidad de enviar mensajes si le consta que está enviando mensajes no solicitados.

Por último, en muchos países, regiones y jurisdicciones, existen diversas sanciones por el envío de mensajes SMS no solicitados. Por ejemplo, en Estados Unidos, la Ley federal de Protección al Usuario Telefónico (TCPA) establece que los consumidores tienen derecho a una indemnización de entre 500 y 1500 USD en concepto de daños (pagados por el remitente) por cada mensaje no solicitado que reciban.

En esta sección se describen diversas prácticas recomendadas que podrían ayudarle a mejorar la interacción con los clientes y evitar sanciones costosas. Sin embargo, tenga en cuenta que esta sección no contiene asesoramiento jurídico. Consulte siempre a un abogado para obtener asesoramiento jurídico.

Cumpla las leyes, normativas y requisitos del operador.

Puede enfrentarse a importantes multas y sanciones si infringe las leyes y normativas de los lugares en los que residen sus clientes. Por este motivo, es fundamental que conozca las leyes relacionadas con la mensajería SMS de cada país o región en los que haga negocios.

La lista siguiente incluye enlaces a las leyes más importantes que se aplican a las comunicaciones SMS en los principales mercados de todo el mundo.

- Estados Unidos: la Ley Federal de Protección al Usuario Telefónico de 1991, también denominada TCPA, se aplica a ciertos tipos de mensajes SMS. Para obtener más información, consulte las normativas y reglamentos en el sitio web de la Comisión Federal de Comunicaciones.
- Reino Unido: las Normativas sobre la Privacidad y las Comunicaciones Electrónicas (directiva CE) de 2003, también denominada PECR, se aplican a ciertos tipos de mensajes SMS. Para obtener más información, consulte What are PECR? en el sitio web de la oficina del comisionado de información del Reino Unido.

- Unión Europea: la Directiva sobre la Privacidad y las Comunicaciones Electrónicas de 2002, también denominada directiva ePrivacy, se aplica a algunos tipos de mensajes SMS. Para obtener más información, consulte el texto completo de la ley en el sitio web Europa.eu.
- Canadá: la Ley de Lucha contra el Spam Inalámbrico y de Internet, más comúnmente denominada Legislación AntiSpam de Canadá o CASL, se aplica a ciertos tipos de mensajes SMS. Para obtener más información, consulte el texto completo de la ley en el sitio web del parlamento canadiense.
- Japón: la Ley del Reglamento de Transmisión de Correo Electrónico Específico puede aplicarse a ciertos tipos de mensajes SMS. Para obtener más información, consulte las contramedidas de Japón contra el spam en el sitio web del Ministerio Japonés de Asuntos Internos y Comunicaciones.

Como remitente, estas leyes pueden aplicarse a usted incluso si su empresa u organización no tiene su sede en uno de estos países. Algunas de las leyes de esta lista se crearon originalmente para abordar el problema de las llamadas telefónicas o del correo electrónico no solicitados, pero se han interpretado o ampliado también para su aplicación a los mensajes SMS. Otros países y regiones pueden tener sus propias leyes relacionadas con la transmisión de mensajes SMS. Consulte a un abogado de cada país o región en la que se encuentren sus clientes para obtener asesoramiento jurídico.

En muchos países, los operadores locales tienen, en última instancia, la autoridad para determinar qué tipo de tráfico fluye a través de sus redes. Esto significa que los operadores podrían imponer restricciones al contenido de los SMS que superen los requisitos mínimos de las leyes locales.

Obtención de permiso

Nunca envíe mensajes a destinatarios que no hayan pedido de forma explícita recibir los tipos específicos de mensajes que tiene pensado enviar. No comparta listas de suscripciones voluntarias, ni siquiera entre organizaciones de la misma empresa.

Si los destinatarios pueden inscribirse para recibir sus mensajes mediante un formulario en línea, añada sistemas que impidan que los scripts automatizados suscriban a las personas sin su conocimiento. También debe limitar el número de veces que un usuario puede enviar un número de teléfono en una sola sesión.

Si recibe una solicitud de susripción voluntaria por SMS, envíe un mensaje al destinatario para pedirle que confirme que desea recibir mensajes suyos. No envíe a dicho destinatario ningún

mensaje adicional hasta que confirme su suscripción. Un mensaje de confirmación de suscripción podría tener un aspecto similar al siguiente:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

Mantenga registros que incluyan la fecha, la hora y el origen de cada solicitud de alta y confirmación. Esto puede ser útil si un operador o agencia reguladora lo solicita y también puede ayudarle a realizar auditorías rutinarias de su lista de clientes.

Flujo de trabajo de suscripciones voluntarias

En algunos casos (como en el registro mediante una llamada gratuita o código abreviado en EE. UU.), los operadores de telefonía móvil exigen que proporcione simulaciones o capturas de pantalla de todo el flujo de trabajo de suscripción voluntaria. Las simulaciones o la captura de pantalla deben parecerse mucho al flujo de trabajo de suscripción que completarán los destinatarios.

Sus simulaciones o capturas de pantalla deben incluir todas las declaraciones obligatorias que se enumeran a continuación para mantener el nivel más alto de conformidad.

Declaraciones obligatorias

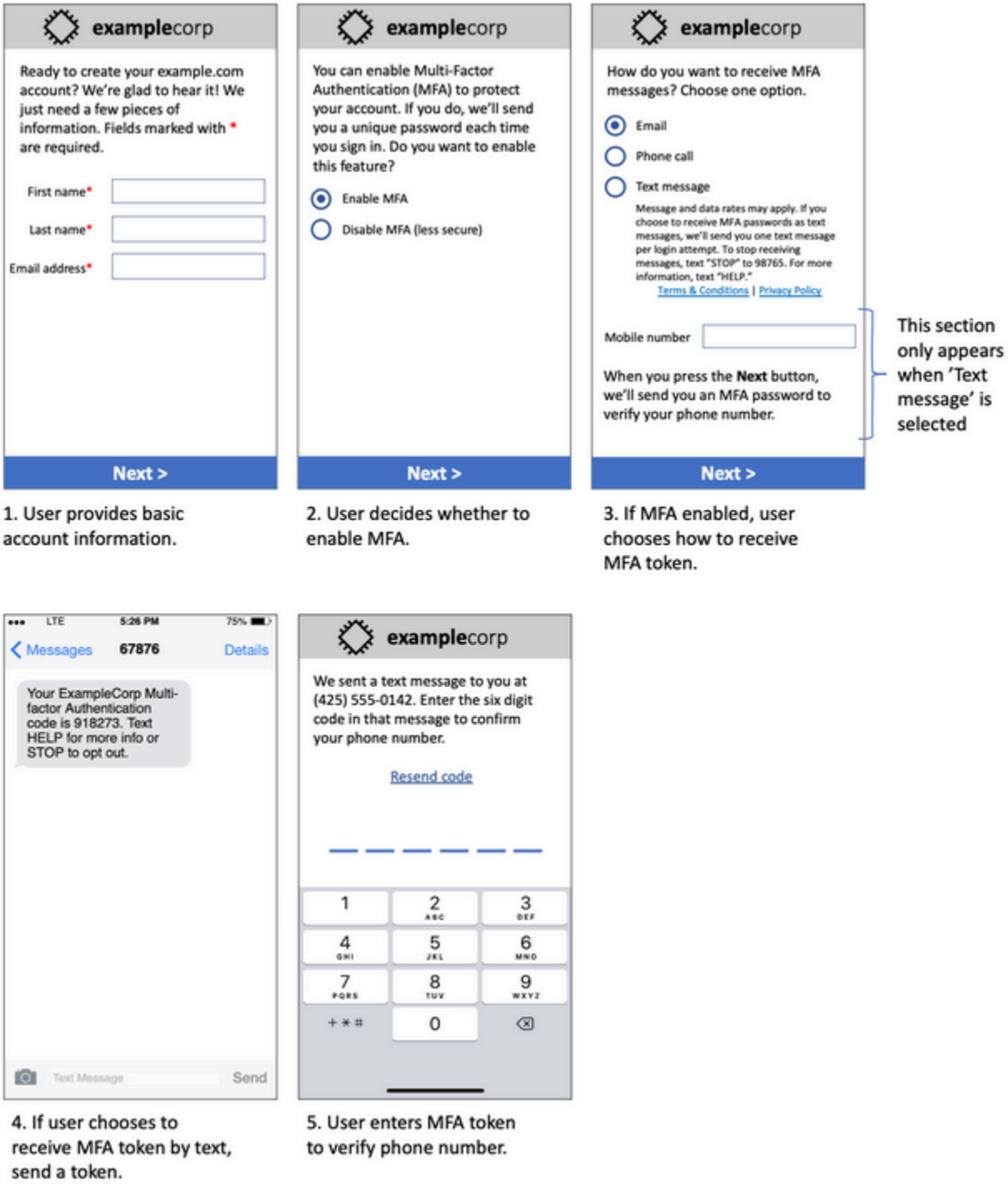
- Una descripción del caso de uso de mensajería que enviará a través de su programa.
- La frase “Se podrían aplicar cargos por el uso de datos y el envío de mensajes”.
- Una indicación de la frecuencia con la que los destinatarios recibirán sus mensajes. Por ejemplo, un programa de mensajería recurrente podría decir “un mensaje a la semana”. En el caso de uso de autenticación multifactor o contraseña de un solo uso, se podría indicar lo siguiente: “la frecuencia de los mensajes varía” o “un mensaje por intento de inicio de sesión”.
- Enlaces a los documentos de términos y condiciones y la política de privacidad.

Motivos comunes de rechazo de sucripciones voluntarias no válidas

- Si el nombre de la empresa proporcionado no coincide con el que aparece en la simulación o captura de pantalla. Cualquier relación que no sea obvia debe explicarse en la descripción del flujo de trabajo de suscripción.
- Si parece que se va a enviar un mensaje al destinatario, pero no se obtiene ningún consentimiento explícito para hacerlo. El consentimiento explícito es un requisito de todos los mensajes.

- Si parece que es necesario recibir un mensaje de texto para suscribirse a un servicio. Esto no es válido si el flujo de trabajo no ofrece ninguna alternativa a recibir un mensaje de suscripción voluntaria en otro formato, como un correo electrónico o una llamada de voz.
- Si el texto de la suscripción voluntaria se encuentra en su totalidad en los términos de servicio. Las declaraciones siempre deben presentarse al destinatario en el momento de la suscripción, en lugar de incluirse en un documento de política vinculado.
- Si un cliente ha dado su consentimiento para recibir un tipo de mensaje y le envía otro tipo de mensaje de texto. Por ejemplo, ha aceptado recibir contraseñas de un solo uso, pero también se le envían mensajes de sondeo y encuestas.
- Si las declaraciones obligatorias (indicadas anteriormente) no se presentan a los destinatarios.

El siguiente ejemplo cumple los requisitos de los operadores de telefonía móvil en un caso de uso de autenticación multifactor.



Simulación de un caso de uso de autenticación multifactor

Contiene texto e imágenes finales y muestra todo el flujo de suscripción voluntaria con anotaciones. En el flujo de suscripción, el cliente debe tomar medidas claras e intencionadas para dar su consentimiento para recibir mensajes de texto y se incluyen todas las declaraciones obligatorias.

Otros tipos de flujo de trabajo de suscripción voluntaria

Los operadores de telefonía móvil también aceptan flujos de trabajo de suscripción voluntaria fuera de las aplicaciones y sitios web, como las suscripciones verbales o por escrito, si cumplen lo que se ha indicado anteriormente. Un flujo de trabajo de suscripción voluntaria y un guion verbal o escrito conformes con los requisitos sirven para recoger el consentimiento explícito del destinatario para recibir un tipo de mensaje específico. Un ejemplo sería el guion verbal que utiliza un agente de asistencia para obtener el consentimiento antes de registrar a alguien en una base de datos de servicios o un número de teléfono que aparece en un folleto promocional. Para crear una maqueta de estos tipos de flujos de trabajo opcionales, puedes incluir una captura de pantalla del guion, el material de marketing o la base de datos en la que se recopilan los números. Los operadores de telefonía móvil podrían hacer preguntas adicionales sobre estos casos de uso si el proceso de suscripción voluntaria no está claro o si el caso de uso supera ciertos volúmenes.

No enviar a listas antiguas

La gente cambia de número de teléfono con frecuencia. Es posible que el número de teléfono del que obtuvo el consentimiento para contactar con esa persona hace dos años pertenezca ahora a otra persona. No utilice una lista antigua de números de teléfono para un nuevo programa de mensajería; de lo contrario, es probable que algunos mensajes fallen porque el número ya no está en servicio y que algunas personas se den de baja por no recordar haberle dado nunca su consentimiento.

Auditoría de las listas de clientes

Si envía campañas de SMS recurrentes, audite periódicamente sus listas de clientes. La auditoría de su lista de clientes garantiza que los únicos clientes que reciban sus mensajes sean aquellos interesados en recibirlos.

Al auditar su lista, envíe a cada cliente que ha solicitado el alta un mensaje que le recuerde que ya está suscrito y facilítele información sobre cómo anular la suscripción. Un mensaje de recordatorio podría tener un aspecto similar al siguiente:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

Mantenimiento de registros

Mantenga registros que muestren cuándo ha solicitado cada cliente que usted le envíe mensajes SMS y qué mensajes envía a cada cliente. Muchos países y regiones de todo el mundo requieren

que los remitentes de SMS mantengan estos registros de forma que se puedan recuperar fácilmente. Los operadores de telefonía móvil también podrían pedirle esta información en cualquier momento. La información exacta que debe proporcionar varía según el país o la región. Para obtener más información sobre los requisitos de conservación de registros, lea la normativa sobre mensajería SMS de cada país o región en que se encuentren sus clientes.

En algunas ocasiones, un operador o una agencia reguladora nos solicita que proporcionemos una prueba de que un cliente aceptó recibir mensajes de usted. En estas situaciones, se pondrá en Soporte contacto con usted para proporcionarle una lista de la información que necesita el transportista o la agencia. Si no puede proporcionar la información necesaria, podemos suspender su capacidad de enviar mensajes SMS adicionales.

Procure que sus mensajes sean claros, sinceros y concisos

Los SMS son un medio único. El character-per-message límite de 160 significa que tus mensajes deben ser concisos. Es posible que las técnicas que se utilizan en otros canales de comunicación, como el correo electrónico, no se apliquen al canal de SMS e incluso parezcan deshonestas o engañosas cuando se utilizan con mensajes SMS. Si el contenido de sus mensajes no se ajusta a las prácticas recomendadas, es posible que los destinatarios hagan caso omiso de sus mensajes. En el peor de los casos, los operadores de telefonía móvil podrían considerar sus mensajes como spam y bloquear los mensajes que procedan de su número de teléfono.

En esta sección se ofrecen algunos consejos e ideas para crear un cuerpo de mensaje SMS eficaz.

Identifíquese como remitente

Sus destinatarios deberían poder saber inmediatamente que el mensaje es suyo. Los remitentes que siguen esta práctica recomendada utilizan un nombre identificativo (“nombre del programa”) al principio de cada mensaje.

No haga lo siguiente:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Mejor, pruebe esto:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

No intentes hacer que tu mensaje parezca un person-to-person mensaje

Algunos profesionales de marketing se sienten tentados a darle un toque personal a sus mensajes SMS haciendo que parezcan provenir de una persona. Sin embargo, esta técnica puede hacer que parezca un intento de suplantación de identidad.

No haga lo siguiente:

```
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.
```

Mejor, pruebe esto:

```
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.
```

Tenga cuidado cuando hable de dinero

Los estafadores suelen aprovecharse del deseo de las personas de ahorrar y ganar dinero. No haga que las ofertas parezcan demasiado buenas para ser verdad. No intente engañar a la gente con el cebo del dinero. No utilice símbolos de monedas para hacer referencia al dinero.

No haga lo siguiente:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Mejor, pruebe esto:

```
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.
```

Use solo los caracteres necesarios

Las empresas suelen incluir símbolos de marca comercial como ™ o ® en sus mensajes para proteger sus marcas comerciales. Sin embargo, estos símbolos no forman parte del conjunto de caracteres estándar (conocido como alfabeto GSM) que se puede incluir en un mensaje SMS de 160 caracteres. Cuando envía un mensaje que contiene uno de estos caracteres, se utiliza automáticamente un sistema de codificación de caracteres diferente, que solo admite 70 caracteres

por cada parte del mensaje. Como resultado, el mensaje podría dividirse en varias partes. Dado que se le factura por cada parte del mensaje, enviar el mensaje completo podría costarte más de lo que esperaba. Además, es posible que sus destinatarios reciban varios mensajes secuenciales en lugar de un solo mensaje. Para obtener más información acerca de la codificación de caracteres SMS, consulte Límites de caracteres de SMS en Amazon SNS.

No haga lo siguiente:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Mejor, pruebe esto:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> Los dos ejemplos anteriores son casi idénticos, pero el primero contiene un símbolo de marca registrada (®), que no forma parte del alfabeto GSM. Por consiguiente, el primer ejemplo se envía en dos partes, mientras que el segundo ejemplo se envía como un solo mensaje.

Use enlaces válidos y seguros

Si el mensaje incluye enlaces, compruébelos para asegurarse de que funcionan. Pruebe los enlaces en un dispositivo fuera de la red corporativa para asegurarse de que se resuelven correctamente. Debido al límite de 160 caracteres de los mensajes SMS, los mensajes muy largos URLs pueden dividirse en varios mensajes. Deberías usar dominios de redireccionamiento para acortarlos. URLs Sin embargo, no debe utilizar servicios gratuitos de acortamiento de enlaces, como tinyurl.com o bitly.com, porque los operadores suelen filtrar los mensajes que incluyen enlaces de estos dominios. Sin embargo, puede usar servicios de acortamiento de enlaces de pago siempre que sus enlaces apunten a un dominio que esté dedicado a un uso exclusivo por parte de su empresa u organización.

No haga lo siguiente:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Mejor, pruebe esto:

```
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.
```

Limite la cantidad de abreviaturas

La limitación de 160 caracteres del canal de SMS lleva a algunos remitentes a utilizar muchas abreviaturas en sus mensajes. Sin embargo, el uso excesivo de abreviaturas les puede resultar poco profesional a muchos lectores y algunos usuarios podrían marcar su mensaje como spam. Es totalmente posible escribir un mensaje coherente sin utilizar un número excesivo de abreviaturas.

No haga lo siguiente:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Mejor, pruebe esto:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Responda correctamente

Cuando un destinatario conteste a sus mensajes, asegúrese de responder con información útil. Por ejemplo, cuando un cliente responde a uno de sus mensajes con la palabra clave "HELP", envíele información sobre el programa al que está suscrito, el número de mensajes que va a enviar cada mes y las formas en que puede ponerse en contacto con usted para obtener más información. Una respuesta a un mensaje HELP podría tener un aspecto similar al siguiente:

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Cuando un cliente responda con la palabra clave "STOP", hágale saber que ya no recibirá más mensajes. Una respuesta STOP podría tener un aspecto similar al siguiente:

```
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.
```

Ajuste el envío en función del interés

Las prioridades de los clientes pueden cambiar a lo largo del tiempo. Si los clientes dejan de encontrar útiles sus mensajes, podrían darse de baja de sus mensajes o incluso registrar los mensajes como no solicitados. Por estas razones, es importante que ajuste sus prácticas de envío en función del interés de los clientes.

Para los clientes que no suelan interaccionar con sus mensajes, debe ajustar la frecuencia de los mismos. Por ejemplo, si envía mensajes semanales a clientes interesados, podría crear un resumen mensual independiente para los clientes menos interesados.

Por último, elimine de su lista a aquellos clientes que no muestren ningún interés. Este paso evita que los clientes se sientan frustrados con sus mensajes. También le ahorra dinero y ayuda a proteger su reputación como remitente.

Envíe los mensajes en los momentos apropiados

Envíe mensajes solo durante el horario laborable diurno normal. Si envía mensajes a la hora de la cena o a medianoche, hay muchas probabilidades de que sus clientes se den de baja de sus listas para evitar ser molestados. Además, no tiene sentido enviar mensajes SMS cuando los clientes no pueden responder a ellos de forma inmediata.

Si envía campañas o recorridos a audiencias muy grandes, compruebe bien las tasas de rendimiento de sus números de origen. Divida el número de destinatarios entre su tasa de rendimiento para determinar cuánto tiempo tardará en enviar los mensajes a todos sus destinatarios.

Evite la fatiga del uso de distintos canales

En sus campañas, si utiliza varios canales de comunicación (como, por ejemplo, correo electrónico, SMS y mensajes push), no envíe el mismo mensaje en cada canal. Al enviar el mismo mensaje a la vez en más de un canal, los clientes perciben su comportamiento de envío como molesto en lugar de útil.

Utilice códigos cortos dedicados

Si utiliza códigos cortos, mantenga un código corto independiente para cada marca y cada tipo de mensaje. Por ejemplo, si su empresa tiene dos marcas, utilice un código corto independiente para cada una de ellas. Del mismo modo, si se envían mensajes transaccionales y promocionales, utilice un código corto independiente para cada tipo de mensaje. Para obtener más información sobre la

solicitud de códigos cortos, consulte Solicitud de códigos cortos para la mensajería SMS con AWS End User Messaging SMS en la Guía del usuario de AWS End User Messaging SMS .

Verifique los números de teléfono de destino

Cuando envía mensajes SMS a través de Amazon SNS, se le factura por cada parte del mensaje que envía. El precio que paga por cada parte del mensaje varía en función del país o la región del destinatario. Para obtener información acerca de los precios de SMS, consulte AWS Worldwide SMS Pricing.

Cuando Amazon SNS acepta una solicitud para enviar un mensaje SMS (como resultado de una llamada a la SendMessagesAPI o como resultado del lanzamiento de una campaña o un viaje), se te cobrará por enviar ese mensaje. Esto es así incluso si el destinatario previsto no recibe el mensaje al final. Por ejemplo, aunque el número de teléfono del destinatario ya no esté en servicio o el número al que le ha enviado el mensaje no fuera un número de teléfono móvil válido, se le facturará por enviar el mensaje.

Amazon SNS acepta solicitudes válidas para enviar mensajes SMS e intenta entregarlos. Por este motivo, debe asegurarse de que los números de teléfono a los que envía los mensajes sean números de teléfono móviles válidos. Puedes usarlo AWS End User Messaging SMS para enviar un mensaje de prueba para determinar si un número de teléfono es válido y qué tipo de número es (por ejemplo, móvil, fijo o VoIP). Para obtener más información, consulte Envío de un mensaje de prueba con el simulador de SMS en la Guía del usuario de AWS End User Messaging SMS .

Diseñe teniendo en cuenta la redundancia

Para los programas de mensajería de misión crítica, le recomendamos que configure Amazon SNS en más de una Región de AWS. Amazon SNS está disponible en varios. Regiones de AWS Para obtener una lista de las regiones en las que está disponible Amazon SNS, consulte la Referencia general de AWS.

Los números de teléfono que usa para los mensajes SMS, incluidos los códigos cortos, los códigos largos, los números gratuitos y los números 10DLC, no se pueden replicar en las Regiones de AWS. Por lo tanto, para utilizar Amazon SNS en varias regiones, debe solicitar números de teléfono distintos en cada región en la que vaya a utilizarlo. Por ejemplo, si utilizas un código corto para enviar mensajes de texto a destinatarios en los Estados Unidos, tendrás que solicitar códigos cortos separados en cada uno de los Región de AWS que vayas a usar.

En algunos países, también puede usar varios tipos de números de teléfono para aumentar la redundancia. Por ejemplo, en los Estados Unidos, puede solicitar códigos cortos, números 10DLC

y números gratuitos. Cada uno de estos tipos de números de teléfono llega al destinatario por una ruta diferente. Tener varios tipos de números de teléfono disponibles, ya sea en el mismo número Región de AWS o repartidos entre varios Regiones de AWS, proporciona una capa adicional de redundancia, que puede ayudar a mejorar la resiliencia.

Límites y restricciones de SMS

Para conocer los límites y restricciones de SMS, consulte Límites y restricciones de SMS y MMS en la Guía del usuario de AWS End User Messaging SMS .

Administración de palabras clave de cancelación de la suscripción

Los destinatarios de los SMS pueden usar los dispositivos para cancelar la suscripción a los mensajes respondiendo con una palabra clave. Para obtener más información, consulte Desactivación de la recepción de mensajes SMS.

CreatePool

Utilice la acción de la API `CreatePool` para crear un grupo nuevo y asociar una identidad de origen especificada al grupo. Para obtener más información, consulte CreatePoolla referencia AWS End User Messaging SMS de la API.

PutKeyword

Utilice la acción de la API `PutKeyword` para crear o actualizar una configuración de palabras clave en un número de teléfono o grupo de origen. Para obtener más información, consulta la referencia de PutKeywordla AWS End User Messaging SMS API.

Administración de la configuración de números

Para administrar la configuración de los códigos cortos y largos dedicados que solicitó a AWS Support y que asignó a su cuenta, consulte Cambiar las capacidades de un número de teléfono con AWS CLI la entrada AWS End User Messaging SMS.

Límites de caracteres de SMS en Amazon SNS

Cada mensaje SMS puede contener hasta 140 bytes de información. El número de caracteres que se pueden incluir en cada mensaje SMS depende del tipo de caracteres que contiene el mensaje.

Si el mensaje solo utiliza caracteres del conjunto de caracteres GSM 03.38, también conocido como alfabeto GSM de 7 bits, puede contener hasta 160 caracteres. Si el mensaje contiene caracteres que

no pertenecen al conjunto de caracteres GSM 03.38, puede tener hasta 70 caracteres. Cuando se envía un mensaje SMS, Amazon SNS determina automáticamente la codificación más eficiente que puede utilizarse.

Cuando un mensaje supera el número máximo de caracteres, se divide en varias partes. Cuando los mensajes se dividen en varias partes, cada parte contiene información adicional sobre la parte del mensaje que la precede. Cuando el dispositivo del destinatario recibe partes de mensajes separados de esta forma, utiliza esta información adicional para asegurarse de que todas las partes del mensaje se muestran en el orden correcto. Según el operador móvil y del dispositivo del destinatario, es posible que se muestren varios mensajes como un solo mensaje o como una secuencia de mensajes separados. Como resultado, el número máximo de caracteres de cada parte del mensaje se reduce a 153 (para los mensajes que solo contienen caracteres GSM 03.38) o a 67 (para los mensajes que contienen otros caracteres). Puede calcular cuántas partes de mensajes contiene su mensaje antes de enviarlo utilizando las herramientas de calculadora de longitud de SMS, varias de las cuales están disponibles en línea. El tamaño máximo admitido de cualquier mensaje es de 1600 caracteres GSM o 630 caracteres no GSM. Para obtener más información acerca del rendimiento y el tamaño de los mensajes, consulte SMS character limits in Amazon Pinpoint en la Guía del usuario de Amazon Pinpoint.

Para ver el número de partes del mensaje de cada mensaje que envíe, primero debe habilitar los ajustes de streaming de eventos. Al hacerlo, Amazon SNS produce un evento `_SMS.SUCCESS` cuando el mensaje se entrega al proveedor de telefonía móvil del destinatario. El registro de eventos `_SMS.SUCCESS` contiene un atributo llamado `attributes.number_of_message_parts`. Este atributo especifica el número de partes de mensaje que contiene el mensaje.

> **Important**
>
> Cuando envía un mensaje que contiene más de una parte de mensaje, se le cobrará el número de partes de mensaje incluidas en él.

Conjunto de caracteres GSM 03.38

En la tabla siguiente, se muestran todos los caracteres del conjunto de caracteres GSM 03.38. Si envía un mensaje que solo incluye los caracteres que se muestran en la tabla, el mensaje puede contener hasta 160 caracteres.

Caracteres estándar GSM 03.38												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	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						

El conjunto de caracteres GSM 03.38 incluye varios símbolos además de los que se muestran en la tabla anterior. Sin embargo, cada uno de estos caracteres se cuenta como dos caracteres, ya que también incluye un carácter de escape invisible:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Por último, el conjunto de caracteres GSM 03.38 también incluye los siguientes caracteres no imprimibles:

- Un carácter de espacio.
- Un control de salto de línea, que indica el final de una línea de texto y el principio de otra.
- Un control de retorno de carro, que cambia al principio de una línea de texto (normalmente después de un carácter de salto de línea).
- Un control de escape, que se añade automáticamente a los caracteres de la lista anterior.

Mensajes de ejemplo

Esta sección contiene varios mensajes SMS de ejemplo. Para cada ejemplo, esta sección muestra el número total de caracteres, así como el número de partes del mensaje.

Ejemplo 1: mensaje largo que solo contiene caracteres en el alfabeto GSM 03.38

El siguiente mensaje solo contiene caracteres que están en el 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.
```

El mensaje anterior contiene 180 caracteres, por lo que debe dividirse en varias partes de mensaje. Cuando un mensaje se divide en varias partes de mensaje, cada una puede contener 153 caracteres GSM 03.38. Como resultado, este mensaje se envía como dos partes de mensaje.

Ejemplo 2: mensaje que contiene caracteres de varios bytes

El siguiente mensaje contiene varios caracteres chinos, que no están incluidos en el alfabeto GSM 03.38.

```
##################################################·####1994#7#########
```

El mensaje anterior contiene 71 caracteres. Sin embargo, debido a que casi todos los caracteres del mensaje no están incluidos en el alfabeto GSM 03.38, se envía como dos partes de mensaje. Cada una de estas partes de mensaje puede contener un máximo de 67 caracteres.

Ejemplo 3: mensaje que contiene un único carácter que no es GSM

El siguiente mensaje contiene un único carácter que no forma parte del alfabeto GSM 03.38. En este ejemplo, el carácter es una comilla simple de cierre (’), que es un carácter diferente al apóstrofo

normal ('). Las aplicaciones de procesamiento de textos como Microsoft Word suelen reemplazar automáticamente los apóstrofos por comillas simples de cierre. Si redacta sus mensajes SMS en Microsoft Word y los pega en Amazon SNS, debe quitar estos caracteres especiales y reemplazarlos por apóstrofos.

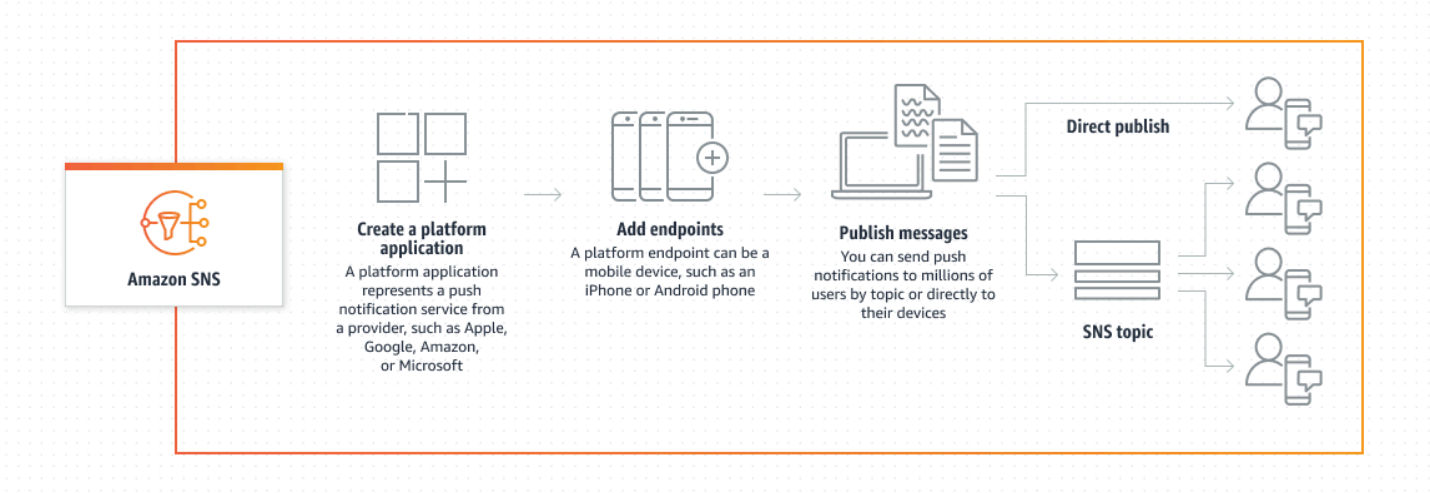
```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

El mensaje anterior contiene 130 caracteres. Sin embargo, debido a que contiene el carácter de comilla simple de cierre, que no forma parte del alfabeto GSM 03.38, se envía como dos partes de mensaje.

Si reemplaza el carácter de comilla simple de cierre en este mensaje por un apóstrofo (que forma parte del alfabeto GSM 03.38), el mensaje se envía como una sola parte de mensaje.

Envío de notificaciones push para móvil con Amazon SNS

Puede usar Amazon SNS para enviar mensajes de notificaciones push directamente a aplicaciones instaladas en dispositivos móviles. Los mensajes de notificaciones push enviados a un punto de conexión móvil pueden mostrarse en la aplicación móvil como mensajes de alerta, actualizaciones de insignias o incluso alertas de sonido.



Temas

- Cómo funcionan las notificaciones de usuario de Amazon SNS
- Configuración de notificaciones push con Amazon SNS
- Configuración de una aplicación móvil en Amazon SNS

- Uso de Amazon SNS para notificaciones push para móvil
- Atributos de aplicaciones móviles de Amazon SNS
- Notificaciones de eventos de aplicación de Amazon SNS para aplicaciones móviles
- Acciones de la API de inserción móvil
- Errores comunes de la API de notificaciones push para móvil de Amazon SNS
- Uso del atributo de mensaje de “time-to-live” de Amazon SNS para las notificaciones push para móvil
- Regiones compatibles con aplicaciones móviles de Amazon SNS
- Prácticas recomendadas para administrar notificaciones push para móvil en Amazon SNS

Cómo funcionan las notificaciones de usuario de Amazon SNS

Los mensajes de notificaciones de inserción se envían a dispositivos móviles y a escritorios utilizando uno de los siguientes servicios de notificaciones de inserción compatibles:

- Amazon Device Messaging (ADM)
- Servicio de notificaciones push de Apple (APNs) para iOS y Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Servicio de notificaciones push de Microsoft para Windows Phone (MPNS)
- Servicios de notificación push de Windows (WNS)

Los servicios de notificaciones push, como APNs FCM, mantienen una conexión con cada aplicación y dispositivo móvil asociado registrados para usar su servicio. Cuando una aplicación y un dispositivo móvil se registran, el servicio de notificaciones push devuelve un token de dispositivo. Amazon SNS utiliza el token de dispositivo para crear un punto de enlace móvil al que puede enviar mensajes de notificaciones push directos. Para que Amazon SNS pueda comunicarse con los diferentes servicios de notificaciones push, debe enviar las credenciales de su servicio de notificaciones push a Amazon SNS para que las utilice en su nombre. Para obtener más información, consulte Configuración de notificaciones push con Amazon SNS.

Además de enviar mensajes de notificaciones push directos, también puede utilizar Amazon SNS para enviar mensajes a puntos de enlace móviles suscritos a un tema. El concepto es el mismo que

para la suscripción a un tema de otros tipos de puntos de enlace, como Amazon SQS, HTTP/S, correo electrónico y SMS, tal y como se describe en ¿Qué es Amazon SNS?. La diferencia radica en que Amazon SNS se comunica mediante servicios de notificaciones push para que los puntos de enlace móviles suscritos reciban los mensajes de notificaciones push enviados al tema.

Configuración de notificaciones push con Amazon SNS

1. Obtenga las credenciales y el token de dispositivo para las plataformas móviles que desea admitir.
2. Utilice las credenciales para crear un objeto de aplicación de plataforma (`PlatformApplicationArn`) mediante Amazon SNS. Para obtener más información, consulte Creación de una aplicación de plataforma de Amazon SNS.
3. Utilice las credenciales obtenidas para solicitar un token de dispositivo para su aplicación móvil y el dispositivo del servicio de notificaciones push. El token que reciba representa su aplicación móvil y el dispositivo.
4. Utilice el token de dispositivo y `PlatformApplicationArn` para crear un objeto de punto de enlace de plataforma (`EndpointArn`) mediante Amazon SNS. Para obtener más información, consulte Configuración de un punto de conexión de plataforma de Amazon SNS para notificaciones móviles.
5. Utilice el `EndpointArn` para publicar un mensaje en una aplicación de un dispositivo móvil. Para obtener más información, consulte Mensajería directa para dispositivos móviles de Amazon SNS y la API Publicar en la Referencia de la API de Amazon Simple Notification Service.

Configuración de una aplicación móvil en Amazon SNS

En este tema se describe cómo configurar aplicaciones móviles AWS Management Console utilizando la información descrita enRequisitos previos para las notificaciones de usuario de Amazon SNS.

Requisitos previos para las notificaciones de usuario de Amazon SNS

Para empezar a utilizar las notificaciones push para móvil de Amazon SNS, necesita lo siguiente:

- Un conjunto de credenciales para conectarse a uno de los servicios de notificaciones push compatibles: ADM, Baidu APNs, FCM, MPNS o WNS.
- Un token de dispositivo o ID de registro para la aplicación y el dispositivo móviles.

- Amazon SNS configurado para enviar mensajes de notificaciones de inserción a los puntos de enlace móviles.
- Una aplicación móvil que esté registrada y configurada para utilizar uno de los servicios de notificaciones de inserción compatibles.

Para registrar su aplicación a un servicio de notificaciones push, tiene que seguir varios pasos. Amazon SNS necesita parte de la información que proporciona al servicio de notificaciones push para poder enviar mensajes de notificaciones push al punto de enlace móvil. En general, necesita las credenciales necesarias para establecer una conexión con el servicio de notificaciones de inserción, un token de dispositivo o ID de registro (que represente el dispositivo y la aplicación móviles) que haya recibido del servicio de notificaciones de inserción y la aplicación móvil registrada en el servicio de notificaciones de inserción.

La forma exacta que las credenciales adoptan varía según la plataforma móvil, pero, en todos los casos, estas credenciales se deben enviar mientras se establece conexión con la plataforma. Se genera un conjunto de credenciales para cada aplicación móvil, que debe utilizarse para enviar un mensaje a cualquier instancia de dicha aplicación.

Los nombres específicos variarán según el servicio de notificaciones de inserción que se utilice. Por ejemplo, si se utiliza APNs como servicio de notificaciones push, se necesita un token de dispositivo. O bien, cuando utilice FCM, el token de dispositivo equivalente se denomina ID de registro. El token de dispositivo o el ID de registro es una cadena que el sistema operativo del dispositivo móvil envía a la aplicación. Sirve para identificar de forma exclusiva una instancia de una aplicación móvil que se ejecuta en un determinado dispositivo móvil y puede considerarse un identificador único de este par concreto de aplicación y dispositivo.

Amazon SNS almacena las credenciales (además de otras configuraciones) como recurso de aplicación de una plataforma. Los tokens de dispositivo (de nuevo con algunos parámetros adicionales) se representan como objetos denominados puntos de conexión de la plataforma. Cada punto de enlace de la plataforma pertenece a una aplicación de plataforma específica y es posible comunicarse con cada uno de ellos usando las credenciales que se almacenan en su correspondiente aplicación de plataforma.

En las secciones siguientes se incluyen los requisitos previos de cada uno de los servicios de notificaciones push compatibles. Una vez que hayas obtenido la información necesaria, puedes enviar un mensaje de notificación push mediante la AWS Management Console función push móvil de Amazon SNS. APIs Para obtener más información, consulte Configuración de notificaciones push con Amazon SNS.

Creación de una aplicación de plataforma de Amazon SNS

Para enviar notificaciones desde Amazon SNS a puntos de conexión móviles, ya sea directamente o mediante suscripciones a un tema, primero debe crear una aplicación de plataforma. Tras registrar la aplicación en ella AWS, debe crear un punto de conexión tanto para la aplicación como para el dispositivo móvil. Este punto de conexión permite a Amazon SNS enviar mensajes al dispositivo.

Para crear una aplicación de plataforma, siga estos pasos:

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, selecciona Notificaciones push.
3. En la sección Platform applications (Aplicaciones de plataforma), elija Create platform application (Crear aplicación de plataforma).
4. Elija su Región de AWS. Para obtener una lista de las regiones de AWS en las que puede crear aplicaciones móviles, consulte Regiones compatibles con aplicaciones móviles de Amazon SNS.
5. Introduce los siguientes detalles de la aplicación:
   - Nombre de la aplicación: proporcione un nombre para la aplicación de la plataforma. El nombre debe tener entre 1 y 256 caracteres y puede contener letras mayúsculas y minúsculas, números, guiones bajos, guiones y puntos.
   - Plataforma de notificaciones push: selecciona el servicio de notificaciones adecuado en el que esté registrada la aplicación (por ejemplo, Apple Push Notification Service () o Firebase Cloud Messaging (APNsFCM)).
6. En función de la plataforma que hayas seleccionado, tendrás que proporcionar credenciales específicas:
   - Para APNs(servicio de notificaciones push de Apple): elige entre la autenticación basada en fichas o la basada en certificados.
     - Para la autenticación basada en fichas, carga un archivo.p8 (generado mediante Keychain Access).
     - Para la autenticación basada en certificados, carga un archivo.p12 (también exportado desde Keychain Access).
   - Para FCM (Firebase Cloud Messaging): ingresa la clave del servidor desde Firebase Console.
   - Para otras plataformas (como ADM o GCM): introduce las claves o credenciales de API correspondientes.

7. Tras introducir los detalles necesarios, selecciona Crear aplicación de plataforma. Esta acción registra la aplicación en Amazon SNS y crea el objeto de aplicación de plataforma correspondiente.
8. Tras la creación, Amazon SNS genera y devuelve un (nombre de recurso de `PlatformApplicationArn`Amazon). Este ARN identifica de forma exclusiva la aplicación de su plataforma y se utiliza al crear puntos finales para dispositivos móviles.

Configuración de un punto de conexión de plataforma de Amazon SNS para notificaciones móviles

Cuando una aplicación y un dispositivo móvil se registran en un servicio de notificaciones push (como APNs Firebase Cloud Messaging), el servicio de notificaciones push devuelve un token de dispositivo. Amazon SNS utiliza este token de dispositivo para crear un punto final de plataforma, que actúa como destino para enviar mensajes de notificación push directos a la aplicación del dispositivo. El punto final de la plataforma actúa como puente y enruta los mensajes enviados por Amazon SNS al servicio de notificaciones push para su entrega al dispositivo móvil correspondiente. Para obtener más información, consulte Requisitos previos para las notificaciones de usuario de Amazon SNS y Configuración de notificaciones push con Amazon SNS.

Descripción de los tokens de los dispositivos y los puntos finales de la plataforma

Un token de dispositivo identifica de forma exclusiva un dispositivo móvil registrado en un servicio de notificaciones push (por ejemplo APNs, Firebase Cloud Messaging). Cuando una aplicación se registra en el servicio de notificaciones push, genera un token de dispositivo específico para esa aplicación y dispositivo. Amazon SNS utiliza este token de dispositivo para crear un punto final de plataforma dentro de la aplicación de plataforma correspondiente.

El punto final de la plataforma permite a Amazon SNS enviar mensajes de notificación push al dispositivo a través del servicio de notificaciones push, manteniendo la conexión entre la aplicación y el dispositivo del usuario.

Creación de un punto de enlace de plataforma

Para insertar notificaciones push en una aplicación con Amazon SNS, primero debe registrarse el token de dispositivo de la aplicación en Amazon SNS llamando a la acción de creación del punto de enlace de plataforma. Esta acción toma el nombre de recurso de Amazon (ARN) de la aplicación de plataforma y el token de dispositivo como parámetros y devuelve el ARN del punto de enlace de plataforma creado.

La acción `CreatePlatformEndpoint` hace lo siguiente:

- Si el punto de conexión de plataforma ya existe, no lo vuelve a crear. Devuelva al intermediario el ARN del punto de enlace de plataforma existente.
- Si ya existe el punto de conexión de plataforma con el mismo token de dispositivo pero diferentes opciones, no lo vuelve a crear. Envíe una excepción al intermediario.
- Si el punto de conexión de plataforma no existe, lo crea. Devuelva al intermediario el ARN del punto de enlace de plataforma que acaba de crear.

No debe llamar a la acción de creación de un punto de enlace de plataforma inmediatamente cada vez que se inicie una aplicación; este enfoque no siempre proporciona un punto de enlace que funcione. Esto puede ocurrir, por ejemplo, cuando una aplicación se desinstala y se vuelve a instalar en el mismo dispositivo, y su punto de enlace que ya existe, pero está deshabilitado. Un proceso de registro correcto debe realizar las operaciones siguientes:

1. Asegurarse de que exista un punto de enlace de plataforma para esta combinación de aplicación y dispositivo.
2. Asegurarse de que el token de dispositivo del punto de enlace de plataforma es el último token de dispositivo válido.
3. Asegurarse de que el punto de enlace de plataforma esté habilitado y listo para ser utilizado.

Pseudocódigo

El siguiente pseudocódigo describe una práctica recomendada para crear un punto de enlace de plataforma que funcione, sea actual y esté habilitado en una amplia variedad de condiciones de partida. Este enfoque funciona tanto si se trata de la primera vez que la aplicación se registra o no, como si el punto de enlace de plataforma de esta aplicación ya existe, o si el punto de enlace de plataforma está habilitado, tiene el token de dispositivo correcto, etc. Es seguro llamarlo varias veces seguidas, ya que no creará puntos de enlace de plataforma duplicados ni cambiará un punto de enlace de plataforma si ya está actualizado y activado.

```
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
```

Este enfoque se puede utilizar siempre que la aplicación quiera registrarse o volver a registrarse. También se puede utilizar para notificar a Amazon SNS un cambio en el token del dispositivo. En este caso, solo tiene que llamar a la acción con el valor de token del último dispositivo. Tenga en cuenta los elementos siguientes de este enfoque:

- Hay dos casos en los que puede llamar a la acción de crear un punto de enlace de plataforma. Puede llamarse justo al principio, cuando la aplicación no conoce su propio ARN de punto de enlace de plataforma, como es el caso durante un primer registro. También se puede llamar si la llamada inicial a la acción `GetEndpointAttributes` genera un error con una excepción “no encontrado”, como ocurriría si la aplicación conoce su ARN de punto de conexión, pero este se ha eliminado.
- Se llama a la acción `GetEndpointAttributes` para verificar el estado del punto de conexión de plataforma, aunque dicho punto de conexión se acabe de crear. Esto ocurre cuando el punto de enlace de plataforma ya existe, pero está deshabilitado. En este caso, la acción de creación del punto de enlace de plataforma se realiza correctamente, pero no habilita el punto de enlace de plataforma, por lo que debe comprobar el estado del punto de enlace de plataforma antes de devolver el resultado correcto.

AWS Ejemplo de SDK

En el siguiente código se muestra cómo implementar el pseudocódigo anterior mediante los clientes de Amazon SNS proporcionados por. AWS SDKs

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

CLI

AWS CLI

Creación de un punto de conexión de aplicación de plataforma

En el siguiente ejemplo de `create-platform-endpoint`, se crea un punto de conexión para la aplicación de plataforma especificada mediante el token especificado.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Salida:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte Acciones de la API de inserción móvil.

Solución de problemas

Llamar repetidamente a la acción de creación de un punto de enlace de plataforma con un token de dispositivo obsoleto

En especial para los puntos de conexión de FCM, puede que piense que es mejor almacenar el primer token de dispositivo generado por la aplicación y llamar a la acción de creación de punto de conexión de plataforma con dicho token de dispositivo cada vez que se inicia la aplicación. Esto puede parecer correcto, dado que la aplicación no tiene que administrar el estado del token de dispositivo y Amazon SNS actualizará automáticamente el token de dispositivo a su valor más reciente. Sin embargo, esta solución presenta una serie de problemas graves:

- Amazon SNS depende de los comentarios de FCM para actualizar los tokens de dispositivo vencidos en tokens de dispositivo nuevos. FCM retiene información en tokens de dispositivo antiguos durante un tiempo, aunque no de forma indefinida. Cuando FCM se olvide de la conexión entre el token de dispositivo antiguo y el nuevo, Amazon SNS ya no podrá actualizar el token de dispositivo almacenado en el punto de enlace de plataforma a su valor correcto; en su lugar, desactivará el punto de enlace de plataforma.
- La aplicación de plataforma contendrá varios puntos de enlace de plataforma correspondientes al mismo token de dispositivo.
- Amazon SNS impone una cuota a la cantidad de puntos de enlace de plataforma que se pueden crear empezando por el mismo token de dispositivo. Al final, la creación de los nuevos puntos de enlace generará un error con la excepción de parámetro no válido y el siguiente mensaje de error: "This endpoint is already registered with a different token".

Para obtener más información acerca de la administración de puntos de conexión de FCM, consulte Administración de los puntos de conexión de Firebase Cloud Messaging en Amazon SNS.

Reactivación de un punto de enlace de plataforma asociado a un token de dispositivo no válido

Cuando una plataforma móvil (como APNs FCM) informa a Amazon SNS de que el token de dispositivo utilizado en la solicitud de publicación no es válido, Amazon SNS deshabilita el punto de enlace de la plataforma asociado a ese token de dispositivo. A continuación, Amazon SNS rechaza las publicaciones posteriores que se efectúen en ese token de dispositivo. Aunque le parezca que es mejor volver a activar el punto de enlace de plataforma y seguir publicando, en la mayoría de los casos esta solución no funciona: los mensajes que se publican no se entregan y el punto de enlace de plataforma se vuelve a desactivar poco después.

Esto se debe a que el token de dispositivo asociado al punto de enlace de plataforma en realidad no es válido. Las entregas que se le hacen no pueden realizarse correctamente, puesto que el token ya no corresponde a ninguna aplicación instalada. La próxima vez que se publique en ella, la plataforma móvil volverá a informar a Amazon SNS que el token de dispositivo no es válido y Amazon SNS volverá a desactivar el punto de enlace de plataforma.

Para volver a habilitar un punto de enlace de plataforma desactivado, debe asociarlo a un token de dispositivo válido (con una llamada de acción de definición de los atributos del punto de enlace) y después habilitarlo. Solo entonces las entregas a dicho punto de enlace de plataforma se realizarán correctamente. La única vez en que volver a habilitar un punto de enlace de plataforma sin actualizar su token de dispositivo funcione será cuando un token de dispositivo que no era válido y estaba

asociado a dicho punto de enlace vuelva a ser válido. Esto puede ocurrir, por ejemplo, cuando se desinstala una aplicación y se vuelve a instalar en el mismo dispositivo móvil y recibe el mismo token de dispositivo. El enfoque que acabamos de presentar realiza esta operación asegurándose de volver a habilitar un punto de enlace de plataforma solo después de comprobar que el token de dispositivo que tiene asociado es el más actual disponible.

Integración de tokens de dispositivo con Amazon SNS para las notificaciones móviles

La primera vez que registras una aplicación y un dispositivo móvil en un servicio de notificaciones, como el Servicio de Notificaciones Push de Apple (APNs) y Firebase Cloud Messaging (FCM), el servicio devuelve IDs los identificadores del dispositivo o el registro. Estos tokens/ IDs se añaden a Amazon SNS para crear un punto de conexión para la aplicación y el dispositivo mediante la API. `PlatformApplicationArn` Una vez creado el punto de conexión, `EndpointArn`se devuelve un, que Amazon SNS utiliza para dirigir las notificaciones a la aplicación o dispositivo correctos.

Puede añadir los identificadores de dispositivo o el registro IDs a Amazon SNS de las siguientes maneras:

- Añada manualmente un único token a través del AWS Management Console
- Cargando varios tokens utilizando la API `CreatePlatformEndpoint`.
- Registre los tokens para los dispositivos del futuro

Para añadir manualmente un token de dispositivo o un identificador de registro

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, selecciona Notificaciones push.
3. En la sección Aplicaciones de plataforma, seleccione la aplicación y, a continuación, elija Editar. Si aún no ha creado una aplicación de plataforma, siga la Creación de una aplicación de plataforma de Amazon SNS guía para hacerlo ahora.
4. Seleccione Crear punto de conexión.
5. En el cuadro Token de punto final, introduce el token o el identificador de registro, en función del servicio de notificaciones que utilices (por ejemplo, el identificador de registro de FCM).
6. (Opcional) Introduce datos adicionales en el campo Datos de usuario. Estos datos deben estar codificados en UTF-8 y tener menos de 2 KB.
7. Seleccione Crear punto de conexión.

Una vez creado el punto de conexión, puede enviar mensajes directamente al dispositivo móvil o a los dispositivos móviles suscritos a un tema de Amazon SNS.

Para cargar varios tokens mediante la API **`CreatePlatformEndpoint`**

En los siguientes pasos se muestra cómo utilizar la aplicación Java (`bulkupload`paquete) de ejemplo proporcionada por AWS para cargar varios tokens (tokens de dispositivo o registro IDs) a Amazon SNS. Puede utilizar esta aplicación de muestra como ayuda para comenzar a cargar sus tokens.

> **Note**
>
> Los siguientes pasos utilizan el IDE de Eclipse Java. En los pasos se da por sentado que ha instalado AWS SDK para Java y dispone de las credenciales AWS de seguridad correspondientes. Cuenta de AWS Para obtener más información, consulte AWS SDK para Java. Para obtener más información sobre las credenciales, consulte las credenciales AWS de seguridad en la Guía del usuario de IAM.

1. Descargue y descomprima el archivo snsmobilepush.zip.
2. Cree un nuevo proyecto de Java en Eclipse e importe la `SNSSamples` carpeta al proyecto.
3. Descarga la biblioteca OpenCSV y agrégala a la ruta de compilación.
4. En el `BulkUpload.properties` archivo, especifique lo siguiente:

   - Su `ApplicationArn` (ARN de la aplicación de plataforma).
   - La ruta absoluta al archivo CSV que contiene los tokens.
   - Registra los nombres de los archivos de los tokens correctos y fallidos. Por ejemplo, `goodTokens.csv` y `badTokens.csv`.
   - (Opcional) Una configuración para el delimitador, el carácter de comilla y el número de hilos que se van a utilizar.

   El `BulkUpload.properties` finalizado ha de tener un aspecto similar al siguiente:

```
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. Ejecute la aplicación BatchCreatePlatformEndpointSample.java para cargar los tokens en Amazon SNS. Los tokens que se hayan cargado correctamente se registrarángoodTokens.csv, mientras que los que tengan un formato incorrecto se registrarán. badTokens.csv

Para registrar los tokens de los dispositivos para futuras instalaciones de aplicaciones

Tienes dos opciones para este proceso:

Utilice el servicio Amazon Cognito

Su aplicación móvil puede usar credenciales de seguridad temporales para crear puntos de conexión. Se recomienda Amazon Cognito para generar credenciales temporales. Para obtener más información, consulte la Guía para desarrolladores de Amazon Cognito

Para realizar un seguimiento de los registros de aplicaciones, utilice los eventos de Amazon SNS para recibir notificaciones cuando se cree un nuevo ARNs punto final.

Como alternativa, puede utilizar la `ListEndpointByPlatformApplication`API para recuperar la lista de puntos de enlace registrados.

Uso de un servidor proxy

Si la infraestructura de tu aplicación ya admite el registro de dispositivos en el momento de la instalación, puedes usar tu servidor como proxy. Reenviará los tokens del dispositivo a Amazon SNS a través de la `CreatePlatformEndpoint`API.

El ARN del punto final creado por Amazon SNS se devolverá y su servidor podrá almacenarlo para la publicación de mensajes en el futuro.

Métodos de autenticación de notificaciones push de Amazon SNS Apple

Puede autorizar a Amazon SNS a enviar notificaciones push a su aplicación de iOS o macOS proporcionando información que le identifique como desarrollador de esa aplicación. Para autenticarse, proporcione una clave o un certificado al crear una aplicación de plataforma; ambas cosas puede obtenerlas en su cuenta de Apple Developer.

Clave de firma de token

Clave de firma privada que Amazon SNS utiliza para firmar los tokens de autenticación del Servicio de Notificaciones Push de Apple (APNs).

Si proporciona una clave de firma, Amazon SNS utiliza un token para autenticarse en cada notificación push que envíe. APNs Con su clave de firma, puede enviar notificaciones automáticas a entornos de APNs producción y entornos aislados.

La clave de firma no caduca, y se puede utilizar la misma clave de firma para varias aplicaciones. Para obtener más información, consulta Cómo comunicarse APNs mediante el uso de tokens de autenticación en la sección de ayuda de la cuenta de desarrollador del sitio web de Apple.

Certificate

Un certificado TLS que Amazon SNS utiliza para autenticarse cuando envías APNs notificaciones push. Puede obtener este certificado en su cuenta de Apple Developer.

Los certificados caducan al cabo de un año. Cuando eso sucede, se debe crear un nuevo certificado y proporcionárselo a Amazon SNS. Para obtener más información, consulte Establecer una conexión basada en certificados en el APNs sitio web para desarrolladores de Apple.

Para administrar la APNs configuración mediante la consola de AWS administración

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, selecciona Notificaciones push.
3. En la sección Aplicaciones de plataforma, seleccione la aplicación cuya APNs configuración desee editar y, a continuación, elija Editar. Si aún no ha creado una aplicación de plataforma, siga la Creación de una aplicación de plataforma de Amazon SNS guía para hacerlo ahora.
4. Seleccione Editar para modificar la configuración de la aplicación de su plataforma.
5. En la sección Tipo de autenticación, elija una de las siguientes opciones:
   - Autenticación basada en fichas (recomendada para integraciones modernas APNs)
   - Autenticación basada en certificados (método anterior)
6. Configure sus credenciales en función del tipo de autenticación:
   - Para la autenticación basada en tokens:
     - Carga el archivo.p8, que es la clave de firma del token de autenticación que descargaste de tu cuenta de desarrollador de Apple.

- Introduce el identificador de clave de firma que encontrarás en tu cuenta de desarrollador de Apple. Ve a Certificados, perfiles IDs y claves y selecciona la clave que quieras usar.
- Proporciona el identificador de equipo de tu cuenta de desarrollador de Apple. Puedes encontrarlo en la página de membresía.
- Introduce el identificador de paquete asignado a tu aplicación. Puedes encontrarlo en Certificados IDs y perfiles, Aplicación IDs.

- Para la autenticación basada en certificados:
  - Cargue el archivo.p12 de su certificado TLS. Este archivo se puede exportar desde Keychain Access en macOS después de descargar el certificado de tu cuenta de desarrollador de Apple.
  - Si ha asignado una contraseña a su certificado .p12, introdúzcala aquí.

7. Tras introducir las credenciales necesarias, seleccione Guardar cambios para actualizar la configuración.

Configuración de autenticación en la integración de Amazon SNS con Firebase Cloud Messaging

En este tema se describe cómo obtener las credenciales de la API de FCM (HTTP v1) necesarias de Google para utilizarlas con la AWS API, AWS CLI y la. AWS Management Console

> **Important**
>
> 26 de marzo de 2024: Amazon SNS admite la API de HTTP v1 de FCM para dispositivos Apple y destinos de Webpush. Le recomendamos que migre sus aplicaciones de notificaciones push para el móvil existentes a la última API de HTTP v1 de FCM el 1 de junio de 2024 o antes para evitar que se interrumpan las aplicaciones.
> 18 de enero de 2024: Amazon SNS introdujo la compatibilidad con la API de HTTP v1 de FCM para la entrega de notificaciones push para móvil a dispositivos Android.
> 20 de junio de 2023: Google dejó de utilizar su API de HTTP antigua de Firebase Cloud Messaging (FCM). Amazon SNS ahora admite la entrega a todos los tipos de dispositivos mediante la API de HTTP v1 de FCM. Le recomendamos que migre sus aplicaciones de notificaciones push para el móvil existentes a la última API de HTTP v1 de FCM el 1 de junio de 2024 o antes para evitar que se interrumpa el servicio.

Puede autorizar a Amazon SNS a enviar notificaciones push a sus aplicaciones proporcionando información que le identifique como desarrollador de esa aplicación. Para autenticarse, proporcione una clave de API o un token al crear una aplicación de plataforma. Puede obtener la siguiente información desde la consola de aplicaciones de Firebase:

Clave de API

La clave de API es una credencial que se utiliza al llamar a la API heredada de Firebase. Google eliminará el legado APIs de FCM el 20 de junio de 2024. Si utiliza actualmente una clave de API como credencial de plataforma, puede actualizar la credencial de plataforma seleccionando Token como opción y subiendo el archivo JSON asociado a su aplicación de Firebase.

Token

Al llamar a la API de HTTP v1, se utiliza un token de acceso de corta duración. Esta es la API sugerida de Firebase para enviar notificaciones push. Para generar los tokens de acceso, Firebase proporciona a los desarrolladores un conjunto de credenciales en forma de archivo de clave privada (también denominado archivo service.json).

Requisito previo

Debe obtener sus credenciales service.json de FCM para poder empezar a administrar la configuración de FCM en Amazon SNS. Para obtener tus credenciales de service.json, consulta Migrar de una versión antigua de FCM APIs a HTTP v1 en la documentación de Google Firebase.

Administración de la configuración de FCM mediante la CLI

Puedes crear notificaciones push de FCM mediante la AWS API. La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Puntos de conexión y cuotas de Amazon Simple Notification Service en la Guía de Referencia general de AWS .

Para crear una notificación push de FCM junto con un tema AWS (API) de Amazon SNS

Cuando se utilizan credenciales de clave, `PlatformCredential` es `API key`. Cuando se utilizan credenciales de token, `PlatformCredential` es un archivo de clave privada con formato JSON:

- `CreatePlatformApplication`

Para recuperar un tipo de credencial de FCM para un tema (API) de Amazon SNS existente AWS

Recupera el tipo de credencial `"AuthenticationMethod": "Token"` o `"AuthenticationMethod": "Key"`:

- GetPlatformApplicationAttributes

Para establecer un atributo de FCM para un tema existente de Amazon SNS (API de AWS)

Establece el atributo de FCM:

- SetPlatformApplicationAttributes

Administración de la configuración de FCM mediante la consola

Puedes crear notificaciones push de FCM mediante la AWS Command Line Interface (CLI). La cantidad y el tamaño de los recursos de Amazon SNS en una AWS cuenta son limitados. Para obtener más información, consulte Amazon Simple Notification Service endpoints and quotas (Limitación y cuotas de Amazon Simple Notification Service).

Para crear una notificación push de FCM junto con un tema de Amazon SNS (AWS CLI)

Cuando se utilizan credenciales de clave, `PlatformCredential` es `API key`. Cuando se utilizan credenciales de token, `PlatformCredential` es un archivo de clave privada con formato JSON. Al utilizar la AWS CLI, el archivo debe estar en formato de cadena y se deben ignorar los caracteres especiales. Para formatear el archivo correctamente, Amazon SNS recomienda utilizar el siguiente comando: `SERVICE_JSON=`jq @json <<< cat service.json``

- create-platform-application

Para recuperar un tipo de credencial de FCM para un tema existente de Amazon SNS (AWS CLI)

Recupera el tipo de credencial `"AuthenticationMethod": "Token"` o `"AuthenticationMethod": "Key"`:

- get-platform-application-attributes

Para establecer un atributo de FCM para un tema existente de Amazon SNS (AWS CLI)

Establece el atributo de FCM:

- set-platform-application-attributes

Administración de la configuración de FCM (consola)

Sigue los siguientes pasos para introducir y gestionar tus credenciales de Firebase Cloud Messaging (FCM) en Amazon SNS.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, selecciona Notificaciones push.
3. En la sección Aplicaciones de plataforma, selecciona la aplicación de plataforma de FCM cuyas credenciales quieres editar y, a continuación, selecciona Editar.
4. En la sección Credenciales de Firebase Cloud Messaging, elige una de las siguientes opciones:
   - Autenticación basada en fichas (método recomendado): carga el archivo de clave privada (JSON) que descargaste de Firebase Console. Este archivo contiene las credenciales necesarias para generar tokens de acceso de corta duración para las notificaciones de FCM. Para obtener este archivo:
     1. Ve a la consola de aplicaciones de Firebase.
     2. En la configuración del proyecto, selecciona Cloud Messaging.
     3. Descarga el archivo JSON de clave privada (para usarlo en el método de autenticación basado en tokens).
   - Autenticación con clave de API: si prefieres utilizar el método de autenticación de clave de API anterior, introduce la clave de API de Google en el campo proporcionado. Para obtener este archivo:
     1. Ve a la consola de aplicaciones de Firebase.
     2. En la configuración del proyecto, selecciona Cloud Messaging.
     3. Copie la clave del servidor (clave de API) que se utilizará para enviar las notificaciones.
5. Cuando termine de realizar los cambios, seleccione Save changes (Guardar cambios).

Temas relacionados

- Uso de cargas útiles de Google Firebase Cloud Messaging v1 en Amazon SNS

Administración de los puntos de conexión de Firebase Cloud Messaging en Amazon SNS

Administración y mantenimiento de los tokens de dispositivo

Puede garantizar la capacidad de entrega de las notificaciones push de su aplicación móvil siguiendo estos pasos:

1. Guarde todos los tokens del dispositivo, el punto final ARNs de Amazon SNS correspondiente y las marcas de tiempo en el servidor de aplicaciones.
2. Elimine todos los tokens obsoletos y elimine el punto de enlace de Amazon ARNs SNS correspondiente.

Cuando la aplicación se inicie por primera vez, recibirá un token de dispositivo (también llamado token de registro) para el dispositivo. Este token de dispositivo lo crea el sistema operativo del dispositivo y está vinculado a su aplicación de FCM. Una vez que reciba este token de dispositivo, podrá registrarlo en Amazon SNS como punto de conexión de la plataforma. Le recomendamos que almacene el token de dispositivo, el ARN del punto de conexión de la plataforma de Amazon SNS y la marca de tiempo guardándolos en su servidor de aplicaciones o en otro almacén persistente. Para configurar su aplicación de FCM para recuperar y almacenar tokens de dispositivos, consulte Retrieve and store registration tokens en la documentación de Firebase de Google.

Es importante que mantenga up-to-date los tokens. Los tokens de los dispositivos de sus usuarios pueden cambiar si se produce alguna de las circunstancias siguientes:

1. La aplicación móvil se restaura en un dispositivo nuevo.
2. El usuario desinstala o actualiza la aplicación.
3. El usuario borra los datos de la aplicación.

Cuando el token de su dispositivo cambie, le recomendamos que actualice el punto de conexión de Amazon SNS correspondiente con el nuevo token. Esto permite a Amazon SNS seguir comunicándose con el dispositivo registrado. Puede hacerlo implementando el siguiente pseudocódigo en su aplicación móvil. Describe una práctica recomendada para crear y mantener puntos de conexión de plataforma habilitados. Este enfoque se puede utilizar cada vez que se inician las aplicaciones móviles o como un trabajo programado en segundo plano.

Pseudocódigo

Use el siguiente pseudocódigo de FCM para administrar y mantener los tokens de dispositivo.

```
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 obtener más información sobre los requisitos de actualización de los tokens, consulte Update Tokens on a Regular Basis en la documentación de Firebase de Google.

Detección de tokens no válidos

Cuando se envíe un mensaje a un punto de conexión de FCM v1 con un token de dispositivo no válido, Amazon SNS recibirá una de las siguientes excepciones:

- `UNREGISTERED` (HTTP 404): cuando Amazon SNS reciba esta excepción, usted recibirá un error en la entrega con un `FailureType` de `InvalidPlatformToken` y un `FailureMessage` de Platform token associated with the endpoint is not valid. Amazon SNS deshabilitará el punto de conexión de la plataforma cuando se produzca un error en una entrega con esta excepción.
- `INVALID_ARGUMENT` (HTTP 400): cuando Amazon SNS recibe esta excepción, significa que el token del dispositivo o la carga útil del mensaje no son válidos. Para obtener más información, consulta ErrorCodela documentación de Firebase de Google.

Como `INVALID_ARGUMENT` se puede devolver en cualquiera de estos casos, Amazon SNS devolverá un `FailureType` de `InvalidNotification` y un `FailureMessage` de Notification body is invalid. Cuando reciba este error, compruebe que la carga útil es correcta. Si es correcto, verifica que el token del dispositivo lo sea up-to-date. Amazon SNS deshabilitará el punto de conexión de la plataforma cuando se produzca un error en una entrega con esta excepción.

Otro caso en el que se producirá un evento de error de entrega `InvalidPlatformToken` es cuando el token de dispositivo registrado no pertenezca a la aplicación que intenta enviar ese mensaje. En este caso, Google devolverá un error SENDER_ID_MISMATCH. Amazon SNS deshabilitará el punto de conexión de la plataforma cuando se produzca un error en una entrega con esta excepción.

Todos los códigos de error observados recibidos de la API v1 de FCM están disponibles CloudWatch cuando configuras el registro del estado de entrega de tu aplicación.

Para recibir los eventos de entrega de su aplicación, consulte Eventos de aplicaciones disponibles.

Eliminación de tokens obsoletos

Los tokens se consideran obsoletos una vez que la entrega de mensajes al dispositivo de punto de conexión comienza a fallar. Amazon SNS establece estos tokens obsoletos como puntos de conexión deshabilitados para su aplicación de plataforma. Cuando publique en un punto de conexión deshabilitado, Amazon SNS devolverá un evento `EventDeliveryFailure` con el `FailureType` de `EndpointDisabled` y un `FailureMessage` de Endpoint is disabled. Para recibir los eventos de entrega de su aplicación, consulte Eventos de aplicaciones disponibles.

Cuando reciba este error de Amazon SNS, tendrá que eliminar o actualizar el token obsoleto de su aplicación de plataforma.

Uso de Amazon SNS para notificaciones push para móvil

En esta sección, se describe cómo enviar un mensaje de notificaciones push móviles.

Publicación de un tema

También puede utilizar Amazon SNS para enviar mensajes a puntos de enlace móviles suscritos a un tema. El concepto es el mismo que para la suscripción a un tema de otros tipos de puntos de enlace, como Amazon SQS, HTTP/S, correo electrónico y SMS, tal y como se describe en ¿Qué es Amazon SNS?. La diferencia es que Amazon SNS se comunica a través de servicios de

notificación, tales como Apple Push Notification Service (APNS) y Google Firebase Cloud Messaging (FCM). Mediante los servicios de notificaciones, los puntos de enlace móviles suscritos reciben las notificaciones enviadas al tema.

Mensajería directa para dispositivos móviles de Amazon SNS

Puede enviar mensajes de notificaciones push de Amazon SNS directamente a un punto de enlace que represente una aplicación en un dispositivo móvil.

Para enviar un mensaje directo

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Push notifications (Notificaciones push).
3. En la página de notificaciones push móviles, en la sección Aplicaciones de la plataforma, elige el nombre de la aplicación, por ejemplo***MyApp***.
4. En la ***MyApp*** página, en la sección Puntos finales, elija un punto final y, a continuación, elija Publicar mensaje.
5. En la página Publish message to endpoint (Publicar mensaje en punto de enlace), escriba el mensaje que aparecerá en la aplicación del dispositivo móvil y, a continuación, elija Publish message (Publicar mensaje).

   Amazon SNS envía el mensaje de notificación al servicio de notificaciones de la plataforma que, a su vez, envía el mensaje a la aplicación.

Publicación de notificaciones de Amazon SNS con cargas útiles específicas de la plataforma

Puede usar Amazon SNS AWS Management Console o Amazon APIs para enviar mensajes personalizados con cargas útiles específicas de la plataforma a dispositivos móviles. Para obtener información sobre el uso de Amazon SNS APIs, consulte Acciones de la API de inserción móvil y el `SNSMobilePush.java` archivo en. `snsmobilepush.zip`

Envío de mensajes con formato JSON

Cuando envíe cargas específicas de la plataforma, los datos deben tener un formato de cadenas de pares de clave-valor JSON, con las comillas incluidas entre caracteres de escape.

En los siguientes ejemplos, se muestra un mensaje personalizado para la plataforma de FCM.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

Envío de mensajes específicos de la plataforma

Además de enviar datos personalizados como pares de clave-valor, puede enviar pares de clave-valor específicos de la plataforma.

En el siguiente ejemplo, se muestra la inclusión de los parámetros de `time_to_live` y `collapse_key` después de los pares clave-valor de datos personalizados en el parámetro `data` de FCM.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Para obtener una lista de los pares de clave-valor admitidos en cada uno de los servicios de notificaciones push admitidos en Amazon SNS, consulte lo siguiente:

> **Important**
>
> Amazon SNS ahora admite la API de HTTP v1 de Firebase Cloud Messaging (FCM) para enviar notificaciones push para móvil a dispositivos Android.
> 26 de marzo de 2024: Amazon SNS admite la API de HTTP v1 de FCM para dispositivos Apple y destinos de Webpush. Le recomendamos que migre sus aplicaciones de notificaciones push para el móvil existentes a la última API de HTTP v1 de FCM el 1 de junio de 2024 o antes para evitar que se interrumpan las aplicaciones.

- Referencia clave de carga útil en la documentación APNs
- Protocolo HTTP de Firebase Cloud Messaging en la documentación de FCM
- Enviar un mensaje en la documentación de ADM

Envío de mensajes a una aplicación en varias plataformas

Para enviar un mensaje a una aplicación instalada en dispositivos para varias plataformas, como FCM y APNs, primero debes suscribir los puntos de enlace móviles a un tema de Amazon SNS y, a continuación, publicar el mensaje en el tema.

El siguiente ejemplo muestra un mensaje para enviar a los puntos de enlace móviles suscritos en APNs FCM y 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\"}}"
}
```

Enviar mensajes APNs como alertas o notificaciones en segundo plano

Amazon SNS puede enviar mensajes a APNs as `alert` o `background` notificaciones (para obtener más información, consulte Cómo enviar actualizaciones en segundo plano a su aplicación en la APNs documentación).

- Una `alert` APNs notificación informa al usuario mediante la visualización de un mensaje de alerta, la reproducción de un sonido o la adición de una insignia al icono de la aplicación.
- Una `background` APNs notificación activa o indica a la aplicación que actúe en función del contenido de la notificación, sin informar al usuario.

Especificar valores de encabezado personalizados APNs

Recomendamos especificar valores personalizados para el atributo de mensaje `AWS.SNS.MOBILE.APNS.PUSH_TYPE` reservado mediante la acción de la `Publish` API Amazon SNS AWS SDKs, o la. AWS CLI En el siguiente ejemplo de la CLI `content-available` se establece como 1 y `background` como `apns-push-type` para el tema 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**
>
> Asegúrese de que la estructura JSON sea válida. Añada una coma después de cada par de clave-valor, excepto en el último.

Deducir el encabezado del tipo APNs push a partir de la carga útil

Si no establece el `apns-push-type` APNs encabezado, Amazon SNS establece el encabezado en `alert` o en `background` función de la `content-available` clave del `aps` diccionario de la configuración de carga con formato JSON. APNs

> **Note**
>
> Amazon SNS se puede inferir solo en encabezados `alert` o `background`, aunque el encabezado `apns-push-type` se puede establecer en otros valores.

- `apns-push-type` toma el valor `alert`
  - Si el diccionario `aps` contiene `content-available` defina en `1` y una o varias claves que activan las interacciones del usuario.
  - Si el diccionario `aps` contiene `content-available` defina en `0` o si la clave `content-available` está ausente.
  - Si el valor de la clave `content-available` no es un entero o un booleano.
- `apns-push-type` toma el valor `background`

- Si en el diccionario de `aps` solo se encuentra `content-available` establecida en `1` y no hay otras claves que desencadenen interacciones del usuario.

> **Important**
>
> Si Amazon SNS envía un objeto de configuración sin procesar APNs como notificación solo en segundo plano, debes incluir `content-available` set to `1` en el diccionario. `aps` Aunque puede incluir claves personalizadas, el diccionario de `aps` no debe contener ninguna clave que desencadene las interacciones del usuario (por ejemplo, alertas, insignias o sonidos).

A continuación se muestra un ejemplo de objeto de configuración sin procesar.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

En este ejemplo, Amazon SNS establece el `apns-push-type` APNs encabezado del mensaje en. `background` Cuando Amazon SNS detecta que en el diccionario de `apn` se encuentra la clave de `content-available` definida en `1`, pero no hay ninguna otra clave que pueda desencadenar las interacciones del usuario, establece el encabezado en `background`.

Uso de cargas útiles de Google Firebase Cloud Messaging v1 en Amazon SNS

Amazon SNS admite el uso de la API de HTTP v1 de FCM para enviar notificaciones a destinos de Android, iOS y Webpush. En este tema se proporcionan ejemplos de la estructura de carga útil al publicar notificaciones push para el móvil mediante la CLI o la API de Amazon SNS.

Puede incluir los siguientes tipos de mensajes en su carga útil al enviar una notificación de FCM:

- Mensaje de datos: la aplicación cliente gestiona un mensaje de datos, que contiene pares de clave-valor personalizados. Al crear un mensaje de datos, debe incluir la clave `data` con un objeto JSON como valor y, a continuación, introducir los pares de clave-valor personalizados.
- Mensaje de notificación o mensaje para mostrar: un mensaje de notificación contiene un conjunto predefinido de claves administradas por el SDK de FCM. Estas claves varían en función del tipo de dispositivo en el que se realice la entrega. Para obtener más información sobre las claves de notificación específicas de la plataforma, consulte lo siguiente:

- Claves de notificación de Android
- Claves de notificación de APNS
- Claves de notificación Webpush

Para obtener más información sobre los tipos de mensajes de FCM, consulte Message types en la documentación de Firebase de Google.

Uso de la estructura de carga útil de FCM v1 para enviar mensajes

Si va a crear una aplicación de FCM por primera vez o desea utilizar las características de FCM v1, puede optar por enviar una carga útil con formato de FCM v1. Para ello, debe incluir la clave de nivel superior `fcmV1Message`. Para obtener más información sobre la creación de cargas útiles de FCM v1, consulte Migrar de una versión antigua de FCM APIs a HTTP v1 y Personalizar un mensaje en todas las plataformas en la documentación de Firebase de Google.

Ejemplo de carga útil de FCM v1 enviada a Amazon SNS:

> **Note**
>
> El valor de la clave `GCM` utilizado en el siguiente ejemplo debe codificarse como una cadena al publicar una notificación mediante 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\"
          }
        }
      }
    }
  }"
```

```
}
```

Al enviar una carga útil JSON, asegúrese de incluir el atributo `message-structure` en la solicitud y establecerlo en `json`.

Ejemplo con la 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 obtener más información sobre el envío de cargas útiles con formato FCM v1, consulte lo siguiente en la documentación de Firebase de Google:

- Migre de la versión antigua de FCM a la versión HTTP v1 APIs
- About FCM messages
- Recurso REST: projects.messages

Uso de la estructura de carga útil de FCM v1 para enviar mensajes a la API de FCM v1

Al migrar a FCM v1, no tiene que cambiar la estructura de la carga útil que utilizaba para sus credenciales antiguas. Amazon SNS transforma la carga útil en la nueva estructura de carga útil de FCM v1 y la envía a Google.

Formato de carga útil del mensaje 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\"}}"
}
```

Mensaje enviado a 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"
    }
  }
}
```

Riesgos potenciales

- La correspondencia entre la carga útil antigua y v1 no admite los `headers` ni las claves `fcm_options` del servicio de notificaciones push de Apple (APNS). Si quiere utilizar estos campos, envíe una carga útil de FCM v1.
- En algunos casos, la versión 1 de FCM requiere los encabezados de los mensajes para enviar notificaciones silenciosas a tus APNs dispositivos. Si actualmente envías notificaciones silenciosas a tus APNs dispositivos, no funcionarán con el enfoque tradicional. En su lugar, le recomendamos que utilice la carga útil de FCM v1 para evitar problemas inesperados. Para ver una lista de APNs los encabezados y para qué se utilizan, consulta Cómo comunicarse con ellos APNs en la Guía para desarrolladores de Apple.
- Si utiliza el atributo TTL de Amazon SNS al enviar la notificación, solo se actualizará en el campo `android`. Si quiere establecer el atributo TTL de APNS, utilice la carga útil de FCM v1.

- Se establecerá una correspondencia con las claves `android`, `apns` y `webpush`, que se rellenarán con todas las claves pertinentes proporcionadas. Por ejemplo, si proporciona `title`, que es una clave compartida entre las tres plataformas, la correspondencia con FCM v1 rellenará las tres plataformas con el título proporcionado.
- Algunas claves compartidas entre plataformas esperan tipos de valores diferentes. Por ejemplo, la clave `badge` que se pasa a `apns` espera un valor entero, mientras que la clave `badge` que se pasa a `webpush` espera un valor de cadena. En los casos en los que proporcione la clave `badge`, la correspondencia de FCM v1 solo rellenará la clave para la que proporcionó un valor válido.

Eventos de error de entrega de FCM

En la siguiente tabla se proporciona el tipo de error de Amazon SNS que corresponde a los códigos de error o estado recibidos de Google para las solicitudes de notificación de FCM v1. Todos los códigos de error observados recibidos de la API v1 de FCM están disponibles CloudWatch cuando configuras el registro del estado de entrega de tu aplicación.

Código de error/estado de FCM	Tipo de error de Amazon SNS	Mensaje de error	Causa y mitigación
`UNREGISTERED`	`InvalidPlatformToken`	El token de la plataforma asociado al punto de conexión no es válido.	El token de dispositivo asociado al punto de conexión está obsoleto o no es válido. Amazon SNS ha deshabilitado el punto de conexión. Actualice el punto de conexión de Amazon SNS al token de dispositivo más reciente.
`INVALID_ARGUMENT`	`InvalidNotification`	El cuerpo de la notificación no es válido.	Es posible que el token del dispositivo o la carga útil del mensaje no sean válidos. Compruebe

Código de error/estado de FCM	Tipo de error de Amazon SNS	Mensaje de error	Causa y mitigación
			que la carga útil del mensaje sea válida. Si la carga útil del mensaje es válida, actualice el punto de conexión de Amazon SNS al token de dispositivo más reciente.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	El token de la plataforma asociado al punto de conexión no es válido.	La aplicación de plataforma asociada al token del dispositivo no tiene permiso para enviar al token del dispositivo. Compruebe que está utilizando las credenciales de FCM correctas en su aplicación de plataforma de Amazon SNS.

Código de error/estado de FCM	Tipo de error de Amazon SNS	Mensaje de error	Causa y mitigación
`UNAVAILABLE`	`DependencyUnavailable`	La dependencia no está disponible.	FCM no pudo procesar la solicitud a tiempo. Fallaron todos los reintentos ejecutados por Amazon SNS. Puede almacenar estos mensajes en una cola de mensajes fallidos (DLQ) y redirigirlos más adelante.
`INTERNAL`	`UnexpectedFailure`	Error inesperado; póngase en contacto con Amazon. Frase de error [error interno].	El servidor de FCM ha detectado un error al procesar la solicitud. Fallaron todos los reintentos ejecutados por Amazon SNS. Puede almacenar estos mensajes en una cola de mensajes fallidos (DLQ) y redirigirlos más adelante.
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	Las credenciales de la aplicación de plataforma no son válidas.	No se ha podido enviar un mensaje dirigido a un dispositivo iOS o a un dispositivo Webpush. Compruebe que sus credenciales de desarrollo y producción sean válidas.

Código de error/estado de FCM	Tipo de error de Amazon SNS	Mensaje de error	Causa y mitigación
`QUOTA_EXCEEDED`	`Throttled`	Solicitud restringida por [gcm].	Se ha superado una cuota de tasa de mensajes, una cuota de tasa de mensajes del dispositivo o una cuota de tasa de mensajes del tema. Para obtener información sobre cómo resolver este problema, consulta la ErrorCodedocumentación de Firebase de Google.
`PERMISSION_DENIED`	`InvalidNotification`	El cuerpo de la notificación no es válido.	En el caso de una excepción `PERMISSION_DENIED` , el autor de la llamada (su aplicación de FCM) no tiene permiso para ejecutar la operación especificada en la carga útil. Vaya a la consola de FCM y verifique que sus credenciales tengan habilitadas las acciones de API necesarias.

Atributos de aplicaciones móviles de Amazon SNS

Con Amazon Simple Notification Service (Amazon SNS), se puede registrar el estado de entrega de los mensajes de notificaciones push. Tras configurar los atributos de la aplicación, las entradas de registro se enviarán a CloudWatch Logs para los mensajes enviados desde Amazon SNS a puntos de conexión móviles. El log del estado de entrega de los mensajes aporta información operativa de mejor calidad, como la siguiente:

- Saber si Amazon SNS ha entregado un mensaje de notificación push al servicio de notificaciones push.
- Identificar la respuesta enviada por el servicio de notificaciones push a Amazon SNS.
- Determinar el tiempo de permanencia del mensaje (el tiempo entre la marca temporal de publicación y justo antes de entregarlo a un servicio de notificaciones push)

Para configurar los atributos de la aplicación para el estado de entrega de los mensajes, puede utilizar los AWS Management Console kits de desarrollo de AWS software (SDKs) o la API de consulta.

Configurar los atributos del estado de entrega de los mensajes mediante el AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Móvil y, a continuación, Notificaciones push.
3. En la sección Aplicaciones de la plataforma, elija la aplicación que contenga los puntos finales de los que desee recibir CloudWatch los registros.
4. Elija Application Actions (Acciones de la aplicación) y después Delivery Status (Estado de entrega).
5. En el cuadro de diálogo Delivery Status (Estado de entrega), elija Create IAM Roles (Crear roles de IAM).

   Se lo redirigirá a la consola de IAM.
6. Selecciona Permitir para conceder a Amazon SNS acceso de escritura para que utilice CloudWatch Logs en tu nombre.
7. Ahora, de nuevo en el cuadro de diálogo del estado de la entrega, introduzca un número en el campo Porcentaje de éxito del muestreo (0-100) para indicar el porcentaje de mensajes enviados correctamente de los que desea recibir CloudWatch registros.

> **Note**
>
> Tras configurar los atributos de la aplicación para el estado de entrega de los mensajes, todas las entregas de mensajes fallidas generan CloudWatch registros.

8. Por último, elija Save Configuration (Guardar configuración). Ahora podrá ver y analizar los CloudWatch registros que contienen el estado de entrega de los mensajes. Para obtener más información sobre su uso CloudWatch, consulte la CloudWatchdocumentación.

Ejemplos de registro de estado CloudWatch de entrega de mensajes de Amazon SNS

Después de configurar los atributos de estado de entrega de mensajes para un punto final de la aplicación, se generarán CloudWatch los registros. A continuación, se muestran registros de ejemplo en formato JSON:

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 obtener una lista de códigos de respuesta del servicio de notificaciones push, consulte Códigos de respuesta de la plataforma.

Configurar los atributos de estado de entrega de mensajes con el AWS SDKs

AWS SDKsProporcionan APIs en varios idiomas el uso de los atributos de estado de entrega de los mensajes con Amazon SNS.

El ejemplo de Java siguiente muestra cómo utilizar la API `SetPlatformApplicationAttributes` para configurar atributos de las aplicaciones para el estado de entrega de los mensajes de notificaciones de inserción. Puede utilizar los atributos siguientes para el estado de entrega de los mensajes: `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn` y `SuccessFeedbackSampleRate`. Los `FailureFeedbackRoleArn` atributos `SuccessFeedbackRoleArn` y se utilizan para conceder a Amazon SNS acceso de escritura para usar CloudWatch Logs en su nombre. El atributo `SuccessFeedbackSampleRate` permite especificar el porcentaje de la frecuencia de muestreo (0-100) de los mensajes entregados correctamente. Tras configurar el `FailureFeedbackRoleArn` atributo, todas las entregas de mensajes fallidas generarán 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 obtener más información sobre el SDK para Java, consulte Introducción a AWS SDK para Java.

Códigos de respuesta de la plataforma

A continuación se incluye una lista de enlaces de códigos de respuesta del servicio de notificaciones push:

Servicio de notificaciones de inserción	Códigos de respuesta
Amazon Device Messaging (ADM)	Consulte Formato de respuesta en la documentación de ADM.
Servicio APNs de notificaciones push de Apple ()	Consulte la respuesta HTTP/2 de APNs In Communication with APNs en la Guía de programación de notificaciones locales y remotas.
Firebase Cloud Messaging (FCM)	Consulte Códigos de respuesta de errores de mensajes descendentes en la documentación de Firebase Cloud Messaging.
Servicio de notificaciones push de Microsoft para Windows Phone (MPNS)	Consulte Push Notification Service Response Codes for Windows Phone 8 en la documenta ción de desarrollo de Windows 8.
Servicios de notificación push de Windows (WNS)	Consulte "Response codes" en Push Notificat ion Service Request and Response Headers (Windows Runtime Apps) en la documentación de desarrollo de Windows 8.

Notificaciones de eventos de aplicación de Amazon SNS para aplicaciones móviles

Con Amazon SNS, se proporciona compatibilidad con la activación de notificaciones cuando se producen determinados eventos de aplicaciones. Después, puede ejecutar algunas acciones mediante programación en dicho evento. La aplicación debe incluir soporte para un servicio de notificaciones push, como Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) y Windows Push Notification Services (WNS). Las notificaciones de eventos de la aplicación se configuran mediante la consola de Amazon SNS o la. AWS CLI AWS SDKs

Eventos de aplicaciones disponibles

Las notificaciones de eventos de aplicaciones controlan cuándo se crean, eliminan o actualizan los distintos puntos de conexión de la plataforma, así como los errores de entrega. A continuación, se muestran los nombres de los atributos para los eventos de la aplicación.

Nombre de atributo	Desencadenador de la notificación
`EventEndpointCreated`	Se añade a la aplicación un nuevo punto de conexión de la plataforma.
`EventEndpointDeleted`	Se elimina cualquier punto de conexión de la plataforma asociado a la aplicación.
`EventEndpointUpdated`	Se cambia cualquiera de los atributos de los puntos de conexión de la plataforma asociados a la aplicación.
`EventDeliveryFailure`	Una entrega a cualquiera de los puntos de conexión de la plataforma asociados a la aplicación encuentra un error permanente. **Note** Para realizar un seguimiento de los errores de entrega en la aplicación de la plataforma, suscríbase a los eventos de estado de entrega de los mensajes de la aplicación. Para obtener más información, consulte Uso de los atributos de la

Nombre de atributo	Desencadenador de la notificación
	aplicaciones de Amazon SNS para el estado de entrega de los mensajes.

Puede asociar cualquier atributo a una aplicación, que podrá recibir estas notificaciones de eventos.

Envío de notificaciones push en móviles

Para enviar notificaciones de eventos de aplicaciones, debe especificar un tema para recibir las notificaciones de cada tipo de evento. Como Amazon SNS envía las notificaciones, el tema puede direccionarlas a los puntos de conexión que adoptarán acciones programáticas.

> **Important**
>
> Las aplicaciones de alto volumen crearán un gran número de notificaciones de eventos de aplicaciones (por ejemplo, decenas de miles), que sobrecargarán los puntos de conexión destinados a uso humano, como, por ejemplo, números de teléfono, direcciones de correo electrónico y aplicaciones móviles. Tenga en cuenta las siguientes directrices cuando envíe notificaciones de eventos de aplicaciones a un tema:
>
> - Cada tema que reciba notificaciones debe contener únicamente suscripciones para puntos de enlace programáticos, como puntos de enlace HTTP o HTTPS, colas de Amazon SQS o funciones. AWS Lambda
> - Para reducir la cantidad de procesamiento que las notificaciones activan, limite las suscripciones de cada tema a un número reducido (por ejemplo, cinco o menos).

Puede enviar notificaciones de eventos de aplicaciones mediante la consola de Amazon SNS, el AWS Command Line Interface (AWS CLI) o el. AWS SDKs

AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Móvil, Notificaciones push.
3. En la página Notificaciones push para dispositivos móviles, en la sección Aplicaciones de plataforma, seleccione una aplicación y, a continuación, elija Editar.

4. Expanda la sección Notificaciones de eventos.
5. Elija Acciones, Configurar eventos.
6. Introduzca ARNs los cuatro temas que se utilizarán en los siguientes eventos:
   - Punto de conexión creado
   - Punto de conexión eliminado
   - Punto de conexión actualizado
   - Error de entrega
7. Elija Guardar cambios.

AWS CLI

Ejecute el comando set-platform-application-attributes.

En el siguiente ejemplo, se establece el mismo tema de Amazon SNS para los cuatro eventos de aplicación:

```
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

Configure las notificaciones de eventos de la aplicación enviando una `SetPlatformApplicationAttributes` solicitud con la API de Amazon SNS mediante un AWS SDK.

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, incluida la ayuda para comenzar e información sobre las versiones anteriores, consulteUso de Amazon SNS con un SDK AWS.

Acciones de la API de inserción móvil

Para utilizar el servicio de notificaciones push móviles de Amazon SNS APIs, primero debe cumplir los requisitos previos para el servicio de notificaciones push, como Apple Push Notification Service (APNs) y Firebase Cloud Messaging (FCM). Para obtener más información acerca de los requisitos previos, consulte Requisitos previos para las notificaciones de usuario de Amazon SNS.

Para enviar un mensaje de notificación push a una aplicación móvil o a un dispositivo mediante el APIs, primero debes usar la `CreatePlatformApplication` acción, que devuelve un atributo. `PlatformApplicationArn` A continuación, `PlatformApplicationArn` utiliza el atributo `CreatePlatformEndpoint` y obtiene un atributo `EndpointArn`. Después puede utilizar el atributo `EndpointArn` con la acción `Publish` para enviar un mensaje de notificación a un dispositivo y una aplicación móvil, o bien puede utilizar el atributo `EndpointArn` con la acción `Subscribe` para suscribirse a un tema. Para obtener más información, consulte Configuración de notificaciones push con Amazon SNS.

Los push móviles de Amazon SNS APIs son los siguientes:

`CreatePlatformApplication`

Crea un objeto de aplicación de plataforma para uno de los servicios de notificaciones push compatibles, como APNs FCM, en el que pueden registrarse los dispositivos y las aplicaciones móviles. Devuelve un atributo `PlatformApplicationArn`, que la acción `CreatePlatformEndpoint` utiliza.

`CreatePlatformEndpoint`

Crea un punto de enlace para un dispositivo y una aplicación móvil en uno de los servicios de notificaciones de inserción admitidos. `CreatePlatformEndpoint` utiliza el atributo `PlatformApplicationArn` que devuelve la acción `CreatePlatformApplication`. El atributo `EndpointArn`, que se devuelve cuando se usa `CreatePlatformEndpoint`, se utiliza con la acción `Publish` para enviar un mensaje de notificación a una aplicación móvil y un dispositivo.

`CreateTopic`

Crea un tema en el que se pueden publicar mensajes.

`DeleteEndpoint`

Elimina el punto de enlace de un dispositivo y una aplicación móvil en uno de los servicios de notificaciones de inserción admitidos.

`DeletePlatformApplication`

Elimina un objeto de aplicación de plataforma.

`DeleteTopic`

Elimina un tema y todas sus suscripciones.

`GetEndpointAttributes`

Recupera los atributos del punto de enlace de un dispositivo y una aplicación móvil.

`GetPlatformApplicationAttributes`

Recupera los atributos del objeto de aplicación de plataforma.

`ListEndpointsByPlatformApplication`

Genera una lista de los puntos de enlace y los atributos de los puntos de enlace de los dispositivos y aplicaciones móviles de un servicio de notificaciones de inserción compatible.

`ListPlatformApplications`

Genera una lista de objetos de aplicación de plataforma para los servicios de notificaciones de inserción compatibles.

`Publish`

Envía un mensaje de notificación a todos los puntos de enlace suscritos a un tema.

`SetEndpointAttributes`

Establece los atributos de un punto de enlace de un dispositivo y una aplicación móvil.

`SetPlatformApplicationAttributes`

Establece los atributos del objeto de aplicación de plataforma.

`Subscribe`

Prepara la suscripción de un punto de enlace enviando a dicho punto de enlace un mensaje de confirmación. Para crear realmente una suscripción, el propietario del terminal debe ejecutar la ConfirmSubscription acción con el token del mensaje de confirmación.

`Unsubscribe`

Elimina una suscripción.

Errores comunes de la API de notificaciones push para móvil de Amazon SNS

Los errores devueltos por Amazon SNS APIs para mobile push se muestran en la siguiente tabla. Para obtener más información sobre Amazon SNS APIs for mobile push, consulte. Acciones de la API de inserción móvil

Error	Descripción	Código de estado HTTPS	Acción API
Application Name is null string	El nombre de aplicación solicitado está establecido en null.	400	`CreatePlatformApplication`
Platform Name is null string	El nombre de plataforma solicitado está establecido en null.	400	`CreatePlatformApplication`
Platform Name is invalid	Se proporcionó un out-of-range valor o un valor no válido para el nombre de la plataforma.	400	`CreatePlatformApplication`
APNs — El certificado principal no es válido	Se proporcionó un certificado no válido para el APNs principal, que es el certificado SSL. Para obtener más información, consulte la referencia a CreatePlatformApplicationde la API de Amazon Simple Notification Service.	400	`CreatePlatformApplication`

Error	Descripción	Código de estado HTTPS	Acción API
APNs — El certificado principal es válido, pero no está en formato.pem	Se proporcionó un certificado válido que no está en el formato.pem para el APNs principal, que es el certificado SSL.	400	`CreatePlatformApplication`
APNs — El principal es un certificado caducado	Se proporcionó un certificado caducado para el APNs principal, que es el certificado SSL.	400	`CreatePlatformApplication`
APNs — El certificado principal no es un certificado emitido por Apple	Se ha proporcionado un certificado que no ha generado Apple para el principal de APNs, que es el certificado SSL.	400	`CreatePlatformApplication`
APNs — No se proporciona el principal	No se proporcionó el APNs principal, que es el certificado SSL.	400	`CreatePlatformApplication`
APNs — No se proporciona la credencial	No se proporcionó la APNs credencial, que es la clave privada. Para obtener más información, consulte la referencia a CreatePlatformApplicationde la API de Amazon Simple Notification Service.	400	`CreatePlatformApplication`

Error	Descripción	Código de estado HTTPS	Acción API
APNs — Las credenciales no tienen un formato.pem válido	La APNs credencial, que es la clave privada, no tiene un formato .pem válido.	400	`CreatePlatformApplication`
FCM: no se proporciona el servidor APIKey	La credencial de CM, que es la clave de API, no se ha proporcionado. Para obtener más información, consulte la referencia CreatePlatformApplicationde la API de Amazon Simple Notification Service.	400	`CreatePlatformApplication`
FCM: el servidor APIKey está vacío	La credencial de FCM, que es la clave de API, está vacía.	400	`CreatePlatformApplication`
FCM: el servidor APIKey es una cadena nula	La credencial de FCM, que es la clave de API, es nula.	400	`CreatePlatformApplication`
FCM: el servidor APIKey no es válido	La credencial de FCM, que es la clave de API, no es válida.	400	`CreatePlatformApplication`
ADM — clientsecret is not provided	La clave secreta de cliente requerida no se ha proporcionado.	400	`CreatePlatformApplication`

Error	Descripción	Código de estado HTTPS	Acción API
ADM — clientsecret is a null string	La cadena requerida para la clave secreta de cliente es nula.	400	`CreatePlatformApplication`
ADM — client_secret is empty string	La cadena requerida para la clave secreta de cliente está vacía.	400	`CreatePlatformApplication`
ADM — client_secret is not valid	La cadena requerida para la clave secreta de cliente no es válida.	400	`CreatePlatformApplication`
ADM — client_id is empty string	La cadena requerida para el ID de cliente está vacía.	400	`CreatePlatformApplication`
ADM — clientId is not provided	La cadena requerida para el ID de cliente no se ha proporcionado.	400	`CreatePlatformApplication`
ADM — clientid is a null string	La cadena requerida para el ID de cliente es nula.	400	`CreatePlatformApplication`
ADM — client_id is not valid	La cadena requerida para el ID de cliente no es válida.	400	`CreatePlatformApplication`
EventEndpointCreated tiene un formato de ARN no válido	EventEndpointCreated tiene un formato de ARN no válido.	400	`CreatePlatformApplication`

Error	Descripción	Código de estado HTTPS	Acción API
EventEndpointDeleted tiene un formato de ARN no válido	EventEndpointDeleted tiene un formato de ARN no válido.	400	`CreatePlatformApplication`
EventEndpointUpdated tiene un formato de ARN no válido	EventEndpointUpdated tiene un formato de ARN no válido.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure tiene un formato de ARN no válido	EventDeliveryAttemptFailure tiene un formato de ARN no válido.	400	`CreatePlatformApplication`
EventDeliveryFailure tiene un formato de ARN no válido	EventDeliveryFailure tiene un formato de ARN no válido.	400	`CreatePlatformApplication`
EventEndpointCreated no es un tema existente	EventEndpointCreated no es un tema existente.	400	`CreatePlatformApplication`
EventEndpointDeleted no es un tema existente	EventEndpointDeleted no es un tema existente.	400	`CreatePlatformApplication`
EventEndpointUpdated no es un tema existente	EventEndpointUpdated no es un tema existente.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure no es un tema existente	EventDeliveryAttemptFailure no es un tema existente.	400	`CreatePlatformApplication`

Error	Descripción	Código de estado HTTPS	Acción API
EventDeliveryFailure no es un tema existente	EventDeliveryFailure no es un tema existente.	400	`CreatePlatformApplication`
Platform ARN is invalid	El ARN de la plataforma no es válido.	400	`SetPlatformAttributes`
Platform ARN is valid but does not belong to the user	El ARN de la plataforma es válido, pero no pertenece al usuario.	400	`SetPlatformAttributes`
APNs — El principal no es un certificado válido	Se proporcionó un certificado no válido para el APNs principal, que es el certificado SSL. Para obtener más información, consulte la referencia a CreatePlatformApplicationde la API de Amazon Simple Notification Service.	400	`SetPlatformAttributes`
APNs — El certificado principal es válido, pero no está en formato.pem	Se proporcionó un certificado válido que no está en el formato.pem para el APNs principal, que es el certificado SSL.	400	`SetPlatformAttributes`

Error	Descripción	Código de estado HTTPS	Acción API
APNs — El principal es un certificado caducado	Se proporcionó un certificado caducado para el APNs principal, que es el certificado SSL.	400	`SetPlatformAttributes`
APNs — El certificado principal no es un certificado emitido por Apple	Se ha proporcionado un certificado que no ha generado Apple para el principal de APNs, que es el certificado SSL.	400	`SetPlatformAttributes`
APNs — No se proporciona el principal	No se proporcionó el APNs principal, que es el certificado SSL.	400	`SetPlatformAttributes`
APNs — No se proporciona la credencial	No se proporcionó la APNs credencial, que es la clave privada. Para obtener más información, consulte la referencia a CreatePlatformApplicationde la API de Amazon Simple Notification Service.	400	`SetPlatformAttributes`
APNs — Las credenciales no tienen un formato.pem válido	La APNs credencial, que es la clave privada, no tiene un formato .pem válido.	400	`SetPlatformAttributes`

Error	Descripción	Código de estado HTTPS	Acción API
FCM: no se proporciona el servidor APIKey	La credencial de CM, que es la clave de API, no se ha proporcionado. Para obtener más información, consulte la referencia CreatePlatformApplicationde la API de Amazon Simple Notification Service.	400	`SetPlatformAttributes`
FCM: el servidor APIKey es una cadena nula	La credencial de FCM, que es la clave de API, es nula.	400	`SetPlatformAttributes`
ADM — clientId is not provided	La cadena requerida para el ID de cliente no se ha proporcionado.	400	`SetPlatformAttributes`
ADM — clientid is a null string	La cadena requerida para el ID de cliente es nula.	400	`SetPlatformAttributes`
ADM — clientsecret is not provided	La clave secreta de cliente requerida no se ha proporcionado.	400	`SetPlatformAttributes`
ADM — clientsecret is a null string	La cadena requerida para la clave secreta de cliente es nula.	400	`SetPlatformAttributes`

Error	Descripción	Código de estado HTTPS	Acción API
EventEndpointUpdated tiene un formato de ARN no válido	EventEndpointUpdated tiene un formato de ARN no válido.	400	SetPlatformAttributes
EventEndpointDeleted tiene un formato de ARN no válido	EventEndpointDeleted tiene un formato de ARN no válido.	400	SetPlatformAttributes
EventEndpointUpdated tiene un formato de ARN no válido	EventEndpointUpdated tiene un formato de ARN no válido.	400	SetPlatformAttributes
EventDeliveryAttemptFailure tiene un formato de ARN no válido	EventDeliveryAttemptFailure tiene un formato de ARN no válido.	400	SetPlatformAttributes
EventDeliveryFailure tiene un formato de ARN no válido	EventDeliveryFailure tiene un formato de ARN no válido.	400	SetPlatformAttributes
EventEndpointCreated no es un tema existente	EventEndpointCreated no es un tema existente.	400	SetPlatformAttributes
EventEndpointDeleted no es un tema existente	EventEndpointDeleted no es un tema existente.	400	SetPlatformAttributes
EventEndpointUpdated no es un tema existente	EventEndpointUpdated no es un tema existente.	400	SetPlatformAttributes
EventDeliveryAttemptFailure no es un tema existente	EventDeliveryAttemptFailure no es un tema existente.	400	SetPlatformAttributes

Error	Descripción	Código de estado HTTPS	Acción API
EventDeliveryFailure no es un tema existente	EventDeliveryFailure no es un tema existente.	400	`SetPlatformAttributes`
Platform ARN is invalid	El ARN de la plataforma no es válido.	400	`GetPlatformApplicationAttributes`
Platform ARN is valid but does not belong to the user	El ARN de la plataforma es válido, pero no pertenece al usuario.	403	`GetPlatformApplicationAttributes`
Token specified is invalid	El token especificado no es válido.	400	`ListPlatformApplications`
Platform ARN is invalid	El ARN de la plataforma no es válido.	400	`ListEndpointsByPlatformApplication`
Platform ARN is valid but does not belong to the user	El ARN de la plataforma es válido, pero no pertenece al usuario.	404	`ListEndpointsByPlatformApplication`
Token specified is invalid	El token especificado no es válido.	400	`ListEndpointsByPlatformApplication`
Platform ARN is invalid	El ARN de la plataforma no es válido.	400	`DeletePlatformApplication`

Error	Descripción	Código de estado HTTPS	Acción API
Platform ARN is valid but does not belong to the user	El ARN de la plataforma es válido, pero no pertenece al usuario.	403	`DeletePlatformApplication`
Platform ARN is invalid	El ARN de la plataforma no es válido.	400	`CreatePlatformEndpoint`
Platform ARN is valid but does not belong to the user	El ARN de la plataforma es válido, pero no pertenece al usuario.	404	`CreatePlatformEndpoint`
Token is not specified	El token no se ha especificado.	400	`CreatePlatformEndpoint`
Token is not of correct length	El token no tiene la longitud correcta.	400	`CreatePlatformEndpoint`
Customer User data is too large	Los datos de usuario del cliente no pueden tener más de 2048 bytes en la codificación UTF-8.	400	`CreatePlatformEndpoint`
Endpoint ARN is invalid	El ARN del punto de enlace no es válido.	400	`DeleteEndpoint`
Endpoint ARN is valid but does not belong to the user	El ARN del punto de enlace es válido, pero no pertenece al usuario.	403	`DeleteEndpoint`

Error	Descripción	Código de estado HTTPS	Acción API
Endpoint ARN is invalid	El ARN del punto de enlace no es válido.	400	`SetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	El ARN del punto de enlace es válido, pero no pertenece al usuario.	403	`SetEndpointAttributes`
Token is not specified	El token no se ha especificado.	400	`SetEndpointAttributes`
Token is not of correct length	El token no tiene la longitud correcta.	400	`SetEndpointAttributes`
Customer User data is too large	Los datos de usuario del cliente no pueden tener más de 2048 bytes en la codificación UTF-8.	400	`SetEndpointAttributes`
Endpoint ARN is invalid	El ARN del punto de enlace no es válido.	400	`GetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	El ARN del punto de enlace es válido, pero no pertenece al usuario.	403	`GetEndpointAttributes`
Target ARN is invalid	El ARN de destino no es válido.	400	`Publish`
Target ARN is valid but does not belong to the user	El ARN de destino es válido, pero no pertenece al usuario.	403	`Publish`

Error	Descripción	Código de estado HTTPS	Acción API
Message format is invalid	El formato del mensaje no es válido.	400	`Publish`
Message size is larger than supported by protocol/end-service	El tamaño del mensaje es superior al tamaño compatibl e con el protocolo/ servicio final.	400	`Publish`

Uso del atributo de mensaje de “time-to-live” de Amazon SNS para las notificaciones push para móvil

Con Amazon Simple Notification Service (Amazon SNS), se admite la configuración de un atributo de mensaje de período de vida (TTL) para los mensajes de notificaciones push móviles. Esto se suma a la capacidad de configuración de TTL dentro del cuerpo del mensaje de Amazon SNS para los servicios de notificaciones push para móvil que admiten esta funcionalidad, como Amazon Device Messaging (ADM) y Firebase Cloud Messaging (FCM) cuando se envía a Android.

El atributo de mensaje TTL se utiliza para especificar metadatos de vencimiento de un mensaje. Esto te permite especificar la cantidad de tiempo que el servicio de notificaciones push, como el Servicio de Notificaciones Push de Apple (APNs) o FCM, tiene para entregar el mensaje al punto final. Si, por algún motivo, (por ejemplo, el dispositivo móvil se ha apagado) no se puede entregar el mensaje en el TTL especificado, se abandonará dicho mensaje y no se realizará ningún otro intento de entrega. Para especificar el TTL en los atributos de los mensajes, puedes usar los AWS Management Console kits de desarrollo de AWS software (SDKs) o la API de consultas.

Atributos de los mensajes TTL para los servicios de notificaciones de inserción

A continuación, se incluye una lista de los atributos de los mensajes TTL para los servicios de notificaciones push que puede utilizar para configurar al utilizar la API AWS SDKs o la API de consulta:

Servicio de notificaciones de inserción	Atributo de los mensajes TTL
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Servicio de notificaciones push de Apple () APNs	`AWS.SNS.MOBILE.APNS.TTL`
Sandbox del servicio de notificaciones push de Apple (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCM cuando se envía a Android)	`AWS.SNS.MOBILE.FCM.TTL`
Servicios de notificación push de Windows (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Cada servicio de notificaciones push administra el TTL de forma distinta. Con Amazon SNS, se ofrece una vista resumida de TTL de todos los servicios de notificaciones push, lo que facilita la especificación del TTL. Si utiliza el AWS Management Console para especificar el TTL (en segundos), solo tiene que introducir el valor TTL una vez y Amazon SNS calculará el TTL de cada uno de los servicios de notificaciones push seleccionados al publicar el mensaje.

El TTL depende de la hora de publicación. Antes de entregar un mensaje de notificación push a un servicio de notificaciones push concreto, Amazon SNS calcula el tiempo de permanencia (la marca de tiempo entre la publicación y el momento previo a la entrega de un servicio de notificaciones push) de la notificación push y traslada el resto del TTL al servicio de notificaciones push específico. Si TTL es inferior al tiempo de permanencia, Amazon SNS no intentará publicar.

Si especificas un TTL para un mensaje de notificación push, el valor TTL debe ser un entero positivo, a menos que el valor de `0` tenga un significado específico para el servicio de notificaciones push, como con APNs y FCM (cuando se envía a Android). Si el valor de TTL se establece en `0` y el servicio de notificaciones push no tiene un significado específico para `0`, Amazon SNS eliminará el mensaje. Para obtener más información sobre el parámetro TTL establecido `0` al usarlo APNs, consulta la tabla A-3 sobre los identificadores de elementos para las notificaciones remotas en la documentación de la API de proveedores binarios.

Orden de prioridad para determinar el TTL

La prioridad que Amazon SNS utiliza para determinar el TTL de un mensaje de notificación push sigue el orden siguiente, en el que el número más bajo tiene la máxima prioridad:

1. TTL del atributo de mensaje
2. TTL del cuerpo del mensaje
3. TTL predeterminado del servicio de notificaciones de inserción (varía según el servicio)
4. TTL predeterminado de Amazon SNS (4 semanas)

Si configura diferentes valores de TTL (uno en los atributos del mensaje y otro en el cuerpo del mensaje) para el mismo mensaje, Amazon SNS modificará el TTL del cuerpo del mensaje para que coincida con el TTL especificado en el atributo del mensaje.

Especificar el TTL mediante AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Móvil, Notificaciones push.
3. En la página Notificaciones push móviles, en la sección Aplicaciones de la plataforma, seleccione una aplicación y, a continuación, elija Editar.
4. En la ***MyApplication*** página, en la sección Puntos finales, elija un punto final de la aplicación y, a continuación, elija Publicar mensaje.
5. En la sección Message details (Detalles del mensaje), escriba el TTL (los segundos que tiene el servicio de notificaciones push para entregar el mensaje al punto de enlace).
6. Elija Publish message (Publicar mensaje).

Regiones compatibles con aplicaciones móviles de Amazon SNS

En este momento, puede crear aplicaciones móviles en las siguientes regiones:

- Este de EE. UU. (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- África (Ciudad del Cabo)

- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Yakarta)
- Asia-Pacífico (Bombay)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milán)
- Europa (París)
- Europa (Estocolmo)
- Medio Oriente (Baréin)
- Medio Oriente (EAU)
- América del Sur (São Paulo)
- AWS GovCloud (EE. UU.-Oeste)

Prácticas recomendadas para administrar notificaciones push para móvil en Amazon SNS

En esta sección se describen diversas prácticas recomendadas que es posible que le ayuden a mejorar la implicación de los clientes.

Administración de puntos de conexión

Pueden producirse problemas de entrega en situaciones en las que los tokens de dispositivo cambien debido a la acción de un usuario en el dispositivo (por ejemplo, se vuelve a instalar una aplicación en el dispositivo), o actualizaciones de certificados que afectan a los dispositivos que se ejecutan en una versión de iOS determinada. Apple recomienda registrarse APNs cada vez que se lance la aplicación.

Dado que el token del dispositivo no cambia cada vez que un usuario abre una aplicación, se puede usar la API de `CreatePlatformEndpoint` idempotente. Sin embargo, esto puede crear duplicados para el mismo dispositivo en los casos en que el token en sí no es válido o si el punto de conexión es válido pero está desactivado (por ejemplo, una discrepancia entre los entornos de pruebas y de producción).

Se puede usar un mecanismo de administración de tokens de dispositivo como el del pseudocódigo.

Para obtener más información sobre la administración y el mantenimiento de los tokens de dispositivo de FCM v1, consulte Administración de los puntos de conexión de Firebase Cloud Messaging en Amazon SNS.

Registro de estado de entrega

Para monitorear el estado de entrega de notificaciones push, le recomendamos que habilite el registro del estado de entrega para la aplicación de la plataforma de Amazon SNS. Esto le ayuda a solucionar los errores de entrega porque los registros contienen códigos de respuesta de un proveedor devueltos del servicio de plataforma push. Para obtener más información sobre cómo habilitar el registro del estado de entrega, consulte ¿Cómo accedo a los registros de entrega de temas de Amazon SNS para notificaciones push?.

Notificaciones de eventos

Para administrar puntos de conexión de forma impulsada por eventos, puede utilizar la funcionalidad notificaciones de eventos. Esto permite que el tema de Amazon SNS configurado elimine eventos a los suscriptores, como una función Lambda, para eventos de aplicaciones de plataforma de creación, eliminación, actualizaciones y errores de entrega de puntos de conexión.

Configuración y administración de suscripciones de correo electrónico de Amazon SNS

Puede suscribir una dirección de correo electrónico a un tema de Amazon SNS mediante AWS Management Console AWS SDK para Java, o. AWS SDK para .NET

> **Notas**
>
> - No se admite la personalización del cuerpo del mensaje de correo electrónico. La función de entrega de correo electrónico está diseñada para proporcionar alertas internas del sistema, no mensajes de marketing.

- Los puntos de conexión de correo electrónico de suscripción directa se admiten solo para temas estándar.
- El rendimiento de la entrega de correo electrónico está limitado. Para obtener más información, consulte Cuotas de Amazon SNS.

Important

Para evitar que los destinatarios de la lista de correo cancelen la suscripción a los correos electrónicos de temas de Amazon SNS, consulte Configurar una suscripción de correo electrónico que requiera autenticación para cancelar la suscripción en AWS Support.

Suscribir una dirección de correo electrónico a un tema de Amazon SNS mediante el AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación izquierdo, elija Suscripciones.
3. En la página Subscriptions (Suscripciones), elija Create subscription (Crear suscripción).
4. En la página Crear suscripción, en la sección Detalles, haga lo siguiente:
   a. En ARN de tema, elija el nombre de recurso de Amazon (ARN) de un tema.
   b. En Protocolo, elige Correo electrónico.
   c. En Punto de enlace, ingrese su dirección de correo electrónico.
   d. (Opcional) Para configurar una política de filtro, expanda la sección Política de filtro de suscripción. Para obtener más información, consulte Políticas de filtro de suscripciones de Amazon SNS.
   e. (Opcional) Para habilitar el filtrado basado en cargas, configure `Filter Policy Scope` en `MessageBody`. Para obtener más información, consulte Alcance de políticas de filtrado de suscripciones de Amazon SNS.
   f. (Opcional) Para configurar una cola de mensajes fallidos en la suscripción, expanda la sección Política de reconducción (cola de mensajes fallidos). Para obtener más información, consulte Colas de mensajes fallidos de Amazon SNS.
   g. Seleccione Crear subscripción.

En la consola se crea la suscripción y se abre la página Detalles de la suscripción.

Debe confirmar la suscripción antes de que se pueda comenzar a recibir mensajes en la dirección de correo electrónico.

Para confirmar una suscripción, siga estos pasos:

1. Verifique la bandeja de entrada de correo electrónico y elija Confirmar la suscripción en el correo electrónico de Amazon SNS.
2. En Amazon SNS, se abre su navegador web y se muestra una confirmación de suscripción con su ID de suscripción.

Suscribir una dirección de correo electrónico a un tema de Amazon SNS mediante un SDK AWS

Para usar un AWS SDK, debe configurarlo con sus credenciales. Para obtener más información, consulte Los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

Los siguientes ejemplos de código muestran cómo utilizar `Subscribe`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

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

Suscriba una cola a un tema con filtros opcionales.

```
    /// <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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

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

Suscriba una aplicación móvil a un tema.

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

Suscriba una función de Lambda a un tema.

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

Suscriba una cola de SQS a un tema.

```
        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;
            }
```

Suscribirse con un filtro a un tema.

```
       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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para C++ .

CLI

AWS CLI

Suscripción a un tema

El siguiente comando `subscribe` suscribe una dirección de correo electrónico al tema especificado.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Salida:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Para obtener detalles sobre la API, consulte Subscribe en la Referencia de comandos de la AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una cola a un tema con filtros opcionales.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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);
        }
    }
}
```

Suscriba un punto de conexión HTTP a un tema.

```
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);
        }
    }
}
```

Suscriba una función de Lambda a un tema.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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'
  // }
};
```

Suscriba una aplicación móvil a un tema.

```
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;
};
```

Suscriba una función de Lambda a un tema.

```
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;
};
```

Suscriba una cola de SQS a un tema.

```
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;
};
```

Suscribirse con un filtro a un tema.

```
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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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()
    }
}
```

Suscriba una función de Lambda a un tema.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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());
}
```

Suscriba un punto de conexión HTTP a un tema.

```
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 obtener información sobre la API, consulte Suscríbase en la Referencia de la API de AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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 obtener información sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener información sobre la API, consulte Suscríbase en la Referencia de la API de AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
    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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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.")
```

Suscribe un número de teléfono a un tema para recibir notificaciones 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 obtener más información sobre la API, consulta la referencia sobre la API de Suscríbete en el AWS SDK para Swift.

Ejemplos de código para Amazon SNS mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar Amazon SNS con un kit de desarrollo de AWS software (SDK).

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción

Hola Amazon SNS

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;
```

```
public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Código para el CMake archivo 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 del archivo de origen 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 obtener más información sobre la API, consulte ListTopicsla Referencia AWS SDK para C++ de la API.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

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

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Inicialice un cliente de SNS y muestre los temas de la cuenta.

```
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 obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import aws.sdk.kotlin.services.sns.SnsClient
```

```
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla referencia sobre el AWS SDK para la API de Kotlin.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

El archivo 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")

    ]
)
```

El programa principal de 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 obtener más información sobre la API, consulta ListTopicsla referencia sobre la API de AWS SDK para Swift.

Ejemplos de código

- Ejemplos básicos de uso de Amazon SNS AWS SDKs
  - Hola Amazon SNS
  - Acciones para Amazon SNS mediante AWS SDKs
    - Úselo CheckIfPhoneNumberIsOptedOut con un AWS SDK o CLI
    - Úselo ConfirmSubscription con un AWS SDK o CLI
    - Úselo CreateTopic con un AWS SDK o CLI
    - Úselo DeleteTopic con un AWS SDK o CLI
    - Úselo GetSMSAttributes con un AWS SDK o CLI
    - Úselo GetTopicAttributes con un AWS SDK o CLI
    - Úselo ListPhoneNumbersOptedOut con un AWS SDK o CLI
    - Úselo ListSubscriptions con un AWS SDK o CLI

Ejemplos básicos de uso de Amazon SNS AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar los conceptos básicos de Amazon Simple Notification Service con AWS SDKs.

Ejemplos

- Hola Amazon SNS
- Acciones para Amazon SNS mediante AWS SDKs
  - Úselo CheckIfPhoneNumberIsOptedOut con un AWS SDK o CLI
  - Úselo ConfirmSubscription con un AWS SDK o CLI
  - Úselo CreateTopic con un AWS SDK o CLI
  - Úselo DeleteTopic con un AWS SDK o CLI
  - Úselo GetSMSAttributes con un AWS SDK o CLI
  - Úselo GetTopicAttributes con un AWS SDK o CLI
  - Úselo ListPhoneNumbersOptedOut con un AWS SDK o CLI
  - Úselo ListSubscriptions con un AWS SDK o CLI
  - Úselo ListTopics con un AWS SDK o CLI
  - Úselo Publish con un AWS SDK o CLI
  - Úselo SetSMSAttributes con un AWS SDK o CLI
  - Úselo SetSubscriptionAttributes con un AWS SDK o CLI
  - SetSubscriptionAttributesRedrivePolicyÚselo con un AWS SDK
  - Úselo SetTopicAttributes con un AWS SDK o CLI
  - Úselo Subscribe con un AWS SDK o CLI
  - Úselo TagResource con un AWS SDK o CLI
  - Úselo Unsubscribe con un AWS SDK o CLI

Hola Amazon SNS

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Código para el CMake archivo 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 del archivo de origen 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 obtener más información sobre la API, consulte ListTopicsla Referencia AWS SDK para C++ de la API.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
```

```
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

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

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Inicialice un cliente de SNS y muestre los temas de la cuenta.

```
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 obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
```

```
}
```

- Para obtener más información sobre la API, consulta ListTopicsla referencia sobre el AWS SDK para la API de Kotlin.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

El archivo 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")

    ]
)
```

El programa principal de 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 obtener más información sobre la API, consulta ListTopicsla referencia sobre la API de AWS SDK para Swift.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones para Amazon SNS mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo realizar acciones individuales de Amazon SNS con. AWS SDKs Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones para configurar y ejecutar el código.

Estos fragmentos llaman a la API de Amazon SNS y son fragmentos de código de programas más grandes que se deben ejecutar en contexto. Puede ver las acciones en contexto en Escenarios de uso de Amazon SNS AWS SDKs .

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para obtener una lista completa, consulte la Referencia de la API de Amazon Simple Notification Service.

Ejemplos

- Úselo CheckIfPhoneNumberIsOptedOut con un AWS SDK o CLI
- Úselo ConfirmSubscription con un AWS SDK o CLI
- Úselo CreateTopic con un AWS SDK o CLI
- Úselo DeleteTopic con un AWS SDK o CLI
- Úselo GetSMSAttributes con un AWS SDK o CLI
- Úselo GetTopicAttributes con un AWS SDK o CLI
- Úselo ListPhoneNumbersOptedOut con un AWS SDK o CLI
- Úselo ListSubscriptions con un AWS SDK o CLI
- Úselo ListTopics con un AWS SDK o CLI
- Úselo Publish con un AWS SDK o CLI
- Úselo SetSMSAttributes con un AWS SDK o CLI
- Úselo SetSubscriptionAttributes con un AWS SDK o CLI
- SetSubscriptionAttributesRedrivePolicyÚselo con un AWS SDK

- Úselo SetTopicAttributes con un AWS SDK o CLI
- Úselo Subscribe con un AWS SDK o CLI
- Úselo TagResource con un AWS SDK o CLI
- Úselo Unsubscribe con un AWS SDK o CLI

Úselo **`CheckIfPhoneNumberIsOptedOut`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out.
/// </summary>
public class IsPhoneNumOptedOut
{
    public static async Task Main()
    {
        string phoneNumber = "+15551112222";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await CheckIfOptedOutAsync(client, phoneNumber);
    }
```

```
        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Comprobar que se ha cancelado la recepción de mensajes SMS en un número de teléfono

En el siguiente `check-if-phone-number-is-opted-out` ejemplo, se comprueba si el número de teléfono especificado está excluido de la recepción de mensajes SMS de la AWS cuenta corriente.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Salida:

```
{
    "isOptedOut": false
}
```

- Para obtener más información sobre la API, consulte CheckIfPhoneNumberIsOptedOutla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
```

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

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

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

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

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
```

```
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK para JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta CheckIfPhoneNumberIsOptedOutla Referencia AWS SDK para PHP de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ConfirmSubscription`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ConfirmSubscription`.

CLI

AWS CLI

Confirmar una suscripción

El siguiente comando `confirm-subscription` completa el proceso de confirmación que se inició al suscribirse a un tema de SNS denominado `my-topic`. El parámetro --token proviene del mensaje de confirmación enviado al punto de conexión de notificación especificado en la llamada de suscripción.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Salida:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Para obtener más información sobre la API, consulte ConfirmSubscriptionla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

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

- Para obtener más información sobre la API, consulta ConfirmSubscriptionla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta ConfirmSubscriptionla Referencia AWS SDK para JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

```
/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta ConfirmSubscriptionla Referencia AWS SDK para PHP de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`CreateTopic`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateTopic`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- Creación y publicación en un tema FIFO
- Publicación de mensajes en colas

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree un tema con un nombre 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;
        }
    }
```

Cree un tema nuevo con un nombre y atributos específicos de FIFO y desduplicación.

```
    /// <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 obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Creación de un tema de SNS

En el siguiente ejemplo de `create-topic` se crea un tema de SNS denominado `my-topic`.

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

Salida:

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

Para obtener más información, consulte Uso de la interfaz de línea de AWS comandos con Amazon SQS y Amazon SNS en la Guía del usuario de AWS la interfaz de línea de comandos.

- Para obtener más información sobre la API, consulte la Referencia CreateTopicde AWS CLI comandos.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;
```

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

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

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

- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulta CreateTopicla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
```

```
        else:
            return topic
```

- Para obtener más información sobre la API, consulta CreateTopicla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener más información sobre la API, consulta CreateTopicla Referencia AWS SDK para Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Para obtener más información sobre la API, consulta CreateTopicla referencia sobre la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte CreateTopicla referencia sobre la API ABAP del AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta CreateTopicla referencia sobre la API de AWS SDK for Swift.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteTopic`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteTopic`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Publicación de mensajes en colas

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Elimine un tema por su ARN de tema.

```
    /// <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 obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

    Aws::SNS::Model::DeleteTopicRequest request;
```

```
    request.SetTopicArn(topicARN);

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

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

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Eliminación de un tema de SNS

El siguiente ejemplo de `delete-topic` elimina el tema de SNS especificado.

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

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia de AWS CLI comandos.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

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

```
        }

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

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulta DeleteTopicla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta DeleteTopicla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Para obtener más información sobre la API, consulta DeleteTopicla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
```

```
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte DeleteTopicla referencia sobre la API ABAP del AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import AWSSNS

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

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

- Para obtener más información sobre la API, consulta DeleteTopicla referencia sobre la API de AWS SDK for Swift.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetSMSAttributes`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
```

```
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta la referencia sobre cómo entrar SMSAttributes en la AWS SDK para C++ API.

CLI

AWS CLI

Mostrar los atributos predeterminados de los mensajes SMS

En el siguiente ejemplo de `get-sms-attributes`, se muestran los atributos predeterminados para enviar mensajes SMS.

```
aws sns get-sms-attributes
```

Salida:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Para obtener información sobre la API, consulte GetSMSAttributes en la Referencia de comandos de la AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

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

```
            System.exit(1);
        }

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

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

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

        System.out.println("\n\nStatus was good");
    }
}
```

- Para obtener más información sobre la API, consulta la referencia sobre cómo entrar SMSAttributes en la AWS SDK for Java 2.x API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta la referencia sobre cómo entrar SMSAttributes en la AWS SDK para JavaScript API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta la referencia sobre cómo entrar SMSAttributes en la AWS SDK para PHP API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetTopicAttributes`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetTopicAttributes`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
 topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
 GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
```

```
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
 topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- Para obtener más información sobre la API, consulta GetTopicAttributesla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta GetTopicAttributesla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Recuperación de los atributos de un tema

En el siguiente ejemplo de `get-topic-attributes`, se muestran los atributos del tema especificado.

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

Salida:

```
{
    "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 obtener más información sobre la API, consulta GetTopicAttributesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

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

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

        System.out.println("Getting attributes for a topic with name: " +
 topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
 topicArn) {
        try {
            GetTopicAttributesRequest request =
 GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
```

```
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

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

- Para obtener más información sobre la API, consulta GetTopicAttributesla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta GetTopicAttributesla Referencia AWS SDK para JavaScript de la API.

SDK para JavaScript (v2)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Importe el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta GetTopicAttributesla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Para obtener más información sobre la API, consulta GetTopicAttributesla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta GetTopicAttributesla Referencia AWS SDK para PHP de la API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte GetTopicAttributesla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListPhoneNumbersOptedOut`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Mostrar exclusiones de mensajes SMS

El siguiente ejemplo de `list-phone-numbers-opted-out` muestra los números de teléfono excluidos de la recepción de mensajes SMS.

```
aws sns list-phone-numbers-opted-out
```

Salida:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Para obtener más información sobre la API, consulte ListPhoneNumbersOptedOutla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

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

- Para obtener más información sobre la API, consulta ListPhoneNumbersOptedOutla Referencia AWS SDK for Java 2.x de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta ListPhoneNumbersOptedOutla Referencia AWS SDK para PHP de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListSubscriptions`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListSubscriptions`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example will retrieve a list of the existing Amazon Simple
/// Notification Service (Amazon SNS) subscriptions.
/// </summary>
public class ListSubscriptions
{
    public static async Task Main()
    {
```

```
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            Console.WriteLine("Enter a topic ARN to list subscriptions for a
 specific topic, " +
                              "or press Enter to list subscriptions for all
 topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client,
 topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
```

```
                }
            }
            else
            {
                var paginateAllSubscriptions =
client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }

        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
```

```
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Mostrar las suscripciones de SNS

En el siguiente `list-subscriptions` ejemplo, se muestra una lista de las suscripciones de redes sociales de su AWS cuenta.

```
aws sns list-subscriptions
```

Salida:

```
{
    "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 obtener más información sobre la API, consulte ListSubscriptionsla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

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

```
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result =
 snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

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

- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listSNSSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListSubscriptionsla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

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

- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
 returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- Para obtener más información sobre la API, consulta ListSubscriptionsla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener más información sobre la API, consulta ListSubscriptionsla Referencia AWS SDK para Ruby de la API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
        MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list subscribers.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte ListSubscriptionsla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListTopics`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListTopics`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// Lists the Amazon Simple Notification Service (Amazon SNS)
    /// topics for the current account.
    /// </summary>
    public class ListSNSTopics
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
        public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }

        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
```

```
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Mostrar los temas de SNS

En el siguiente `list-topics` ejemplo, se enumeran todos los temas de SNS de tu AWS cuenta.

```
aws sns list-topics
```

Salida:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Para obtener más información sobre la API, consulte ListTopicsla Referencia de AWS CLI comandos.

Go

SDK para Go V2

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para Go de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

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

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListTopicsla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


/**
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

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

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

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- Para obtener más información sobre la API, consulta ListTopicsla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.

- Para obtener más información sobre la API, consulta ListTopicsla Referencia AWS SDK para Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- Para obtener más información sobre la API, consulta ListTopicsla referencia sobre la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        oo_result = lo_sns->listtopics( ).           " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte ListTopicsla referencia sobre la API ABAP del AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta ListTopicsla referencia sobre la API de AWS SDK for Swift.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`Publish`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `Publish`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- Creación y publicación en un tema FIFO
- Publicación de un mensaje SMS
- Publicación de mensajes en colas

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Publique un mensaje en un tema

```
    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 un mensaje en un tema con opciones de grupo, duplicación y 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 las selecciones del usuario a la acción de publicación.

```
    /// <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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

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

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

```
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Publique un mensaje con un 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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para C++ .

CLI

AWS CLI

Ejemplo 1: Publicar un mensaje en un tema

En el siguiente ejemplo de `publish` se publica el mensaje especificado en el tema de SNS especificado. El mensaje proviene de un archivo de texto que le permite incluir saltos de línea.

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

Contenido de `message.txt`:

```
Hello World
Second Line
```

Salida:

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

```
}
```

Ejemplo 2: publicar un mensaje SMS en un número de teléfono

En el siguiente ejemplo de `publish`, se publica el mensaje `Hello world!` en el número de teléfono `+1-555-555-0100`.

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

Salida:

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia del comando de la AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 un mensaje en un tema con opciones de grupo, duplicación y 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para PHP .

PowerShell

Herramientas para la PowerShell V4

Ejemplo 1: En este ejemplo se muestra la publicación de un mensaje con una sola línea MessageAttribute declarada.

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

Ejemplo 2: En este ejemplo se muestra la publicación de un mensaje con varios MessageAttributes declarados de antemano.

```
$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 obtener información sobre la API, consulte Publish in Herramientas de AWS para PowerShell Cmdlet Reference (V4).

Herramientas para la versión 5 PowerShell

Ejemplo 1: En este ejemplo se muestra la publicación de un mensaje con una sola MessageAttribute línea declarada.

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

Ejemplo 2: En este ejemplo se muestra la publicación de un mensaje con varios MessageAttributes declarados de antemano.

```
$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 obtener más información sobre la API, consulte Publish in Herramientas de AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información sobre. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Publique un mensaje con atributos para que una suscripción pueda filtrar en función de los 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 un mensaje que toma diferentes formas en función del protocolo del suscriptor.

```
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 obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener información sobre la API, consulte Publicar en la Referencia de la API de AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;
```

```
    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta Publicar en el AWS SDK para obtener información sobre la API de Swift.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`SetSMSAttributes`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cómo utilizar Amazon SNS para establecer el atributo predeterminadoSMSType .

```
//! 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 obtener más información sobre la API, consulte Establecer SMSAttributes en la referencia de AWS SDK para C++ la API.

CLI

AWS CLI

Establecimiento de los atributos de los mensajes SMS

En el siguiente ejemplo de `set-sms-attributes`, se establece el ID de remitente predeterminado para los mensajes SMS a `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte Set SMSAttributes in AWS CLI Command Reference.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta Establecer SMSAttributes en la referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta Establecer SMSAttributes en la referencia AWS SDK para JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener más información sobre la API, consulta Establecer SMSAttributes en la referencia AWS SDK para PHP de la API.

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **SetSubscriptionAttributes** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SetSubscriptionAttributes`.

CLI

AWS CLI

Establecimiento de los atributos de suscripción

En el siguiente ejemplo de `set-subscription-attributes`, se establece el atributo `RawMessageDelivery` en una suscripción de 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 no genera ninguna salida.

En el siguiente ejemplo de `set-subscription-attributes`, se establece un atributo `FilterPolicy` en una suscripción de 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 no genera ninguna salida.

En el siguiente ejemplo de `set-subscription-attributes`, se elimina el atributo `FilterPolicy` de una suscripción de 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 no genera ninguna salida.

- Para obtener más información sobre la API, consulte SetSubscriptionAttributesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta SetSubscriptionAttributesla Referencia AWS SDK for Java 2.x de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta SetSubscriptionAttributesla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon SNS con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

`SetSubscriptionAttributesRedrivePolicy`Úselo con un AWS SDK

En el siguiente ejemplo de código, se muestra cómo utilizar `SetSubscriptionAttributesRedrivePolicy`.

Java

SDK para Java 1.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`SetTopicAttributes`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SetTopicAttributes`.

CLI

AWS CLI

Establecimiento de un atributo para un tema

En el ejemplo de `set-topic-attributes` siguiente, se establece el atributo `DisplayName` del tema especificado.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte SetTopicAttributesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

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

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
```

```
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

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

- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK para JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Para obtener más información sobre la API, consulta SetTopicAttributesla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK para PHP de la API.

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener más información sobre la API, consulta SetTopicAttributesla Referencia AWS SDK para Ruby de la API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obtener más información sobre la API, consulte SetTopicAttributesla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`Subscribe`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `Subscribe`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- Creación y publicación en un tema FIFO
- Publicación de mensajes en colas

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

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

Suscriba una cola a un tema con filtros opcionales.

```
    /// <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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

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

Suscriba una aplicación móvil a un tema.

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

Suscriba una función de Lambda a un tema.

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

Suscriba una cola de SQS a un tema.

```
        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;
           }
```

Suscribirse con un filtro a un tema.

```
       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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para C++ .

CLI

AWS CLI

Suscripción a un tema

El siguiente comando `subscribe` suscribe una dirección de correo electrónico al tema especificado.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Salida:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Para obtener detalles sobre la API, consulte Subscribe en la Referencia de comandos de la AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una cola a un tema con filtros opcionales.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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);
        }
    }
}
```

Suscriba un punto de conexión HTTP a un tema.

```
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);
        }
    }
}
```

Suscriba una función de Lambda a un tema.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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'
  // }
};
```

Suscriba una aplicación móvil a un tema.

```
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;
};
```

Suscriba una función de Lambda a un tema.

```
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;
};
```

Suscriba una cola de SQS a un tema.

```
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;
};
```

Suscribirse con un filtro a un tema.

```
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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.
- Para obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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()
    }
}
```

Suscriba una función de Lambda a un tema.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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());
}
```

Suscriba un punto de conexión HTTP a un tema.

```
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 obtener información sobre la API, consulte Suscríbase en la Referencia de la API de AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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 obtener información sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para Ruby.
- Para obtener información sobre la API, consulte Suscríbase en la Referencia de la API de AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
   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 obtener detalles sobre la API, consulte Subscribe en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
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.")
```

Suscribe un número de teléfono a un tema para recibir notificaciones 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 obtener más información sobre la API, consulta la referencia sobre la API de Suscríbete en el AWS SDK para Swift.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`TagResource`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `TagResource`.

CLI

AWS CLI

Añadir una etiqueta a un tema

El siguiente ejemplo de `tag-resource` añade una etiqueta de metadatos al tema de Amazon SNS especificado.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
```

```
    --tags Key=Team,Value=Alpha
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte TagResourcela Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta TagResourcela Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta TagResourcela referencia sobre el AWS SDK para la API de Kotlin.

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`Unsubscribe`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `Unsubscribe`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Publicación de mensajes en colas

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Darse de baja de un tema mediante un ARN de suscripción

```
    /// <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 obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);
```

```
    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para C++ .

CLI

AWS CLI

Cancelación de la suscripción a un tema

En el siguiente ejemplo de `unsubscribe`, se elimina la suscripción especificada de un tema.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Este comando no genera ninguna salida.

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de comandos de la AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

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

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

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

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

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

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurarlo y ejecutarlo en el Repositorio de ejemplos de código de AWS.

Cree el cliente en un módulo separado y expórtelo.

```
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 el SDK y los módulos cliente, y llame a la 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para JavaScript.

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener información sobre la API, consulte Cancelar suscripción en la Referencia de la API de AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- Para obtener detalles sobre la API, consulte Unsubscribe en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener más información sobre la API, consulta la sección Cancelar suscripción en el AWS SDK para ver la referencia sobre la API de Swift.

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Escenarios de uso de Amazon SNS AWS SDKs

Los siguientes ejemplos de código muestran cómo implementar escenarios comunes en Amazon SNS con. AWS SDKs Estos escenarios muestran cómo llevar a cabo tareas específicas con llamadas a varias funciones dentro de Amazon SNS o en combinación con otros Servicios de AWS. En cada escenario se incluye un enlace al código fuente completo, con instrucciones de configuración y ejecución del código.

Los escenarios requieren un nivel intermedio de experiencia para ayudarlo a entender las acciones de servicio en su contexto.

Ejemplos

- Creación de una aplicación para enviar datos a una tabla de DynamoDB
- Creación de una aplicación de publicación y suscripción que traduzca mensajes
- Cree un punto final de plataforma para las notificaciones push de Amazon SNS mediante un SDK AWS
- Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas
- Creación de una aplicación de exploración de Amazon Textract
- Cree y publique en un tema FIFO de Amazon SNS mediante un SDK AWS
- Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK

- Publicar mensajes SMS en un tema de Amazon SNS mediante un SDK AWS
- Publique un mensaje grande en Amazon SNS con Amazon S3 mediante un SDK AWS
- Publicar un mensaje de texto SMS de Amazon SNS mediante un SDK AWS
- Publique mensajes de Amazon SNS en las colas de Amazon SQS mediante un SDK AWS
- Uso de API Gateway para invocar una función de Lambda
- Uso de eventos programados para invocar una función de Lambda

Creación de una aplicación para enviar datos a una tabla de DynamoDB

Los siguientes ejemplos de código indican cómo crear una aplicación que envíe datos a una tabla de Amazon DynamoDB y que le notifique cuando un usuario actualice la tabla.

Java

SDK para Java 2.x

Indica cómo crear una aplicación web dinámica que envíe datos mediante la API Java de Amazon DynamoDB y un mensaje de texto mediante la API Java de Amazon Simple Notification Service.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Este ejemplo indica cómo crear una aplicación que permita a los usuarios enviar datos a una tabla de Amazon DynamoDB y un mensaje de texto al administrador mediante Amazon Simple Notification Service (Amazon SNS).

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Este ejemplo también está disponible en la Guía para desarrolladores de AWS SDK para JavaScript v3.

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon SNS

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Creación de una aplicación de publicación y suscripción que traduzca mensajes

Los siguientes ejemplos de código muestran cómo crear una aplicación que cuente con la funcionalidad de suscripción y publicación y traduzca mensajes.

.NET

SDK para .NET

Indica cómo utilizar la API .NET de Amazon Simple Notification Service para crear una aplicación web con la funcionalidad de suscripción y publicación. Además, esta aplicación de ejemplo también traduce los mensajes.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon SNS
- Amazon Translate

Java

SDK para Java 2.x

Indica cómo utilizar la API de Java de Amazon Simple Notification Service para crear una aplicación web con la funcionalidad de suscripción y publicación. Además, esta aplicación de ejemplo también traduce los mensajes.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar y ejecutar el ejemplo que usa la API Java Async, consulta el ejemplo completo en. GitHub

Servicios utilizados en este ejemplo

- Amazon SNS
- Amazon Translate

Kotlin

SDK para Kotlin

Muestra cómo utilizar la API de Kotlin de Amazon SNS para crear una aplicación que cuente con la funcionalidad de suscripción y publicación. Además, esta aplicación de ejemplo también traduce los mensajes.

Para ver el código fuente completo y las instrucciones sobre cómo crear una aplicación web, consulta el ejemplo completo en. GitHub

Para obtener el código fuente completo y las instrucciones sobre cómo crear una aplicación Android nativa, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon SNS
- Amazon Translate

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cree un punto final de plataforma para las notificaciones push de Amazon SNS mediante un SDK AWS

Los siguientes ejemplos de código muestran cómo crear un punto de conexión de la plataforma para las notificaciones push de Amazon SNS.

CLI

AWS CLI

Creación de un punto de conexión de aplicación de plataforma

En el siguiente ejemplo de `create-platform-endpoint`, se crea un punto de conexión para la aplicación de plataforma especificada mediante el token especificado.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Salida:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas

En los siguientes ejemplos de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

.NET

SDK para .NET

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Creación de una aplicación de exploración de Amazon Textract

Los siguientes ejemplos de código indican cómo explorar la salida de Amazon Textract mediante una aplicación interactiva.

JavaScript

SDK para JavaScript (v3)

Muestra cómo utilizarla AWS SDK para JavaScript para crear una aplicación de React que utilice Amazon Textract para extraer datos de la imagen de un documento y mostrarlos en una página web interactiva. Este ejemplo se ejecuta en un navegador web y requiere una identidad autenticada de Amazon Cognito para las credenciales. Para el almacenamiento utiliza Amazon Simple Storage Service (Amazon S3) y para las notificaciones consulta una cola de Amazon Simple Queue Service (Amazon SQS) que está suscrita a un tema de Amazon Simple Notification Service (Amazon SNS).

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Muestra cómo utilizar Amazon Textract para detectar elementos de texto, formulario y tabla en una imagen de documento. AWS SDK para Python (Boto3) La imagen de entrada y la salida de Amazon Textract aparecen en una aplicación Tkinter que permite explorar los elementos detectados.

- Envía la imagen de un documento a Amazon Textract y explora el resultado de los elementos detectados.
- Envía imágenes directamente a Amazon Textract o mediante un bucket de Amazon Simple Storage Service (Amazon S3).
- Utilice la función asincrónica APIs para iniciar un trabajo que publique una notificación en un tema del Amazon Simple Notification Service (Amazon SNS) cuando se complete el trabajo.
- Consulta una cola de Amazon Simple Queue Service (Amazon SQS) en busca de un mensaje de finalización de trabajo y muestra los resultados.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en. GitHub

Servicios utilizados en este ejemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cree y publique en un tema FIFO de Amazon SNS mediante un SDK AWS

Los siguientes ejemplos de código muestran cómo crear y publicar en un tema FIFO de Amazon SNS.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este ejemplo

- crea un tema FIFO de Amazon SNS, dos colas FIFO de Amazon SQS y una cola estándar.
- suscribe las colas al tema y publica un mensaje en el tema.

La prueba verifica la recepción del mensaje en cada cola. El ejemplo completo también muestra la incorporación de políticas de acceso y, al final, elimina los recursos.

```
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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
  - CreateTopic
  - Publish
  - Subscribe

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crea un tema FIFO de Amazon SNS, suscribe una cola FIFO de Amazon SQS al tema y publica un mensaje en el tema.

```
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 obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
  - CreateTopic
  - Publish
  - Subscribe

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crea un tema de FIFO, suscribe una cola FIFO de Amazon SQS al tema y publica un mensaje en un tema de 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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para SAP ABAP.
  - CreateTopic
  - Publish
  - Subscribe

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK

Los siguientes ejemplos de código indican cómo detectar personas y objetos en un video con Amazon Rekognition.

Java

SDK para Java 2.x

Muestra cómo utilizar la API Java de Amazon Rekognition para crear una aplicación que detecte rostros y objetos en vídeos ubicados en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Utilice Amazon Rekognition para detectar caras, objetos y personas en videos iniciando trabajos de detección asíncronos. Este ejemplo también configura Amazon Rekognition para que notifique un tema de Amazon Simple Notification Service (Amazon SNS) cuando se finalicen los trabajos y suscriba una cola de Amazon Simple Queue Service (Amazon SQS) al tema. Cuando la cola recibe un mensaje sobre un trabajo, se recupera el trabajo y se muestran los resultados

Este ejemplo se ve mejor en GitHub. Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Publicar mensajes SMS en un tema de Amazon SNS mediante un SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema de Amazon SNS
- Suscribir números de teléfono al tema
- Publique mensajes SMS en el tema para que todos los números de teléfono suscritos reciban el mensaje a la vez.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree un tema y devuelva su 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 "";
    }
}
```

Suscriba un punto de enlace a un tema.

```
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);
        }
    }
}
```

Establezca atributos en el mensaje, como el ID del remitente, el precio máximo y su tipo. Los atributos de mensaje son opcionales.

```
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 un mensaje en un tema. El mensaje se envía a cada suscriptor.

```
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 obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Publique un mensaje grande en Amazon SNS con Amazon S3 mediante un SDK AWS

El siguiente ejemplo de código muestra cómo publicar un mensaje de gran tamaño en Amazon SNS utilizando Amazon S3 para almacenar la carga útil del mensaje.

Java

SDK para Java 1.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Para publicar mensajes grandes, utilice la biblioteca de clientes extendidos de Amazon SNS para Java. El mensaje que envía hace referencia a un objeto de Amazon S3 en el que se incluye el contenido real del mensaje.

```
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 obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Publicar un mensaje de texto SMS de Amazon SNS mediante un SDK AWS

En los siguientes ejemplos de código, se muestra cómo publicar mensajes SMS mediante Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
```

```
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

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

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

    return outcome.IsSuccess();
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para C++ .

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
```

```
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
```

```
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

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

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

```
    }
}
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de 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 obtener más información, consulte la Guía para desarrolladores de AWS SDK para PHP.
- Para obtener información sobre la API, consulte Publicar en la Referencia de la API de AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
```

```
        subscription.

        :param phone_number: The phone number that receives the message. This
must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Para obtener detalles sobre la API, consulte Publish en la Referencia de la API de AWS SDK para Python (Boto3).

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Publique mensajes de Amazon SNS en las colas de Amazon SQS mediante un SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema (FIFO o no FIFO)
- Suscribir varias colas al tema con la opción de aplicar un filtro
- Publicar mensajes en el tema
- Sondear las colas en busca de los mensajes recibidos

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario interactivo en un símbolo del sistema

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

Crear una clase que encapsule las operaciones de 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;
```

```
    }
}
```

Crear una clase que encapsule las operaciones de 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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

C++

SDK para C++

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
```

```
   printAsterisksLine();
   std::cout << "In this workflow, you will create an SNS topic and subscribe "
             << NUMBER_OF_QUEUES <<
             " SQS queues to the topic." << std::endl;
   std::cout
           << "You can select from several options for configuring the topic and
the subscriptions for the "
           << NUMBER_OF_QUEUES << " queues." << std::endl;
   std::cout << "You can then post to the topic and see the results in the
queues."
             << std::endl;

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

   printAsterisksLine();

   std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
             << std::endl;
   std::cout
           << "FIFO topics deliver messages in order and support deduplication
and message filtering."
           << std::endl;
   bool isFifoTopic = askYesNoQuestion(
           "Would you like to work with FIFO topics? (y/n) ");

   bool contentBasedDeduplication = false;
   Aws::String topicName;
   if (isFifoTopic) {
       printAsterisksLine();
       std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                 << std::endl;
       std::cout
               << "Deduplication IDs are either set in the message or
automatically generated "
               << "from content using a hash function." << std::endl;
       std::cout
               << "If a message is successfully published to an SNS FIFO topic,
any message "
               << "published and determined to have the same deduplication ID, "
               << std::endl;
       std::cout
               << "within the five-minute deduplication interval, is accepted
but not delivered."
```

```
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

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

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
```

```
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
```

```
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
```

```
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
```

```
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
```

```
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;
```

```
                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
```

```
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

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

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
```

```
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
```

```
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);
```

```
                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

```
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

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

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publish
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Go

SDK para Go V2

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario interactivo en un símbolo del sistema.

```
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 una estructura que encapsule las acciones de Amazon SNS utilizadas en este ejemplo.

```
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 una estructura que encapsule las acciones de Amazon SQS utilizadas en este ejemplo.

```
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
}
```

Eliminación de 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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
```

```
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Publish
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este es el punto de partida para este escenario.

```
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();
};
```

El código anterior proporciona las dependencias necesarias e inicia el escenario. La siguiente sección contiene la mayor parte del ejemplo.

```
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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package com.example.sns
```

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this 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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Swift

SDK para Swift

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de API Gateway para invocar una función de Lambda

Los siguientes ejemplos de código muestran cómo crear una AWS Lambda función invocada por Amazon API Gateway.

Java

SDK para Java 2.x

Muestra cómo crear una AWS Lambda función mediante la API de tiempo de ejecución Lambda Java. En este ejemplo, se invocan diferentes AWS servicios para realizar un caso de uso específico. En este ejemplo se indica cómo crear una función de Lambda invocada por Amazon API Gateway que escanea una tabla de Amazon DynamoDB en busca de aniversarios laborales y utiliza Amazon Simple Notification Service (Amazon SNS) para enviar un mensaje de texto a sus empleados que les felicite en la fecha de su primer aniversario.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Muestra cómo crear una AWS Lambda función mediante la API de tiempo de JavaScript ejecución de Lambda. En este ejemplo, se invocan diferentes AWS servicios para realizar un caso de uso específico. En este ejemplo se indica cómo crear una función de Lambda invocada por Amazon API Gateway que escanea una tabla de Amazon DynamoDB en busca de aniversarios laborales y utiliza Amazon Simple Notification Service (Amazon SNS)

para enviar un mensaje de texto a sus empleados que les felicite en la fecha de su primer aniversario.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Este ejemplo también está disponible en la Guía para desarrolladores de AWS SDK para JavaScript v3.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

En este ejemplo se muestra cómo crear y utilizar una API de REST de Amazon API Gateway dirigida a una función AWS Lambda . El controlador Lambda muestra cómo enrutar según los métodos HTTP; cómo obtener datos de la cadena de consulta, el encabezado y el cuerpo; y cómo devolver una respuesta JSON.

- Implemente una función de Lambda.
- Cree una API de REST mediante API Gateway.
- Cree un recurso REST que se dirija a la función de Lambda.
- Otorgue permiso para permitir que API Gateway invoque la función de Lambda.
- Utilice el paquete Requests para enviar solicitudes a la API de REST.
- Limpie todos los recursos creados durante la demostración.

Este ejemplo se ve mejor en GitHub. Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon SNS

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de eventos programados para invocar una función de Lambda

Los siguientes ejemplos de código muestran cómo crear una AWS Lambda función invocada por un evento EventBridge programado de Amazon.

Java

SDK para Java 2.x

Muestra cómo crear un evento EventBridge programado de Amazon que invoque una AWS Lambda función. Configure EventBridge para usar una expresión cron para programar cuándo se invoca la función Lambda. En este ejemplo, creará una función de Lambda utilizando la API de tiempo de ejecución de Java de Lambda. En este ejemplo, se invocan diferentes AWS servicios para realizar un caso de uso específico. Este ejemplo indica cómo crear una aplicación que envíe un mensaje de texto a sus empleados para felicitarles por su primer aniversario.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Muestra cómo crear un evento EventBridge programado de Amazon que invoque una AWS Lambda función. Configure EventBridge para usar una expresión cron para programar cuándo se invoca la función Lambda. En este ejemplo, se crea una función de Lambda mediante la API de tiempo de ejecución de JavaScript Lambda. En este ejemplo, se invocan diferentes AWS servicios para realizar un caso de uso específico. Este ejemplo indica cómo crear una aplicación que envíe un mensaje de texto a sus empleados para felicitarles por su primer aniversario.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Este ejemplo también está disponible en la Guía para desarrolladores de AWS SDK para JavaScript v3.

Servicios utilizados en este ejemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

En este ejemplo se muestra cómo registrar una AWS Lambda función como destino de un EventBridge evento programado de Amazon. El controlador Lambda escribe un mensaje descriptivo y los datos completos del evento en Amazon CloudWatch Logs para su posterior recuperación.

- Implementa una función de Lambda.
- Crea un evento EventBridge programado y convierte la función Lambda en el objetivo.
- Otorga permiso para EventBridge invocar la función Lambda.

- Imprime los datos más recientes de CloudWatch los registros para mostrar el resultado de las invocaciones programadas.
- Limpia todos los recursos creados durante la demostración.

Es mejor ver este ejemplo en GitHub. Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de sistemas sin servidor para Amazon SNS

Los siguientes ejemplos de código muestran cómo utilizar Amazon SNS con. AWS SDKs

Ejemplos

- Invocación de una función de Lambda desde un desencadenador de Amazon SNS

Invocación de una función de Lambda desde un desencadenador de Amazon SNS

En los siguientes ejemplos de código, se muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante .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**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante 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**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante 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**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Consumir un evento de SNS con JavaScript Lambda mediante.

```
// 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;
  }
}
```

Consumir un evento de SNS con TypeScript Lambda mediante.

```
// 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**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante 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**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante 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**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante 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**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos de tecnología sin servidor.

Uso de un evento de SNS con Lambda mediante 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 obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon SNS con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Seguridad en Amazon SNS

El modelo de se aplica a protección de datos en Amazon Simple Notification Service. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. Este contenido incluye las tareas de configuración y administración de la seguridad de AWS los servicios que utiliza. Para obtener más información sobre la privacidad de los datos, consulta las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulta la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure cuentas de usuario individuales con AWS Identity and Access Management (IAM). De esta manera, cada usuario recibe únicamente los permisos necesarios para cumplir con sus obligaciones laborales. También recomendamos proteger sus datos de las siguientes maneras:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Recomendamos TLS 1.2 o una versión posterior.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados de AWS los servicios.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.
- Protección de datos de mensajes
  - La protección de datos de mensajes es una nueva función importante de Amazon SNS
  - Utilice MDP para analizar el mensaje en busca de información confidencial o sensible
  - Realice una auditoría de mensajes en todo el contenido que pasa por el tema
  - Proporcione controles de acceso al contenido para los mensajes publicados en el tema y los mensajes que entrega el tema

> **Important**
>
> Recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, direcciones de email de sus clientes, en etiquetas o en los campos de formato libre, como el campo Name (Nombre). Esto incluye cuando trabaja con Amazon SNS u otros Amazon Web Services mediante la consola, la API o. AWS CLI AWS SDKs Los datos que ingresa en etiquetas o campos de formato libre utilizados para los nombres se pueden utilizar para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, le recomendamos encarecidamente que no incluya información de credenciales en la URL para validar la solicitud para ese servidor.

Cifrado de datos de Amazon SNS

La protección de datos se refiere a salvaguardarlos en tránsito (al desplazarse desde y hacia Amazon SNS) y en reposo (almacenado en discos en centros de datos Amazon SNS). Puede proteger los datos en tránsito con una Capa de sockets seguros (SSL, Secure Sockets Layer) o con el cifrado del lado del cliente. De forma predeterminada, Amazon SNS almacena los mensajes y archivos mediante el cifrado de disco. Puede proteger los datos en reposo solicitando a Amazon SNS que cifre los mensajes antes de guardarlos en el sistema de archivos cifrados en sus centros de datos. Amazon SNS recomienda usar SSE para optimizar el cifrado de datos.

Protección de los datos de Amazon SNS con cifrado del servidor

El cifrado del lado del servidor (SSE) le permite almacenar datos confidenciales en temas cifrados al proteger el contenido de los mensajes en los temas de Amazon SNS mediante claves AWS Key Management Service administradas en ().AWS KMS

SSE cifra los mensajes en cuanto Amazon SNS los recibe. Los mensajes se almacenan cifrados y solo se descifran cuando se envían.

- Para obtener información sobre cómo administrar el SSE mediante AWS Management Console o la AWS SDK para Java (configurando el `KmsMasterKeyId` atributo mediante las acciones de la `SetTopicAttributes` API `CreateTopic` y), consulte. Configuración del cifrado de temas de Amazon SNS con cifrado del servidor
- Para obtener información sobre cómo crear temas cifrados mediante AWS CloudFormation (configurando la `KmsMasterKeyId` propiedad con el `AWS::SNS::Topic` recurso), consulte la Guía del AWS CloudFormation usuario.

> **Important**
>
> Todas las solicitudes hechas a los temas con SSE habilitado deben usar HTTPS y Signature Version 4.
> Para obtener información acerca de la compatibilidad de otros servicios con temas de cifrado, consulte la documentación de los servicios.
> Amazon SNS solo admite claves de KMS de cifrado simétricas. No puede utilizar ningún otro tipo de clave de KMS para cifrar los recursos del servicio. Para obtener ayuda para determinar si una clave de KMS es una clave de cifrado simétrica, consulte Identificar claves de KMS asimétricas.

AWS KMS combina hardware y software seguros y de alta disponibilidad para proporcionar un sistema de administración de claves adaptado a la nube. Cuando utiliza Amazon SNS con AWS KMS, las claves de datos que cifran los datos de sus mensajes también se cifran y almacenan con los datos que protegen.

A continuación, se describen los beneficios de usar AWS KMS:

- Puede crear y administrar la AWS KMS key usted mismo.
- También puede usar claves de KMS AWS administradas para Amazon SNS, que son únicas para cada cuenta y región.
- Los estándares AWS KMS de seguridad pueden ayudarle a cumplir los requisitos de conformidad relacionados con el cifrado.

Para obtener más información, consulte ¿Qué es? AWS Key Management Service en la Guía para AWS Key Management Service desarrolladores.

Ámbito de cifrado

SSE cifra el cuerpo de un mensaje en un tema de Amazon SNS.

SSE no cifra lo siguiente:

- Metadatos del tema (atributos y nombre del tema)
- Metadatos del mensaje (asunto, ID de mensaje, marca temporal y atributos)
- Política de protección de datos

- Métricas por temas

> **Note**
>
> - Un mensaje se cifra únicamente si se envía con posterioridad a la habilitación del cifrado de un tema. Amazon SNS no cifra mensajes atrasados.
> - Cualquier mensaje cifrado permanece en dicho estado aunque se deshabilite el cifrado de su tema.

Términos clave

Los siguientes términos clave pueden ayudarle a comprender mejor la funcionalidad de SSE. Para obtener descripciones detalladas, consulte la Referencia de la API de Amazon Simple Notification Service.

Clave de datos

La clave de cifrado de datos (DEK) es responsable de cifrar el contenido de los mensajes de Amazon SNS.

Para obtener más información, consulte Claves de datos en la Guía para desarrolladores de AWS Key Management Service y Cifrado de sobre en la Guía para desarrolladores de AWS Encryption SDK .

AWS KMS key ID

El alias, el alias ARN, el identificador clave o el ARN clave de una cuenta o de una AWS KMS key cuenta personalizada de su cuenta o AWS KMS de otra cuenta. Si bien el alias del SNS AWS administrado AWS KMS para Amazon SNS es siempre`alias/aws/sns`, el alias de un cliente AWS KMS puede, por ejemplo, serlo. `alias/`*`MyAlias`* Puede utilizar estas claves AWS KMS para proteger los mensajes que se encuentran en los temas de Amazon SNS.

> **Note**
>
> Tenga en cuenta lo siguiente:
>
> - La primera vez que utilice AWS Management Console para especificar el KMS AWS administrado para Amazon SNS para un tema, AWS KMS crea el KMS AWS administrado para Amazon SNS.

- Como alternativa, la primera vez que utilice la `Publish` acción en un tema con SSE habilitado, AWS KMS creará el KMS AWS administrado para Amazon SNS.

Puede crear AWS KMS claves, definir las políticas que controlan el uso de AWS KMS las claves y auditar el AWS KMS uso mediante la AWS KMS keyssección de la AWS KMS consola o la `CreateKey` AWS KMS acción. Para obtener más información, consulte AWS KMS keys y Creación de claves en la Guía para desarrolladores de AWS Key Management Service . Para ver más ejemplos de AWS KMS identificadores, consulta la referencia KeyIdde la AWS Key Management Service API. Para obtener información sobre cómo buscar AWS KMS identificadores, consulte Buscar el ID de clave y el ARN en AWS Key Management Service la Guía para desarrolladores.

Important

Su uso conlleva cargos adicionales. AWS KMS Para obtener más información, consulte Estimación de costos AWS KMS y Precios de AWS Key Management Service.

Administración de las claves de cifrado y los costos de Amazon SNS

En las siguientes secciones se proporciona información sobre cómo trabajar con claves administradas en AWS Key Management Service (AWS KMS).

Note

Amazon SNS solo admite claves de KMS de cifrado simétricas. No puede utilizar ningún otro tipo de clave de KMS para cifrar los recursos del servicio. Para obtener ayuda para determinar si una clave de KMS es una clave de cifrado simétrica, consulte Identificar claves de KMS asimétricas.

Estimación de costos AWS KMS

Para predecir los costes y entender mejor tu AWS factura, quizá te interese saber con qué frecuencia Amazon SNS utiliza tu factura. AWS KMS key

> **Note**
>
> Si bien la siguiente fórmula puede brindarle una muy buena idea de los costos esperados, los costos reales podrían ser más elevados debido a la naturaleza distribuida de Amazon SNS.

Para calcular el número de solicitudes de la API (R) por tema, utilice la siguiente fórmula:

```
R = B / D * (2 * P)
```

B es el período de facturación (en segundos).

D es el período de reutilización de claves de datos (en segundos, Amazon SNS reutiliza una clave de datos durante un máximo de 5 minutos).

P es el número de entidades principales de publicación que realizan envíos al tema de Amazon SNS.

A continuación se muestran algunos cálculos de ejemplo. Para obtener información exacta sobre precios, consulte Precios de AWS Key Management Service.

Ejemplo 1: Calcular el número de llamadas a la AWS KMS API para un editor y un tema

En este ejemplo se presupone lo siguiente:

- El período de facturación va del 1 al 31 de enero (2 678 400 segundos).
- El periodo de reutilización de la clave de datos es de 5 minutos (300 segundos).
- Hay 1 tema.
- Hay una 1 entidad principal de publicación.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Ejemplo 2: Cálculo del número de llamadas a la API de AWS KMS con varios publicadores y 2 temas

En este ejemplo se presupone lo siguiente:

- El período de facturación va del 1 al 28 de febrero (2 419 200 segundos).
- El periodo de reutilización de la clave de datos es de 5 minutos (300 segundos).
- Hay 2 temas.

- El primer tema tiene 3 entidades principales de publicación.
- El segundo tema tiene 5 entidades principales de publicación.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Configuración de AWS KMS permisos

Antes de poder utilizar SSE, debe configurar AWS KMS key políticas que permitan el cifrado de temas y el cifrado y descifrado de mensajes. Para obtener ejemplos y más información sobre los permisos de AWS KMS , consulte Permisos de API de AWS KMS : Referencia de recursos y acciones en la Guía para desarrolladores de AWS Key Management Service . Para obtener más información sobre cómo configurar un tema de Amazon SNS con cifrado del servidor, consulte Información adicional.

> **Note**
>
> También puede administrar los permisos para las claves de KMS de cifrado simétrico mediante políticas de IAM. Para obtener más información, consulte Uso de políticas de IAM con. AWS KMS
> Si bien puede configurar los permisos globales para enviar y recibir desde Amazon SNS, es AWS KMS necesario mencionar explícitamente el ARN completo de regiones específicas KMSs en la `Resource` sección de una política de IAM.

También debe asegurarse de que las políticas clave del AWS KMS key permiten los permisos necesarios. Para ello, asigne un nombre a las principales que producen y consumen mensajes cifrados en Amazon SNS como usuarios de la política de claves de KMS.

Como alternativa, puede especificar AWS KMS las acciones necesarias y el ARN de KMS en una política de IAM asignada a las entidades principales que publican y se suscriben para recibir mensajes cifrados en Amazon SNS. Para obtener más información, consulte Administración del acceso a AWS KMS en la Guía para desarrolladores de AWS Key Management Service .

Si selecciona una clave administrada por el cliente para el tema de Amazon SNS y utiliza alias para controlar el acceso a las claves KMS mediante políticas de IAM o políticas de claves de KMS con la clave de condición `kms:ResourceAliases`, asegúrese de que la clave administrada por el cliente seleccionada también tenga un alias asociado. Para obtener más información sobre el uso de alias

para controlar el acceso a las claves KMS, consulte Uso de alias para controlar el acceso a las claves KMS en la Guía para desarrolladores de AWS Key Management Service .

Permitir que un usuario envíe mensajes a un tema con SSE

El publicador debe tener los permisos `kms:GenerateDataKey*` y `kms:Decrypt` para 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 la compatibilidad entre las fuentes de eventos de AWS los servicios y los temas cifrados

Varios AWS servicios publican eventos sobre temas de Amazon SNS. Para que estos orígenes de eventos funcionen con los temas cifrados, es preciso llevar a cabo los pasos que se describen a continuación:

1. Utilice una clave administrada por el cliente. Para obtener más información, consulte Creación de claves en la Guía para desarrolladores de AWS Key Management Service .
2. Para permitir que el AWS servicio tenga los `kms:Decrypt` permisos `kms:GenerateDataKey*` y, añada la siguiente declaración a la política de KMS.

   ```
   {
     "Statement": [{
       "Effect": "Allow",
       "Principal": {
         "Service": "service.amazonaws.com"
   ```

```
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Origen del evento	Entidad principal de servicio
Amazon CloudWatch	`cloudwatch.amazonaws.com`
CloudWatch Eventos de 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`
AWS Gestor de Sistemas Gestor de Incidentes	AWS Systems Manager Incident Manager consta de dos principios de servicio: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Algunas fuentes de eventos de Amazon SNS requieren que proporciones un rol de IAM (en lugar del principal del servicio) en la política: AWS KMS key
>
> - Amazon EC2 Auto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. Agregue las claves de condición `aws:SourceAccount` y `aws:SourceArn` a la política de recursos de KMS para proteger aún más la clave de KMS de los ataques de suplente confuso. Consulte la lista de documentación específica del servicio (arriba) para obtener detalles exactos de cada caso.

> **Important**
>
> Los EventBridge-to-encrypted temas no admiten la `aws:SourceAccount` adición de los caracteres`aws:SourceArn`, y `aws:SourceOrgID` a una AWS KMS política.

```
{
  "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 SSE para el tema mediante la KMS.
5. Proporcione el ARN del tema cifrado al origen de eventos.

AWS KMS errores

Cuando trabaja con Amazon SNS AWS KMS, es posible que se produzcan errores. En la siguiente lista se describen los errores y sus posibles soluciones.

KMSAccessDeniedException

El texto cifrado hace referencia a una clave que no existe o a la que no tiene acceso.

Código de estado HTTP: 400

KMSDisabledExcepción

La solicitud se rechazó porque la KMS especificada no está habilitada.

Código de estado HTTP: 400

KMSInvalidStateException

La solicitud se rechazó porque el estado del recurso especificado no es válido para esta solicitud. Para obtener más información, consulte Estados de clave de AWS KMS keys en la Guía para desarrolladores de AWS Key Management Service .

Código de estado HTTP: 400

KMSNotFoundException

La solicitud se rechazó porque la entidad o el recurso especificado no se encontraron.

Código de estado HTTP: 400

KMSOptInRequired

El identificador de clave de AWS acceso necesita una suscripción al servicio.

Código de estado HTTP: 403

KMSThrottlingExcepción

La solicitud fue denegada debido a una limitación de la solicitud. Para obtener más información sobre la limitación, consulte Cuotas en la Guía para desarrolladores de AWS Key Management Service .

Código de estado HTTP: 400

Configuración del cifrado de temas de Amazon SNS con cifrado del servidor

Amazon SNS admite el cifrado del lado del servidor (SSE) para proteger el contenido de los mensajes mediante (). AWS Key Management Service AWS KMS Siga las instrucciones que aparecen a continuación para habilitar SSE mediante la consola Amazon SNS o la CDK.

Opción 1: Habilite el cifrado mediante AWS Management Console

1. Inicie sesión en la consola de Amazon SNS.
2. Navegue a la página de temas, seleccione su tema y elija Editar.
3. Expanda la sección Cifrado y haga lo siguiente:
   - Active el cifrado para activarlo.
   - Seleccione el gestionado AWS SNS Clave (alias/aws/sns) como clave de cifrado. Está seleccionada de manera predeterminada.
4. Elija Guardar cambios.

> **Note**
>
> - Clave administrada de AWS Se crea automáticamente si aún no existe.
> - Si no ve la clave o no tiene permisos suficientes, pida al administrador que le pida `kms:ListAliases` y`kms:DescribeKey`.

Opción 2: habilitar el cifrado mediante AWS CDK

Para utilizar el gestionado AWS SNS clave en su aplicación CDK, añada el siguiente fragmento:

```
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();
    }
}
```

Información adicional

- Clave KMS personalizada: puede especificar una clave personalizada si es necesario. En la consola de Amazon SNS, seleccione su clave de KMS personalizada de la lista o introduzca el ARN.
- Permisos para claves de KMS personalizadas: si utiliza una clave de KMS personalizada, incluya lo siguiente en la política de claves para que Amazon SNS pueda cifrar y descifrar los mensajes:

```
{
    "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 en los consumidores

Habilitar el SSE no cambia la forma en que los suscriptores consumen los mensajes. AWS gestiona el cifrado y el descifrado de forma transparente. Los mensajes permanecen cifrados en reposo y se descifran automáticamente antes de entregarlos a los suscriptores. Para una seguridad óptima, AWS recomienda habilitar HTTPS en todos los puntos finales para garantizar la transmisión segura de los mensajes.

Configuración del cifrado de temas de Amazon SNS con una suscripción a una cola cifrada de Amazon SQS

Puede habilitar el cifrado del lado del servidor (SSE) para un tema con el fin de proteger sus datos. Para permitir que Amazon SNS envíe mensajes a colas de Amazon SQS cifradas, la clave administrada por el cliente asociada a la cola de Amazon SQS debe tener una instrucción de política que conceda a la entidad principal de servicio de Amazon SNS acceso a las acciones de la API de AWS KMS `GenerateDataKey` y `Decrypt`. Para obtener más información acerca del uso de SSE, consulte Protección de los datos de Amazon SNS con cifrado del servidor.

En este tema se explica cómo habilitar SSE para un tema de Amazon SNS con una suscripción de cola de Amazon SQS cifrada mediante el. AWS Management Console

Paso 1: crear una clave de KMS personalizada

1. Inicie sesión en la consola de AWS KMS con un usuario que tenga al menos la política `AWSKeyManagementServicePowerUser`.
2. Elija Create a key (Crear clave).
3. Para crear una clave KMS de cifrado simétrica, para Key type (Tipo de clave) seleccione Symmetric (Simétrica).

   Para obtener información acerca de cómo crear una clave de KMS asimétrica en la consola de AWS KMS , consulte Creación de claves de KMS asimétricas (consola).

4. En Key usage (Uso de claves), se selecciona la opción Encrypt and decrypt (Cifrar y descifrar) para usted.

   Para obtener información acerca de cómo crear claves de KMS que generan y verifican códigos MAC, consulte Creación de claves de KMS HMAC.

   Para obtener más información acerca de las Opciones avanzadas, consulte Claves para fines especiales.

5. Elija Next (Siguiente).

6. Escriba un alias para la clave KMS. El nombre del alias no puede empezar por **`aws/`**. Amazon Web Services se reserva el **`aws/`** prefijo para representarlo Claves administradas por AWS en su cuenta.

   > **Note**
   >
   > Agregar, eliminar o actualizar un alias puede permitir o denegar el permiso a la clave KMS. Para obtener más información, consulte ABAC para AWS KMS y Uso de alias para controlar el acceso a las claves de KMS.

   Un alias es un nombre de visualización que puede usar para identificar a una clave KMS. Le recomendamos que elija un alias que indique el tipo de datos que piensa proteger o la aplicación que piensa usar con la clave KMS.

   Los alias son necesarios para crear una clave KMS en la AWS Management Console. Son opcionales cuando se utiliza la CreateKeyoperación.

7. (Opcional) Escriba una descripción de la clave KMS.

   Puede agregar una descripción ahora o actualizarla en cualquier momento, a menos que el estado de la clave sea `Pending Deletion` o `Pending Replica Deletion`. Para añadir, cambiar o eliminar la descripción de una clave gestionada por el cliente existente, edite la descripción en la operación AWS Management Console o utilice la UpdateKeyDescriptionoperación.

8. (Opcional) Escriba una clave de etiqueta y un valor de etiqueta opcional. Para agregar más de una etiqueta a la clave KMS, elija Add tag (Agregar etiqueta).

> **Note**
>
> Etiquetar o quitar las etiquetas de la clave KMS puede permitir o denegar permiso a la clave KMS. Para obtener más información, consulte ABAC para AWS KMS y Uso de etiquetas para controlar el acceso a las claves de KMS.

Al añadir etiquetas a AWS los recursos, AWS genera un informe de asignación de costes con el uso y los costes agregados por etiquetas. Las etiquetas también pueden utilizarse para controlar el acceso a una clave KMS. Para obtener información acerca del etiquetado de claves de KMS, consulte Etiquetado de claves y ABAC para AWS KMS.

9. Elija Next (Siguiente).
10. Seleccione los usuarios y roles de IAM que pueden administrar la clave de KMS.

> **Note**
>
> Esta política clave proporciona el control Cuenta de AWS total de esta clave de KMS. Permite a los administradores de cuentas utilizar las políticas de IAM para dar permiso a otras entidades principales para administrar la clave KMS. Para obtener más detalles, consulte Política de claves predeterminada.
>
> Las prácticas recomendadas de IAM desalientan el uso de usuarios de IAM con credenciales a largo plazo. Siempre que sea posible, utilice los roles de IAM, que proporcionan credenciales temporales. Para obtener más información, consulte Prácticas recomendadas de seguridad de IAM en la Guía del usuario de IAM.

11. (Opcional) Para evitar que los usuarios y los roles de IAM seleccionados eliminen esta clave de KMS, en la sección Eliminación de claves situada en la parte inferior de la página, desactive la casilla Permitir que los administradores de claves eliminen esta clave.
12. Elija Next (Siguiente).
13. Seleccione los usuarios y roles de IAM que pueden usar la clave en operaciones criptográficas. Elija Next (Siguiente).
14. En la página Review and edit key policy (Revisar y editar política de claves), agregue la instrucción siguiente a la política de claves y, a continuación, elija Finish (Finalizar).

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

La nueva clave administrada por el cliente aparece en la lista de claves.

Paso 2: crear un tema de Amazon SNS cifrado

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. Seleccione Crear tema.
4. En la página Create new topic (Crear nuevo tema), en Name (Nombre), escriba un nombre para el tema (por ejemplo, `MyEncryptedTopic`) y, a continuación, elija Create topic (Crear tema).
5. Expanda la sección Cifrado y haga lo siguiente:

   a. Elija Enable server-side encryption (Habilitar cifrado del lado del servidor).

   b. Especifique la clave administrada por el cliente. Para obtener más información, consulte Términos clave.

      Para cada tipo de clave administrada por el cliente, se muestran la Descripción, la Cuenta y el ARN de la clave administrada por el cliente.

      > **Important**
      >
      > Si no es el propietario de la clave administrada por el cliente o si ha iniciado sesión con una cuenta que no tiene los permisos `kms:ListAliases` y `kms:DescribeKey`, no podrá ver la información sobre la clave administrada por el cliente en la consola de Amazon SNS.

> Pida al propietario de la clave administrada por el cliente que le conceda estos permisos. Para obtener más información, consulte Permisos API de AWS KMS : referencia de recursos y acciones en la Guía para desarrolladores de AWS Key Management Service .

c. Para la clave administrada por el cliente, elija la MyCustomKeyque creó anteriormente y, a continuación, elija Habilitar el cifrado del lado del servidor.

6. Elija Guardar cambios.

   El SSE está activado para el tema y se muestra la MyTopicpágina.

   El estado Cifrado del tema, la Cuenta de AWS , la clave administrada por el cliente, el ARN de la clave administrada por el cliente y la Descripción del tema se muestran en la pestaña Cifrado.

El nuevo tema cifrado aparecerá en la lista de temas.

Paso 3: crear colas de Amazon SQS cifradas y suscribirse a ellas

1. Inicie sesión en la consola de Amazon SQS.
2. Elija Create New Queue (Crear nueva cola).
3. En la página Create New Queue (Crear nueva cola), haga lo siguiente:

   a. Escriba un nombre en Queue Name (Nombre de cola) (por ejemplo, `MyEncryptedQueue1`).

   b. Seleccione Standard Queue (Cola estándar) y, a continuación, elija Configure Queue (Configurar cola).

   c. Elija Use SSE (Usar SSE).

   d. Para AWS KMS keyello, elija MyCustomKeyel que creó anteriormente y, a continuación, elija Crear cola.

4. Repita el proceso para crear una segunda cola (por ejemplo, denominada `MyEncryptedQueue2`).

   Las nuevas colas cifradas aparecerán en la lista de colas.

5. En la consola de Amazon SQS, seleccione `MyEncryptedQueue1` y `MyEncryptedQueue2`. A continuación, elija Acciones de colas, Suscribir colas al tema de SNS.

6. En el cuadro de diálogo Suscribirse a un tema, en Elegir un tema, seleccione y MyEncryptedTopic, a continuación, elija Suscribirse.

   Las suscripciones de las colas cifradas al tema cifrado se muestran en el cuadro de diálogo Topic Subscription Result (Resultado de suscripción al tema).

7. Seleccione OK.

Paso 4: publicar un mensaje en el tema cifrado

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. En la lista de temas, selecciona MyEncryptedTopicy, a continuación, selecciona Publicar mensaje.
4. En la página Publish a message (Publicar mensaje) haga lo siguiente:

   a. (Opcional) En la sección Message details (Detalles del mensaje), introduzca el Subject (Asunto) (por ejemplo, `Testing message publishing`).

   b. En la sección Message body (Cuerpo del mensaje), introduzca el cuerpo del mensaje (por ejemplo, `My message body is encrypted at rest.`).

   c. Elija Publish message (Publicar mensaje).

El mensaje se publica en las colas cifradas suscritas.

Paso 5: verificar la entrega de mensajes

1. Inicie sesión en la consola de Amazon SQS.
2. En la lista de colas, selecciona MyEncryptedQueue1 y, a continuación, selecciona Enviar y recibir mensajes.
3. En la página Enviar y recibir mensajes en MyEncryptedQueue 1, selecciona Buscar mensajes.

   Aparecerá el mensaje que envió antes.

4. Elija More Details (Más información) para ver el mensaje.
5. Cuando haya finalizado, elija Close (Cerrar).
6. Repite el proceso para MyEncryptedQueue2.

Protección del tráfico de Amazon SNS con puntos de conexión de VPC

El punto de enlace de Amazon Virtual Private Cloud (Amazon VPC) para Amazon SNS es una entidad lógica dentro de una VPC que permite la conectividad solo a Amazon SNS. La VPC direcciona las solicitudes a Amazon SNS y vuelve a direccionar las respuestas a la VPC. En las siguientes secciones se proporciona información sobre cómo trabajar con puntos de enlace de la VPC y crear políticas de puntos de enlace de la VPC.

Si utiliza Amazon Virtual Private Cloud (Amazon VPC) para alojar sus AWS recursos, puede establecer una conexión privada entre su VPC y Amazon SNS. Con esta conexión, puede publicar mensajes en sus temas de Amazon SNS sin enviarlos a través de la infraestructura pública de Internet.

Amazon VPC es un AWS servicio que puede utilizar para lanzar AWS recursos en una red virtual que usted defina. Con una VPC, puede controlar la configuración de la red, como el rango de direcciones IP, las subredes, las tablas de ruteo y las gateways de red. Para conectar su VPC a Amazon SNS, debe definir un punto de enlace de la VPC de tipo interfaz. Este tipo de punto final le permite conectar su VPC a los AWS servicios. Con el punto de enlace, se ofrece conectividad escalable de confianza con Amazon SNS sin necesidad de utilizar una gateway de Internet, una instancia de conversión de las direcciones de red (instancia NAT) o una conexión de VPN. Para obtener más información, consulte Acceso y Servicio de AWS uso de un punto final de VPC de interfaz en la Guía del usuario de Amazon VPC.

La información de esta sección va dirigida a usuarios de Amazon VPC. Para obtener más información y empezar a crear una VPC, consulte Planear su VPC en la Guía del usuario de Amazon VPC.

> **Note**
>
> No puede suscribir un tema de Amazon SNS a una dirección IP privada con los puntos de enlace de la VPC.

Creación de un punto de enlace de la VPC para Amazon SNS

Para publicar mensajes en los temas de Amazon SNS desde una Amazon VPC, cree un punto de enlace de la VPC de tipo interfaz. A continuación, puede publicar mensajes en sus temas a la vez que mantiene el tráfico dentro de la red que administra con la VPC.

Utilice la siguiente información para crear el punto de enlace y probar la conexión entre su VPC y Amazon SNS. O, si desea ver un tutorial que le ayude a comenzar desde cero, consulte Publicación de un mensaje de Amazon SNS desde Amazon VPC.

Creación del punto de enlace

Puede crear un punto de conexión de Amazon SNS en su VPC mediante el AWS Management Console, el, un AWS SDK AWS CLI, la API de Amazon SNS o. AWS CloudFormation

Para obtener información sobre la creación y configuración de un punto de conexión mediante la consola de Amazon VPC o la AWS CLI, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

> **Important**
>
> Puede usar Amazon Virtual Private Cloud solo con puntos de conexión HTTPS de Amazon SNS.
> Al crear el punto de enlace, especifique que Amazon SNS es el servicio al que desea que se conecte la VPC. En la consola de Amazon VPC, los nombres de los servicios varían en función de la región que haya elegido. Por ejemplo, si elige EE. UU. Este (Norte de Virginia), el nombre del servicio es com.amazonaws.us-east-1.sns.
> Al configurar Amazon SNS para enviar mensajes desde Amazon VPC, debe habilitar el DNS privado y especificar los puntos de conexión en el formato `sns.us-east-2.amazonaws.com`.
> Los DNS privados no admiten los puntos de enlace heredados, como `queue.amazonaws.com` o `us-east-2.queue.amazonaws.com`.

Para obtener información sobre cómo crear y configurar un punto de conexión mediante AWS CloudFormation, consulte el *`AWS::EC2::VPCEndpoint`*recurso en la Guía del AWS CloudFormation usuario.

Comprobación de la conexión entre la VPC y Amazon SNS

Después de crear un punto de enlace para Amazon SNS, puede publicar mensajes desde la VPC en sus temas de Amazon SNS. Para probar esta conexión, haga lo siguiente:

1. Conéctese a una EC2 instancia de Amazon que resida en su VPC. Para obtener información sobre la conexión, consulte Conectarse a su instancia de Linux o Conectarse a su instancia de Windows en la EC2 documentación de Amazon.

   Por ejemplo, para conectarse a una instancia de Linux mediante un cliente SSH, ejecute el siguiente comando desde un terminal:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Donde:

   - ec2-key-pair.pem es el archivo que contiene el par de claves que Amazon EC2 proporcionó al crear la instancia.
   - instance-hostname es nombre de host público de la instancia. Para obtener el nombre de host en la EC2consola de Amazon: elige Instances, elige tu instancia y busca el valor de Public DNS.

2. Desde su instancia, utilice el comando `publish` de Amazon SNS con la AWS CLI. Puede enviar un mensaje sencillo a un tema con el siguiente comando:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   Donde:

   - aws-region es la región en AWS la que se encuentra el tema.
   - sns-topic-arnes el nombre del recurso de Amazon (ARN) del tema. Para obtener el ARN desde la consola de Amazon SNS, seleccione Temas, busque su tema y busque el valor en la columna ARN.

   Si Amazon SNS ha recibido correctamente el mensaje, el terminal imprime un ID de mensaje, como el siguiente:

   ```
   {
      "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
   ```

```
}
```

Creación de una política de punto de enlace de la VPC para Amazon SNS

Puede crear una política para los puntos de enlace de Amazon VPC correspondiente a Amazon SNS y especificar lo siguiente:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para obtener más información, consulte Controlar el acceso a servicios con puntos de conexión de VPC en la Guía del usuario de Amazon VPC.

En el siguiente ejemplo de política de punto de enlace de la VPC, se especifica que el usuario de `MyUser` puede publicar en el tema `MyTopic` de 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"
    }
  }]
}
```

Se deniega lo siguiente:

- Otras acciones de la API de Amazon SNS, como `sns:Subscribe` y `sns:Unsubscribe`.
- Otros usuarios y reglas de IAM que intentan utilizar este punto de enlace de la VPC.
- Publicación de `MyUser` en otro tema de Amazon SNS.

> **Note**
>
> El usuario de IAM puede seguir utilizando otras acciones de la API de Amazon SNS desde fuera de la VPC.

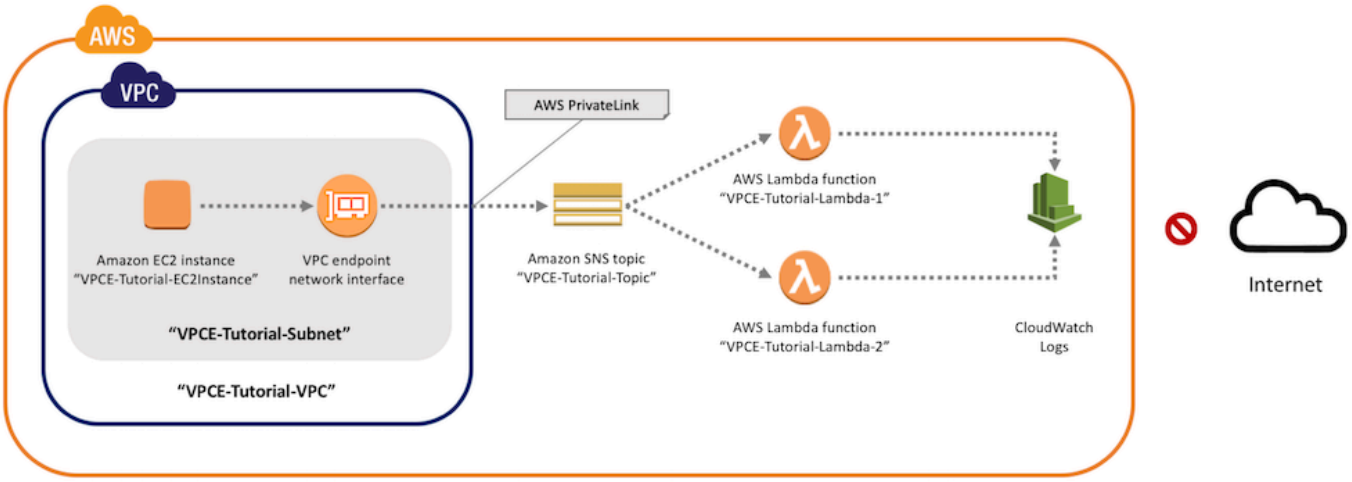
Publicación de un mensaje de Amazon SNS desde Amazon VPC

En esta sección, se describe cómo publicar en un tema de Amazon SNS a la vez que mantiene los mensajes seguros en una red privada. Publicas un mensaje desde una EC2 instancia de Amazon alojada en Amazon Virtual Private Cloud (Amazon VPC). El mensaje permanece dentro de la AWS red sin circular por la Internet pública. Al publicar mensajes de forma privada desde una VPC, puede mejorar la seguridad del tráfico entre sus aplicaciones y Amazon SNS. Esta seguridad es importante cuando publica información personalmente identificable (PII) sobre sus clientes o cuando su aplicación está sujeta a regulaciones del mercado. Por ejemplo, la publicación de forma privada es útil si tiene un sistema de sanidad que debe cumplir con la Ley de portabilidad y responsabilidad de los seguros médicos (HIPAA, por sus siglas en inglés) o un sistema financiero que debe cumplir con el estándar de seguridad de datos del sector de tarjetas de pago (PCI DSS, por sus siglas en inglés).

Los pasos generales son los siguientes:

- Utilice una AWS CloudFormation plantilla para crear automáticamente una red privada temporal en su Cuenta de AWS.
- Crear un punto de enlace de la VPC que conecte la VPC con Amazon SNS.
- Inicia sesión en una EC2 instancia de Amazon y publica un mensaje de forma privada en un tema de Amazon SNS.
- Verificar que el mensaje se entregó correctamente.
- Elimine los recursos que creó durante este proceso para que no permanezcan en su Cuenta de AWS servidor.

El siguiente diagrama muestra la red privada que crearás en tu AWS cuenta al completar estos pasos:



Esta red se compone de una VPC que contiene una instancia de Amazon EC2 . La instancia se conecta a Amazon SNS a través de un punto de enlace de la VPC de interfaz. Este tipo de punto final se conecta a los servicios que funcionan con la tecnología de AWS PrivateLink. Con esta conexión establecida, puede iniciar sesión en la EC2 instancia de Amazon y publicar mensajes en el tema Amazon SNS, aunque la red esté desconectada de la Internet pública. El tema distribuye los mensajes que recibe en dos funciones de suscripción AWS Lambda . Estas funciones registran los mensajes que reciben en Amazon CloudWatch Logs.

Se tarda unos 20 minutos en completar estos pasos.

Temas

Antes de empezar

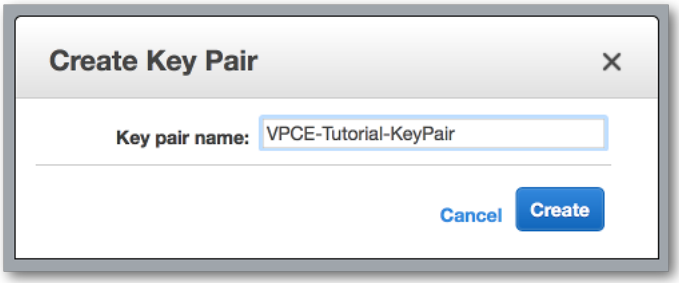
Antes de empezar, necesita una cuenta de Amazon Web Services (AWS). Cuando te registras, tu cuenta se registra automáticamente en todos los servicios de Amazon AWS, incluidos Amazon SNS y Amazon VPC. Si todavía no ha creado una cuenta, vaya a https://aws.amazon.com/ y, a continuación, elija Crear una cuenta gratuita.

Paso 1: Crear un EC2 key pair de Amazon

Se usa un key pair para iniciar sesión en una EC2 instancia de Amazon. Consta de una clave pública que se utiliza para cifrar la información de inicio de sesión y de una clave privada que se utiliza para descifrarla. Al crear un par de claves, se descarga una copia de la clave privada. Más adelante, utilizarás el key pair para iniciar sesión en una EC2 instancia de Amazon. Para iniciar sesión, debe especificar el nombre del par de claves y proporcionar la clave privada.

Para crear el par de claves

1. Inicia sesión en la EC2 consola de Amazon AWS Management Console y ábrela en https://console.aws.amazon.com/ec2/.
2. En el menú de navegación de la izquierda, busque la sección Network & Security (Red y seguridad). A continuación, elija Key Pairs (Pares de claves).
3. Seleccione Create Key Pair.
4. En la ventana Create Key Pair (Crear par de claves), en Key pair name (Nombre del par de claves), escriba **`VPCE-Tutorial-KeyPair`**. A continuación, elija Crear.



5. Su navegador descargará el archivo de clave privada automáticamente. Guárdelo en un lugar seguro. Amazon EC2 le da al archivo una extensión de `.pem`.
6. (Opcional) Si está usando un cliente SSH en un equipo Mac o Linux para conectarse a su instancia, utilice el comando `chmod` para establecer los permisos de su archivo de clave privada de modo que solo usted pueda leerlo:

   a. Abra un terminal y vaya al directorio que contiene la clave privada:

```
$ cd /filepath_to_private_key/
```

b. Establezca los permisos mediante el comando siguiente:

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

Paso 2: Crea los AWS recursos

Para configurar la infraestructura, se utiliza una AWS CloudFormation plantilla. Una plantilla es un archivo que actúa como modelo para crear AWS recursos, como EC2 instancias de Amazon y temas de Amazon SNS. La plantilla de este proceso está disponible GitHub para que la descargue.

Usted proporciona la plantilla y AWS CloudFormation AWS CloudFormation aprovisiona los recursos que necesita como una pila en su Cuenta de AWS. Una pila es una colección de recursos que administra como una única unidad. Cuando termines estos pasos, podrás AWS CloudFormation eliminarlos todos los recursos de la pila a la vez. Estos recursos no permanecen en tus manos Cuenta de AWS, a menos que así lo desees.

En la pila de este proceso, se incluyen los siguientes recursos:

- Una VPC y los recursos de red asociados, incluida una subred, un grupo de seguridad, una gateway de Internet y una tabla de ruteo.
- Una EC2 instancia de Amazon que se lanza a la subred de la VPC.
- Un tema de Amazon SNS.
- Dos funciones AWS Lambda . Estas funciones reciben los mensajes que se publican en el tema Amazon SNS y registran los eventos en CloudWatch los registros.
- Estadísticas CloudWatch y registros de Amazon.
- Una función de IAM que permite a la EC2 instancia de Amazon utilizar Amazon SNS y una función de IAM que permite a las funciones de Lambda escribir en los registros. CloudWatch

Para crear los recursos AWS

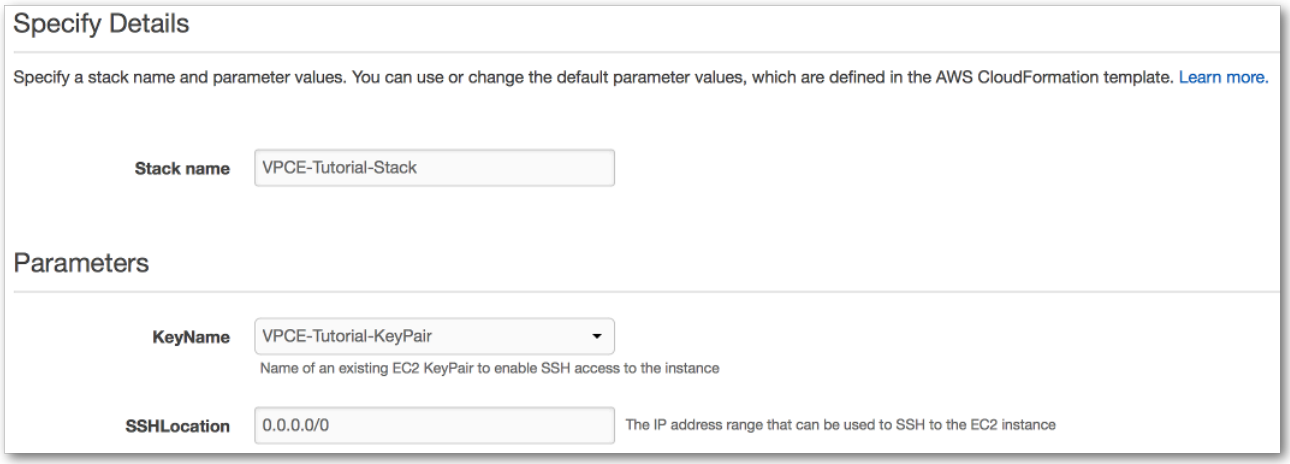
1. Descargue el archivo de plantilla del GitHub sitio web.
2. Inicie sesión en la consola de AWS CloudFormation.
3. Elija Crear pila.

4. En la página Select Template (Seleccionar plantilla), elija Upload a template to Amazon S3 (Cargar una plantilla en Amazon S3), elija el archivo y, a continuación, elija Next (Siguiente).
5. En la página Specify Details (Especificar detalles), especifique el nombre de la pila y el de la clave:

   a. Para Stack name (Nombre de pila), escriba **`VPCE-Tutorial-Stack`**.

   b. Para KeyName, elija VPCE-Tutorial-. KeyPair

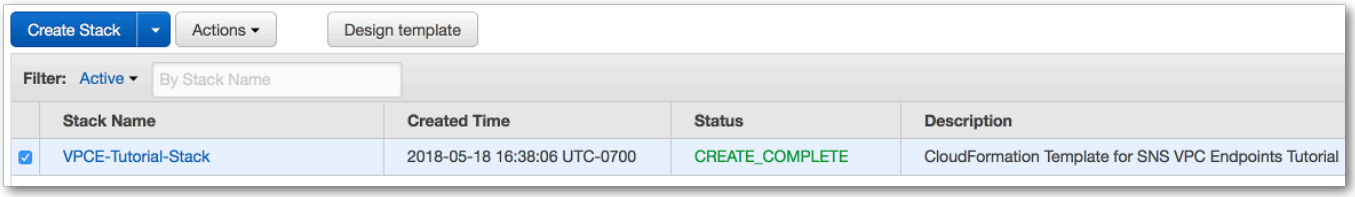
   c. Para SSHLocation, mantenga el valor predeterminado de. **`0.0.0.0/0`**



   d. Elija Siguiente.

6. En la página Options (Opciones), mantenga todos los valores predeterminados y elija Next (Siguiente).
7. En la página Review (Revisar), verifique los detalles de la pila.
8. En Capacidades, reconozca que AWS CloudFormation podría crear recursos de IAM con nombres personalizados.
9. Seleccione Crear.

   La AWS CloudFormation consola abre la página Stacks. VPCE-Tutorial-StackTiene el estado CREATE_IN_PROGRESS. En unos minutos, después de que se complete el proceso de creación, el estado cambia a CREATE_COMPLETE.



> **Tip**
>
> Elija el botón Refresh (Actualizar) para ver el estado más reciente de la pila.

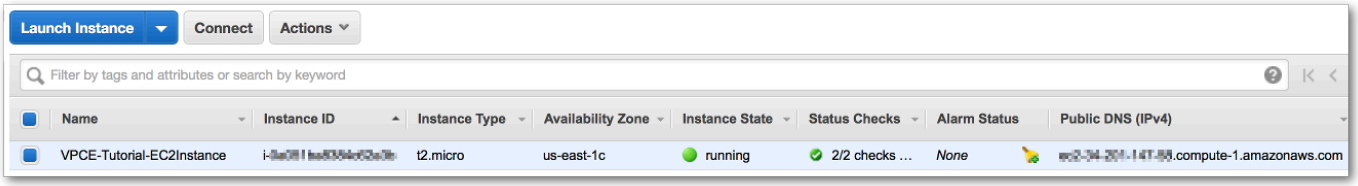
Paso 3: Confirma que tu EC2 instancia de Amazon carece de acceso a Internet

La EC2 instancia de Amazon que se lanzó en tu VPC en el paso anterior carece de acceso a Internet. No permite el tráfico saliente y no puede publicar mensajes en Amazon SNS. Verifíquelo iniciando sesión en la instancia. A continuación, intente conectarse a un punto de enlace público e intentar enviar mensajes a Amazon SNS.

En esta parte, se produce un error en la publicación. En un paso posterior, después de crear un punto de enlace de la VPC para Amazon SNS, el intento de publicación finaliza correctamente.

Para conectarte a tu EC2 instancia de Amazon

1. Abre la EC2 consola de Amazon en https://console.aws.amazon.com/ec2/.
2. En el menú de navegación de la izquierda, busque la sección Instances (Instancias). A continuación, elija Instances (Instancias).
3. En la lista de instancias, selecciona VPCE-. Tutorial-EC2Instance
4. Copia el nombre de host que se proporciona en la columna DNS público.



5. Abra un terminal. Desde el directorio que contiene el key pair, conéctate a la instancia mediante el siguiente comando, donde *instance-hostname* está el nombre de host que copiaste de la EC2 consola de Amazon:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

Para verificar que la instancia carece de conectividad a Internet

- En su terminal, intenta conectarse a cualquier punto de enlace público, como amazon.com:

```
$ ping amazon.com
```

Debido a que se produce un error en el intento de conexión, puede cancelar en cualquier momento (Ctrl + C en Windows o Comando + C en macOS).

Para verificar que la instancia carece de conectividad a Amazon SNS, siga estos pasos:

1. Inicie sesión en la consola de Amazon SNS.
2. En el menú de navegación de la izquierda, elija (Temas).
3. En la página Topics (Temas), copie el nombre de recurso de Amazon (ARN) para el tema VPCE-Tutorial-Topic.
4. En su terminal, intente publicar un mensaje en el tema:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Debido a que se produce un error en el intento de publicación, puede cancelar en cualquier momento.

Paso 4: Crear un punto de enlace de la VPC para Amazon SNS

Para conectar la VPC a Amazon SNS, debe definir un punto de enlace de la VPC de tipo interfaz. Tras añadir el punto de enlace, puede iniciar sesión en la EC2 instancia de Amazon de su VPC y, desde allí, utilizar la API de Amazon SNS. Puede publicar mensajes en el tema, que se publican de forma privada. Permanecen dentro de la AWS red y no viajan por la Internet pública.

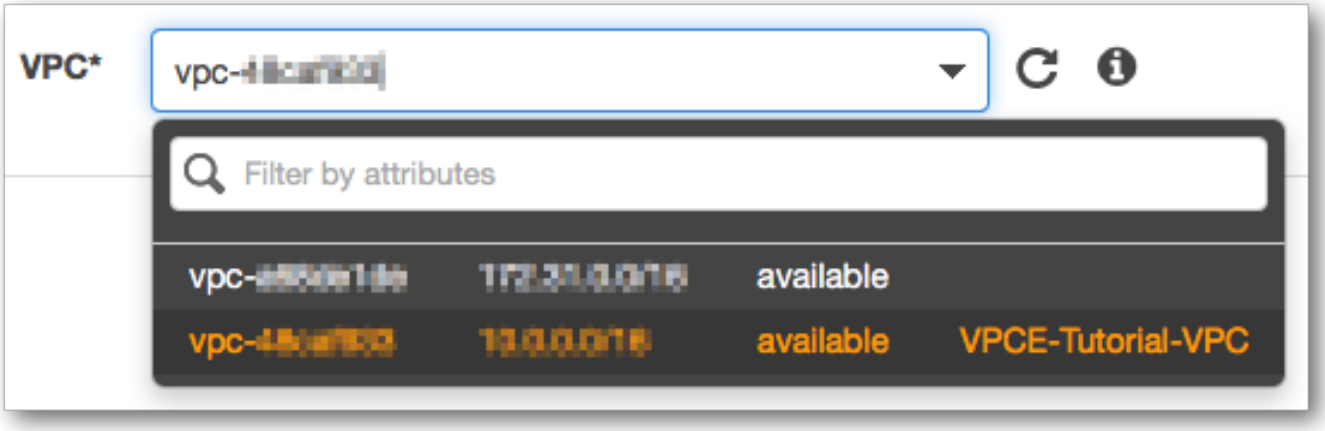
> **Note**
>
> La instancia aún carece de acceso a otros AWS servicios y puntos finales de Internet.

Para crear el punto de enlace

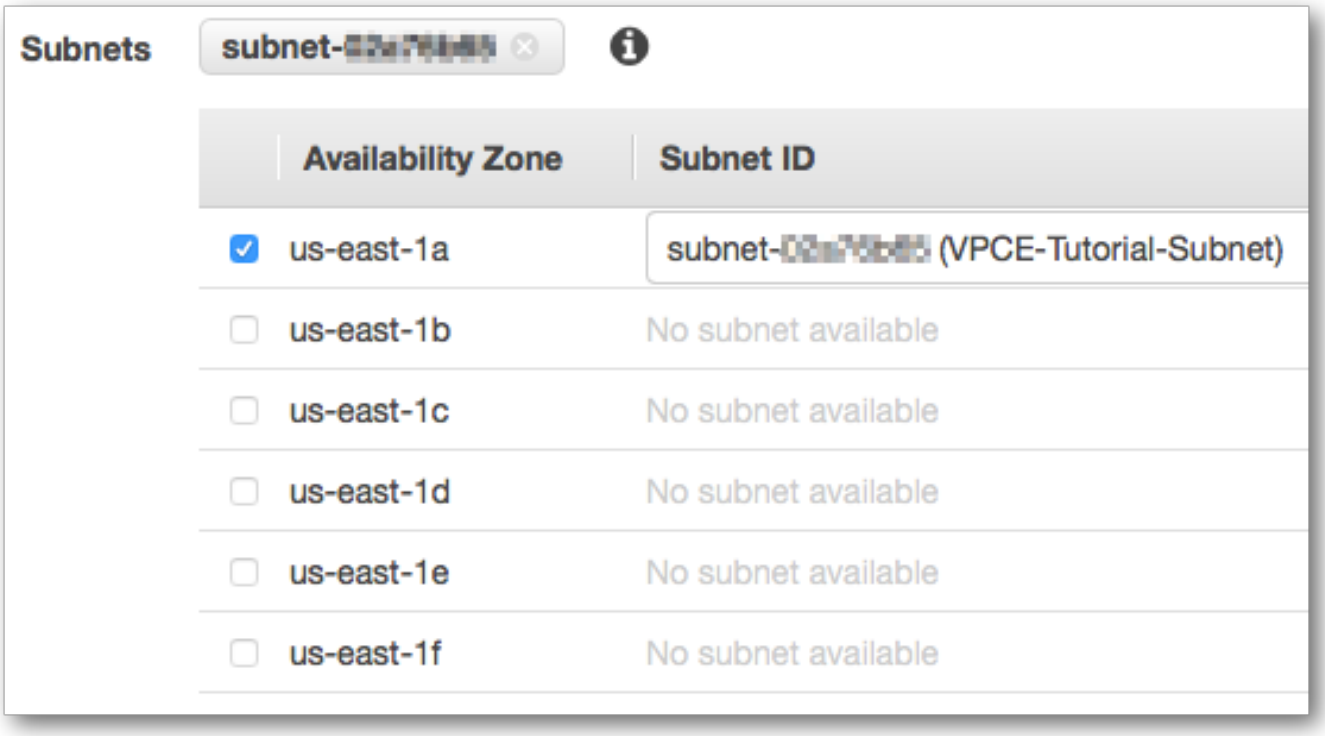
1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el menú de navegación de la izquierda, elija Endpoints (Puntos de enlace).
3. Seleccione Crear punto de conexión.

4. En la página Crear punto de enlace, en Categoría de servicio, mantenga la opción predeterminada Servicios de AWS .
5. En Nombre de servicio, seleccione el nombre de servicio de Amazon SNS.

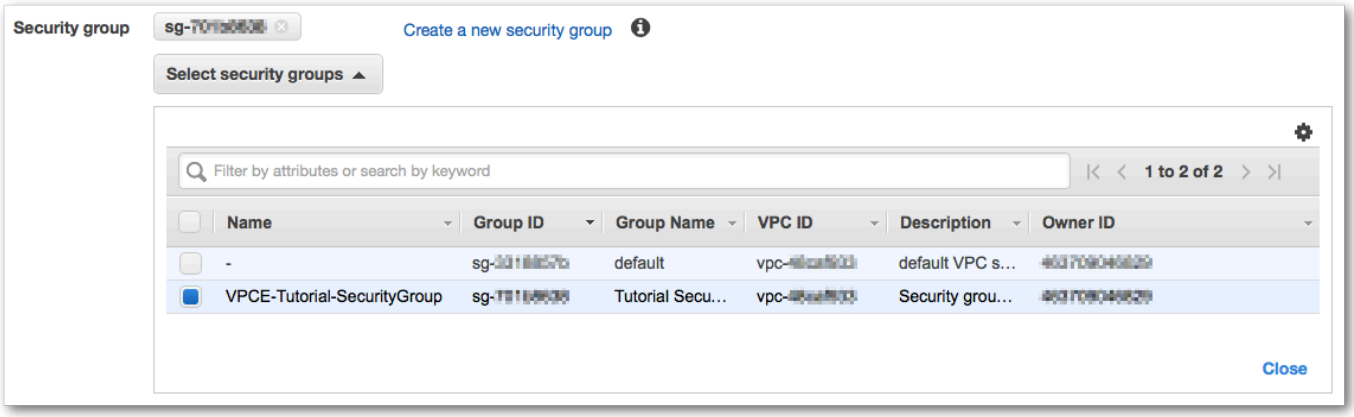
   Los nombres de los servicios varían en función de la región que haya elegido. Por ejemplo, si eliges US East (Virginia del Norte), el nombre del servicio es com.amazonaws. ***us-east-1***.sns.
6. En VPC, elija la VPC denominada VPCE-Tutorial-VPC.



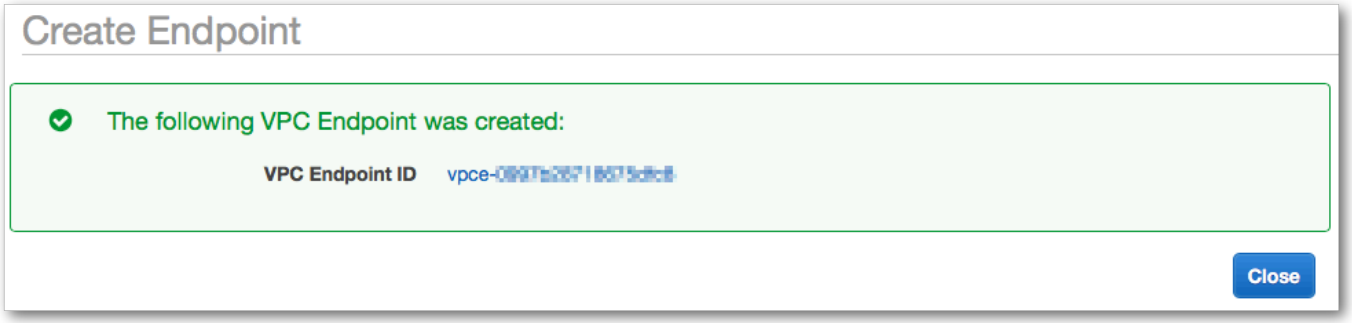
7. En Subnets (Subredes), seleccione la subred que tiene VPCE-Tutorial-Subnet en el ID de subred.



8. En Enable Private DNS Name (Habilitar nombre de DNS privado), seleccione Enable for this endpoint (Habilitar para este punto de enlace).
9. En Grupo de seguridad, elija Seleccionar grupo de seguridad y elija VPCE-Tutorial -. SecurityGroup

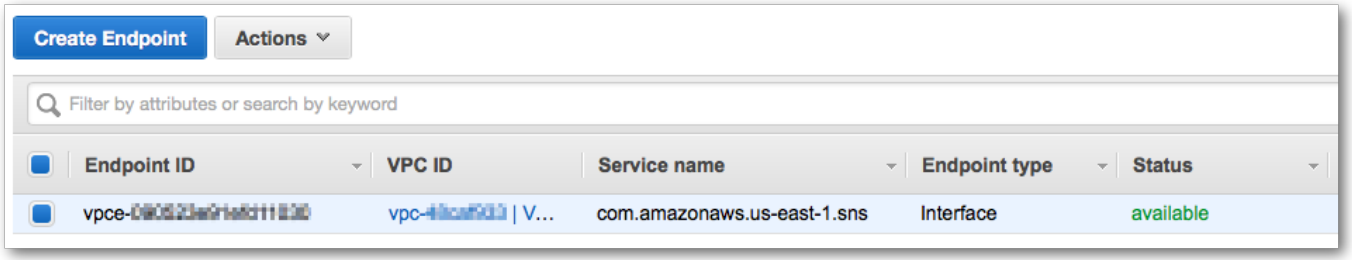


10. Elija Crear punto de conexión. En la consola de Amazon VPC, se confirma que se creó un punto de enlace de la VPC.



11. Seleccione Cerrar.

    En la consola de Amazon VPC, se abre la página Puntos de enlace. El nuevo punto de enlace tiene el estado pending (pendiente). En unos minutos, después de que se complete el proceso de creación, el estado cambia a available (disponible).



Paso 5: Publicar un mensaje en el tema de Amazon SNS

Ahora que su VPC incluye un punto de enlace para Amazon SNS, puede iniciar sesión en la instancia de EC2 Amazon y publicar mensajes en el tema.

Para publicar un mensaje

1. Si tu terminal ya no está conectado a tu EC2 instancia de Amazon, vuelve a conectarte:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Ejecute el mismo comando que usó anteriormente para publicar un mensaje en el tema de Amazon SNS. Esta vez, el intento de publicación se realiza correctamente y Amazon SNS devuelve un ID de mensaje:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"


{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

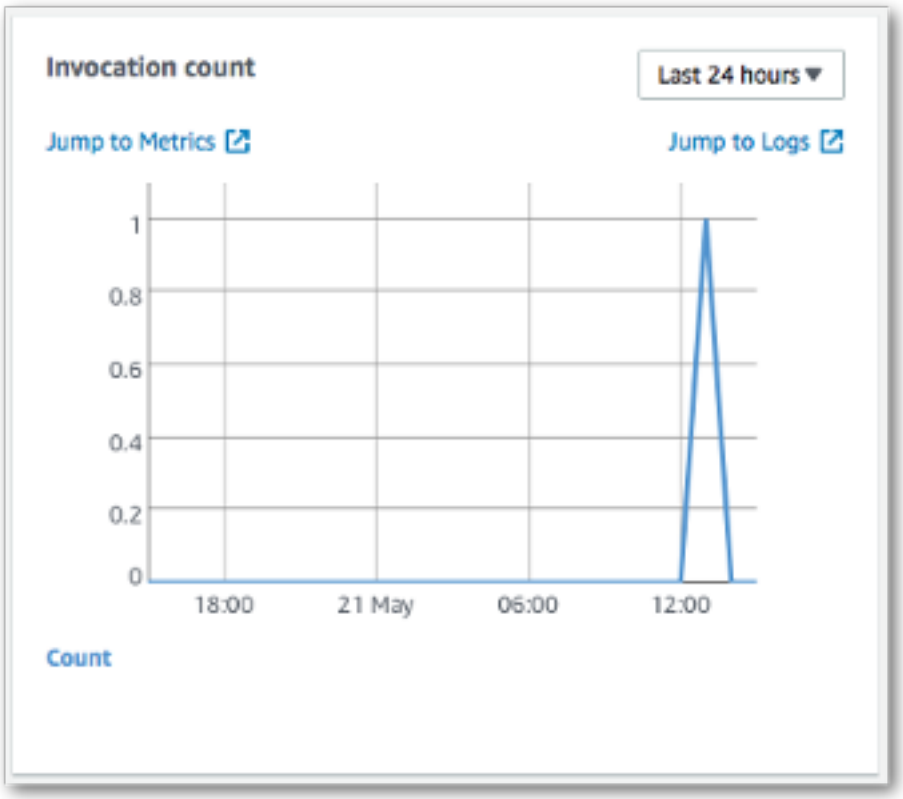
Paso 6: Verificar las entregas de mensajes

Cuando el tema de Amazon SNS recibe un mensaje, envía el mensaje a las dos funciones de Lambda de la suscripción. Cuando estas funciones reciben el mensaje, registran el evento en los CloudWatch registros. Para comprobar que la entrega del mensaje se ha realizado correctamente, compruebe que se hayan invocado las funciones y compruebe que los CloudWatch registros se hayan actualizado.

Para verificar que se invocaron las funciones Lambda, siga estos pasos:

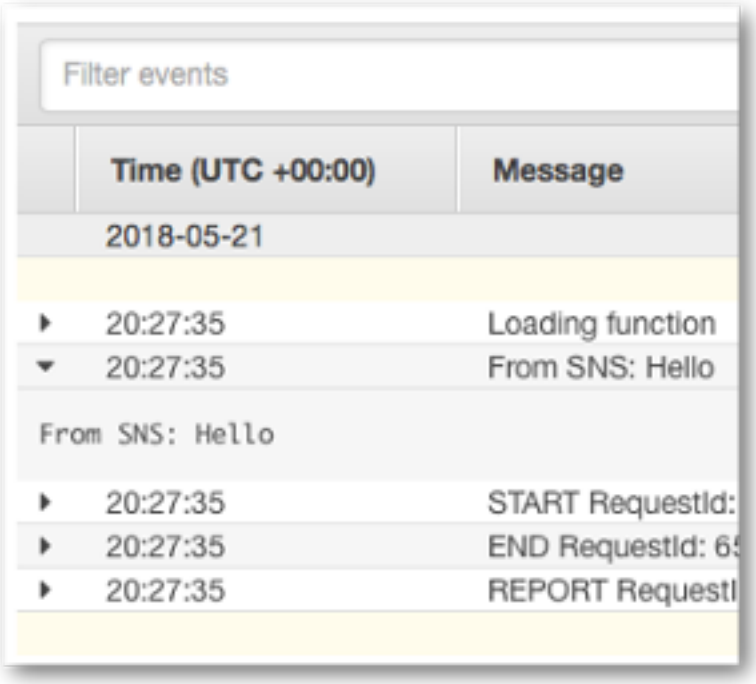
1. Abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. En la página Functions (Funciones), elija VPCE-Tutorial-Lambda-1.
3. Elija Monitoring (Monitorización).
4. Consulte el gráfico Invocation count (Número de invocaciones). En este gráfico, se muestra la cantidad de veces que se ha ejecutado la función Lambda.

El número de invocaciones coincide con el número de veces que ha publicado un mensaje en el tema.



Para comprobar que los CloudWatch registros se han actualizado

1. Abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. En el menú de navegación de la izquierda, elija Logs (Registros).
3. Compruebe que las funciones Lambda hayan escrito los registros:
   a. Elija el grupo de registros/aws/lambda/VPCE-Tutorial-Lambda-1/.
   b. Elija el flujo de registros.
   c. Compruebe que el registro incluye la entrada `From SNS: Hello`.



d. Elija Log Groups (Grupos de registros) en la parte superior de la consola para volver a la página Log Groups (Grupos de registros). A continuación, repita los pasos anteriores para el grupo de registros the /aws/lambda/VPCE -Tutorial-Lambda-2/.

¡Enhorabuena! Al agregar un punto de enlace para Amazon SNS a una VPC, pudo publicar un mensaje en un tema desde de la red que administra la VPC. El mensaje se publicó de forma privada sin exponerse a la red pública de Internet.

Paso 7: limpiar

A menos que desee retener los recursos que creó, puede eliminarlos ahora. Al eliminar AWS los recursos que ya no utilizas, evitas que se te cobren cargos innecesarios. Cuenta de AWS

En primer lugar, elimine el punto de enlace de la VPC mediante la consola de Amazon VPC. A continuación, elimina los demás recursos que hayas creado eliminando la pila de la AWS CloudFormation consola. Al eliminar una pila, AWS CloudFormation elimina los recursos de la pila de los tuyos Cuenta de AWS.

Para eliminar su punto de enlace de la VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el menú de navegación de la izquierda, elija Endpoints (Puntos de enlace).

3. Seleccione el punto de enlace que ha creado.
4. Elija Actions (Acciones) y, a continuación, elija Delete Endpoint (Eliminar punto de enlace).
5. En la ventana Delete Endpoint (Eliminar punto de enlace), elija Yes, Delete (Sí, eliminar).

   El estado del punto de enlace cambia a deleting (eliminando). Cuando finaliza la eliminación, el punto de enlace se quita de la página.

Para eliminar tu AWS CloudFormation pila

1. Abre la AWS CloudFormation consola en https://console.aws.amazon.com/cloudformation.
2. Seleccione la pila VPCE-Tutorial-Stack.
3. Elija Actions (Acciones) y, a continuación, elija Delete Stack (Eliminar pila).
4. En la ventana Delete Stack (Eliminar pila), elija Yes, Delete (Sí, eliminar).

   El estado de la pila cambia a DELETE_IN_PROGRESS. Cuando finaliza la eliminación, la pila se quita de la página.

Recursos relacionados

Para obtener más información, consulte los siguientes recursos.

- AWS Blog de seguridad: Cómo proteger los mensajes publicados en Amazon SNS con AWS PrivateLink
- ¿Qué es Amazon VPC?
- Puntos de enlace de la VPC
- ¿Qué es Amazon EC2?
- Conceptos de AWS CloudFormation

Mejora de la seguridad de Amazon SNS con la protección de datos de mensajes

- Protección de datos de mensajes es una función de Amazon SNS que se utiliza para definir sus propias reglas y políticas para auditar y controlar el contenido de los datos en movimiento, en lugar de los datos en reposo.

- La función de protección de datos de mensajes proporciona servicios de control, conformidad y auditoría para aplicaciones empresariales centradas en los mensajes, de modo que el propietario del tema de Amazon SNS pueda controlar la entrada y salida de datos, y se pueda realizar un seguimiento y registro de los flujos de contenido.
- Puede escribir reglas de control basadas en la carga para evitar que el contenido no autorizado de la carga entre en sus flujos de mensajes.
- Puede conceder diferentes permisos de acceso al contenido a suscriptores individuales y auditar todo el proceso de flujo de contenido.

Identity and Access Management en Amazon SNS

El acceso a Amazon SNS requiere credenciales que AWS pueda utilizar para autenticar sus solicitudes. Estas credenciales deben tener permisos para acceder a AWS los recursos, como los temas y mensajes de Amazon SNS. En las secciones, se presenta información detallada acerca de cómo puede utilizar AWS Identity and Access Management (IAM) y Amazon SNS para proteger sus recursos al controlar quién puede obtener acceso a ellos.

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon SNS. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que realice en Amazon SNS.

Usuario de servicio: si utiliza el servicio Amazon SNS para realizar el trabajo, el administrador proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Amazon EKS para realizar el trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos al administrador. Si no puede acceder a una característica de Amazon SNS, consulte Solución de problemas de identidad y acceso de Amazon Simple Notification Service.

Administrador de servicio: si está a cargo de los recursos de Amazon SNS de la empresa, probablemente tenga acceso completo a Amazon SNS. El trabajo consiste en determinar a qué características y recursos de Amazon SNS deben acceder los usuarios del servicio. Luego, debe

enviar solicitudes a su gestionador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información acerca de cómo la empresa puede utilizar IAM con Amazon SNS, consulte Cómo funciona Amazon SNS con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que desee obtener información sobre cómo escribir políticas para administrar el acceso a Amazon SNS. Para consultar ejemplos de políticas de Amazon SNS basadas en identidades que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidades de Amazon Simple Notification Service.

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario Cuenta de AWS raíz, como usuario de IAM o asumiendo una función de IAM.

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su gestionador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre la firma de solicitudes, consulte AWS Signature Versión 4 para solicitudes API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Autenticación multifactor AWS en IAM en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utiliza el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulta Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utiliza AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulta ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puedes iniciar sesión como grupo. Puedes usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una persona determinada. Para asumir temporalmente un rol de IAM en el AWS Management Console, puede cambiar de un rol de usuario a uno de IAM (consola). Puedes asumir un rol llamando a una operación de AWS API AWS CLI o usando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulta Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles de federación, consulte Crear un rol para un proveedor de identidad de terceros (federación) en la Guía de usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué puedes acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulta Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puedes asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener

información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulta Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en AWS ellas, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puedes asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.
- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y realizan AWS CLI solicitudes a la AWS API. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte Usar un rol de IAM para conceder permisos a las aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede agregar las políticas de IAM a roles y los usuarios puedes asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utiliza para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades puedes clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para obtener más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios puede utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios compatibles. AWS WAF ACLs Para obtener más información ACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puedes establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.

- Políticas de control de servicios (SCPs): SCPs son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y administrar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluido cada usuario Cuenta de AWS raíz. Para obtener más información sobre Organizations SCPs, consulte las políticas de control de servicios en la Guía del AWS Organizations usuario.
- Políticas de control de recursos (RCPs): RCPs son políticas de JSON que puedes usar para establecer los permisos máximos disponibles para los recursos de tus cuentas sin actualizar las políticas de IAM asociadas a cada recurso que poseas. El RCP limita los permisos de los recursos en las cuentas de los miembros y puede afectar a los permisos efectivos de las identidades, incluido el usuario Cuenta de AWS raíz, independientemente de si pertenecen o no a su organización. Para obtener más información sobre Organizations e RCPs incluir una lista de Servicios de AWS ese apoyo RCPs, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también puedes proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Control de acceso

Amazon SNS tiene su propio sistema de permisos basado en recursos que utiliza políticas redactadas en el mismo idioma que las políticas AWS Identity and Access Management (IAM). Esto significa que puede obtener resultados similares con las políticas de Amazon SNS que con las políticas de IAM.

> **Note**
>
> Es importante entender que todos Cuentas de AWS pueden delegar sus permisos a los usuarios de sus cuentas. El acceso entre cuentas permite compartir el acceso a los recursos de AWS sin necesidad de administrar usuarios adicionales. Para obtener información sobre el uso del acceso entre cuentas, consulte Habilitación del acceso entre cuentas en la Guía del usuario de IAM.

Casos de uso de control de acceso de Amazon SNS

Tiene una gran flexibilidad en cuanto al modo de conceder o denegar el acceso a un recurso. Sin embargo, los casos de uso típicos son bastante sencillos:

- Quieres conceder a otra persona Cuenta de AWS un tipo concreto de acción temática (por ejemplo, publicar). Para obtener más información, consulte Conceder Cuenta de AWS acceso a un tema.
- Desea limitar las suscripciones a su tema únicamente al protocolo HTTPS. Para obtener más información, consulte Limitar las suscripciones a HTTPS.
- Desea permitir que Amazon SNS publique mensajes en su cola de Amazon SQS. Para obtener más información, consulte Publique en una cola de Amazon SQS..

Conceptos clave de las políticas de acceso de Amazon SNS

En las siguientes secciones, se describen los conceptos que necesita comprender para utilizar el lenguaje de la política de acceso. Se presentan en un orden lógico, con los primeros términos que necesita conocer en la parte superior de la lista.

Permiso

Un permiso es el mecanismo por el que se concede o se deniega algún tipo de acceso a un recurso concreto. Los permisos tienen básicamente esta forma: "A tiene/no tiene permiso para ejecutar B en C donde D se aplica". Por ejemplo, Jane (A) tiene permiso para publicar (B) en TopicA (C), siempre y cuando utilice el protocolo HTTP (D). Cuando Jane publica en TopicA, el servicio comprueba si Jane tiene permiso y si la solicitud cumple las condiciones establecidas en el permiso.

Instrucción

Una instrucción es la descripción formal de un único permiso, escrita en el lenguaje de la política de acceso. Una instrucción se escribe siempre como parte de un documento contenedor más amplio conocido como política (consulte el concepto siguiente).

Política

Una política es un documento (escrito en el lenguaje de la política de acceso) que actúa como contenedor de una o varias instrucciones. Por ejemplo, una política puede tener dos instrucciones: una que indique que Jane se puede suscribir mediante el protocolo de correo electrónico y otra que afirme que Bob no puede publicar en el Tema A. Tal y como se muestra en la figura siguiente, un escenario equivalente sería tener dos políticas; una que indique que Jane se puede suscribir mediante el protocolo de correo electrónico y otra que indique que Bob no puede publicar en el Tema A.

Solo se permiten caracteres ASCII en los documentos de política. Puede utilizar `aws:SourceAccount` y `aws:SourceOwner` solucionar el escenario en el que necesite conectar otros AWS servicios que contengan caracteres ARNs que no sean ASCII. Vea la diferencia entre `aws:SourceAccount` frente a `aws:SourceOwner`.

Emisor

El emisor es la persona que escribe una política para conceder los permisos para un recurso. El emisor (por definición) es siempre el propietario del recurso. AWS no permite a los usuarios del AWS servicio crear políticas para recursos que no son de su propiedad. Si John es el propietario del recurso, AWS autentica la identidad de John al enviar la política que ha redactado para conceder permisos para ese recurso.

Entidad principal

El principal es la persona o las personas que reciben el permiso en la política. El principal es A en la instrucción “A tiene permiso para hacer B en C, donde D se aplica”. En una política puede configurar el principal en "anyone" (es decir, puede especificar un comodín para representar a todas las personas). Puede hacerlo, por ejemplo, si no desea restringir el acceso en función de la identidad real del solicitante, sino de alguna otra característica identificatoria, como la dirección IP del solicitante.

Acción

La acción es la actividad para cuya ejecución se le da permiso al principal. La acción es B en la declaración “A tiene permiso para hacer B a C cuando D sea de aplicación”. Por lo general, la acción es solo la operación de la solicitud a. AWS Por ejemplo, Jane envía una solicitud a Amazon SNS con `Action=Subscribe`. Puede especificar una o varias acciones en una política.

Recurso

El recurso es el objeto al que el principal solicita acceso. El recurso es C en la instrucción “A tiene permiso para hacer B en C, donde D se aplica”.

Condiciones y claves

Las condiciones consisten en cualquier restricción o detalle sobre el permiso. La condición es D en la declaración “A tiene permiso para hacer B a C cuando D sea de aplicación”. La parte de la política que especifica las condiciones puede ser la más detallada y compleja de todas las partes. Las condiciones típicas están relacionadas con los siguientes elementos:

- Fecha y hora (por ejemplo, la solicitud debe llegar antes de un día determinado).
- Dirección IP (por ejemplo, la dirección IP del solicitante debe formar parte de un determinado intervalo de CIDR).

Una clave es la característica específica que es la base de la restricción del acceso. Por ejemplo, la fecha y la hora de la solicitud.

Las condiciones y las claves se utilizan conjuntamente para expresar la restricción. La forma más sencilla de entender cómo implementar una restricción es utilizando un ejemplo: si desea restringir un acceso antes del 30 de mayo de 2010, utilice la condición denominada `DateLessThan`. Utiliza la clave llamada `aws:CurrentTime` y la configura en el valor `2010-05-30T00:00:00Z`. AWS define

las condiciones y las claves que puede utilizar. El propio AWS servicio (por ejemplo, Amazon SQS o Amazon SNS) también puede definir claves específicas del servicio. Para obtener más información, consulte Permisos de la API de Amazon SNS: referencia de recursos y acciones.

Solicitante

El solicitante es la persona que envía una solicitud a un AWS servicio y solicita acceso a un recurso en particular. El solicitante envía una solicitud a la AWS que, en esencia, dice: «¿Me permitirían ir de B a C cuando proceda D?»

Evaluación

La evaluación es el proceso que utiliza el AWS servicio para determinar si se debe denegar o permitir una solicitud entrante en función de las políticas aplicables. Para obtener más información acerca de la lógica de evaluación, consulte Lógica de evaluación.

Efecto

El efecto es el resultado que desea que devuelva una instrucción de política al evaluarla. Debe especificar este valor cuando escriba las instrucciones de una política, y los valores posibles son deny (denegar) y allow (permitir).

Por ejemplo, puede escribir una política que tenga una instrucción que deniegue todas las solicitudes que procedan de la Antártida (efecto = denegar, dado que la solicitud utiliza una dirección IP asignada a la Antártida). O bien puede escribir una política que tenga una instrucción que permita todas las solicitudes que no procedan de la Antártida (efecto = permitir, dado que la solicitud no proviene de la Antártida). Aunque parece que las dos instrucciones hagan lo mismo, en la lógica del lenguaje de la política de acceso, son diferentes. Para obtener más información, consulte Lógica de evaluación.

Aunque solo se pueden especificar dos valores para el efecto, allow (permitir) o deny (denegar), pueden obtenerse tres resultados diferentes en el momento de la evaluación de la política: denegación predeterminada, permitir o denegación explícita. Para obtener más información, consulte los conceptos siguientes y Lógica de evaluación.

Denegación predeterminada

Una denegación predeterminada es el resultado predeterminado de una política a falta de una instrucción de "permitir" o de una denegación explícita.

Permitir

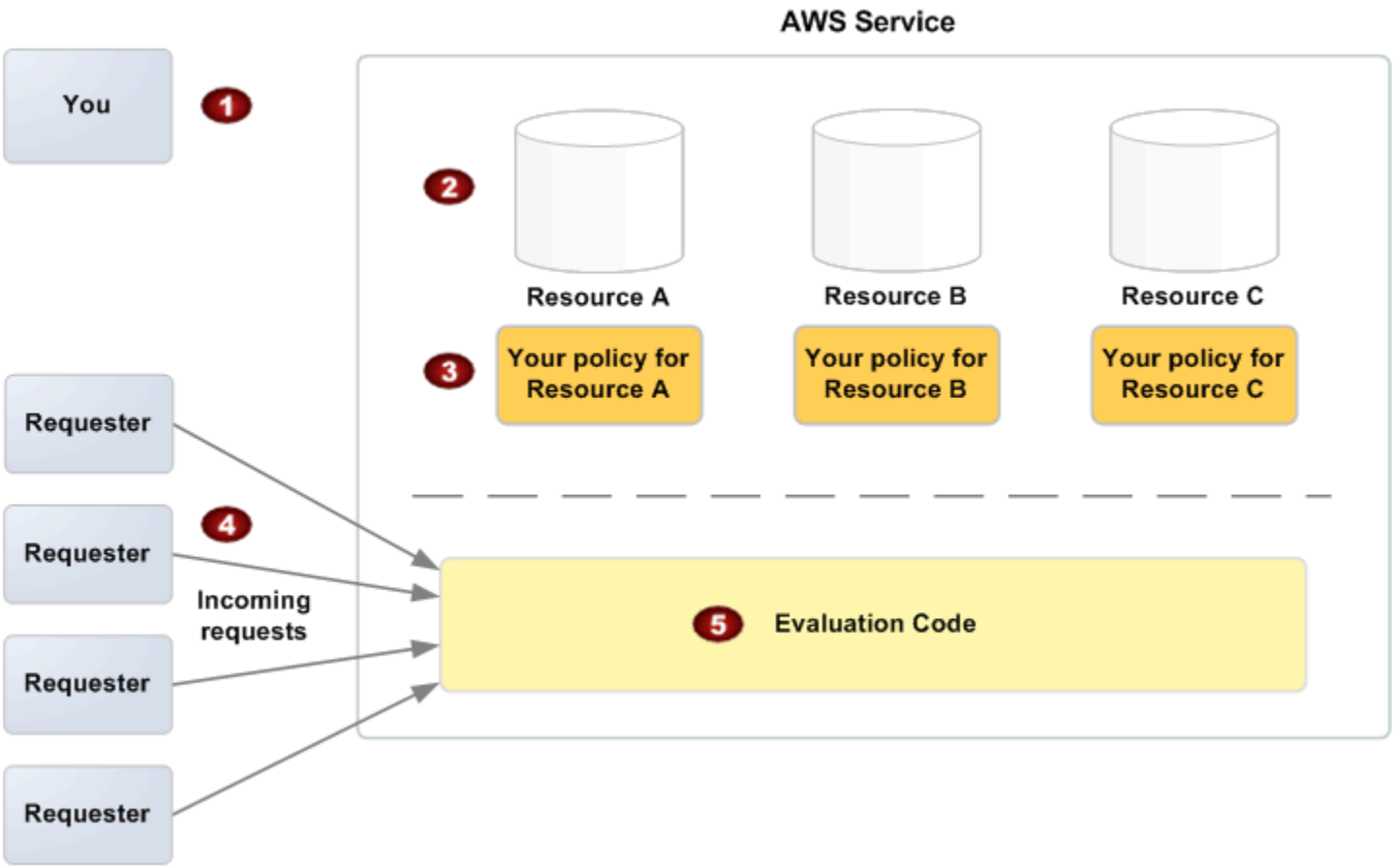
Permitir es el resultado de una instrucción que tenga effect=allow, suponiendo que se cumplan todas las condiciones indicadas. Ejemplo: permitir las solicitudes si se reciben antes de las 13:00 horas del 30 de abril de 2010. Permitir anula todas las denegaciones predeterminadas, aunque nunca una denegación explícita.

Denegación explícita

Una denegación explícita es el resultado de una instrucción que tenga effect=deny, suponiendo que se cumplan todas las condiciones indicadas. Ejemplo: denegar todas las solicitudes si proceden de la Antártida. Cualquier solicitud que proceda de la Antártida siempre se denegará, independientemente de lo que cualquier otra política pueda permitir.

Información general sobre la arquitectura de control de acceso de Amazon SNS

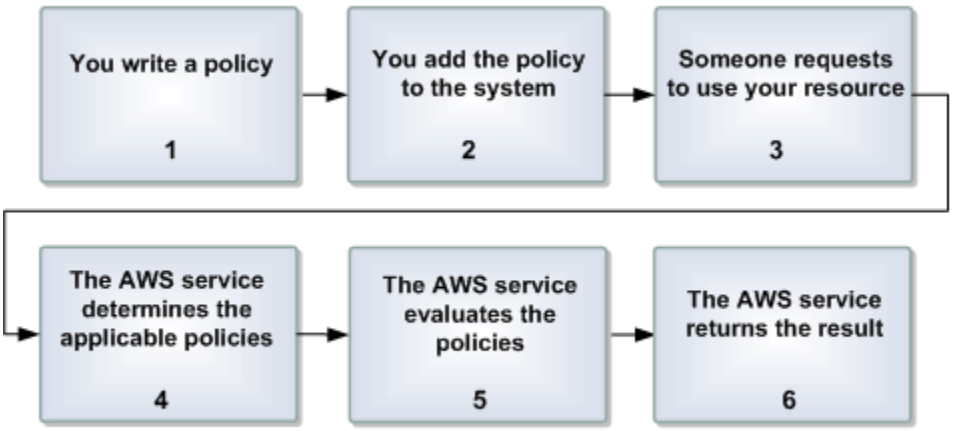
En la figura y la tabla siguientes se describen los componentes principales que interactúan para proporcionar control sobre el acceso a los recursos.



1	Usted, el propietario del recurso.
2	Sus recursos (incluidos en el AWS servicio; por ejemplo, las colas de Amazon SQS).
3	Sus políticas. Normalmente tiene una política por recurso, aunque puede tener varias. El propio AWS servicio proporciona una API que puede utilizar para cargar y gestionar sus políticas.
4	Los solicitantes y sus solicitudes entrantes al AWS servicio.
5	El código de evaluación del lenguaje de la política de acceso. Se trata del conjunto de códigos del AWS servicio que evalúa las solicitudes entrantes en función de las políticas aplicables y determina si el solicitante puede acceder al recurso. Para obtener más información acerca de cómo toma la decisión el servicio, consulte Lógica de evaluación.

Uso del lenguaje de la política de acceso en Amazon SNS

En la figura y la tabla siguiente, se describe el proceso general de cómo funciona el control de acceso con el lenguaje de la política de acceso.



Proceso para utilizar el control de acceso con el lenguaje de la política de acceso

1	Escribe una política para su recurso. Por ejemplo, puede escribir una política para especificar permisos para sus temas de Amazon SNS.
2	Subes tu política a. AWS El AWS servicio en sí proporciona una API que puede utilizar para cargar sus políticas. Por ejemplo, utilice la acción `SetTopicAttributes` de Amazon SNS con el fin de cargar una política para un tema de Amazon SNS concreto.
3	Alguien envía una solicitud para utilizar su recurso. Por ejemplo, un usuario envía una solicitud a Amazon SNS para utilizar uno de sus temas.
4	El AWS servicio determina qué políticas se aplican a la solicitud. Por ejemplo, Amazon SNS busca en todas las políticas de Amazon SNS disponibles y determina cuáles son aplicables (en función de cuál es el recurso, quién es el solicitante, etcétera).
5	El AWS servicio evalúa las políticas.

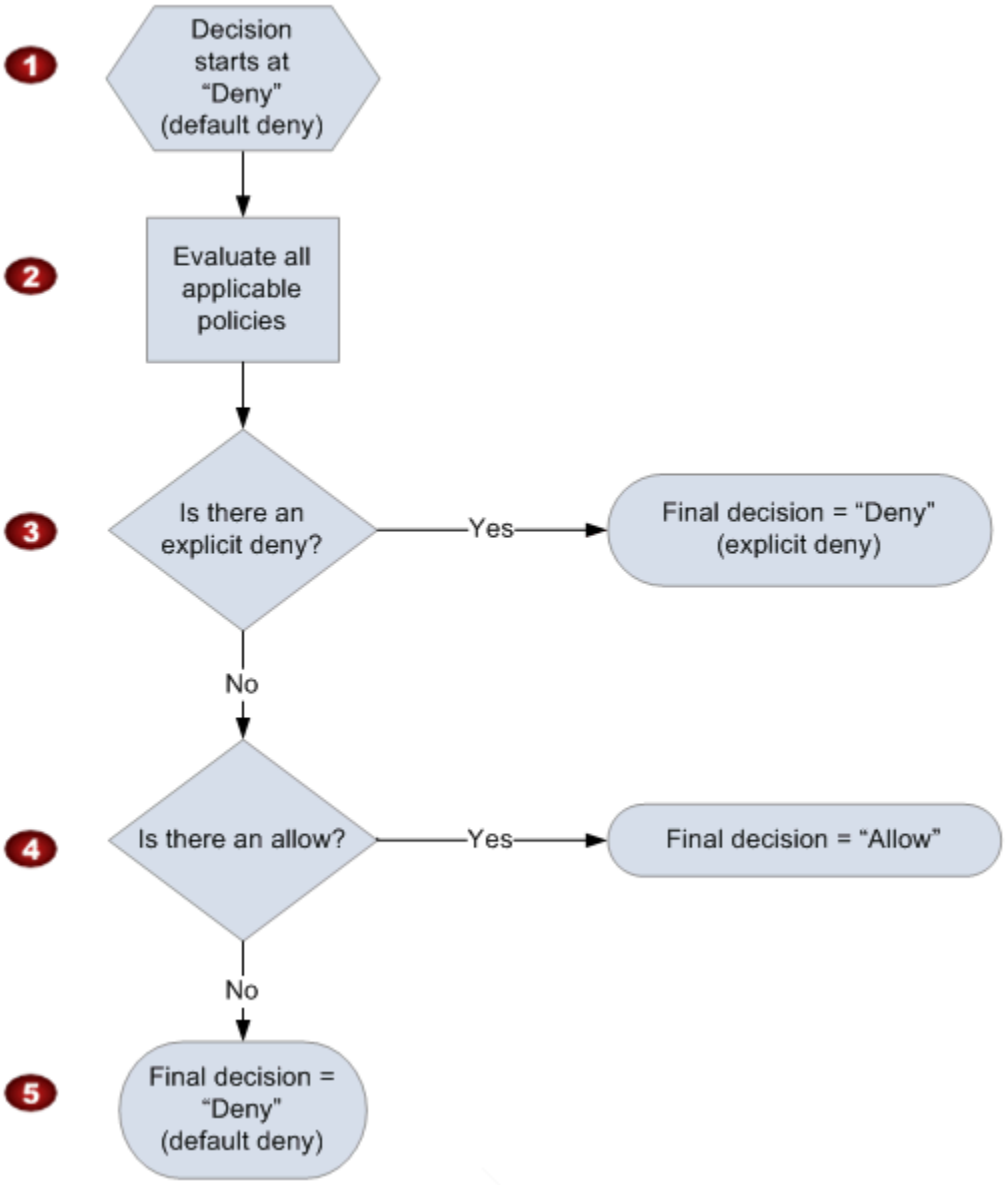
	Por ejemplo, Amazon SNS evalúa las políticas y determina si se permite al solicitante utilizar su tema o no. Para obtener más información acerca de la lógica de decisión, consulte Lógica de evaluación.
6	El AWS servicio deniega la solicitud o continúa procesándola. Por ejemplo, según el resultado de la evaluación de la política, el servicio devuelve al solicitante un error del tipo acceso denegado o continúa procesando la solicitud.

Lógica de evaluación

El objetivo en el momento de la evaluación es decidir si una determinada solicitud debe autorizarse o denegarse. La lógica de evaluación sigue varias reglas básicas:

- De forma predeterminada, todas las solicitudes para utilizar su recurso procedentes de cualquier persona que no sea usted se deniegan
- Una instrucción "permitir" anula todas las denegaciones predeterminadas.
- Una denegación explícita invalida cualquier instrucción "permitir"
- El orden en que se evalúan las políticas no es importante

El siguiente diagrama de flujo y el debate describen con más detalle cómo se toma la decisión.



1	La decisión comienza con una denegación predeterminada.
2	El código de aplicación evalúa todas las políticas aplicables a la solicitud (en función del recurso, el principal, la acción y las condiciones). El orden en que el código de aplicación evalúa las políticas no es importante.
3	En todas estas políticas, el código de aplicación buscará una instrucción de denegación explícita que se aplique a la solicitud.

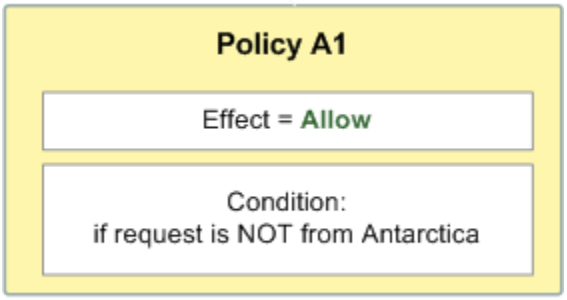
	Aunque solo detecte una, el código de cumplimiento devolverá una decisión de "denegación" y el proceso finalizará (es una denegación explícita; para obtener más información, consulte Denegación explícita).
4	Si no se encuentra ninguna denegación explicita, el código de cumplimiento buscará una instrucción "permitir" que se aplique a la solicitud. Si encuentra alguna, el código de aplicación devolverá una decisión de "permitir" y el proceso se llevará a cabo (el servicio seguirá procesando la solicitud).
5	Si no se detecta ninguna instrucción de "permitir", la decisión final será "denegar" (dado que no había ninguna denegación explícita ni ninguna instrucción de "permitir", se trata de una denegación predeterminada). Para obtener más información, consulte Denegación predeterminada.

Interacción entre denegaciones explícitas y predeterminadas

En una denegación predeterminada se deniega una política si esta no se aplica directamente a la solicitud. Por ejemplo, si un usuario solicita utilizar Amazon SNS, pero la política sobre el tema no hace referencia en Cuenta de AWS absoluto a la del usuario, esa política se traduce en una denegación predeterminada.

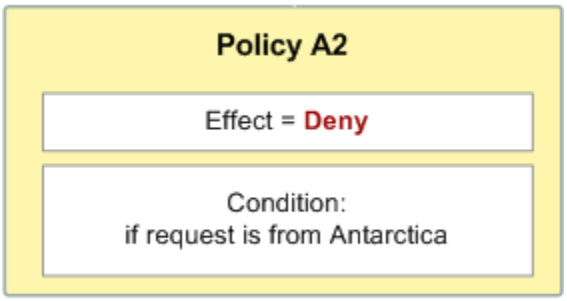
También se deniega una política en una denegación predeterminada si no se cumple una condición de una instrucción. Si se cumplen todas las condiciones de la instrucción, la política dará como resultado una instrucción "permitir" o una denegación explícita, en función del valor del elemento Effect en la política. Las políticas no especifican qué es lo que se debe hacer si no se cumple una condición, por lo que el resultado predeterminado en ese caso es una denegación predeterminada.

Por ejemplo, supongamos que desea evitar las solicitudes que procedan de la Antártida. Puede escribir una política (denominada Policy A1) que permita una solicitud solo si no procede de la Antártida. El siguiente diagrama ilustra esta política.



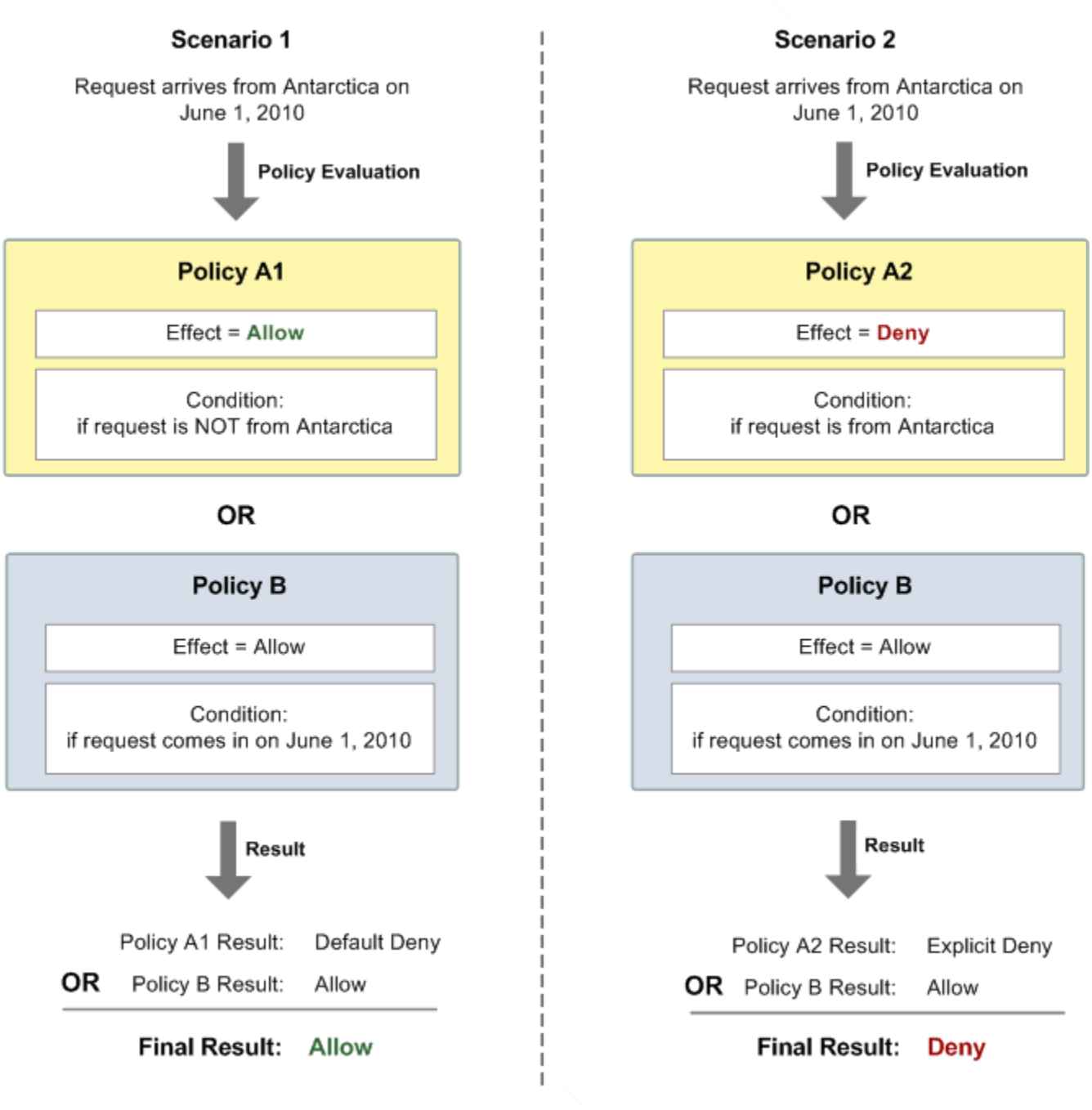
Si alguien envía una solicitud desde los Estados Unidos, dicha condición se cumple (la solicitud no procede de la Antártida). Por lo tanto, la solicitud se permite. Pero, si alguien envía una solicitud desde la Antártida, la condición no se cumple y, por lo tanto, la política genera una denegación predeterminada.

Puede convertir el resultado en una denegación explícita reescribiendo la política (denominada Policy A2) como se indica en el siguiente diagrama. Aquí la política deniega de forma explícita una solicitud si procede de la Antártida.



Si alguien envía una solicitud desde la Antártida, la condición se cumple y, por lo tanto, el resultado de la política es una denegación explícita.

La diferencia entre una denegación predeterminada y una denegación explícita es importante, ya que una denegación predeterminada se puede anular mediante una instrucción "permitir", mientras que una denegación explícita, no. Por ejemplo, supongamos que hay otra política que permite las solicitudes que llegan el 1 de junio de 2010. ¿Cómo afecta esta política al resultado general cuando se asocia a la política que restringe el acceso desde la Antártida? Vamos a comparar el resultado general cuando se asocia la política basada en la fecha (denominada Policy B) a las políticas anteriores A1 y A2. En el escenario 1 se asocia Policy A1 a Policy B y, en el escenario 2, se asocia Policy A2 a Policy B. La figura y la discusión siguientes muestran el resultado que se obtiene cuando llega una solicitud procedente de la Antártida el 1 de junio de 2010.



En el escenario 1, Policy A1 devuelve una denegación predeterminada, tal y como se ha descrito anteriormente en esta sección. Policy B devuelve "permitir", ya que la política (por definición) permite las solicitudes que entran el 1 de junio de 2010. El resultado "permitir" de la Policy B anula la denegación predeterminada de la Policy A1 y, por dicho motivo, la solicitud se permite.

En el escenario 2, la Policy A2 devuelve una denegación explícita, tal y como se ha descrito anteriormente en esta sección. De nuevo, la Policy B da como resultado "permitir". La denegación

explícita de Policy A2 anula el resultado "permitir" de Policy B y, por dicho motivo, la solicitud se deniega.

Ejemplos de casos de control de acceso con Amazon SNS

En esta sección, se ofrecen algunos ejemplos de casos de uso habituales de control de acceso.

Conceder Cuenta de AWS acceso a un tema

Supongamos que tiene un tema en Amazon SNS y desea permitir que uno o varios de ellos Cuentas de AWS realicen una acción específica sobre ese tema, como publicar mensajes. Para ello, puede utilizar la acción de API de Amazon SNS `AddPermission`.

La `AddPermission` acción le permite especificar un tema, una lista de Cuenta de AWS IDs, una lista de acciones y una etiqueta. A continuación, Amazon SNS genera y añade automáticamente una nueva instrucción de política a la política de control de acceso del tema. No es necesario que escriba usted mismo la instrucción de política: Amazon SNS se encarga de ello por usted. Si necesita eliminar la política más adelante, puede hacerlo llamando a `RemovePermission` y proporcionando la etiqueta que utilizó al añadir el permiso.

Por ejemplo, si llamas al `AddPermission` tema arn:aws:sns:us-east- 2:444455556666:MyTopic, especificas el Cuenta de AWS ID 1111-2222-3333, la `Publish` acción y la etiqueta`grant-1234-publish`, Amazon SNS generará e insertará la siguiente declaración de política en la política de control de acceso del tema:

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Tras añadir esta declaración, el 1111-2222-3333 tendrá permiso para publicar mensajes sobre el tema. Cuenta de AWS

Información adicional:

- Administración de políticas personalizada: aunque `AddPermission` resulta práctico para conceder permisos, a menudo es útil administrar manualmente la política de control de acceso del tema en escenarios más complejos, como añadir condiciones o conceder permisos a roles de IAM o servicios específicos. Para ello, puede utilizar la API `SetTopicAttributes` para actualizar el atributo de política directamente.
- Prácticas recomendadas de seguridad: Tenga cuidado al conceder permisos para asegurarse de que solo las entidades Cuentas de AWS o de confianza tengan acceso a sus temas de Amazon SNS. Revise y audite periódicamente las políticas adjuntas a sus temas para mantener la seguridad.
- Límites de las políticas: tenga en cuenta que existen límites en cuanto al tamaño y la complejidad de las políticas de Amazon SNS. Si necesita añadir muchos permisos o condiciones complejas, asegúrese de que su política se mantenga dentro de estos límites.

Limitar las suscripciones a HTTPS

Para restringir el protocolo de entrega de notificaciones para su tema de Amazon SNS a HTTPS, debe crear una política personalizada. La acción `AddPermission` de Amazon SNS no le permite especificar restricciones de protocolo al conceder acceso a su tema. Por lo tanto, debe crear manualmente una política que imponga esta restricción y, a continuación, utilizar la acción `SetTopicAttributes` para aplicar la política a su tema.

A continuación, le explicamos cómo crear una política que limite las suscripciones a HTTPS:

1. Escriba la política. La política debe especificar el ID de Cuenta de AWS al que desea conceder el acceso e imponer la condición de que solo se permitan las suscripciones HTTPS. A continuación, se muestra un ejemplo de política que concede a la Cuenta de AWS con el ID 1111-2222-3333 permiso para suscribirse al tema, pero solo si el protocolo utilizado es 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 la política. Utilice la acción `SetTopicAttributes` de la API de Amazon SNS para aplicar esta política a su tema. Establezca el atributo `Policy` del tema en la política JSON que creó.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Información adicional:

- Personalización del control de acceso. Este enfoque le permite aplicar controles de acceso más detallados, como restringir los protocolos de suscripción, lo que no es posible solo con una acción `AddPermission`. Las políticas personalizadas ofrecen flexibilidad en situaciones que requieren condiciones específicas, como la aplicación de protocolos o las restricciones de direcciones IP.
- Prácticas recomendadas de seguridad. Limitar las suscripciones a HTTPS mejora la seguridad de las notificaciones al garantizar que los datos en tránsito estén cifrados. Revise periódicamente las políticas de los temas para asegurarse de que cumplen sus requisitos de seguridad y conformidad.
- Pruebas de la política. Antes de aplicar la política en un entorno de producción, pruébela en un entorno de desarrollo para asegurarse de que se comporta según lo previsto. Esto ayuda a evitar problemas de acceso accidental o restricciones no deseadas.

Publique en una cola de Amazon SQS.

Para publicar mensajes desde su tema de Amazon SNS en una cola de Amazon SQS, debe configurar los permisos correctos en la cola de Amazon SQS. Si bien tanto Amazon SNS como Amazon SQS AWS utilizan el lenguaje de políticas de control de acceso, debe establecer explícitamente una política en la cola de Amazon SQS para permitir el envío de mensajes desde el tema Amazon SNS.

Para ello, utilice la acción `SetQueueAttributes` para aplicar una política personalizada a la cola de Amazon SQS. A diferencia de Amazon SNS, Amazon SQS no admite la acción `AddPermission` para crear instrucciones de política con condiciones. Por lo tanto, debe escribir la política manualmente.

A continuación, se muestra un ejemplo de una política de Amazon SQS que concede permiso a Amazon SNS para enviar mensajes a la cola. Tenga en cuenta que esta política está asociada a la cola de Amazon SQS, no al tema de Amazon SNS. Las acciones especificadas son acciones de Amazon SQS y el recurso es el Nombre de recurso de Amazon (ARN) de la cola. Puede recuperar el ARN de la cola mediante la acción `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"
      }
    }
  }]
}
```

Esta política utiliza la condición `aws:SourceArn` para restringir el acceso a la cola de SQS en función del origen de los mensajes que se envían. Esto garantiza que solo los mensajes que se originen en el tema de SNS especificado (en este caso, arn:aws:sns:us-east- 2:444455556666:) puedan entregarse a la cola. MyTopic

Información adicional:

- ARN de la cola. Asegúrese de recuperar el ARN correcto de la cola de Amazon SQS mediante la acción `GetQueueAttributes`. Este ARN es esencial para establecer los permisos correctos.
- Prácticas recomendadas de seguridad. Al configurar políticas, siga siempre el principio de privilegios mínimos. Conceda únicamente los permisos necesarios al tema Amazon SNS para

interactuar con la cola de Amazon SQS y revise sus políticas periódicamente para asegurarse de que son seguras. up-to-date

- Políticas predeterminadas en Amazon SNS. Amazon SNS no concede automáticamente una política predeterminada que permita a otras cuentas Servicios de AWS o a otras cuentas acceder a los temas recién creados. De forma predeterminada, los temas de Amazon SNS se crean sin permisos, lo que significa que son privados y solo puede acceder a ellos la cuenta que los creó. Para permitir el acceso a otras Servicios de AWS cuentas o entidades principales, debe definir y adjuntar de forma explícita una política de acceso al tema. Esto se ajusta al principio del privilegio mínimo, que garantiza que no se conceda ningún acceso no intencionado de forma predeterminada.
- Pruebas y validación. Tras configurar la política, pruebe la integración publicando los mensajes en el tema Amazon SNS y verificando que se hayan entregado correctamente a la cola de Amazon SQS. Esto ayuda a confirmar que la política está configurada correctamente.

Permitir que las notificaciones de eventos de Amazon S3 publiquen en un tema

Para permitir que un bucket de Amazon S3 de otro Cuenta de AWS publique notificaciones de eventos en su tema de Amazon SNS, debe configurar la política de acceso del tema en consecuencia. Esto implica escribir una política personalizada que conceda permiso al servicio Amazon S3 desde la Cuenta de AWS específica y, a continuación, aplicar esta política a su tema de Amazon SNS.

Así es como se puede configurar:

1. Escriba la política. La política debe conceder el servicio Amazon S3 (s3.amazonaws.com) los permisos necesarios para publicar en su tema de Amazon SNS. Utilizará la `SourceAccount` condición para asegurarse de que solo la persona especificada Cuenta de AWS, que es la propietaria del bucket de Amazon S3, pueda publicar notificaciones en su tema.

   A continuación se muestra un ejemplo 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"
      }
    }
  }]
}
```

- Propietario del tema: 111122223333 es el Cuenta de AWS ID propietario del tema de Amazon SNS.
- Propietario del bucket de Amazon S3: 444455556666 es el Cuenta de AWS ID propietario del bucket de Amazon S3 que envía las notificaciones.

2. Aplique la política. Utilice la acción `SetTopicAttributes` para establecer esta política en su tema de Amazon SNS. Esto actualizará el control de acceso del tema para incluir los permisos especificados en su 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());
```

Información adicional:

- Uso de la condición **`SourceAccount`**. La `SourceAccount` condición garantiza que solo los eventos que se originen en lo especificado Cuenta de AWS (444455556666 en este caso) puedan activar el tema de Amazon SNS. Se trata de una medida de seguridad para evitar que cuentas no autorizadas envíen notificaciones a su tema.
- Otros servicios que admiten **`SourceAccount`**. La condición `SourceAccount` se admite en los siguientes servicios. Es fundamental utilizar esta condición si quiere restringir el acceso a su tema de Amazon SNS en función de la cuenta de origen.
  - Amazon API Gateway
  - Amazon CloudWatch
  - El DevOps gurú de Amazon
  - Amazon EventBridge

- GameLift Servidores Amazon
- API de SMS y voz de Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- Administrador de incidentes de AWS Systems Manager

- Pruebas y validación. Tras aplicar la política, pruebe la configuración desencadenando un evento en el bucket de Amazon S3 y confirmando que se ha publicado correctamente en su tema de Amazon SNS. Esto ayudará a garantizar que la política esté configurada correctamente.
- Prácticas recomendadas de seguridad. Revise y audite periódicamente las políticas de los temas de Amazon SNS para asegurarse de que cumplen sus requisitos de seguridad y conformidad. Limitar el acceso únicamente a cuentas y servicios de confianza es fundamental para mantener la seguridad de las operaciones.

Permitir que Amazon SES publique en un tema propiedad de otra cuenta

Puedes permitir que otra Servicio de AWS persona publique en un tema que sea propiedad de otra persona Cuenta de AWS. Supongamos que ha iniciado sesión en la cuenta 111122223333, ha abierto Amazon SES y ha creado un correo electrónico. Para publicar notificaciones sobre este correo electrónico en un tema de Amazon SNS que posee la cuenta 444455556666, debe crear una política como la siguiente. Para ello, debe proporcionar información sobre la entidad principal (el otro servicio) y la propiedad de cada recurso. En la instrucción `Resource`, se proporciona el tema ARN, que incluye el ID de cuenta del propietario del tema, 444455556666. En la instrucción `"aws:SourceOwner": "111122223333"`, se especifica que su cuenta es propietaria del correo electrónico.

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

Al publicar eventos en Amazon SNS, los siguientes servicios son compatibles con `aws:SourceOwner`:

- Amazon API Gateway
- Amazon CloudWatch
- El DevOps gurú de Amazon
- GameLift Servidores Amazon
- API de SMS y voz de Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- Administrador de incidentes de AWS Systems Manager

`aws:SourceAccount` frente a `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner` está obsoleto y los nuevos servicios pueden integrarse con Amazon SNS solo a través de `aws:SourceArn` y `aws:SourceAccount`. Amazon SNS sigue manteniendo la compatibilidad con versiones anteriores para los servicios existentes que actualmente admiten `aws:SourceOwner`.

Cuando publican en un tema de Amazon SNS, algunos Servicios de AWS establecen las claves de condición `aws:SourceAccount` y `aws:SourceOwner`. Si se admite, el valor será el ID de AWS cuenta de 12 dígitos en cuyo nombre el servicio publica los datos. Algunos servicios admiten uno, y otros apoyan el otro.

- Permitir que las notificaciones de eventos de Amazon S3 publiquen en un temaConsulte cómo se utilizan las notificaciones de Amazon S3 `aws:SourceAccount` y una lista de AWS los servicios que admiten esa condición.
- Permitir que Amazon SES publique en un tema propiedad de otra cuentaConsulte cómo utiliza Amazon SES `aws:SourceOwner` y una lista de AWS los servicios que cumplen con esa condición.

Permita que las cuentas de una organización AWS Organizations publiquen en un tema de una cuenta diferente

El AWS Organizations servicio le ayuda a gestionar de forma centralizada la facturación, controlar el acceso y la seguridad y compartir los recursos entre sus clientes Cuentas de AWS.

Puede encontrar su ID de organización en la consola de organizaciones. Para obtener más información, consulte Ver detalles de una organización desde la cuenta de administración.

En este ejemplo, cualquier miembro de la organización `myOrgId` puede publicar Cuenta de AWS en Amazon SNS el tema `MyTopic` de la cuenta. `444455556666` La política comprueba el valor del ID de organización mediante la clave de condición 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"
                }
            }
        }
    ]
}
```

Permita que cualquier CloudWatch alarma se publique en un tema de otra cuenta

Siga los siguientes pasos para invocar un tema de Amazon SNS con CloudWatch una alarma en diferentes canales. Cuentas de AWS En este ejemplo se utilizan dos cuentas:

- La cuenta A se utiliza para crear la CloudWatch alarma.
- La cuenta B se usa para crear un tema de SNS.

Crea un tema de SNS en la cuenta B.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elige Temas y, a continuación, seleccione Crear tema.
3. Elija Estándar como tipo de tema y, a continuación, cree un nombre para el tema.
4. Seleccione Crear tema y, a continuación, copie el ARN del tema.
5. En el panel de navegación, seleccione Suscripciones y, a continuación, elija Crear suscripción.
6. Agregue el ARN del tema en la sección ARN del tema, elija Correo electrónico como protocolo y, a continuación, introduzca una dirección de correo electrónico.
7. Selecciona Crear suscripción y, a continuación, comprueba tu correo electrónico para confirmar la suscripción.

Crea una CloudWatch alarma en la cuenta A

1. Abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.

2. En el panel de navegación, selecciona Alarmas y, a continuación, selecciona Crear alarmas.
3. Si aún no ha creado una alarma, cree una ahora. De lo contrario, seleccione su métrica y, a continuación, proporcione los detalles del umbral y los parámetros de comparación.
4. En Configurar acciones, en Notificaciones, elija Usar el ARN del tema para notificar a otras cuentas y, a continuación, introduzca el tema ARN de la cuenta B.
5. Cree un nombre para la alarma y, a continuación, elija Crear alarma.

Actualice la política de acceso del tema de SNS en la cuenta B

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación, elija Temas y, a continuación, seleccione el tema.
3. Elija Editar y, a continuación, añada lo siguiente a la política:

> **Note**
>
> Sustituya los valores de ejemplo de la política siguiente por los suyos propios.

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

Pruebe la alarma

Para probar la alarma, cambie el umbral de la alarma en función de los puntos de datos métricos o cambie manualmente el estado de la alarma. Al cambiar el umbral o el estado de la alarma, recibirá una notificación por correo electrónico.

Solución alternativa para usar un tema local de Amazon SNS y reenviar mensajes

Siga los siguientes pasos para activar las notificaciones CloudWatch multicuenta de Amazon SNS para las alarmas:

1. Crea un tema de Amazon SNS en la misma cuenta que la CloudWatchalarma (111122223333).
2. Suscriba una función de Lambda o una EventBridgeregla de Amazon a ese tema de Amazon SNS.
3. A continuación, la función o EventBridge regla de Lambda puede publicar el mensaje en el tema Amazon SNS de la cuenta de destino (444455556666).

Restringir las publicaciones en un tema de Amazon SNS únicamente desde un punto de enlace de la VPC específico

En este caso, el tema de la cuenta 444455556666 puede publicar únicamente desde el punto de enlace de la VPC con el 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"
      }
    }
  }]
}
```

Cómo funciona Amazon SNS con IAM

Antes de utilizar IAM para administrar el acceso a Amazon SNS, obtenga información sobre qué características de IAM se encuentran disponibles con Amazon SNS.

Características de IAM que puede utilizar con Amazon Simple Notification Service

Característica de IAM	Compatibilidad de Amazon SNS
Políticas basadas en identidades	Sí
Políticas basadas en recursos	Sí
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACLs	No
ABAC (etiquetas en políticas)	Parcial
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	Sí
Roles vinculados al servicio	No

Para obtener una visión general de cómo funcionan Amazon SNS y otros AWS servicios con la mayoría de las funciones de IAM, consulte los AWS servicios que funcionan con IAM en la Guía del usuario de IAM.

AWS políticas gestionadas para Amazon Simple Notification Service

Una política AWS administrada es una política independiente creada y administrada por. AWS AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

AWS política gestionada: Amazon SNSFull Access

`AmazonSNSFullAccess`proporciona acceso completo a Amazon SNS mediante. AWS Management Console Esta política también incluye las siguientes acciones de lectura y escritura para AWS End User Messaging SMS cuando se llame mediante Amazon SNS. Puede asociar esta política a sus usuarios, grupos o roles.

Detalles de los permisos

Los siguientes permisos se aplican únicamente cuando se utiliza Amazon SNS APIs:

- `sns:*`: concede permisos completos para realizar cualquier acción relacionada con Amazon SNS. Este comodín (*) significa que el usuario puede ejecutar todas las acciones posibles de Amazon SNS.

- `sms-voice:DescribeVerifiedDestinationNumbers`: permite recuperar una lista de números de teléfono que se han verificado para el envío de mensajes SMS dentro de la Cuenta de AWS.
- `sms-voice:CreateVerifiedDestinationNumber`— Le permite verificar un nuevo número de teléfono para usarlo con los servicios de mensajería SMS internos AWS.
- `sms-voice:SendDestinationNumberVerificationCode`: permite enviar un código de verificación a un número de teléfono que esté en proceso de verificación para el envío de mensajes SMS dentro de AWS.
- `sms-voice:SendTextMessage`: permite crear un nuevo mensaje de texto y enviarlo al número de teléfono del destinatario. `SendTextMessage` solo envía un mensaje SMS a un destinatario cada vez que se invoca.
- `sms-voice:DeleteVerifiedDestinationNumber`— Permite eliminar un número de teléfono de la lista de números verificados del Cuenta de AWS
- `sms-voice:VerifyDestinationNumber`: permite iniciar y completar el proceso de verificación de un número de teléfono que se utilizará en los servicios de mensajería SMS dentro de AWS.
- `sms-voice:DescribeAccountAttributes`: permite recuperar información detallada sobre los atributos de nivel de cuenta relacionados con los servicios de mensajería SMS internos dentro de AWS.
- `sms-voice:DescribeSpendLimits`: permite recuperar información sobre los límites de gasto asociados a los servicios de mensajería SMS dentro de la Cuenta de AWS.
- `sms-voice:DescribePhoneNumbers`: permite recuperar información sobre los números de teléfono asociados a los servicios de mensajería SMS dentro de la Cuenta de AWS .
- `sms-voice:SetTextMessageSpendLimitOverride`— Permite establecer o anular el límite de gasto en mensajes de texto SMS dentro del Cuenta de AWS
- `sms-voice:DescribeOptedOutNumbers`: permite recuperar una lista de números de teléfono que se han dado de baja en la recepción de mensajes SMS desde su cuenta de AWS .
- `sms-voice:DeleteOptedOutNumber`— Permite eliminar un número de teléfono de la lista de números excluidos del Cuenta de AWS

Política **`AmazonSNSFullAccess`** de ejemplo

```
{
    "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 los permisos de esta política, consulta Amazon SNSFull Access en la Referencia de políticas AWS gestionadas.

AWS política gestionada: Amazon SNSRead OnlyAccess

`AmazonSNSReadOnlyAccess` proporciona acceso completo a Amazon SNS mediante la AWS Management Console. Esta política también incluye las siguientes acciones de solo lectura para AWS End User Messaging SMS cuando se llame a través de Amazon SNS. Puede asociar esta política a sus usuarios, grupos y roles.

Detalles de los permisos

Los siguientes permisos se aplican únicamente cuando se utiliza Amazon SNS APIs:

- `sns:GetTopicAttributes`: permite recuperar los atributos de un tema de Amazon SNS. Esto incluye información como el ARN (Nombre del recurso de Amazon) del tema, la lista de suscriptores, las políticas de entrega, las políticas de control de acceso y cualquier otro metadato asociado al tema.
- `sns:List*`: permite realizar cualquier operación que comience por `List` para los recursos de Amazon SNS. Esto incluye permisos para mostrar varios elementos relacionados con Amazon SNS, como:
  - `sns:ListTopics`: permite recuperar una lista de todos los temas de Amazon SNS en la Cuenta de AWS.
  - `sns:ListSubscriptions`: permite recuperar una lista de todas las suscripciones a temas de Amazon SNS.
  - `sns:ListSubscriptionsByTopic`: permite mostrar todas las suscripciones de un tema específico de Amazon SNS.
  - `sns:ListPlatformApplications`: permite mostrar todas las aplicaciones de plataforma que se crean para las notificaciones push para móvil.
  - `sns:ListEndpointsByPlatformApplication`: permite mostrar todos los puntos de conexión asociados a una aplicación de plataforma.
- `sns:CheckIfPhoneNumberIsOptedOut`: permite comprobar si un número de teléfono específico se ha dado de baja de la recepción de mensajes SMS a través de Amazon SNS.
- `sns:GetEndpointAttributes`: permite recuperar los atributos de un punto de conexión asociado a una aplicación de plataforma de Amazon SNS. Esto podría incluir atributos como el estado de activación del punto de conexión, los datos de usuario personalizados y cualquier otro metadato asociado al punto de conexión.
- `sns:GetDataProtectionPolicy`: permite recuperar la política de protección de datos asociada a un tema de Amazon SNS.
- `sns:GetPlatformApplicationAttributes`: permite recuperar los atributos de una aplicación de plataforma de Amazon SNS. Las aplicaciones de plataforma se utilizan en Amazon SNS para enviar notificaciones push a dispositivos móviles a través de servicios como Apple Push Notification Service (APNS) o Firebase Cloud Messaging (FCM).
- `sns:GetSMSAttributes`: permite recuperar la configuración de SMS predeterminada para la Cuenta de AWS.

- `sns:GetSMSSandboxAccountStatus`: permite recuperar el estado actual del entorno de pruebas de SMS para su Cuenta de AWS.
- `sns:GetSubscriptionAttributes`: permite recuperar los atributos de una suscripción específica en un tema de Amazon SNS.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Le permite ver o recuperar una lista de números de teléfono que se han verificado para el envío de mensajes SMS en el Cuenta de AWS
- `sms-voice:DescribeAccountAttributes`: permite ver o recuperar información sobre los atributos de nivel de cuenta relacionados con los servicios de mensajería SMS internos dentro de AWS.
- `sms-voice:DescribeSpendLimits`: permite ver o recuperar información sobre los límites de gasto asociados a los servicios de mensajería SMS dentro de su Cuenta de AWS.
- `sms-voice:DescribePhoneNumbers`: permite ver o recuperar información sobre los números de teléfono usados para los servicios de mensajería SMS dentro de la Cuenta de AWS.
- `sms-voice:DescribeOptedOutNumbers`: permite ver o recuperar una lista de números de teléfono que se han dado de baja en la recepción de mensajes SMS de su Cuenta de AWS.

Política **`AmazonSNSReadOnlyAccess`** de ejemplo

```
{
    "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 los permisos de esta política, consulta Amazon SNSFull Access en la Referencia de políticas AWS gestionadas.

Amazon SNS actualiza las políticas gestionadas AWS

Consulta los detalles sobre las actualizaciones de las políticas AWS gestionadas de Amazon SNS desde que este servicio comenzó a realizar el seguimiento de estos cambios. Para obtener alertas automáticas sobre los cambios en esta página, suscríbase a la fuente RSS en la página de historial de documentos de Amazon SNS.

Cambio	Descripción	Fecha
AmazonSNSFullAccess: actualización de una política actual	Amazon SNS añadió nuevos permisos para permitir el acceso completo a Amazon SNS mediante la AWS Management Console.	24/09/2024

Cambio	Descripción	Fecha
Amazon SNSRead OnlyAccess: actualización de una política existente	Amazon SNS añadió nuevos permisos para permitir el acceso de solo lectura a Amazon SNS mediante la AWS Management Console.	24/09/2024
Amazon SNS comenzó a hacer un seguimiento de los cambios	Amazon SNS comenzó a realizar un seguimiento de los cambios en sus políticas AWS gestionadas.	27/08/2024

Acciones de políticas para Amazon SNS

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de las acciones de Amazon SNS, consulte Recursos definidos por Amazon Simple Notification Service en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Amazon SNS utilizan el siguiente prefijo antes de la acción:

```
sns
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
```

```
      "sns:action1",
      "sns:action2"
         ]
```

Para ver ejemplos de políticas basadas en identidad de Amazon SNS, consulte Ejemplos de políticas basadas en identidades de Amazon Simple Notification Service.

Recursos de políticas para Amazon SNS

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puedes hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utiliza un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de recursos de Amazon SNS y sus tipos ARNs, consulte Acciones definidas por Amazon Simple Notification Service en la Referencia de autorización del servicio. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Recursos definidos por Amazon Elastic Notification Service.

Para ver ejemplos de políticas basadas en identidad de Amazon SNS, consulte Ejemplos de políticas basadas en identidades de Amazon Simple Notification Service.

Claves de condición de políticas para Amazon SNS

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puedes crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puedes utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puedes conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulta Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para obtener una lista de las claves de condición de Amazon SNS, consulte Claves de condición de Amazon Simple Notification Service en la Referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Recursos definidos por Amazon Simple Notification Service.

Para ver ejemplos de políticas basadas en identidad de Amazon SNS, consulte Ejemplos de políticas basadas en identidades de Amazon Simple Notification Service.

ACLs en Amazon SNS

Soporta ACLs: No

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Amazon SNS

Compatibilidad con ABAC (etiquetas en las políticas): parcial

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulta Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Uso de credenciales temporales con Amazon SNS

Compatibilidad con credenciales temporales: sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluidas las que Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para obtener más

información sobre el cambio de roles, consulte Cambio de un usuario a un rol de IAM (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM.

Permisos de entidades principales entre servicios de Amazon SNS

Admite sesiones de acceso directo (FAS): sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en AWSél, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Roles de servicio para Amazon SNS

Compatibilidad con roles de servicio: sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Amazon SNS. Edite los roles de servicio solo cuando Amazon SNS proporcione orientación para hacerlo.

Roles vinculado a servicios para Amazon SNS

Compatibilidad con roles vinculados al servicio: no

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puedes asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puedes ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulta Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Ejemplos de políticas basadas en identidades de Amazon Simple Notification Service

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar recursos de Amazon SNS. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o la AWS API. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puedes añadir las políticas de IAM a roles y los usuarios puedes asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Amazon SNS, incluido el formato de cada uno de los tipos de recursos, consulte Acciones, recursos y claves de condición de Amazon Simple Notification Service en la Referencia de autorización del servicio. ARNs

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon SNS de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas administradas que otorgan permisos para muchos casos de uso comunes.AWS Están

disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulta las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.

- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulta Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulta Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de Amazon SNS

Para acceder a la consola de Amazon Simple Notification Service, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de Amazon SNS que tiene en su cuenta. Cuenta de AWS Si crea una política basada en identidades que sea

más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No necesita conceder permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite el acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles puedan seguir utilizando la consola de Amazon SNS, adjunte también la política gestionada *ReadOnly* AWS o de Amazon *ConsoleAccess* SNS a las entidades. Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puedes establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): SCPs son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y administrar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluido cada usuario Cuenta de AWS raíz. Para obtener más información sobre Organizations SCPs, consulte las políticas de control de servicios en la Guía del AWS Organizations usuario.
- Políticas de control de recursos (RCPs): RCPs son políticas de JSON que puedes usar para establecer los permisos máximos disponibles para los recursos de tus cuentas sin actualizar las políticas de IAM asociadas a cada recurso que poseas. El RCP limita los permisos de los recursos en las cuentas de los miembros y puede afectar a los permisos efectivos de las

identidades, incluido el usuario Cuenta de AWS raíz, independientemente de si pertenecen o no a su organización. Para obtener más información sobre Organizations e RCPs incluir una lista de Servicios de AWS ese apoyo RCPs, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.

- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también puedes proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas gestionadas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o 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 de Amazon SNS basadas en identidades

Compatibilidad con las políticas basadas en identidad: sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está asociada. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades para Amazon SNS

Para ver ejemplos de políticas basadas en identidad de Amazon SNS, consulte Ejemplos de políticas basadas en identidades de Amazon Simple Notification Service.

Políticas basadas en recursos de Amazon SNS

Compatibilidad con las políticas basadas en recursos	Sí

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios puede utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política basada en recursos concede acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para obtener más información, consulte Cross account resource access in IAM en la Guía del usuario de IAM.

Uso de políticas basadas en identidades con Amazon SNS

Amazon Simple Notification Service se integra con AWS Identity and Access Management (IAM) para que pueda especificar qué acciones de Amazon SNS puede realizar un usuario suyo con los Cuenta de AWS recursos de Amazon SNS. Puede especificar un tema determinado de la política. Por ejemplo, puede utilizar variables cuando crea una política de IAM a fin de conceder permiso a determinados usuarios de su organización para utilizar la acción `Publish` en temas específicos de su Cuenta de AWS. Para obtener más información, consulte Variables de las políticas en la Guía del usuario de IAM.

> **Important**
>
> El uso de Amazon SNS con IAM no cambia la forma de utilizar Amazon SNS. No hay cambios en las acciones de Amazon SNS ni acciones nuevas de Amazon SNS relacionados con los usuarios y el control de acceso.

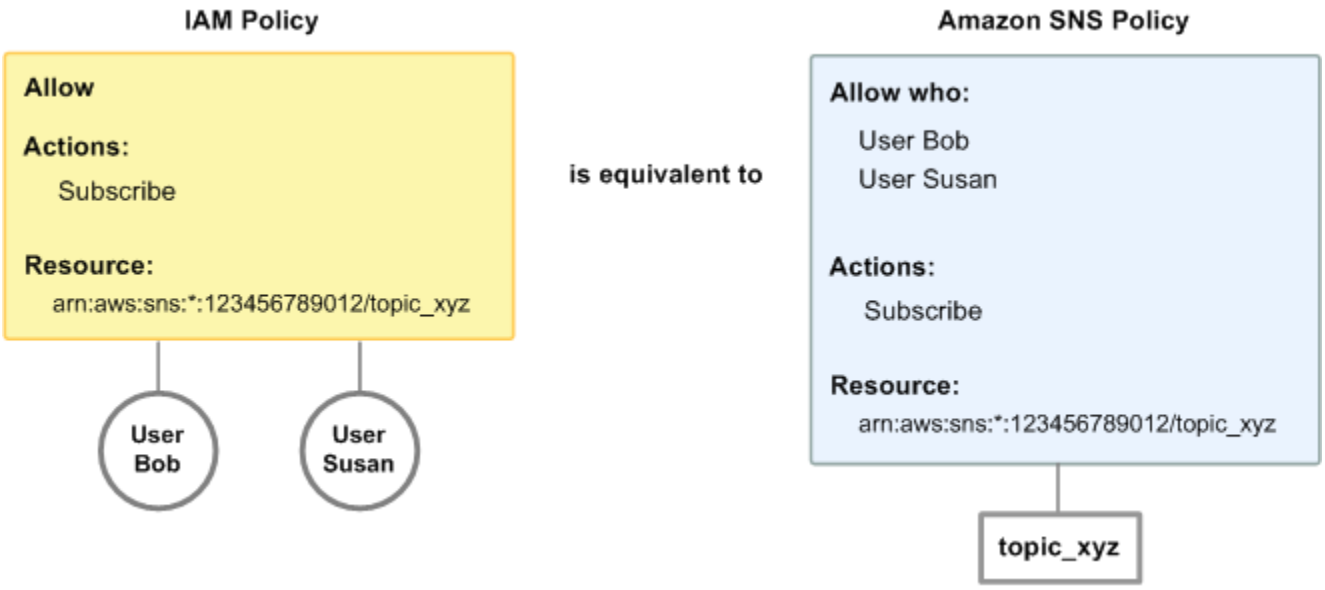
Para ver ejemplos de políticas que abarcan acciones y recursos de Amazon SNS, consulte Ejemplos de políticas para Amazon SNS.

Uso en conjunto de políticas y roles de IAM y Amazon SNS

Las políticas de IAM se utilizan para restringir el acceso de los usuarios a acciones y temas de Amazon SNS. Una política de IAM puede restringir el acceso únicamente a los usuarios de su AWS cuenta, no a otros. Cuentas de AWS

Puede utilizar una política de Amazon SNS con un tema determinado para restringir quién puede trabajar en ese tema (por ejemplo, quién puede publicar mensajes en el tema, quién puede suscribirse al tema, etcétera). Las políticas de Amazon SNS pueden conceder acceso a otros Cuentas de AWS usuarios o a los propios usuarios. Cuenta de AWS

Con el fin de otorgar a sus usuarios permisos para sus temas de Amazon SNS, puede utilizar políticas de IAM, políticas de Amazon SNS o ambas. La mayoría de las veces, puede conseguir los mismos resultados con ambas. Por ejemplo, en el siguiente diagrama se muestra una política de IAM y una política de Amazon SNS equivalentes. La política de IAM permite la acción de Amazon `Subscribe` SNS para el tema denominado topic_xyz en Cuenta de AWS tu página. La política de IAM está vinculada a los usuarios Bob y Susan (lo que significa que Bob y Susan tienen los permisos establecidos en la política). Con la política de Amazon SNS, también se les concede a Bob y Susan permiso para obtener acceso a `Subscribe` en relación con topic_xyz.



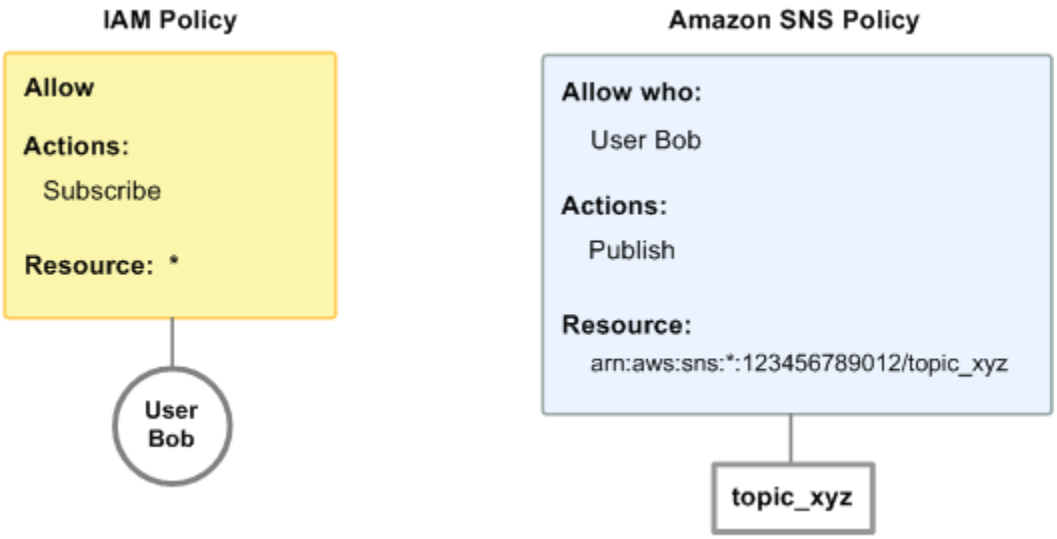
> **Note**
>
> En el ejemplo anterior se muestran políticas sencillas sin condiciones. Puede especificar una condición determinada en cualquiera de las dos políticas y obtener el mismo resultado.

Hay una diferencia entre las políticas de AWS IAM y Amazon SNS: el sistema de políticas de Amazon SNS te permite conceder permisos a Cuentas de AWS otras personas, mientras que la política de IAM no.

Basándose en sus necesidades, decida el uso que quiere hacer de ambos sistemas para administrar los permisos. Los siguientes ejemplos muestran cómo funcionan conjuntamente los dos sistemas de política.

Example 1

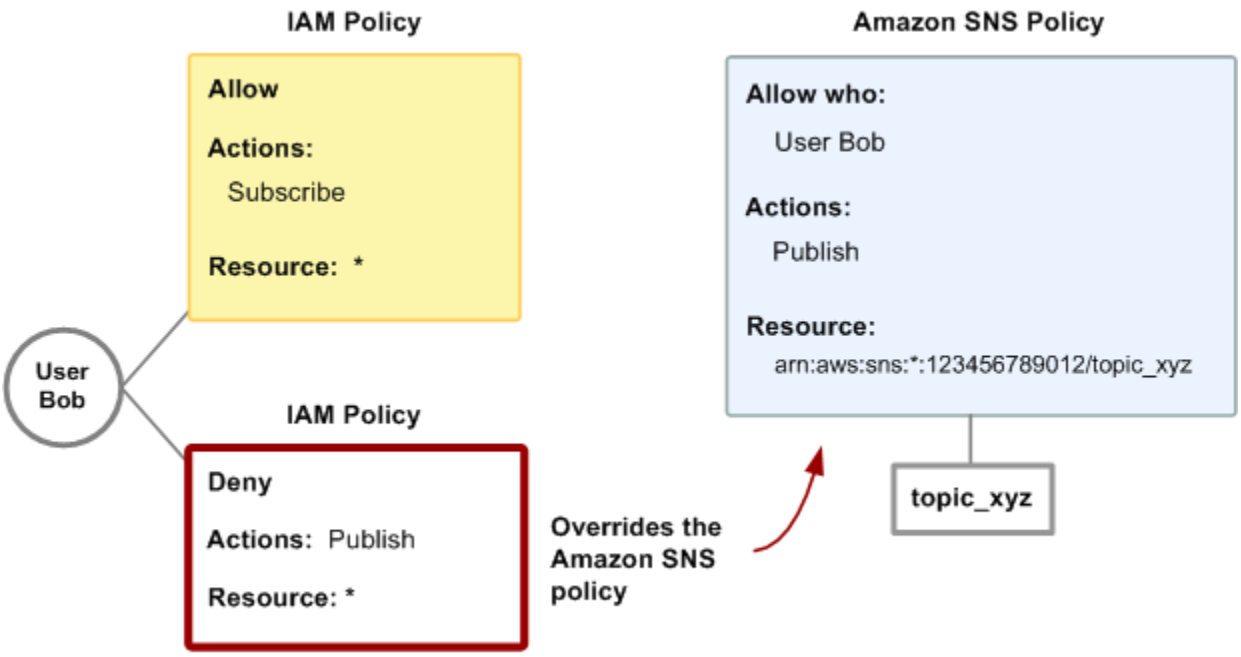
En este ejemplo, se aplica a Bob tanto una política de IAM como una política de Amazon SNS. La política de IAM le otorga permiso para hablar `Subscribe` sobre cualquiera Cuenta de AWS de los temas, mientras que la política de Amazon SNS le otorga permiso para `Publish` usarlo en un tema específico (topic_xyz). El siguiente diagrama ilustra este concepto.



Si Bob enviara una solicitud para suscribirse a cualquier tema de la AWS cuenta, la política de IAM permitiría esa acción. Si Bob quiere enviar una solicitud para publicar un mensaje en topic_xyz, la política de Amazon SNS permitirá la acción.

Example 2

En este ejemplo, nos basamos en el ejemplo 1 (en el que se aplican dos políticas a Bob). Supongamos que Bob publica mensajes en topic_xyz que no debería haber publicado y usted decide quitarle por completo su capacidad para publicar en temas. Para ello, lo más fácil es agregar una política de IAM que le deniegue acceso a la acción `Publish` en todos los temas. Esta tercera política anula la política de Amazon SNS que anteriormente le daba permiso para publicar en topic_xyz, ya que una denegación explícita siempre anula una instrucción “permitir” (para obtener más información sobre la lógica de evaluación de políticas, consulte Lógica de evaluación). El siguiente diagrama ilustra este concepto.



Para ver ejemplos de políticas que abarcan acciones y recursos de Amazon SNS, consulte Ejemplos de políticas para Amazon SNS.

Formato de ARN de recursos de Amazon SNS

Para Amazon SNS, los temas son el único tipo de recurso que puede especificar en una política. A continuación se muestra el formato de nombre de recurso de Amazon (ARN) para los temas.

```
arn:aws:sns:region:account_ID:topic_name
```

Para obtener más información al respecto ARNs, consulte la ARNsGuía del usuario de IAM.

Example

El siguiente es un ARN de un tema denominado MyTopic en la región us-east-2, que pertenece a 123456789012. Cuenta de AWS

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

Example

Si tuviera un tema nombrado MyTopic en cada una de las distintas regiones que admite Amazon SNS, puede especificar los temas con el siguiente ARN.

```
arn:aws:sns:*:123456789012:MyTopic
```

Puede utilizar los caracteres comodín * e ? en el nombre del tema. Por ejemplo, el ejemplo siguiente podría hacer referencia a todos los temas creados por Bob a los que ha puesto el prefijo `bob_`.

```
arn:aws:sns:*:123456789012:bob_*
```

Para facilitarle su tarea, cuando crea un tema, Amazon SNS devuelve el ARN del tema en la respuesta.

Acciones de API de Amazon SNS

En una política de IAM, puede especificar cualquier acción que Amazon SNS ofrezca. Sin embargo, las acciones `ConfirmSubscription` y `Unsubscribe` no requieren autenticación, lo que significa que, incluso si especifica dichas acciones en una política, IAM no restringirá el acceso de los usuarios a estas acciones.

Cada acción que especifique en una política debe ir prefijada con la cadena en minúsculas `sns:`. Para especificar todas las acciones de Amazon SNS, utilizaría, por ejemplo, `sns:*`. Para obtener una lista de las acciones, vaya a la Referencia de la API de Amazon Simple Notification Service.

Claves de política de Amazon SNS

Amazon SNS implementa las siguientes claves de política AWS amplias, además de algunas claves específicas del servicio.

Para ver una lista de las claves de condición compatibles con cada una de ellas Servicio de AWS, consulte las acciones, los recursos y las claves de condición de la Guía del Servicios de AWS usuario de IAM. Para obtener una lista de las claves de condición que se pueden utilizar en varias Servicios de AWS, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Amazon SNS utiliza las siguientes claves específicas de servicio. Utilice estas claves en políticas que restringen el acceso a solicitudes `Subscribe`.

- sns:Endpoint: la URL, la dirección de correo electrónico o el ARN de una solicitud `Subscribe` o una suscripción confirmada anteriormente. Utilícela con las condiciones de la cadena (consulte Ejemplos de políticas para Amazon SNS) para restringir el acceso a puntos de enlace específicos (por ejemplo, *@yourcompany.com).

- sns:Protocol: el valor `protocol` de una solicitud `Subscribe` o de una suscripción confirmada anteriormente. Utilícela con las condiciones de la cadena (consulte Ejemplos de políticas para Amazon SNS) para restringir la publicación en protocolos de entrega (por ejemplo, https).

Ejemplos de políticas para Amazon SNS

En esta sección, se muestran varias políticas sencillas para controlar el acceso de los usuarios a Amazon SNS.

> **Note**
>
> En el futuro, Amazon SNS podría agregar nuevas acciones que deberán, lógicamente, incluirse en una de las políticas siguientes, en función de los objetivos indicados en esta.

Example 1: Permitir a un grupo crear y administrar temas

En este ejemplo, creamos una política que concede acceso a `CreateTopic`, `ListTopics`, `SetTopicAttributes` y `DeleteTopic`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: Permitir que el grupo de TI publique mensajes en un tema determinado

En este ejemplo, creamos un grupo de TI y asignamos una política que concede acceso a `Publish` en el tema de interés específico.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
```

```
  }]
}
```

Example 3: Ofrezca a los usuarios la Cuenta de AWS posibilidad de suscribirse a los temas

En este ejemplo, creamos una política que concede acceso a la acción `Subscribe`, con condiciones de coincidencia de la cadena para las claves de política `sns:Protocol` y `sns:Endpoint`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "sns:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: Permitir a un socio publicar mensajes en un tema determinado

Puede utilizar una política de Amazon SNS o una política de IAM para permitir a un socio publicar en un tema concreto. Si tu socio tiene una Cuenta de AWS, podría ser más fácil usar una política de Amazon SNS. Sin embargo, cualquier miembro de la empresa del socio que posea las credenciales AWS de seguridad podría publicar mensajes sobre el tema. En este ejemplo se presupone que quiere limitar el acceso a una determinada persona (o aplicación). Para ello, debe tratar al socio como un usuario de su propia compañía y utilizar una política de IAM en vez de una política de Amazon SNS.

Para este ejemplo, creamos un grupo denominado WidgetCo que representa a la empresa asociada; creamos un usuario para la persona (o aplicación) específica de la empresa asociada que necesita acceso y, a continuación, incluimos al usuario en el grupo.

A continuación, adjuntamos una política que concede al grupo `Publish` acceso al tema específico mencionado WidgetPartnerTopic.

También queremos evitar que el WidgetCo grupo haga cualquier otra cosa con los temas, por lo que añadimos una declaración que deniega el permiso a cualquier acción de Amazon SNS que no sea `Publish` sobre cualquier tema que no sea. WidgetPartnerTopic Este paso es necesario solo si existe una política amplia en cualquier otra parte del sistema que concede a los usuarios un acceso amplio a 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"
    }
  ]
}
```

Administración de políticas de IAM personalizadas de Amazon SNS

Las políticas de IAM personalizadas le permiten especificar permisos para usuarios, grupos o roles individuales de IAM, concediendo o restringiendo el acceso a AWS recursos y acciones específicos. Al administrar los recursos de Amazon SNS, las políticas de IAM personalizadas le permiten personalizar los permisos de acceso de acuerdo con los requisitos operativos y de seguridad de su organización.

Siga estos pasos para administrar políticas de IAM personalizadas para Amazon SNS:

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en. https://console.aws.amazon.com/iam/
2. En el panel de navegación, elija Políticas.
3. Para crear una nueva política de IAM personalizada, selecciona Crear política y elige SNS. Para editar una política existente, selecciónela de la lista y elija Editar política.
4. En el editor de políticas, defina los permisos para acceder a los recursos de Amazon SNS. Puede especificar acciones, recursos y condiciones en función de sus requisitos específicos.
5. Para conceder permisos para las acciones de Amazon SNS, incluya las acciones de Amazon SNS pertinentes, como `sns:Publish`, `sns:Subscribe` y `sns:DeleteTopic`, en su política

de IAM. Defina el ARN (Nombre de recurso de Amazon) de los temas de Amazon SNS a los que se aplican los permisos.

6. Especifique los usuarios, grupos o roles de IAM a los que se debe asociar la política. Puede asociar la política directamente a los usuarios o grupos de IAM o asociarla a los roles de IAM usados por Servicios de AWS o las aplicaciones.
7. Revise la configuración de las políticas de IAM para asegurarse de que se ajusta a sus requisitos de control de acceso. Una vez verificadas, guarde los cambios realizados en ellas.
8. Asocie la política de IAM personalizada a los usuarios, grupos o roles de IAM pertinentes de su Cuenta de AWS. Esto les concede los permisos definidos en la política para administrar los recursos de Amazon SNS.

Uso de credenciales de seguridad temporales con Amazon SNS

AWS Identity and Access Management (IAM) le permite conceder credenciales de seguridad temporales a los usuarios y aplicaciones que necesitan acceder a sus AWS recursos. Estas credenciales de seguridad temporales se utilizan principalmente para los roles de IAM y el acceso federado a través de protocolos estándar del sector, como SAML y OpenID Connect (OIDC).

Para gestionar de forma eficaz el acceso a AWS los recursos, es fundamental comprender los siguientes conceptos clave:

- Funciones de IAM: las funciones se utilizan para delegar el acceso a AWS los recursos. Las funciones pueden ser asumidas por entidades como EC2 instancias de Amazon, funciones de Lambda o usuarios de otras entidades. Cuentas de AWS
- Usuarios federados: son usuarios autenticados mediante proveedores de identidad externos (IdPs) mediante SAML u OIDC. Se recomienda el acceso federado para los usuarios humanos, mientras que los roles de IAM deben utilizarse para las aplicaciones de software.
- Roles en cualquier lugar: para las aplicaciones externas que requieren AWS acceso, puede utilizar IAM Roles Anywhere para gestionar el acceso de forma segura sin crear credenciales a largo plazo.

Puede utilizar estas credenciales de seguridad temporales para realizar solicitudes a Amazon SNS. Las bibliotecas de API SDKs y las bibliotecas de API calculan la firma necesaria con estas credenciales para autenticar sus solicitudes. Amazon SNS rechazará las solicitudes con credenciales caducadas.

Para obtener más información sobre las credenciales de seguridad temporales, consulte Uso de roles de IAM y Proporcionar acceso a usuarios autenticados externamente (federación de identidades) en la Guía del usuario de IAM.

Example Ejemplo de solicitud HTTPS

El siguiente ejemplo muestra cómo autenticar una solicitud de Amazon SNS mediante credenciales de seguridad temporales AWS Security Token Service obtenidas de (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>
```

Pasos para autenticar la solicitud

1. Obtenga credenciales de seguridad temporales: utilice AWS STS para asumir un rol u obtener credenciales de usuario federado. Esto le proporcionará un ID de clave de acceso, una clave de acceso secreta y un token de seguridad.
2. Cree la solicitud: incluya los parámetros necesarios para su acción de Amazon SNS (por ejemplo, CreateTopic) y asegúrese de utilizar HTTPS para una comunicación segura.
3. Firme la solicitud: utilice el proceso de AWS Signature Version 4 para firmar la solicitud. Esto implica crear una solicitud canónica y, a continuación string-to-sign, calcular la firma. Para obtener más información sobre la versión 4 de AWS Signature, consulte Utilizar la firma de la versión 4 de Signature en la Guía del usuario de Amazon EBS.
4. Envíe la solicitud: incluya el X-Amz-Security-Token en el encabezado de la solicitud para pasar las credenciales de seguridad temporales a Amazon SNS.

Permisos de la API de Amazon SNS: referencia de recursos y acciones

En la siguiente lista, se proporciona información específica sobre la implementación por parte de Amazon SNS del control de acceso:

- Cada política debe cubrir únicamente un único tema (a la hora de escribir una política, no incluya instrucciones que abarquen diferentes temas).
- Cada política debe tener un `Id` de política único.
- Cada instrucción de una política debe tener un `sid` de instrucción exclusivo.

Cuotas de políticas

En la siguiente tabla se muestran las cuotas máximas para una instrucción de política.

Nombre	Cuota máxima
Bytes	30 kb
Instrucciones	100
Entidades principales	De 1 a 200 (0 no es válido).
Recurso	1 (0 no es válido. El valor debe coincidir con el ARN del tema de la política).

Acciones de políticas de Amazon SNS válidas

Amazon SNS es compatible con las acciones que se muestran en la siguiente tabla.

Acción	Descripción
sns: AddPermission	Da permiso para añadir permisos a la política del tema.
sns: DeleteTopic	Da permiso para eliminar un tema.
sns: GetDataProtectionPolicy	Otorga permiso para recuperar la política de protección de datos de un tema
sns: GetTopicAttributes	Da permiso para recibir todos los atributos del tema.
sns: ListSubscriptionsByTopic	Da permiso para recuperar todas las suscripciones a un determinado tema.

Acción	Descripción
sns: ListTagsForResource	Otorga permiso para mostrar todas las etiquetas agregadas a un tema específico.
sns:Publish	Concede permiso para publicar y publicar por lotes en un tema o punto de conexión. Para obtener más información, consulte Publicar y PublishBatchen la referencia de la API de Amazon Simple Notification Service.
sns: PutDataProtectionPolicy	Otorga permiso para definir la política de protección de datos de un tema
sns: RemovePermission	Da permiso para eliminar cualquier permiso de la política del tema.
sns: SetTopicAttributes	Da permiso para establecer los atributos de un tema.
sns:Subscribe	Da permiso para suscribirse a un tema.

Claves específicas del servicio

Amazon SNS utiliza las siguientes claves específicas de servicio. Puede utilizarlas en políticas que restrinjan el acceso a solicitudes `Subscribe`.

- sns:Endpoint: la URL, la dirección de correo electrónico o el ARN de una solicitud `Subscribe` o una suscripción confirmada anteriormente. Utilícela con las condiciones de la cadena (consulte Ejemplos de políticas para Amazon SNS) para restringir el acceso a puntos de enlace específicos (por ejemplo, *@example.com).
- sns:Protocol: el valor `protocol` de una solicitud `Subscribe` o de una suscripción confirmada anteriormente. Utilícela con las condiciones de la cadena (consulte Ejemplos de políticas para Amazon SNS) para restringir la publicación en protocolos de entrega (por ejemplo, https).

> **Important**
>
> Cuando utiliza una política para controlar el acceso por sns:Endpoint, debe ser consciente de que los problemas de DNS podrían afectar posteriormente a la resolución de nombres del punto de enlace.

Solución de problemas de identidad y acceso de Amazon Simple Notification Service

Utilice la siguiente información para diagnosticar y solucionar los problemas habituales que pueden surgir cuando se trabaja con Amazon SNS e IAM.

No tengo autorización para realizar una acción en Amazon SNS

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `sns:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

En este caso, la política de Mateo se debe actualizar para permitirle acceder al recurso *`my-example-widget`* mediante la acción `sns:`*`GetWidget`*.

Si necesita ayuda, póngase en contacto con su AWS administrador. El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibe un error que indica que no tiene autorización para llevar a cabo la acción `iam:PassRole`, las políticas se deben actualizar para permitirle pasar un rol a Amazon SNS.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Amazon SNS. Sin embargo, la acción requiere

que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Amazon SNS

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admiten políticas basadas en recursos o listas de control de acceso (ACLs), puede usar esas políticas para permitir que las personas accedan a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon SNS admite estas características, consulte Cómo funciona Amazon SNS con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro de su propiedad en la Cuenta de AWS Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registro y monitoreo en Amazon SNS

Amazon SNS le permite realizar un seguimiento y supervisar la actividad de mensajería mediante el registro de las llamadas a la API CloudTrail y la supervisión de los temas con. CloudWatch Estas herramientas le ayudan a obtener información sobre la entrega de mensajes, solucionar problemas y garantizar el buen estado de sus flujos de trabajo de mensajería. Este tema cubre lo siguiente:

- Registro de llamadas a la API de AWS SNS mediante AWS CloudTrail. Este registro le permite realizar un seguimiento de las acciones realizadas en sus temas de Amazon SNS, como la creación de temas, la administración de suscripciones y la publicación de mensajes. Al analizar CloudTrail los registros, puede identificar quién realizó solicitudes de API específicas y cuándo se hicieron esas solicitudes, lo que le ayuda a auditar y solucionar problemas de uso de Amazon SNS.
- Supervisión de temas de Amazon SNS mediante CloudWatch. CloudWatch proporciona métricas que le permiten observar el rendimiento y el estado de sus temas de Amazon SNS en tiempo real. Configure alarmas en función de estas métricas, lo que le permitirá responder con prontitud a cualquier anomalía, como errores en la entrega o una latencia elevada de los mensajes. Esta capacidad de supervisión garantiza que pueda mantener la fiabilidad de su sistema de mensajería basado en el SNS al abordar de forma proactiva los posibles problemas.

Registro de llamadas a la API de AWS SNS mediante AWS CloudTrail

AWS SNS está integrado con AWS CloudTrailun servicio que proporciona un registro de las acciones realizadas por un usuario, rol o un. Servicio de AWS CloudTrail captura todas las llamadas a la API de SNS como eventos. Las llamadas capturadas incluyen llamadas desde la consola de SNS y llamadas en código a las operaciones de la API de SNS. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a SNS, la dirección IP desde la que se realizó la solicitud, cuándo se realizó y detalles adicionales.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales del usuario raíz o del usuario.
- Si la solicitud se realizó en nombre de un usuario de IAM Identity Center.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro Servicio de AWS.

CloudTrail está activa en tu cuenta Cuenta de AWS cuando creas la cuenta y tienes acceso automáticamente al historial de CloudTrail eventos. El historial de CloudTrail eventos proporciona un registro visible, consultable, descargable e inmutable de los últimos 90 días de eventos de gestión registrados en un. Región de AWSPara obtener más información, consulte Cómo trabajar con el historial de CloudTrail eventos en la Guía del usuario.AWS CloudTrail La visualización del historial de eventos no conlleva ningún CloudTrail cargo.

Para tener un registro continuo de los eventos de Cuenta de AWS los últimos 90 días, crea un almacén de datos de eventos de senderos o CloudTraillagos.

CloudTrail senderos

Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. Todos los senderos creados con él AWS Management Console son multirregionales. Puede crear un registro de seguimiento de una sola región o de varias regiones mediante la AWS CLI. Se recomienda crear un sendero multirregional, ya que puedes capturar toda la actividad de tu Regiones de AWS cuenta. Si crea un registro de seguimiento de una sola región, solo podrá ver los eventos registrados en la Región de AWS del registro de seguimiento. Para obtener más información acerca de los registros de seguimiento, consulte Creación de un registro de seguimiento para su Cuenta de AWS y Creación de un registro de seguimiento para una organización en la Guía del usuario de AWS CloudTrail .

Puede enviar una copia de sus eventos de administración en curso a su bucket de Amazon S3 sin coste alguno CloudTrail mediante la creación de una ruta; sin embargo, hay cargos por almacenamiento en Amazon S3. Para obtener más información sobre CloudTrail los precios, consulte AWS CloudTrail Precios. Para obtener información acerca de los precios de Amazon S3, consulte Precios de Amazon S3.

CloudTrail Almacenes de datos de eventos en Lake

CloudTrail Lake le permite ejecutar consultas basadas en SQL en sus eventos. CloudTrail Lake convierte los eventos existentes en formato JSON basado en filas al formato Apache ORC. ORC es un formato de almacenamiento en columnas optimizado para una recuperación rápida de datos. Los eventos se agregan en almacenes de datos de eventos, que son recopilaciones inmutables de eventos en función de criterios que se seleccionan aplicando selectores de eventos avanzados. Los selectores que se aplican a un almacén de datos de eventos controlan los eventos que perduran y están disponibles para la consulta. Para obtener más información sobre CloudTrail Lake, consulte Cómo trabajar con AWS CloudTrail Lake en la Guía del AWS CloudTrail usuario.

CloudTrail Los almacenes de datos y las consultas sobre eventos de Lake conllevan costes. Cuando crea un almacén de datos de eventos, debe elegir la opción de precios que desee utilizar para él. La opción de precios determina el costo de la incorporación y el almacenamiento de los eventos, así como el período de retención predeterminado y máximo del almacén de datos de eventos. Para obtener más información sobre CloudTrail los precios, consulte AWS CloudTrail Precios.

Eventos de datos de SNS en CloudTrail

Los eventos de datos proporcionan información sobre las operaciones de recursos realizadas en o dentro de un recurso (por ejemplo, leer o escribir en un objeto de Amazon S3). Se denominan también operaciones del plano de datos. Los eventos de datos suelen ser actividades de gran volumen. De forma predeterminada, CloudTrail no registra los eventos de datos. El historial de CloudTrail eventos no registra los eventos de datos.

Se aplican cargos adicionales a los eventos de datos. Para obtener más información sobre CloudTrail los precios, consulta AWS CloudTrail Precios.

Puede registrar eventos de datos para los tipos de recursos de SNS mediante la CloudTrail consola o las operaciones de la CloudTrail API. AWS CLI Para obtener más información sobre cómo registrar los eventos de datos, consulte Registro de eventos de datos con la AWS Management Console y Registro de eventos de datos con la AWS Command Line Interface en la Guía del usuario de AWS CloudTrail .

En la siguiente tabla se enumeran los tipos de recursos de SNS para los que puede registrar eventos de datos. La columna Tipo de evento de datos (consola) muestra el valor que se puede elegir en la lista de tipos de eventos de datos de la CloudTrail consola. La columna de valores resources.type muestra el `resources.type` valor, que se especificaría al configurar los selectores de eventos avanzados mediante o. AWS CLI CloudTrail APIs La CloudTrail columna Datos APIs registrados muestra las llamadas a la API registradas CloudTrail para el tipo de recurso.

Tipo de evento de datos (consola)	resources.type value	Datos APIs registrados en CloudTrail
Tema de SNS	`AWS::SNS::Topic`	• `Publish` • `PublishBatch`

Tipo de evento de datos (consola)	resources.type value	Datos APIs registrados en CloudTrail
Punto de conexión de la plataforma de SNS	`AWS::SNS::PlatformEndpoint`	• Publish Para obtener más información, consulta AdvancedEventSelector la referencia de la AWS CloudTrail API.

> **Note**
>
> El tipo de recurso SNS no `AWS::SNS::PhoneNumber` está registrado por CloudTrail.

Puede configurar selectores de eventos avanzados para filtrar según los campos `eventName`, `readOnly` y `resources.ARN` y así registrar solo los eventos que son importantes para usted. Para obtener más información sobre estos campos, consulte AdvancedFieldSelector en la Referencia de la API de AWS CloudTrail .

Para obtener información sobre el registro de eventos de datos, consulte Registrar eventos de datos con el AWS Management Console y Registrar eventos de datos con el AWS CLI en la Guía del CloudTrail usuario.

Eventos de administración de SNS en CloudTrail

Los eventos de administración proporcionan información sobre las operaciones de administración que se llevan a cabo en los recursos de su Cuenta de AWS empresa. Se denominan también operaciones del plano de control. De forma predeterminada, CloudTrail registra los eventos de administración.

AWS El SNS registra las siguientes operaciones del plano de control del SNS CloudTrail como eventos de administración.

- `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**
>
> Si no ha iniciado sesión en Amazon Web Services (modo no autenticado) y se invocan Unsubscribelas acciones ConfirmSubscriptiono, no se iniciará sesión en ellas. CloudTrail Por ejemplo, al elegir el vínculo proporcionado en una notificación por correo electrónico para confirmar la suscripción pendiente en un tema, se invoca la acción `ConfirmSubscription` en modo sin autenticar. En este ejemplo, no se registraría la `ConfirmSubscription` acción. CloudTrail

Ejemplos de eventos de SNS

Un evento representa una solicitud única de cualquier fuente e incluye información sobre la operación de API solicitada, la fecha y la hora de la operación, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que los eventos no aparecen en ningún orden específico.

En el siguiente ejemplo **`ListTopics`**, **`CreateTopic`** se muestra un CloudTrail evento que demuestra las **`DeleteTopic`** acciones y.


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

En el siguiente ejemplo, se muestra un CloudTrail evento que demuestra la **`Publish`** acción.

```
{
  "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"
  }
}
```

En el siguiente ejemplo, se muestra un CloudTrail evento que demuestra la **`PublishBatch`** acción.

```
{
  "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 obtener información sobre el contenido de los CloudTrail registros, consulte el contenido de los CloudTrail registros en la Guía del AWS CloudTrail usuario.

Supervisión de temas de Amazon SNS mediante CloudWatch

Amazon SNS y Amazon CloudWatch están integrados para que pueda recopilar, ver y analizar las métricas de cada notificación de Amazon SNS activa. Una vez que haya configurado CloudWatch Amazon SNS, podrá obtener una mejor visión del rendimiento de los temas, las notificaciones push y las entregas de SMS de Amazon SNS. Por ejemplo, puede configurar una alarma que le envíe una notificación por correo electrónico si se llega a un umbral especificado en una métrica de Amazon SNS, como `NumberOfNotificationsFailed`. Para obtener una lista de todas las métricas a las que envía Amazon SNS CloudWatch, consulte. Métricas de Amazon SNS Para obtener más información sobre las notificaciones push de Amazon SNS, consulte Envío de notificaciones push para móvil con Amazon SNS.

> **Note**
>
> Las métricas que configure CloudWatch para sus temas de Amazon SNS se recopilan y actualizan automáticamente a CloudWatch intervalos de 1 minuto. Estas métricas se recopilan sobre todos los temas que cumplen con las CloudWatch pautas para mantenerse activo. Un tema se considera activo hasta seis horas después de la última actividad (es decir, cualquier llamada a la API) sobre el tema. CloudWatch
> Las métricas de Amazon SNS incluidas en las que se informa son gratuitas CloudWatch; se proporcionan como parte del servicio Amazon SNS.

Ver CloudWatch las métricas de Amazon SNS

Puede supervisar las métricas de Amazon SNS mediante la CloudWatch consola, la propia interfaz CloudWatch de línea de comandos (CLI) o mediante programación mediante la API. CloudWatch Los procedimientos siguientes muestran cómo aceptar las métricas mediante la AWS Management Console.

Para ver las métricas mediante la consola CloudWatch

1. Inicie sesión en la consola de CloudWatch.
2. En el panel de navegación, elija Metrics.
3. En la pestaña All metrics (Todas las métricas), elija SNS y, a continuación, elija una de las dimensiones siguientes:

   - Country, SMS Type (País, tipo de SMS)

- PhoneNumber
- Topic Metrics (Métricas del tema)
- Metrics with no dimensions (Métricas sin dimensiones)

4. Para ver más detalles, elija un elemento específico. Por ejemplo, si elige Métricas de tema y, a continuación NumberOfMessagesPublished, elige, se mostrará el número medio de mensajes de Amazon SNS publicados durante un período de 1 minuto en un intervalo de tiempo de 6 horas.
5. Para ver las métricas de uso de Amazon SNS, en la pestaña All metrics (Todas las métricas), elija Usage (Uso) y seleccione una opción en Target Amazon SNS usage metric (Métrica de uso de Amazon SNS de destino) (por ejemplo, `NumberOfMessagesPublishedPerAccount`).

Configurar CloudWatch alarmas para las métricas de Amazon SNS

CloudWatch también permite configurar alarmas cuando se alcanza un umbral para una métrica. Por ejemplo, puede configurar una alarma para la métrica NumberOfNotificationsFailed, de modo que, cuando se alcance el umbral especificado dentro del período de muestreo, se le envíe una notificación por correo electrónico para informarle del evento.

Para configurar las alarmas mediante la CloudWatch consola

1. Inicie sesión en AWS Management Console y abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Elija Alarms (Alarmas) y, a continuación, seleccione el botón Create Alarm (Crear alarma). Esto lanzará el asistente Create Alarm (Crear alarma).
3. Desplácese por las métricas de Amazon SNS para localizar la métrica en la que desea colocar una alarma. Seleccione la métrica para crear una alarma y elija Continue (Continuar).
4. Rellene los valores Name (Nombre), Description (Descripción), Threshold (Umbral) y Time (Fecha y hora) de la métrica y elija Continue (Continuar).
5. Elija Alarm (Alarma) como estado de alarma. Si CloudWatch quieres enviarte un correo electrónico cuando se alcance el estado de alarma, selecciona un tema existente de Amazon SNS o selecciona Crear nuevo tema de correo electrónico. Si elige Create New Email Topic (Crear nuevo tema de correo electrónico), puede definir el nombre y las direcciones de correo electrónico de un tema nuevo. Esta lista se guardará y aparecerá en el cuadro desplegable para futuras alarmas. Elija Continuar.

> **Note**
>
> Si utiliza Crear nuevo tema de correo electrónico para crear un tema nuevo de Amazon SNS, deberá verificar las direcciones de correo electrónico para que puedan recibir las notificaciones. Los correos electrónicos solo se envían cuando la alarma entra en estado de alarma. Si este cambio en el estado de la alarma se produce antes de que se verifiquen las direcciones de correo electrónico, no recibirá una notificación.

6. En este momento, el asistente Create Alarm (Crear alarma) le da la oportunidad de revisar la alarma que está a punto de crear. Si necesita hacer algún cambio, puede utilizar los enlaces Edit (Editar) de la derecha. Cuando esté satisfecho, elija Create Alarm (Crear alarma).

Para obtener más información sobre el uso CloudWatch y las alarmas, consulte la CloudWatchdocumentación.

Métricas de Amazon SNS

Amazon SNS envía las siguientes métricas a. CloudWatch

Espacio de nombres	Métrica	Descripción
AWS/SNS	`NumberOfMessagesPublished`	La cantidad de mensajes publicados en los temas de Amazon SNS. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum
AWS/SNS	`NumberOfNotificationsDelivered`	La cantidad de mensajes entregados de forma correcta desde los temas de Amazon SNS a los puntos de enlace suscritos.

Espacio de nombres	Métrica	Descripción
		Para que un intento de entrega se lleve a cabo correctamente, la suscripción del punto de enlace debe aceptar el mensaje. En una suscripción, se acepta un mensaje si a.) carece de una política de filtro o b.) su política de filtro incluye atributos que coinciden con los asignados al mensaje. Si la suscripción rechaza el mensaje, el intento de entrega no se tiene en cuenta para esta métrica. Unidades: recuento Dimensiones válidas: aplicació n PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum

Espacio de nombres	Métrica	Descripción
AWS/SNS	`NumberOfNotificationsFailed`	La cantidad mensajes que Amazon SNS no pudo entregar. En el caso de Amazon SQS, correo electrónico, SMS o puntos de enlace push móviles, la métrica aumenta en 1 cuando Amazon SNS deja de intentar la entrega de mensajes. En los puntos de enlace HTTP o HTTPS, la métrica incluye todos los intentos de entrega erróneos, incluidos los intentos repetidos que siguen al intento inicial. Para todos los demás puntos de enlace, el recuento aumenta en 1 cuando no se logra entregar el mensaje (independientemente del número de intentos). Esta métrica no incluye los mensajes que han rechazado las políticas de filtro de suscripciones. Puede controlar el número de reintentos para los puntos de enlace HTTP. Para obtener más información, consulte Reintento de entrega de mensajes de Amazon SNS. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName

Espacio de nombres	Métrica	Descripción
		Estadísticas válidas: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut`	El número de mensajes que han rechazado las políticas de filtro de suscripciones. Una política de filtro rechaza un mensaje si los atributos de este no coinciden con los atributos de la política. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	El número de mensajes rechazados por las políticas de filtrado de suscripciones para el filtrado basado en atributos. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average

Espacio de nombres	Métrica	Descripción
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	El número de mensajes rechazados por las políticas de filtrado de suscripciones para el filtrado basado en cargas. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	La cantidad de mensajes que han rechazado las políticas de filtro de suscripciones debido a que los atributos de los mensajes no son válidos, por ejemplo, porque el atributo JSON tiene un formato incorrecto. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average

Espacio de nombres	Métrica	Descripción
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	El número de mensajes que han rechazado las políticas de filtro de suscripciones debido a que los mensajes no tienen atributos. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	La cantidad de mensajes que han rechazado las políticas de filtro de suscripciones debido a que el cuerpo del mensaje no es válido para el filtro, por ejemplo, cuerpo del mensaje JSON no válido. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average

Espacio de nombres	Métrica	Descripción
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Número de mensajes que se han movido a una cola de mensajes fallidos. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Número de mensajes que no se pudieron mover a una cola de mensajes fallidos. Unidades: recuento Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Sum, Average
AWS/SNS	`PublishSize`	El tamaño de los mensajes publicados. Unidades: Bytes Dimensiones válidas: aplicación PhoneNumber, plataforma y TopicName Estadísticas válidas: Minimum, Maximum, Average y Count

Espacio de nombres	Métrica	Descripción
AWS/SNS	`SMSMonthToDateSpentUSD`	Los cargos que se han acumulado desde el principio del mes actual por enviar mensajes SMS. Puedes configurar una alarma para esta métrica para saber si tus month-to-date cargos se acercan a la cuota mensual de gastos de SMS de tu cuenta. Cuando Amazon SNS determina que el envío de un mensaje SMS generaría un costo que supera esta cuota, deja de publicar mensajes SMS en cuestión de minutos. Para obtener información acerca de la configuración de su cuota de gasto mensual de SMS o acerca de cómo solicitar un aumento de la cuota de gasto en AWS, consulte Configuración de las preferencias de mensajería SMS en Amazon SNS. Unidades: USD Dimensiones válidas: None Estadísticas válidas: Sum

Espacio de nombres	Métrica	Descripción
AWS/SNS	`SMSSuccessRate`	El número de entregas de mensajes SMS realizadas correctamente. Unidades: recuento Dimensiones válidas: PhoneNumber Estadísticas válidas: Sum, Average, Data Samples

Dimensiones de las métricas de Amazon SNS

Amazon Simple Notification Service envía las siguientes dimensiones a CloudWatch.

Dimensión	Descripción
`Application`	Filtra los objetos de la aplicación, que representan una aplicación y un dispositivo registrados en uno de los servicios de notificaciones push compatibles, como APNs FCM.
`Application,Platform`	Filtra los objetos de aplicación y plataforma, donde los objetos de plataforma son para los servicios de notificaciones push compatibles, como APNs FCM.
`Country`	Filtra por el país o la región de destino de un mensaje SMS. El país o la región se representa mediante su código alpha-2 ISO 3166-1.
`PhoneNumber`	Filtra el número de teléfono cuando publica SMS de forma directa en un número de teléfono (sin tema).
`Platform`	Filtra los objetos de plataforma para los servicios de notificaciones push, como APNs FCM.

Dimensión	Descripción
`TopicName`	Filtra por nombres de temas de Amazon SNS.
`SMSType`	Filtra por el tipo de mensaje de un mensaje SMS. Puede ser promotional o transactional.

Métricas de uso de Amazon SNS

Amazon Simple Notification Service envía las siguientes métricas de uso a CloudWatch.

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
AWS/Uso	SNS	`ResourceCount`	`NumberOfMessagesPublishedPerAccount`	Recurso	• El número de mensajes publicados en tus temas de Amazon SNS en tu AWS cuenta. • Unidades: ninguna • Estadísticas válidas: Sum
AWS/Uso	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Recurso	• El número aproximado de temas en su AWS cuenta.

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
					• Unidades: ninguna • Estadísticas válidas: promedio, mínimo, máximo, suma
AWS/Uso	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Recurso	• El número aproximado de políticas de filtro en la cuenta de AWS . • Unidades: ninguna • Estadísticas válidas: promedio, mínimo, máximo, suma

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
AWS/Uso	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Recurso	• El número aproximado de suscripciones pendientes en tu AWS cuenta. • Unidades: ninguna • Estadísticas válidas: promedio, mínimo, máximo, suma

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
AWS/Uso	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber`	API	• El número de llamadas a la API de Amazon SNS seleccionada en su AWS cuenta. • No se permite el contenido en la sección final. Unidades: ninguna • Estadísticas válidas: Sum

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
			• DeleteTopic • GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut		

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
			• ListPlatformApplications • ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes		

Espacio de nombres	Servicio	Métrica	Recurso	Tipo	Descripción
			• SetSubscriptionAttributes • SetTopicAttributes • Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Validación de la conformidad de Amazon SNS

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Cumplimiento de seguridad y gobernanza: en estas guías se explican las consideraciones de arquitectura y se proporcionan pasos para implementar las características de seguridad y cumplimiento.
- Referencia de servicios válidos de HIPAA: muestra una lista con los servicios válidos de HIPAA. No todos Servicios de AWS cumplen con los requisitos de la HIPAA.
- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde la perspectiva del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en Amazon SNS

La resiliencia de Amazon SNS se garantiza mediante el aprovechamiento de la infraestructura AWS global, que gira en torno Regiones de AWS a las zonas de disponibilidad. Regiones de AWS ofrecen zonas de disponibilidad separadas y aisladas físicamente conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Esta arquitectura permite una conmutación por

error perfecta entre las zonas de disponibilidad sin interrupciones, lo que hace que las aplicaciones y bases de datos sean intrínsecamente más tolerantes a los errores y escalables en comparación con las infraestructuras de centros de datos tradicionales. Al utilizar las zonas de disponibilidad, los suscriptores de Amazon SNS se benefician de una mayor disponibilidad y fiabilidad, lo que garantiza la entrega de mensajes a pesar de las posibles interrupciones. Para obtener más información sobre las zonas de disponibilidad Regiones de AWS y las zonas de disponibilidad, consulte Infraestructura global.AWS

Además, las suscripciones a los temas de Amazon SNS se pueden configurar con reintentos de entrega y colas de mensajes fallidos, lo que permite tratar automáticamente los errores transitorios y garantizar que los mensajes lleguen de forma fiable a sus destinos previstos.

Amazon SNS también admite el filtrado de mensajes y los atributos de los mensajes, lo que le permite adaptar las estrategias de resiliencia a sus casos de uso específicos, mejorando así la robustez general de sus aplicaciones.

Seguridad de la infraestructura en Amazon SNS

Como servicio gestionado, Amazon SNS está protegido por los procedimientos de seguridad de la red AWS global que se encuentran en la documentación sobre prácticas recomendadas de seguridad, identidad y conformidad.

Utilice las acciones de la AWS API para acceder a Amazon SNS a través de la red. Los clientes deben ser compatibles con Transport Layer Security (TLS) 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE).

Debe firmar las solicitudes mediante un ID de clave de acceso y una clave de acceso secreta asociada a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Puede llamar a estas acciones de la API desde cualquier ubicación de red, pero Amazon SNS admite políticas de acceso basadas en recursos, que pueden incluir restricciones en función de la dirección IP de la fuente. También puede utilizar las políticas de Amazon SNS para controlar el acceso desde puntos de enlace de Amazon VPC específicos o específicos. VPCs Esto aísla eficazmente el acceso de red a un tema de Amazon SNS determinado únicamente desde la VPC específica de la red de AWS . Para obtener más información, consulte Restringir las publicaciones en un tema de Amazon SNS únicamente desde un punto de enlace de la VPC específico.

Prácticas recomendadas de seguridad para Amazon SNS

AWS proporciona muchas funciones de seguridad para Amazon SNS. Revise estas características de seguridad en el contexto de su propia política de seguridad.

> **Note**
>
> Las instrucciones para estas características de seguridad se aplican a los casos de uso comunes y a las implementaciones. Le sugerimos que revise estas prácticas recomendadas en el contexto de su caso de uso, arquitectura y modelo de amenaza concretos.

Prácticas recomendadas preventivas

A continuación, se indican las prácticas recomendadas de seguridad preventiva para Amazon SNS.

Temas

- Asegúrese de que los temas no sean accesibles de forma pública
- Implementación del acceso a los privilegios mínimos
- Utilice funciones de IAM para aplicaciones y AWS servicios que requieren acceso a Amazon SNS
- Implementación del cifrado en el servidor
- Aplicación del cifrado de los datos en tránsito
- Considere el uso de puntos de enlace de la VPC para obtener acceso a Amazon SNS
- Asegúrese de que las suscripciones no están configuradas para entregar en puntos de enlace HTTP sin procesar

Asegúrese de que los temas no sean accesibles de forma pública

A menos que exijas explícitamente a cualquier persona en Internet que pueda leer o escribir en tu tema de Amazon SNS, debes asegurarte de que tu tema no sea de acceso público (cualquiera en todo el mundo o cualquier usuario autenticado AWS).

- Evite la creación de políticas con `Principal` establecido en `""`.
- Evite usar un carácter comodín (`*`). En su lugar, designe a un usuario o usuarios específicos.

Implementación del acceso a los privilegios mínimos

Cuando concede permisos, decide quién los recibe, para qué temas son los permisos y las acciones específicas de la API que desea permitir en estos temas. La aplicación del principio de privilegios mínimos es importante para reducir los riesgos de seguridad. También ayuda a reducir el efecto negativo de los errores o intenciones maliciosas.

Siga el consejo de seguridad estándar de concesión del privilegio mínimo. Es decir, conceda solo los permisos necesarios para realizar una tarea específica. Puede implementar los privilegios mínimos mediante una combinación de políticas de seguridad relacionadas con el acceso de los usuarios.

Amazon SNS utiliza el modelo de editor-suscriptor, que requiere tres tipos de acceso a la cuenta de usuario:

- Administradores: acceso a la creación, modificación y eliminación de temas. Los administradores también controlan las políticas de temas.
- Publicadores: acceso al envío de mensajes a los temas.
- Suscriptores: acceso a la suscripción a los temas.

Para obtener más información, consulte las siguientes secciones:

- Identity and Access Management en Amazon SNS
- Permisos de la API de Amazon SNS: referencia de recursos y acciones

Utilice funciones de IAM para aplicaciones y AWS servicios que requieren acceso a Amazon SNS

Para que las aplicaciones o AWS los servicios, como Amazon EC2, puedan acceder a los temas de Amazon SNS, deben usar AWS credenciales válidas en sus solicitudes de AWS API. Como estas credenciales no se rotan automáticamente, no debes almacenar AWS las credenciales directamente en la aplicación o EC2 la instancia.

Debe utilizar un rol de IAM para administrar de manera temporal credenciales para las aplicaciones o los servicios que necesiten acceder a Amazon SNS. Cuando usas un rol, no necesitas distribuir credenciales de larga duración (como un nombre de usuario, contraseña y claves de acceso) a una EC2 instancia o AWS servicio, como AWS Lambda. En su lugar, el rol proporciona permisos temporales que las aplicaciones pueden usar cuando realizan llamadas a otros AWS recursos.

Para obtener más información, consulte Roles de IAM y Situaciones habituales con los roles: usuarios, aplicaciones y servicios en la Guía del usuario de IAM.

Implementación del cifrado en el servidor

Para mitigar los problemas de fuga de datos, utilice el cifrado en reposo para cifrar sus mensajes mediante una clave almacenada en una ubicación distinta de la ubicación en la que se almacenan los mensajes. Con el cifrado del lado del servidor (SSE), se proporciona cifrado de datos en reposo. Amazon SNS cifra sus datos a nivel de mensaje cuando los almacena y descifra los mensajes para usted cuando accede a ellos. El SSE usa claves administradas en. AWS Key Management Service Siempre que autentique su solicitud y tenga permisos de acceso, no existe diferencia alguna entre obtener acceso a los temas cifrados y sin cifrar.

Para obtener más información, consulte Protección de los datos de Amazon SNS con cifrado del servidor y Administración de las claves de cifrado y los costos de Amazon SNS.

Aplicación del cifrado de los datos en tránsito

Es posible, pero no se recomienda, publicar mensajes que no están cifrados durante el tránsito mediante HTTP. Sin embargo, cuando un tema se cifra en reposo AWS KMS, es necesario utilizar HTTPS para publicar los mensajes a fin de garantizar el cifrado tanto en reposo como en tránsito. Aunque el tema no rechaza automáticamente los mensajes HTTP, es necesario usar HTTPS para mantener los estándares de seguridad.

AWS recomienda utilizar HTTPS en lugar de HTTP. Cuando utiliza HTTPS, los mensajes se cifran de manera automática durante el tránsito, incluso si el tema de SNS no está cifrado. Sin HTTPS, un atacante basado en la red puede espiar el tráfico de la red o manipularlo mediante un ataque como. man-in-the-middle

Para imponer solo conexiones cifradas a través de HTTPS, agregue la condición `aws:SecureTransport` en la política de IAM adjunta a temas de SNS no cifrados. De esta manera, los publicadores utilizan HTTPS en lugar de HTTP. Puede utilizar la política de ejemplo siguiente como guía:

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

Considere el uso de puntos de enlace de la VPC para obtener acceso a Amazon SNS

Si tiene temas con los que debe poder interactuar pero que no deben estar expuestos a Internet, utilice los puntos de enlace de la VPC para poner en la cola el acceso solo a los hosts dentro de una VPC concreta. Puede utilizar las políticas de temas para controlar el acceso a los temas desde puntos de enlace de Amazon VPC específicos o desde puntos específicos. VPCs

Los puntos de enlace de la VPC de Amazon SNS brindan dos maneras de controlar el acceso a los mensajes:

- Puede controlar qué solicitudes, usuarios o grupos obtienen acceso a través de un punto de conexión de la VPC específico.
- Puedes controlar qué puntos finales VPCs o puntos finales de VPC tienen acceso a tu tema mediante una política de tema.

Para obtener más información, consulte Creación del punto de enlace y Creación de una política de punto de enlace de la VPC para Amazon SNS.

Asegúrese de que las suscripciones no están configuradas para entregar en puntos de enlace HTTP sin procesar

Evite configurar suscripciones para entregarlas a puntos de enlace HTTP sin procesar. Siempre tiene suscripciones que se entregan a un nombre de dominio de punto de enlace. Por ejemplo, una suscripción configurada para entregar a un punto de enlace `http://1.2.3.4/my-path` debe cambiarse a `http://my.domain.name/my-path`.

Solución de problemas de Amazon SNS mediante AWS X-Ray

AWS X-Ray recopila datos sobre las solicitudes que atiende su aplicación y le permite ver y filtrar los datos para identificar posibles problemas y oportunidades de optimización. Para cualquier solicitud rastreada hasta su aplicación, puede ver información detallada sobre la solicitud, la respuesta y las llamadas que la aplicación realiza a los AWS recursos intermedios, los microservicios, las bases de datos y la web HTTP. APIs

Puede utilizar X-Ray con Amazon SNS para rastrear y analizar los mensajes que pasan por su aplicación. Puede utilizarla AWS Management Console para ver el mapa de conexiones entre Amazon SNS y otros servicios que utiliza su aplicación. También puede utilizar la consola para ver métricas como la latencia media y las tasas de errores. Para obtener más información, consulte Amazon SNS y AWS X-Ray en la Guía para desarrolladores de AWS X-Ray .

Rastreo activo en Amazon SNS

Úselo AWS X-Ray para rastrear y analizar las solicitudes de los usuarios a medida que pasan por sus temas de Amazon SNS hasta las suscripciones de puntos de conexión Amazon Data Firehose, Amazon AWS LambdaSQS y HTTP/S.

Con X-Ray, obtiene una end-to-end vista de cada solicitud, lo que le permite:

- Identifique qué es lo que llama a su tema de Amazon SNS y qué servicios son posteriores a sus suscripciones.
- Analice las latencias, como las siguientes:
  - Tiempo dedicado al tema Amazon SNS antes de procesarlo.
  - Tiempos de entrega para cada punto de conexión suscrito.

> **Important**
>
> Es posible que los temas de Amazon SNS con numerosas suscripciones alcancen el límite de tamaño y no se rastreen por completo. Para obtener información sobre los límites de tamaño de los documentos de rastreo, consulte las cuotas de servicio de rayos X en la Referencia AWS general.

Si llama a una API de Amazon SNS desde un servicio que ya se está rastreando, Amazon SNS transmite el rastreo, aunque el rastreo de X-Ray no esté habilitado en la API.

Amazon SNS solo admite rastreo de X-Ray para temas estándar y FIFO. Puede habilitar X-Ray para un tema de Amazon SNS mediante la consola de Amazon SNS, la API `SetTopicAttributes` de Amazon SNS, la referencia de la CLI de Amazon Simple Notification Service o AWS CloudFormation.

Para obtener más información acerca del uso de Amazon SNS con X-Ray, consulte Amazon SNS and AWS X-Ray (Amazon SNS y AWS X-Ray) en la Guía para desarrolladores de AWS X-Ray .

Permisos de rastreo activo

Al utilizar la consola de Amazon SNS, Amazon SNS intenta crear los permisos necesarios para que el tema de Amazon SNS llame a X-Ray. El intento puede rechazarse si no tiene los permisos suficientes para usar la consola de Amazon SNS. Para obtener más información, consulte Identity and Access Management en Amazon SNS y Ejemplos de casos de control de acceso con Amazon SNS.

Cuando utilice la CLI, debe configurar los permisos manualmente. Estos permisos se configuran mediante políticas de recursos. Para obtener más información acerca del uso de los permisos necesarios en X-Ray, consulte Amazon SNS and AWS X-Ray (Amazon SNS y AWS X-Ray).

Habilitar el rastreo activo en un tema de Amazon SNS mediante la consola AWS

Cuando se habilita el rastreo activo en un tema de Amazon SNS, este lee el identificador de rastreo, envía los datos al cliente en función de ese identificador y lo propaga a los servicios posteriores.

1. Inicie sesión en la consola de Amazon SNS.
2. Elija un tema o cree uno nuevo. Para obtener más información acerca de la creación de temas, consulte Creación de un tema de Amazon SNS.
3. En la página Crear tema, en la sección Detalles, elija un tipo de tema: FIFO o Estándar.
    a. Ingrese un nombre para el nuevo tema.
    b. (Opcional) Ingrese un nombre para mostrar para el tema.
4. Expanda Active tracing (Rastreo activo) y seleccione Use active tracing (Usar rastreo activo).

Una vez que hayas activado X-Ray para tu tema de Amazon SNS, puedes usar el mapa del servicio de X-Ray para ver las end-to-end trazas y los mapas de servicio del tema.

Habilitar el rastreo activo en un tema de Amazon SNS mediante el SDK AWS

El siguiente ejemplo de código muestra cómo habilitar el rastreo activo en un tema de Amazon SNS mediante AWS el 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());
        }
    }
```

Habilitar el rastreo activo en un tema de Amazon SNS mediante la CLI AWS

El siguiente ejemplo de código muestra cómo habilitar el rastreo activo en un tema de Amazon SNS mediante la AWS CLI.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Habilitar el rastreo activo en un tema de Amazon SNS mediante AWS CloudFormation

La siguiente AWS CloudFormation pila muestra cómo habilitar el rastreo activo en un tema de Amazon SNS.

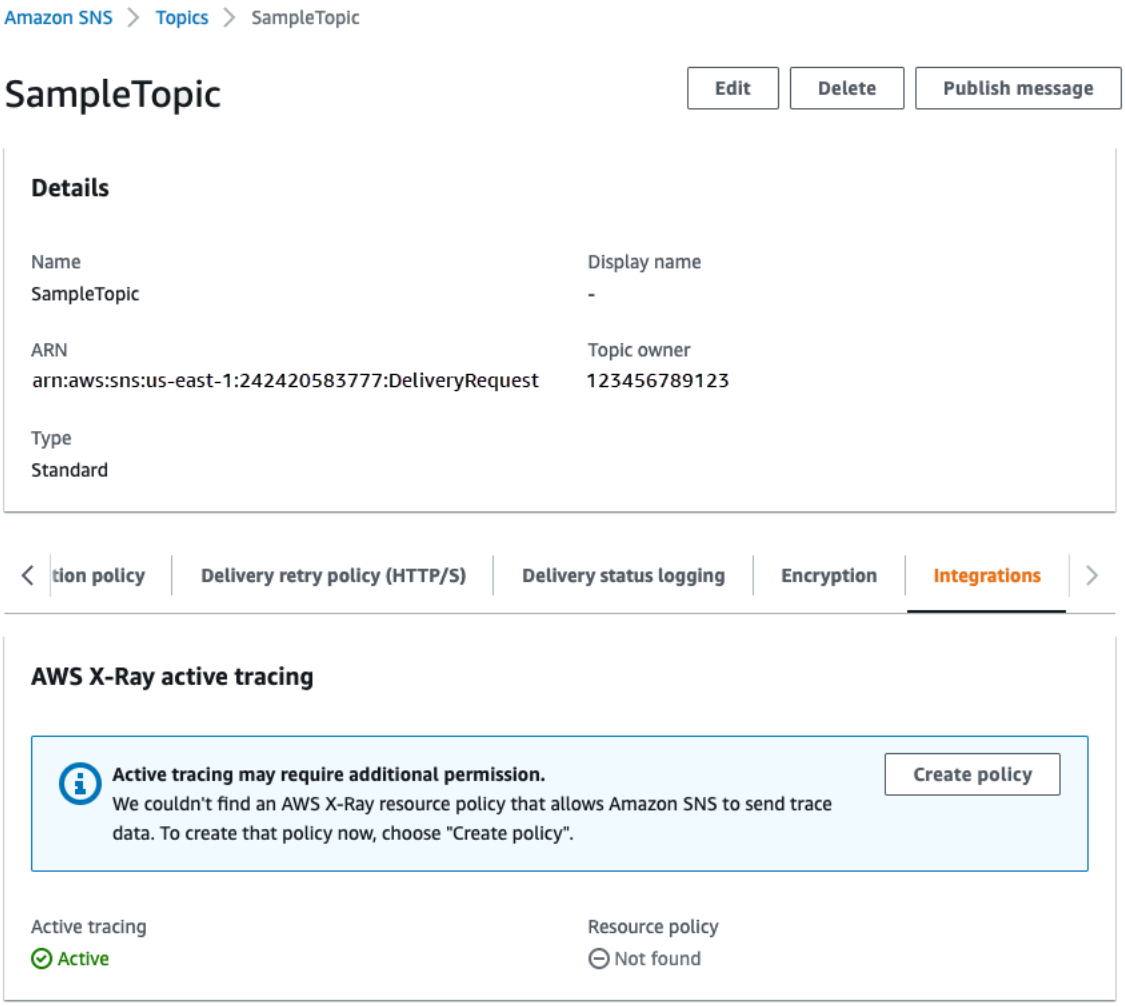
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Verificación de que el rastreo activo está habilitado para su tema

Puede utilizar la consola de Amazon SNS para comprobar si el rastreo activo está habilitado para su tema o si no se ha podido añadir la política de recursos.

1. Inicie sesión en la consola de Amazon SNS.
2. En el panel de navegación izquierdo, elija Topics (Temas).
3. Elija un tema en la página Topics (Temas).
4. Elija la pestaña Integrations (Integraciones).

   Cuando el rastreo activo está habilitado, aparece un icono Active (Activo) de color verde.
5. Si ha habilitado el rastreo activo y no ve que se haya añadido la política de recursos, elija Create policy (Crear política) para añadir los permisos adicionales necesarios.



Prueba del rastreo activo

1. Inicie sesión en la consola de Amazon SNS.
2. Crear un tema de Amazon SNS Para obtener más detalles sobre cómo hacerlo, consulte Para crear un tema mediante el AWS Management Console.
3. Expanda Active tracing (Rastreo activo) y seleccione Use active tracing (Usar rastreo activo).
4. Publique un mensaje en el tema de Amazon SNS. Para obtener más detalles sobre cómo hacerlo, consulte Para publicar mensajes en temas de Amazon SNS mediante la AWS Management Console, siga estos pasos:.
5. Utilice el mapa del servicio de X-Ray para ver las end-to-end trazas y los mapas de servicio del tema.


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

Historial de documentos de Amazon SNS

En la siguiente tabla, se describen los cambios recientes en la Guía para desarrolladores de Amazon Simple Notification Service.

A veces, las funciones del servicio se implementan de forma gradual en las AWS regiones en las que el servicio está disponible. Actualizamos esta documentación solo para la primera versión. No proporcionamos información sobre la disponibilidad en regiones ni anunciamos lanzamientos posteriores en regiones. Para obtener información sobre la disponibilidad regional de las funciones del servicio y suscribirse a las notificaciones sobre actualizaciones, consulte ¿Qué hay de nuevo? AWS .

Cambio	Descripción	Fecha
Se agregó soporte para terminales de doble pila (IPv4 y IPv6)	Amazon SNS ahora admite IPv6 solicitudes de API, lo que permite a los clientes conectarse a puntos de enlace públicos mediante IPv4 IPv6, o doble pila.	3 de abril de 2025
Amazon SNS agregó el `FifoThroughputScope` atributo para los temas de FIFO de Amazon SNS.	Amazon SNS admite el FifoThroughputScope atributo, que especifica la cuota de rendimiento y el comportam iento de deduplicación que se deben aplicar al tema FIFO. Los valores válidos son `Topic` o `MessageGroup` .	21 de enero de 2025
Actualizaciones de las políticas administradas `AmazonSNSFullAccess` y `AmazonSNSReadOnlyAccess`	Amazon SNS agregó nuevos permisos a AmazonSNS FullAccess y AmazonSNS ReadOnlyAccess políticas gestionadas, que permiten un acceso adicional a Amazon	24 de septiembre de 2024

	SNS a través del. AWS Management Console	
Integración de Amazon SNS con la entrega AWS End User Messaging SMS de mensajes SMS	Amazon SNS admite nuevas funciones, como la administración de recursos de SMS, la mensajería bidireccional, los permisos de recursos detallados, las reglas de bloqueo de países y la facturación centralizada de todos los mensajes AWS SMS sin realizar ningún cambio en las configuraciones o en la red global de AWS SMS que utiliza Amazon SNS.	24 de septiembre de 2024
Compatibilidad de la región Oeste de Canadá (Calgary) para los temas FIFO	Amazon SNS admite el tema FIFO en la región Oeste de Canadá (Calgary).	28 de marzo de 2024
Compatibilidad con SMS de Amazon SNS en cinco nuevas regiones	Amazon SNS ha añadido compatibilidad con SMS a las siguientes regiones: Asia-Pacífico (Hyderabad), Asia-Pacífico (Melbourne), Oriente Medio (UAE), Europa (Zúrich) y Europa (España).	8 de febrero de 2024
Compatibilidad con HTTP v1 de Firebase Cloud Messaging (FCM)	Amazon SNS admite las credenciales de FCM v1.	18 de enero de 2024
SMS de Amazon SNS admitido en Asia-Pacífico (Yakarta)	Amazon SNS admite mensajes SMS en Asia-Pacífico (Yakarta).	14 de diciembre de 2023

AWS CloudFormation soporte `DeliveryStatusLogging` para configurar temas de Amazon SNS	AWS CloudFormation hay soporte disponible para configurar los temas de Amazon SNS `DeliveryStatusLogging` al crear o actualizar.	7 de diciembre de 2023
Se han añadido operadores de filtrado de mensajes nuevos	Ahora puede utilizar los operadores de coincidencia de sufijos, omisión de mayúsculas y minúsculas y OR al filtrar los mensajes de Amazon SNS.	16 de noviembre de 2023
Se ha añadido compatibilidad para el archivo y la reproducción de mensajes	Los propietarios de los temas pueden archivar los mensajes relacionados con un tema durante un máximo de 365 días. Los suscriptores de un tema pueden reproducir los mensajes archivados devueltos a un punto de conexión suscrito para recuperar los mensajes que se hayan producido debido a un error en una aplicación posterior o para replicar el estado de una aplicación existente.	26 de octubre de 2023

Se ha añadido compatibilidad para suscribir una cola estándar a un tema FIFO	Puede suscribir una cola FIFO de Amazon SQS o una cola estándar a un tema FIFO de Amazon SNS. Solo las colas FIFO de Amazon SQS garantizan que los mensajes se reciban en orden y sin duplicados.	14 de septiembre de 2023
Se ha añadido compatibilidad de SMS para Israel (Tel Aviv)	Los mensajes SMS de Amazon SNS ahora se admiten en la región de Israel (Tel Aviv).	28 de agosto de 2023
Se ha añadido compatibilidad con el rastreo activo de X-Ray a los temas FIFO	Anteriormente solo era compatible con los temas estándar de Amazon SNS, AWS X-Ray ahora rastrea y analiza las solicitudes de los usuarios a medida que recorren los temas de FIFO hasta sus suscripciones de puntos de conexión a Amazon Data Firehose, Amazon AWS Lambda SQS y HTTP/S.	31 de mayo de 2023
Compatibilidad mejorada con el encabezado Content-Type	Puede establecer el encabezado Content-Type en la política de solicitud para especificar el tipo de medio de la notificación.	22 de marzo de 2023

Se ha añadido compatibilidad con el rastreo activo	AWS X-Ray rastrea y analiza las solicitudes de los usuarios a medida que recorren los temas estándar de Amazon SNS hasta sus suscripci ones de puntos de conexión Amazon Data Firehose, Amazon AWS Lambda SQS y HTTP/S.	8 de febrero de 2023
Registro de ID de remitente de Singapur	Se agregaron instrucciones para registrar al remitente IDs en Singapur.	10 de enero de 2023
Filtrado de mensajes basado en cargas útiles	El filtrado basado en cargas útiles le permite filtrar los mensajes en función de la carga útil del mensaje y evitar los costos asociados al procesamiento de datos no deseados.	22 de noviembre de 2022
SHA256 Se agregó un algoritmo hash para la firma de mensajes de Amazon SNS	Support se ha añadido para el algoritmo SHA256 hash al utilizar la firma de mensajes de Amazon SNS.	15 de septiembre de 2022
Se han añadido regiones adicionales a los mensajes SMS	Amazon SNS admite la mensajería SMS en las siguientes regiones: África (Ciudad del Cabo), Asia Pacífico (Osaka), Europa (Milán) y AWS GovCloud (EE. UU. Este).	9 de septiembre de 2022

Se ha añadido compatibilidad con la función de protección de datos de mensajes	La protección de datos de los mensajes protege los datos que se publican en sus temas de Amazon SNS mediante el uso de políticas de protección de datos para auditar y bloquear la información confidencial que se transfiere entre aplicaciones o AWS servicios.	8 de septiembre de 2022
Nuevo proceso de registro para números gratuitos	Se ha añadido compatibilidad para el envío de mensajes de Amazon SNS mediante números de teléfono gratuitos a destinatarios de Estados Unidos.	1 de agosto de 2022
Se ha añadido compatibilidad con el control de acceso basado en atributos (ABAC)	Se ha añadido compatibilidad con el control de acceso basado en atributos (ABAC) para acciones de API que incluyen `Publish` y `PublishBatch` . ABAC es una estrategia de autorización que define los permisos de acceso en función de las etiquetas que se pueden adjuntar a los recursos de IAM, como los usuarios y roles de IAM, y a los AWS recursos, como los temas de Amazon SNS, para simplificar la administración de permisos.	10 de enero de 2022

Se ha añadido compatibilidad con la autenticación basada en tokens de Apple para notificaciones push	Puede autorizar a Amazon SNS para que envíe notificaciones push a su aplicación de iOS o macOS proporcionando información que le identifique como desarrollador de esa aplicación.	28 de octubre de 2021
Los nuevos remitentes de mensajes SMS se colocan en el entorno de pruebas de SMS	Con el entorno de pruebas de SMS, evitará fraudes y abusos, y protegerá su reputación como remitente. Mientras su AWS cuenta esté en el entorno limitado de SMS, solo podrá enviar mensajes SMS a números de teléfono de destino verificados.	1 de junio de 2021
Los nuevos remitentes de mensajes SMS se colocan en el entorno de pruebas de SMS	Con el entorno de pruebas de SMS, evitará fraudes y abusos, y protegerá su reputación como remitente. Mientras tu AWS cuenta esté en el entorno limitado de SMS, solo podrás enviar mensajes SMS a números de teléfono de destino verificados.	1 de junio de 2021
Nuevos atributos para enviar mensajes SMS a destinatarios de la India	Dos nuevos atributos, ID de identidad e ID de plantilla, son necesarios para enviar mensajes SMS a destinatarios de la India.	22 de abril de 2021

Actualizaciones de operadores de filtrado de mensajes	Se dispone de nuevo operador, `cidr`, para la búsqueda de coincidencias de subredes y direcciones IP de origen del mensaje. Ahora también puede comprobar la ausencia de una clave de atributo y usar un prefijo con el operador `anything-but` para buscar coincidencias de cadenas de atributos.	7 de abril de 2021
Fin del soporte de códigos largos P2P a destinos de EE. UU.	A partir del 1 de junio de 2021, los proveedores de telecomunicaciones estadounidenses ya no admiten el uso de códigos largos person-to-person (P2P) para las comunicaciones application-to-person (A2P) con destinos estadounidenses. En su lugar, puede usar códigos cortos, números gratuitos o un nuevo tipo de número de fuente llamado 10DLC.	16 de febrero de 2021
Support para métricas de Amazon CloudWatch de 1 minuto	La CloudWatch métrica de 1 minuto de Amazon SNS ya está disponible en AWS todas las regiones.	28 de enero de 2021

Se ha añadido compatibilidad con los puntos de conexión de Amazon Data Firehose	Puede suscribir los flujos de entrega de Firehose a los temas de SNS. Esto le permite enviar notificaciones a puntos finales de archivado y análisis, como depósitos de Amazon Simple Storage Service (Amazon S3), tablas de Amazon Redshift, Amazon Service (Service) y más. OpenSearch OpenSearch	12 de enero de 2021
Números de origen disponibles	Puede utilizar números de origen al enviar mensajes de texto (SMS).	23 de octubre de 2020
Se ha añadido compatibilidad con temas FIFO de Amazon SNS	Para integrar aplicaciones distribuidas que requieren coherencia de datos casi en tiempo real, puede utilizar temas de primero en entrar, primero en salir (FIFO) de Amazon SNS con colas FIFO de Amazon SQS.	22 de octubre de 2020
La biblioteca de clientes ampliada de Amazon SNS para Java ya está disponible	Puede utilizar esta bibliotec a para publicar mensajes de Amazon SNS grandes.	25 de agosto de 2020
SSE está disponible en las regiones de China	El cifrado del servidor (SSE) de Amazon SNS está disponible en las regiones de China.	20 de enero de 2020

Support para capturar DLQs mensajes que no se pueden entregar	Para capturar los mensajes que no se pueden entregar, puede utilizar una cola de mensajes fallidos (DLQ) de Amazon SQS con una suscripción a Amazon SNS.	14 de noviembre de 2019
Support para especificar valores de APNs cabecera personalizados	Puede especificar un valor de APNs encabezado personalizado.	18 de octubre de 2019
Support para el campo de encabezado `apns-push-type` " para APNs	Puedes usar el campo de `apns-push-type` encabezado para las notificaciones móviles que se envíen APNs.	10 de septiembre de 2019
Support para la solución de problemas por temas mediante AWS X-Ray	Puede utilizar X-Ray para solucionar los problemas de los mensajes que pasan por los temas SNS.	24 de julio de 2019
Compatibilidad con la búsqueda de coincidencias de claves de atributos mediante el operador “`exists`”	Para verificar si un mensaje entrante tiene un atributo cuya clave figura en la política de filtrado, puede utilizar el operador `exists`.	5 de julio de 2019
Se ha añadido compatibilidad con la coincidencia anything-but de varios valores numéricos	Además de varias cadenas, Amazon SNS permite la coincidencia “anything-but” de varios valores numéricos.	5 de julio de 2019
Las notas de la versión de Amazon SNS están disponibles como una fuente RSS	Siga el título de esta página (Historial de documentos) y elija RSS.	22 de junio de 2019

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.