aws

Guide du développeur

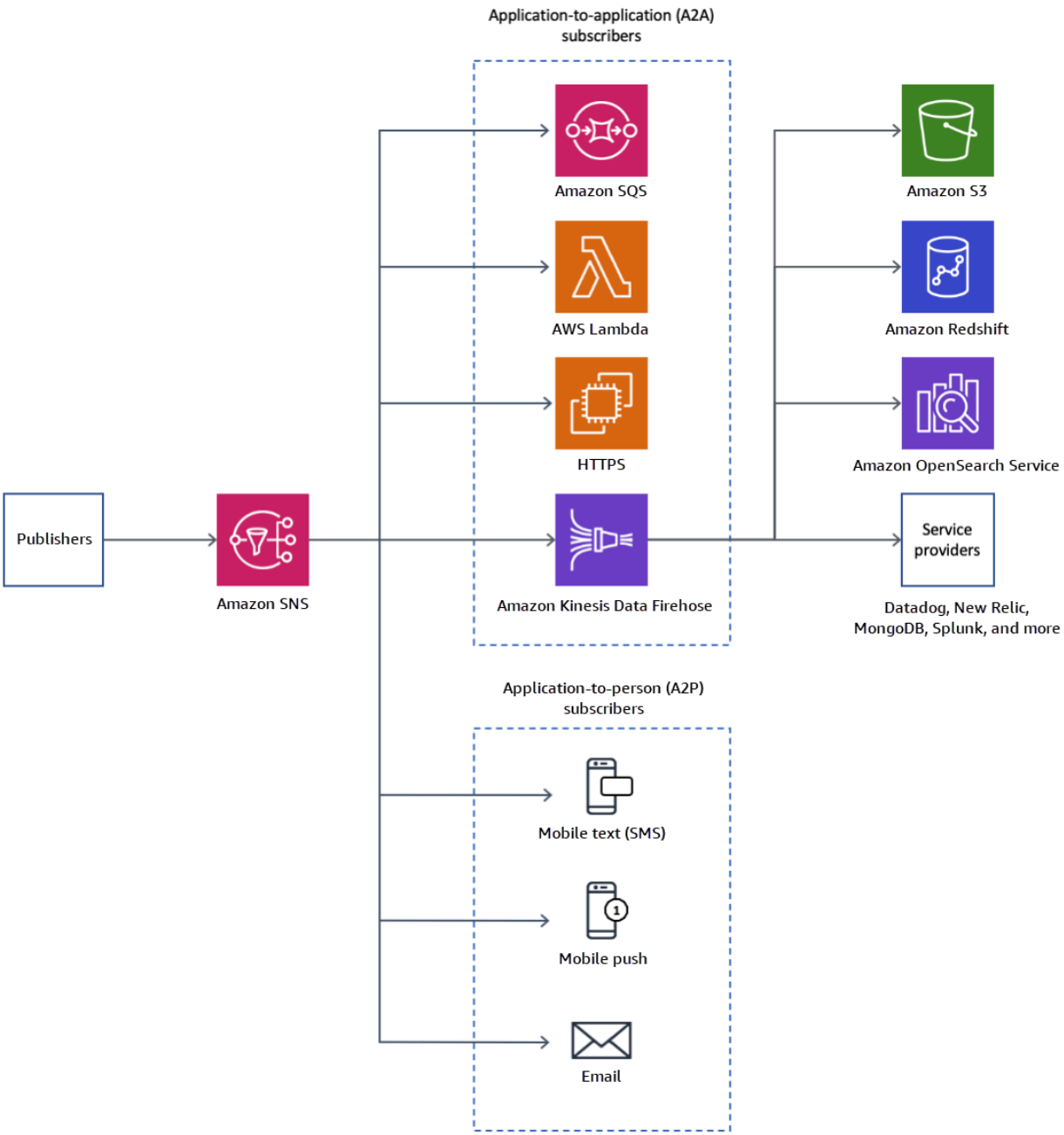
Amazon Simple Notification Service

Amazon Simple Notification Service: Guide du développeur

Table of Contents

Qu'est-ce qu'Amazon SNS ?

Amazon Simple Notification Service (Amazon SNS) est un service géré qui fournit la diffusion des messages des éditeurs aux abonnés (également appelé producteurs and consommateurs). Les éditeurs communiquent de façon asynchrone avec les abonnés en envoyant un message à une rubrique, qui est un point d'accès logique et un canal de communication. Les clients peuvent s'abonner à la rubrique SNS et recevoir des messages publiés à l'aide d'un type de point de terminaison compatible, tel qu'Amazon Data Firehose, Amazon SQS AWS Lambda, HTTP, e-mail, notifications push mobiles et messages texte (SMS) mobiles.



Rubriques

- Fonctionnalités et capacités
- Services connexes
- Accès à Amazon SNS

- Tarification pour Amazon SNS
- Scénarios Amazon SNS courants
- Utilisation d'Amazon SNS avec un SDK AWS

Fonctionnalités et capacités

Amazon SNS fournit les fonctions et capacités suivantes :

- Un pplication-to-application message

  Une pplication-to-application messagerie prend en charge les abonnés tels que les flux de diffusion Amazon Data Firehose, les fonctions Lambda, les files d'attente Amazon SQS, les points de terminaison HTTP/S et les pipelines Event Fork. AWS Pour plus d’informations, consultez Messagerie d'application à application (A2A).
- pplication-to-person Notifications A.

  pplication-to-person Les notifications fournissent des notifications aux utilisateurs aux abonnés, telles que les applications mobiles, les numéros de téléphone portable et les adresses e-mail. Pour plus d’informations, consultez Messagerie d'application à personne (A2P).
- Rubriques standard et FIFO

  Utilisez une rubrique FIFO pour garantir un ordre strict des messages, définir des groupes de messages et empêcher la duplication des messages. Vous pouvez utiliser à la fois des files d'attente FIFO et des files d'attente standard pour vous abonner à une rubrique FIFO. Pour plus d’informations, consultez Classement et déduplication des messages (rubriques FIFO).

  Utilisez une rubrique standard lorsque l'ordre de remise des messages et la duplication possible des messages ne sont pas critiques. Tous les protocoles de livraison pris en charge peuvent s'abonner à une rubrique standard.
- Durabilité des messages

  Amazon SNS utilise un certain nombre de stratégies qui fonctionnent ensemble pour assurer la durabilité des messages :
  - Les messages publiés sont stockés sur plusieurs serveurs et centres de données séparés géographiquement.
  - Si aucun point de terminaison abonné n'est disponible, Amazon SNS exécute une stratégie de nouvelle tentative de livraison.

  - Pour conserver tous les messages qui ne sont pas remis avant la fin de la stratégie de nouvelle tentative de livraison, vous pouvez créer une file d'attente de lettres mortes.
- Archivage, relecture et analyse des messages

  Vous pouvez archiver les messages avec Amazon SNS de différentes manières, notamment en abonnant les flux de diffusion Firehose aux rubriques SNS, ce qui vous permet d'envoyer des notifications à des points de terminaison d'analyse tels que les buckets Amazon Simple Storage Service (Amazon S3), les tables Amazon Redshift, etc. Par ailleurs, les rubriques FIFO d'Amazon SNS prennent en charge l'archivage-relecture des messages sous forme d'archive de messages sur place et sans code, ce qui permet aux propriétaires de rubrique de stocker (ou archiver) les messages dans leur rubrique. Les abonnés à une rubrique peuvent ensuite récupérer (ou relire) les messages archivés sur un point de terminaison abonné. Pour en savoir plus, consultez Archivage-relecture des messages pour les rubriques FIFO.
- Attributs de message

  Les attributs de message vous permettent de fournir des métadonnées arbitraires relatives à un message. the section called “Attributs de message”.
- Filtrage de messages

  Par défaut, un abonné reçoit chaque message publié dans la rubrique. Pour recevoir uniquement un sous-ensemble des messages, un abonné doit attribuer une politique de filtre à l'abonnement à la rubrique. Un abonné peut également définir l'étendue de la politique de filtre pour activer le filtrage basé sur la charge utile ou sur les attributs. La valeur par défaut pour l'étendue de la politique de filtre est `MessageAttributes`. Lorsque les attributs de message entrant correspondent aux attributs de stratégie de filtrage, le message est remis au point de terminaison abonné. Sinon, le message est filtré. Lorsque l'étendue de la politique de filtre est `MessageBody`, les attributs de la politique de filtre sont mis en correspondance avec la charge utile. Pour plus d’informations, consultez Filtrage de messages.
- Sécurité des messages

  Le chiffrement côté serveur protège le contenu des messages stockés dans les rubriques Amazon SNS à l'aide des clés de chiffrement fournies par. AWS KMS Pour plus d’informations, consultez the section called “Chiffrement au repos”.

  Vous pouvez également établir une connexion privée entre Amazon SNS et votre Virtual Private Cloud (VPC). Pour plus d'informations, consultez the section called “Confidentialité du trafic inter-réseau”.

Services connexes

Vous pouvez utiliser les services suivants avec Amazon SNS :

- Amazon SQS offre une file d'attente hébergée sécurisée, durable et disponible qui vous permet d'intégrer et de découpler les systèmes et les composants de logiciels distribués. Amazon SQS est lié à Amazon SNS de la manière suivante :
  - Amazon SNS fournit des files d'attente de lettres mortes optimisées par Amazon SQS pour les messages non livrables.
  - Vous pouvez abonner une file d'attente Amazon SQS à une rubrique Amazon SNS.
  - Vous pouvez abonner une file d'attente FIFO Amazon SQS ou une file d’attente standard à une rubrique FIFO Amazon SNS. Seules les files d'attente FIFO Amazon SQS garantissent la réception des messages dans l'ordre et sans doublons.
- AWS Lambda vous permet de créer des applications très réactives par rapport aux nouvelles informations. Exécutez votre code dans les fonctions Lambda sur une infrastructure de calcul hautement disponible. Pour plus d'informations, consultez le Guide du développeur AWS Lambda. Vous pouvez abonner une fonction Lambda à une rubrique SNS.
- AWS Identity and Access Management (IAM) vous aide à contrôler en toute sécurité l'accès aux AWS ressources pour vos utilisateurs. Utilisez IAM pour contrôler qui peut utiliser vos rubriques Amazon SNS (authentification), quelles rubriques peuvent être utilisées et comment elles peuvent être utilisées (autorisation). Pour plus d’informations, consultez Utilisation de politiques basées sur l'identité avec Amazon SNS.
- AWS CloudFormationvous permet de modéliser et de configurer vos AWS ressources. Créez un modèle qui décrit les AWS ressources que vous souhaitez, y compris les rubriques et les abonnements Amazon SNS. AWS CloudFormation s'occupe du provisionnement et de la configuration de ces ressources pour vous. Pour plus d'informations, veuillez consulter le Guide de l'utilisateur AWS CloudFormation.

Accès à Amazon SNS

Vous pouvez configurer et gérer les rubriques et les abonnements SNS à l'aide de la console Amazon SNS, des outils de ligne de commande ou des kits SDK. AWS

- La console Amazon SNS fournit une interface utilisateur pratique pour créer des rubriques et des abonnements, envoyer et recevoir des messages et surveiller les évènements et les journaux.

- Le AWS Command Line Interface (AWS CLI) vous donne un accès direct à l'API Amazon SNS pour des cas d'utilisation avancés en matière de configuration et d'automatisation. Pour plus d'informations, consultez Utilisation d'Amazon SNS avec la AWS CLI.
- AWS fournit des SDK dans différentes langues. Pour plus d'informations, consultez Kits SDK et boîtes à outils.

Tarification pour Amazon SNS

Amazon SNS n'a pas de coûts initiaux. Vous payez en fonction du nombre de messages que vous publiez, du nombre de notifications que vous envoyez et des appels d'API supplémentaires pour la gestion des rubriques et des abonnements. Le prix de livraison varie selon le type de point de terminaison. Vous pouvez commencer gratuitement avec l'offre gratuite Amazon SNS.

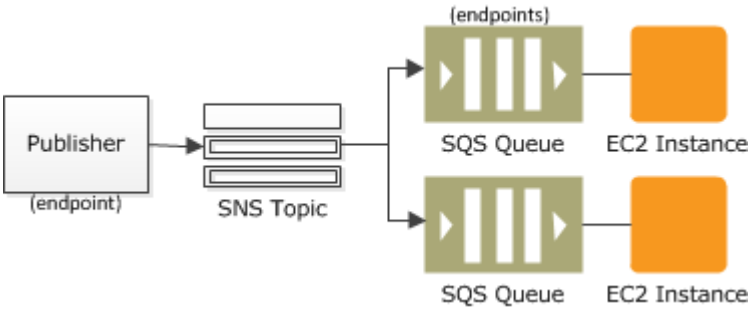
Pour plus d'informations, consultez Tarification Amazon SNS.

Scénarios Amazon SNS courants

Intégration d’applications

Le scénario Fanout se produit lorsqu'un message publié sur une rubrique SNS est répliqué et transmis à plusieurs points de terminaison, tels que les flux de diffusion Firehose, les files d'attente Amazon SQS, les points de terminaison HTTP (S) et les fonctions Lambda. Cela permet un traitement asynchrone parallèle.

Par exemple, vous pouvez développer une application qui envoie un message vers une rubrique SNS lorsqu'une commande est passée pour un produit. Ensuite, les files d'attente SQS qui sont abonnées à cette rubrique SNS reçoivent des notifications identiques pour la nouvelle commande. Une instance de serveur Amazon Elastic Compute Cloud (Amazon EC2) attachée à l'une des files d'attente SQS peut gérer le traitement ou l'exécution de la commande. Vous pouvez également joindre une autre instance de serveur Amazon EC2 à un entrepôt de données pour l'analyse de toutes les commandes reçues.



Vous pouvez aussi utiliser la distribution ramifiée pour répliquer les données envoyées à votre environnement de production avec votre environnement de test. En approfondissant l'exemple précédent, vous pouvez abonner une autre file d'attente SQS à la même rubrique SNS pour les nouvelles commandes entrantes. Ensuite, en attachant cette nouvelle file d'attente SQS à votre environnement de test, vous pouvez continuer à améliorer et tester votre application à l'aide des données reçues de votre environnement de production.

> **Important**
>
> Veuillez à tenir compte de la confidentialité et de la sécurité des données avant d'envoyer des données de production à votre environnement de test.

Pour plus d'informations, veuillez consulter les ressources suivantes :

- Streams de diffusion de Fanout to Firehose
- Diffusion dans les fonctions Lambda
- Distribution ramifiée vers des files d'attente Amazon SQS
- Diffusion en éventail vers les points de terminaison HTTP(S)
- Informatique axée sur les événements avec Amazon SNS AWS et les services de calcul, de stockage, de base de données et de mise en réseau

Alertes d'application

Les alertes d'application et du système sont des notifications qui sont déclenchées par des seuils prédéfinis. Amazon SNS peut envoyer ces notifications à des utilisateurs spécifiés par SMS et e-mail. Par exemple, vous pouvez recevoir une notification immédiate lorsqu'un événement se produit, tel qu'une modification spécifique apportée à votre groupe Amazon EC2 Auto Scaling, un nouveau fichier chargé dans un compartiment Amazon S3 ou un seuil métrique dépassé dans Amazon.

CloudWatch Pour plus d'informations, consultez la section Configuration des notifications Amazon SNS dans le guide de CloudWatch l'utilisateur Amazon.

Notifications utilisateur

Amazon SNS peut envoyer des messages e-mail push et des messages texte (SMS) à des individus ou à des groupes. Par exemple, vous pouvez envoyer des confirmations de commande d'e-commerce sous forme de notifications utilisateur. Pour plus d'informations, sur l'utilisation d'Amazon SNS pour envoyer des messages SMS, consultez Messagerie texte mobile (SMS).

Notifications push mobile

Les notifications push mobile vous permettent d'envoyer des messages directement aux applications mobiles. Par exemple, vous pouvez utiliser Amazon SNS pour envoyer des notifications de mise à jour à une application. Le message de notification peut inclure un lien pour télécharger et installer la mise à jour. Pour plus d'informations, sur l'utilisation d'Amazon SNS pour envoyer des messages de notification push, consultez Notifications push mobile.

Utilisation d'Amazon SNS avec un SDK AWS

AWS des kits de développement logiciel (SDK) sont disponibles pour de nombreux langages de programmation populaires. Chaque SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d'applications par les développeurs dans leur langage préféré.

Documentation SDK	Exemples de code
AWS SDK for C++	AWS SDK for C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK for Go	AWS SDK for Go exemples de code
AWS SDK for Java	AWS SDK for Java exemples de code
AWS SDK for JavaScript	AWS SDK for JavaScript exemples de code
Kit AWS SDK pour Kotlin	Kit AWS SDK pour Kotlin exemples de code
AWS SDK for .NET	AWS SDK for .NET exemples de code

Documentation SDK	Exemples de code
AWS SDK for PHP	AWS SDK for PHP exemples de code
AWS Tools for PowerShell	Outils pour des exemples PowerShell de code
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) exemples de code
AWS SDK for Ruby	AWS SDK for Ruby exemples de code
Kit AWS SDK pour Rust	Kit AWS SDK pour Rust exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
Kit AWS SDK pour Swift	Kit AWS SDK pour Swift exemples de code

Pour des exemples spécifiques à Amazon SNS, consultez Exemples de code pour Amazon SNS à l'aide de kits SDK AWS.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code en utilisant le lien Provide feedback (Fournir un commentaire) en bas de cette page.

Sources et destinations des évènements Amazon SNS

Amazon SNS peut recevoir des notifications basées sur des évènements de la part de nombreuses sources AWS sources et diffuser les notifications vers des destinations application à application (A2A) et application à personne (A2P). Cette section répertorie les sources et les destinations d'évènements prises en charge, et fournit des liens pour obtenir plus d'informations.

Rubriques

Sources des évènements Amazon SNS

Cette page répertorie les services AWS qui peuvent publier des évènements sur des rubriques Amazon SNS, regroupés par Catégories de produits AWS.

> **Note**
>
> Amazon SNS introduit les rubriques FIFO en octobre 2020. Actuellement, la plupart des services AWS prennent en charge l'envoi d'évènements vers des rubriques standard uniquement.

Services analytiques

Service AWS	Avantages offerts par Amazon SNS
Amazon Athena – Vous permet d'analyser des données dans Amazon S3 à l'aide du langage SQL standard.	Recevez des notifications lorsque les limites de contrôle sont dépassées. Pour plus d'informations, consultez Définition des limites de contrôle d'utilisation des données dans le Guide de l'utilisateur Amazon Athena.
AWS Data Pipeline – Aide à automatiser le mouvement et la transformation de données.	Recevez des notifications sur l'état des composants du pipeline. Pour plus d'informa

Service AWS	Avantages offerts par Amazon SNS
	tions, consultez SnsAlarm dans le Guide du développeur AWS Data Pipeline.
Amazon Redshift – Gère toutes les tâches de configuration, d'exploitation et de mise à l'échelle d'un entrepôt de données.	Recevoir des notifications pour les évènement s Amazon Redshift. Pour plus d'informations, consultez Notifications d'évènements Amazon Redshift dans le Guide de gestion Amazon Redshift.

Services d'intégration d'applications

Service AWS	Avantages offerts par Amazon SNS
Amazon EventBridge — Fournit un flux de données en temps réel à partir de vos propres applications, applications software-as-a-serv ice (SaaS) et AWS services et achemine ces données vers des cibles, notamment Amazon SNS. EventBridge s'appelait auparavant CloudWatch Events.	Recevez des notifications d' EventBrid ge événements. Pour plus d'informations, consultez les EventBridgecibles Amazon dans le guide de EventBridge l'utilisateur Amazon.
AWS Step Functions – Vous permet de combiner les fonctions AWS Lambda et d'autres services AWS pour créer des applicati ons stratégiques.	Recevez une notification des évènements de Step Functions. Pour plus d'informations, consultez Appeler Amazon SNS avec Step Functions dans le Guide du développeur AWS Step Functions.

Services de gestion de la facturation et des coûts

Service AWS	Avantages offerts par Amazon SNS
AWS Billing and Cost Management – Fournit des fonctions qui vous aident à surveiller vos coûts et à payer votre facture.	Recevez des notifications de budget, des notifications de modification de prix et des alertes en cas d'anomalie. Pour plus d'informa

Service AWS	Avantages offerts par Amazon SNS
	tions, consultez les pages suivantes dans le Guide de l'utilisateur AWS Billing : • Création d'une rubrique Amazon SNS pour les notifications de budget • Configuration des notifications • Détection de dépenses inhabituelles avec AWS la détection des anomalies de coûts

Services d'applications métier

Service AWS	Avantages offerts par Amazon SNS
Amazon Chime – vous permet de rencontrer, de discuter et de passer des appels professionnels à l'intérieur et à l'extérieur de votre organisation.	Recevez des notifications pour les évènements de réunion importants. Pour plus d'informations, consultez Notifications d'évènements du SDK Amazon Chime dans le Guide du développeur Amazon Chime.

Sevices de calcul

Service AWS	Avantages offerts par Amazon SNS
Amazon EC2 Auto Scaling – Vous aide à disposer du nombre approprié d'instances Amazon Elastic Compute Cloud (Amazon EC2) disponibles pour le traitement de la charge de votre application.	Recevez des notifications lorsqu'Auto Scaling lance ou arrête des instances Amazon EC2 dans votre groupe Auto Scaling. Pour plus d'informations, consultez Réception de notifications Amazon SNS lors du dimensionnement de votre groupe Auto Scaling dans le Guide de l'utilisateur Amazon EC2 Auto Scaling.

Service AWS	Avantages offerts par Amazon SNS
EC2 Image Builder : permet d'automatiser la création, la gestion et le déploiement d'images personnalisées, sécurisées et de serveur préinstallées up-to-date et préconfigurées avec des logiciels et des paramètres répondant à des normes informatiques spécifiques.	Recevez des notifications lorsque les créations sont terminées. Pour plus d'informations, consultez Suivi des dernières images de serveur dans les pipelines EC2 Image Builder sur le Blog AWS Compute.
AWS Elastic Beanstalk – Gère les détails de l'approvisionnement des capacités, de l'équilib rage de la charge, de la mise à l'échelle et de la surveillance de l'état de l'application, et fournit la surveillance de l'état de l'application.	Recevez des notifications pour les évènement s importants qui affectent votre application. Pour plus d'informations, consultez Notificat ions d'environnement Elastic Beanstalk avec Amazon SNS dans le Guide du développeur AWS Elastic Beanstalk.
AWS Lambda – Vous permet d'exécuter du code sans avoir à allouer ou gérer des serveurs.	Recevoir des données de sortie de fonction en définissant une rubrique SNS comme une file d'attente de lettres mortes Lambda ou une destination Lambda. Pour plus d'informations, consultez Appel asynchrone dans le Guide du développeur AWS Lambda.
Amazon Lightsail – Aide les développeurs à commencer à utiliser AWS pour créer des sites web ou des applications web.	Recevez des notifications lorsqu'une métrique pour une instance, une base de données ou un équilibreur de charge franchit un seuil donné. Pour plus d'informations, consultez Ajout de contacts de notification dans Amazon Lightsail dans le Guide du développeur Amazon Lightsail.

Services de conteneurs

Service AWS	Avantages offerts par Amazon SNS
Amazon EKS Distro – Vous permet de créer des clusters fiables et sécurisés partout où vos applications sont déployées.	Suivez les mises à jour et les correctifs de sécurité pour les clusters créés avec Amazon EKS Distro. Pour plus d'informations, consultez Présentation d'Amazon EKS Distro – une distribution Kubernetes open source utilisée par Amazon EKS.
Amazon Elastic Container Service (Amazon ECS) – Vous permet d'exécuter, arrêter et gérer des conteneurs sur un cluster.	Recevez des notifications lorsqu'une nouvelle AMI optimisée pour Amazon ECS est disponibl e. Pour plus d'informations, consultez S'abonner aux notifications de mise à jour d'AMI optimisée pour Amazon ECS dans le Guide du développeur Amazon Elastic Container Service.

Services d'engagement client

Service AWS	Avantages offerts par Amazon SNS
Amazon Connect – Vous permet de configure r un centre de contact cloud omnicanal pour vous connecter à vos clients.	Recevez des alertes et des validations. Pour plus d'informations, consultez La puissance d'AWS avec Amazon Connect dans le Guide de l'administrateur Amazon Connect.
Amazon Pinpoint – Vous aide à attirer vos clients en leur envoyant des e-mails, des SMS et des messages vocaux, ainsi que des notificat ions push.	Configurez des SMS bidirectionnels, ce qui vous permet de recevoir des messages de vos clients. Pour plus d'informations, consultez Utilisation de SMS bidirectionnels dans Amazon Pinpoint dans le Guide de l'utilisateur Amazon Pinpoint.

Service AWS	Avantages offerts par Amazon SNS
Amazon Simple Email Service (Amazon SES) – Offre un moyen économique d'envoyer et de recevoir des e-mails en utilisant vos propres adresses et domaines e-mail.	Recevez des notifications de retours à l'expéditeur, de réclamations et de messages remis à l'expéditeur. Pour plus d'informations, consultez Configuration des notifications Amazon SNS pour Amazon SES dans le Guide du développeur Amazon Simple Email Service.

Services de base de données

Service AWS	Avantages offerts par Amazon SNS
AWS Database Migration Service – migre les données des bases de données sur site vers le Cloud AWS.	Recevoir des notifications lorsque des évènements AWS DMS se produisent ; par exemple, lorsqu'une instance de réplication est créée ou supprimée. Pour plus d'informations, consultez Utilisation d'évènements et de notifications dans AWS Database Migration Service dans le Guide de l'utilisateur AWS Database Migration Service.
Amazon DynamoDB – Service de base de données NoSQL entièrement géré offrant des performances exceptionnelles et prévisibles en termes de rapidité et d'évolutivité.	Recevoir des notifications lorsque des évènements de maintenance se produisent. Pour plus d'informations, consultez Personnalisation des paramètres de cluster DAX dans le Guide du développeur Amazon DynamoDB.
Amazon ElastiCache — Fournit un cache en mémoire performant, redimensionnable et économique, tout en éliminant la complexité associée au déploiement et à la gestion d'un environnement de cache distribué.	Recevez des notifications lorsque des évènements importants se produisent. Pour plus d'informations, consultez les sections Notifications d'événements et Amazon SNS dans le guide de l'utilisateur d'Amazon ElastiCache for Memcached.

Service AWS	Avantages offerts par Amazon SNS
Amazon Neptune – Permet de créer et d'exécuter des applications qui fonctionnent avec des ensembles de données hautement connectés.	Recevez des notifications lorsqu'un évènement Neptune se produit. Pour plus d'informations, consultez Utilisation de la notification d'évènement Neptune dans le Guide de l'utilisateur Neptune.
Amazon Redshift – Gère toutes les tâches de configuration, d'exploitation et de mise à l'échelle d'un entrepôt de données.	Recevoir des notifications pour les évènements Amazon Redshift. Pour plus d'informations, consultez Notifications d'évènements Amazon Redshift dans le Guide de gestion Amazon Redshift.
Amazon Relational Database Service – Facilite la configuration, l'exploitation et la mise à l'échelle d'une base de données relationnelle dans le cloud AWS.	Recevez des notifications pour les évènements Amazon RDS. Pour plus d'informations, consultez Utilisation de la notification d'évènement Amazon RDS dans le Guide de l'utilisateur Amazon RDS.

Services et outils pour développeurs

Service AWS	Avantages offerts par Amazon SNS
AWS CodeBuild – Compile votre code source, exécute des tests unitaires et produit des artefacts prêts à être déployés.	Recevez des notifications lorsque les créations réussissent, échouent ou passent d'une phase de création à une autre. Pour plus d'informations, consultez la section Exemple de génération de notifications CodeBuild dans le guide de AWS CodeBuild l'utilisateur.
AWS CodeCommit – Fournit un contrôle de version pour le stockage et la gestion privés des ressources dans le cloud.	Recevez des notifications concernant les événements CodeCommit du référentiel. Pour plus d'informations, consultez Exemple : Créer un déclencheur AWS CodeCommit pour une

Service AWS	Avantages offerts par Amazon SNS
	rubrique Amazon SNS dans le Guide de l'utilisateur AWS CodeCommit.
AWS CodeDeploy – Automatise les déploiements d'applications sur des instances Amazon EC2, des instances locales, des fonctions Lambda sans serveur ou des services Amazon EC2.	Recevez des notifications pour les CodeDeploy déploiements ou les événements liés aux instances. Pour plus d'informations, voir Création d'un déclencheur pour un CodeDeploy événement dans le Guide de AWS CodeDeploy l'utilisateur.
Amazon CodeGuru — Collecte les données de performances d'exécution de vos applications en ligne et fournit des recommandations qui peuvent vous aider à affiner les performances de vos applications.	Recevoir des notifications lorsque des anomalies se produisent. Pour plus d'informations, consultez la section Gestion des anomalies et rapports de recommandation dans le guide de CodeGuru l'utilisateur Amazon.
AWS CodePipeline – Automatise les étapes requises pour publier les modifications logicielles en continu.	Recevez des notifications sur les actions d'approbation. Pour plus d'informations, voir Gérer les actions d'approbation CodePipeline dans le Guide de AWS CodePipeline l'utilisateur.
AWS CodeStar – Permet de créer, gérer et utiliser des projets de développement logiciel sur AWS.	Recevez des notifications sur les évènements qui se produisent dans les ressources que vous utilisez. Pour plus d'informations, consultez Configuration des rubriques Amazon SNS pour les notifications dans le Guide de l'utilisateur de la console Outils pour les développeurs.

Services web et mobiles frontaux

Service AWS	Avantages offerts par Amazon SNS
Amazon Pinpoint – Vous aide à attirer vos clients en leur envoyant des e-mails, des SMS	Configurez des SMS bidirectionnels, ce qui vous permet de recevoir des messages de vos

Service AWS	Avantages offerts par Amazon SNS
et des messages vocaux, ainsi que des notifications push.	clients. Pour plus d'informations, consultez Utilisation de SMS bidirectionnels dans Amazon Pinpoint dans le Guide de l'utilisateur Amazon Pinpoint.

Services de développement de jeux

Service AWS	Avantages offerts par Amazon SNS
Amazon GameLift — Fournit des solutions pour héberger des serveurs de jeux multijoueurs basés sur des sessions dans le cloud, y compris un service entièrement géré pour le déploiement, l'exploitation et le dimensionnement des serveurs de jeux.	Recevez des notifications d'évènements de mise en relation et de file d'attente. Pour plus d'informations, consultez les pages suivantes : • Pour les notifications de matchmaking, consultez Configurer les notifications FlexMatch d'événements dans le manuel Amazon GameLift FlexMatch Developer Guide. • Pour les notifications relatives aux files d'attente, consultez la section Configurer les notifications d'événements pour le placement des sessions de jeu dans le manuel Amazon GameLift Developer Guide.

Services d'Internet des objets (IoT)

Service AWS	Avantages offerts par Amazon SNS
AWS IoT Core – Fournit les services cloud qui connectent vos appareils IoT aux autres appareils et aux services cloud AWS.	Recevoir des notifications d'évènements AWS IoT Core. Pour plus d'informations, consultez Création d'une règle Amazon SNS dans le Guide du développeur AWS IoT.

Service AWS	Avantages offerts par Amazon SNS
AWS IoT Device Defender – Vous permet d'auditer la configuration de vos appareils, de surveiller les appareils connectés afin de détecter les comportements anormaux, et d'atténuer les risques pour la sécurité.	Recevez des alarmes lorsqu'un appareil enfreint un comportement. Pour plus d'informations, consultez Utilisation de la détection AWS IoT Device Defender dans le Guide du développeur AWS IoT.
AWS IoT Events – Vous permet de pour surveiller vos flottes d'équipements et d'appareils pour détecter les défaillances ou les modifications de fonctionnement, et pour déclencher des actions lorsque ces évènements se produisent.	Recevoir des notifications d'évènements AWS IoT Events. Pour plus d'informations, consultez Amazon Simple Notification Service dans le Guide du développeur AWS IoT Events.
AWS IoT Greengrass – Étend AWS aux appareils physiques pour une exploitation en local des données qu'ils génèrent, tout en continuant à utiliser les ressources du cloud pour la gestion, l'analyse et le stockage de longue durée.	Recevoir des notifications d'évènements AWS IoT Greengrass. Pour plus d'informations, consultez Connecteur SNS dans le Guide du développeur AWS IoT Greengrass Version 1.

Services Machine Learning

Service AWS	Avantages offerts par Amazon SNS
Amazon CodeGuru — Collecte les données de performances d'exécution de vos applications en ligne et fournit des recommandations qui peuvent vous aider à affiner les performances de vos applications.	Recevoir des notifications lorsque des anomalies se produisent. Pour plus d'informations, consultez la section Gestion des anomalies et rapports de recommandation dans le guide de CodeGuru l'utilisateur Amazon.
Amazon DevOps Guru — Génère des informations opérationnelles à l'aide de l'apprentissage automatique pour vous aider à améliorer les	Transférez les informations et les confirmations. Pour plus d'informations, consultez la section Fournir des informations opérationnelles basées sur le ML à vos équipes de garde

Service AWS	Avantages offerts par Amazon SNS
performances de vos applications opération nelles.	via PagerDuty Amazon DevOps Guru sur le blog de AWSgestion et de gouvernance.
Amazon Lookout for Metrics – Détecte les anomalies dans vos données, détermine leurs causes profondes et vous permet de prendre rapidement des mesures.	Recevez des notifications d'anomalies. Pour plus d'informations, consultez Utilisation d'Amazon SNS avec Lookout for Metrics dans le Guide du développeur Amazon Lookout for Metrics.
Amazon Rekognition – Vous permet d'ajouter une analyse image et vidéo à vos applications	Recevez des notifications des résultats de la demande. Pour plus d'informations, consultez Référence : Notification des résultats d'une analyse vidéo dans le Guide du développeur Amazon Rekognition.
Amazon SageMaker — Permet aux data scientists et aux développeurs de créer et de former des modèles d'apprentissage automatiq ue, puis de les déployer directement dans un environnement hébergé prêt pour la production.	Recevez des notifications lorsqu'un objet de données est labélisé. Pour plus d'informa tions, consultez la section Création d'une tâche d'étiquetage en streaming dans le manuel Amazon SageMaker Developer Guide.

Services de gestion et de gouvernance

Service AWS	Avantages offerts par Amazon SNS
AWS Chatbot— Permet aux équipes de développement logiciel DevOps et aux équipes de développement d'utiliser les forums de discussion Amazon Chime et Slack pour surveiller les événements opérationnels dans le cloud et y répondre. AWS	Envoyez des notifications aux salles de chat. Pour plus d'informations, consultez Configura tion de AWS Chatbot dans le Guide de l'administrateur AWS Chatbot.
AWS CloudFormation – Permet de créer et d'allouer des déploiements d'infrastructure AWS de manière prévisible et répétée.	Recevez des notifications lorsque des piles sont créées et mises à jour. Pour plus d'informa tions, consultez Paramétrage des options de

Service AWS	Avantages offerts par Amazon SNS
	la pileAWS CloudFormation dans le Guide de l'utilisateur AWS CloudFormation.
AWS CloudTrail – Fournit l'historique des évènements de votre activité Compte AWS.	Recevez des notifications lorsque vous CloudTrail publiez de nouveaux fichiers journaux dans votre compartiment Amazon S3. Pour plus d'informations, consultez la section Configuration des notifications Amazon SNS CloudTrail dans le guide de l'AWS CloudTrai lutilisateur.
Amazon CloudWatch — Surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel.	Recevez des notifications lorsque les alarmes changent d'état. Pour plus d'informations, consultez la section Utilisation des CloudWatch alarmes Amazon dans le guide de CloudWatch l'utilisateur Amazon.
AWS Config – Fournit une vue détaillée de la configuration des ressources AWS de votre compte Compte AWS.	Recevez des notifications lorsque les ressource s sont mises à jour ou lorsque AWS Config évalue les règles personnalisées ou gérées par rapport à vos ressources. Pour plus d'informa tions, consultez Notifications que AWS Config envoie à une rubrique SNS and Exemple de notifications de modification d'élément de configuration dans le Guide du développeur AWS Config.
AWS Control Tower – Vous permet de configurer et de gérer un environnement AWS multicompte sécurisé et conforme.	Utilisez les alertes pour vous aider à prévenir la dérive dans votre zone de destination et recevez des notifications liées à la conformité. Pour plus d'informations, consultez Suivi des alertes via Amazon Simple Notification Service dans le Guide de l'utilisateur AWS Control Tower.

Service AWS	Avantages offerts par Amazon SNS
AWS License Manager – Vous aide à gérer vos licences logicielles à partir de fournisseurs de logiciels de manière centralisée sur AWS et vos environnements sur site.	Recevez des notifications et des alertes du License Manager. Pour plus d'informations, consultez les sections Paramètres du License Manager dans le guide de l'utilisateur du License Manager et Création d' ServiceNo w incidents pour AWS License Manager les notifications sur le blog AWS Management & Governance.
AWS Service Catalog – Permet aux administr ateurs informatiques de créer, gérer et distribue r des portefeuilles de produits approuvés aux utilisateurs finaux, qui peuvent accéder aux produits dont ils ont besoin via un portail personnalisé.	Recevez des notifications sur les évènement s de pile. Pour plus d'informations, consultez Contraintes de notification AWS Service Catalog dans le Guide de l'administrateur Service Catalog.
AWS Systems Manager – Vous permet d'afficher et de contrôler votre infrastructure sur AWS.	Recevez des notifications sur l'état des composants. Pour plus d'informations, consultez Surveillance des changements d'état du Systems Manager à l'aide des notifications Amazon SNS dans le Guide de l'utilisateur AWS Systems Manager.

Services multimédias

Service AWS	Avantages offerts par Amazon SNS
Amazon Elastic Transcoder – Vous permet de convertir des fichiers multimédias stockés dans Amazon S3 aux formats requis par les appareils de lecture grand public.	Recevez des notifications lorsque les alarmes changent d'état. Pour plus d'informations, consultez Notifications d'état des tâches dans le Guide du développeur Amazon Elastic Transcoder.

Services de migration et de transfert

Service AWS	Avantages offerts par Amazon SNS
AWS Application Discovery Service – Vous aide à planifier votre migration vers le cloud AWS en recueillant les données d'utilisation et de configuration relatives à vos serveurs sur site.	Recevez des notifications d'évènements via AWS CloudTrail. Pour plus d'informations, consultez Journalisation des appels d'API Application Discovery Service avec AWS CloudTrail dans le Guide de l'utilisateur Application Discovery Service.
AWS Database Migration Service – Migre les données des bases de données sur site vers le Cloud AWS.	Recevoir des notifications lorsque des évènements AWS DMS se produisent ; par exemple, lorsqu'une instance de réplication est créée ou supprimée. Pour plus d'informations, consultez Utilisation d'évènements et de notifications dans AWS Database Migration Service dans le Guide de l'utilisateur AWS Database Migration Service.
AWS Snowball— Utilise des périphériques de stockage physiques pour transférer rapidement de grandes quantités de données entre Amazon S3 et votre site de stockage de données sur faster-than-internet site.	Recevoir des notifications pour les travaux Snowball. Pour plus d'informations, consultez les ressources suivantes : • Notifications Snowball dans le AWS Snowball Guide de l'utilisateur • Étape 5 : Choisissez vos préférences de notification dans le AWS Guide du développeur Snowball Edge • Étape 5 : Choisissez vos préférences de notification dans le AWS Guide du développeur Snowcone

Services de réseaux et de diffusion de contenu

Service AWS	Avantages offerts par Amazon SNS
Amazon API Gateway : vous permet de créer et de déployer vos propres REST et WebSocket API à n'importe quelle échelle.	Recevez des messages postés sur un point de terminaison API Gateway. Pour plus d'informations, consultez Tutoriel : création d'une API REST API Gateway avec intégration AWS dans le Guide du développeur API Gateway.
Amazon CloudFront — Accélère la distribution de votre contenu Web statique et dynamique, tel que les fichiers .html, .css, .php, images et médias.	Recevez des notifications lorsque des alarmes basées sur CloudFront des mesures spécifiées se produisent. Pour plus d'informations, consultez la section Configuration des alarmes pour recevoir des notifications dans le manuel Amazon CloudFront Developer Guide.
AWS Direct Connect – Relie votre réseau interne à un emplacement AWS Direct Connect via un câble en fibres optiques Ethernet standard.	Recevez des notifications lorsque les alarmes qui surveillent l'état d'une connexion AWS Direct Connect changent d'état. Pour plus d'informations, consultez la section Création d' CloudWatch alarmes pour surveiller AWS Direct Connect les connexions dans le Guide de AWS Direct Connect l'utilisateur.
Elastic Load Balancing – Distribue automatiquement votre trafic entrant sur plusieurs cibles (par exemple, des instances Amazon EC2, des conteneurs et des adresses IP) dans une ou plusieurs zones de disponibilité.	Recevez des notifications des alarmes que vous avez créées pour les évènements d'équilibrage de charge. Pour plus d'informations, voir Créer des CloudWatch alarmes pour votre équilibreur de charge dans le Guide de l'utilisateur des équilibreurs de charge classiques.
Amazon Route 53 – Fournit l’enregistrement de domaine, le routage DNS et la vérification du bon fonctionnement.	Recevez des notifications lorsque le statut d'une surveillance de l'état n'est pas sain. Pour plus d'informations, consultez Pour recevoir une notification Amazon SNS lorsque le statut d'une vérification de l'état n'est pas

Service AWS	Avantages offerts par Amazon SNS
	sain (console) dans le Guide du développeur Amazon Route 53.
Amazon Virtual Private Cloud (Amazon VPC) – Vous permet de lancer des ressources AWS dans un réseau virtuel défini par vos soins.	Recevez des notifications pour les évènements spécifiques qui se produisent sur les points de terminaison de l'interface. Pour plus d'informations, consultez Créer et gérer une notification pour un service de point de terminaison dans le Guide de l'utilisateur Amazon VPC.

Services de sécurité, d'identité et de conformité

Service AWS	Avantages offerts par Amazon SNS
AWS Directory Service – Offre plusieurs façons d'utiliser Microsoft Active Directory (AD) avec d'autres services AWS.	Recevez des e-mails ou des messages texte (SMS) lorsque le statut de votre annuaire change. Pour plus d'informations, consultez Configurer les notifications d'état d'annuaire dans le AWS Directory Service Guide d'administration.
Amazon GuardDuty — Assure une surveillance continue de la sécurité pour aider à identifier les activités inattendues, potentiellement non autorisées ou malveillantes dans votre AWS environnement.	Recevez des notifications sur les types de résultats nouvellement publiés, les mises à jour des types de résultats existants et les autres modifications de fonctionnalités. Pour plus d'informations, consultez la rubrique Abonnement aux GuardDuty annonces SNS du guide de GuardDuty l'utilisateur Amazon.
Amazon Inspector – Teste l'accessibilité réseau de vos instances Amazon EC2 et l'état de sécurité de vos applications qui s'exécutent sur ces instances.	Recevez des notifications pour les évènements Amazon Inspector. Pour plus d'informations, consultez Configuration d'une rubrique SNS pour les notifications Amazon Inspector dans le Guide de l'utilisateur Amazon Inspector.

Service AWS	Avantages offerts par Amazon SNS
AWS Security Hub – Automatise les contrôles de sécurité AWS et centralise les alertes de sécurité.	Recevez des notifications sur les annonces AWS Security Hub, y compris des notifications sur les contrôles ou les normes AWS Security Hub qui ont été ajoutés, modifiés ou retirés. Pour en savoir plus, consultez Abonnement aux annonces AWS Security Hub avec Amazon SNS.

Services sans serveur

Service AWS	Avantages offerts par Amazon SNS
Amazon DynamoDB – Service de base de données NoSQL entièrement géré offrant des performances exceptionnelles et prévisibles en termes de rapidité et d'évolutivité.	Recevoir des notifications lorsque des évènements de maintenance se produisent. Pour plus d'informations, consultez Personnalisation des paramètres de cluster DAX dans le Guide du développeur Amazon DynamoDB.
Amazon EventBridge — Fournit un flux de données en temps réel à partir de vos propres applications, applications software-as-a-service (SaaS) et AWS services et achemine ces données vers des cibles, notamment Amazon SNS. EventBridge s'appelait auparavant CloudWatch Events.	Recevez des notifications d' EventBridge événements. Pour plus d'informations, consultez les EventBridgecibles Amazon dans le guide de EventBridge l'utilisateur Amazon.
AWS Lambda – Vous permet d'exécuter du code sans avoir à allouer ou gérer des serveurs.	Recevoir des données de sortie de fonction en définissant une rubrique SNS comme une file d'attente de lettres mortes Lambda ou une destination Lambda. Pour plus d'informations, consultez Appel asynchrone dans le Guide du développeur AWS Lambda.

Services de stockage

Service AWS	Avantages offerts par Amazon SNS
AWS Backup – Vous aide à centraliser et à automatiser la sauvegarde des données sur les services AWS dans le cloud et sur site.	Recevoir des notifications d'évènementsAWS Backup. Pour plus d'informations, consultez Utilisation d'Amazon SNS pour suivre les évènements AWS Backup dans le Guide du développeur AWS Backup.
Amazon Elastic File System – Fournit un stockage de fichiers pour vos instances Amazon EC2.'	Recevez des notifications des alarmes que vous avez créées pour les évènement s Amazon EFS. Pour plus d'informations, consultez Outils de surveillance automatisée dans le Guide de l'utilisateur Amazon Elastic File System.
Amazon S3 Glacier – Fournit un stockage pour les données rarement utilisées.	Définissez une configuration de notification sur un coffre de sorte qu'un message soit envoyé à une rubrique SNS dès que la tâche est terminée. Pour plus d'informations, consultez Configuration des notifications de coffre-fort dans Amazon S3 Glacier dans le Guide du développeur Amazon S3 Glacier.
Amazon Simple Storage Service (Amazon S3) – Fournit un stockage d'objets.	Recevez des notifications lorsque des modificat ions sont apportées à un compartiment Amazon S3 ou dans les rares cas où les objets ne sont pas répliqués vers leur région de destination. Pour plus d’informations, consultez Démonstration : configurer un compartiment pour les notifications (rubrique SNS ou file d’attente SQS) et Surveiller la progression avec des métriques de réplication et des notificat ions d’évènements Amazon S3 dans le Guide de l’utilisateur de la console Amazon Simple Storage Service.

Service AWS	Avantages offerts par Amazon SNS
AWS Snowball— Utilise des périphériques de stockage physiques pour transférer rapidement de grandes quantités de données entre Amazon S3 et votre site de stockage de données sur faster-than-internet site.	Recevoir des notifications pour les travaux Snowball. Pour plus d'informations, consultez les ressources suivantes : • Notifications Snowball dans le AWS Snowball Guide de l'utilisateur • Étape 5 : Choisissez vos préférences de notification dans le AWS Guide du développeur Snowball Edge • Étape 5 : Choisissez vos préférences de notification dans le Guide du développeur AWS Snowcone

Sources supplémentaires des évènements

Source	Avantages offerts par Amazon SNS
Mises à jour quotidiennes des fonctionnalités AWS	Recevez rapidement des informations détaillées sur les versions et les mises à jour AWS via une rubrique Amazon SNS. Ces versions incluent les points de terminaison Amazon VPC Régions AWSServices AWS, Services AWS intégrés à AWS Service Quotas, les types d'instances Amazon EC2, les types d'instances Amazon, les types d'instances Amazon SageMaker Nimble Studio, les versions du moteur de base de données Amazon RDS et les versions Amazon MSK Apache Kafka. Pour plus d'informations, consultez S'abonner aux mises à jour quotidiennes des fonctionnalités

Source	Avantages offerts par Amazon SNS
	AWS via Amazon SNS dans le Blog d'actualités AWS.
AWSPlages d’adresses IP	Recevez des notifications pour les modifications apportées aux plages IP AWS via une rubrique Amazon SNS. Pour plus d'informations, consultez Notifications des plages d'adresses IP AWS dans la Référence générale d'Amazon Web Services et S'abonner aux modifications d'adresses IP publiques AWS via Amazon SNS dans le Blog d'actualités AWS.

Pour plus d'informations sur le calcul pilotée par les évènements, consultez les sources suivantes :

- Qu'est-ce qu'une architecture pilotée par les évènements ?
- Calcul piloté par les évènements avec Amazon SNS et AWS services de calcul, stockage, base de données et réseaux sur le AWS Blog Compute
- Enrichir les architectures pilotées par les évènements avec AWS Event Fork Pipelines sur le AWS Blog Compute

Destinations des évènements Amazon SNS

Cette page répertorie toutes les destinations pouvant recevoir des informations sur les événements, regroupées par messagerie application-to-application (A2A) et notifications application-to-person (A2P).

> **Note**
>
> Amazon SNS introduit les rubriques FIFO en octobre 2020. Actuellement, la plupart des services AWS prennent en charge la réception d'évènements provenant de rubriques SNS standard uniquement. Amazon SQS prend en charge la réception d'évènements à partir de rubriques SNS standard et FIFO.

Destinations A2A

Destination de l'événement	Avantages offerts par Amazon SNS
Amazon Data Firehose	Diffuser des évènements dans des flux de diffusion à des fins d'archivage et d'analyse . Grâce aux flux de diffusion, vous pouvez diffuser des événements vers des AWS destinations telles qu'Amazon Simple Storage Service (Amazon S3), Amazon Redshift et OpenSearch Amazon OpenSearch Service (Service), ou vers des destinations tierces telles que Datadog, New Relic, MongoDB et Splunk. Pour plus d'informations, consultez Streams de diffusion de Fanout to Firehose.
AWS Lambda	Fournir des évènements aux fonctions pour déclencher l'exécution de la logique métier personnalisée. Pour plus d'informations, consultez Diffusion dans les fonctions Lambda.
Amazon SQS	Diffuser des évènements vers des files d'attente à des fins d'intégration d'application. Pour plus d'informations, consultez Distribution ramifiée vers des files d'attente Amazon SQS.
AWS Event Fork Pipelines	Fournissez des évènements pour la sauvegard e et le stockage d'évènements, la recherche et l'analyse d'évènements ou des pipelines de rediffusion d'évènements. Pour plus d'informa tions, consultez Diffusion dans les Event Fork Pipelines AWS.
HTTP/S	Diffuser des évènements à des webhooks externes. Pour plus d'informations, consultez Diffusion en éventail vers les points de terminaison HTTP(S).

Destinations A2P

Destination de l'événement	Avantages offerts par Amazon SNS
SMS	Diffuser des évènements sur les téléphones mobiles sous forme de messages texte. Pour plus d'informations, consultez Messagerie texte mobile (SMS).
E-mails	Diffuser les évènements dans les boîtes de réception sous forme d'e-mails. Pour plus d'informations, consultez Notifications par e-mail.
Point de terminaison de plateforme	Diffusez des évènements sur les téléphone s mobiles sous forme de notifications push natives. Pour plus d'informations, consultez Notifications push mobile.
AWS Chatbot	Diffusez des évènements sur les salons de chat Amazon Chime ou les chaînes Slack. Pour plus d'informations, consultez les pages suivantes dans le Guide de l'administrateur AWS Chatbot : • Configuration AWS Chatbot avec Amazon Chime • Configuration AWS Chatbot avec Slack • Utilisation d'AWS Chatbot avec d'autres services AWS
PagerDuty	Fournissez des informations opérationnelles aux équipes de garde. Pour plus d'informations, consultez la section Fournir des informations opérationnelles basées sur le ML à vos équipes

Destination de l'événement	Avantages offerts par Amazon SNS
	de garde via PagerDuty Amazon DevOps Guru sur le blog de AWSgestion et de gouvernance.

> **Note**
>
> Vous pouvez fournir à la fois des évènements AWS natifs et des évènements personnalisés pour les applications de chat :
>
> - Évènements AWS natifs – Vous pouvez utiliser AWS Chatbot pour envoyer des évènements AWS natifs, via les rubriques Amazon SNS, à Amazon Chime et Slack. L'ensemble d'AWSévénements natifs pris en charge inclut des événements provenant de AWS Billing and Cost Management AWS HealthAWS CloudFormation, CloudWatch,, Amazon, etc. Pour plus d'informations, consultez Utilisation de AWS Chatbot avec d'autres services dans le Guide de l'administrateurAWS Chatbot.
> - Evénements personnalisés – Vous pouvez également envoyer vos évènements personnalisés via les rubriques Amazon SNS, à Amazon Chime, Slack et Microsoft Teams. Pour ce faire, publiez des évènements personnalisés dans une rubrique SNS qui envoie les évènements à une fonction Lambda abonnée. La fonction Lambda utilise ensuite le webhook de l'application de chat pour livrer les évènements aux destinataires. Pour plus d'informations, consultez Comment utiliser les webhooks pour publier des messages Amazon SNS sur Amazon Chime, Slack ou Microsoft Teams ?

Configuration de l'accès à Amazon SNS

Avant de pouvoir utiliser Amazon SNS, vous devez suivre la procédure ci-dessous.

Rubriques

Étape 1 : Création d'un utilisateur Compte AWS et d'un utilisateur IAM

Pour accéder à AWS un service, vous devez d'abord créer un Compte AWS. Vous pouvez utiliser votre Compte AWS pour consulter vos rapports d'activité et d'utilisation et pour gérer l'authentification et l'accès.

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un utilisateur Compte AWS root est créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Une fois que vous vous êtes inscrit Compte AWS, sécurisez votre utilisateur Compte AWS root AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre utilisateur Compte AWS root

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Étapes suivantes

Maintenant que vous êtes prêt à travailler avec Amazon SNS, commencez par créer une rubrique, créez ensuite un abonnement pour la rubrique, puis publiez un message dans la rubrique et supprimez l'abonnement et la rubrique.

Démarrage avec Amazon SNS

Cette section vous aide à vous familiariser avec Amazon SNS en vous montrant comment gérer les rubriques, les abonnements et les messages avec la console Amazon SNS.

Rubriques

Prérequis

Avant de commencer, complétez les étapes détaillées dans Configuration de l'accès à Amazon SNS.

Étape 1 : Créer une rubrique

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation de gauche, choisissez Rubriques.
3. Sur la page Rubriques, choisissez Créer une rubrique.
4. Par défaut, la console crée une rubrique FIFO. Choisissez Standard.
5. Dans la section Détails, entrez un nom pour le sujet, tel que `MyTopic`.
6. Faites défiler la page jusqu'en bas et choisissez Créer une rubrique.

   La console ouvre la page Détails de la nouvelle rubrique.

Étape 2 : Créer un abonnement à la rubrique

1. Dans le volet de navigation de gauche, choisissez Abonnements.
2. Sur la page Abonnements, choisissez Créer un abonnement.

3. Dans la page Créer un abonnement, choisissez le champ ARN de rubrique pour voir une liste des rubriques dans votre Compte AWS.
4. Choisissez la rubrique que vous avez créée à l'étape précédente.
5. Pour Protocole, choisissez E-mail.
6. Pour Point de terminaison, entrez une adresse e-mail qui peut recevoir les notifications.
7. Choisissez Créer un abonnement.

   La console ouvre la page Détails du nouvel abonnement.
8. Vérifiez votre boîte de réception et choisissez Confirmer l'abonnement dans l'e-mail depuis Notifications AWS. L'ID de l'expéditeur est généralement « no-reply@sns.amazonaws.com ».
9. Amazon SNS ouvre votre navigateur web et affiche une confirmation d'abonnement avec votre ID d'abonnement.

Étape 3 : Publier un message dans la rubrique

1. Dans le panneau de navigation de gauche, choisissez Rubriques.
2. Sur la page Rubriques, choisissez la rubrique que vous avez créée précédemment, puis cliquez sur Publier le message.

   La console ouvre la page Publier un message sur une rubrique.
3. (Facultatif) Dans la section Détails du message, saisissez l'Objet, par exemple :

   ```
   Hello from Amazon SNS!
   ```
4. Dans la section Corps du message, choisissez Charge utile identique pour tous les protocoles de distribution, puis saisissez un corps de message, tel que :

   ```
   Publishing a message to an SNS topic.
   ```
5. Choisissez Publier le message.

   Le message est publié dans la rubrique et la console ouvre la page Détails de la rubrique.
6. Vérifiez dans votre boîte de réception si vous avez reçu un e-mail d'Amazon SNS contenant le message publié.

Étape 4 : Supprimer l'abonnement et la rubrique

1. Dans le panneau de navigation, choisissez Abonnements.
2. Sur la page Abonnements, choisissez un abonnement confirmé puis Supprimer.

   > **Note**
   >
   > Vous ne pouvez pas supprimer une confirmation en attente. Au bout de 48 heures, Amazon SNS le supprime automatiquement.

3. Dans la boîte de dialogue Supprimer l'abonnement, sélectionnez Supprimer.

   L'abonnement est supprimé.
4. Dans le panneau de navigation, choisissez Rubriques.
5. Sur la page Rubriques, sélectionnez une rubrique puis choisissez Supprimer.

   > **Important**
   >
   > Lorsque vous supprimez une rubrique, vous supprimez également tous les abonnements à cette rubrique.

6. Dans la boîte de *`MyTopic`*dialogue Supprimer le sujet, entrez `delete me` puis choisissez Supprimer.

   La rubrique est supprimée.

Étapes suivantes

Maintenant que vous avez créé une rubrique avec un abonnement et envoyé des messages à la rubrique, testez ce qui suit :

- Explorez le Centre pour développeurs AWS.
- Découvrez comment protéger vos données dans la section Sécurité.
- Activez le chiffrement côté serveur pour une rubrique.
- Activez le chiffrement côté serveur pour une rubrique avec une file d'attente Amazon Simple Queue Service (Amazon SQS) chiffrée abonnée.

- Abonnez AWS Event Fork Pipelines à une rubrique.

Configuration d'Amazon SNS

Utilisez la console Amazon SNS pour créer et configurer des rubriques et des abonnements Amazon SNS. Pour plus d'informations sur Amazon SNS, consultez Qu'est-ce qu'Amazon SNS ?.

Rubriques

- Création d'une rubrique Amazon SNS
- Abonnement à une rubrique Amazon SNS
- Supprimer un sujet et un abonnement Amazon SNS
- Identification des rubriques Amazon SNS

Création d'une rubrique Amazon SNS

Une rubrique Amazon SNS est un point d'accès logique qui joue le rôle de canal de communication. Une rubrique vous permet de regrouper plusieurs points de terminaison (tels qu'Amazon SQS AWS Lambda, HTTP/S ou une adresse e-mail).

Pour diffuser les messages d'un système producteur de messages (par exemple, un site web d'e-commerce) qui utilise plusieurs autres services ayant besoin de ses messages (par exemple, les systèmes d'enregistrement et de distribution de commandes), vous pouvez créer une rubrique pour votre système producteur.

La tâche Amazon SNS la plus importante et la plus courante consiste à créer une rubrique. Cette page montre comment vous pouvez utiliser le AWS Management Console AWS SDK for Java, le et le AWS SDK for .NET pour créer un sujet.

Lors de la création, vous choisissez un type de rubrique (standard ou FIFO) et vous nommez la rubrique. Après avoir créé une rubrique, vous ne pouvez pas modifier son type ou son nom. Tous les autres choix de configuration sont facultatifs lors de la création d'une rubrique et vous pouvez les modifier ultérieurement.

> **Important**
>
> N'ajoutez pas de données d'identification personnelle (PII) ou d'autres données confidentielles ou sensibles dans les noms de rubriques. Les noms des rubriques sont accessibles aux autres Amazon Web Services, y compris aux CloudWatch journaux. Les

noms de rubriques ne sont pas destinées à être utilisées pour des données privées ou sensibles.

Rubriques

- Pour créer un sujet à l'aide du AWS Management Console
- Pour créer une rubrique à l'aide d'un AWS SDK

Pour créer un sujet à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Effectuez l'une des actions suivantes :
   - Si aucun sujet n'a encore été créé sous votre Compte AWS nom, lisez la description d'Amazon SNS sur la page d'accueil.
   - Si des sujets ont déjà été créés sous votre Compte AWS nom, dans le panneau de navigation, sélectionnez Sujets.
3. Sur la page Rubriques, choisissez Créer une rubrique.
4. Sur la page Créer une rubrique, dans la section Détails, procédez comme suit :
   a. Pour Type, choisissez un type de rubrique (Standard ou FIFO).
   b. Entrez un Nom pour la rubrique. Pour une rubrique FIFO, ajoutez .fifo à la fin du nom.
   c. (Facultatif) Entrez un Nom d'affichage pour votre rubrique.

> **Important**
>
> Lorsque vous vous abonnez à un point de terminaison d'e-mail, le nombre de caractères combinés pour le nom d'affichage de la rubrique Amazon SNS et l'adresse e-mail d'envoi (par exemple, no-reply@sns.amazonaws.com) ne doit pas dépasser 320 caractères UTF-8. Vous pouvez utiliser un outil d'encodage tiers pour vérifier la longueur de l'adresse d'envoi avant de configurer un nom d'affichage pour votre rubrique Amazon SNS.

d. (Facultatif) Pour une rubrique FIFO, vous pouvez choisir déduplication des messages basée sur le contenu pour activer la déduplication des messages par défaut. Pour plus d'informations, consultez Déduplication de message pour les rubriques FIFO.

5. (Facultatif) Développez la section Chiffrement et effectuez les opérations suivantes. Pour plus d'informations, consultez Chiffrement au repos.

a. Choisissez Activer le chiffrement.

b. Spécifiez la AWS KMS clé. Pour plus d'informations, consultez Termes clés.

Pour chaque type de KMS, les champs Description, Account (Compte) et KMS ARN (ARN KMS) s'affichent.

> **Important**
>
> Si vous n'êtes pas le propriétaire de la clé KMS, ou si vous vous connectez avec un compte n'ayant pas les autorisations `kms:ListAliases` et `kms:DescribeKey`, vous ne pouvez pas afficher les informations relatives à la KMS sur la console Amazon SNS.
> Demandez au propriétaire de la KMS de vous accorder ces autorisations. Pour plus d'informations, consultez Autorisations d'API AWS KMS : référence des actions et ressources dans le Guide du développeur AWS Key Management Service .

- Le KMS AWS géré pour Amazon SNS (par défaut) alias/aws/sns est sélectionné par défaut.

> **Note**
>
> Gardez à l'esprit les points suivants :
>
> - La première fois que vous utilisez le AWS Management Console pour spécifier le KMS AWS géré pour Amazon SNS pour une rubrique, le KMS AWS géré pour Amazon SNS est AWS KMS créé.
> - Sinon, la première fois que vous utilisez l'`Publish`action sur un sujet pour lequel SSE est activé, le KMS AWS géré est AWS KMS créé pour Amazon SNS.

- Pour utiliser un KMS personnalisé depuis votre AWS compte, choisissez le champ clé KMS, puis choisissez le KMS personnalisé dans la liste.

> **Note**
>
> Pour obtenir des instructions pour la création de KMS personnalisées, consultez la section Création de clés dans le Guide du développeur AWS Key Management Service .

- Pour utiliser un ARN KMS personnalisé à partir de votre AWS compte ou d'un autre AWS compte, saisissez-le dans le champ clé KMS.

6. (Facultatif) Par défaut, seul le propriétaire de la rubrique peut publier dans la rubrique ou s'abonner à la rubrique. Pour configurer des autorisations d'accès supplémentaires, développez la section Politique d'accès. Pour plus d'informations, consultez Gestion des identités et des accès dans Amazon SNS et Cas d'exemple pour le contrôle d'accès Amazon SNS.

> **Note**
>
> Lorsque vous créez une rubrique à l'aide de la console, la politique par défaut utilise la clé de condition `aws:SourceOwner`. Cette clé est similaire à `aws:SourceAccount`.

7. (Facultatif) Pour configurer la façon dont Amazon SNS retente la distribution des messages en échec, développez la section Politique de nouvelle tentative de distribution (HTTP/S). Pour plus d'informations, consultez Nouvelle tentative de distribution des messages Amazon SNS.
8. (Facultatif) Pour configurer la manière dont Amazon SNS enregistre la livraison des messages CloudWatch, développez la section Enregistrement de l'état de livraison. Pour plus d'informations, consultez Statut de distribution de message Amazon SNS.
9. (Facultatif) Pour ajouter des balises de métadonnées à la rubrique, développez la section Balises, saisissez une clé et une valeur (facultatif) et choisissez Ajouter une balise. Pour plus d'informations, consultez Identification des rubriques Amazon SNS.
10. Choisissez Créer une rubrique.

    Le sujet est créé et la ***MyTopic***page s'affiche.

    Le nom, l'ARN, le nom d'affichage (facultatif) et l'ID de AWS compte du propriétaire de la rubrique sont affichés dans la section Détails.
11. Copiez l'ARN de rubrique dans le Presse-papiers, par exemple :

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

Pour créer une rubrique à l'aide d'un AWS SDK

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

Les exemples de code suivants montrent comment utiliser`CreateTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une rubrique avec un nom spécifique.

```
    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;
        }
    }
```

Créez une nouvelle rubrique avec un nom et des attributs FIFO et de déduplication spécifiques.

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

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for .NET API.

C++

Kit SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour créer une rubrique SNS

L'exemple `create-topic` suivant crée une rubrique SNS nommée `my-topic`.

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

Sortie :

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

Pour plus d'informations, consultez la section Utilisation de l'interface de ligne de AWS commande avec Amazon SQS et Amazon SNS dans le Guide de l'utilisateur de AWS l'interface de ligne de commande.

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS CLI commandes.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



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

```
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for Go API.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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 "";
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for JavaScript API.

Kotlin

Kits SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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()
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for PHP API.

Python

Kit SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez CreateTopicle AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for Ruby API.

Rust

Kit SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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(())
}
```

- Pour plus de détails sur l'API, voir CreateTopicla section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section de référence du AWS SDK pour l'API SAP ABAP.

Abonnement à une rubrique Amazon SNS

Pour recevoir les messages publiés dans une rubrique, vous devez abonner un point de terminaison à cette rubrique. Une fois que vous avez abonné un point de terminaison à une rubrique, le point de terminaison commence à recevoir tous les messages publiés dans la rubrique associée.

> **Note**
>
> Les points de terminaison HTTP(S), les adresses e-mail et les ressources AWS dans d'autres Comptes AWS exigent une confirmation de l'abonnement avant de pouvoir recevoir des messages.

Pour abonner un point de terminaison à une rubrique Amazon SNS

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation de gauche, choisissez Subscriptions (Abonnements).
3. Sur la page Abonnements, choisissez Créer un abonnement.
4. Sur la page Créer un abonnement, dans la section Détails, procédez comme suit :

   a. Pour ARN de rubrique, choisissez l'ARN (Amazon Resource Name) d'une rubrique. Cette valeur est l'ARN AWS qui a été généré lors de la création de la rubrique Amazon SNS, par exemple `arn:aws:sns:us-east-2:123456789012:your_topic`.

   b. Pour Protocole, choisissez un type de point de terminaison. Les types de point de terminaison disponibles sont les suivants :

      - HTTP/HTTPS
      - E-mail/E-mail-JSON
      - Amazon Data Firehose
      - Amazon SQS

      > **Note**
      >
      > Pour vous abonner à une Rubrique SNS FIFO, choisissez cette option.

      - AWS Lambda
      - Point de terminaison de l'application de plateforme
      - SMS

   c. Pour Point de terminaison, saisissez la valeur du point de terminaison, telle qu'une adresse e-mail ou l'ARN d'une file d'attente Amazon SQS.

   d. Points de terminaison Firehose uniquement : pour l'ARN du rôle d'abonnement, spécifiez l'ARN du rôle IAM que vous avez créé pour écrire dans les flux de diffusion Firehose. Pour plus d'informations, consultez Conditions préalables à l'abonnement aux flux de diffusion Firehose aux rubriques Amazon SNS.

   e. (Facultatif) Pour les points de terminaison Firehose, Amazon SQS et HTTP/S, vous pouvez également activer la livraison de messages bruts. Pour plus d'informations, consultez Remise des messages bruts Amazon SNS.

f. (Facultatif) Pour configurer une politique de filtre, développez la section Politique de filtre d'abonnement. Pour plus d'informations, consultez Stratégies de filtre d'abonnement Amazon SNS.

g. (Facultatif) Pour activer le filtrage basé sur la charge utile, configurez `Filter Policy Scope` sur `MessageBody`. Pour plus d'informations, consultez Étendue de la politique de filtre d'abonnement Amazon SNS.

h. (Facultatif) Pour configurer une file d'attente de lettres mortes pour l'abonnement, développez la section Politique de reconduite (File d'attente de lettres mortes). Pour de plus amples informations, consultez la section Files d'attente de lettres mortes (DLQ)) d'Amazon SNS.

i. Choisissez Créer un abonnement.

   La console crée l'abonnement et ouvre la page Détails de l'abonnement.

Supprimer un sujet et un abonnement Amazon SNS

Lorsqu'un sujet est supprimé, les abonnements qui lui sont associés sont supprimés de manière asynchrone. Bien que les clients puissent toujours accéder à ces abonnements, ceux-ci ne sont plus associés au sujet, même si vous recréez le sujet sous le même nom.

Si un abonné tente de publier un message dans la rubrique supprimée, le diffuseur de publication reçoit un message d'erreur indiquant que la rubrique n'existe pas. De même, toute tentative d'abonnement à la rubrique supprimée se traduira par un message d'erreur.

Vous ne pouvez pas supprimer un abonnement qui est en attente de confirmation. Amazon SNS supprime automatiquement les abonnements non confirmés au bout de 48 heures.

Rubriques

- Pour supprimer un sujet ou un abonnement Amazon SNS à l'aide du AWS Management Console
- Pour supprimer un abonnement et une rubrique à l'aide d'un kit SDK AWS

Pour supprimer un sujet ou un abonnement Amazon SNS à l'aide du AWS Management Console

Pour supprimer un sujet à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation de gauche, choisissez Rubriques.
3. Sur la page Rubriques, sélectionnez une rubrique puis choisissez Supprimer.
4. Dans la boîte de dialogue Supprimer la rubrique, saisissez `delete me`, puis choisissez Supprimer.

   La console supprime la rubrique.

Pour supprimer un abonnement à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation de gauche, choisissez Subscriptions (Abonnements).
3. Sur la page Abonnements, sélectionnez un abonnement dont le statut est Confirmé, puis choisissez Supprimer.
4. Dans la boîte de dialogue Supprimer l'abonnement, sélectionnez Supprimer.

   La console supprime l'abonnement.

Pour supprimer un abonnement et une rubrique à l'aide d'un kit SDK AWS

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

Les exemples de code suivants montrent comment utiliser`DeleteTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Supprimez une rubrique à l'aide de son ARN de rubrique.

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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour supprimer une rubrique SNS

L'exemple `delete-topic` suivant supprime la rubrique SNS spécifiée.

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

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

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS CLI commandes.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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
  //   }
  // }
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus de détails sur l'API, consultez DeleteTopicla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for PHP API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez DeleteTopicle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section de référence du AWS SDK pour l'API SAP ABAP.

Identification des rubriques Amazon SNS

Amazon SNS prend en charge l'identification des rubriques Amazon SNS. Cela peut vous aider à suivre et à gérer les coûts associés à vos rubriques, à fournir une sécurité renforcée dans vos politiques AWS Identity and Access Management (IAM) et à vous permettre de rechercher ou de filtrer facilement parmi des milliers de rubriques. L'identification vous permet de gérer vos rubriques Amazon SNS à l'aide d'AWS Resource Groups. Pour plus d'informations sur Resource Groups, consultez le Guide de l'utilisateur AWS Resource Groups.

Rubriques

- Identification pour l'allocation des coûts
- Identification pour le contrôle d'accès
- Identification pour la recherche et le filtrage des ressources
- Configuration des identifications de rubrique Amazon SNS

Identification pour l'allocation des coûts

Pour organiser et identifier vos rubriques Amazon SNS en vue de l'allocation des coûts, vous pouvez ajouter des identifications qui identifient l'objectif d'une rubrique. Cette approche est utile lorsque vous avez un grand nombre de rubriques. Vous pouvez utiliser des identifications d'allocation des coûts pour organiser votre facture AWS afin de refléter votre propre structure de coût. Pour ce faire, inscrivez-vous pour que votre facture de compte AWS inclut les clés et valeurs de balise. Pour plus d'informations, consultez Configuration du rapport de l'allocation des coûts mensuel dans le Guide de l'utilisateur AWS Billing and Cost Management.

Par exemple, vous pouvez ajouter des identifications qui représentent le centre de coût et l'objectif de vos rubriques Amazon SNS, comme suit :

Ressource	Clé	Valeur
Rubrique 1	Centre de coûts	43289
	Application	Traitement de commandes
Rubrique 2	Centre de coûts	43289
	Application	Traitement des paiements
Rubrique 3	Centre de coûts	76585
	Application	Archivage

Ce schéma d'identification vous permet de regrouper deux rubriques effectuant des tâches connexes dans le même centre de coûts, tout en identifiant une activité indépendante avec une identification pour l'allocation des coûts différente.

Identification pour le contrôle d'accès

AWS Identity and Access Management prend en charge le contrôle de l'accès aux ressources basé sur des balises. Après avoir attribué des identifications à vos ressources, fournissez des informations sur ces identifications dans l'élément de condition d'une politique IAM pour gérer l'accès basé sur les identifications. Pour plus d'informations sur la manière d'attribuer des identifications à vos ressources à l'aide de la console Amazon SNS ou du kit AWS SDK, consultez Configuration des identifications.

Vous pouvez restreindre l'accès pour une identité IAM. Par exemple, vous pouvez restreindre l'accès `Publish` et `PublishBatch` à toutes les rubriques Amazon SNS qui incluent une identification avec la clé `environment` et la valeur `production`, tout en autorisant l'accès à toutes les autres rubriques Amazon SNS. Dans l'exemple ci-dessous, la politique limite la possibilité de publier des messages aux rubriques identifiées avec `production`, tout en autorisant la publication de messages aux rubriques identifiées avec `development`. Pour plus d'informations, consultez Contrôle de l'accès à l'aide d'identifications dans le Guide de l'utilisateur IAM.

> **Note**
>
> La définition de l'autorisation IAM pour `Publish` définit l'autorisation pour `Publish` et `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"
      }
    }
  }]
}
```

Identification pour la recherche et le filtrage des ressources

Un compte AWS peut avoir des dizaines de milliers de rubriques Amazon SNS (consultez Quotas Amazon SNS pour plus de détails). En attribuant des identifications à vos rubriques, vous pouvez simplifier le processus de recherche ou de filtrage des rubriques.

Par exemple, vous pouvez avoir des centaines de rubriques associées à votre environnement de production. Plutôt que de devoir rechercher manuellement ces rubriques, vous pouvez rechercher toutes les rubriques avec une identification donnée :

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

Configuration des identifications de rubrique Amazon SNS

Cette page explique comment utiliser le AWS Management Console AWS SDK et la AWS CLI pour configurer les balises d'une rubrique Amazon SNS.

> **Important**
>
> N'ajoutez pas de données d'identification personnelle (PII) ou d'autres informations confidentielles ou sensibles dans les étiquettes. Les identifications sont accessibles à d'autres Amazon Web Services, y compris la facturation. Les étiquettes ne sont pas destinées à être utilisées pour des données privées ou sensibles.

Rubriques

- Répertorier, ajouter et supprimer des balises pour une rubrique Amazon SNS à l'aide du AWS Management Console
- Ajout d'identifications à une rubrique à l'aide d'un kit AWS SDK
- Gestion des identifications avec les actions de l'API Amazon SNS
- Actions d'API qui prennent en charge ABAC

Répertorier, ajouter et supprimer des balises pour une rubrique Amazon SNS à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sur la page Rubriques, sélectionnez une rubrique, puis choisissez Modifier.
4. Développez la section identification.

   Les identifications ajoutées à la rubrique sont répertoriées.
5. Modifiez les identifications de la rubrique :
   - Pour ajouter une identification, choisissez Add tag (Ajouter une identification) et saisissez une Key (Clé) et une Value (Valeur) (facultatif).
   - Pour supprimer une balise, choisissez Supprimer une balise en regard d'une paire clé-valeur.
6. Sélectionnez Enregistrer les modifications.

Ajout d'identifications à une rubrique à l'aide d'un kit AWS SDK

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

Les exemples de code suivants montrent comment utiliser`TagResource`.

CLI

AWS CLI

Pour ajouter une balise à une rubrique

L'exemple `tag-resource` suivant ajoute une balise de métadonnées à la rubrique Amazon SNS spécifiée.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS CLI commandes.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS SDK for Java 2.x API.

Kotlin

Kits SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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")
    }
}
```

- Pour plus de détails sur l'API, consultez TagResourcela section AWS SDK pour la référence de l'API Kotlin.

Gestion des identifications avec les actions de l'API Amazon SNS

Pour gérer les identifications à l'aide de l'API Amazon SNS, utilisez les actions d'API suivantes :

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

Actions d'API qui prennent en charge ABAC

Voici une liste d'actions d'API qui prennent en charge le contrôle d'accès basé sur les attributs (ABAC). Pour plus de détails sur l'ABAC, voir À quoi sert l'ABAC ? AWS dans le guide de l'utilisateur IAM.

- `AddPermission`
- `ConfirmSubscription`
- `DeleteTopic`
- `GetDataProtectionPolicy`
- `GetSubscriptionAttributes`
- `GetTopicAttributes`
- `ListSubscriptionsByTopic`
- `ListTagsForResource`
- `Publish`

- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

Classement et déduplication des messages (rubriques FIFO)

Vous pouvez utiliser les rubriques FIFO Amazon SNS avec les files d'attente FIFO Amazon SQS pour fournir un ordre strict des messages et une déduplication des messages. Les fonctionnalités FIFO de chacun de ces services fonctionnent ensemble pour agir en tant que service entièrement géré pour intégrer des applications distribuées nécessitant une cohérence des données en temps quasi réel. L'abonnement des files d'attente standard Amazon SQ aux rubriques FIFO Amazon SNS permet un classement dans les meilleures conditions et une distribution au moins une fois.

Rubriques

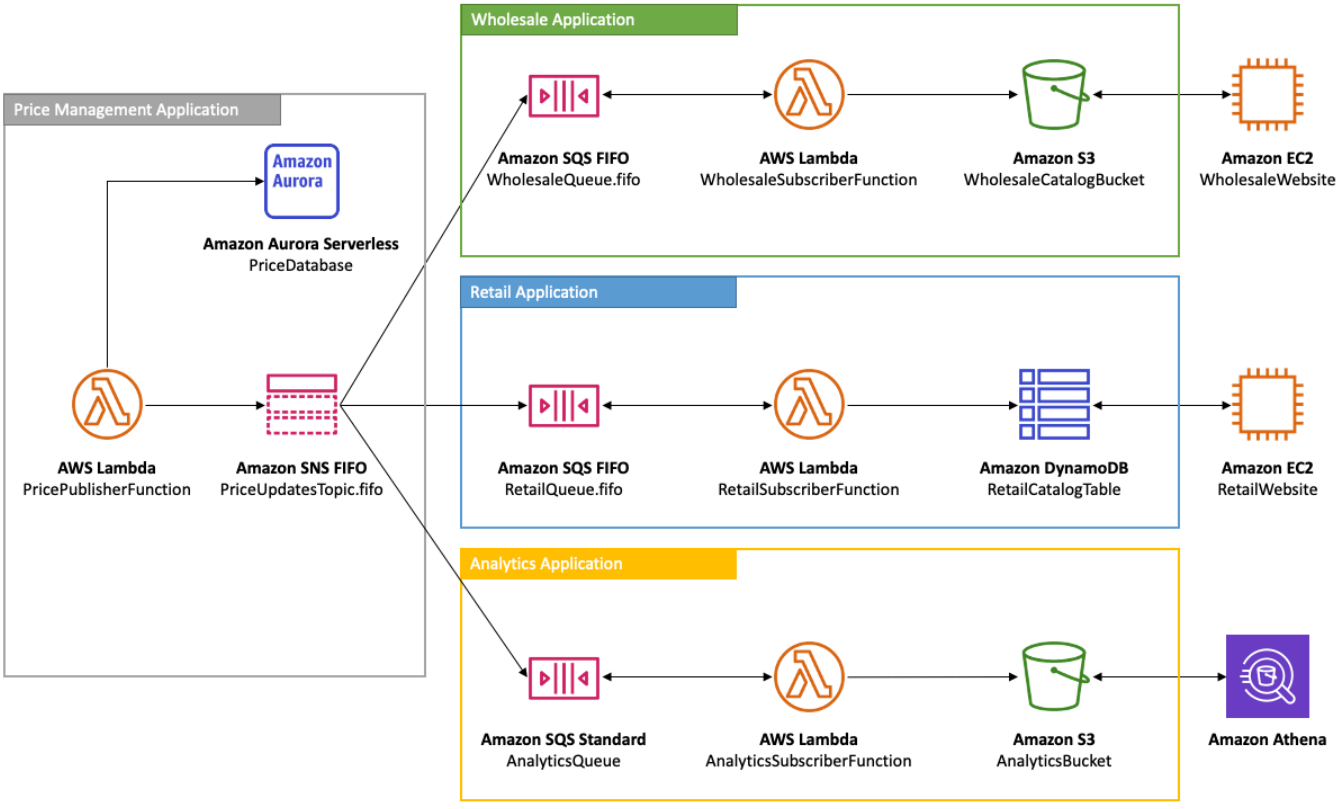
Cas d'utilisation d'exemple de rubrique FIFO

L'exemple suivant décrit une plateforme de e-commerce créée par un fabricant de pièces automobiles à l'aide des rubriques FIFO Amazon SNS et des files d'attente Amazon SQS. La plateforme est constituée de quatre applications sans serveur :

- Les gestionnaires de stock utilisent une application de gestion des prix pour définir le prix de chaque article en stock. Dans cette entreprise, les prix des produits peuvent changer en fonction des fluctuations de change, de la demande du marché et des changements de stratégie de vente. L'application de gestion des prix utilise une fonction AWS Lambda qui publie les mises à jour de prix d'une rubrique FIFO Amazon SNS chaque fois que les prix changent.
- Une application de gros fournit le backend pour un site web où les ateliers de carrosserie et les constructeurs automobiles peuvent acheter les pièces automobiles de l'entreprise en vrac. Pour

obtenir des notifications de changement de prix, l'application de gros abonne sa file d'attente FIFO Amazon SQS à la rubrique FIFO Amazon SNS de l'application de gestion des prix.

- Une application de vente au détail fournit le backend pour un autre site web où les propriétaires de voitures et les amateurs de tuning peuvent acheter des pièces d'auto individuelles pour leurs véhicules. Pour obtenir des notifications de changement de prix, l'application de détail abonne également sa file d'attente FIFO Amazon SQS à la rubrique FIFO Amazon SNS de l'application de gestion des prix.
- Une application d'analyse qui regroupe les mises à jour des prix et les stocke dans un compartiment Amazon S3, permettant à Amazon Athena d'interroger le compartiment à des fins de Business Intelligence (BI). Pour obtenir des notifications de changement de prix, l'application d'analyse abonne sa file d'attente standard Amazon SQS à la rubrique FIFO Amazon SNS de l'application de gestion des prix. Contrairement aux autres applications, l'application d'analyse ne nécessite pas que les mises à jour des prix soient strictement classées.

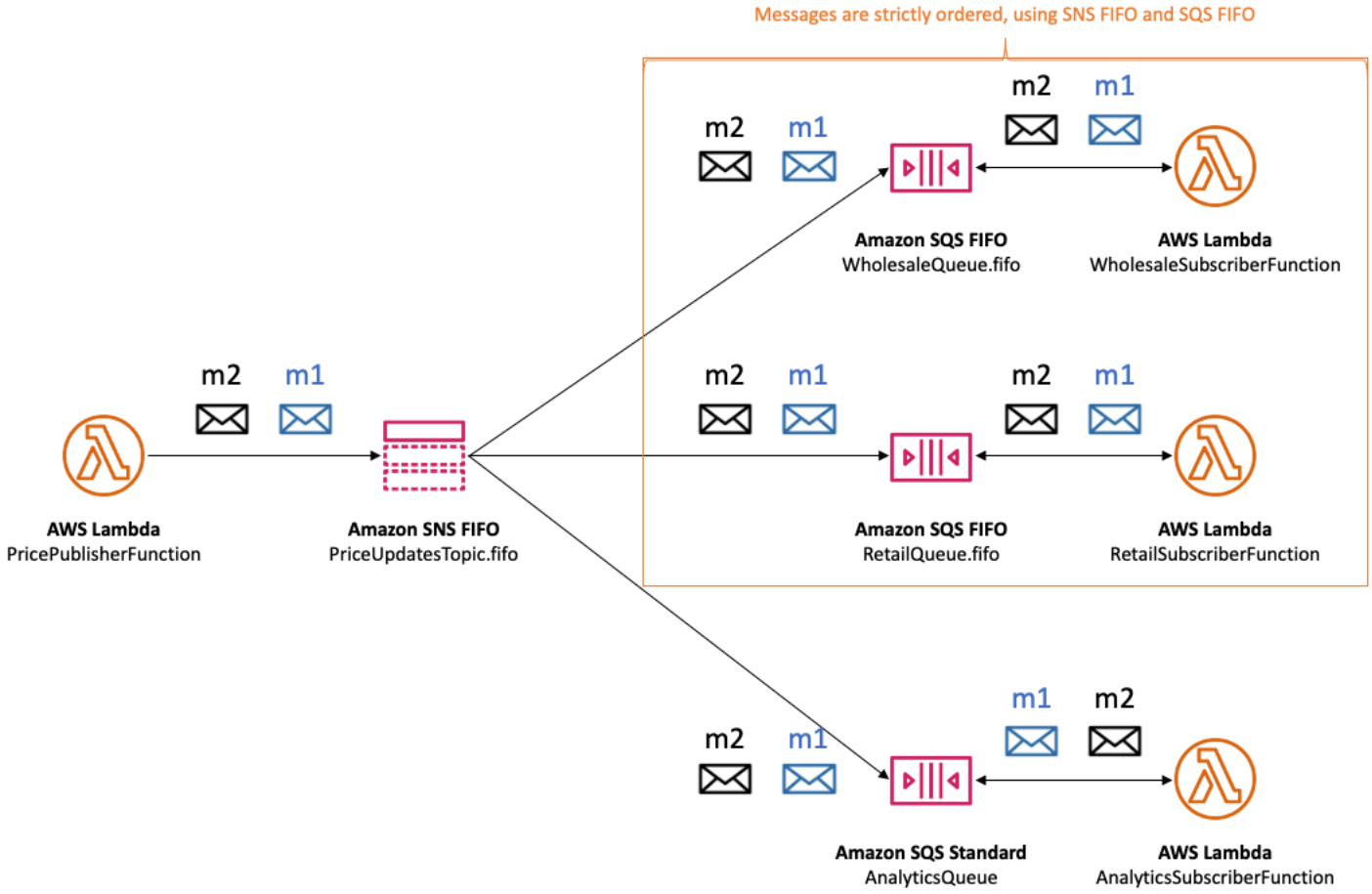


Pour que les applications de gros et de détail reçoivent des mises à jour de prix dans le bon ordre, l'application de gestion des prix doit utiliser un système de distribution de messages strictement ordonnée. L'utilisation des rubriques FIFO Amazon SNS et des files d'attente FIFO Amazon SQS permet de traiter les messages dans l'ordre et sans duplication. Pour de plus amples informations,

veuillez consulter Détails d'ordre des messages pour les rubriques FIFO. Pour des extraits de code qui implémentent ce cas d'utilisation, consultez Exemples de code pour les rubriques FIFO.

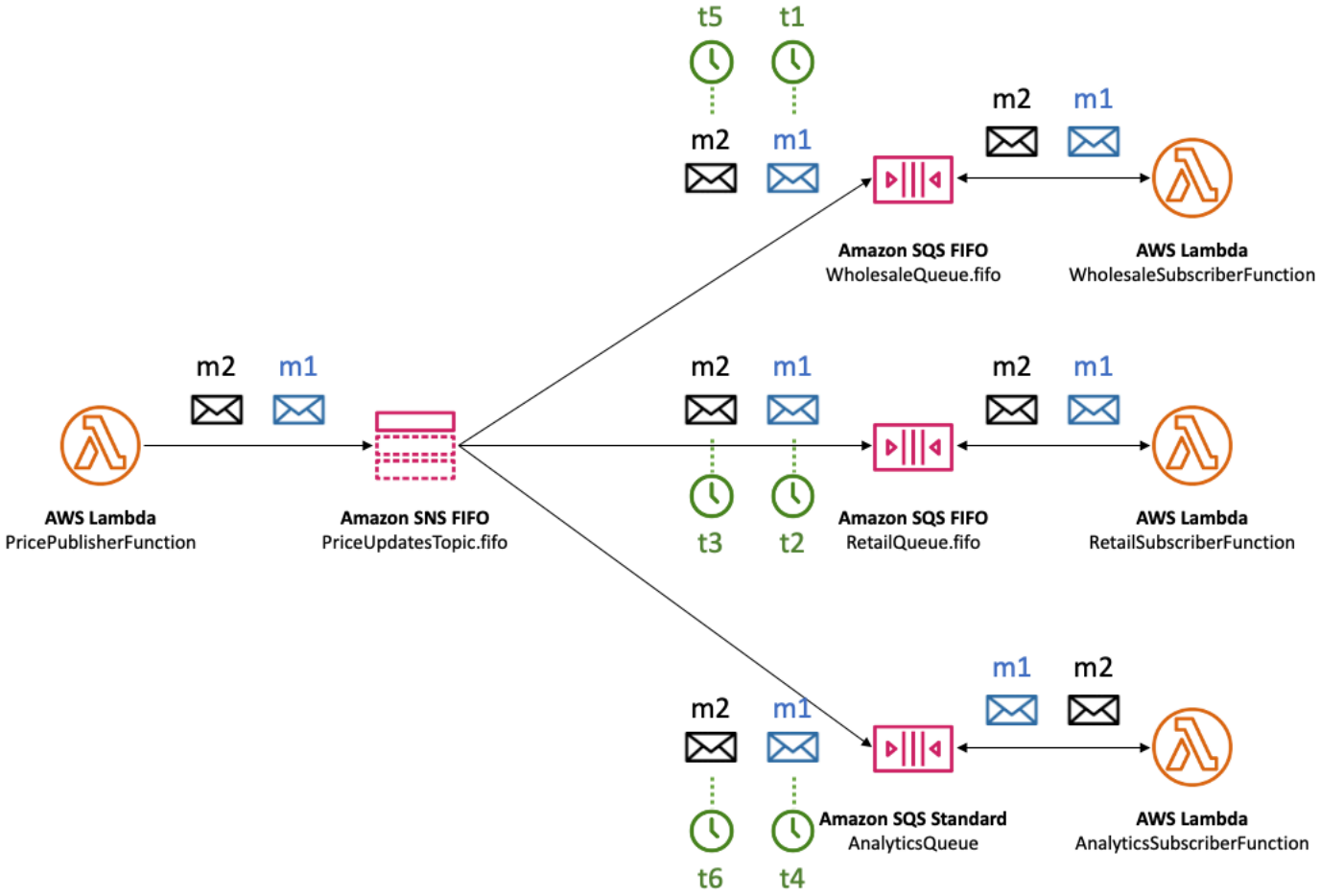
Détails d'ordre des messages pour les rubriques FIFO

Une rubrique FIFO Amazon SNS distribue toujours des messages aux files d'attente Amazon SQS abonnées dans l'ordre exact dans lequel les messages sont publiés dans la rubrique et une seule fois. Avec une file d'attente FIFO Amazon SQS abonnée, le consommateur de la file d'attente reçoit les messages dans l'ordre exact dans lequel ils sont envoyés à la file d'attente, sans doublons. Avec une file d'attente standard SQS abonnée, le consommateur de la file d'attente peut recevoir des messages dans le désordre et ce, plusieurs fois. Cela permet de mieux dissocier les abonnés des diffuseurs de publication, offrant ainsi aux abonnés une plus grande flexibilité en termes de consommation de messages et d'optimisation des coûts, comme le montre le schéma suivant basé sur le Cas d'utilisation d'exemple de rubrique FIFO.

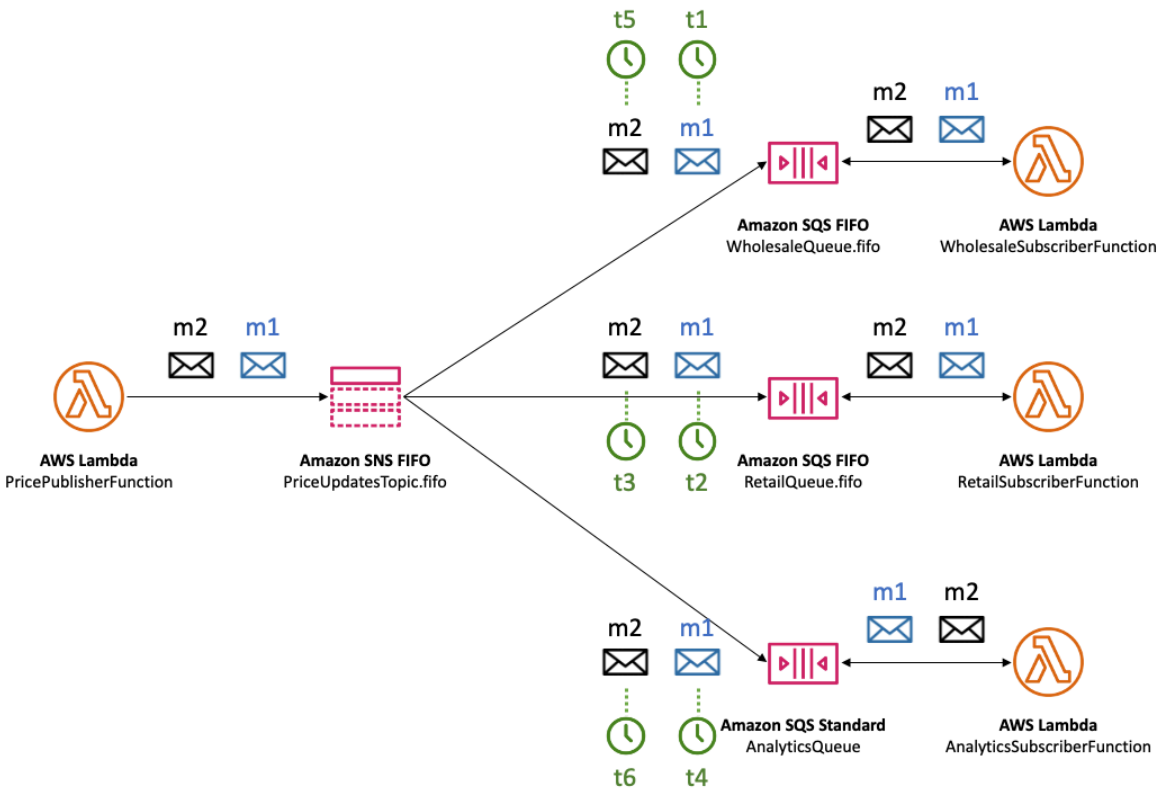


Notez qu'il n'y a pas d'ordre implicite des abonnés. L'exemple suivant montre que le message m1 est d'abord distribué à l'abonné de gros, puis à l'abonné de détail, puis à l'abonné d'analyse. Le

message m2 est d'abord distribué à l'abonné de détail, puis à l'abonné de gros et enfin à l'abonné d'analyse. Bien que les deux messages soient distribué aux abonnés dans un ordre différent, l'ordre des messages est conservé pour chaque abonné FIFO Amazon SQS. Chaque abonné est perçu indépendamment de tout autre abonné.

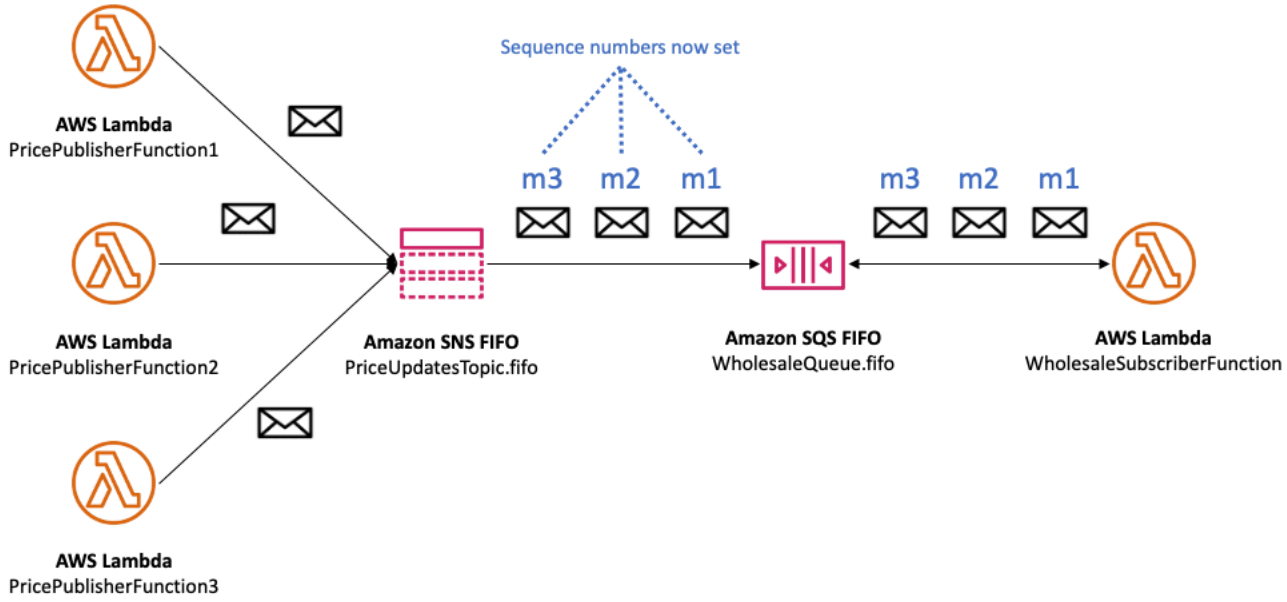


Si un abonné de file d'attente Amazon SQS devient inaccessible, il peut se désynchroniser. Par exemple, supposons que le propriétaire de la file d'attente d'application de gros modifie par erreur la politique de file d'attente Amazon SQS d'une manière qui empêche le principal de service Amazon SNS de distribuer des messages à la file d'attente. Dans ce cas, les envois des mises à jour des prix à la file d'attente de gros échouent, tandis que les envois aux files d'attente de détail et d'analyse réussissent, ce qui entraîne une désynchronisation des abonnés. Lorsque le propriétaire de la file d'attente de l'application de gros corrige sa politique de file d'attente, Amazon SNS reprend la distribution des messages à la file d'attente abonnée. Tous les messages publiés dans la rubrique qui ciblent la file d'attente qui n'était pas correctement configurée sont supprimés, sauf si l'abonnement correspondant dispose d'une file d'attente de lettres mortes configurée.



Vous pouvez avoir plusieurs applications (ou plusieurs threads au sein d'une même application) qui publient des messages vers une rubrique FIFO SNS en parallèle. Lorsque vous effectuez cette opération, vous déléguez effectivement le séquençage des messages au service Amazon SNS. Pour déterminer la séquence de messages établie, vous pouvez vérifier le numéro de séquence.

Le numéro de séquence est un grand numéro, non consécutif, qu'Amazon SNS attribue à chaque message. La longueur du numéro de séquence est de 128 bits et continue d'augmenter pour chaque groupe de messages. Le numéro de séquence est transmis aux files d'attente Amazon SQS abonnées dans le corps du message. Toutefois, si vous activez la distribution de messages bruts, le message qui est distribué à la file d'attente Amazon SQS n'inclut pas le numéro de séquence ou d'autres métadonnées de message Amazon SNS.



Les rubriques FIFO d'Amazon SNS définissent l'ordre dans le contexte d'un groupe de messages. Pour de plus amples informations, consultez Regroupement de messages pour les rubriques FIFO.

Regroupement de messages pour les rubriques FIFO

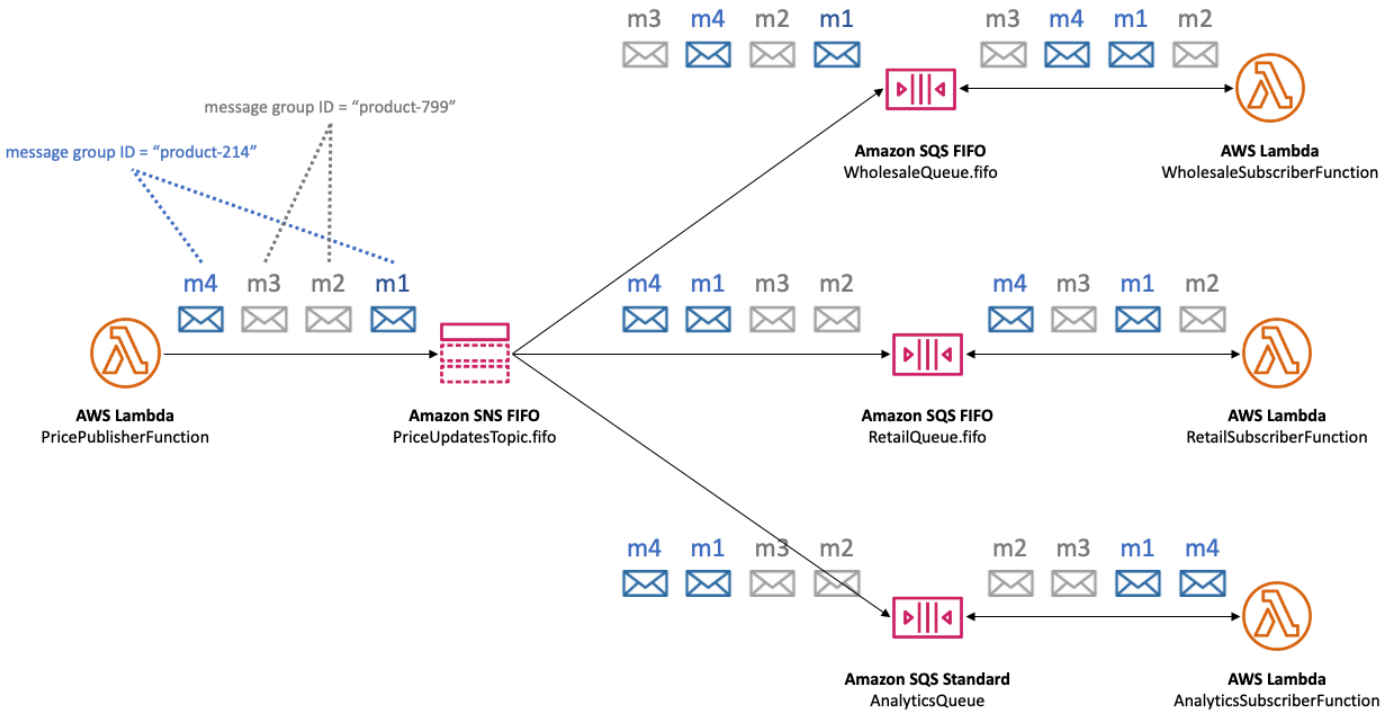
Les messages appartenant au même groupe sont traités un à la fois, dans un ordre strict par rapport au groupe.

Lorsque vous publiez des messages sur une rubrique FIFO Amazon SNS, vous définissez l'ID de groupe de messages. L'ID de groupe est un jeton obligatoire qui spécifie qu'un message appartient à un groupe de messages spécifique. La rubrique FIFO SNS transmet l'ID de groupe aux files d'attente Amazon SQS FIFO abonnées. Il n'y a pas de limite au nombre d'ID de groupe dans les rubriques FIFO SNS ou les files d'attente SQS FIFO. L'ID du groupe de messages n'est pas transmis aux files d'attente standard Amazon SQS.

Il n'y a pas d'affinité entre un groupe de messages et un abonnement. Par conséquent, les messages publiés dans n'importe quel groupe de messages sont remis à toutes les files d'attente abonnées, sous réserve de toute politique de filtrage associée aux abonnements. Pour plus d'informations, consultez Distribution des messages pour les rubriques FIFO et Filtrage des messages pour les rubriques FIFO.

Dans le cas d'utilisation de la gestion des prix des pièces automobiles, il existe un groupe de messages dédié pour chaque produit vendu sur la plateforme. La même rubrique FIFO Amazon SNS est utilisée pour traiter toutes les mises à jour de prix. La séquence des mises à jour des prix est

conservée dans le contexte d'un seul produit de pièces automobiles, mais pas sur plusieurs produits. Le schéma suivant illustre leur fonctionnement. Notez que, pour le produit dont l'ID de groupe de messages est product-214, le message m1 est traité avant le message m4. Cette séquence est conservée tout au long des flux de travail qui utilisent Amazon SNS FIFO et Amazon SQS FIFO. De même, pour le produit dont l'ID de groupe de messages est product-799, le message m2 est traité avant le message m3, à condition que les flux de travail utilisent Amazon SNS FIFO et Amazon SQS FIFO. Toutefois, lorsque vous utilisez des files d'attente standard Amazon SQS, l'ordre des messages n'est plus garanti et les groupes de messages n'existent pas. Les groupes de messages product-214 et le product-799 sont indépendants l'un de l'autre. Il n'y a donc pas de relation entre la façon dont leurs messages sont séquencés.



Distribution des données par ID de groupe de messages pour améliorer les performances

Pour optimiser le débit de diffusion, les rubriques FIFO Amazon SNS diffusent les messages provenant de différents groupes de messages en parallèle, tandis que l'ordre des messages est strictement maintenu au sein de chaque groupe de messages. Chaque groupe de messages peut délivrer un maximum de 300 messages par seconde. Par conséquent, pour obtenir un débit élevé pour un seul sujet, utilisez un grand nombre d'ID de groupes de messages distincts. Lors de

l'utilisation d'un ensemble diversifié de groupes de messages, les rubriques FIFO Amazon SNS distribuent automatiquement les messages sur un plus grand nombre de partitions parallèles.

> **Note**
>
> Les rubriques FIFO Amazon SNS sont optimisées pour une distribution uniforme des messages entre les ID de groupes de messages, quel que soit le nombre de groupes. AWS recommande d'utiliser un grand nombre d'ID de groupes de messages distincts pour optimiser les performances.

Lorsque vous publiez sur votre rubrique FIFO Amazon SNS à débit élevé et qu'une ou plusieurs files d'attente FIFO Amazon SQS sont abonnées, il est recommandé d'activer le débit élevé sur vos files d'attente. Pour en savoir plus, consultez la section Débit élevé pour les files d'attente FIFO dans le Guide du développeur Amazon Simple Queue Service.

Distribution des messages pour les rubriques FIFO

Les rubriques FIFO Amazon SNS prennent en charge la distribution aux files d'attente standard et FIFO Amazon SQS afin d'offrir aux clients la flexibilité et le contrôle nécessaires lors de l'intégration d'applications distribuées exigeant une cohérence des données en temps quasi réel.

Pour les charges de travail qui doivent respecter un ordre strict des messages ou la déduplication, la combinaison des rubriques FIFO Amazon SNS avec les files d'attente FIFO Amazon SQS abonnées en tant que point de terminaison de distribution permet d'améliorer la messagerie entre les applications lorsque l'ordre des opérations et des événements est critique, ou lorsque les doublons ne sont pas tolérés.

Pour les charges de travail qui tolèrent le classement au mieux et la distribution au moins une fois, l'abonnement des files d'attente standard Amazon SQS aux rubriques FIFO Amazon SNS permet de réduire les coûts, en plus de partager les files d'attente entre les charges de travail qui n'utilisent pas FIFO.

> **Note**
>
> Pour une distribution ramifiée des messages des rubriques FIFO Amazon SNS vers des fonctions AWS Lambda, des étapes supplémentaires sont nécessaires. Tout d'abord, abonnez les files d'attente standard ou FIFO Amazon SQS à la rubrique. Ensuite, configurez

les files d'attente pour déclencher les fonctions. Pour plus d'informations, consultez SQS FIFO en tant que source d'évènements sur le Blog AWS Compute.

Les rubriques FIFO SNS ne peuvent pas transmettre de messages aux points de terminaison gérés par le client, tels que les adresses e-mail, les applications mobiles, les numéros de téléphone pour la messagerie texte (SMS) ou les points de terminaison HTTP(S). Ces types de point de terminaison ne garantissent pas un ordre strict des messages. Les tentatives d'abonner des points de terminaison gérés par le client aux rubriques FIFO SNS entraînent des erreurs.

Les rubriques FIFO SNS prennent en charge les mêmes fonctionnalités de filtrage des messages que les rubriques standard. Pour plus d'informations, consultez Filtrage des messages pour les rubriques FIFO et la publication Simplifier votre messagerie Pub/Sub avec le filtrage des messages d'Amazon SNS sur le Blog AWS Compute.

Filtrage des messages pour les rubriques FIFO

Les rubriques FIFO Amazon SNS prennent en charge le filtrage des messages. L'utilisation du filtrage des messages simplifie votre architecture en déchargeant la logique d'acheminement des messages depuis vos systèmes éditeurs et la logique de filtrage des messages depuis vos systèmes abonnés.

Lorsque vous abonnez une file d'attente standard ou FIFO Amazon SQS à une rubrique FIFO SNS, vous pouvez utiliser le filtrage des messages pour spécifier que l'abonné reçoit un sous-ensemble de messages plutôt que tous les messages. Chaque abonné peut définir sa propre politique de filtre en tant qu'attributs d'abonnement. En fonction de l'étendue de la politique de filtre, la politique de filtre est mise en correspondance avec les attributs de message ou le corps de message entrant. En cas de correspondance de la politique de filtre, la rubrique envoie une copie du message à l'abonné. S'il n'y a pas de correspondance, la rubrique n'envoie pas de copie du message.

Dans le cas d'utilisation de la gestion des prix des pièces automobiles, supposons que les politiques de filtre Amazon SNS suivantes sont définies et que l'étendue de la politique de filtre est `MessageBody` :

- Pour la file d'attente de gros, la politique de filtre `{"business":["wholesale"]}` correspond à chaque message contenant une clé nommée `business` et à `wholesale` dans l'ensemble de valeurs. Dans le diagramme suivant, l'une des clés dans le message m1 est `business` avec la valeur `wholesale`. L'une des clés dans le message m3 est `business` avec la valeur

`["wholesale,retail"]`. Ainsi, les deux messages m1 et m3 correspondent aux critères de la politique de filtrage, et les deux messages sont remis à la file d'attente de gros.

- Pour la file d'attente de détail, la politique de filtre `{"business":["retail"]}` correspond à chaque message contenant une clé nommée `business` et à `retail` dans l'ensemble de valeurs. Dans le diagramme, l'une des clés dans le message m2 est `business` avec la valeur `retail`. L'une des clés dans le message m3 est `business` avec la valeur `["wholesale,retail"]`. Ainsi, les deux messages m2 et m3 correspondent aux critères de la politique de filtrage, et les deux messages sont remis à la file d'attente de détail.
- Pour la file d'attente d'analyse, nous voulons qu'Amazon Athena reçoive tous les enregistrements. Aucune politique de filtrage n'est donc appliquée.

Les rubriques FIFO SNS prennent en charge une variété d'opérateurs de correspondance, dont les valeurs de chaîne d'attribut, les valeurs numériques d'attribut et les clés d'attribut. Pour de plus amples informations, veuillez consulter Filtrage des messages Amazon SNS.

Les rubriques FIFO SNS ne fournissent pas de messages en double aux points de terminaison abonnés. Pour de plus amples informations, veuillez consulter Déduplication de message pour les rubriques FIFO.

Déduplication de message pour les rubriques FIFO

Les rubriques FIFO d'Amazon SNS et les files d'attente FIFO d'Amazon SQS prennent en charge la déduplication des messages, qui fournit une distribution et un traitement des messages une fois précisément dès lors que les conditions suivantes sont remplies :

- La file d'attente FIFO Amazon SQS abonnée existe et dispose des autorisations qui permettent au principal de service Amazon SNS de remettre des messages à la file d'attente.
- Le consommateur de file d'attente FIFO Amazon SQS traite le message et le supprime de la file d'attente avant l'expiration du délai d'expiration de la visibilité.
- La rubrique d'abonnement Amazon SNS n'a pas de filtrage de messages. Lorsque vous configurez le filtrage des messages, les rubriques FIFO d'Amazon SNS prennent en charge la at-most-once diffusion, car les messages peuvent être filtrés en fonction de vos politiques de filtrage d'abonnement.
- Il n'y a aucune interruption du réseau qui empêche l'accusé de réception de la remise du message.

> **Note**
>
> La déduplication des messages s'applique à l'ensemble d'une rubrique FIFO Amazon SNS et non à un groupe de messages individuel.

Lorsque vous publiez un message dans une rubrique FIFO Amazon SNS, le message doit inclure un ID de déduplication. Cet ID est inclus dans le message que la rubrique FIFO Amazon SNS remet aux files d'attente FIFO Amazon SQS abonnées.

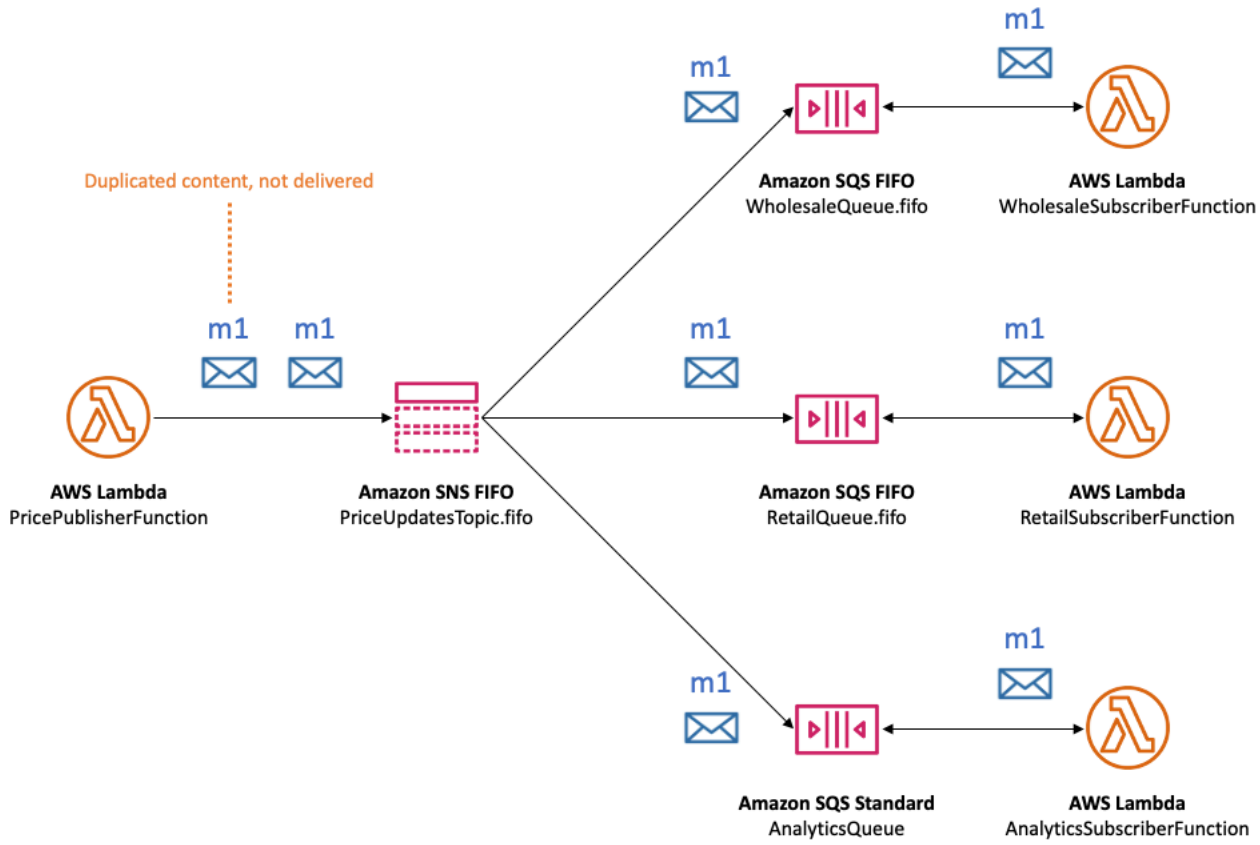
Si un message avec un ID de déduplication particulier est publié avec succès dans une rubrique FIFO Amazon SNS, tout message publié avec le même ID de déduplication, dans l'intervalle de déduplication de cinq minutes, est accepté mais pas remis. La rubrique FIFO Amazon SNS continue de suivre l'ID de déduplication du message, même après la remise du message aux points de terminaison abonnés.

Si le corps du message est garanti pour être unique pour chaque message publié, vous pouvez activer la déduplication basée sur le contenu pour une rubrique FIFO Amazon SNS et les files d'attente FIFO Amazon SQS abonnées. Amazon SNS utilise le corps du message pour générer une valeur de hachage unique à utiliser comme ID de déduplication pour chaque message. Vous n'avez donc pas besoin de définir un ID de déduplication lorsque vous envoyez chaque message.

> **Note**
>
> Les attributs de message ne sont pas inclus dans le calcul de hachage.

Lorsque la déduplication basée sur le contenu est activée pour une rubrique Amazon SNS FIFO et qu'un message est publié avec un ID de déduplication, l'ID de déduplication publié remplace l'ID de déduplication basé sur le contenu généré.

Dans le cas d'utilisation de la gestion des prix des pièces automobiles, l'entreprise doit définir un ID de déduplication universellement unique pour chaque mise à jour de prix. Cela est dû au fait que le corps du message peut être identique même lorsque l'attribut du message est différent pour le commerce de gros et de détail. Toutefois, si la société ajoutait le type d'entreprise (gros ou détail) au corps du message à côté de l'ID du produit et du prix du produit, elle pourrait permettre la déduplication basée sur le contenu dans la rubrique FIFO Amazon SNS et dans les files d'attente FIFO Amazon SQS souscrites.



Outre le classement et la déduplication des messages, les rubriques Amazon SNS FIFO concernent le chiffrement côté serveur des messages (SSE) à l'aide de clés et la confidentialité des messages via les points de terminaison VPC AWS KMS avec. AWS PrivateLink Pour plus d'informations, voir Sécurité des messages pour les rubriques FIFO.

Sécurité des messages pour les rubriques FIFO

Vous pouvez choisir de chiffrer les messages envoyés aux rubriques et files d'attente FIFO par Amazon SNS et Amazon SQS à l'aide des clés principales client (CMK) AWS Key Management Service (AWS KMS). Vous pouvez créer des rubriques et des files d'attente FIFO chiffrées, ou choisir de chiffrer des rubriques et des files d'attente FIFO existantes. Amazon SNS et Amazon SQS chiffrent uniquement le corps du message. Ils ne chiffrent pas les attributs de message, les métadonnées de ressource ou les métriques de ressource.

> **Note**
>
> L'ajout du chiffrement à une rubrique ou une file d'attente FIFO existante ne chiffre pas les messages en attente, et la suppression du chiffrement d'une rubrique ou d'une file d'attente laisse les messages en attente chiffrés.

Les rubriques FIFO SNS déchiffrent les messages immédiatement avant de les remettre aux points de terminaison abonnés. Les files d'attente FIFO SQS déchiffrent le message juste avant de le renvoyer à l'application consommatrice. Pour plus d'informations, consultez Chiffrement des données et la publication Chiffrement des messages publiés vers Amazon SNS avec AWS KMS sur le blog AWS Compute.

En outre, les rubriques FIFO SNS et les files d'attente FIFO SQS prennent en charge la confidentialité des messages avec les points de terminaison de VPC d'interface fournie par AWS PrivateLink. À l'aide des points de terminaison d'interface, vous pouvez envoyer des messages à partir de sous-réseaux Amazon Virtual Private Cloud (Amazon VPC) vers des rubriques et des files d'attente FIFO sans traverser l'Internet public. Ce modèle conserve votre messagerie à l'intérieur de l'infrastructure et du réseau AWS, ce qui améliore la sécurité globale de votre application. Lorsque vous utilisez AWS PrivateLink, vous n'avez pas besoin de configurer une passerelle Internet, une NAT (Network Address Translation) ou un réseau privé virtuel (VPN). Pour plus d'informations, consultez Confidentialité du trafic inter-réseau et la publication Sécurisation des messages publiés vers Amazon SNS avec AWS PrivateLink sur le blog AWS Security.

Les rubriques FIFO SNS prennent également en charge les files d'attente de lettres mortes et le stockage de messages dans les zones de disponibilité. Pour plus d'informations, veuillez consulter Durabilité des messages pour les rubriques FIFO.

Durabilité des messages pour les rubriques FIFO

Les rubriques FIFO Amazon SNS et les files d'attente Amazon SQS sont durables. Les deux types de ressources stockent les messages de manière redondante dans plusieurs zones de disponibilité et fournissent des files d'attente de lettres mortes pour traiter les cas exceptionnels.

Dans Amazon SNS, la livraison des messages échoue lorsque la rubrique Amazon SNS ne peut pas accéder à une file d'attente Amazon SQS abonnée en raison d'une erreur côté client ou côté serveur :

- Des erreurs côté client se produisent lorsque la rubrique FIFO Amazon SNS contient des métadonnées d'abonnement obsolètes. Deux exemples courants d'erreurs côté client sont lorsque le propriétaire de la file d'attente Amazon SQS effectue l'une des opérations suivantes :
  - Supprime la file d'attente.
  - Modifie la stratégie de file d'attente de manière à empêcher le principal de service Amazon SNS de lui envoyer des messages.

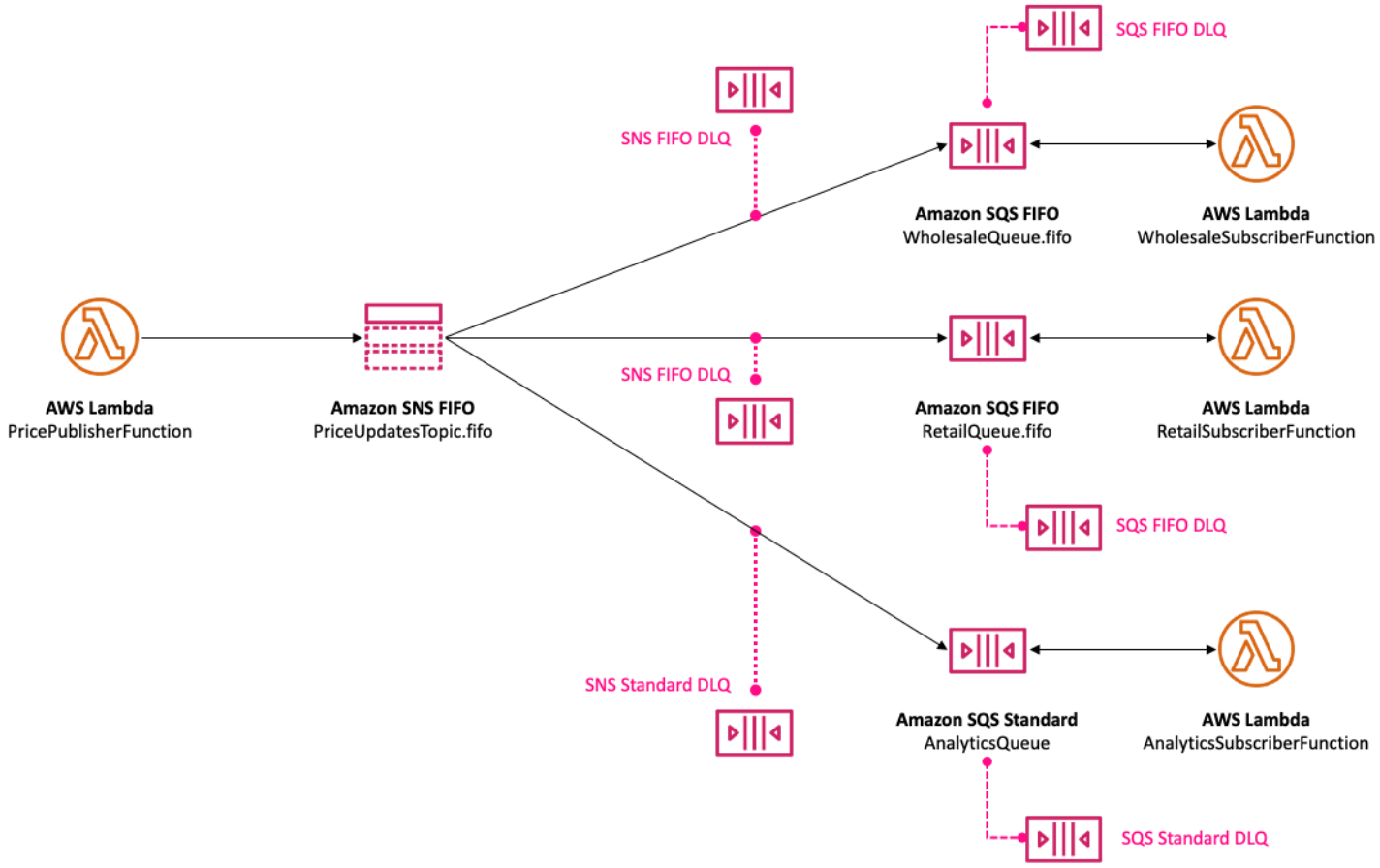
  Amazon SNS ne réessaie pas de remettre les messages qui ont échoué en raison d'erreurs côté client.
- Des erreurs côté serveur peuvent se produire dans les situations suivantes :
  - Le service Amazon SQS n'est pas disponible.
  - Amazon SQS ne parvient pas à traiter une demande valide du service Amazon SNS.

  Lorsque des erreurs côté serveur se produisent, les rubriques FIFO Amazon SNS relancent les livraisons échouées jusqu'à 100 015 fois sur 23 jours. Pour de plus amples informations, veuillez consulter Nouvelle tentative de distribution des messages Amazon SNS.

Pour tout type d'erreur, Amazon SNS peut mettre les messages de côté vers les files d'attente de lettres mortes Amazon SQS afin que les données ne soient pas perdues.

Dans Amazon SQS, le traitement des messages échoue lorsque l'application grand public ne parvient pas à recevoir le message, à le traiter et à le supprimer de la file d'attente. Lorsque le nombre maximal de demandes de réception échoue, Amazon SQS peut mettre les messages de côté en file d'attente de lettres mortes afin que les données ne soient pas perdues.

Dans le cas d'utilisation de la gestion des prix des pièces automobiles, la société peut attribuer une file d'attente de lettres mortes Amazon SQS (DLQ) à chaque abonnement à la rubrique FIFO Amazon SNS, ainsi qu'à chaque file d'attente Amazon SQS abonnée. Cela protège l'entreprise contre toute perte de mise à jour de prix.



La file d'attente de lettres mortes associée à un abonnement Amazon SNS doit être une file d'attente Amazon SQS du même type que la file d'attente d'abonnement. Par exemple, l'abonnement FIFO Amazon SNS pour une file d'attente FIFO Amazon SQS doit posséder une file d'attente FIFO Amazon SQS en tant que file d'attente de lettres mortes. De même, l'abonnement FIFO Amazon SNS pour une file d'attente standard Amazon SQS doit posséder une file d'attente standard Amazon SQS en tant que file d'attente de lettres mortes. Pour plus d'informations, consultez Files d'attente de lettres mortes (DLQ)) d'Amazon SNS et la publication Conception d'applications sans serveur durables avec DLQ pour Amazon SNS, Amazon SQS, AWS Lambda sur le blog AWS Compute.

Pour une durabilité supplémentaire et ainsi faciliter la récupération suite à des défaillances en aval, les propriétaires de rubrique peuvent également utiliser des rubriques FIFO pour bénéficier d'une durée maximale d'archivage des messages de 365 jours. Les abonnés à une rubrique peuvent alors relire ces messages sur un point de terminaison abonné et récupérer les messages perdus suite à

la défaillance d'une application en aval ou répliquer l'état d'une application existante. Pour en savoir plus, consultez Archivage-relecture des messages pour les rubriques FIFO.

Archivage-relecture des messages pour les rubriques FIFO

Rubriques

Qu'est-ce que l'archivage-relecture des messages ?

L'archivage-relecture des messages Amazon SNS est une archive de messages sur place et sans code qui permet aux propriétaires de rubrique de stocker (ou archiver) les messages dans leur rubrique. Les abonnés à une rubrique peuvent ensuite récupérer (ou relire) les messages archivés sur un point de terminaison abonné, ce qui permet de :

- Récupérer les messages qui ont pu se perdre suite à la défaillance d'une application en aval.
- Répliquer l'état d'une application existante dans une nouvelle application en abonnant le nouveau point de terminaison et en sélectionnant l'horodatage à partir duquel la réplication doit démarrer.

Vous pouvez utiliser l'archivage-relecture des messages avec l'API AWS, le kit SDK, AWS CloudFormation et la AWS Management Console.

> **Note**
>
> La fonctionnalité d'archivage-relecture des messages d'Amazon SNS est disponible uniquement pour les rubriques FIFO d'application à application (A2A).

L'archivage-relecture des messages se compose de deux éléments principaux :

1. Archivage des messages : le propriétaire de rubrique active la fonctionnalité d'archivage-relecture au niveau de la rubrique et définit une période de conservation des messages (jusqu'à 365 jours). Le propriétaire de rubrique peut également surveiller les messages archivés à l'aide de

métriques Amazon CloudWatch. Pour en savoir plus, consultez Archivage des messages pour les propriétaires de rubrique FIFO.

2. Rediffusion des messages : l'abonné à la rubrique lance la relecture pour un ensemble de messages de la rubrique sur son point de terminaison abonné. Pour en savoir plus, consultez Relecture des messages pour les abonnés à une rubrique FIFO.

Archivage des messages pour les propriétaires de rubrique FIFO

L'archivage des messages permet d'archiver une copie unique de tous les messages publiés dans votre rubrique. Vous pouvez stocker les messages publiés dans votre rubrique en activant la politique d'archivage des messages au niveau de la rubrique. L'archivage des messages est alors activé pour tous les abonnements liés à cette rubrique. Les messages peuvent être archivés au minimum un jour et au maximum 365 jours.

La définition d'une politique d'archivage entraîne des frais supplémentaires. Pour en savoir plus sur la tarification, consultez Tarification Amazon SNS.

Rubriques

- Création d'une politique d'archivage des messages à l'aide de la AWS Management Console
- Création d'une politique d'archivage des messages à l'aide de l'API
- Création d'une politique d'archivage des messages à l'aide du SDK
- Création d'une politique d'archivage des messages à l'aide d'AWS CloudFormation
- Autorisation d'accès à une archive chiffrée
- Surveillance des métriques d'archivage des messages avec Amazon CloudWatch

Création d'une politique d'archivage des messages à l'aide de la AWS Management Console

Cette option vous permet de créer une politique d'archivage des messages à l'aide de la AWS Management Console.

1. Connectez-vous à la console Amazon SNS.
2. Choisissez une rubrique ou créez-en une. Pour en savoir plus sur la création de rubriques, consultez Création d'une rubrique Amazon SNS.

> **Note**
>
> La fonctionnalité d'archivage-relecture des messages d'Amazon SNS est disponible uniquement pour les rubriques FIFO d'application à application (A2A).

3. Sur la page Modifier la rubrique, développez la section Politique d'archivage.
4. Activez la fonctionnalité Politique d'archivage, puis saisissez le nombre de jours durant lesquels les messages de la rubrique doivent être archivés.
5. Choisissez Enregistrer les modifications.

Pour afficher, modifier et désactiver une politique de rubrique relative à l'archivage des messages

- Sur la page Détails de la rubrique, Politique de conservation indique le statut de la politique d'archivage, y compris le nombre de jours pour lequel elle est définie. Sélectionnez l'onglet Politique d'archivage pour afficher les détails d'archivage des messages suivants :
  - Statut : le statut d'archivage-relecture indique actif lorsqu'une politique d'archivage est appliquée. Le statut d'archivage-relecture indique inactif lorsque la politique d'archivage est définie sur un objet JSON vide.
  - Période de conservation des messages : nombre de jours spécifié pour la conservation des messages.
  - Date de début de l'archivage : date à partir de laquelle les abonnés peuvent relire les messages.
  - Aperçu JSON : aperçu JSON de la politique d'archivage.
- (Facultatif) Pour modifier une politique d'archivage, accédez à la page récapitulative de la rubrique et sélectionnez Modifier.
- (Facultatif) Pour désactiver une politique d'archivage, accédez à la page récapitulative de la rubrique et sélectionnez Modifier. Désactivez la Politique d'archivage et sélectionnez Enregistrer les modifications.
- (Facultatif) Pour supprimer une rubrique assortie d'une politique d'archivage, vous devez d'abord désactiver la politique d'archivage comme décrit précédemment.

> **Important**
>
> Pour éviter les suppressions accidentelles de messages, vous ne pouvez pas supprimer une rubrique assortie d'une politique d'archivage des messages active. La politique d'archivage des messages de la rubrique doit être désactivée avant de pouvoir supprimer

la rubrique. Lorsque vous désactivez une politique d'archivage des messages, Amazon SNS supprime tous les messages archivés. Lorsque vous supprimez une rubrique, les abonnements sont supprimés et la livraison des messages en transit risque d'échouer.

Création d'une politique d'archivage des messages à l'aide de l'API

Pour créer une politique d'archivage des messages à l'aide de l'API, vous devez ajouter l'attribut `ArchivePolicy` à votre rubrique. Vous pouvez définir un attribut `ArchivePolicy` à l'aide des actions d'API `CreateTopic` et `SetTopicAttributes`. `ArchivePolicy` présente une seule valeur, `MessageRetentionPeriod`, qui représente le nombre de jours durant lesquels Amazon SNS conserve les messages. Pour activer l'archivage des messages pour votre rubrique, attribuez à `MessageRetentionPeriod` une valeur entière supérieure à zéro. Par exemple, pour conserver les messages de votre archive pendant 30 jours, définissez l'attribut `ArchivePolicy` comme suit :

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

Pour désactiver l'archivage des messages pour votre rubrique et effacer l'archive, annulez la définition de `ArchivePolicy` comme suit :

```
{}
```

Création d'une politique d'archivage des messages à l'aide du SDK

Pour utiliser un kit SDK AWS, vous devez le configurer avec vos informations d'identification. Pour en savoir plus, consultez Fichiers `config` et `credentials` partagés dans le Guide de référence des kits SDK et outils AWS.

L'exemple de code suivant montre comment définir l'attribut `ArchivePolicy` pour une rubrique Amazon SNS de sorte que tous les messages publiés dans la rubrique soient conservés pendant 30 jours.

```
// 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);
```

Création d'une politique d'archivage des messages à l'aide d'AWS CloudFormation

Pour créer une politique d'archivage à l'aide d'AWS CloudFormation, consultez `AWS::SNS::Topic` dans le Guide de l'utilisateur AWS CloudFormation.

Autorisation d'accès à une archive chiffrée

Pour permettre à un abonné de commencer à relire les messages d'une rubrique chiffrée, vous devez d'abord effectuer les tâches suivantes. Sachant que des messages anciens sont relus, Amazon SNS doit obtenir un accès `Decrypt` à la clé KMS qui a servi à chiffrer les messages dans l'archive.

1. Lorsque vous chiffrez des messages avec une clé KMS et que vous les stockez dans la rubrique, vous devez accorder à Amazon SNS la possibilité de déchiffrer ces messages via la stratégie de clé. Pour en savoir plus, consultez Octroi d'autorisations de déchiffrement à Amazon SNS.
2. Activez AWS KMS pour Amazon SNS. Pour en savoir plus, consultez Configuration des autorisations AWS KMS.

> **Important**
>
> Lorsque vous ajoutez les nouvelles sections à votre politique de clé KMS, ne changez pas les sections existantes dans la politique. Si le chiffrement est activé au niveau d'une rubrique et que la clé KMS est désactivée ou supprimée, ou encore que la stratégie de clé KMS n'est pas correctement configurée pour Amazon SNS, Amazon SNS ne peut pas relire les messages pour vos abonnés.

Octroi d'autorisations de déchiffrement à Amazon SNS

Pour permettre à Amazon SNS d'accéder aux messages chiffrés depuis l'archive de votre rubrique et de les relire sur les points de terminaison abonnés, vous devez autoriser le principal du service Amazon SNS à déchiffrer ces messages.

Voici un exemple de politique dont a besoin le principal du service Amazon SNS pour déchiffrer les messages stockés lors de la relecture des messages historiques de votre rubrique.

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

Surveillance des métriques d'archivage des messages avec Amazon CloudWatch

Vous pouvez surveiller les messages archivés avec Amazon CloudWatch en utilisant les métriques suivantes. Pour être averti de l'existence d'anomalies dans vos charges de travail et éviter de subir un impact, vous pouvez configurer des alarmes Amazon CloudWatch sur ces métriques. Pour en savoir plus, consultez Journalisation et surveillance dans Amazon SNS.

Métrique	Description
ApproximateNumberOfMessagesArchived	Indique au propriétaire de la rubrique le nombre total de messages archivés dans l'archive de la rubrique, selon une résolution de 60 minutes.
ApproximateNumberOfBytesArchived	Indique au propriétaire de la rubrique le nombre total d'octets archivés pour tous les messages contenus dans l'archive de la rubrique, selon une résolution de 60 minutes.

Métrique	Description
NumberOfMessagesArchiveProcessing	Indique au propriétaire de la rubrique le nombre de messages enregistrés dans l'archive de la rubrique sur la période, selon une résolution d'une minute.
NumberOfBytesArchiveProcessing	Indique au propriétaire de la rubrique le nombre total d'octets enregistrés dans l'archive de la rubrique sur la période, selon une résolution d'une minute.

L'API `GetTopicAttributes` possède une propriété `BeginningArchiveTime` qui représente l'horodatage le plus ancien à partir duquel un abonné peut lancer une relecture. Voici un exemple de réponse pour cette action d'API :

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

Relecture des messages pour les abonnés à une rubrique FIFO

La fonctionnalité de relecture d'Amazon SNS permet aux abonnés à un rubrique de récupérer les messages archivés dans le magasin de données de la rubrique et de les relivrer (ou relire) sur un point de terminaison abonné. Les messages peuvent être relus aussitôt que l'abonnement est créé. Un message relu présente un contenu, un `MessageId` et un `Timestamp` identiques à ceux de la copie d'origine. Il contient également l'attribut `Replayed`, qui vous permet de déterminer qu'il s'agit d'un message relu. Pour relire uniquement certains messages, vous pouvez ajouter une politique de filtre à votre abonnement. Pour en savoir plus sur le filtrage des messages, consultez Filtrage des messages relus.

Rubriques

- Création d'une politique de relecture des messages à l'aide de la AWS Management Console

- Ajout d'une politique de relecture à l'abonnement à l'aide de l'API
- Ajout d'une politique de relecture à l'abonnement à l'aide du SDK
- Filtrage des messages relus
- Surveillance des métriques de relecture des messages avec Amazon CloudWatch

Création d'une politique de relecture des messages à l'aide de la AWS Management Console

Cette option vous permet de créer une politique de relecture à l'aide de la AWS Management Console.

1. Connectez-vous à la console Amazon SNS.
2. Choisissez un abonnement à une rubrique ou créez-en un. Pour en savoir plus sur la création d'abonnements, consultez Abonnement à une rubrique Amazon SNS.
3. Pour démarrer la relecture des messages, accédez au menu déroulant Relecture et sélectionnez Lancer la relecture.
4. Dans la fenêtre modale Période de relecture, effectuez les sélections suivantes :

    a. Choisissez la date et l'heure de début de la relecture : choisissez la date (format AAAA/MM/JJ) et l'heure (format hh:mm:ss de 24 heures) auxquelles vous voulez lancer la relecture des messages archivés. L'heure de début doit être postérieure au début de l'heure approximative de l'archivage.

    b. (Facultatif) Choisissez la date et l'heure de fin de la relecture : choisissez la date (format AAAA/MM/JJ) et l'heure (format hh:mm:ss de 24 heures) auxquelles vous voulez arrêter la relecture des messages archivés.

    c. Sélectionnez Lancer la relecture.
5. (Facultatif) Pour arrêter une relecture de messages, accédez à la page Détails de l'abonnement et sélectionnez Arrêter la relecture dans le menu déroulant Relecture.
6. (Facultatif) Pour surveiller les métriques de relecture des messages depuis ce flux de travail en utilisant CloudWatch, consultez Surveillance des métriques de relecture des messages avec Amazon CloudWatch.

Pour afficher et modifier une politique de relecture des messages

Vous pouvez effectuer les actions suivantes depuis la page Détails de l'abonnement :

- Pour afficher le statut de relecture des messages, le champ État de la relecture affiche les valeurs suivantes :
  - Terminée : la relecture a permis de relivrer tous les messages et livre à présent les messages nouvellement publiés.
  - En cours : la relecture relit actuellement les messages sélectionnés.
  - Échec : la relecture n'a pas pu se terminer.
  - En attente : état par défaut au lancement de la relecture.
- (Facultatif) Pour modifier une politique de relecture des messages, accédez à la page Détails de l'abonnement et sélectionnez Lancer la relecture dans le menu déroulant Relecture. Le lancement d'une relecture remplace la lecture existante.

Ajout d'une politique de relecture à l'abonnement à l'aide de l'API

Pour relire des messages archivés, utilisez l'attribut `ReplayPolicy`. `ReplayPolicy` peut être utilisé avec les actions d'API `Subscribe` et `SetSubscriptionAttributes`. Cette politique présente les valeurs suivantes :

- `StartingPoint` (Obligatoire) : indique à quel moment lancer la relecture des messages.
- `EndingPoint` (Facultatif) : indique à quel moment arrêter la lecture des messages. Si `EndingPoint` est omis, la relecture se poursuit jusqu'à ce qu'elle rattrape l'heure actuelle.
- `PointType` (Obligatoire) : définit le type des point de départ et d'arrivée. Pour l'heure, la valeur prise en charge pour `PointType` est `Timestamp`.

Par exemple, pour récupérer suite à une défaillance en aval et renvoyer tous les messages pour une période de deux heures le 1er octobre 2023, utilisez l'action d'API `SetSubscriptionAttributes` pour définir un attribut `ReplayPolicy` comme suit :

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

Pour relire tous les messages envoyés à la rubrique à partir du 1er octobre 2023 et continuer à recevoir tous les messages nouvellement publiés dans votre rubrique, utilisez l'action d'API

`SetSubscriptionAttributes` pour définir un attribut `ReplayPolicy` pour votre abonnement comme suit :

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

Pour vérifier qu'un message a été relu, l'attribut booléen `Replayed` est ajouté à chaque message relu.

Ajout d'une politique de relecture à l'abonnement à l'aide du SDK

Pour utiliser un kit SDK AWS, vous devez le configurer avec vos informations d'identification. Pour en savoir plus, consultez Fichiers `config` et `credentials` partagés dans le Guide de référence des kits SDK et outils AWS.

L'exemple de code suivant montre comment définir l'attribut `ReplayPolicy` pour un abonnement de façon à relire les messages de l'archive de la rubrique FIFO d'Amazon SNS pour une fenêtre de temps de 2 heures le 1er octobre 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);
```

Filtrage des messages relus

Le filtrage des messages Amazon SNS vous permet de contrôler les messages relus par Amazon SNS sur le point de terminaison de votre abonné. Lorsque le filtrage et l'archivage des messages sont tous deux activés, Amazon SNS récupère le message dans le magasin de données de la rubrique avant de l'appliquer à l'attribut `FilterPolicy` de l'abonnement. Le message est remis au point de terminaison abonné s'il existe une correspondance ; sinon, le message est filtré. Pour de plus amples informations, consultez Stratégies de filtre d'abonnement Amazon SNS.

Surveillance des métriques de relecture des messages avec Amazon CloudWatch

Vous pouvez surveiller les messages relus avec Amazon CloudWatch en utilisant les métriques suivantes. Pour être averti de l'existence d'anomalies dans vos charges de travail et éviter de subir un impact, vous pouvez configurer des alarmes Amazon CloudWatch sur ces métriques. Pour en savoir plus, consultez Journalisation et surveillance dans Amazon SNS.

Métrique	Description
NumberOfReplayedNotificationsDelivered	Indique à l'abonné le nombre total de messages relus dans l'archive de la rubrique, selon une résolution d'une minute.
NumberOfReplayedNotificationsFailed	Indique à l'abonné le nombre total de messages relus dans l'archive de la rubrique et dont la remise a échoué, selon une résolution d'une minute.

Exemples de code pour les rubriques FIFO

Vous pouvez utiliser les exemples de code suivants pour intégrer le cas d'utilisation de la gestion des prix des pièces automobiles en utilisant une rubrique FIFO Amazon SNS avec une file d'attente FIFO Amazon SQS ou une file d’attente standard.

Rubriques

- Utilisation d'un kit SDK AWS
- Utiliser AWS CloudFormation

Utilisation d'un kit SDK AWS

En utilisant un kit SDK AWS, vous créez une rubrique FIFO Amazon SNS en définissant son attribut `FifoTopic` sur **`true`**. Vous créez une file d'attente FIFO Amazon SQS en définissant son attribut `FifoQueue` sur **`true`**. Vous devez également ajouter le suffixe **`.fifo`** au nom de chaque ressource FIFO. Après avoir créé une rubrique ou une file d'attente FIFO, vous ne pouvez pas la convertir en une rubrique ou une file d'attente standard.

Les exemples de code suivants créent les ressources de file d’attente standard et FIFO suivantes :

- La rubrique FIFO Amazon SNS qui distribue les mises à jour de prix
- Les files d'attente FIFO Amazon SQS qui fournissent ces mises à jour aux applications de gros et de détail
- La file d'attente standard Amazon SQS pour l'application d'analyse qui stocke les enregistrements, qui peuvent être interrogée à des fins de Business Intelligence (BI)
- Les abonnements FIFO Amazon SNS qui connectent les trois files d'attente à la rubrique

Cet exemple définit les politiques de filtre dans les abonnements. Si vous testez l'exemple en publiant un message dans la rubrique, assurez-vous de publier le message avec l'attribut `business`. Spécifiez `retail` ou `wholesale` pour la valeur d'attribut. Sinon le message est filtré et n'est pas remis aux files d'attente abonnées. Pour de plus amples informations, veuillez consulter Filtrage des messages pour les rubriques FIFO.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l’exemple complet et découvrez comment le configurer et l’exécuter dans le référentiel d’exemples de code AWS.

Cet exemple

- crée une rubrique FIFO Amazon SNS, deux files d'attente FIFO Amazon SQS et une file d'attente standard.
- abonne les files d’attente à la rubrique et publie un message dans la rubrique.

Le test vérifie que le message a bien été reçu pour chaque file d'attente. L'exemple complet montre également l'ajout de stratégies d'accès et supprime les ressources à la fin.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
 created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
 will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
 ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);
```

```
        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

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

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

            return topicArn;

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

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

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

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

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

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

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

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

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

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

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x.
  - CreateTopic
  - Publier
  - Subscribe

Python

Kit SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une rubrique FIFO Amazon SNS, abonnez les files d'attentes standard et FIFO Amazon SQS à la rubrique et publiez un message dans la rubrique.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)
```

```
    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
```

```
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

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

```
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
```

```
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Sid": "test-sid",
                                "Effect": "Allow",
                                "Principal": {"AWS": "*"},
                                "Action": "SQS:SendMessage",
                                "Resource": queue.attributes["QueueArn"],
                                "Condition": {
                                    "ArnLike": {"aws:SourceArn": topic_arn}
                                },
                            }
                        ],
                    }
                )
            }
        )
        logger.info("Added trust policy to the queue.")
    except ClientError as error:
        logger.exception("Couldn't add trust policy to the queue!")
        raise error


@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error
```

```
    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Python (Boto3) API Reference.
  - CreateTopic
  - Publier
  - Subscribe

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une rubrique FIFO, abonnez une file d'attente FIFO Amazon SQS à la rubrique et publiez un message dans une rubrique 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.
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API du kit AWS SDK pour SAP ABAP.
  - CreateTopic
  - Publier
  - Subscribe

Réception de messages provenant d'abonnements FIFO

Vous pouvez désormais recevoir les mises à jour des prix dans les trois applications abonnées. Comme indiqué dans le the section called “Cas d'utilisation des rubriques FIFO”, le point d'entrée de chaque application grand public est la file d'attente Amazon SQS, que sa fonction AWS Lambda correspondante peut interroger automatiquement. Lorsqu'une file d'attente Amazon SQS est une source d'évènements pour une fonction Lambda, Lambda met à l'échelle sa flotte de sondeurs en fonction des besoins pour consommer efficacement les messages.

Pour plus d'informations, consultez Utilisation de AWS Lambda avec Amazon SQS dans le Guide du développeur AWS Lambda. Pour plus d'informations sur l'écriture de vos propres sondeurs de file d'attente, consultez Recommandations pour les files d'attente standard et FIFO Amazon SQS dans le Guide du développeur Amazon Simple Queue Service et ReceiveMessage dans la Référence d'API Amazon Simple Queue Service.

Utiliser AWS CloudFormation

AWS CloudFormation vous permet d'utiliser un fichier de modèle pour créer et configurer simultanément un ensemble de ressources AWS en tant qu'unité unique. Cette section comporte un exemple de modèle qui crée les éléments suivants :

- La rubrique FIFO Amazon SNS qui distribue les mises à jour de prix
- Les files d'attente FIFO Amazon SQS qui fournissent ces mises à jour aux applications de gros et de détail
- La file d'attente standard Amazon SQS pour l'application d'analyse qui stocke les enregistrements, qui peuvent être interrogée à des fins de Business Intelligence (BI)
- Les abonnements FIFO Amazon SNS qui connectent les trois files d'attente à la rubrique
- Une stratégie de filtrage qui spécifie que les applications abonnées reçoivent uniquement les mises à jour de prix dont elles ont besoin

> **Note**
>
> Si vous testez cet exemple de code en publiant un message dans la rubrique, assurez-vous de publier le message avec l'attribut `business`. Spécifiez `retail` ou `wholesale` pour la valeur d'attribut. Sinon le message est filtré et n'est pas remis aux files d'attente abonnées.

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

Pour plus d'informations sur le déploiement de ressources AWS à l'aide d'un modèle AWS CloudFormation, consultez la page Démarrer du Guide de l'utilisateur AWS CloudFormation.

Publication de messages Amazon SNS

Une fois que vous avez créé une rubrique Amazon SNS et y avez abonné un point de terminaison, vous pouvez y publier des messages. Lorsqu'un message est publié, Amazon SNS tente de le remettre aux points de terminaison abonnés.

Rubriques

- Pour publier des messages dans une rubrique Amazon SNS à l'aide de la AWS Management Console
- Pour publier un message dans une rubrique à l'aide d'un kit SDK AWS
- Publication de messages volumineux avec Amazon SNS et Amazon S3
- Attributs de message Amazon SNS
- Traitement par lots de messages Amazon SNS

Pour publier des messages dans une rubrique Amazon SNS à l'aide de la AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation de gauche, choisissez Rubriques.
3. Sur la page Rubriques, sélectionnez une rubrique, puis choisissez Publier un message.

   La console ouvre la fenêtre Publier un message sur une rubrique.
4. Dans la section Détails de base, procédez comme suit :

   a. (Facultatif) Saisissez un message dans le champ Objet.

   b. Pour une rubrique FIFO, saisissez une valeur d'ID de groupe de messages. Les messages du même groupe de messages sont remis dans l'ordre dans lequel ils sont publiés.

   c. Pour une rubrique FIFO, saisissez un ID de déduplication de message. Cet ID est facultatif si vous avez activé l'option Déduplication des messages basée sur le contenu pour la rubrique.

   d. (Facultatif) Pour notifications push mobile, saisissez une valeur de Durée de vie (TTL) en secondes. Il s'agit du temps laissé à un service de notification push – tel qu'Apple Push Notification Service (APNs) ou Firebase Cloud Messaging (FCM) – pour diffuser le message au point de terminaison.

5. Dans la section Message body (Corps du messages), effectuez l'une des actions suivantes :

   a. Choisissez Charge utile identique pour tous les protocoles de livraison, puis saisissez un message.

   b. Choisissez Charge utile personnalisée pour chaque protocole de distribution, puis utilisez un objet JSON pour définir le message à envoyer à chaque protocole.

   Pour plus d'informations, consultez Publication avec des charges utiles spécifiques à la plate-forme.

6. Dans la section Attributs du message, ajoutez tous les attributs que vous souhaitez qu'Amazon SNS associe à l'attribut d'abonnement `FilterPolicy` afin de déterminer si le point de terminaison abonné est intéressé par le message publié.

   a. Pour Type, choisissez un type d'attribut, tel que String.Array.

   > **Note**
   >
   > Pour le type d'attribut String.Array, placez le tableau entre crochets (`[]`). Dans le tableau, placez les chaînes entre guillemets. Vous n'avez pas besoin de guillemets pour les chiffres ou les mots-clés `true`, `false` et `null`.

   b. Saisissez un attribut Nom, comme `customer_interests`.

   c. Saisissez un attribut Valeur, comme `["soccer", "rugby", "hockey"]`.

   Si le type d'attribut est String, String.Array ou Number, Amazon SNS évalue l'attribut du message par rapport à la politique de filtre d'un abonnement, le cas échéant, avant d'envoyer le message à cet abonnement, étant donné que l'étendue de la politique de filtre n'est pas explicitement définie sur `MessageBody`.

   Pour plus d'informations, consultez Attributs de message Amazon SNS.

7. Choisissez Publier le message.

   Le message est publié dans la rubrique et la console ouvre la page Détails de la rubrique.

Pour publier un message dans une rubrique à l'aide d'un kit SDK AWS

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

Les exemples de code suivants montrent comment utiliser`Publish`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Publier un message dans une rubrique

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

Publiez un message dans une rubrique avec des options de groupe, de duplication et d'attribut.

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

Appliquez les sélections de l'utilisateur à l'action de publication.

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

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for .NET Référence de l'API.

C++

Kit SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

Publiez un message avec un attribut.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
```

```
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

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

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Exemple 1 : pour publier un message dans une rubrique

L'exemple `publish` suivant publie le message spécifié sur la rubrique SNS spécifiée. Le message provient d'un fichier texte qui vous permet d'inclure des sauts de ligne.

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

Contenu de `message.txt` :

```
Hello World
Second Line
```

Sortie :

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

Exemple 2 : pour publier un SMS sur un numéro de téléphone

L'exemple `publish` suivant publie le message `Hello world!` sur le numéro de téléphone `+1-555-555-0100`.

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

Sortie :

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

- Pour plus d'informations sur l'API, consultez Publish dans la Référence des commandes AWS CLI .

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
    filterKey: {DataType: aws.String("String"), StringValue:
  aws.String(filterValue)},
   }
  }
  _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
  if err != nil {
   log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
  err)
  }
  return err
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for Go Référence de l'API.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
```

```
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

Publiez un message dans une rubrique avec des options de groupe, de duplication et d'attribut.

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

    let groupId, deduplicationId, choices;

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

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
```

```
        message: MESSAGES.deduplicationIdPrompt,
      });
    }

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

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

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

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

```
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for JavaScript.
- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for JavaScript .

Kotlin

Kit SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section Publier de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
```

```
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for PHP .

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple montre la publication d'un message avec un seul élément MessageAttribute déclaré en ligne.

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

Exemple 2 : Cet exemple montre la publication d'un message dont plusieurs MessageAttributes ont été déclarés à l'avance.

```
$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
```

- Pour plus de détails sur l'API, consultez la section Publication dans la référence des AWS Tools for PowerShell applets de commande.

Python

Kit SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Publiez un message avec des attributs afin qu'un abonnement puisse filtrer en fonction des attributs.

```
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
```

Publiez un message qui prend différentes formes en fonction du protocole de l'abonné.

```
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
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS Référence de l'API du kit SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for Ruby .

Rust

Kit SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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(())
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence du kit SDK AWS pour l'API Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus d'informations sur l'API, consultez Publish dans la Référence d'API du kit SDK AWS pour SAP ABAP.

Publication de messages volumineux avec Amazon SNS et Amazon S3

Pour publier des messages Amazon SNS volumineux, vous pouvez utiliser la bibliothèque client étendue Amazon SNS pour Java ou la bibliothèque client étendue Amazon SNS pour Python. Ces bibliothèques sont utiles pour les messages qui dépassent le maximum actuel de 256 Ko, avec un maximum de 2 Go. Les deux bibliothèques enregistrent la charge utile réelle dans un compartiment Amazon S3 et publient la référence de l'objet Amazon S3 stocké dans la rubrique Amazon SNS. Les files d'attente Amazon SQS abonnées peuvent utiliser la bibliothèque client étendue Amazon SQS pour Java pour déréférencer et récupérer les charges utiles depuis Amazon S3. D'autres points de terminaison, tels que Lambda, peuvent utiliser la bibliothèque commune Java de déchargement de la charge utile pour AWS afin de déréférencer et récupérer la charge utile.

> **Note**
>
> Les bibliothèques clientes étendues Amazon SNS sont compatibles avec les rubriques standard et FIFO.

Rubriques

Bibliothèque client étendue pour Java

Rubriques

Prérequis

Voici les prérequis à satisfaire pour utiliser la bibliothèque client étendue Amazon SNS pour Java :

- Un AWS SDK.

  L'exemple de cette page utilise le SDK AWS Java. Pour installer et configurer le kit SDK, consultez Configuration du kit SDK AWS pour Java dans le Guide du développeur AWS SDK for Java .
- Et Compte AWS avec les informations d'identification appropriées.

  Pour créer un Compte AWS, accédez à la page d'AWS accueil, puis choisissez Créer un AWS compte. Suivez les instructions à l'écran.

  Pour plus d'informations sur les informations d'identification, voir Configurer les AWS informations d'identification et la région pour le développement dans le guide du AWS SDK for Java développeur.
- Java 8 ou une version ultérieure.
- Bibliothèque client étendue Amazon SNS pour Java (également disponible à partir de Maven).

Configurer le stockage de messages

La bibliothèque Amazon SNS Extended Client utilise la bibliothèque commune Java Payload Offloading AWS pour le stockage et la récupération des messages. Vous pouvez configurer les options de stockage de messages suivantes pour Amazon S3 :

- Seuil personnalisé de tailles de messages – les messages dont les charges utiles et les attributs dépassent cette taille sont automatiquement stockés dans Amazon S3.
- Drapeau `alwaysThroughS3` – Définissez cette valeur sur `true` pour forcer le stockage de toutes les charges utiles de messages dans Amazon S3. Par exemple :

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

- Clé KMS personnalisée – Clé à utiliser pour le chiffrement côté serveur dans votre compartiment Amazon S3.
- Nom du compartiment – Nom du compartiment Amazon S3 où stocker les charges utiles de messages.

Exemple : publication de messages sur Amazon SNS avec la charge utile stockée dans Amazon S3

L'exemple de code suivant illustre comment :

- Créez un exemple de rubrique et de file d'attente.
- Abonnez la file d'attente pour recevoir des messages de la rubrique.
- Publiez un message de test.

La charge utile du message est stockée dans Amazon S3 et sa référence est publiée. Le client étendu Amazon SQS est utilisé pour recevoir le message.

Kit SDK pour Java 1.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Pour publier un message volumineux, vous pouvez utiliser la bibliothèque client étendue Amazon SNS pour Java. Le message que vous envoyez fait référence à un objet Amazon S3 contenant le contenu réel du message.

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

Autres protocoles de point de terminaison

Les bibliothèques Amazon SNS et Amazon SQS utilisent la bibliothèque commune Java de déchargement de la charge utile pour AWS pour stocker et récupérer les charges utiles des messages avec Amazon S3. Tout point de terminaison compatible Java (par exemple, un point de terminaison HTTPS implémenté en Java) peut utiliser la même bibliothèque pour déréférencer le contenu du message.

Les points de terminaison qui ne peuvent pas utiliser la bibliothèque commune Java de déchargement de charge utile AWS peuvent toujours publier des messages avec des charges utiles stockées dans Amazon S3. Voici un exemple de référence Amazon S3 qui est publiée par l'exemple de code ci-dessus :

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

Bibliothèque client étendue pour Python

Rubriques

Prérequis

Voici les prérequis à satisfaire pour utiliser la bibliothèque client étendue Amazon SNS pour Python :

- Un AWS SDK.

L'exemple de cette page utilise le SDK AWS Python Boto3. Pour installer et configurer le kit SDK, consultez la documentation Kit AWS SDK pour Python.

- Et Compte AWS avec les informations d'identification appropriées.

  Pour créer un Compte AWS, accédez à la page d'AWS accueil, puis choisissez Créer un AWS compte. Suivez les instructions à l'écran.

  Pour obtenir des informations sur les informations d'identification, consultez Informations d'identification dans le Guide du développeur AWS SDK pour Python.
- Python 3.x (ou version ultérieure) et pip.
- Bibliothèque client étendue Amazon SNS pour Python (également disponible à partir de PyPI).

Configurer le stockage de messages

Les attributs ci-dessous sont disponibles sur Boto3 Amazon SNS Client, Topic PlatformEndpointet Objects pour configurer les options de stockage des messages Amazon S3.

- `large_payload_support` – Nom du compartiment Amazon S3 pour stocker des messages volumineux.
- `message_size_threshold` – Seuil de stockage du message dans le compartiment de messages volumineux. Ne peut pas être inférieur à 0 ni supérieur à 262 144. La valeur par défaut est 262 144.
- `always_through_s3` – Si la valeur est `True`, tous les messages sont stockés dans Amazon S3. L'argument par défaut est `False`.
- `s3` : l'objet `resource` Boto3 Amazon S3 utilisé pour stocker les objets dans Amazon S3. Utilisez-le si vous souhaitez contrôler la ressource Amazon S3 (par exemple, une configuration Amazon S3 personnalisée ou des informations d'identification). S'il n'est pas défini précédemment lors de la première utilisation, la valeur par défaut est `boto3.resource("s3")`.

Exemple : publication de messages dans Amazon SNS avec la charge utile stockée dans Amazon S3

L'exemple de code suivant illustre comment :

- Créez un exemple de rubrique Amazon SNS et de file d'attente Amazon SQS.
- Abonnez la file d'attente pour recevoir des messages de la rubrique.

- Publiez un message de test.
- La charge utile du message est stockée dans Amazon S3 et sa référence est publiée.
- Imprimer le message publié depuis la file d'attente avec le message d'origine extrait d'Amazon S3.

Pour publier un message volumineux, vous pouvez utiliser la bibliothèque client étendue Amazon SNS pour Python. Le message que vous envoyez fait référence à un objet Amazon S3 contenant le contenu réel du message.

```
import boto3
import sns_extended_client
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store"
TOPIC_NAME = "---TOPIC-NAME---"
QUEUE_NAME = "---QUEUE-NAME---"

# Create an helper to fetch message from S3
def get_msg_from_s3(body):
    json_msg = loads(body)
    s3_client = boto3.client("s3")
    s3_object = s3_client.get_object(
        Bucket=json_msg[1].get("s3BucketName"), Key=json_msg[1].get("s3Key")
    )
    msg = s3_object.get("Body").read().decode()
    return msg

# Create an helper to fetch and print message SQS queue and S3
def fetch_and_print_from_sqs(sqs, queue_url):
    """Handy Helper to fetch and print message from SQS queue and S3"""
    message = sqs.receive_message(
        QueueUrl=queue_url, MessageAttributeNames=["All"], MaxNumberOfMessages=1
    ).get("Messages")[0]
    message_body = message.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is: {}\n".format(get_msg_from_s3(message_body)))

# Initialize the SNS client and create SNS Topic
sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
demo_topic_arn = create_topic_response.get("TopicArn")
```

```
# Create and subscribe an SQS queue to the SNS client
sqs = boto3.client("sqs")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
demo_queue_arn = sqs.get_queue_attributes(QueueUrl=demo_queue_url,
AttributeNames=["QueueArn"])["Attributes"].get("QueueArn")
# Set the RawMessageDelivery subscription attribute to TRUE
sns_extended_client.subscribe(TopicArn=demo_topic_arn, Protocol="sqs",
Endpoint=demo_queue_arn, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# To store all messages content in S3, set always_through_s3 to True
# In the example, we set message size threshold as 32 bytes, adjust this threshold as
 per your usecase
# Message will only be uploaded to S3 when its payload size exceeded threshold
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=demo_topic_arn,
    Message="This message should be published to S3 as it exceeds the
 message_size_threshold limit",
)
# Print message stored in s3
fetch_and_print_from_sqs(sqs, demo_queue_url)
```

Sortie

```
Published Message:
[
    "software.amazon.payloadoffloading.PayloadS3Pointer",
    {
        "s3BucketName": "extended-client-bucket-store",
        "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
    }
]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the message_size_threshold limit
```

Attributs de message Amazon SNS

Amazon SNS prend en charge la distribution des attributs de message qui vous permettent de fournir des éléments de métadonnées structurés (tels que des horodatages, des données

géospatiales, des signatures et des identifiants) sur le message. Pour les abonnements SQS, un maximum de 10 attributs de message peuvent être envoyés lorsque la remise des messages bruts est activée. Pour envoyer plus de 10 attributs de message, la remise des messages bruts doit être désactivée. Les messages comportant plus de 10 attributs de message dirigés vers des abonnements Amazon SQS ayant le paramètre Diffusion brute des messages activé sont supprimés en tant qu'erreurs côté client.

Les attributs de message sont facultatifs et séparés du corps du message, bien qu'ils soient envoyés en même temps. Le récepteur peut utiliser ces informations pour traiter le message sans avoir à traiter le corps du message en premier.

Pour plus d'informations sur l'envoi de messages avec des attributs à l'aide de AWS Management Console ou du kit AWS SDK for Java, veuillez consulter le didacticiel Pour publier des messages dans une rubrique Amazon SNS à l'aide de la AWS Management Console.

> **Note**
>
> Les attributs de message sont envoyés uniquement lorsque la structure du message est Chaîne et non JSON.

Vous pouvez également utiliser les attributs de message pour mieux structurer le message de notification push pour les points de terminaison mobiles. Dans ce scénario, les attributs du message sont uniquement utilisés pour vous aider à structurer le message de notification push. Les attributs ne sont pas distribués au point de terminaison, comme c'est le cas lors de l'envoi de messages avec les attributs du message aux points de terminaison Amazon SQS.

Vous pouvez également utiliser les attributs du message pour que vos messages puissent être filtrés à l'aide de stratégies de filtre d'abonnement. Vous pouvez appliquer des stratégies de filtre aux abonnements aux rubriques. Lorsqu'une politique de filtre est appliquée avec une étendue de politique de filtre définie sur `MessageAttributes` (par défaut), un abonnement reçoit uniquement les messages ayant des attributs acceptés par la politique. Pour de plus amples informations, veuillez consulter Filtrage des messages Amazon SNS.

> **Note**
>
> Lorsque des attributs de message sont utilisés pour le filtrage, la valeur doit être une chaîne JSON valide. Cela garantit que le message est remis à un abonnement avec le filtrage des attributs de message activé.

Éléments d'attribut de message et validation

Chaque attribut de message se compose des éléments suivants :

- Name – Le nom d'attribut de message peut contenir les caractères suivants : A-Z, a-z, 0-9, trait de soulignement (_), tiret (-) et point (.). Le nom ne doit pas commencer ou se terminer par un point et ne doit pas contenir plusieurs points à la suite. Le nom est sensible à la casse et doit être unique parmi tous les noms d'attribut pour le message. La longueur maximale du nom est de 256 caractères. Le nom ne peut pas commencer par `AWS.` ou `Amazon.` (indépendamment de la casse), car ces préfixes sont réservés pour une utilisation par Amazon Web Services.
- Type – Les types de données d'attribut de message pris en charge sont `String`, `String.Array`, `Number` et `Binary`. Le type de données est soumis aux mêmes restrictions de contenu que le corps du message. Le type de données est sensible à la casse et peut contenir jusqu'à 256 octets. Pour plus d'informations, consultez la section Types de données d'attribut de message et validation.
- Value – La valeur d'attribut du message spécifiée par l'utilisateur. Pour les données de type chaîne, l'attribut de valeur est soumis aux mêmes restrictions de contenu que le corps du message. Pour plus d'informations, consultez Publish dans la Référence d'API Amazon Simple Notification Service.

Les nom, type et valeur ne doivent pas être vides ni contenir la valeur null. Le corps du message ne doit pas être vide ni contenir la valeur null non plus. Toutes les parties de l'attribut du message, y compris le nom, le type et la valeur, sont incluses dans la restriction de taille du message, qui est de 256 Ko.

Types de données d'attribut de message et validation

Les types de données d'attribut de message identifient la façon dont les valeurs d'attribut de message sont traitées par Amazon SNS. Par exemple, si le type est un nombre, Amazon SNS valide le fait qu'il s'agit d'un nombre.

Amazon SNS prend en charge les types de données logiques suivants pour tous les points de terminaison, sauf indication contraire :

- String – Les chaînes sont au format Unicode avec un codage binaire UTF-8.
  Pour obtenir la liste des valeurs des codes, consultez http://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange.

  > **Note**
  >
  > Les valeurs de substitution ne sont pas prises en charge dans les attributs du message. Par exemple, l'utilisation d'une valeur de substitution pour représenter un emoji vous donnera l'erreur suivante :`Invalid attribute value was passed in for message attribute`.

- String.Array – Tableau, formaté sous la forme d'une chaîne, qui peut contenir plusieurs valeurs. Les valeurs peuvent être des chaînes, des nombres ou les mots-clés `true`, `false` et `null`. Un String.Array de type nombre ou booléen ne nécessite pas de guillemets. Les valeurs String.Array multiples sont séparées par des virgules.

  Ce type de données n'est pas pris en charge pour les abonnements AWS Lambda. Si vous spécifiez ce type de données pour les points de terminaison Lambda, il est transmis comme type de données `String` dans la charge utile JSON qu'Amazon SNS distribue à Lambda.

- Number – Les nombres sont des entiers positifs ou négatifs, ou des nombres à virgule flottante. La plage et la précision des nombres sont suffisantes pour englober la plupart des valeurs qui peuvent généralement être prises en charge par les entiers, les nombres flottants et les doubles. Un nombre peut avoir une valeur de -10^9 à 10^9, avec une précision de 5 chiffres après la virgule. Les zéros de début et de fin sont tronqués.

  Ce type de données n'est pas pris en charge pour les abonnements AWS Lambda. Si vous spécifiez ce type de données pour les points de terminaison Lambda, il est transmis comme type de données `String` dans la charge utile JSON qu'Amazon SNS distribue à Lambda.

- Binary – Les attributs de type binaire peuvent stocker n'importe quelle donnée binaire, notamment des données compressées, des données chiffrées ou des images.

Attributs de message réservés pour les notifications push mobiles

Le tableau suivant répertorie les attributs de message réservés pour les services de notification push mobiles que vous pouvez utiliser pour structurer votre message de notification push :

Service de notification push	Attribut de message réservé
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`

Service de notification push	Attribut de message réservé
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 rejettera les notifications Amazon SNS si les attributs de message ne répondent pas à ses exigences. Pour plus de détails, consultez Envoi de demandes de notification aux APN sur le site Web des développeurs Apple.

Traitement par lots de messages Amazon SNS

Qu'est-ce que le traitement par lots de messages ?

Une alternative à la publication de messages sur des rubriques Standard ou FIFO dans des demandes d'API `Publish` individuelles, consiste à utiliser l'API `PublishBatch` Amazon SNS permettant de publier jusqu'à 10 messages dans une seule demande d'API. L'envoi de messages par lots peut vous aider à réduire les coûts associés à la connexion d'applications distribuées (Messagerie A2A) ou en envoyant des notifications à des personnes (Messagerie A2P) avec Amazon SNS selon un facteur de 10 maximum. Amazon SNS dispose de quotas sur le nombre de messages que vous pouvez publier sur une rubrique par seconde en fonction de la région dans laquelle vous opérez. Consultez la page Points de terminaison et quotas Amazon SNS du guide Références générales AWS pour en savoir plus sur les quotas d'API.

> **Note**
>
> La taille agrégée totale de tous les messages que vous envoyez dans une seule demande d'API `PublishBatch` ne peut pas dépasser 262 144 octets (256 Ko).
> L'API `PublishBatch` utilise la même action d'API `Publish` pour les politiques IAM.

Comment fonctionne le traitement par lots de messages ?

La publication de messages avec l'API `PublishBatch` est similaire à la publication de messages avec l'API `Publish`. La principale différence réside dans le fait qu'un ID de lot unique doit être attribué à chaque message d'une demande d'API `PublishBatch` (jusqu'à 80 caractères). De cette façon, Amazon SNS peut renvoyer des réponses d'API individuelles pour chaque message d'un lot pour confirmer que chaque message a été publié ou qu'un échec s'est produit. Pour les messages publiés sur des rubriques FIFO, en plus de l'ajout de l'attribution d'un ID de lot unique, vous devez toujours inclure un `MessageDeduplicationID` et `MessageGroupId` pour chaque message individuel.

Exemples

Publication d'un lot de 10 messages sur une rubrique Standard

```
// Imports
```

```
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
```

```
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });
    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Publication d'un lot de 10 messages sur une rubrique FIFO

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToFifoTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i)
                        .withMessageGroupId("groupId")
                        .withMessageDeduplicationId("deduplicationId" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
```

```
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
            System.out.println("SequenceNumber for successful message: " +
 publishBatchResultEntry.getSequenceNumber());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });

    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Filtrage des messages Amazon SNS

Par défaut, un abonné d'une rubrique Amazon SNS reçoit chaque message publié dans la rubrique. Pour recevoir uniquement un sous-ensemble des messages, un abonné doit attribuer une politique de filtre à l'abonnement à la rubrique.

Une politique de filtre est un objet JSON qui contient des propriétés qui définissent les messages reçus par l'abonné. Amazon SNS prend en charge les politiques qui agissent sur les attributs du message ou sur le corps de message, en fonction de l'étendue de la politique de filtre que vous avez définie pour l'abonnement. Les politiques de filtre pour le corps de message supposent que la charge utile du message est un objet JSON bien formé.

Si un abonnement ne comporte pas de politique de filtre, l'abonné reçoit chaque message publié dans sa rubrique. Lorsque vous publiez un message dans une rubrique ayant une politique de filtre en place, Amazon SNS compare les attributs du message aux attributs de la politique de filtre pour chacun des abonnements de la rubrique. Si l'un des attributs ou l'une des propriétés du corps de message correspond, Amazon SNS envoie le message à l'abonné. Sinon, Amazon SNS n'envoie pas le message à cet abonné.

Pour plus d'informations, consultez Filtrer les messages publiés dans les rubriques.

Rubriques

- Étendue de la politique de filtre d'abonnement Amazon SNS
- Stratégies de filtre d'abonnement Amazon SNS
- Application d'une politique de filtrage d'abonnement
- Suppression d'une politique de filtrage d'abonnement

Étendue de la politique de filtre d'abonnement Amazon SNS

L'attribut de l'abonnement `FilterPolicyScope` vous permet de choisir l'étendue du filtrage en définissant l'une des valeurs suivantes :

- `MessageAttributes` : la politique de filtre est appliquée aux attributs du message. Il s'agit de l'option par défaut.
- `MessageBody` : la politique de filtre est appliquée au corps de message.

> **Note**
>
> Si aucune étendue de politique de filtre n'est définie pour une politique de filtre existante, l'étendue par défaut est `MessageAttributes`.

Stratégies de filtre d'abonnement Amazon SNS

Une politique de filtre d'abonnement vous permet de spécifier des noms de propriété et d'attribuer une liste de valeurs à chaque nom de propriété. Pour plus d'informations, consultez Filtrage des messages Amazon SNS.

Quand Amazon SNS analyse les attributs du message ou les propriétés du corps de message par rapport à la politique de filtre d'abonnement, il ignore ceux qui ne sont pas spécifiés dans la politique.

> **Important**
>
> AWS des services tels que IAM et Amazon SNS utilisent un modèle informatique distribué appelé cohérence éventuelle. Les ajouts ou modifications de politique de filtrage d'abonnement nécessitent jusqu'à 15 minutes pour prendre effet.

Un abonnement accepte un message dans les conditions suivantes :

- Lorsque l'étendue de la politique de filtre est définie sur `MessageAttributes`, chaque nom de propriété de la politique de filtre correspond à un nom d'attribut de message. Pour chaque nom de propriété correspondant dans la politique de filtre, au moins une valeur de propriété correspond à la valeur de l'attribut du message.
- Lorsque l'étendue de la politique de filtre est définie sur `MessageBody`, chaque nom de propriété de la politique de filtre correspond à un nom de propriété de corps de message. Pour chaque nom de propriété correspondant dans la politique de filtre, au moins une valeur de propriété correspond à la valeur de propriété de corps de message.

Amazon SNS prend en charge les opérateurs de filtre suivants :

- Logique AND
- Logique OR

- Opérateur OR
- Correspondance de clé
- Correspondance exacte des valeurs numériques
- Valeur numérique avec correspondance « anything-but »
- Correspondance de plage de valeur numérique
- Correspondance exacte de valeur de chaîne
- Valeur de chaîne avec correspondance « anything-but »
- Correspondance de chaînes utilisant un préfixe avec l'opérateur « anything-but »
- Equals-ignore case pour la valeur de chaîne
- Correspondance des adresses IP pour la valeur de chaîne
- Correspondance de préfixe pour la valeur de chaîne
- Correspondance de suffixe pour la valeur de chaîne

Exemples de stratégies de filtre

L'exemple suivant montre une charge utile de message diffusée par une rubrique Amazon SNS qui traite des transactions clients.

Le premier exemple comprend le champ `MessageAttributes` avec des attributs qui décrivent la transaction :

- Intérêts du client
- Nom du magasin
- État de l'événement
- Prix d'achat en USD

Étant donné que ce message comprend le champ `MessageAttributes`, tout abonnement à une rubrique qui définit un `FilterPolicy` peut accepter ou rejeter le message de façon sélective, à condition que `FilterPolicyScope` soit défini sur `MessageAttributes` dans l'abonnement. Pour plus d'informations sur l'application d'attributs à un message, consultez Attributs de message 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"
      }
   }
}
```

L'exemple suivant montre les mêmes attributs inclus dans le champ `Message`, également appelés charge utile du message ou corps de message. Tout abonnement à une rubrique qui définit un `FilterPolicy` peut accepter ou rejeter le message de façon sélective, à condition que `FilterPolicyScope` soit défini sur `MessageBody` dans l'abonnement.

```
{
"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"
}
```

Les politiques de filtre suivantes acceptent ou rejettent les messages en fonction de leurs noms et valeurs d'attribut.

Une politique qui accepte l'exemple de message

Les propriétés de la politique de filtre d'abonnement suivante correspondent aux attributs affectés à l'exemple de message. Notez que la même politique de filtre fonctionne pour un `FilterPolicyScope` qu'elle soit définie sur `MessageAttributes` ou `MessageBody`. Chaque abonné choisit son étendue de filtre en fonction de la composition des messages qu'il reçoit de la rubrique.

Si une seule propriété de cette politique ne correspond pas à un attribut affecté au message, la politique rejette le message.

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

Une politique qui rejette l'exemple de message

La politique de filtre d'abonnement suivante comporte plusieurs différences entre ses propriétés et les attributs affectés à l'exemple de message. Par exemple, comme le nom de propriété `encrypted` n'est pas présent dans les attributs du message, cette propriété de la politique entraîne le rejet du message, quelle que soit la valeur qui lui est affectée.

En cas de non-correspondance, la politique rejette le message.

```
{
   "store": ["example_corp"],
```

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

Contraintes de politique de filtre

Lorsque vous créez une politique de filtrage pour un abonnement Amazon SNS, il est important de comprendre comment les clés de cette politique sont prises en compte. Les principaux aspects à garder à l'esprit sont les suivants :

1. Clés parents : les clés parents sont les clés de niveau supérieur de la politique de filtrage. Il s'agit des clés pour lesquelles vous spécifiez des valeurs ou des contraintes.
2. Noms d'attribut — Les clés parents sont considérées comme les noms d'attributs dans la politique de filtrage. Les valeurs ou les contraintes que vous spécifiez pour ces clés seront appliquées aux attributs correspondants dans la charge utile du message.
3. Valeurs valides — Les valeurs spécifiées pour les clés parents doivent être une chaîne, un tableau de chaînes ou un nombre. Si la valeur est un objet (par exemple, un objet JSON), elle ne sera pas considérée comme une clé valide dans la politique de filtrage.

Examinons l'exemple de politique de filtrage suivant :

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

Dans cet exemple, les clés suivantes sont prises en compte dans le cadre de la politique de filtrage :

- `state`

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

Le résultat clé n'est pas pris en compte, car il contient un objet JSON comme valeur, plutôt qu'une chaîne, un tableau de chaînes ou un nombre.

Voici un autre exemple :

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

Dans ce cas, seules les clés `key_d` et 4 `key_f` sont prises en compte dans le cadre de la politique de filtrage, car les valeurs qui leur sont attribuées sont soit une chaîne, soit un tableau de chaînes. Les clés parents `key_a`key_b, et ne `key_c` sont pas prises en compte, car elles contiennent des objets JSON imbriqués comme valeurs.

Rubriques

- Contraintes de politique de filtre
- Contraintes de politique pour le filtrage basé sur les attributs
- Contraintes de politique pour le filtrage basé sur la charge utile

Contraintes de politique de filtre

- Correspondance de chaînes : pour la correspondance de chaînes dans la politique de filtrage, la comparaison fait la distinction majuscules/majuscules.
- Correspondance numérique — Pour la correspondance numérique, la valeur peut être comprise entre -10^{9} et 10^{9} (-1 milliard à 1 milliard), avec une précision de cinq chiffres après la virgule décimale.
- Complexité de la politique de filtre — Compte tenu de la complexité de la politique de filtrage, la combinaison totale de valeurs ne doit pas dépasser 150. Pour calculer la combinaison totale, multipliez le nombre de valeurs de chaque tableau dans la politique de filtrage.

Prenons l'exemple de politique suivant :

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

Dans cette politique :

- Le premier tableau comporte 3 valeurs
- Le deuxième tableau a une valeur
- Le troisième tableau contient 2 valeurs

La combinaison totale est calculée comme suit :

- 3 x 1 x 2 = 6

Syntaxe de la politique de filtrage

Le JSON de la politique de filtre peut contenir ce qui suit :

- Chaînes entre guillemets
- Nombres
- Mots-clés `true`, `false` et `null`, sans guillemets

Lorsque vous utilisez l'API Amazon SNS, vous devez transmettre le JSON de la politique de filtrage sous forme de chaîne UTF-8 valide.

Limites de la politique de filtrage

- La taille maximale d'une politique de filtrage est de 256 Ko.
- Par défaut, vous pouvez avoir jusqu'à 200 politiques de filtrage par sujet et 10 000 politiques de filtrage par AWS compte.
- Cette limite de politique n'empêcherait pas la création d'abonnements aux files d'attente Amazon SQS avec l'`Subscribe`API. Toutefois, elle échouera lorsque vous attachez la politique de filtre à l'appel d'API `Subscribe` (ou à l'appel d'API `SetSubscriptionAttributes`).
- Pour demander l'augmentation de ce quota, vous pouvez utiliser AWS Service Quotas.

Contraintes de politique pour le filtrage basé sur les attributs

- Le filtrage basé sur les attributs est l'option par défaut. `FilterPolicyScope` est défini sur `MessageAttributes` dans l'abonnement.
- Amazon SNS n'accepte pas de politique de filtre imbriquée pour le filtrage basé sur les attributs.
- Amazon SNS compare les propriétés de politique uniquement aux attributs de message ayant les types de données suivants :
  - `String`
  - `String.Array`

    > **Important**
    >
    > Il n'est pas recommandé de transmettre des objets dans des tableaux car cela peut produire des résultats inattendus en raison de l'imbrication, qui n'est pas prise en charge par le filtrage basé sur les attributs. Utilisez le filtrage basé sur la charge utile pour les politiques imbriquées.

  - `Number`
- Amazon SNS ignore les attributs de message ayant le type de données `Binary`.
- Une politique de filtre peut comporter un maximum de cinq noms d'attributs.

Contraintes de politique pour le filtrage basé sur la charge utile

Amazon SNS accepte une politique de filtrage imbriquée pour le filtrage basé sur la charge utile. Pour calculer la combinaison totale de valeurs dans la politique de filtrage, multipliez le nombre de valeurs dans chaque tableau imbriqué.

Prenons l'exemple de politique suivant :

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

Dans cette politique :

- Le premier tableau contient quatre valeurs dans une clé imbriquée à trois niveaux.
- La seconde contient trois valeurs dans une clé imbriquée à deux niveaux.

La combinaison totale est calculée comme suit :

- 4 x 3 x 3 x 2 = 72

Limites des politiques

Une politique de filtrage peut comporter un maximum de cinq clés parentes (clés de niveau supérieur). Dans le cas d'une politique imbriquée, seules les clés parentes sont prises en compte dans le calcul de la limite de cinq clés.

Plage numérique

Pour la correspondance numérique dans la politique de filtrage, la valeur peut être comprise entre -10^9 et 10^9 (-1 milliard à 1 milliard), avec une précision de cinq chiffres après la virgule décimale.

Passage au filtrage basé sur la charge utile

Pour passer du filtrage basé sur les attributs (par défaut) au filtrage basé sur la charge utile, vous devez définir l'étendue `FilterPolicyScope` sur `MessageBody` dans l'abonnement.

Logique AND/OR

Vous pouvez utiliser les opérations qui incluent la logique AND/OR pour mettre en correspondance les attributs de message ou les propriétés de corps de message.

Rubriques

Logique AND

Vous pouvez appliquer une logique AND en utilisant plusieurs noms de propriété.

Examinons la politique suivante :

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

Elle correspond à tous les attributs de message ou propriétés de corps de message ayant la valeur `customer_interests` définie sur `rugby` et la valeur `price_usd` définie sur un nombre supérieur à 100.

> **Note**
>
> Vous ne pouvez pas appliquer la logique AND à des valeurs d'un même attribut.

Logique OR

Vous pouvez appliquer une logique OR en affectant plusieurs valeurs à un nom de propriété.

Examinons la politique suivante :

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

Elle correspond à tous les attributs de message ou propriétés de corps de message ayant la valeur `customer_interests` définie sur `rugby`, `football` ou `baseball`.

Opérateur OR

Vous pouvez utiliser l'opérateur `"$or"` pour définir explicitement une politique de filtrage afin d'exprimer la relation OR entre plusieurs attributs de la politique.

Amazon SNS ne reconnaît une relation `"$or"` que lorsque la politique répond à toutes les conditions suivantes. Lorsque toutes ces conditions ne sont pas remplies, `"$or"` est traité comme un nom d'attribut normal, comme toute autre chaîne de la politique.

- Il existe un attribut de champ `"$or"` dans la règle suivi d'un tableau, par exemple, `“$or” : []`.
- Il existe au moins 2 objets dans le tableau `"$or"` : `"$or": [{}, {}]`.
- Aucun des objets du tableau `"$or"` ne possède de nom de champ correspondant à des mots clés réservés.

Sinon, `"$or"` est traité comme un nom d'attribut normal, comme les autres chaînes de la politique.

La politique suivante n'est pas analysée comme une relation OR, car le numérique et le préfixe sont des mots-clés réservés.

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

Exemples d'opérateurs **OR**

0R standard :

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

```
  ]
}
```

La logique de filtrage pour cette politique est la suivante :

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

Elle met en correspondance l'un ou l'autre des attributs de message suivants :

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

or

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

Elle correspond également à l'un des corps de message suivants :

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

or

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

Contraintes politiques qui incluent les relations **`OR`**

Examinons la politique suivante :

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

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

La logique de cette politique peut également être simplifiée comme suit :

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

Le calcul de la complexité des politiques comportant des relations OR peut être simplifié en additionnant les complexités des combinaisons pour chaque instruction OR.

La combinaison totale est calculée comme suit :

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

`source` possède une valeur, `metricName` deux valeurs, `metricType` une valeur, `metricId` deux valeurs et `spaceId` trois valeurs.

Examinons la politique de filtre imbriqué suivante :

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

```
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

La logique de cette politique peut être simplifiée comme suit :

```
("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")
```

Le calcul du total des combinaisons est le même pour les politiques non imbriquées, mais nous devons prendre en compte le niveau d'imbrication d'une clé.

La combinaison totale est calculée comme suit :

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

`metricName` possède deux valeurs, `namespace` en possède deux, `scope` est une clé imbriquée à deux niveaux avec une valeur, `source` est une clé imbriquée à deux niveaux avec une valeur et `type` est une clé imbriquée à deux niveaux avec une valeur.

Correspondance de clé

Vous pouvez utiliser l'opérateur `exists` pour faire correspondre les messages entrants avec ou sans propriétés spécifiées dans la politique de filtre. La correspondance `exists` ne fonctionne que sur les nœuds leaf. Elle ne fonctionne pas sur des nœuds intermédiaires.

- Utilisez `"exists": true` pour faire correspondre les messages entrants qui incluent la propriété spécifiée. La clé doit avoir une valeur non nulle et non vide.

  Par exemple, la propriété de politique suivante utilise l'opérateur `exists` avec une valeur de `true` :

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

Elle correspond à toute liste d'attributs de message contenant la clé d'attribut `store`, comme :

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

Elle correspond également à l'un des corps de message suivants :

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

Cependant, elle ne correspond à aucune liste d'attributs de message sans l'attribut de clé `store`, tel que :

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

Elle ne correspond pas non plus au corps de message suivant :

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

- Utilisez `"exists": false` pour faire correspondre les messages entrants qui n'incluent pas la propriété spécifiée.

> **Note**
>
> `"exists": false` ne correspond que si au moins un attribut est présent. Si un jeu d'attributs est vide, le filtre ne correspond pas.

Par exemple, la propriété de politique suivante utilise l'opérateur `exists` avec une valeur de `false` :

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

Elle ne correspond à aucune liste d'attributs de message contenant la clé d'attribut `store`, comme :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Cependant, elle correspond à toute liste d'attributs de message sans l'attribut de clé `store`, tel que :

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

Elle correspond également au corps de message suivant :

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

Correspondance de valeur numérique

Vous pouvez filtrer les messages en faisant correspondre des valeurs numériques à des valeurs d'attributs de message, ou à des valeurs de propriétés du corps de message. Les valeurs numériques ne sont pas placées entre guillemets doubles dans la politique JSON. Vous pouvez utiliser les opérations numériques suivantes pour filtrer.

> **Note**
>
> Les préfixes sont pris en charge pour la correspondance de chaîne uniquement.

Rubriques

Correspondance exacte

Lorsqu'une valeur de propriété de politique inclut le mot-clé `numeric` et l'opérateur =, elle correspond à toute valeur d'attribut de message ou de propriété de corps de message qui a le même nom et la même valeur numérique.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Correspondance de type « anything-but » (tout-sauf)

Lorsqu'une valeur de propriété de politique inclut le mot-clé `anything-but`, elle correspond à toute valeur d'attribut de message ou de propriété de corps de message qui n'inclut aucune des valeurs de propriété de la politique.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Elle correspond également à l'attribut de message suivant (parce qu'il contient une valeur qui n'est pas `100` ou `500`) :

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

Elle correspond également au corps de message suivant (parce qu'il contient une valeur qui n'est pas `100` ou `500`) :

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Correspondance de plage de valeurs

En plus de l'opérateur =, une propriété de politique numérique peut inclure les opérateurs suivants : <, <=, > et >=.

Examinons la propriété de politique suivante :

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

Elle correspond à tout attribut de message ou propriété de corps de message avec des valeurs numériques négatives.

Prenons l'exemple d'un autre attribut de message :

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

Elle correspond à tout attribut de message ou propriété de corps de message avec des nombres positifs allant jusqu'à 150 inclus.

Correspondance de valeur de chaîne

Vous pouvez filtrer les messages en faisant correspondre des valeurs de chaîne à des valeurs d'attributs de message, ou à des valeurs de propriétés du corps de message. Les valeurs de chaîne sont placées entre guillemets doubles dans la politique JSON. Vous pouvez utiliser les opérations de chaîne suivantes pour mettre en correspondance les attributs de message ou le corps de message.

Rubriques

- Correspondance de préfixe
- Correspondance des suffixes

Correspondance exacte

La mise en correspondance exacte a lieu lorsqu'une valeur de propriété de politique correspond à une ou plusieurs valeurs d'attribut de message.

Examinons la propriété de politique suivante :

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

Elle correspond aux attributs de message suivants :

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

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

Elle correspond également aux corps de message suivants :

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

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Correspondance de type « anything-but » (tout-sauf)

Lorsqu'une valeur de propriété de politique inclut le mot-clé `anything-but`, elle correspond à toute valeur d'attribut de message ou de corps de message qui n'inclut pas l'une des valeurs de propriété de la politique. `anything-but` peut être combiné avec `"exists": false`.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Elle correspond également à l'attribut de message suivant (parce qu'il contient une valeur qui n'est pas `rugby` ou `tennis`) :

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

Elle correspond également au corps de message suivant (parce qu'il contient une valeur qui n'est pas `rugby` ou `tennis`) :

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Utilisation d'un préfixe avec l'opérateur **anything-but**

Pour la correspondance de la chaîne, vous pouvez également utiliser un préfixe avec l'opérateur `anything-but`. Par exemple, la propriété de politique suivante refuse le préfixe `order-` :

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

Elle correspond à l'un des attributs suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

```
{
```

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

quals-ignore-case Correspondance E

Lorsqu'une propriété de politique inclut le mot-clé `equals-ignore-case`, elle effectue une correspondance non sensible à la casse avec tout attribut de message ou toute valeur de propriété de corps.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Correspondance d'adresses IP

Vous pouvez utiliser l’opérateur `cidr` pour vérifier si un message entrant provient d'une adresse IP ou d'un sous-réseau spécifique.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Correspondance de préfixe

Lorsqu'une propriété de politique inclut le mot-clé `prefix`, elle correspond à toute valeur d'attribut de message ou de propriété de corps qui commence par les caractères spécifiés.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Correspondance des suffixes

Lorsqu'une propriété de politique inclut le mot-clé `suffix`, elle correspond à toute valeur d'attribut de message ou de propriété de corps qui termine par les caractères spécifiés.

Examinons la propriété de politique suivante :

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

Il met en correspondance l'un ou l'autre des attributs de message suivants :

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

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

Elle correspond également à l'un des corps de message suivants :

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

```
}
```

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

Cependant, il ne met pas en correspondance l'attribut de message suivant :

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

Elle ne correspond pas non plus au corps de message suivant :

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

Application d'une politique de filtrage d'abonnement

Vous pouvez appliquer une politique de filtrage à un abonnement Amazon SNS à l'aide de la console Amazon SNS. Ou, pour appliquer des politiques par programmation, vous pouvez utiliser l'API Amazon SNS, AWS Command Line Interface le AWS CLI() ou AWS tout autre SDK compatible avec Amazon SNS. Vous pouvez également utiliser AWS CloudFormation.

> **Important**
>
> AWS des services tels que IAM et Amazon SNS utilisent un modèle informatique distribué appelé cohérence éventuelle. Les ajouts ou modifications de politique de filtrage d'abonnement nécessitent jusqu'à 15 minutes pour prendre effet.

AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Abonnements.
3. Sélectionnez un abonnement, puis choisissez Edit (Modifier).
4. Sur la page Edit (Modifier), développez la section Subscription filter policy (Stratégie de filtrage d'abonnement).

5. Choisissez entre le filtrage basé sur les attributs ou le filtrage basé sur la charge utile.
6. Dans le champ JSON editor (Éditeur JSON), indiquez le corps JSON de votre stratégie de filtrage.
7. Sélectionnez Enregistrer les modifications.

   Amazon SNS applique votre politique de filtrage à l'abonnement.

AWS CLI

Pour appliquer une politique de filtrage avec le AWS Command Line Interface (AWS CLI), utilisez la `set-subscription-attributes`commande, comme indiqué dans l'exemple suivant. Pour l'option `--attribute-name`, spécifiez `FilterPolicy`. Pour `--attribute-value`, spécifiez votre politique JSON.

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

Pour fournir un code JSON valide pour votre politique, placez les noms et les valeurs d'attributs entre guillemets. Vous devez également placer la totalité de l'argument de la politique entre guillemets. Pour éviter l'échappement des guillemets, vous pouvez utiliser des apostrophes pour entourer la politique et des guillemets pour entourer les noms et les valeurs JSON, comme illustré dans l'exemple ci-dessus.

Si vous souhaitez passer du filtrage des messages basé sur les attributs (par défaut) au filtrage des messages basé sur la charge utile, vous pouvez également utiliser la set-subscription-attributescommande. Pour l'option `--attribute-name`, spécifiez `FilterPolicyScope`. Pour `--attribute-value`, spécifiez `MessageBody`.

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

Pour vérifier que votre politique de filtre a été appliquée, utilisez la commande `get-subscription-attributes`. Les attributs dans la sortie du terminal doivent afficher votre politique de filtre pour la clé `FilterPolicy`, comme illustré dans l'exemple suivant:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
```

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

AWS SDK

Les exemples de code suivants montrent comment utiliser`SetSubscriptionAttributes`.

> **Important**
>
> Si vous utilisez l'exemple SDK for Java 2.x, la classe `SNSMessageFilterPolicy` n'est pas disponible dès le départ. Pour obtenir des instructions sur l'installation de cette classe, consultez l'exemple du GitHub site Web.

CLI

AWS CLI

Pour définir des attributs d'abonnement

L'exemple `set-subscription-attributes` suivant définit l'attribut `RawMessageDelivery` sur un abonnement 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
```

Cette commande ne produit aucun résultat.

L'exemple `set-subscription-attributes` suivant définit un attribut `FilterPolicy` sur un abonnement 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\"] }"
```

Cette commande ne produit aucun résultat.

L'exemple `set-subscription-attributes` suivant supprime l'attribut `FilterPolicy` d'un abonnement 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 "{}"
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous SetSubscriptionAttributesà la section Référence des AWS CLI commandes.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous SetSubscriptionAttributesà la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez SetSubscriptionAttributesle AWS manuel de référence de l'API SDK for Python (Boto3).

API Amazon SNS

Pour appliquer une politique de filtrage à l'API Amazon SNS, effectuez une demande d'action `SetSubscriptionAttributes`. Définissez le paramètre `AttributeName` sur `FilterPolicy` et définissez le paramètre `AttributeValue` sur votre politique de filtre JSON.

Si vous souhaitez passer du filtrage des messages basé sur les attributs (par défaut) au filtrage des messages basé sur la charge utile, vous pouvez également utiliser l'action `SetSubscriptionAttributes`. Définissez le paramètre `AttributeName` sur `FilterPolicyScope`, puis le paramètre `AttributeValue` sur `MessageBody`.

AWS CloudFormation

Pour appliquer une politique de filtrage en utilisant AWS CloudFormation, utilisez un modèle JSON ou YAML pour créer une AWS CloudFormation pile. Pour plus d'informations, consultez la `FilterPolicy`propriété de la `AWS::SNS::Subscription` ressource dans le guide de AWS CloudFormation l'utilisateur et l'exemple de AWS CloudFormation modèle.

1. Connectez-vous à la console AWS CloudFormation.
2. Sélectionnez Créer une pile.
3. Sur la page Select Template (Sélectionner un modèle), choisissez Upload a template to Amazon S3 (Charger un modèle dans Amazon S3), sélectionnez le fichier, puis Next (Suivant).
4. Sur la page Spécification de détails de base de données, procédez comme suit :
   a. Pour le Nom de la pile, tapez `MyFilterPolicyStack`.
   b. Pour myHttpEndpoint, saisissez le point de terminaison HTTP auquel vous souhaitez vous abonner à votre sujet.

   > **Tip**
   >
   > Si vous n'avez pas de point de terminaison HTTP, vous pouvez créer en.

5. Dans la page Options, choisissez Next (Suivant).
6. Sur la page Review (Vérification), choisissez Create (Créer).

Suppression d'une politique de filtrage d'abonnement

Pour arrêter de filtrer les messages qui sont envoyés à un abonnement, supprimez la politique de filtre de l'abonnement en la remplaçant par un corps JSON vide. Après que vous avez supprimé la politique, l'abonnement accepte tous les messages qui sont publiés dans celui-ci.

AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Abonnements.
3. Sélectionnez un abonnement, puis choisissez Edit (Modifier).
4. Dans la page Modifier ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***, développez la section politique de filtrage d'abonnement.
5. Dans le champ Éditeur JSON, indiquez un corps JSON vide pour votre politique de filtre : `{}`.
6. Sélectionnez Save Changes (Enregistrer les modifications).

   Amazon SNS applique votre politique de filtrage à l'abonnement.

AWS CLI

Pour supprimer une politique de filtre avec l'AWS CLI, utilisez la commande `set-subscription-attributes` et fournissez un corps JSON vide pour l'argument `--attribute-value` :

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

API Amazon SNS

Pour supprimer une politique de filtrage avec l'API Amazon SNS, effectuez une demande d'action `SetSubscriptionAttributes`. Définissez le paramètre `AttributeName` sur `FilterPolicy` et fournissez un corps JSON vide pour le paramètre `AttributeValue`.

Message Data Protection

Rubriques

Qu'est-ce que Message Data Protection ?

Message Data Protection protège les données publiées dans vos rubriques Amazon SNS en utilisant des politiques de protection des données pour auditer, masquer, supprimer ou bloquer les informations sensibles qui circulent entre les applications ou les services AWS.

Message Data Protection analyse les données en mouvement à la recherche de données d'identification personnelle (PII) et d'informations protégées sur la santé (PHI) à l'aide d'identifiants de données. Vous pouvez choisir d'utiliser des identifiants de données prédéfinis (ou gérés Amazon SNS) tels que des noms, des adresses, des numéros de carte de crédit ou des codes de médicaments sur ordonnance, ou vous pouvez créer vos propres identifiants de données personnalisés, adaptés à votre cas d'utilisation métier spécifique. À l'aide des informations analysées, Message Data Protection fournit des journaux d'audit détaillés et vous permet de prendre des mesures pour protéger ces données.

Message Data Protection prend en charge les actions suivantes afin de protéger les informations sensibles des clients :

- Audit - Auditez jusqu'à 99 % des données publiées dans une rubrique Amazon SNS. Vous pouvez ensuite choisir d'envoyer les résultats à Amazon CloudWatch, Amazon S3 ou Amazon Data Firehose.
- Anonymisation : masque ou supprime les données sensibles sans interrompre la publication ou la distribution des messages.
- Refuser - Bloquer la transmission de données entre les applications et les ressources AWS si des données sensibles sont présentes dans la charge utile.

> **Note**
>
> Amazon SNS prend en charge Message Data Protection pour les rubriques standards Amazon SNS uniquement.

Pourquoi utiliser Message Data Protection ?

En introduisant Message Data Protection dans vos programmes de gouvernance, de gestion des risques et de conformité, vous pouvez mettre en œuvre des politiques de protection des données qui vous aident à identifier et à prévenir les fuites de données. Cela fournit à vos équipes des outils qui peuvent aider à réduire les risques financiers, juridiques et réglementaires en se conformant aux réglementations de confidentialité telles que HIPAA, le RGPD, PCI et FedRAMP. Cela libère également vos développeurs des coûts opérationnels associés à la création et à la gestion de vos propres outils pour protéger les données sensibles.

Par exemple, vous pouvez utiliser Message Data Protection pour créer une politique d'audit afin de déterminer si l'un de vos systèmes envoie ou reçoit par inadvertance des données sensibles. Si les résultats de votre audit indiquent que les systèmes envoient des informations de carte de crédit à des systèmes qui n'en ont pas besoin, vous pouvez mettre en œuvre une politique de blocage pour empêcher la transmission des données.

> **Note**
>
> Amazon SNS prend en charge Message Data Protection pour les rubriques standards Amazon SNS uniquement.

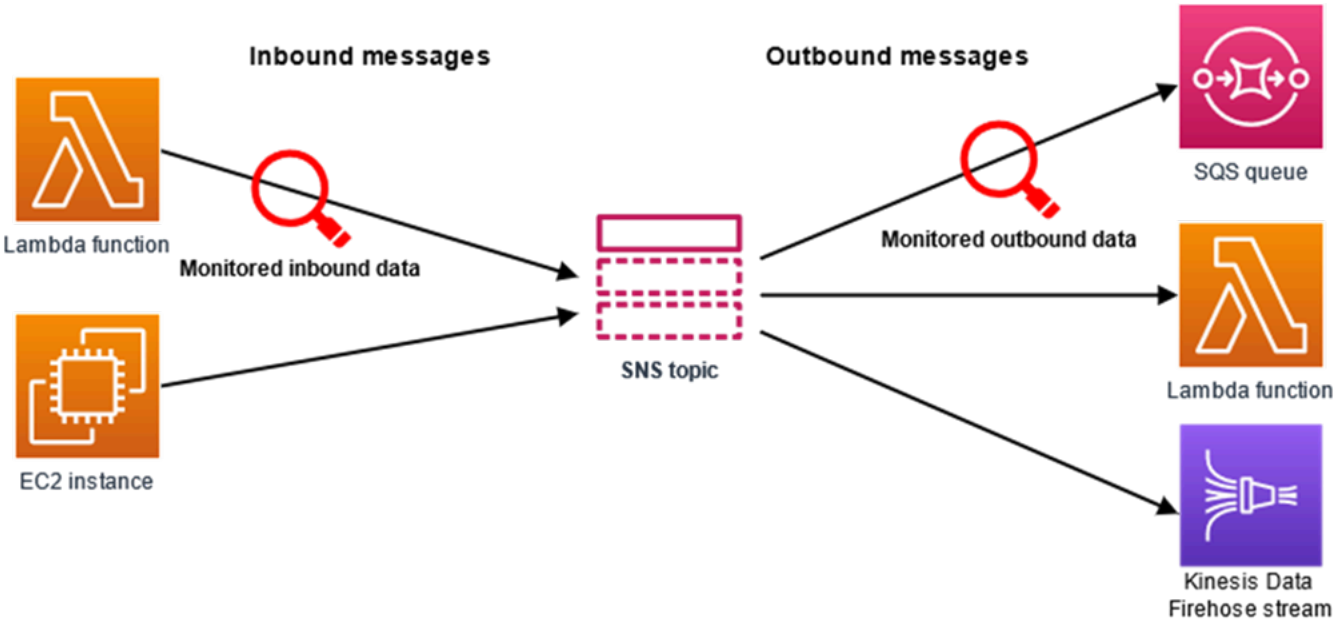
Comprendre les politiques de protection des données

Rubriques

- Création de politiques de protection des données
- Suppression de politiques de protection des données dans Amazon SNS

En quoi consistent les politiques de protection des données ?

Amazon SNS utilise des politiques de protection des données pour sélectionner les données sensibles que vous souhaitez analyser et les mesures que vous souhaitez prendre pour empêcher que ces données ne soient échangées par vos rubriques Amazon SNS. Pour sélectionner les données sensibles qui vous intéressent, vous utilisez des identifiants de données. Message Data Protection Amazon SNS détecte ensuite les données sensibles à l'aide du machine learning et de la correspondance de modèles. Pour agir sur les identifiants de données détectés, vous pouvez définir une opération d'audit, d'anonymisation ou de refus. Ces opérations vous permettent de consigner les données sensibles détectées (ou non détectées), de masquer ou de supprimer les données sensibles, ou de refuser la diffusion des messages.

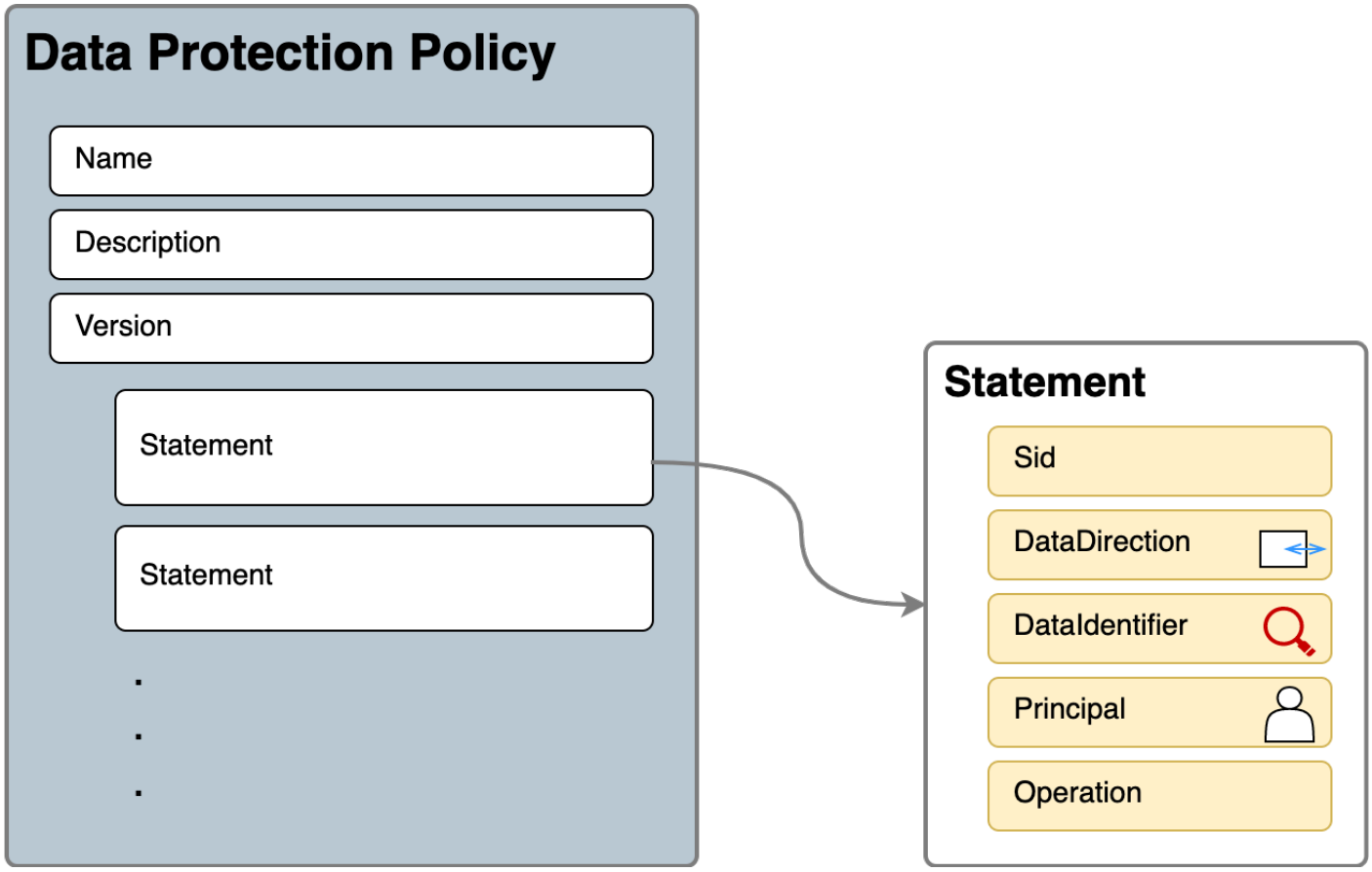


Comment est structurée la politique de protection des données ?

Comme illustré dans la figure suivante, un document de politique de protection des données inclut les éléments suivants :

- Informations facultatives sur l'ensemble de la politique (en haut du document)
- Une ou plusieurs instructions individuelles

Chaque instruction inclut des informations sur une seule autorisation.



Une seule politique de protection des données peut être définie par rubrique Amazon SNS. La politique de protection des données peut comporter une ou plusieurs instructions de refus ou d'anonymisation, mais seulement une instruction d'audit.

Propriétés JSON pour la politique de protection des données

Une politique de protection des données nécessite les informations de politique de base suivantes à des fins d'identification :

- Nom - Nom de la politique.
- Description (Facultatif) - Description de la politique.
- Version - Version du langage de la politique. La version actuelle est 2021-06-01.
- Instruction - Liste d'instructions qui spécifie les actions de la politique de protection des données.

```
{
  "Name": "basicPII-protection",
```

```
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

Propriétés JSON pour une instruction de politique

Une instruction de politique définit le contexte de détection pour l'opération de protection des données.

- Sid (facultatif) - L'identifiant de l’instruction.
- DataDirection - Entrant (pour les demandes d'API de publication) ou sortant (pour les envois de notifications) en ce qui concerne la rubrique Amazon SNS.
- DataIdentifier - Les données sensibles que la rubrique Amazon SNS doit analyser. Par exemple, nom, adresse ou numéro de téléphone.
- Principal - Le principal IAM qui est publié dans la rubrique ou qui y est abonné.
- Opération : l'action suivante, Audit (Auditer), De-identify (Anonymiser) (masquage ou suppression) ou Deny (Refuser) (blocage), que la rubrique Amazon SNS exécute lorsqu'elle trouve des données sensibles.

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

Propriétés JSON pour une opération d’instruction de politique

Une instruction de politique définit l'une des opérations de protection des données suivantes.

- Audit - Émet des métriques et des journaux de recherche sans interrompre la publication ni la transmission des messages.
- Anonymisation : masque ou supprime les données sensibles sans interrompre la publication des messages.
- Refuser - Bloque la demande de publication Amazon SNS ou fait échouer la transmission du message.

Comment puis-je déterminer les principaux IAM de ma politique de protection des données ?

Message Data Protection utilise deux principaux IAM qui interagissent avec Amazon SNS.

1. Publier le principal de l'API (entrant) - Le principal IAM authentifié appelant l'API `Publish` Amazon SNS.
2. Principal de l'abonnement (sortant) - Le principal IAM authentifié qui a appelé l'API `Subscribe` lors de la création d'un abonnement.

`SubscriptionPrincipal` est une propriété d'abonnement Amazon SNS accessible au public qui peut être récupérée depuis l'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"
  }
}
```

Opérations de la politique de protection des données

Voici des exemples de politiques de protection des données que vous pouvez utiliser pour auditer et refuser des données sensibles. Pour accéder à un didacticiel complet comprenant un exemple d'application, consultez le billet de blog Présentation de Message Data Protection pour Amazon SNS.

Rubriques

- Opération d'audit
- Opération d'anonymisation
- Refuser une opération

Opération d'audit

L'opération Audit réalise des échantillons de messages entrants de rubrique, et consigne les résultats relatifs aux données sensibles dans une destination AWS. La fréquence d'échantillonnage peut être un entier compris entre 0 et 99. Cette opération nécessite l'un des types de destinations de journalisation suivants :

1. FindingsDestination— La destination de journalisation lorsque la rubrique Amazon SNS trouve des données sensibles dans la charge utile.
2. NoFindingsDestination— La destination de journalisation lorsque la rubrique Amazon SNS ne trouve aucune donnée sensible dans la charge utile.

Vous pouvez utiliser les Services AWS suivants dans chacun des types de destination de journaux suivants :

- Amazon CloudWatch Logs (facultatif) — Le nom `LogGroup` doit figurer dans la région du sujet et commencer par /aws/vendedlogs/.
- Amazon Data Firehose (facultatif) — Le flux `DeliveryStream` doit figurer dans la région du sujet et indiquer Direct PUT comme source du flux de diffusion. Pour plus de détails, consultez les sections Source, Destination et Nom dans le manuel Amazon Data Firehose Developer Guide.
- Amazon S3 (facultatif) - Nom de compartiment Amazon S3. Des actions supplémentaires sont requises pour utiliser le compartiment Amazon S3 avec le chiffrement SSE-KMS activé.

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

Autorisations requises pour spécifier les destinations des journaux

Lorsque vous spécifiez les destinations de journalisation dans la politique de protection des données, vous devez ajouter les autorisations suivantes à la politique d'identité IAM du principal IAM qui appelle l'API `PutDataProtectionPolicy` Amazon SNS, ou l'API `CreateTopic` avec le paramètre `--data-protection-policy`.

Destination d'audit	Autorisation IAM
Par défaut	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery

Destination d'audit	Autorisation IAM
	logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Des actions supplémentaires sont requises pour utiliser le compartiment Amazon S3 avec le chiffrement SSE-KMS activé.

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

Politique de clé obligatoire à utiliser avec SSE-KMS

Si vous utilisez un compartiment Amazon S3 comme destination des journaux, vous pouvez protéger les données dans votre compartiment en activant le chiffrement côté serveur avec des clés gérées par Amazon S3 (SSE-S3) ou en activant le chiffrement côté serveur avec AWS KMS keys (SSE-KMS). Pour plus d'informations, consultez la section Protection des données à l'aide d'un chiffrement côté serveur dans le Guide de l'utilisateur d'Amazon S3.

Si vous choisissez l'option SSE-S3, aucune configuration supplémentaire n'est requise. Amazon S3 gère la clé de chiffrement.

Si vous choisissez l'option SSE-KMS, vous devez utiliser une clé gérée par le client. Vous devez mettre à jour la politique de votre clé gérée par le client afin que le compte de diffusion des journaux

puisse écrire des données dans votre compartiment S3. Pour plus d'informations sur la politique de clé requise pour une utilisation avec SSE-KMS, consultez le chiffrement côté serveur du compartiment Amazon S3 dans le guide de l'utilisateur Amazon CloudWatch Logs.

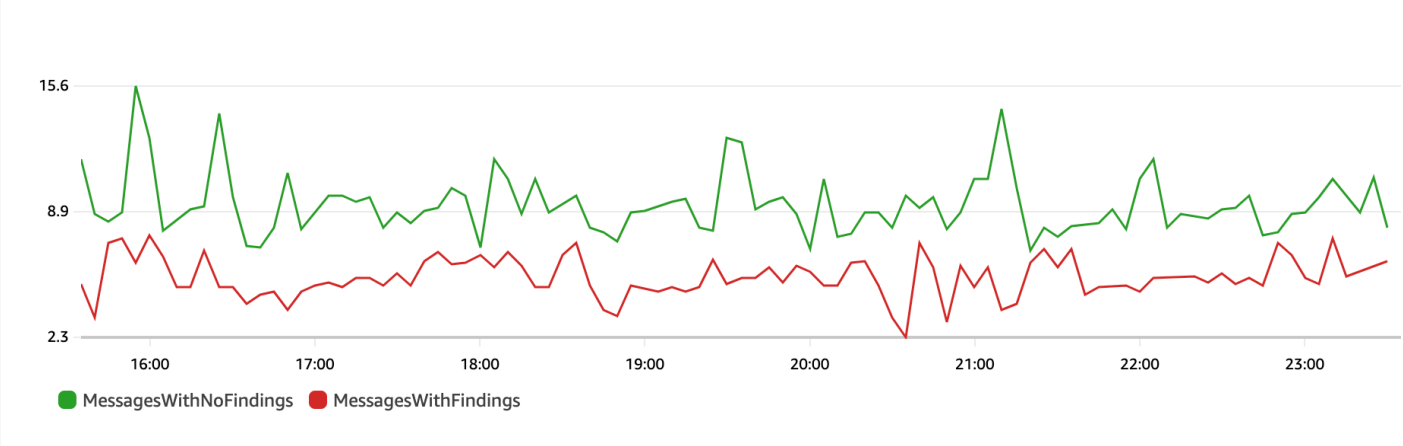
Exemple de journal de destination d'audit

Dans l'exemple suivant, `callerPrincipal` est utilisé pour identifier la source du contenu sensible et `messageID` est utilisé comme référence pour vérifier la conformité à la réponse d'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
        }
      ]
    }
  ]
}
```

Métriques de l’opération d'audit

Lorsqu'une opération d'audit a spécifié la propriété `FindingsDestination` ou la `NoFindingsDestination` propriété, les propriétaires des rubriques reçoivent également CloudWatch `MessagesWithFindings` des `MessagesWithNoFindings` métriques.



Opération d'anonymisation

L'opération De-identify (Anonymiser) masque ou supprime les données sensibles des messages publiés ou envoyés. Cette opération est disponible pour les messages entrants et sortants et nécessite l'un des types de configurations suivants :

- MaskConfig— Masque à l'aide d'un caractère compatible indiqué dans le tableau suivant. Par exemple, ssn : `123-45-6789` devient ssn : `###########`.

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

Caractère de masque pris en charge	Nom
*	Astérisque
A à Z, a à z et 0 à 9	Alphanumérique

Caractère de masque pris en charge	Nom
	Espace
!	Point d'exclamation
$	Symbole dollar
%	Signe pourcentage
&	Esperluette
()	Parenthèse
+	Signe plus
,	Virgule
-	Trait d'union
.	Période
/\	Barre oblique, barre oblique inverse
#	Signe numérique
:	Deux-points
;	Point-virgule
=, <>	Égal, inférieur ou supérieur à
@	Signe arobase
[]	Crochets
^	Symbole caret
_	Trait de soulignement
`	Accent grave

Caractère de masque pris en charge	Nom
|	Barre verticale
~	Symbole Tilde

- RedactConfig— Rédigez en supprimant complètement les données. Par exemple, ssn : `123-45-6789` devient ssn : .

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

Dans un message entrant, les données sensibles sont anonymisées après l'opération d'audit, et l'appelant d'API `SNS:Publish` reçoit l'erreur de paramètre non valide suivante lorsque l'ensemble du message est sensible.

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

Refuser une opération

L'opération Refuser interrompt soit la demande d'API `Publish`, soit la transmission du message s'il contient des données sensibles. L'objet de l'opération Refuser est vide, car il ne nécessite aucune configuration supplémentaire.

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

Dans un message entrant, l'appelant de l'API `SNS:Publish` reçoit une erreur d'autorisation.

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

Dans le cas d'un message sortant, la rubrique Amazon SNS ne transmet pas le message à l'abonnement. Pour suivre les transmissions non autorisées, activez la journalisation de l'état de transmission de la rubrique. Voici un exemple de journal de l'état de transmission :

```
{
    "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"
}
```

Exemples de politiques de protection des données

Voici des exemples de politiques de protection des données que vous pouvez utiliser pour auditer et refuser des données sensibles. Pour accéder à un didacticiel complet comprenant un exemple d'application, consultez le billet de blog Présentation de Message Data Protection pour Amazon SNS.

Rubriques

- Exemple de politique d'audit
- Exemple de politique avec instruction de masque et d'anonymisation entrante
- Exemple de politique avec instruction de suppression et d'anonymisation entrante
- Exemple de politique avec instruction de masque et d'anonymisation sortante
- Exemple de politique avec instruction de suppression et d'anonymisation sortante
- Exemple de politique avec instruction de refus entrant
- Exemple de politique avec instruction de refus sortant

Exemple de politique d'audit

Les politiques d'audit vous permettent d'auditer jusqu'à 99 % des messages entrants et d'envoyer les résultats à Amazon CloudWatch, Amazon Data Firehose et Amazon S3.

Par exemple, vous pouvez créer une politique d’audit afin de déterminer si l'un de vos systèmes envoie ou reçoit par inadvertance des données sensibles. Si les résultats de votre audit indiquent que les systèmes envoient des informations de carte de crédit à des systèmes qui n'en ont pas besoin, vous pouvez mettre en œuvre une politique de protection des données pour bloquer la transmission des données.

L'exemple suivant analyse 99 % des messages qui circulent dans le sujet en recherchant des numéros de carte de crédit et en envoyant les résultats à CloudWatch Logs, Firehose et Amazon S3.

Politique de protection des données :

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

Exemple de format des résultats d'audit :

```
{
    "messageId": "...",
    "callerPrincipal": "arn:aws:sts::123456789012:assumed-role/ExampleRole",
    "resourceArn": "arn:aws:sns:us-east-1:123456789012:ExampleArn",
    "dataIdentifiers": [
        {
            "name": "CreditCardNumber",
            "count": 1,
            "detections": [
                { "start": 1, "end": 2 }
            ]
        }
    ],
    "timestamp": "2021-04-20T00:33:40.241Z"
}
```

Exemple de politique avec instruction de masque et d'anonymisation entrante

L'exemple suivant empêche un utilisateur de publier dans une rubrique un message contenant `CreditCardNumber` en masquant les données sensibles du contenu du message.

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

```
  ]
}
```

Exemple de résultats de masque et d'anonymisation entrante :

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

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

Exemple de politique avec instruction de suppression et d'anonymisation entrante

L'exemple suivant empêche un utilisateur de publier dans une rubrique un message contenant `CreditCardNumber` en supprimant les données sensibles du contenu du message.

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

Exemple de résultats de suppression et d'anonymisation entrants :

```
// original message
```

```
My credit card number is 4539894458086459

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

Exemple de politique avec instruction de masque et d'anonymisation sortante

L'exemple suivant empêche un utilisateur de recevoir un message contenant `CreditCardNumber` en masquant les données sensibles du contenu du message.

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

Exemple de résultats de masque et d'anonymisation sortante :

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

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

Exemple de politique avec instruction de suppression et d'anonymisation sortante

L'exemple suivant empêche un utilisateur de recevoir un message contenant `CreditCardNumber` en supprimant les données sensibles du contenu du message.

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

Exemple de résultats de suppression et d'anonymisation sortants :

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

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

Exemple de politique avec instruction de refus entrant

L'exemple suivant empêche un utilisateur de publier un message dans une rubrique contenant `CreditCardNumber`. Les charges utiles refusées dans la réponse de l'API ont un code d'état « 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": {}
      }
    }
  ]
}
```

Exemple de politique avec instruction de refus sortant

L'exemple suivant empêche un compte AWS de recevoir des messages contenant `CreditCardNumber`.

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

Exemple de résultats de refus de sortie, connecté à 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"
}
```

Création de politiques de protection des données

Les politiques de protection des données vous aident à protéger les données publiées dans vos rubriques Amazon SNS en auditant, en désidentifiant (masquage ou suppression) et en refusant des informations sensibles qui se déplacent entre des applications ou des Services AWS. Vous pouvez utiliser l'API AWS, la AWS CLI, AWS CloudFormation ou la AWS Management Console pour créer des politiques de protection des données dans Amazon SNS. Une seule politique peut être définie par rubrique Amazon SNS. Chaque politique de protection des données peut comporter une ou plusieurs instructions de désidentification ou de refus, mais seulement une instruction d'audit.

Rubriques

- Création de politiques de protection des données pour sécuriser les données des messages (API)
- Création de politiques de protection des données pour sécuriser les données des messages (CLI)
- Création de politiques de protection des données pour sécuriser les données des messages (CloudFormation)
- Création de politiques de protection des données pour sécuriser les données des messages (console)
- Création de politiques de protection des données pour sécuriser les données des messages (SDK)

Création de politiques de protection des données pour sécuriser les données des messages (API)

Le nombre et la taille des ressources Amazon SNS dans un compte AWS sont limités. Pour plus d'informations, consultez la section Amazon Simple Notification Service points de terminaison et quotas.

Création de politiques de protection des données (API AWS)

Vous pouvez créer une politique de protection des données Amazon SNS à l'aide de l’API AWS.

Pour créer une politique de protection des données avec une rubrique Amazon SNS (API AWS)

Utilisez la propriété `DataProtectionPolicy` d'une rubrique Amazon SNS standard :

- `CreateTopic`

Pour récupérer ou créer une politique de protection des données pour une rubrique Amazon SNS existante (API AWS)

Appelez l'une des opérations suivantes :

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Création de politiques de protection des données pour sécuriser les données des messages (CLI)

Le nombre et la taille des ressources Amazon SNS dans un compte AWS sont limités. Pour plus d'informations, consultez la section Amazon Simple Notification Service points de terminaison et quotas.

Création de politiques de protection des données (AWS CLI)

Vous pouvez créer une politique de protection des données Amazon SNS à l'aide de la AWS Command Line Interface.

Pour créer une politique de protection des données avec une rubrique Amazon SNS (AWS CLI)

Utilisez cette option pour créer une politique de protection des données ainsi qu'une rubrique Amazon SNS standard :

- create-topic

Pour créer ou récupérer une politique de protection des données pour une rubrique Amazon SNS existante (AWS CLI)

Appelez l'une des opérations suivantes :

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

Création de politiques de protection des données pour sécuriser les données des messages (CloudFormation)

Le nombre et la taille des ressources Amazon SNS dans un compte AWS sont limités. Pour plus d'informations, consultez la section Amazon Simple Notification Service points de terminaison et quotas.

Création de politiques de protection des données (CloudFormation)

Vous pouvez créer une politique de protection des données Amazon SNS à l'aide de AWS CloudFormation.

Pour créer une politique de protection des données avec une rubrique Amazon SNS (CloudFormation)

Utilisez cette option pour créer une politique de protection des données ainsi qu'une rubrique Amazon SNS standard :

- AWS::SNS::Topic

Création de politiques de protection des données pour sécuriser les données des messages (console)

Le nombre et la taille des ressources Amazon SNS dans un compte AWS sont limités. Pour plus d'informations, consultez la section Amazon Simple Notification Service points de terminaison et quotas.

Pour créer une politique de protection des données ainsi qu'une rubrique Amazon SNS (console)

Cette option permet de créer une politique de protection des données ainsi qu'une rubrique Amazon SNS standard.

1. Connectez-vous à la console Amazon SNS.
2. Choisissez une rubrique ou créez-en une. Pour plus d'informations sur la création de rubriques, consultez Création d'une rubrique Amazon SNS.
3. Sur la page Create topic (Création d'une rubrique), dans la section Details (Détails), choisissez Standard.

   a. Entrez un Nom pour la rubrique.

   b. (Facultatif) Entrez un Nom d'affichage pour votre rubrique.
4. Développez Data protection policy (Politique de protection des données).
5. Choisissez un Configuration mode (Mode de configuration) :

   - Basic (Basique) : définissez une politique de protection des données à l'aide d'un menu simple.

   - Advanced (Avancé) : définissez une politique de protection des données personnalisée à l'aide de JSON.
6. (Facultatif) Pour créer votre propre identifiant de données personnalisé, développez la section Configuration d'un identifiant de données personnalisé et procédez comme suit :

   a. Attribuez un nom unique à l'identificateur de données personnalisé. Les noms d'identifiants de données personnalisés prennent en charge les caractères alphanumériques, le trait de soulignement (_) et le tiret (-). La prise en charge se limite à 128 caractères. Ce nom ne peut pas être identique à un identifiant de données géré. Pour obtenir la liste complète des limitations des identifiants de données personnalisés, consultez Contraintes liées aux identifiants de données personnalisés.

   b. Entrez une expression régulière (RegEx) pour l'identifiant de données personnalisé. RegExprend en charge les caractères alphanumériques, les caractères RegEx réservés et les symboles. RegEx a une longueur maximale de 200 caractères. Si RegEx c'est trop compliqué, Amazon SNS échouera à l'appel d'API. Pour une liste complète des RegEx limitations, voirContraintes liées aux identifiants de données personnalisés.

   c. (Facultatif) Sélectionnez Ajouter un identifiant de données personnalisé pour ajouter des identifiants de données supplémentaires, si nécessaire. Le nombre d'identifiants de données

personnalisés pris en charge pour chaque politique de protection des données est limité à 10.

7. Choisissez la ou les instructions que vous souhaitez ajouter à votre politique de protection des données. Vous pouvez ajouter les types de déclarations audit (auditer), de-identify (anonymiser) (masquage ou suppression) et deny (refuser) (blocage) à la même politique de protection des données.

   a. Add audit statement (Ajouter une instruction d'audit) : configurez les données sensibles à auditer, le pourcentage de messages que vous souhaitez auditer pour ces données et l'endroit où envoyer les journaux d'audit.

      > **Note**
      >
      > Une seule instruction d'audit est autorisée par politique ou rubrique de protection des données.

      i. Sélectionnez les identifiants des données pour définir les données sensibles que vous souhaitez auditer.

      ii. Pour Audit sample rate (Taux d'échantillonnage d'audit), entrez le pourcentage de messages à auditer pour détecter des informations sensibles, jusqu'à un maximum de 99 %.

      iii. Pour Audit destination (Destination d'audit), sélectionnez les Services AWS auxquels envoyer les résultats de l'audit, et saisissez un nom de destination pour chaque Service AWS que vous utilisez. Vous pouvez faire votre choix parmi les Amazon Web Services suivants :

      - Amazon CloudWatch — CloudWatch Logs est la solution de journalisation AWS standard. À l'aide de CloudWatch Logs, vous pouvez effectuer des analyses de journaux à l'aide de Logs Insights (voir des exemples ici) et créer des métriques et des alarmes. CloudWatch Les journaux sont l'endroit où de nombreux services publient des journaux, ce qui facilite l'agrégation de tous les journaux à l'aide d'une seule solution. Pour plus d'informations sur Amazon CloudWatch, consultez le guide de CloudWatch l'utilisateur Amazon.
      - Amazon Data Firehose — Firehose répond aux demandes de diffusion en temps réel vers Splunk, et OpenSearch Amazon Redshift pour des analyses supplémentaires

des journaux. Pour plus d'informations sur Amazon Data Firehose, consultez le guide de l'utilisateur d'Amazon Data Firehose.

- Amazon Simple Storage Service : Amazon S3 est une destination économique pour les journaux à des fins d'archivage. Il se peut que vous deviez conserver des journaux pendant plusieurs années. Dans ce cas, vous pouvez les placer dans Amazon S3 pour réduire les coûts. Pour plus d'informations sur Amazon Simple Storage Service, consultez le Guide de l'utilisateur Amazon Simple Storage Service.

b. Add a de-identify statement (Ajouter une déclaration d'anonymisation) : configurez les données sensibles que vous souhaitez anonymiser dans le message, que vous souhaitiez masquer ou supprimer ces données, et les comptes pour arrêter la diffusion de ces données.

   i. Pour Data identifiers (Identifiants de données), sélectionnez les données sensibles que vous souhaitez anonymiser.

   ii. Pour Define this de-identify statement for (Définir cette déclaration d'anonymisation pour), sélectionnez les comptes AWS ou les principaux IAM auxquels cette déclaration d'anonymisation s'applique. Vous pouvez l'appliquer à tous les comptes AWS, à des comptes AWSspécifiques ou à des entités IAM (racines de compte, rôles ou utilisateurs) qui utilisent des ID de compte ou des ARN d'entités IAM. Séparez les différents ID ou ARN par une virgule (,).

   Les principaux IAM suivants sont pris en charge :

   - IAM account principals (Principaux du compte IAM) : par exemple, `arn:aws:iam::AWS-account-ID:root`.
   - IAM role principals (Principaux du rôle IAM) : par exemple, `arn:aws:iam::AWS-account-ID:role/role-name`.
   - IAM user principals (Principaux de l'utilisateur IAM) : par exemple, `arn:aws:iam::AWS-account-ID:user/user-name`.

   iii. Pour De-identify Option (Option d'anonymisation), sélectionnez les données sensibles que vous souhaitez anonymiser. Les options suivantes sont prises en charge :

   - Redact (Supprimer) : supprime complètement les données. Par exemple, l'e-mail : `classified@amazon.com` devient l'e-mail : .
   - Mask (Masquer) : remplace les données par des caractères uniques. Par exemple, l'e-mail : `classified@amazon.com` devient l'e-mail : `**********`.

iv. (Facultatif) Continuez à ajouter des instructions d'anonymisation si nécessaire.

c. Add deny statement (Ajouter une instruction de refus) : configurez les données sensibles dont vous souhaitez empêcher la circulation dans votre rubrique et les principaux qui ne doivent pas fournir ces données.

i. Pour Data direction (Direction des données), choisissez la direction des messages pour l'instruction de refus :

- Inbound messages (Messages entrants) : appliquez cette instruction de refus aux messages envoyés à la rubrique.
- Outbound messages (Messages sortants) : appliquez cette instruction de refus aux messages que la rubrique envoie aux points de terminaison d'abonnement.

ii. Sélectionnez les identifiants des données pour définir les données sensibles que vous souhaitez refuser.

iii. Sélectionnez les principaux IAM auxquels s'applique cette instruction de refus. Vous pouvez l'appliquer à tous les comptes AWS à des Comptes AWS spécifiques ou à des entités IAM (par exemple, des racines de compte, rôles ou utilisateurs) qui utilisent des ID de compte ou des ARN d'entités IAM. Séparez les différents ID ou ARN par une virgule (,). Les principaux IAM suivants sont pris en charge :

- IAM account principals (Principaux du compte IAM) : par exemple, `arn:aws:iam::AWS-account-ID:root`.
- IAM role principals (Principaux du rôle IAM) : par exemple, `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM user principals (Principaux de l'utilisateur IAM) : par exemple, `arn:aws:iam::AWS-account-ID:user/user-name`.

iv. (Facultatif) Continuez à ajouter des instructions de refus si nécessaire.

Création de politiques de protection des données pour sécuriser les données des messages (SDK)

Le nombre et la taille des ressources Amazon SNS dans un compte AWS sont limités. Pour plus d'informations, consultez la section Amazon Simple Notification Service points de terminaison et quotas.

Création de politiques de protection des données (SDK AWS)

Vous pouvez créer une politique de protection des données Amazon SNS à l'aide du SDK AWS.

Pour créer une politique de protection des données avec une rubrique Amazon SNS (SDK AWS)

Utilisez les options suivantes pour créer une politique de protection des données ainsi qu'une rubrique Amazon SNS standard :

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();
```

Pour créer ou récupérer une politique de protection des données pour une rubrique Amazon SNS existante (SDK AWS)

Utilisez les options suivantes pour créer ou récupérer une politique de protection des données ainsi qu'une rubrique Amazon SNS standard :

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();
```

Suppression de politiques de protection des données dans Amazon SNS

Vous pouvez supprimer des politiques de protection des données Amazon SNS avec l'API AWS, la AWS CLI, AWS CloudFormation ou la AWS Management Console.

Pour obtenir des informations générales sur les politiques de protection des données Amazon SNS, consultez Comprendre les politiques de protection des données.

Le nombre et la taille des ressources de règles de protection des données Amazon SNS dans un compte AWS sont limités. Pour plus d'informations, consultez Limitation des API Amazon SNS dans la Références générales AWS.

Rubriques

- Suppression de politiques de protection des données (console)
- Suppression d'une politique de protection des données à l'aide d'une chaîne JSON vide
- Suppression d'une politique de protection des données à l'aide de la AWS CLI

Suppression de politiques de protection des données (console)

Pour supprimer une politique de protection des données gérée (console)

1. Connectez-vous à la console Amazon SNS.
2. Choisissez la rubrique qui contient la politique de protection des données que vous souhaitez supprimer.
3. Choisissez Edit (Modifier).
4. Développez la section Data protection policy (Politique de protection des données).
5. Choisissez Remove (Supprimer), en regard de l'instruction de politique de protection des données que vous souhaitez supprimer.
6. Sélectionnez Save Changes (Enregistrer les modifications).

Suppression d’une politique de protection des données à l'aide d'une chaîne JSON vide

Vous pouvez supprimer une politique de protection des données en la mettant à jour avec une chaîne JSON vide.

Suppression d’une politique de protection des données à l'aide de la AWS CLI

Vous pouvez supprimer une politique de protection des données à l'aide de la AWS CLI.

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

identifiants des données

Amazon SNS utilise une combinaison de critères et de techniques, notamment le machine learning et la correspondance de modèles, pour détecter les données sensibles. Ces critères et techniques, généralement appelés identifiants de données, peuvent détecter une longue liste croissante de types de données sensibles pour de nombreux pays et régions. Les identifiants de données gérés Amazon SNS proposent des types de données préconfigurés pour protéger les données financières, les informations personnelles sur la santé (PHI) et les informations d'identification personnelle (PII). Vous pouvez également utiliser des identifiants de données personnalisés pour créer vos propres identifiants de données adaptés à votre cas d'utilisation spécifique.

Rubriques

Utilisation des identifiants de données gérés dans Amazon SNS

Rubriques

Qu'entend-on par identifiants de données gérés ?

Les identifiants de données gérés Amazon SNS sont conçus pour détecter un type spécifique de données sensibles, comme les numéros de carte de crédit, les clés d'accès secrètes AWS ou les numéros de passeport pour un pays ou une région en particulier. Lorsque vous créez une politique de protection des données, vous pouvez configurer Amazon SNS pour qu'il utilise ces identifiants afin d'analyser les messages traversant la rubrique, et prendre des mesures lorsqu'ils sont détectés.

Amazon SNS peut détecter les catégories de données sensibles suivantes à l'aide d'identifiants de données gérés :

- Les informations d'identification, telles que des clés privées ou des clés d'accès secrètes AWS
- Les identifiants d'appareil, tels que l'adresse IP ou l'adresse MAC
- Les informations financières, telles que les numéros de carte de crédit
- Les informations de santé, pour les PHI comme les numéros d'assurance maladie ou d'identification médicale
- Les informations personnelles, pour les informations personnelles telles que les permis de conduire ou les numéros de sécurité sociale

Au sein de chaque catégorie, Amazon SNS peut détecter plusieurs types de données sensibles. Les rubriques de cette section répertorient et décrivent chaque type ainsi que toutes les exigences pertinentes pour le détecter. Pour chaque type, elles indiquent également l'identifiant unique (ID) de l'identifiant de données gérées conçu pour détecter les données. Lorsque vous créez une politique de protection des données, vous pouvez utiliser cet ID pour inclure l'identifiant de données gérées à détecter par Message Data Protection.

Exigences relatives aux mots-clés

Pour détecter certains types de données sensibles, Amazon SNS analyse les mots clés à proximité des données. Si cela s’applique à un type de données particulier, une rubrique suivante de cette section indique les exigences spécifiques en matière de mots clés pour ces données.

Les mots clés ne sont pas sensibles à la casse. En outre, si un mot clé contient une espace, Amazon SNS met automatiquement en correspondance les variantes de mots clés qui ne contiennent pas l’espace, ou qui contiennent un trait de soulignement (_) ou un trait d'union (-) à la place de l'espace. Dans certains cas, Amazon SNS développe ou abrège également un mot clé afin de prendre en compte ses variantes courantes.

Identifiants de données gérés par Amazon SNS pour les types de données sensibles

Le tableau ci-dessous répertorie et décrit les types d'informations d’identification, d’informations sur les appareils, financières, médicales et sur la santé personnelles (PHI) qu'Amazon SNS peut détecter à l'aide d'identifiants de données gérés. Ces données s'ajoutent à certains types de données qui pourraient également être considérés comme des données d'identification personnelle (PII).

Les identifiants de données dépendants de la région nécessitent le nom de l'identifiant avec un tiret et les codes à deux lettres (ISO 3166-1 alpha-2). Par exemple, DriversLicense -US.

Identifiant	Catégorie	Pays/Langues
BankAccountNumber	Services financiers	DE, ES, FR, GB, IT
CepCode	Personnel	BR
Cnpj	Personnel	BR
CpfCode	Personnel	BR
DriversLicense	Personnel	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB,

Identifiant	Catégorie	Pays/Langues
		GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Santé	ETATS-UNIS
ElectoralRollNumber	Personnel	Go
HealthInsuranceCardNumber	Santé	UE
HealthInsuranceClaimNumber	Santé	ETATS-UNIS
HealthInsuranceNumber	Santé	FR
HealthcareProcedureCode	Santé	ETATS-UNIS
IndividualTaxIdentificationNumber	Personnel	ETATS-UNIS
InseeCode	Personnel	FR
MedicareBeneficiaryNumber	Santé	ETATS-UNIS
NationalDrugCode	Santé	ETATS-UNIS
NationalIdentificationNumber	Personnel	DE, ES, IT
NationalInsuranceNumber	Personnel	Go
NationalProviderId	Santé	ETATS-UNIS
NhsNumber	Santé	Go
NieNumber	Personnel	ES
NifNumber	Personnel	ES
PassportNumber	Personnel	CA, DE, ES, FR, GB, IT, US

Identifiant	Catégorie	Pays/Langues
PermanentResidenceNumber	Personnel	CA
PersonalHealthNumber	Santé	CA
PhoneNumber	Personnel	BR, DE, ES, FR, GB, IT, US
PostalCode	Personnel	CA
RgNumber	Personnel	BR
SocialInsuranceNumber	Personnel	CA
Ssn	Personnel	ES, US
TaxId	Personnel	DE, ES, FR, GB
ZipCode	Personnel	ETATS-UNIS

Identifiants pris en charge indépendants de la langue/de la région

Identifiant	Catégorie
Address	Personnel
AwsSecretKey	Informations d’identification
CreditCardExpiration	Services financiers
CreditCardNumber	Services financiers
CreditCardSecurityCode	Services financiers
EmailAddress	Personnel
IpAddress	Personnel
LatLong	Personnel

Identifiant	Catégorie
Nom	Personnel
OpenSshPrivateKey	Informations d’identification
PgpPrivateKey	Informations d’identification
PkcsPrivateKey	Informations d’identification
PuttyPrivateKey	Informations d’identification
VehicleIdentificationNumber	Personnel

Types de données sensibles : informations d'identification

Le tableau ci-dessous répertorie et décrit les types d'informations d'identification qu'Amazon SNS peut détecter à l'aide d'identifiants de données gérés.

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Pays et régions
Clé d'accès secrète AWS	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	N’importe quel compte
Clé privée OpenSSH	OpenSshPrivateKey	Non	N’importe quel compte
Clé privée PGP	PgpPrivateKey	Non	N’importe quel compte
Clé privée PKCS (Public Key Cryptography Standard)	PkcsPrivateKey	Non	N’importe quel compte

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Pays et régions
Clé privée PuTTY	PuttyPrivateKey	Non	N’importe quel compte

ARN d'identifiant de données pour les types de données d'information d’identification

La liste suivante répertorie les noms Amazon Resource Names (ARN) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

Données d'informations d'identification : ARN d'identifiant
arn:aws:dataprotection : :aws:data-identifier/ AwsSecretKey
arn:aws:dataprotection : :aws:data-identifier/ OpenSshPrivateKey
arn:aws:dataprotection : :aws:data-identifier/ PgpPrivateKey
arn:aws:dataprotection : :aws:data-identifier/ PkcsPrivateKey
arn:aws:dataprotection : :aws:data-identifier/ PuttyPrivateKey

Types de données sensibles : appareils

Le tableau ci-dessous répertorie et décrit les types d'identifiants d’appareils qu'Amazon SNS peut détecter à l'aide d'identifiants de données gérés.

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Pays et régions
Adresse IP	IpAddress	Non	N’importe quel compte

ARN d'identifiant de données pour les types de données d’appareils

La liste suivante répertorie les noms Amazon Resource Names (ARN) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

ARN d'identifiant de données d'appareil
arn:aws:dataprotection : :aws:data-identifier/ IpAddress

Types de données sensibles : financières

Le tableau ci-dessous répertorie et décrit les types d'informations financières qu'Amazon SNS peut détecter à l'aide d'identifiants de données gérés.

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro de compte bancaire	BankAccountNumber BankAccountNumber-US	Oui, consultez Mots clés pour les numéros de compte bancaire.	Ces informations comprennent les éléments suivants : numéros de comptes bancaires internationaux (IBAN) comprenant jusqu'à 34 caractères alphanumériques, y compris des éléments tels qu'un code pays.	France, Allemagne, Italie, Espagne, Royaume-Uni

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Date d'expiration de carte de crédit	CreditCardExpiration	j exp, m exp, a exp, expiration	–	N’importe quel compte
Données de cartes de crédit à bande magnétique	CreditCardMagneticStripe	Oui, y compris : données de la carte, ISO7813, mag, bande magnétique, bande magnétique, swipe.	Cela inclut les pistes 1 et 2.	N’importe quel compte

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro de carte de crédit	CreditCardNumber	numéro de compte, american express, amex, carte bancaire, carte, num. de carte, numéro de carte, n° cc, ncc, chèque, crédit, numéro de carte de crédit, dankort, débit, carte de débit, diners club, discover, electron, code de vérification elo, japanese card bureau, jcb, mastercard, mc, pan, numéro de compte de paiement, numéro de carte de paiement, pcn, union pay, visa	La détection nécessite que les données soient une séquence de 13 à 19 chiffres conforme à la formule de vérification de Luhn et utilise un préfixe de numéro de carte standard pour les types de cartes de crédit suivants : American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card Bureau (JCB) UnionPay, Mastercard et Visa (lien en exposant ci-dessous 1).	N'importe quel compte

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Code de vérification de carte de crédit	CreditCardSecurityCode	identifiant de carte, code d'identification de carte, numéro d'identification de carte, code de sécurité de carte, code de validation de carte, numéro de validation de carte, données de vérification de carte, valeur de vérification de carte, cvc, cvc2, cvv, cvv2, code de vérification elo	–	N'importe quel compte

1. Amazon SNS ne signale pas les occurrences des séquences suivantes, que les émetteurs de cartes de crédit ont réservées aux tests publics :

   122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017, 5204740009900014, 5420923878724339, 5454545454545454, 5455330760000018,

5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194, 5555555555554444, 5610591081018250, 6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441, 630495060000000000, 6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.

Mots clés pour les numéros de compte bancaire

Utilisez les mots clés suivants pour détecter les numéros de comptes bancaires internationaux (IBAN) comprenant jusqu'à 34 caractères alphanumériques, y compris des éléments tels qu'un code pays.

Pays ou région	Mots clés			
France	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			
Allemagne	account code, account number, accountno #, accountnumber#, bankleitzahl, bban, customer account id,			

Pays ou région	Mots clés			
	customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
Italie	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			

Pays ou région	Mots clés			
Espagne	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			
Royaume-Uni	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

Pays ou région	Mots clés			
ETATS-UNIS	compte bancaire, cpte bancaire, compte courant, cpte courant, compte de dépôt, cpte de dépôt, compte d'épargne, cpte d'épargne, compte chèques, cpte chèques			

ARN d'identifiant de données pour les types de données financières

La liste suivante répertorie les noms Amazon Resource Names (ARN) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

ARN d'identifiant de données financières
arn:aws:dataprotection : :aws:data-identifier/ -DE BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -GB BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -IT BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -US BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ CreditCardExpiration
arn:aws:dataprotection : :aws:data-identifier/ CreditCardNumber
arn:aws:dataprotection : :aws:data-identifier/ CreditCardSecurityCode

Types de données sensibles : informations protégées sur la santé (PHI)

Le tableau ci-dessous répertorie et décrit les types d'informations protégées sur la santé (PHI) qu'Amazon SNS peut détecter à l'aide d'identifiants de données gérés.

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Pays et régions
Numéro d'enregistrement de DEA (Drug Enforcement Agency)	DrugEnforcementAgencyNumber	dea number, dea registration	ETATS-UNIS
Numéro de carte de sécurité sociale (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance,	UE

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Pays et régions
		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	
Numéro de règlement de sécurité sociale (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	ETATS-UNIS
Numéro d'assurance maladie ou d'identification médicale	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR
Code du système de codage des procédures communes pour les soins de santé (HCPCS)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	ETATS-UNIS

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Pays et régions
Numéro de bénéficiaire Medicare (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	ETATS-UNIS
Code national des médicaments (NCD)	NationalDrugCode	national drug code, ndc	ETATS-UNIS
Identifiant de fournisseur national (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	ETATS-UNIS
Numéro du service national de santé (NHS)	NhsNumber	national health service, NHS	Go
Numéro de santé personnel (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Mots clés pour les numéros d'assurance maladie et d'identification médicale

Pour détecter différents types de numéros d'assurance maladie et d'identification médicale, Amazon SNS exige qu'un mot clé se trouve à proximité des numéros. Cela comprend les numéros de carte européenne d'assurance maladie (UE, Finlande), les numéros de sécurité sociale (France), les numéros de bénéficiaire Medicare (États-Unis), les numéros d'assurance nationale (Royaume-Uni), les numéros NHS (Royaume-Uni) et les numéros de santé personnels (Canada).

Le tableau suivant répertorie les mots clés reconnus par Amazon SNS pour des pays et des régions spécifiques.

Pays ou région	Mots clés
Canada	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
Finlande	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti

Pays ou région	Mots clés
France	carte d'assuré social, carte vitale, insurance card
Royaume-Uni	service national de santé, NHS
ETATS-UNIS	mbi, medicare beneficiary

ARN d’identifiant de données pour les types de données d'informations protégées sur la santé (PHI)

La liste suivante répertorie les noms Amazon Resource Names (ARN) que vous pouvez utiliser dans les politiques de protection des données PHI.

ARN d’identifiant de données PHI
arn:aws:dataprotection : :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:dataprotection : :aws:data-identifier/ -US HealthcareProcedureCode
arn:aws:dataprotection : :aws:data-identifier/ -EU HealthInsuranceCardNumber
arn:aws:dataprotection : :aws:data-identifier/ -US HealthInsuranceClaimNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR HealthInsuranceNumber
arn:aws:dataprotection : :aws:data-identifier/ -US MedicareBeneficiaryNumber
arn:aws:dataprotection : :aws:data-identifier/ -US NationalDrugCode
arn:aws:dataprotection : :aws:data-identifier/ -GB NationalInsuranceNumber
arn:aws:dataprotection : :aws:data-identifier/ -US NationalProviderId
arn:aws:dataprotection : :aws:data-identifier/ -GB NhsNumber
arn:aws:dataprotection : :aws:data-identifier/ -CA PersonalHealthNumber

Types de données sensibles : données d'identification personnelle (PII)

Le tableau ci-dessous répertorie et décrit les types de données d'identification personnelle (PII) qu'Amazon SNS peut détecter à l'aide d'identifiants de données gérés.

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Date de naissance	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	Cela inclut la plupart des formats de date, tels que les formats avec des chiffres ainsi que les combinaisons de chiffres et de noms de mois. Les composants de date peuvent être séparés par des espaces, des barres obliques (/) ou des traits d'union (-).	N’importe quel compte
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brésil
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da	–	Brésil

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
		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	–	Brésil

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro d'identification du permis de conduire	DriversLicense	Oui, consultez Mots clés pour les numéros d'identification du permis de conduire.	–	Australie, Autriche, Belgique, Bulgarie, Canada, Croatie, Chypre, République tchèque, Danemark, Estonie, Finlande, France, Allemagne, Grèce, Hongrie, Irlande, Italie, Lettonie, Lituanie, Luxembourg, Malte, Pays-Bas, Pologne, Portugal, Roumanie, Slovaquie, Slovénie, Espagne, Suède, Royaume-Uni, États-Unis

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro de liste électorale	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	Royaume-Uni
Identification individuelle de contribuable	IndividualTaxIdentification Number	Oui, consultez Mots clés pour les numéros d'identification et de référence des contribuables.	–	ETATS-UNIS
Institut national de la statistique et des études économiques (INSEE)	InseeCode	Oui, consultez Mots clés pour les numéros d'identification nationaux.	–	France

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro d'identification nationale	NationalIdentificationNumber	Oui, consultez Mots clés pour les numéros d'identification nationaux.	Cela inclut les identifiants Documento Nacional de Identidad (DNI) (Espagne), les codes Codice fiscale (Italie) et les numéros de carte d'identité nationale (Allemagne).	Allemagne, Italie, Espagne
Numéro d'assurance nationale (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	Royaume-Uni
Número de identidad de extranjero (NIE)	NieNumber	Oui, consultez Mots clés pour les numéros d'identification et de référence des contribuables.	–	Espagne

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Número de Identificación Fiscal (NIF)	NifNumber	Oui, consultez Mots clés pour les numéros d'identification et de référence des contribuables.	–	Espagne
Numéro de passeport	PassportNumber	Oui, consultez Mots clés pour les numéros de passeport.	–	Canada, France, Allemagne, Italie, Espagne, Royaume-Uni, États-Unis

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro de résidence 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	–	Canada

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Phone number (Numéro de téléphone)	PhoneNumber	Brésil : les mots clés incluent également : cel, celular, fone, móvel, número residencial, numero residencial, telefone Autres : portable, contact, fax, numéro de fax, téléphone portable, téléphone, numéro de téléphone, tel, téléphone, n° de téléphone	Cela inclut les numéros gratuits aux États-Unis et les numéros de télécopieur. Si un mot clé se trouve à proximité des données, il n'est pas nécessaire que le numéro inclue un code de pays. Si un mot clé ne se trouve pas à proximité des données, il est nécessaire que le numéro inclue un code de pays.	Brésil, Canada, France, Allemagne, Italie, Espagne, Royaume-Uni, États-Unis
Code postal	PostalCode	No	–	Canada
Registro Geral (RG)	RgNumber	Oui, consultez Mots clés pour les numéros d'identification nationaux.	–	Brésil

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Numéro de sécurité sociale	SocialIns uranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Canada
Numéro de sécurité sociale	Ssn	Espagne : número de la seguridad social, n° de sécurité sociale, numéro de sécurité sociale, número de la seguridad social, numéro de sécurité sociale, socialsec urityno#, ssn, ssn# États-Unis : social security, ss#, ssn	–	Espagne, États-Unis
Numéro d'identif ication ou de référence du contribuable	TaxId	Oui, consultez Mots clés pour les numéros d'identification et de référence des contribuables.	Cela inclut : TIN (France) ; Steueridentifikati onsnummer (Allemagne) ; CIF (Espagne) ; et TRN, UTR (Royaume-Uni).	France, Allemagne , Espagne, Royaume-Uni

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Code postal américain	ZipCode	zip code, zip+4	–	ETATS-UNIS
Adresse postale	Address	No	Bien qu'aucun mot-clé ne soit requis, la détection nécessite que l'adresse comprenne le nom d'une ville ou d'un lieu ainsi qu’un code postal.	Australie, Canada, France, Allemagne, Italie, Espagne, Royaume-Uni, États-Unis
Adresse électronique	EmailAddress	e-mail, adresse e-mail, email, adresse email	–	N’importe quel compte

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Coordonnées du système de positionnement mondial (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS peut détecter les coordonnées GPS si les coordonnées de latitude et de longitude sont stockées sous forme de paire et si elles sont au format Degrés décimaux (DD), par exemple, 41.948614 ,-87.655311. La prise en charge n'inclut pas les coordonnées au format Degrés décimaux minutes (DDM), par exemple 41°56.9168'N 87°39.3187'W, ni au format Degrés, minutes, secondes (DMS), par exemple 41°56'55.0104"N 87°39'19. 1196"W.	N'importe quel compte

Type de détection	ID d’identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
Nom complet	Name	No	Amazon SNS ne peut détecter que les noms complets. Le support est limité aux jeux de caractères latins.	N’importe quel compte
Numéro d'identification de véhicule (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 peut détecter les VIN qui se composent d'une séquence de 17 caractères et qui respectent les normes ISO 3779 et 3780. Ces normes ont été conçues pour être utilisées dans le monde entier.	N’importe quel compte

Mots clés pour les numéros d'identification du permis de conduire

Pour détecter différents types de numéros d’identification du permis de conduire, Amazon SNS exige qu'un mot clé se trouve à proximité des numéros. Le tableau suivant répertorie les mots clés reconnus par Amazon SNS pour des pays et des régions spécifiques.

Pays ou région	Mots clés
Australie	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
Autriche	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Belgique	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgarie	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Canada	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit, permis de conduire
Croatie	vozačka dozvola
Chypre	άδεια οδήγησης

Pays ou région	Mots clés
République tchèque	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
Danemark	kørekort, kørekortnummer
Estonie	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finlande	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
France	permis de conduire
Allemagne	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Grèce	δεια οδήγησης, adeia odigisis
Hongrie	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Irlande	ceadúnas tiomána
Italie	patente di guida, patente di guida numero, patente guida, patente guida numero
Lettonie	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Lituanie	vairuotojo pažymėjimas

Pays ou région	Mots clés
Luxembourg	fahrerlaubnis, führerschäin
Malte	liċenzja tas-sewqan
Pays-Bas	permis de conduire, rijbewijs, rijbewijsnummer
Pologne	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
Roumanie	numărul permisului de conducere, permis de conducere
Slovaquie	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Slovénie	vozniško dovoljenje
Espagne	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
Suède	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

Pays ou région	Mots clés
Royaume-Uni	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
ETATS-UNIS	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

Mots clés pour les numéros d'identification nationaux

Pour détecter différents types de numéros d’identification nationaux, Amazon SNS exige qu'un mot clé se trouve à proximité des numéros. Cela inclut les identifiants Documento Nacional de Identidad (DNI) (Espagne), les codes de l'Institut national de la statistique et des études économiques (INSEE), les numéros de carte nationale d'identité allemande (Allemagne) et les numéros du Registro Geral (RG) (Brésil).

Le tableau suivant répertorie les mots clés reconnus par Amazon SNS pour des pays et des régions spécifiques.

Pays ou région	Mots clés
Brésil	registro geral, rg
France	assurance sociale, carte nationale d’identité, cni, code sécurité sociale, French social

Pays ou région	Mots clés
	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#
Allemagne	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italie	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Espagne	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#

Mots clés pour les numéros de passeport

Pour détecter différents types de numéros de passeport, Amazon SNS exige qu'un mot clé se trouve à proximité des numéros. Le tableau suivant répertorie les mots clés reconnus par Amazon SNS pour des pays et des régions spécifiques.

Pays ou région	Mots clés
Canada	passeport, passeport#, passport, passport#, passportno, passportno#

Pays ou région	Mots clés
France	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
Allemagne	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Italie	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Espagne	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
Royaume-Uni	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
ETATS-UNIS	passport, travel document

Mots clés pour les numéros d'identification et de référence des contribuables

Pour détecter différents types de numéros d’identification et de référence des contribuables, Amazon SNS exige qu'un mot clé se trouve à proximité des numéros. Le tableau suivant répertorie les mots clés reconnus par Amazon SNS pour des pays et des régions spécifiques.

Pays ou région	Mots clés
Brésil	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa

Pays ou région	Mots clés
	jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
France	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Allemagne	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
Espagne	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#
Royaume-Uni	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
ETATS-UNIS	numéro individuel d'identification de contribua ble, itin, i.t.i.n.

ARN d'identifiant de données pour les données d'identification personnelle (PII)

Le tableau suivant répertorie les noms Amazon Resource Names (ARN) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

ARN d’identifiant de données PII
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection : :aws:data-identifier/ -BR CepCode

ARN d’identifiant de données PII
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:dataprotection : :aws:data-identifier/ -BR CpfCode
arn:aws:dataprotection : :aws:data-identifier/ DateOfBirth
arn:aws:dataprotection : :aws:data-identifier/ -AT DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -AU DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -BE DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -BG DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -CA DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -CY DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -CZ DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -DE DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -DK DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -EE DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -ES DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -FI DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -FR DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -GB DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -GR DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -HR DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -HU DriversLicense

ARN d’identifiant de données PII
arn:aws:dataprotection : :aws:data-identifier/ -IE DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -IT DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -LT DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -LU DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -LV DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -MT DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -FR DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -PL DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -PT DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -RO DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -SE DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -SI DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -SK DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -US DriversLicense
arn:aws:dataprotection : :aws:data-identifier/ -GB ElectoralRollNumber
arn:aws:dataprotection : :aws:data-identifier/ EmailAddress
arn:aws:dataprotection : :aws:data-identifier/ -US IndividualTaxIdentificationNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR InseeCode
arn:aws:dataprotection : :aws:data-identifier/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name

ARN d’identifiant de données PII
arn:aws:dataprotection : :aws:data-identifier/ -DE NationalIdentificationNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES NationalIdentificationNumber
arn:aws:dataprotection : :aws:data-identifier/ -IT NationalIdentificationNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES NieNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES NifNumber
arn:aws:dataprotection : :aws:data-identifier/ -CA PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -DE PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -GB PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -IT PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -US PassportNumber
arn:aws:dataprotection : :aws:data-identifier/ -CA PermanentResidenceNumber
arn:aws:dataprotection : :aws:data-identifier/ -BR PhoneNumber
arn:aws:dataprotection : :aws:data-identifier/ -DE PhoneNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES PhoneNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR PhoneNumber
arn:aws:dataprotection : :aws:data-identifier/ -GB PhoneNumber
arn:aws:dataprotection : :aws:data-identifier/ -IT PhoneNumber
arn:aws:dataprotection : :aws:data-identifier/ -US PhoneNumber

ARN d’identifiant de données PII
arn:aws:dataprotection : :aws:data-identifier/ -CA PostalCode
arn:aws:dataprotection : :aws:data-identifier/ -BR RgNumber
arn:aws:dataprotection : :aws:data-identifier/ -CA SocialInsuranceNumber
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:dataprotection : :aws:data-identifier/ -DE TaxId
arn:aws:dataprotection : :aws:data-identifier/ -ES TaxId
arn:aws:dataprotection : :aws:data-identifier/ -FR TaxId
arn:aws:dataprotection : :aws:data-identifier/ -GB TaxId
arn:aws:dataprotection : :aws:data-identifier/ VehicleIdentificationNumber
arn:aws:dataprotection : :aws:data-identifier/ -US ZipCode

Utilisation d'identifiants de données personnalisés dans Amazon SNS

Rubriques

- En quoi consistent les identifiants de données personnalisés ?
- Utilisation d'identifiants de données personnalisés dans votre politique de protection des données
- Contraintes liées aux identifiants de données personnalisés

En quoi consistent les identifiants de données personnalisés ?

Les identifiants de données personnalisés (CDI) vous permettent de définir vos propres expressions régulières personnalisées et de les utiliser éventuellement dans votre politique de protection des données. En utilisant des identifiants de données personnalisés, vous pouvez cibler des cas d'utilisation de données d'identification personnelle (PII) propres à un domaine d'activité, ce que les identifiants de données gérés ne peuvent pas offrir. Par exemple, vous pouvez utiliser un

identifiant de données personnalisé pour rechercher les ID d'employés spécifiques d'une société. Les identifiants de données personnalisés peuvent être utilisés conjointement avec des identifiants de données gérés.

Utilisation d'identifiants de données personnalisés dans votre politique de protection des données

La politique de protection des données suivante donne instruction à la rubrique Amazon SNS de détecter les charges utiles contenant les ID d'employés spécifiques d'une société, puis de masquer ces ID à l'aide du symbole dièse (#).

1. Créez un bloc `Configuration` dans votre politique de protection des données.
2. Nommez votre identifiant de données personnalisé dans `Name`. Par exemple, **`EmployeeId`**.
3. Nommez votre identifiant de données personnalisé dans `Regex`. Par exemple, **`EID-\d{9}-US`**.
4. Faites référence à l'identifiant de données personnalisé suivant dans une déclaration de politique.

```
{
 "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. (Facultatif) Continuez à ajouter des identificateurs de données personnalisés supplémentaires au bloc `Configuration`, si nécessaire. Les politiques de protection des données prennent actuellement en charge au maximum 10 identifiants de données personnalisés.

Contraintes liées aux identifiants de données personnalisés

Les identificateurs de données personnalisés Amazon SNS présentent les limitations suivantes :

- Le nombre d'identifiants de données personnalisés pris en charge pour chaque politique de protection des données est limité à 10.
- Les noms d'identificateurs de données personnalisés ont une longueur maximale de 128 caractères. Les caractères pris en charge sont les suivants :
  - Alphanumériques : (a-zA-Z0-9)
  - Symboles : ('_' | '-')
- La longueur maximale de RegEx est de 200 caractères. Les caractères pris en charge sont les suivants :
  - Alphanumériques : (a-zA-Z0-9)
  - Symboles : ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - Caractères réservés pour RegEx : ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' \ \ '|' * '|' * '|' + '|'. ')
- Les identifiants de données personnalisés ne peuvent pas avoir le même nom qu'un identifiant de données géré.
- Les identifiants de données personnalisés doivent être spécifiés dans chaque politique de protection des données de chaque rubrique Amazon SNS.

Distribution des messages Amazon SNS

Cette section décrit le fonctionnement de la distribution des messages.

Rubriques

Remise des messages bruts Amazon SNS

Pour éviter que les points de terminaison Amazon Data Firehose, Amazon SQS et HTTP/S ne traitent le formatage JSON des messages, Amazon SNS autorise la livraison de messages bruts :

- Lorsque vous activez la livraison de messages bruts pour les points de terminaison Amazon Data Firehose ou Amazon SQS, toutes les métadonnées Amazon SNS sont supprimées du message publié et le message est envoyé tel quel.
- Lorsque vous activez la remise de messages bruts pour les points de terminaison HTTP/S, l'en-tête HTTP `x-amz-sns-rawdelivery` avec sa valeur définie sur `true` est ajouté au message, indiquant que le message a été publié sans formatage JSON.
- Lorsque vous activez la remise de messages bruts pour les points de terminaison HTTP/S, le corps du message, l'adresse IP du client et les en-têtes requis sont fournis. Lorsque vous spécifiez des attributs de message, ils ne seront pas envoyés.
- Lorsque vous activez la livraison de messages bruts pour les points de terminaison Firehose, le corps du message est délivré. Lorsque vous spécifiez des attributs de message, ils ne seront pas envoyés.

Pour activer la livraison de messages bruts à l'aide d'un AWS SDK, vous devez utiliser l'action `SetSubscriptionAttribute` API et définir la valeur de l'`RawMessageDelivery`attribut sur. `true`

Activation de la remise des messages bruts avec AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sur la page Rubriques, choisissez une rubrique abonnée à un point de terminaison Firehose, Amazon SQS ou HTTP/S.
4. Sur la ***MyTopic***page, dans la section Abonnement, choisissez un abonnement, puis sélectionnez Modifier.
5. Sur la page Modifier ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***, dans la section Détails, choisissez Activer la remise des messages bruts.
6. Sélectionnez Enregistrer les modifications.

Exemples de format des messages

Dans les exemples suivants, le même message est envoyé deux fois à la même file d'attente Amazon SQS. La seule différence est que la remise des messages bruts est désactivée pour le premier message et activée pour le second.

- La remise des messages bruts est désactivée


```
{
  "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 remise des messages bruts est activée

```
This is a test message.
```

Attributs des messages et envoi de messages bruts pour les abonnements Amazon SQS

Amazon SNS prend en charge la livraison d'attributs de message, qui vous permettent de fournir des éléments de métadonnées structurés, tels que des horodatages, des données géospatiales, des signatures et des identifiants, concernant le message. Pour les abonnements Amazon SQS avec la livraison de messages bruts activée, un maximum de 10 attributs de message peuvent être envoyés. Pour envoyer plus de 10 attributs de message, vous devez désactiver la livraison de messages bruts. Cependant, Amazon SNS rejette les messages contenant plus de 10 attributs de message destinés aux abonnements Amazon SQS avec l'option Raw Message Delivery activée, les traitant comme des erreurs côté client.

Envoi de messages Amazon SNS à une file d'attente Amazon SQS d'un autre compte

Ce document explique comment publier une notification dans une rubrique Amazon SNS avec un ou plusieurs abonnements à des files d'attente Amazon SNS dans un autre compte. Configurez la rubrique et les files d'attente de la même manière que vous le feriez s'ils se trouvaient dans le même compte (voir Distribution ramifiée vers des files d'attente Amazon SQS). La différence majeure réside dans le traitement de la confirmation d'abonnement, qui varie selon le mode d'abonnement de la file d'attente à la rubrique.

Il est recommandé de suivre les étapes référencées dans la section Création de l'abonnement par le propriétaire de la file d'attente lorsque c'est possible, car la confirmation est automatique lorsque le propriétaire de la file d'attente crée l'abonnement.

> **Note**
>
> Si la file d'attente Amazon SQS contient un volume élevé de messages, nous recommandons au propriétaire de la file d'attente de créer l'abonnement.

Rubriques

Création de l'abonnement par le propriétaire de la file d'attente

Le compte qui a créé la file d'attente Amazon SNS est le propriétaire de la file d'attente. Lorsque le propriétaire de la file d'attente crée un abonnement, ce dernier ne requiert pas de confirmation. La file d'attente commence à recevoir des notifications de la rubrique dès que l'action `Subscribe` est terminée. Pour permettre au propriétaire de la file d'attente de s'abonner à la rubrique du propriétaire de la rubrique, ce dernier doit accorder au propriétaire de la file d'attente l'autorisation d'appeler l'action `Subscribe` sur la rubrique.

Étape 1 : Pour définir la politique de rubrique à l'aide de AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sélectionnez une rubrique, puis choisissez Modifier.
4. Dans la page Edit ***MyTopic*** (Modifier MyTopic) développez la section Access policy (politique d'accès).
5. Saisissez la politique suivante :

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
```

```
            },
            "Action": "sns:Subscribe",
            "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
    ]
}
```

Cette stratégie donne au compte `111122223333` l'autorisation d'appeler `sns:Subscribe`sur `MyTopic` dans le compte `123456789012`.

Un utilisateur possédant les informations d'identification pour le compte `111122223333` peut s'abonner à `MyTopic`. Cette autorisation permet à l'ID de compte de déléguer l'autorisation à son utilisateur/rôle IAM. Seuls les utilisateurs du compte racine ou les administrateurs seront autorisés à appeler `sns:Subscribe`. L'utilisateur/le rôle IAM devra également avoir `sns:subscribe` pour permettre à sa file d'attente de s'abonner.

6. Choisissez Enregistrer les modifications.

   Un utilisateur possédant les informations d'identification pour le compte `111122223333` peut s'abonner à MyTopic.

Étape 2 : Pour ajouter un abonnement à une file d'attente Amazon SQS à une rubrique d'un autre Compte AWS à l'aide de AWS Management Console

Avant de commencer, assurez-vous que vous disposez des ARN de votre rubrique et de votre file d'attente, et que vous avez autorisé la rubrique à envoyer des messages à la file d'attente.

1. Connectez-vous à la console Amazon SQS.
2. Dans le panneau de navigation, choisissez Queues (Files d'attente).
3. Dans la liste des files d'attente, choisissez la file d'attente à abonner à la rubrique Amazon SNS.
4. Choisissez Subscribe to Amazon SNS topic (Abonner à la rubrique Amazon SNS).
5. Dans Spécifiez une rubrique Amazon SNS disponible pour ce menu de file d'attente, choisissez la Rubrique Amazon SNS pour votre file d'attente.
6. Choisissez Entrer l'ARN de la rubrique Amazon SNS, puis entrez la rubrique Amazon Resource Name (ARN).
7. Choisissez Save (Enregistrer).

> **Note**
>
> - Pour pouvoir communiquer avec le service, la file d'attente doit disposer des autorisations pour Amazon SNS.
> - Parce que vous êtes le propriétaire de la file d'attente, vous n'avez pas à confirmer l'abonnement.

Création de l'abonnement par un utilisateur qui n'est pas le propriétaire de la file d'attente

Tout utilisateur qui crée un abonnement mais qui n'est pas le propriétaire de la file d'attente doit confirmer l'abonnement.

Lorsque vous utilisez l'action `Subscribe`, Amazon SNS envoie une confirmation d'abonnement à la file d'attente. L'abonnement s'affiche dans la console Amazon SNS et son ID d'abonnement est défini sur Confirmation en attente.

Pour confirmer l'abonnement, un utilisateur autorisé à lire des messages dans la file d'attente doit récupérer l'URL de confirmation d'abonnement, et le propriétaire de l'abonnement doit confirmer l'abonnement à l'aide de l'URL de confirmation d'abonnement. Aucune notification publiée dans la rubrique n'est envoyée à la file d'attente, tant que l'abonnement n'est pas confirmé. Pour confirmer l'abonnement, vous pouvez utiliser la console Amazon SQS ou l'action `ReceiveMessage`.

> **Note**
>
> Avant d'abonner un point de terminaison à la rubrique, assurez-vous que la file d'attente peut recevoir des messages de la rubrique en définissant l'autorisation `sqs:SendMessage` pour la file d'attente. Pour de plus amples informations, veuillez consulter Étape 2 : autoriser la rubrique Amazon SNS à envoyer des messages à la file d'attente Amazon SQS.

Étape 1 : Pour ajouter un abonnement à une file d'attente Amazon SQS à une rubrique d'un autre Compte AWS à l'aide de AWS Management Console

Avant de commencer, assurez-vous que vous disposez des ARN de votre rubrique et de votre file d'attente, et que vous avez autorisé la rubrique à envoyer des messages à la file d'attente.

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Abonnements.
3. Sur la page Subscriptions (Abonnements), choisissez Create subscription (Créer un abonnement).
4. Sur la page Créer un abonnement, dans la section Détails, procédez comme suit :
   a. Pour ARN de la rubrique, entrez l'ARN de la rubrique.
   b. Pour Protocole, choisissez Amazon SQS.
   c. Pour Point de terminaison, entrez l'ARN de la file d'attente.
   d. Choisissez Create subscription (Créer un abonnement).

> **Note**
>
> - Pour pouvoir communiquer avec le service, la file d'attente doit disposer des autorisations pour Amazon SNS.

Voici un exemple d'énoncé de stratégie qui permet à la rubrique Amazon SNS d'envoyer un message à la file d'attente 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"
      }
   }
}
```

Étape 2 : Pour confirmer un abonnement à l'aide de AWS Management Console

1. Connectez-vous à la console Amazon SQS.
2. Sélectionnez la file d'attente comportant un abonnement à la rubrique en attente.

3. Choisissez Send and receive messages (Envoyer et recevoir des messages), puis Poll for messages (Rechercher des messages).

   Un message contenant la confirmation de l'abonnement est reçu dans la file d'attente.

4. Dans la colonne Corps procédez comme suit :

   a. Choisissez Plus de détails.

   b. Dans la boîte de dialogue Message Details (Détails des messages), recherchez et notez la valeur de SubscribeURL. Il s'agit du lien de votre abonnement (exemple ci-dessous). Pour plus d'informations sur la validation des jetons d'API, consultez `ConfirmSubscription` dans la référence d'API Amazon SNS.

   ```
   https://sns.us-west-2.amazonaws.com/?
   Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
   east-2:123456789012:MyTopic&Token=2336412f37fb...
   ```

   c. Notez le lien de confirmation d'abonnement. L'URL doit être transmise du propriétaire de la file d'attente au propriétaire de l'abonnement. Le propriétaire de l'abonnement doit entrer l'URL dans la Console Amazon SNS.

5. Connectez-vous en tant que propriétaire de l'abonnement à la Console Amazon SNS Le propriétaire de l'abonnement effectue la confirmation.

6. Choisissez la rubrique appropriée.

7. Choisissez l'abonnement approprié dans le tableau des listes d'abonnements de la rubrique. Il est étiqueté comme « Confirmation en attente ».

8. Choisissez le lien de Confirmer l'abonnement.

9. Un modal apparaît pour indiquer le lien de confirmation de l'abonnement. Collez le lien de confirmation de l'abonnement.

10. Sélectionnez la confirmation de l'abonnement dans le modal.

    Une réponse XML s'affiche, par exemple :

    ```
    <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 file d'attente abonnée est prête à recevoir des messages de la rubrique.

11. (Facultatif) Si vous affichez l'abonnement à la rubrique dans la console Amazon SNS, vous pouvez constater que le message Confirmation en attente a été remplacé par l'ARN d'abonnement dans la colonne ID de l'abonnement.

Comment forcer un abonnement à exiger une authentification sur les demandes de désabonnement ?

Le propriétaire de l'abonnement doit définir l'indicateur `AuthenticateOnUnsubscribe` sur true lors de la confirmation de l'abonnement.

- `AuthenticateOnUnsubscribe` est automatiquement défini sur true lorsque le propriétaire de la file d'attente crée l'abonnement.
- `AuthenticateOnUnsubscribe` ne peut pas être défini sur true lorsque le lien de confirmation de l'abonnement est atteint sans authentification.

Envoi de messages Amazon SNS à une file d'attente Amazon SQS ou à une fonction AWS Lambda dans une autre région

Amazon SNS prend en charge les livraisons interrégionales, à la fois pour les régions activées par défaut et pour les régions d'adhésion. Pour obtenir la liste actuelle des régions AWS prises en charge par Amazon SNS, y compris les régions d'adhésion, consultez Points de terminaison et quotas Amazon Simple Notification Service dans la Référence générale d'Amazon Web Services.

Amazon SNS prend en charge la livraison interrégionale de notifications aux files d'attente Amazon SQS et aux fonctions AWS Lambda Lorsque l'une des régions est une région d'adhésion, vous devez spécifier un principal de service Amazon SNS différent dans la stratégie de la ressource abonnée.

La commande d'abonnement Amazon SNS doit être exécutée dans le compte cible de la région où Amazon SNS est hébergé. Par exemple, si Amazon SNS figure dans le compte « A », dans la région us-east-1, et que la fonction Lambda figure dans le compte « B », dans la région us-east-2, la commande CLI d'abonnement doit être exécutée dans le compte « A », dans la région us-east-1.

Régions d'adhésion

Amazon SNS prend en charge les régions d'adhésion suivantes :

Nom de la région	Région
Région Afrique (Le Cap)	af-south-1
Région Asie-Pacifique (Hong Kong)	ap-east-1
Région Asie-Pacifique (Hyderabad)	ap-south-2
Région Asie-Pacifique (Jakarta)	ap-southeast-3
Région Asie-Pacifique (Melbourne)	ap-southeast-4
Région Asie-Pacifique (Osaka)	ap-northeast-3
Europe (Milan) Region	eu-south-1
Région Europe (Espagne)	eu-south-2
Région Europe (Zurich)	eu-central-2
Région Israël (Tel Aviv)	il-central-1
Middle East (Bahrain) Region	me-south-1
Région du Moyen-Orient (EAU)	me-central-1

Pour plus d'informations sur l'activation d'une région optionnelle, consultez la section Gestion des AWS régions dans le Référence générale d'Amazon Web Services.

Lorsque vous utilisez Amazon SNS pour distribuer des messages à partir de régions d'adhésion vers des régions activées par défaut, vous devez modifier la politique de ressource créée pour la file d'attente. Remplacez le principal `sns.amazonaws.com` par `sns.<opt-in-region>.amazonaws.com`. Par exemple :

- Pour vous abonner à une file d'attente Amazon SQS dans la région USA Est (Virginie du Nord) pour une rubrique SNS en Asie-Pacifique (Hong Kong), remplacez le principal dans la stratégie

de la file d'attente par `sns.ap-east-1.amazonaws.com`. Les régions d'adhésion comprennent toutes les régions lancées après le 20 mars 2019, notamment Asie-Pacifique (Hong Kong), Asie-Pacifique (Jakarta), Moyen-Orient (Bahreïn), Europe (Milan) et Afrique (Le Cap). Les régions lancées avant le 20 mars 2019 sont activées par défaut.

Prise en charge de la livraison interrégionale à Amazon SQS

Type de livraison interrégionale	Prise en charge/Non prise en charge
Région activée par défaut vers région d'inscription	Prise en charge avec `sns.<opt-in-region>.amazonaws.com` dans le principal de service de la file d'attente
Région d'inscription vers région activée par défaut	Prise en charge avec `sns.<opt-in-region>.amazonaws.com` dans le principal de service de la file d'attente
Région d'inscription vers la région d'inscription	Non pris en charge

Voici un exemple de déclaration de politique d'accès qui permet à une rubrique Amazon SNS d'une région optionnelle (af-south-1) d'être envoyée à une file d'attente Amazon SQS d'une région (us-east-1). enabled-by-default Il contient la configuration principale de service régionalisée nécessaire sous le chemin d'accès `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"
        }
      }
    },
    ...
  ]
}
```

- Pour abonner une AWS Lambda fonction située dans l'est des États-Unis (Virginie du Nord) à une rubrique Amazon SNS en Asie-Pacifique (Hong Kong), remplacez le principal de la politique de AWS Lambda fonction par. `sns.ap-east-1.amazonaws.com` Les régions d'adhésion comprennent toutes les régions lancées après le 20 mars 2019, notamment Asie-Pacifique (Hong Kong), Asie-Pacifique (Jakarta), Moyen-Orient (Bahreïn), Europe (Milan) et Afrique (Le Cap). Les régions lancées avant le 20 mars 2019 sont activées par défaut.

  Assistance à la livraison interrégionale pour AWS Lambda

Type de livraison interrégionale	Prise en charge/Non prise en charge	
Région activée par défaut vers région d'inscription	Non pris en charge	
Région d'inscription vers région activée par défaut	Prise en charge avec `sns.<opt-in-region>.amazonaws.com` dans le principal de service de la fonction Lambda	
Région d'inscription vers région d'inscription	Non pris en charge	

Statut de distribution de message Amazon SNS

Amazon SNS prend en charge la journalisation de le statut de distribution des messages de notification envoyés aux rubriques avec les points de terminaison Amazon SNS suivants :

- HTTP
- Amazon Data Firehose
- AWS Lambda
- Point de terminaison d’application de plateforme
- Amazon Simple Queue Service

Une fois que vous avez configuré les attributs d'état de livraison des messages, les entrées du journal sont envoyées aux CloudWatch journaux pour les messages envoyés aux abonnés à la rubrique. La consignation de le statut de distribution du message permet de fournir des informations opérationnelles plus précises, par exemple :

- Savoir si un message a été distribué au point de terminaison Amazon SNS.
- Identifiez la réponse envoyée à Amazon SNS par le point de terminaison Amazon SNS.
- Déterminez la durée de conservation du message (la durée entre l'horodatage de publication et le moment juste avant sa distribution à un point de terminaison Amazon SNS).

Pour configurer les attributs des rubriques en fonction de l'état de livraison des messages AWS Management Console, vous pouvez utiliser les kits de développement AWS logiciel (SDK), l'API de requête ou AWS CloudFormation.

Rubriques

- Configuration de le statut de distribution à l'aide de AWS Management Console
- Configuration de la journalisation de l'état de livraison à l'aide des AWS SDK
- AWS Exemples de SDK pour configurer les attributs des rubriques
- Configuration de la journalisation du statut de distribution à l'aide d' AWS CloudFormation

Configuration de le statut de distribution à l'aide de AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sur la page Rubriques, sélectionnez une rubrique, puis choisissez Modifier.
4. Sur la *MyTopic*page Modifier, développez la section Enregistrement de l'état de livraison.

5. Choisissez le protocole pour lequel vous souhaitez enregistrer le statut de distribution, par exemple AWS Lambda.
6. Entrez le taux d'échantillonnage de réussite (le pourcentage de messages réussis pour lesquels vous souhaitez recevoir des CloudWatch journaux).
7. Dans la section rôles IAM, effectuez l'une des opérations suivantes :

   - Pour choisir un rôle de service existant à partir de votre compte, choisissez Use existing service role (Utiliser un rôle de service existant), puis spécifiez les rôles IAM pour les distributions réussies et celles ayant échoué.

   - Pour créer un nouveau rôle de service dans votre compte, choisissez Create new service role (Créer un nouveau rôle de service), choisissez Create new roles (Créer de nouveaux rôles) pour définir les rôles IAM pour les distributions réussies et celles ayant échoué dans la console IAM.

     Pour autoriser Amazon SNS à utiliser CloudWatch Logs en votre nom, sélectionnez Autoriser.
8. Sélectionnez Enregistrer les modifications.

   Vous pouvez désormais consulter et analyser les CloudWatch journaux contenant l'état de livraison des messages. Pour plus d'informations sur l'utilisation CloudWatch, consultez la CloudWatchdocumentation.

Configuration de la journalisation de l'état de livraison à l'aide des AWS SDK

Les AWS SDK fournissent des API en plusieurs langues pour utiliser les attributs d'état de livraison des messages avec Amazon SNS.

Attributs de rubrique

Vous pouvez utiliser les valeurs de nom d'attribut de rubrique suivantes pour le statut de livraison du message :

HTTP

- `HTTPSuccessFeedbackRoleArn` : indique que l'état de diffusion des messages est réussi pour une rubrique Amazon SNS abonnée à un point de terminaison HTTP.

- `HTTPSuccessFeedbackSampleRate` : indique le pourcentage de messages réussis pour une rubrique Amazon SNS abonnée à un point de terminaison HTTP.
- `HTTPFailureFeedbackRoleArn` : indique que l'état de diffusion des messages est en échec pour une rubrique Amazon SNS abonnée à un point de terminaison HTTP.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn` : indique que l'état de diffusion des messages est réussi pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate` : indique le pourcentage de messages réussis pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn` : indique que l'état de diffusion des messages est en échec pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn` : indique que l'état de diffusion des messages est réussi pour une rubrique Amazon SNS abonnée à un point de terminaison Lambda.
- `LambdaSuccessFeedbackSampleRate` : indique le pourcentage de messages réussis pour une rubrique Amazon SNS abonnée à un point de terminaison Lambda.
- `LambdaFailureFeedbackRoleArn` : indique que l'état de diffusion des messages est en échec pour une rubrique Amazon SNS abonnée à un point de terminaison Lambda.

Point de terminaison de l'application de plateforme

- `ApplicationSuccessFeedbackRoleArn`— Indique l'état de transmission du message réussi pour une rubrique Amazon SNS abonnée à un point de terminaison d' AWS application.
- `ApplicationSuccessFeedbackSampleRate`— Indique le pourcentage de messages réussis à échantillonner pour une rubrique Amazon SNS abonnée à un point de terminaison d' AWS application.
- `ApplicationFailureFeedbackRoleArn`— Indique le statut d'échec de livraison des messages pour une rubrique Amazon SNS abonnée à un point de terminaison d' AWS application.

> **Note**
>
> Outre la possibilité de configurer des attributs de rubrique pour le statut de distribution des messages de notification envoyés à des points de terminaison d'application Amazon SNS, vous pouvez également configurer des attributs d'application pour le statut de distribution des messages de notification push envoyés aux services de notification push. Pour plus d'informations, consultez la page Utilisation des attributs d'application Amazon SNS pour le statut de distribution du message.

Amazon SQS

- `SQSSuccessFeedbackRoleArn` : indique que l'état de diffusion des messages est réussi pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon SQS.
- `SQSSuccessFeedbackSampleRate` : indique le pourcentage de messages réussis pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon SQS.
- `SQSFailureFeedbackRoleArn` : indique que l'état de diffusion des messages est en échec pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon SQS.

> **Note**
>
> Les `<ENDPOINT>FailureFeedbackRoleArn` attributs `<ENDPOINT>SuccessFeedbackRoleArn` et sont utilisés pour donner à Amazon SNS un accès en écriture lui permettant d'utiliser CloudWatch Logs en votre nom. L'attribut `<ENDPOINT>SuccessFeedbackSampleRate` permet de spécifier le pourcentage de la fréquence d'échantillonnage (0-100) des messages diffusés avec succès. Une fois que vous avez configuré l'`<ENDPOINT>FailureFeedbackRoleArn`attribut, toutes les livraisons de messages ayant échoué génèrent des CloudWatch journaux.

AWS Exemples de SDK pour configurer les attributs des rubriques

Les exemples de code suivants montrent comment utiliser`SetTopicAttributes`.

CLI

AWS CLI

Pour définir un attribut pour une rubrique

L'exemple `set-topic-attributes` suivant définit l'attribut `DisplayName` pour la rubrique spécifiée.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for PHP API.

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section de référence du AWS SDK pour l'API SAP ABAP.

Configuration de la journalisation du statut de distribution à l'aide d' AWS CloudFormation

Pour configurer `DeliveryStatusLogging` l'utilisation AWS CloudFormation, utilisez un modèle JSON ou YAML pour créer une AWS CloudFormation pile. Pour plus d'informations, consultez la `DeliveryStatusLogging` propriété de la `AWS::SNS::Topic` ressource dans le guide de AWS CloudFormation l'utilisateur. Vous trouverez ci-dessous des exemples de AWS CloudFormation modèles en JSON et YAML permettant de créer une nouvelle rubrique ou de mettre à jour une rubrique existante avec tous les `DeliveryStatusLogging` attributs du protocole Amazon SQS.

JSON

```
"Resources": {
    "MySNSTopic" : {
```

```
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "TestTopic",
            "DisplayName" : "TEST",
            "SignatureVersion" : "2",
            "DeliveryStatusLogging" : [{
                "Protocol": "sqs",
                "SuccessFeedbackSampleRate": "45",
                "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
                "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
            }]
        }
    }
}
```

YAML

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName:TestTopic
      DisplayName:TEST
      SignatureVersion:2
      DeliveryStatusLogging:
       - Protocol: sqs
         SuccessFeedbackSampleRate: 45
         SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
         FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Nouvelle tentative de distribution des messages Amazon SNS

Amazon SNS définit une politique de distribution pour chaque protocole de distribution. La politique de distribution définit comment Amazon SNS tente à nouveau de livrer des messages lorsque des erreurs côté serveur se produisent (lorsque le système qui héberge le point de terminaison abonné

devient indisponible). Lorsque la politique de distribution expire, Amazon SNS cesse de retenter la distribution et rejette le message, sauf si une file d'attente de lettres mortes est jointe à l'abonnement. Pour plus d'informations, consultez Files d'attente de lettres mortes (DLQ)) d'Amazon SNS.

Rubriques

Protocoles et politiques de distribution

> **Note**
>
> - À l'exception de HTTP/S, vous ne pouvez pas modifier les politiques de distribution définies par Amazon SNS. Seul HTTP/S prend en charge les politiques personnalisées. veuillez consulter Création d'une politique de distribution HTTP/S.
> - Amazon SNS applique l'instabilité aux nouvelles tentatives de distribution. Pour plus d'informations, consultez la publication Exponential Backoff and Jitter sur le blog AWS Architecture.
> - Le temps total de nouvelle tentative de politique pour un point de terminaison HTTP/S ne peut pas être supérieur à 3 600 secondes. Il s'agit d'une limite fixe qui ne peut pas être augmentée.

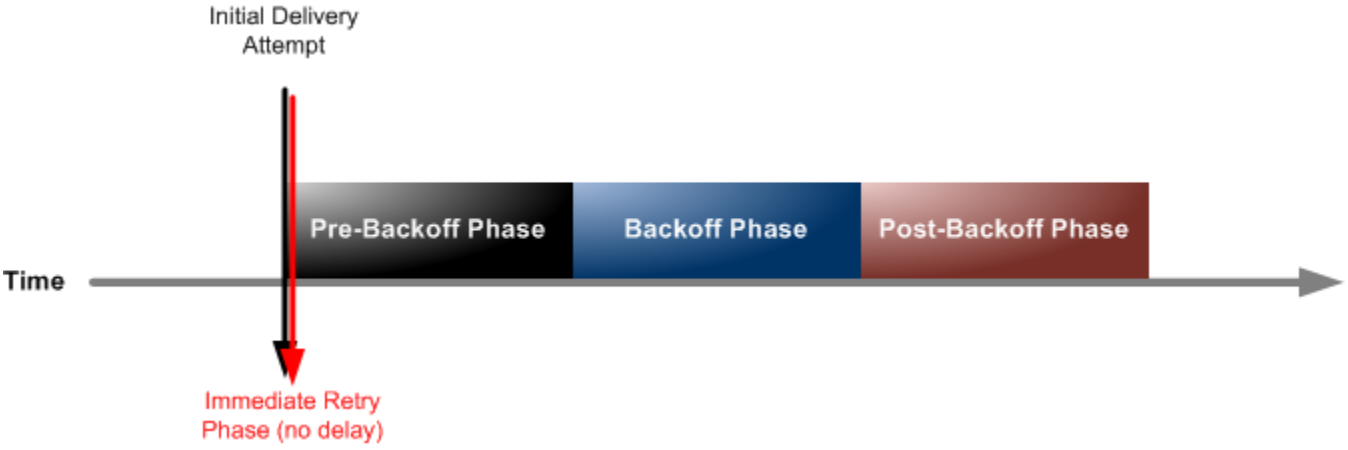
Type de point de terminaison	Protocoles de distribution	Phase nouvelle tentative immédiate (sans délai)	Phase de pré-interruption	Phase d'interruption	Phase de post-interruption	Nombre total de tentatives
AWS points de terminaison gérés	Amazon Data Firehose[1]	3 fois, sans délai	2 fois, 1 seconde d'intervalle	10 fois, avec retour exponentiel, de	100 000 fois, 20 secondes d'intervalle	100 015 fois, plus de 23 jours

Type de point de terminaison	Protocoles de distribution	Phase nouvelle tentative immédiate (sans délai)	Phase de pré-interruption	Phase d'interruption	Phase de post-interruption	Nombre total de tentatives
	AWS Lambda			1 seconde à 20 secondes		
	Amazon SQS					
Points de terminaison gérés par le client	SMTP	0 fois, sans délai	2 fois, 10 secondes d'intervalle	10 fois, avec retour exponentiel, de 10 secondes à 600 secondes (10 minutes)	38 fois, 600 secondes (10 minutes) d'intervalle	50 tentatives, plus de 6 heures
	SMS					
	Push mobile					

[1] Pour limiter les erreurs liées au protocole Firehose, Amazon SNS applique la même politique de livraison que pour les points de terminaison gérés par le client.

Étapes de la politique de distribution

Le diagramme suivant montre les phases d'une politique de distribution.



Chaque politique de distribution comprend quatre phases.

1. Phase de nouvelle tentative immédiate (sans délai) – Cette phase se produit immédiatement après la tentative de distribution initiale. Il n'y a aucun délai entre les relances dans cette phase.
2. Phase de pré-interruption – Cette phase suit la phase de nouvelle tentative immédiate. Amazon SNS utilise cette phase pour tenter une série de nouvelles tentatives avant d'appliquer une fonction d'interruption. Cette phase spécifie le nombre de nouvelles tentatives et le délai entre elles.
3. Phase d'interruption – Cette phase contrôle le délai entre les nouvelles tentatives à l'aide de la fonction retry-backoff. Cette phase définit un délai minimum, un délai maximum et une fonction d'interruption des nouvelles tentatives qui définit la vitesse à laquelle le délai augmente depuis le délai minimum au délai maximum. La fonction d'interruption peut être arithmétique, exponentielle, géométrique ou linéaire.
4. Phase de post-interruption – Cette phase suit la phase d'interruption. Il spécifie un certain nombre de nouvelles tentatives et la longueur du délai entre elles. Il s'agit de la phase finale.

Création d'une politique de distribution HTTP/S

Vous pouvez utiliser une politique de distribution et ses quatre phases pour définir comment Amazon SNS effectue une nouvelle distribution des messages aux points de terminaison HTTP/S. Amazon SNS vous permet de remplacer la politique de nouvelle tentative par défaut pour les points de terminaison HTTP lorsque vous souhaitez, par exemple, la personnaliser en fonction de la capacité de votre serveur HTTP.

Vous pouvez définir votre politique de distribution HTTP/S en tant qu'objet JSON au niveau de l'abonnement ou de la rubrique. Lorsque vous définissez la politique au niveau de la rubrique, elle s'applique à tous les abonnements HTTP/S associés à la rubrique. Pour définir la politique de remise au niveau de l'abonnement, vous pouvez utiliser l'action d'API `Subscribe` ou `SetSubscriptionAttributes`. Pour définir la politique de remise au niveau de la rubrique, vous pouvez utiliser l'action d'API `CreateTopic` ou `SetTopicAttributes`. Vous pouvez également utiliser la AWS::SNS::Subscriptionressource dans vos AWS CloudFormation modèles.

Vous devez personnaliser votre politique de distribution en fonction de la capacité de votre serveur HTTP/S. Vous pouvez définir la politique en tant qu'attribut d'une rubrique ou d'un abonnement. Si tous les abonnements HTTP/S de votre rubrique ciblent le même serveur HTTP/S, nous vous recommandons de définir la politique de remise en tant qu'attribut de rubrique, afin qu'elle reste valable pour tous les abonnements HTTP/S de la rubrique. Sinon, vous devez composer une

politique de distribution pour chaque abonnement HTTP/S de votre rubrique, en fonction de la capacité du serveur HTTP/S ciblé par la politique.

Vous pouvez également définir l'en-tête Content-Type dans la politique de demande pour spécifier le type de support de la notification. Par défaut, Amazon SNS envoie toutes les notifications aux points de terminaison HTTP/S dont le type de contenu est défini sur `text/plain; charset=UTF-8`. Amazon SNS vous permet de remplacer la politique de demande par défaut. Consultez le tableau ci-dessous pour découvrir les contraintes et les types `headerContentType` pris en charge.

L'objet JSON suivant représente une politique de distribution qui indique à Amazon SNS de faire une nouvelle tentative pour une distribution HTTP/S échouée, comme suit :

1. 3 fois immédiatement dans la phase sans délai
2. 2 fois (1 seconde d'intervalle) dans la phase de pré-interruption
3. 10 fois (avec interruption exponentielle de 1 à 60 secondes)
4. 35 fois (60 secondes d'intervalle) dans la phase de post-interruption

Dans cet exemple de politique de distribution, Amazon SNS effectue un total de 50 tentatives avant de rejeter le message. Pour conserver le message après l'épuisement des nouvelles tentatives spécifiées dans la politique de distribution, configurez votre abonnement pour déplacer les messages non délivrables vers une file d'attente de lettres mortes (DLQ). Pour plus d’informations, consultez Files d'attente de lettres mortes (DLQ)) d'Amazon SNS.

> **Note**
>
> En utilisant la propriété `maxReceivesPerSecond`, cette politique de distribution indique également à Amazon SNS de limiter les distributions à un maximum de 10 par seconde. Ce taux d'autorégulation peut entraîner la publication d'un plus grand nombre de messages (trafic entrant) que le nombre de messages délivrés (trafic sortant). Lorsqu'il y a plus de trafic entrant que sortant, votre abonnement peut accumuler un important arriéré de messages, ce qui peut entraîner une latence élevée de remise des messages. Dans vos politiques de distribution, veillez à spécifier une valeur pour `maxReceivesPerSecond` qui n'a pas d'impact négatif sur votre application.

> **Note**
>
> Cette politique de remise remplace le type de contenu par défaut pour les notifications HTTP/S à `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 politique de remise se compose d'une politique de nouvelle tentative, d'une politique de limitation et d'une politique de demande. Au total, une politique de remise comprend 9 attributs.

Politique	Description	Contrainte
`minDelayTarget`	Délai minimal pour une nouvelle tentative. Unité : secondes	1 jusqu'au délai maximal Par défaut : 20
`maxDelayTarget`	Délai maximal pour une nouvelle tentative. Unité : secondes	Délai minimal jusqu'à 3 600 Par défaut : 20

Politique	Description	Contrainte
`numRetries`	Le nombre total de nouvelles tentatives, y compris les tentatives immédiates, antérieures à l'interruption, pendant l'interruption et postérieures à l'interruption.	0 à 100 Par défaut : 3
`numNoDelayRetries`	Le nombre de nouvelles tentatives à effectuer immédiatement, sans délai entre elles.	0 ou plus Par défaut : 0
`numMinDelayRetries`	Nombre de nouvelles tentative s dans la phase antérieure à l'interruption, avec le délai minimum entre elles spécifié.	0 ou plus Par défaut : 0
`numMaxDelayRetries`	Le nombre de nouvelles tentatives dans la phase postérieure à l'interruption, avec le délai maximum entre elles.	0 ou plus Par défaut : 0
`backoffFunction`	Modèle d'interruption entre les nouvelles tentatives.	L'une des quatre options suivantes : • arithmétique • exponentielle • géométrique • linéaire Par défaut : linéaire

Politique	Description	Contrainte
`maxReceivesPerSecond`	Nombre maximal de distributions par seconde, par abonnement.	1 ou plus Par défaut : pas de limitation

Politique	Description	Contrainte
`headerContentType`	Type de contenu de la notification envoyée aux points de terminaison HTTP/S.	Si la politique de demande n'est pas définie, le type de contenu par défaut est `text/plain; charset=UTF-8` . Quand la remise des messages bruts est désactivée pour un abonnement (par défaut) ou quand la politique de remise est définie au niveau de la rubrique, les types de contenu d'en-tête pris en charge sont `application/json` et `text/plain` . Quand la remise des messages bruts est activée pour un abonnement, les types de contenu suivants sont pris en charge : • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • demande/ x-www-form-urlencoded • application/xhtml+xml

Politique	Description	Contrainte
		• application/xml

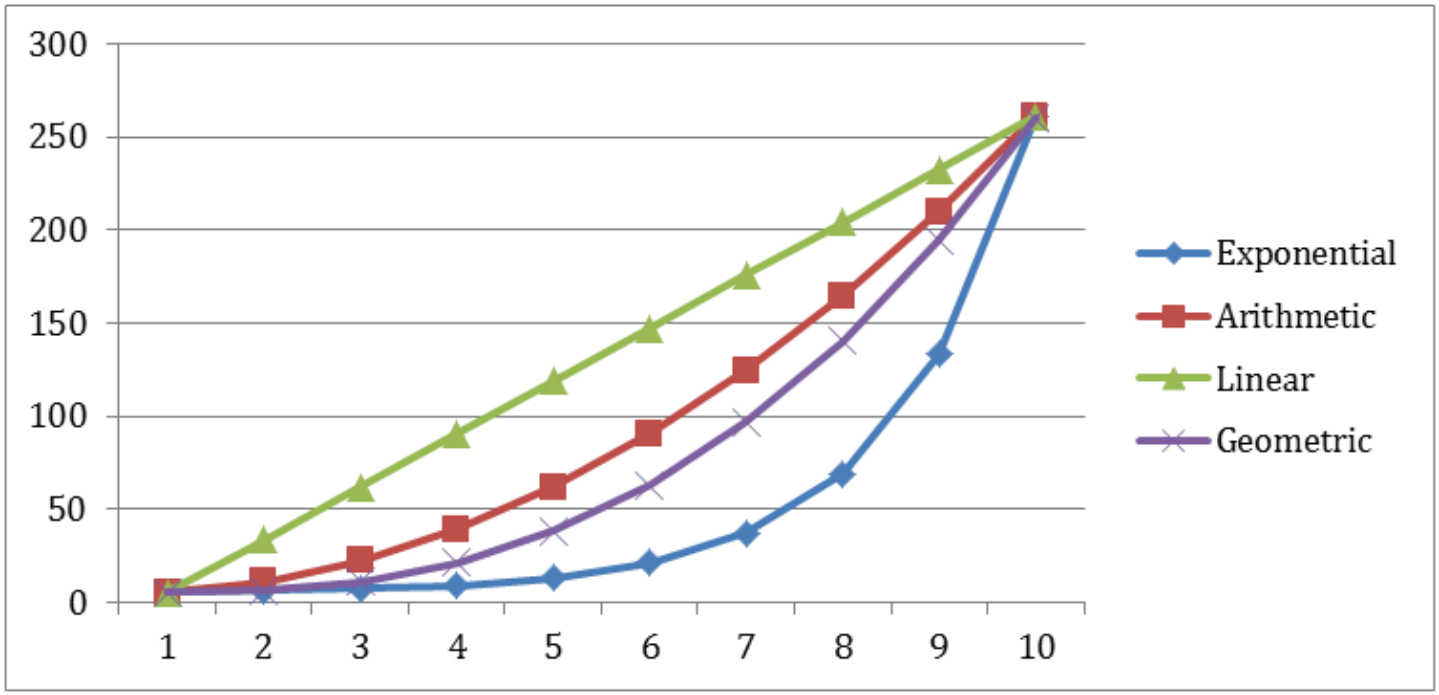
Amazon SNS utilise la formule suivante pour calculer le nombre de nouvelles tentatives dans la phase d'interruption :

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

Vous pouvez utiliser trois paramètres pour contrôler la fréquence des nouvelles tentatives dans la phase d’interruption.

- `minDelayTarget` – Définit le délai associé à la première nouvelle tentative dans la phase d'interruption.
- `maxDelayTarget` – Définit le délai associé à la nouvelle tentative finale dans la phase d'interruption.
- `backoffFunction` – Définit l'algorithme utilisé par Amazon SNS pour calculer les délais associés à toutes les nouvelles tentatives entre la première et la dernière tentative dans la phase d'interruption. Vous pouvez utiliser l'une des quatre fonctions d’interruption de nouvelle tentative.

Le diagramme suivant montre comment chaque fonction d’interruption de nouvelle tentative affecte le délai associé aux nouvelles tentatives pendant la phase d’interruption : politique de distribution avec le nombre total de nouvelles tentatives défini sur 10, le délai minimum défini sur 5 secondes et le délai maximum défini sur 260 secondes. L'axe vertical représente le délai, en secondes, associé à chacune des 10 relances. L'axe horizontal représente le nombre de nouvelles tentatives, de la première à la dixième.



Files d'attente de lettres mortes (DLQ)) d'Amazon SNS

Une file d'attente de lettres mortes est une file d'attente Amazon SQS qu'un abonnement Amazon SNS peut cibler pour les messages qui ne peuvent pas être remis aux abonnés avec succès. Les messages qui ne peuvent pas être remis en raison d'erreurs du client ou d'erreurs de serveur sont conservés dans la file d'attente de lettres mortes pour une analyse ou un retraitement ultérieur. Pour de plus amples informations, consultez Configuration d'une file d'attente de lettres mortes Amazon SNS pour un abonnement et Nouvelle tentative de distribution des messages Amazon SNS.

Note

- L'abonnement Amazon SNS et la file d'attente Amazon SQS doivent être sous le même compte et la même région AWS.
- Pour une rubrique FIFO, vous pouvez utiliser une file d'attente Amazon SQS en tant que file d'attente de lettres mortes pour l'abonnement Amazon SNS. Les abonnements aux rubriques FIFO utilisent des files d'attente FIFO, tandis que les abonnements aux rubriques standard utilisent des files d'attente standard.
- Pour utiliser une file d'attente Amazon SQS chiffrée en tant que file d'attente de lettres mortes, vous devez utiliser une clé KMS personnalisée avec une politique de clé qui accorde au principal de service Amazon SNS l'accès aux actions d'API AWS KMS. Pour

plus d'informations, consultez Chiffrement au repos dans ce guide et Protection des données Amazon SQS à l'aide du chiffrement côté serveur (SSE) et de AWS KMS dans le Guide du développeur Amazon Simple Queue Service.

Rubriques

Pourquoi les distributions de messages échouent-elles ?

En général, la distribution des messages échoue lorsqu'Amazon SNS ne peut pas accéder à un point de terminaison souscrit en raison d'une erreur côté client ou côté serveur. Lorsqu'Amazon SNS reçoit une erreur côté client ou continue de recevoir une erreur côté serveur pour un message au-delà du nombre de tentatives spécifié par la politique de nouvelle tentative correspondante, Amazon SNS rejette le message – sauf si une file d'attente de lettres mortes est attachée à l'abonnement. Les distributions qui échouent ne modifient pas le statut de vos abonnements. Pour de plus amples informations, consultez Nouvelle tentative de distribution des messages Amazon SNS.

Erreurs côté du client

Des erreurs côté client peuvent se produire lorsque des métadonnées d'abonnement Amazon SNS sont obsolètes. Ces erreurs se produisent généralement lorsqu'un propriétaire supprime le point de terminaison (par exemple une fonction Lambda abonnée à une rubrique Amazon SNS) ou lorsqu'un propriétaire modifie la politique attachée au point de terminaison souscrit d'une manière qui empêche Amazon SNS de délivrer des messages au point de terminaison. Amazon SNS ne tente pas de nouveau d'envoyer des messages qui échouent en raison d'une erreur côté client.

Erreurs côté serveur

Des erreurs côté serveur peuvent se produire lorsque le système responsable du point de terminaison souscrit devient indisponible ou renvoie une exception indiquant qu'il ne peut pas traiter une demande valable à partir d'Amazon SNS. Lorsque des erreurs côté serveur se produisent,

Amazon SNS relance les distributions échouées à l'aide d'une fonction backoff exponentiel ou linéaire. Pour les erreurs côté serveur causées par des points de terminaison gérés par AWS qui reposent sur Amazon SQS ou AWS Lambda, Amazon SNS relance la distribution jusqu'à 100 015 fois, sur 23 jours.

Les points de terminaison gérés par le client (notamment HTTP, SMTP, SMS ou push mobile) peuvent également provoquer des erreurs côté serveur. Amazon SNS tente de nouveau l'envoi vers ces types de points de terminaison également. Alors que les points de terminaison HTTP prennent en charge les politiques de nouvelle tentative définies par le client, Amazon SNS définit une politique de nouvelle tentative de distribution interne à 50 fois sur 6 heures, pour les points de terminaison SMTP, SMS et push mobiles.

Fonctionnement des files d'attente de lettres mortes

Une file d'attente de lettres mortes est attachée à un abonnement Amazon SNS (plutôt qu'à une rubrique), car les messages sont délivrés au niveau de l'abonnement. Cela vous permet d'identifier plus facilement le point de terminaison cible d'origine pour chaque message.

Une file d'attente de lettres mortes associée à un abonnement Amazon SNS est une file d'attente Amazon SQS ordinaire. Pour de plus amples informations sur la période de rétention des messages, consultez Quotas liés aux messages dans le Guide du développeur Amazon Simple Queue Service. Vous pouvez modifier la période de rétention des messages à l'aide de l'action d'API `SetQueueAttributes` d'Amazon SQS. Pour rendre vos applications plus résilientes, nous vous recommandons de fixer à 14 jours la période de rétention maximale pour les files d'attente de lettres mortes.

Comment les messages sont-ils déplacés dans une file d'attente de lettres mortes ?

Vos messages sont déplacés dans une file d'attente de lettres mortes à l'aide d'une politique de redirection. Une politique de redirection est un objet JSON qui fait référence à l'ARN de la file d'attente de lettres mortes. L'attribut `deadLetterTargetArn` spécifie l'ARN. L'ARN doit pointer vers une file d'attente Amazon SQS dans le même Compte AWS et la même région que votre abonnement Amazon SNS. Pour de plus amples informations, veuillez consulter Configuration d'une file d'attente de lettres mortes Amazon SNS pour un abonnement.

L'objet JSON suivant est un exemple de politique de redirection attachée à un abonnement SNS.

```
{
```

```
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Comment puis-je déplacer des messages hors d'une file d'attente de lettres mortes ?

Vous pouvez déplacer les messages hors d'une file d'attente de lettres mortes de deux façons :

- Éviter d'écrire une logique de consommateur Amazon SQS – Définissez votre file d'attente de lettres mortes comme source d'évènement pour la fonction Lambda afin de purger votre file d'attente de lettres mortes.
- Écrire une logique de consommateur Amazon SQS – Utilisez l'API Amazon SQS, le kit SDK AWS ou la AWS CLI pour écrire une logique consommateur personnalisée pour l'interrogation, le traitement et la suppression des messages dans la file d'attente de lettres mortes.

Comment puis-je contrôler et consigner les files d'attente de lettres mortes ?

Vous pouvez utiliser les métriques Amazon CloudWatch pour contrôler les files d'attente de lettres mortes associées à vos abonnements Amazon SNS. Toutes les files d'attente Amazon SQS émettent des métriques CloudWatch à intervalles d'une minute. Pour de plus amples informations, consultez Métriques CloudWatch disponibles pour Amazon SQS dans le Guide du développeur Amazon Simple Queue Service. Tous les abonnements Amazon SNS qui comportent des files d'attente de lettres mortes émettent également des métriques CloudWatch. Pour de plus amples informations, veuillez consulter Surveillance des rubriques Amazon SNS à l'aide de CloudWatch.

Pour être averti de l'activité dans vos files d'attente de lettres mortes, vous pouvez utiliser les métriques et les alarmes CloudWatch. Par exemple, lorsque vous vous attendez à ce que la file d'attente des lettres mortes soit toujours vide, vous pouvez créer une alarme CloudWatch pour la métrique `NumberOfMessagesSent`. Vous pouvez définir le seuil d'alarme sur 0 et spécifier une rubrique Amazon SNS à notifier lorsque l'alarme se déclenche. Cette rubrique Amazon SNS peut délivrer votre notification d'alarme à n'importe quel type de point de terminaison (par exemple, une adresse e-mail, un numéro de téléphone ou une application de téléavertisseur mobile).

Vous pouvez utiliser CloudWatch Logs pour rechercher les exceptions qui provoquent l'échec des distributions Amazon SNS et pour envoyer des messages dans des files d'attente de lettres mortes. Amazon SNS peut journaliser les distributions réussies et échouées dans CloudWatch. Pour de plus amples informations, veuillez consulter Statut de distribution de message Amazon SNS.

Configuration d'une file d'attente de lettres mortes Amazon SNS pour un abonnement

Une file d'attente de lettres mortes est une file d'attente Amazon SQS qu'un abonnement Amazon SNS peut cibler pour les messages qui ne peuvent pas être remis aux abonnés avec succès. Les messages qui ne peuvent pas être remis en raison d'erreurs du client ou d'erreurs de serveur sont conservés dans la file d'attente de lettres mortes pour une analyse ou un retraitement ultérieur. Pour de plus amples informations, consultez Files d'attente de lettres mortes (DLQ)) d'Amazon SNS et Nouvelle tentative de distribution des messages Amazon SNS.

Cette page explique comment utiliser le AWS Management Console, un AWS SDK, le AWS CLI, et comment configurer une file AWS CloudFormation d'attente de lettres mortes pour un abonnement Amazon SNS.

> **Note**
>
> Pour une rubrique FIFO, vous pouvez utiliser une file d'attente Amazon SQS en tant que file d'attente de lettres mortes pour l'abonnement Amazon SNS. Les abonnements aux rubriques FIFO utilisent des files d'attente FIFO, tandis que les abonnements aux rubriques standard utilisent des files d'attente standard.

Prérequis

Avant de configurer une file d'attente de lettres mortes, remplissez les conditions préalables suivantes :

1. Créez une rubrique Amazon SNS nommée `MyTopic`.
2. Créez une file d'attente Amazon SQS nommée `MyEndpoint`, à utiliser comme point de terminaison pour l'abonnement Amazon SNS.
3. (Ignorer AWS CloudFormation) Inscrivez la file d'attente au sujet.
4. Créez une autre file d'attente Amazon SQS nommée `MyDeadLetterQueue`, que vous utiliserez en tant que file d'attente de lettres mortes pour l'abonnement Amazon SNS.
5. Pour accorder au principal Amazon SNS l'accès à l'action d'API Amazon SQS, définissez la politique de file d'attente suivante pour `MyDeadLetterQueue`.

```
{
```

```
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
    }
  }]
}
```

Rubriques

- Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide du AWS Management Console
- Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide d'un SDK AWS
- Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide du AWS CLI
- Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide de AWS CloudFormation

Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide du AWS Management Console

Avant de commencer le tutoriel, assurez-vous que vous remplissez les conditions préalables.

1. Connectez-vous à la console Amazon SQS.
2. Créez une file d'attente Amazon SQS ou utilisez une file d'attente existante et notez l'ARN de la file d'attente sous l'onglet Détails de la file d'attente, par exemple :

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

3. Connectez-vous à la console Amazon SNS.

4. Dans le panneau de navigation, choisissez Abonnements.
5. Sur la page Subscriptions (Abonnements), sélectionnez un abonnement existant, puis choisissez Edit (Modifier).
6. Sur la page Modifier ***1234a567-bc89-012d-3e45-6fg7h890123i***, développez la section Politique de redirection (file d'attente de lettres mortes), puis procédez comme suit :

   a. Choisissez Enabled (Activé).

   b. Spécifiez l'ARN d'une file d'attente Amazon SQS.
7. Sélectionnez Enregistrer les modifications.

   Votre abonnement est configuré pour utiliser une file d'attente de lettres mortes.

Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide d'un SDK AWS

Avant d'exécuter cet exemple, assurez-vous de respecter les conditions préalables requises.

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

L'exemple de code suivant montre comment utiliser`SetSubscriptionAttributesRedrivePolicy`.

Java

Kit SDK pour Java 1.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
```

Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide du AWS CLI

Avant de commencer le tutoriel, assurez-vous que vous remplissez les conditions préalables.

1. Installation et configuration de l' AWS CLI. Pour de plus amples informations, consultez le AWS Command Line Interface Guide de l’utilisateur.
2. Utilisez la commande suivante de l’.

```
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\"}"
```

Pour configurer une file d'attente de lettres mortes pour un abonnement Amazon SNS à l'aide de AWS CloudFormation

Avant de commencer le tutoriel, assurez-vous que vous remplissez les conditions préalables.

1. Copiez le code JSON suivant dans un fichier nommé `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. Connectez-vous à la console AWS CloudFormation.
3. Sur la page Select Template (Sélectionner un modèle), choisissez Upload a template to Amazon S3 (Télécharger un modèle sur Amazon S3), puis votre fichier `MyDeadLetterQueue.json`, puis Next (Suivant).
4. Sur la page Specify Details (Spécifier les détails), saisissez `MyDeadLetterQueue` comme Stack Name (Nom de pile), puis choisissez Next (Suivant).
5. Dans la page Options, choisissez Next (Suivant).
6. Sur la page Review (Vérification), choisissez Create (Créer).

   AWS CloudFormation commence à créer la `MyDeadLetterQueue` pile et affiche le statut CREATE_IN_PROGRESS. Lorsque le processus est terminé, AWS CloudFormation affiche le statut CREATE_COMPLETE.

Archivage, relecture et analyse des messages Amazon SNS

Les rubriques standard d'Amazon SNS prennent en charge l'archivage des messages via Amazon Data Firehose. Vous pouvez envoyer des notifications aux flux de diffusion de Firehose, ce qui vous permet d'envoyer des notifications aux destinations de stockage et d'analyse prises en charge par Firehose, notamment Amazon Simple Storage Service (Amazon S3), Amazon Redshift, etc.

Les rubriques FIFO d'Amazon SNS prennent en charge les archives de messages sur place, sans code, qui permettent aux propriétaires de rubrique de stocker (ou archiver) les messages publiés dans une rubrique pendant 365 jours, au maximum. Pour les rubriques associées à une politique `ArchivePolicy` active, les abonnés peuvent alors créer un attribut `ReplayPolicy` pour récupérer (ou relire) les messages archivés sur un point de terminaison abonné. Pour en savoir plus sur cette fonction, veuillez consulter Archivage-relecture des messages pour les rubriques FIFO.

Fonctionnalités	Rubriques standard	Rubriques FIFO
Archivage des messages	Streams de diffusion de Fanout to Firehose	Archivage des messages pour les propriétaires de rubrique FIFO
Relecture des messages	Dans le cas des rubriques standard, la relecture n'est pas une fonctionnalité intégrée. De nombreux clients créent les leurs sur la base de leur archive de messages.	Relecture des messages pour les abonnés à une rubrique FIFO

Utilisation d'Amazon SNS pour la messagerie d'application à application (A2A)

Cette section fournit des informations sur l'utilisation d'Amazon SNS pour la messagerie d'application à application avec les abonnés.

Rubriques

Streams de diffusion de Fanout to Firehose

Vous pouvez abonner les flux de diffusion Amazon Data Firehose aux rubriques Amazon SNS, ce qui vous permet d'envoyer des notifications à des points de terminaison de stockage et d'analyse supplémentaires. Les messages publiés sur une rubrique Amazon SNS sont envoyés au flux de diffusion Firehose auquel vous êtes abonné et envoyés à la destination telle que configurée dans Firehose. Le titulaire d'un abonnement peut abonner jusqu'à cinq flux de diffusion Firehose à une rubrique Amazon SNS. Chaque flux de diffusion Firehose possède un quota par défaut pour les demandes et le débit par seconde. Cette limite peut occasionner la publication de plus de messages (trafic entrant) qu'il n'en est remis (trafic sortant). Lorsque le trafic entrant est plus important que le trafic sortant, votre abonnement peut accumuler un grand nombre de messages en attente, ce qui entraîne une grande latence lors de la remise des messages. Vous pouvez demander une augmentation du quota en fonction du taux de publication afin d'éviter tout impact négatif sur votre charge de travail.

Grâce aux flux de diffusion Firehose, vous pouvez envoyer des notifications Amazon SNS à Amazon Simple Storage Service (Amazon S3), OpenSearch Amazon Redshift, Amazon Service (OpenSearch Service) et à des fournisseurs de services tiers tels que Datadog, New Relic, MongoDB et Splunk.

Par exemple, vous pouvez utiliser cette fonctionnalité pour stocker définitivement les messages envoyés à une rubrique dans un compartiment Amazon S3 à des fins de conformité, d'archivage ou à

d'autres fins. Pour ce faire, créez un flux de diffusion Firehose avec une destination de compartiment S3 et abonnez ce flux de diffusion à la rubrique Amazon SNS. Autre exemple, pour effectuer une analyse des messages envoyés à une rubrique Amazon SNS, créez un flux de diffusion avec une destination d'index de OpenSearch service. Vous pouvez ensuite abonner le flux de diffusion Firehose à la rubrique Amazon SNS.

Amazon SNS prend également en charge l'enregistrement de l'état de livraison des messages pour les notifications envoyées aux points de terminaison Firehose. Pour plus d'informations, consultez Statut de distribution de message Amazon SNS.

Rubriques

Conditions préalables à l'abonnement aux flux de diffusion Firehose aux rubriques Amazon SNS

Pour abonner un flux de diffusion Amazon Data Firehose à une rubrique SNS, vous Compte AWS devez disposer des éléments suivants :

- Une rubrique SNS standard. Pour plus d'informations, consultez Création d'une rubrique Amazon SNS.
- Un flux de diffusion Firehose. Pour plus d'informations, consultez Création d'un flux de diffusion Amazon Data Firehose et accordez à votre application l'accès à vos ressources Firehose dans le guide du développeur Amazon Data Firehose.
- Un rôle (IAM) AWS Identity and Access Management qui approuve le principal de service Amazon SNS et a l'autorisation d'écrire dans le flux de diffusion. Vous saisissez l'Amazon Resource Name (ARN) de ce rôle en tant que `SubscriptionRoleARN` lorsque vous créez l'abonnement. Amazon SNS assume ce rôle, qui permet à Amazon SNS de placer des enregistrements dans le flux de diffusion Firehose.

  L'exemple de politique ci-dessous présente les autorisations recommandées :

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

Pour fournir l'autorisation complète d'utiliser Firehose, vous pouvez également utiliser la politique AWS gérée. `AmazonKinesisFirehoseFullAccess` Ou, pour fournir des autorisations plus strictes pour l'utilisation de Firehose, vous pouvez créer votre propre politique. La politique doit au moins pouvoir autoriser l'exécution de l'opération `PutRecord` sur un flux de diffusion spécifique.

Dans tous les cas, vous devez également modifier la relation de confiance pour inclure le principal de service d'Amazon SNS. Par exemple :

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

Pour en savoir plus sur la création d'un rôle, consulter la section Création d'un rôle pour la délégation d'autorisations à un service AWS du Guide de l'utilisateur IAM.

Une fois ces exigences remplies, vous pouvez abonner le flux de diffusion à la rubrique SNS.

Abonnement d'un flux de diffusion Firehose à une rubrique Amazon SNS

Pour envoyer des notifications Amazon SNS aux flux de diffusion Amazon Data Firehose, assurez-vous d'abord que vous avez rempli toutes les conditions préalables. Pour obtenir la liste des points de terminaison pris en charge, consultez la section Points de terminaison et quotas Amazon Data Firehose dans le. Référence générale d'Amazon Web Services

Pour abonner un stream de diffusion Firehose à un sujet

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Abonnements.
3. Sur la page Abonnements, choisissez Créer un abonnement.
4. Sur la page Créer un abonnement, dans la section Détails, procédez comme suit :
    a. Pour ARN de rubrique, choisissez l'Amazon Resource Name (ARN) d'une rubrique standard.
    b. Pour Protocol, choisissez Firehose.
    c. Pour Endpoint, choisissez l'ARN d'un flux de diffusion Firehose qui peut recevoir des notifications d'Amazon SNS.
    d. Pour l'ARN du rôle d'abonnement, spécifiez l'ARN du rôle AWS Identity and Access Management (IAM) que vous avez créé pour écrire dans les flux de diffusion Firehose. Pour plus d'informations, consultez Conditions préalables à l'abonnement aux flux de diffusion Firehose aux rubriques Amazon SNS.
    e. (Facultatif) Pour supprimer les métadonnées Amazon SNS des messages publiés, choisissez Activer la diffusion brute des messages. Pour en savoir plus, consultez la section Remise des messages bruts Amazon SNS.
5. (Facultatif) Pour configurer une politique de filtre, développez la section Politique de filtre d'abonnement. Pour de plus amples informations, consultez Stratégies de filtre d'abonnement Amazon SNS.
6. (Facultatif) Pour configurer une file d'attente de lettres mortes pour l'abonnement, développez la section Politique de reconduite (File d'attente de lettres mortes). Pour de plus amples informations, consultez la section Files d'attente de lettres mortes (DLQ)) d'Amazon SNS.
7. Choisissez Créer un abonnement.

La console crée l'abonnement et ouvre la page Détails de l'abonnement.

Utilisation des destinations du flux de diffusion

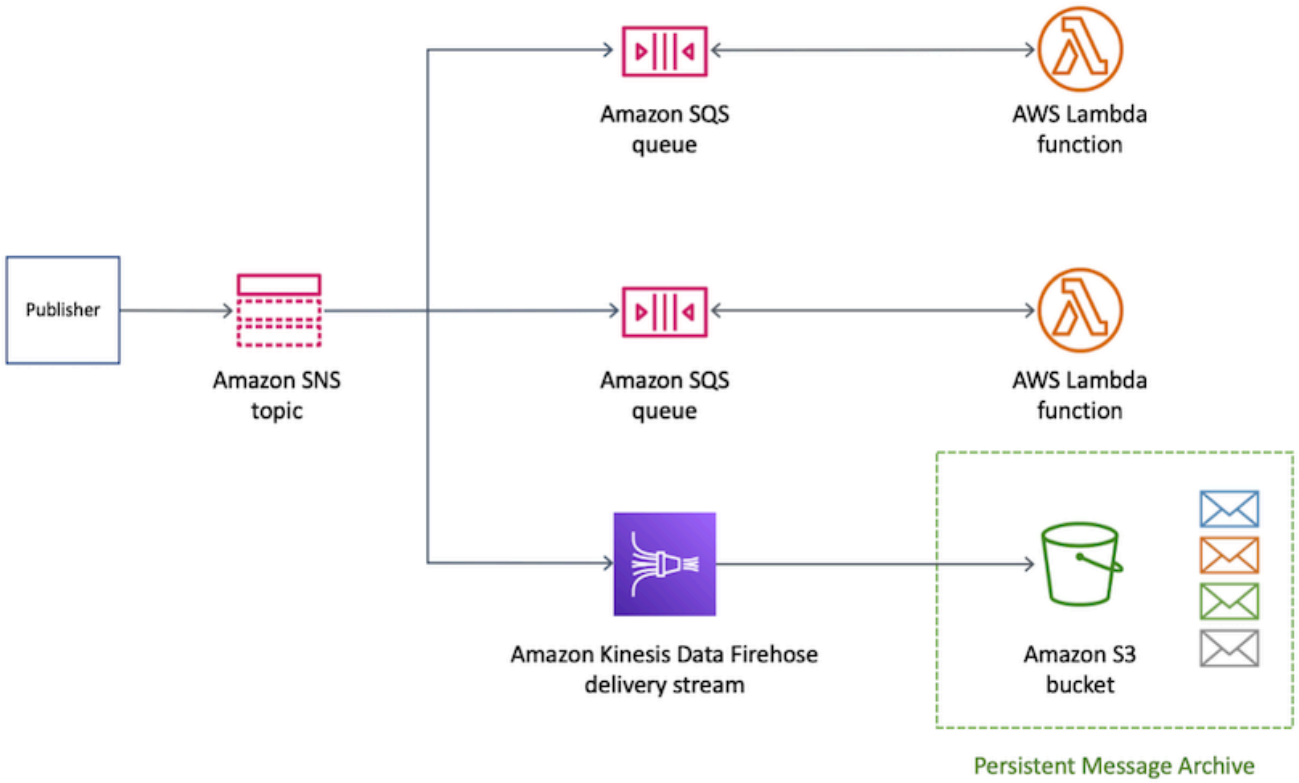
Grâce aux flux de diffusion Amazon Data Firehose, vous pouvez envoyer des messages à des points de terminaison supplémentaires. Cette section décrit comment utiliser les destinations prises en charge.

Rubriques

- Destinations Amazon S3
- OpenSearch Destinations de service
- Destinations Amazon Redshift
- Destinations HTTP

Destinations Amazon S3

Cette section fournit des informations sur les flux de diffusion Amazon Data Firehose qui publient des données sur Amazon Simple Storage Service (Amazon S3).



Rubriques

- Format du message archivé pour les destinations Amazon S3

- Analyser les messages des destinations Amazon S3

Format du message archivé pour les destinations Amazon S3

L'exemple suivant présente une notification Amazon SNS envoyée à un compartiment Amazon Simple Storage Service (Amazon S3), à l'aide de retraits pour plus de lisibilité.

> **Note**
>
> Dans cet exemple, la diffusion des messages bruts est désactivée pour le message publié. Lorsque la diffusion des messages bruts est désactivée, Amazon SNS ajoute des métadonnées JSON au message, y compris les propriétés suivantes :
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Pour en savoir plus sur la diffusion brute, consultez la section Remise des messages bruts 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"
        }
    }
}
```

L'exemple suivant montre trois messages SNS envoyés via un flux de diffusion Amazon Data Firehose vers le même compartiment Amazon S3. La mise en mémoire tampon est prise en compte, et les sauts de ligne séparent les messages.

```
{"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"}
```

Analyser les messages des destinations Amazon S3

Cette page explique comment analyser les messages Amazon SNS envoyés via les flux de livraison Amazon Data Firehose vers des destinations Amazon Simple Storage Service (Amazon S3).

Pour analyser les messages SNS envoyés via les flux de diffusion Firehose vers des destinations Amazon S3

1. Configurez vos ressources Amazon S3. Pour obtenir des instructions, consultez la section Créer un compartiment dans le Guide de mise en route Amazon Simple Storage Service et Utilisation des compartiments Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.
2. Configurez votre flux de diffusion. Pour obtenir des instructions, consultez Choisir Amazon S3 pour votre destination dans le manuel Amazon Data Firehose Developer Guide.
3. Utiliser Amazon Athena pour interroger les objets Amazon S3 à l'aide de SQL standard. Pour en savoir plus, consultez la section Mise en route dans le Guide de l'utilisateur Amazon Athena.

Exemple de requête

Pour cet exemple de requête, supposons ce qui suit :

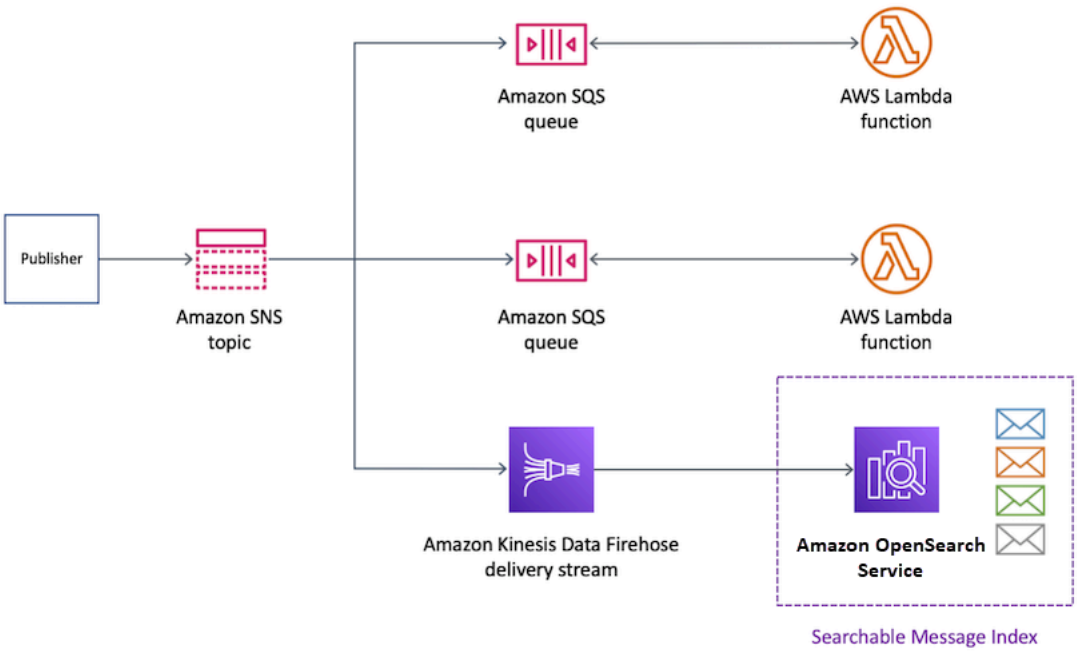
- Les messages sont stockés dans le tableau `notifications` dans le schéma `default`.
- Le tableau `notifications` comprend une colonne `timestamp` avec un type de `string`.

La requête suivante renvoie tous les messages SNS reçus dans la plage de dates spécifiée :

```
SELECT *
FROM default.notifications
WHERE from_iso8601_timestamp(timestamp) BETWEEN TIMESTAMP '2020-12-01 00:00:00' AND
 TIMESTAMP '2020-12-02 00:00:00';
```

OpenSearch Destinations de service

Cette section fournit des informations sur les flux de diffusion Amazon Data Firehose qui publient des données sur Amazon OpenSearch Service (OpenSearch Service).



Rubriques

- Format des messages archivés dans les index OpenSearch Service
- Analyse des messages pour les destinations OpenSearch de service

Format des messages archivés dans les index OpenSearch Service

L'exemple suivant présente une notification Amazon SNS envoyée à un index Amazon OpenSearch Service nommé `my-index`. Cet index a un champ de filtre temporel sur le champ `Timestamp`. La notification SNS est placée dans la propriété `_source` de la charge utile.

> **Note**
>
> Dans cet exemple, la diffusion des messages bruts est désactivée pour le message publié. Lorsque la diffusion des messages bruts est désactivée, Amazon SNS ajoute des métadonnées JSON au message, y compris les propriétés suivantes :
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`

- `UnsubscribeURL`
- `MessageAttributes`

Pour en savoir plus sur la diffusion brute, consultez la section Remise des messages bruts Amazon SNS.


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

Analyse des messages pour les destinations OpenSearch de service

Cette page explique comment analyser les messages Amazon SNS envoyés via les flux de livraison Amazon Data Firehose vers des destinations Amazon OpenSearch Service (Service)OpenSearch .

Pour analyser les messages SNS envoyés via les flux OpenSearch de diffusion Firehose vers les destinations du service

1. Configurez vos ressources OpenSearch de service. Pour obtenir des instructions, consultez Getting Started with Amazon OpenSearch Service dans le manuel Amazon OpenSearch Service Developer Guide.
2. Configurez votre flux de diffusion. Pour obtenir des instructions, consultez la section Choisir le OpenSearch service pour votre destination dans le manuel Amazon Data Firehose Developer Guide.
3. Exécutez une requête à l'aide des requêtes OpenSearch Service et de Kibana. Pour plus d'informations, consultez Étape 3 : Rechercher des documents dans un domaine de OpenSearch service et Kibana dans le manuel Amazon OpenSearch Service Developer Guide.

Exemple de requête

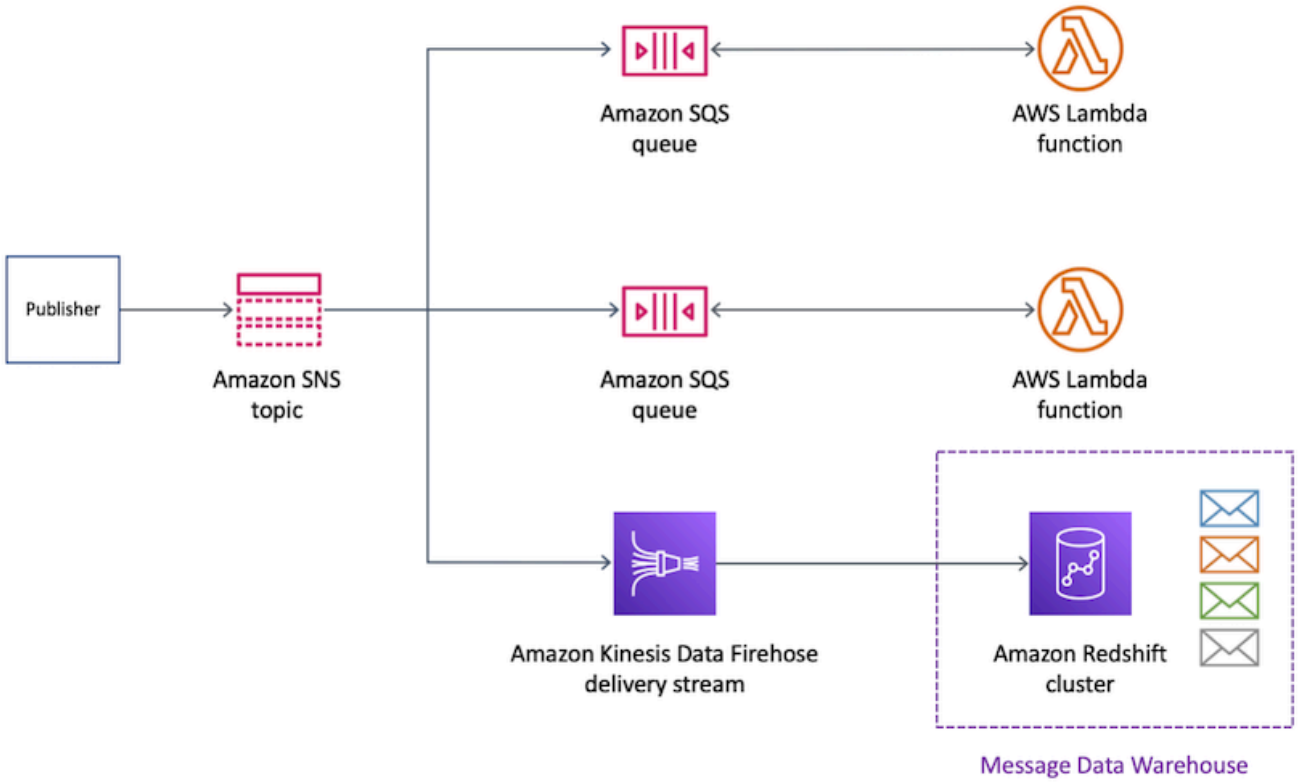
L'exemple suivant interroge l'index `my-index` pour tous les messages SNS reçus dans la plage de dates spécifiée :

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

```
}
```

Destinations Amazon Redshift

Cette section explique comment transférer les notifications Amazon SNS vers un flux de diffusion Amazon Data Firehose qui publie des données sur Amazon Redshift. Avec cette configuration, vous pouvez vous connecter à la base de données Amazon Redshift et utiliser un outil de requêtes SQL afin d'interroger la base de données pour les messages Amazon SNS qui répondent à certains critères.



Rubriques

- Structure du tableau d'archivage pour les destinations Amazon Redshift
- Analysez les messages des destinations Amazon Redshift

Structure du tableau d'archivage pour les destinations Amazon Redshift

Pour les points de terminaison Amazon Redshift, les messages Amazon SNS publiés sont archivés sous forme de lignes dans un tableau. En voici un exemple.

> **Note**
>
> Dans cet exemple, la diffusion des messages bruts est désactivée pour le message publié. Lorsque la diffusion des messages bruts est désactivée, Amazon SNS ajoute des métadonnées JSON au message, y compris les propriétés suivantes :
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Message`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Pour en savoir plus sur la diffusion brute, consultez la section Remise des messages bruts Amazon SNS.
> Bien qu'Amazon SNS ajoute des propriétés au message à l'aide de la majuscule indiquée dans cette liste, les noms de colonnes des tableaux Amazon Redshift sont en minuscules. Pour transformer les métadonnées JSON pour le point de terminaison Amazon Redshift, vous pouvez utiliser la commande SQL `COPY`. Pour en savoir plus, veuillez consulter la section Copier à partir d'exemples JSON et Charger à partir de données JSON à l'aide de l'option 'Ignorer automatiquement' dans le Guide du développeur de base de données Amazon Redshift.

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notification	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Objet de l'exemple	Exemple de message	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?	{\"my_attribute\":{\"Type\":\"String\",\"Valu

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	e\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notification	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Objet de l'exemple 2	Exemple de message 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\": {\"Type\": \"String\", \"Value\": \"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notification	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Objet de l'exemple 3	Exemple de message 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Pour en savoir plus sur la diffusion des notifications aux points de terminaison Amazon Redshift, consultez la section Destinations Amazon Redshift.

Analysez les messages des destinations Amazon Redshift

Cette page explique comment analyser les messages Amazon SNS envoyés via les flux de livraison Amazon Data Firehose vers des destinations Amazon Redshift.

Pour analyser les messages SNS envoyés via les flux de diffusion Firehose vers des destinations Amazon Redshift

1. Configurez vos ressources Amazon Redshift. Pour obtenir des instructions, consultez la section de Mise en route avec Amazon Redshift dans le Guide de mise en route d'Amazon Redshift.
2. Configurez votre flux de diffusion. Pour obtenir des instructions, consultez Choisir Amazon Redshift pour votre destination dans le manuel Amazon Data Firehose Developer Guide.
3. Exécuter une requête. Pour plus d'informations, consultez Interrogation d'une base de données à l'aide de l'éditeur de requêtes dans le Guide de gestion Amazon Redshift.

Exemple de requête

Pour cet exemple de requête, supposons ce qui suit :

- Les messages sont stockés dans le tableau `notifications` dans le schéma `public` par défaut.
- La propriété `Timestamp` du message SNS est stockée dans la colonne `timestamp` du tableau avec un type de données de colonne de `timestamptz`.

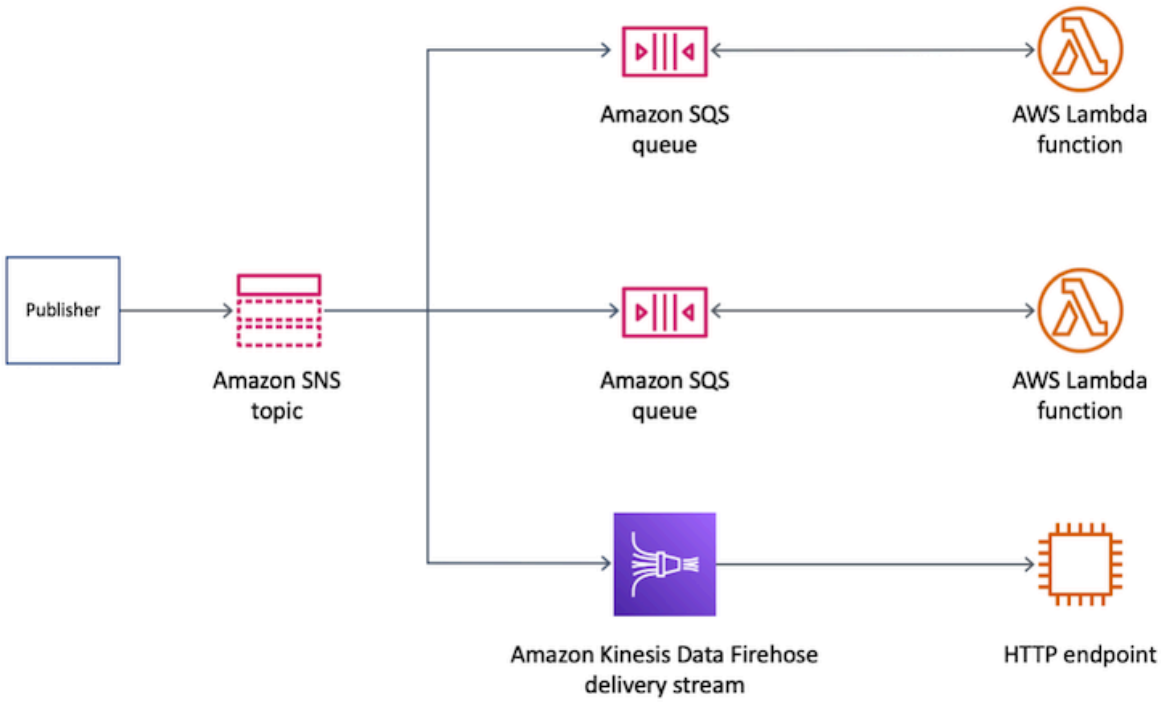
> **Note**
>
> Pour transformer les métadonnées JSON pour le point de terminaison Amazon Redshift, vous pouvez utiliser la commande SQL `COPY`. Pour en savoir plus, consultez la section Copier à partir d'exemples JSON et Charger à partir de données JSON à l'aide de l'option 'ignorer automatiquement' dans le Guide du développeur de base de données Amazon Redshift.

La requête suivante renvoie tous les messages SNS reçus dans la plage de dates spécifiée :

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

Destinations HTTP

Cette section fournit des informations sur les flux de diffusion Amazon Data Firehose qui publient des données sur des points de terminaison HTTP.



Rubriques

- Format des message diffusés pour les destinations HTTP

Format des message diffusés pour les destinations HTTP

Voici un exemple de corps de requête HTTP POST provenant d'Amazon SNS qu'un flux de diffusion Amazon Data Firehose peut envoyer au point de terminaison HTTP. La notification SNS est encodée en tant que charge utile base64 dans la propriété `records`.

> **Note**
>
> Dans cet exemple, la diffusion des messages bruts est désactivée pour le message publié. Pour en savoir plus sur la diffusion brute, consultez la section Remise des messages bruts Amazon SNS.

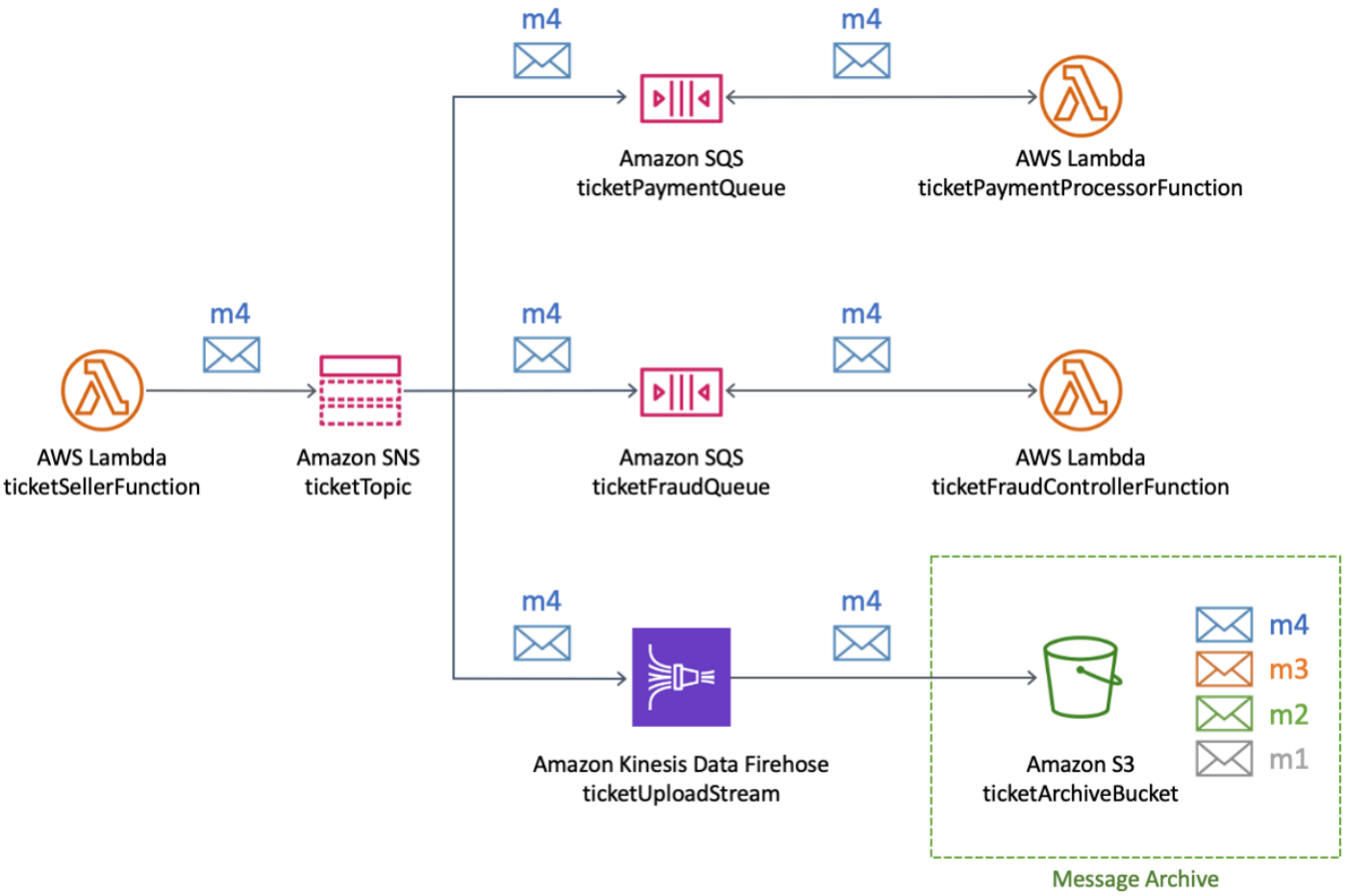
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
```

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


Exemple de cas d'utilisation pour l'archivage et l'analyse des messages

Cette section fournit un didacticiel sur un cas d'utilisation courant pour l'archivage et l'analyse des messages Amazon SNS.

Le cadre de ce cas d'utilisation est une plateforme de billetterie d'une compagnie aérienne qui opère dans un environnement réglementé. La plateforme est soumise à un cadre de conformité qui oblige la compagnie à archiver toutes les ventes de billets pendant au moins cinq ans. Pour atteindre l'objectif de conformité en matière de conservation des données, l'entreprise abonne un flux de diffusion Amazon Data Firehose à une rubrique SNS existante. La destination du flux de diffusion est un compartiment Amazon Simple Storage Service (Amazon S3). Grâce à cette configuration, tous les événements publiés dans la rubrique SNS sont archivés dans le compartiment Amazon S3. Le diagramme suivant illustre l'architecture de cette configuration :



Pour exécuter des analyses et obtenir des informations sur les ventes de billets, ladite compagnie exécute des requêtes SQL à l'aide d'Amazon Athena. Par exemple, elle peut émettre une requête pour en savoir plus sur les destinations les plus populaires et les voyageurs les plus fréquents.

Pour créer les ressources AWS pour ce cas d'utilisation, vous pouvez utiliser le modèle AWS Management Console ou AWS CloudFormation.

Rubriques

- Création des ressources initiales
- Création du flux de diffusion Firehose
- Abonnement du flux de diffusion Firehose à la rubrique Amazon SNS
- Test et interrogation de la configuration
- Utilisation d'un modèle AWS CloudFormation

Création des ressources initiales

Cette page explique comment créer les ressources suivantes pour l'exemple d'archivage des messages et d'analyse de cas d'utilisation :

- Un compartiment Amazon Simple Storage Service (Amazon S3)
- Deux files d'attente Amazon Simple Queue Service (Amazon SQS)
- Une rubrique Amazon SNS
- Deux abonnements Amazon SQS à la rubrique Amazon SNS

Pour créer les ressources initiales

1. Créez un compartiment Amazon S3 :

   a. Ouvrez la console Amazon S3.

   b. Choisissez Créer un compartiment.

   c. Pour Nom de compartiment, saisissez un nom unique dans le monde. Conservez les valeurs par défaut des autres champs.

   d. Choisissez Créer un compartiment.

   Pour de plus amples informations sur les compartiments Amazon S3, voir Créer un compartiment dans le Guide de l'utilisateur du service Amazon Simple Storage et Travailler avec les compartiments Amazon S3 dans le Guide de l'utilisateur du service Amazon Simple Storage.

2. Créez les deux files d'attente Amazon SQS :

   a. Ouvrez la console Amazon SQS.

   b. Choisissez Créez une file d'attente.

   c. Pour Type, choisissez Standard.

   d. Pour Nom, saisissez **`ticketPaymentQueue`**.

   e. Sous Politique d'accès, pour Choisir la méthode, choisissez Avancé.

   f. Dans la zone politique JSON, collez la politique suivante :

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

Dans cette politique d'accès, remplacez le numéro Compte AWS (*123456789012*) par le vôtre, et modifiez la région AWS (*us-east-1*) en conséquence.

g. Choisissez Créer une file d'attente.

h. Répétez ces étapes pour créer une deuxième file d'attente SQS nommée **ticketFraudQueue**.

Pour en savoir plus sur la création de files d'attente SQS, veuillez consulter la section de Création d'une file d'attente Amazon SQS (console) dans le Guide du développeur Amazon Simple Queue Service.

3. Création de la rubrique SNS :

   a. Ouvrez la page Rubriques de la console Amazon SNS.

   b. Choisissez Créer une rubrique.

   c. Sous Détails, pour Type, choisissez Standard.

   d. Pour Nom, saisissez **ticketTopic**.

   e. Choisissez Créer une rubrique.

   Pour en savoir plus sur la création de rubriques SNS, veuillez consulter la section Création d'une rubrique Amazon SNS.

4. Abonner deux files d'attente SQS à la rubrique SNS :

a. Dans la console Amazon SNS, sur la page de détails de la rubrique ticketTopic, choisissez Créer un abonnement.
b. Sous Détails, pour Protocole, choisissez Amazon SQS.
c. Pour Point de terminaison, choisissez l'Amazon Resource Name (ARN) de la file d'attente ticketPaymentQueue.
d. Choisissez Créer un abonnement.
e. Répétez ces étapes pour créer un second abonnement à l'aide de l'ARN de la file d'attente ticketFraudQueue.

   Pour en savoir plus sur l'abonnement aux rubriques SNS, consultez la section Abonnement à une rubrique Amazon SNS. Vous pouvez également abonner des files d'attente SQS aux rubriques SNS à partir de la console Amazon SQS. Pour en savoir plus, consultez la section Abonnement d'une file d'attente Amazon SQS à une rubrique Amazon SNS (console) dans le Guide du développeur Amazon Simple Queue Service.

Vous avez créé les ressources initiales pour cet exemple de cas d'utilisation. Pour continuer, consultez la section Création du flux de diffusion Firehose.

Création du flux de diffusion Firehose

Cette page explique comment créer le flux de diffusion Amazon Data Firehose pour l'exemple d'utilisation de l'archivage et de l'analyse des messages.

Pour créer le flux de diffusion Firehose

1. Ouvrez la console de services Amazon Kinesis.
2. Choisissez Firehose, puis Create delivery stream.
3. Sur la page Nouveau flux de diffusion, pour Nom du flux de diffusion, saisissez **`ticketUploadStream`**, puis choisissez Suivant.
4. Sur la page Traiter les registres, choisissez Suivant.
5. Sur la page Choisir une destination, procédez comme suit :

   a. Pour Destination, choisissez Amazon S3.
   b. Sous Destination S3, pour Compartiment S3, choisissez le compartiment S3 que vous avez initialement créé.
   c. Choisissez Suivant.

6. Sur la page Configuration des paramètres, pour Conditions de tampon S3, procédez comme suit :

   - Pour Taille du tampon, saisissez **1**.
   - Pour Intervalle tampon, saisissez **60**.

   L'utilisation de ces valeurs pour le tampon Amazon S3 vous permet de tester rapidement la configuration. La première condition qui est satisfaite déclenche la diffusion des données au compartiment S3.
7. Dans la page Configuration des paramètres, pour Autorisations, choisissez de créer un rôle (IAM) AWS Identity and Access Management avec les autorisations requises affectées automatiquement. Sélectionnez ensuite Suivant.
8. Sur la page Vérification, choisissez Créer un flux de diffusion.
9. Sur la page des flux de diffusion de Kinesis Data Firehose, sélectionnez le flux de diffusion que vous venez ticketUploadStreamde créer (). Sur la page Détails, notez l'Amazon Resource Name (ARN) du flux pour plus tard.

Pour plus d'informations sur la création de flux de diffusion, consultez la section Création d'un flux de diffusion Amazon Data Firehose dans le manuel du développeur Amazon Data Firehose. Pour en savoir plus sur la création d'un rôle IAM, consultez la section de Création d'un rôle pour la délégation d'autorisations à un AWSservice du Guide de l'utilisateur IAM.

Vous avez créé le flux de diffusion Firehose avec les autorisations requises. Pour continuer, consultez la section Abonnement du flux de diffusion Firehose à la rubrique Amazon SNS.

Abonnement du flux de diffusion Firehose à la rubrique Amazon SNS

Cette page explique comment créer ce qui suit pour exemple d'archivage des messages et d'analyse de cas d'utilisation :

- Rôle AWS Identity and Access Management (IAM) qui permet à l'abonnement Amazon SNS d'enregistrer le flux de diffusion Amazon Data Firehose
- L'abonnement au stream de diffusion Firehose à la rubrique SNS

Pour créer le rôle IAM pour l'abonnement Amazon SNS

1. Ouvrez la page Rôles de la console IAM.

2. Choisissez Créer un rôle.
3. Pour Sélectionner le type d'entité de confiance, choisissez le service AWS.
4. Pour Choix d'un cas d'utilisation, choisissez SNS. Choisissez ensuite Suivant : Autorisations.
5. Choisissez Suivant : Balises.
6. Choisissez Suivant : Vérification.
7. Sur la page Vérification, pour Nom du rôle, saisissez **`ticketUploadStreamSubscriptionRole`**. Puis choisissez Create role (Créer un rôle).
8. Lorsque le rôle est créé, choisissez son nom (ticketUploadStreamSubscriptionRole).
9. Sur la page de Résumé du rôle, choisissez Ajouter une politique en ligne.
10. Sur la page Créer une politique, choisissez l'onglet JSON, puis collez la politique JSON suivante dans la zone de texte :

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

    Dans cette politique, remplacez le numéro Compte AWS (*`123456789012`*) par le vôtre, et modifiez la région AWS(*`us-east-1`*) en conséquence.
11. Choisissez Vérifier une politique.
12. Sur la page Vérifier une politique, pour Nom, saisissez **`FirehoseSnsPolicy`**. Sélectionnez ensuite Créer une politique.

13. Sur la page de Résumé du rôle, notez l'ARN de rôle pour plus tard.

Pour en savoir plus sur la création d'un rôle IAM, consultez la section de Création d'un rôle pour la délégation d'autorisations à un AWSservice du Guide de l'utilisateur IAM.

Pour abonner le stream de diffusion Firehose à la rubrique SNS

1. Ouvrez la page Rubriques de la console Amazon SNS.
2. Sous l'onglet Abonnements, choisissez Créer un abonnement.
3. Dans Détails, dans le champ Protocole, sélectionnez Amazon Data Firehose.
4. Pour Endpoint, entrez le nom de ressource Amazon (ARN) du flux de ticketUploadStreamdiffusion que vous avez créé précédemment. Par exemple, saisissez **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Pour l'ARN du rôle d'abonnement, entrez l'ARN du rôle ticketUploadStreamSubscriptionRoleIAM que vous avez créé précédemment. Par exemple, saisissez **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Sélectionnez la case Activer la diffusion brute des messages.
7. Choisissez Créer un abonnement.

Vous avez créé le rôle IAM et l'abonnement à la rubrique SNS. Pour continuer, consultez la section Test et interrogation de la configuration.

Test et interrogation de la configuration

Cette page explique comment tester l'exemple d'archivage des messages et d'analyse de cas d'utilisation en publiant un message dans la rubrique Amazon SNS. Les instructions comprennent un exemple de requête que vous pouvez exécuter et adapter à vos propres besoins.

Pour tester votre configuration

1. Ouvrez la page Rubriques de la console Amazon SNS.
2. Cliquez sur la rubrique **`ticketTopic`**.
3. Choisissez Publier le message.
4. Sur la page Publier un message dans une rubrique, saisissez ce qui suit pour le corps du message. Ajoutez un caractère de saut de ligne à la fin du message.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

Conservez toutes valeurs par défaut des autres options.

5. Choisissez Publier le message.

   Pour en savoir plus sur la publication de messages, consultez la section Publication de messages Amazon SNS.

6. Après l'intervalle de flux de diffusion de 60 secondes, ouvrez la console Amazon Simple Storage Service (Amazon S3) et choisissez le compartiment Amazon S3 que vous avez initialement créé.

   Le message publié s'affiche dans le compartiment.

Pour des requêtes sur des données

1. Ouvrez la console Amazon Athena.
2. Exécuter une requête.

   Par exemple, supposons que le tableau `notifications` dans le schéma `default`contient les données suivantes :

```
{"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"}
```

   Pour rechercher la destination la plus importante, exécutez la requête suivante :

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

Pour rechercher les tickets vendus au cours d'une période et d'une plage d'heures spécifique, exécutez une requête comme suit :

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

Vous pouvez adapter les deux exemples de requêtes à vos propres besoins. Pour en savoir plus sur l'utilisation d'Athena pour exécuter des requêtes, consultez la section de Mise en route dans le Guide de l'utilisateur Amazon Athena.

Nettoyage

Pour éviter d'encourir des frais d'utilisation après avoir terminé le test, supprimez les ressources suivantes que vous avez créées pendant le didacticiel :

- Abonnements Amazon SNS
- Rubrique Amazon SNS
- Files d'attente Amazon Simple Queue Service (Amazon SQS)
- Compartiment Amazon S3
- Flux de livraison d'Amazon Data Firehose
- AWS Identity and Access Management avec des rôles (IAM) et des politiques

Utilisation d'un modèle AWS CloudFormation

Pour automatiser le déploiement de l'exemple d'archivage des messages et d'analyse de cas d'utilisation d'Amazon SNS, vous pouvez utiliser le modèle YAML suivant :

```
---
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
```

Diffusion dans les fonctions Lambda

Amazon SNS et AWS Lambda sont intégrés afin que vous puissiez appeler des fonctions Lambda avec des notifications Amazon SNS. Lorsqu'un message est publié dans une rubrique SNS à laquelle est abonnée une fonction Lambda, cette dernière est appelée avec la charge utile du message publié. La fonction Lambda reçoit la charge utile du message sous forme de paramètre d'entrée et peut manipuler les informations du message, publier le message dans d'autres rubriques SNS, ou envoyer le message à d'autres services AWS.

Amazon SNS prend également en charge les attributs de l'état de diffusion du message pour les notifications de message envoyées aux points de terminaison Lambda. Pour de plus amples informations, consultez la section Statut de distribution de message Amazon SNS.

Prerequisites

Pour appeler des fonctions Lambda à l'aide de notifications Amazon SNS, vous devez disposer des éléments suivants :

- Fonction Lambda
- Rubrique Amazon SNS

Pour de plus amples informations sur la création d'une fonction Lambda à utiliser avec Amazon SNS, consultez la section Utilisation de Lambda avec Amazon SNS. Pour de plus amples informations sur la création d'une rubrique Amazon SNS, consultez la section de Créer une rubrique.

Lorsque vous utilisez Amazon SNS pour envoyer des messages à partir de régions d'adhésion vers des régions activées par défaut, vous devez modifier la stratégie créée dans la fonction AWS Lambda en remplaçant le principal `sns.amazonaws.com` par `sns.<opt-in-region>.amazonaws.com`.

Par exemple, si vous souhaitez abonner une fonction Lambda dans la région USA Est (Virginie du Nord) à une rubrique SNS en Asie-Pacifique (Hong Kong), remplacez le principal dans la stratégie de la fonction AWS Lambda par `sns.ap-east-1.amazonaws.com`. Les régions d'adhésion comprennent toutes les régions lancées après le 20 mars 2019, notamment Asie-Pacifique (Hong Kong), Moyen-Orient (Bahreïn), UE (Milan) et Afrique (Le Cap). Les régions lancées avant le 20 mars 2019 sont activées par défaut.

> **Note**
>
> AWS ne prend pas en charge la diffusion entre régions à AWS Lambda à partir d'une région activée par défaut vers une région d'adhésion. En outre, le transfert de messages SNS entre régions depuis des régions d'adhésion vers d'autres régions d'adhésion n'est pas pris en charge.

Abonnement d'une fonction à une rubrique

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sur la page Rubriques, choisissez une rubrique.
4. Dans la section Abonnements, choisissez Créer un abonnement.
5. Sur la page Créer un abonnement, dans la section Détails, procédez comme suit :
   a. Vérifiez la valeur choisie pour ARN de la rubrique.
   b. Pour Protocole, choisissez AWS Lambda.
   c. Pour Point de terminaison saisissez l'ARN d'une fonction.

d. Choisissez Créer un abonnement.

Lorsqu'un message est publié dans une rubrique SNS à laquelle est abonnée une fonction Lambda, cette dernière est appelée avec la charge utile du message publié. Pour plus d'informations sur l'utilisation d'AWS LambdaAmazon SNS, y compris un didacticiel, consultez la section d'Utilisation d'AWS LambdaAmazon SNS.

Distribution ramifiée vers des files d'attente Amazon SQS

Amazon SNS travaille en étroite collaboration avec Amazon Simple Queue Service (Amazon SQS). Ces services offrent différents avantages pour les développeurs. Amazon SNS permet aux applications d'envoyer des messages à caractère urgent à plusieurs abonnés via un mécanisme « push », éliminant ainsi le besoin de vérifier ou d'interroger (« poll ») périodiquement les mises à jour. Amazon SQS est un service de file d'attente de messages utilisé par des applications distribuées pour échanger des messages via un modèle d'interrogation. Il peut être utilisé pour découpler des composants d'envoi et de réception, sans que chaque composant soit disponible simultanément. L'utilisation conjointe d'Amazon SNS et d'Amazon SQS permet d'envoyer les messages aux applications qui exigent une notification immédiate d'un évènement et ils peuvent également rester dans une file d'attente Amazon SQS, afin que d'autres applications puissent les traiter ultérieurement.

Lorsque vous abonnez une file d'attente Amazon SQS à une rubrique Amazon SNS, vous publiez un message dans cette rubrique et Amazon SNS envoie un message Amazon SQS à la file d'attente abonnée. Le message Amazon SQS contient l'objet et le message publiés dans la rubrique, ainsi que les métadonnées relatives au message, dans un document JSON. Le message Amazon SQS se présente comme le document JSON suivant.

```
{
   "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"
}
```

Abonnement d'une file d'attente Amazon SQS à une rubrique Amazon SNS

Pour permettre à une rubrique Amazon SNS d'envoyer des messages à une file d'attente Amazon SQS, effectuez l'une des opérations suivantes :

- Utilisez la console Amazon SQS, ce qui simplifie le processus. Pour plus d'informations, consultez Abonnement d'une file d'attente Amazon SQS à une rubrique Amazon SNS dans le Guide du développeur Amazon Simple Queue Service.
- Procédez comme suit :
  1. Obtenez l'Amazon Resource Name (ARN) de la file d'attente à laquelle vous souhaitez envoyer des messages et de la rubrique à laquelle vous souhaitez abonner la file d'attente.
  2. Accordez l'autorisation `sqs:SendMessage` à la rubrique Amazon SNS afin qu'elle puisse envoyer des messages à la file d'attente.
  3. Abonnez la file d'attente à la rubrique Amazon SNS.
  4. Accordez aux utilisateurs IAM ou à Comptes AWS les autorisations d'effectuer une publication dans la rubrique Amazon SNS et de lire les messages à partir de la file d'attente Amazon SQS.
  5. Testez la procédure en publiant un message dans la rubrique et en le lisant à partir de la file d'attente.

Pour découvrir comment configurer une rubrique pour envoyer des messages à une file d'attente située dans un autre compte AWS, consultez Envoi de messages Amazon SNS à une file d'attente Amazon SQS d'un autre compte.

Pour obtenir un modèle AWS CloudFormation créant une rubrique qui envoie des messages à deux files d'attente, consultez Utilisation d'un modèle AWS CloudFormation pour créer une rubrique qui envoie des messages à des files d'attente Amazon SQS.

Étape 1 : obtenir l'ARN de la file d'attente et de la rubrique

Lorsque vous abonnez une file d'attente à votre rubrique, vous avez besoin d'une copie de l'ARN de la file d'attente. De même, lorsque vous accordez à la rubrique l'autorisation d'envoyer des messages à la file d'attente, vous avez besoin d'une copie de l'ARN de la rubrique.

Pour obtenir l'ARN de la file d'attente, vous pouvez utiliser la console Amazon SQS ou l'action d'API GetQueueAttributes.

Pour obtenir l'ARN de la file d'attente à partir de la console Amazon SQS

1. Connectez-vous à l'AWS Management Console et ouvrez la console Amazon SQS à l'adresse https://console.aws.amazon.com/sqs/.
2. Cochez la case correspondant à la file d'attente dont vous souhaitez obtenir l'ARN.
3. Sur l'onglet Détails, copiez la valeur de l'ARN afin de pouvoir l'utiliser pour l'abonnement à la rubrique Amazon SNS.

Pour obtenir l'ARN de la rubrique, vous pouvez utiliser la console Amazon SNS, la commande `sns-get-topic-attributes` ou l'action d'API `GetQueueAttributes`.

Pour obtenir l'ARN de la rubrique à partir de la console Amazon SNS

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, sélectionnez la rubrique dont vous souhaitez obtenir l'ARN.
3. Dans la section Détails, copiez la valeur du champ ARN afin de pouvoir l'utiliser pour accorder à la rubrique Amazon SNS l'autorisation d'envoyer des messages à la file d'attente.

Étape 2 : autoriser la rubrique Amazon SNS à envoyer des messages à la file d'attente Amazon SQS

Pour qu'une rubrique Amazon SNS soit en mesure d'envoyer des messages à une file d'attente, vous devez définir une politique sur la file d'attente qui permette à la rubrique Amazon SNS d'effectuer l'action `sqs:SendMessage`.

Avant d'abonner une file d'attente à une rubrique, vous avez besoin d'une rubrique et d'une file d'attente. Si vous n'avez pas encore créé de rubrique ou de file d'attente, faites-le maintenant. Pour plus d'informations, consultez Création d'une rubrique et Création d'une file d'attente dans le Guide du développeur Amazon Simple Queue Service.

Pour définir une politique sur une file d'attente, vous pouvez utiliser la console Amazon SQS ou l'action d'API SetQueueAttributes. Avant de commencer, assurez-vous que vous disposez de l'ARN de la rubrique que vous souhaitez autoriser à envoyer des messages à la file d'attente. Si vous

abonnez une file d'attente à plusieurs rubriques, votre stratégie doit contenir un élément `Statement` pour chaque rubrique.

Pour définir une politique SendMessage sur une file d'attente à l'aide de la console Amazon SQS

1. Connectez-vous à l'AWS Management Console et ouvrez la console Amazon SQS à l'adresse https://console.aws.amazon.com/sqs/.
2. Cochez la case de la file d'attente dont vous souhaitez définir la politique, choisissez l'onglet Politique d'accès, puis choisissez Modifier.
3. Dans la Politique d'accès, définissez qui peut accéder à votre file d'attente.

   - Ajoutez une condition qui autorise l'action pour la rubrique.
   - Définissez `Principal` en tant que service Amazon SNS, comme indiqué dans l'exemple ci-dessous.
   - Utiliser les clés de condition globales `aws:SourceArn` ou `aws:SourceAccount` pour se protéger contre le scénario Député confus. Pour utiliser ces clés de condition, définissez la valeur de l'ARN de votre sujet. Si votre file d'attente est abonnée à plusieurs rubriques, vous pouvez utiliser `aws:SourceAccount` à la place.

   Par exemple, la politique suivante permet à MyTopic d'envoyer des messages à 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"
        }
      }
    }
  ]
}
```

Étape 3 : abonner la file d'attente à la rubrique Amazon SNS

Pour envoyer des messages à une file d'attente via une rubrique, vous devez abonner la file d'attente à la rubrique Amazon SNS. Spécifiez la file d'attente à l'aide de son ARN. Pour réaliser un abonnement à une rubrique, vous pouvez utiliser la console Amazon SNS, la CLI `sns-subscribe` de commande ou l'action d'API `Subscribe`. Avant de commencer, assurez-vous que vous disposez de l'ARN de la file d'attente que vous souhaitez abonner.

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sur la page Rubriques, choisissez une rubrique.
4. Sur la page ***`MaRubrique`***, sur la page Souscriptions, choisissez Créer un abonnement.
5. Sur la page Créer un abonnement, dans la section Détails , procédez comme suit :

    a. Vérifiez la valeur de ARN de la rubrique.
    b. Pour Protocole, choisissez Amazon SQS.
    c. Pour Point de terminaison, saisissez l'ARN d'une file d'attente Amazon SQS.
    d. Choisissez Créer un abonnement.

    Lorsque l'abonnement est confirmé, le champ ID de l'abonnement de votre nouvel abonnement affiche son ID d'abonnement. Si le propriétaire de la file d'attente crée l'abonnement, ce dernier est automatiquement confirmé et l'abonnement doit être actif presque immédiatement.

    En général, vous abonnez votre propre file d'attente à votre propre rubrique dans votre propre compte. Cependant, vous pouvez également abonner une file d'attente d'un autre compte à votre rubrique. Si l'utilisateur qui crée l'abonnement n'est pas le propriétaire de la file d'attente (par exemple, si un utilisateur du compte A abonne une file d'attente du compte B à une rubrique du compte A), l'abonnement doit être confirmé. Pour plus d'informations sur l'abonnement d'une file d'attente d'un autre compte et la confirmation de l'abonnement, consultez la page Envoi de messages Amazon SNS à une file d'attente Amazon SQS d'un autre compte.

Étape 4 : autoriser les utilisateurs à accéder aux actions appropriées sur la rubrique et la file d'attente

Vous devez utiliser AWS Identity and Access Management (IAM) pour autoriser uniquement les utilisateurs appropriés à effectuer une publication dans la rubrique Amazon SNS et à lire/supprimer

des messages à partir de la file d'attente Amazon SQS. Pour plus d'informations sur le contrôle des actions sur les rubriques et les files d'attente pour les utilisateurs IAM, consultezUtilisation de politiques basées sur l'identité avec Amazon SNS et Gestion des identités et des accès dans Amazon SQS dans le Guide du développeur Amazon Simple Queue Service.

Il existe deux façons de contrôler l'accès à une rubrique ou une file d'attente :

- Ajoutez une politique à un utilisateur ou un groupe IAM. La façon la plus simple d'accorder à des utilisateurs des autorisations d'accès à des rubriques ou des files d'attente consiste à créer un groupe et à lui ajouter la politique appropriée, puis à ajouter des utilisateurs à ce groupe. Il est beaucoup plus facile d'ajouter ou de supprimer des utilisateurs dans un groupe que de suivre les politiques que vous définissez pour des utilisateurs individuels.
- Ajoutez une politique à une rubrique ou une file d'attente. Si vous souhaitez accorder des autorisations d'accès à une rubrique ou une file d'attente à un autre compte AWS, la seule façon de procéder consiste à ajouter une politique dont le principal est le compte Compte AWS auquel vous voulez octroyer des autorisations.

Vous devez utiliser la première méthode dans la plupart des cas (appliquer des politiques à des groupes et gérer les autorisations accordées aux utilisateurs en ajoutant ou supprimant les utilisateurs appropriés dans les groupes). Si vous avez besoin d'accorder des autorisations à un utilisateur d'un autre compte, vous devez utiliser la seconde méthode.

Ajout d'une politique à un utilisateur ou un groupe IAM

Si vous avez ajouté la politique suivante à un utilisateur ou un groupe IAM, vous devez accorder à cet utilisateur ou aux membres de ce groupe l'autorisation d'effectuer l'action `sns:Publish` sur la rubrique MaRubrique.

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

Si vous avez ajouté la politique suivante à un utilisateur ou un groupe IAM, vous devez accorder à cet utilisateur ou aux membres de ce groupe l'autorisation d'effectuer les actions `sqs:ReceiveMessage` et `sqs:DeleteMessage` sur les files d'attente MyQueue1 et MyQueue2.

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

Ajout d'une politique à une rubrique ou une file d'attente

Les exemples de politiques suivants montrent comment accorder à un autre compte des autorisations d'accès à une rubrique et une file d'attente.

> **Note**
>
> Lorsque vous accordez à un autre compte Compte AWS l'accès à une ressource de votre compte, vous permettez également aux utilisateurs IAM disposant d'autorisations d'accès de niveau administrateur (accès générique) d'accéder à cette ressource. Tous les autres utilisateurs IAM de l'autre compte se voient automatiquement refuser l'accès à votre ressource. Si vous voulez accorder à des utilisateurs IAM spécifiques de ce compte Compte AWS l'accès à votre ressource, le compte ou un utilisateur IAM disposant d'un accès de niveau administrateur doit déléguer des autorisations pour la ressource à ces utilisateurs IAM. Pour plus d'informations sur la délégation entre comptes, consultez la section Activation d'accès entre comptes du guide Utilisation d'IAM.

Si vous avez ajouté la politique suivante à une rubrique MyTopic dans le compte 123456789012, vous devez accorder au compte 111122223333 l'autorisation d'effectuer l'action `sns:Publish` sur cette rubrique.

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

Si vous avez ajouté la politique suivante à une file d'attente MyQueue dans le compte 123456789012, vous devez accorder au compte 111122223333 l'autorisation d'effectuer les actions `sqs:ReceiveMessage` et `sqs:DeleteMessage` sur cette file d'attente.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Étape 5 : Tester les abonnements de la file d'attente à une rubrique

Vous pouvez tester les abonnements d'une file d'attente à une rubrique en effectuant une publication dans la rubrique et en affichant le message que la rubrique envoie à la file d'attente.

Pour effectuer une publication dans une rubrique à l'aide de la console Amazon SNS

1. En utilisant les informations d'identification du compte Compte AWS ou de l'utilisateur IAM disposant d'une autorisation de publication dans la rubrique, connectez-vous à AWS Management Console et ouvrez la console Amazon SNS à l'adresse https://console.aws.amazon.com/sns/.
2. Dans le panneau de navigation, sélectionnez la rubrique et choisissez Publier dans la rubrique.
3. Dans la zone Sujet, entrez un objet (par exemple, **`Testing publish to queue`**). Dans la zone Message, saisissez du texte (par exemple, **`Hello world!`**) et choisissez Publier un message. Le message suivant s'affiche : Votre message a été publié.

Pour afficher le message à partir de la rubrique à l'aide de la console Amazon SQS

1. Servez-vous des informations d'identification du compte Compte AWS ou de l'utilisateur IAM disposant des autorisations permettant d'afficher des messages dans la file d'attente, pour vous connecter à la AWS Management Console et ouvrez la console Amazon SQS à l'adresse https://console.aws.amazon.com/sqs/.
2. Choisissez une file d'attente abonnée à la rubrique.
3. Choisissez Send and receive messages (Envoyer et recevoir des messages), puis Poll for messages (Rechercher des messages). Un message de type Notification s'affiche.
4. Dans la colonne Corps, choisissez Plus de détails. La zone Détails des messages contient un document JSON qui inclut l'objet et le message que vous avez publiés dans la rubrique. Le message se présente comme le document JSON suivant.

```
{
  "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. Choisissez Close (Fermer). Vous venez d'effectuer une publication dans une rubrique qui envoie des messages de notification à une file d'attente.

Utilisation d'un modèle AWS CloudFormation pour créer une rubrique qui envoie des messages à des files d'attente Amazon SQS

AWS CloudFormation vous permet d'utiliser un fichier de modèle pour créer et configurer simultanément un ensemble de ressources AWS en tant qu'unité unique. Cette section comporte un exemple de modèle qui simplifie le déploiement de rubriques qui effectuent des publications dans des files d'attente. Les modèles exécutent automatiquement les étapes de configuration en créant deux files d'attente et une rubrique avec des abonnements aux files d'attente, en ajoutant une politique aux files d'attente afin que la rubrique puisse leur envoyer des messages, et en créant des utilisateurs et des groupes IAM pour contrôler l'accès à ces ressources.

Pour plus d'informations sur le déploiement de ressources AWS à l'aide d'un modèle AWS CloudFormation, consultez la page Mise en route du Guide de l'utilisateur AWS CloudFormation.

Utilisation d'un modèle AWS CloudFormation pour configurer des rubriques et des files d'attente dans un compte Compte AWS

L'exemple de modèle crée une rubrique Amazon SNS qui peut envoyer des messages à deux files d'attente Amazon SQS avec les autorisations appropriées pour permettre aux membres d'un groupe IAM d'effectuer une publication dans la rubrique et à un autre de lire les messages en attente. Le modèle crée également des utilisateurs IAM qui sont ajoutés à chaque groupe.

Vous copiez le contenu du modèle dans un fichier. Vous pouvez également télécharger le modèle à partir de la page des modèles AWS CloudFormation. Sur la page des modèles, choisissez Parcourir les exemples de modèle par service AWS, puis choisissez Amazon Simple Queue Service.

MySNSTopic est configuré pour effectuer une publication dans deux points de terminaison abonnés, qui sont deux files d'attente Amazon SQS (MyQueue1 et MyQueue2). MyPublishTopicGroup est un groupe IAM dont les membres sont autorisés à effectuer une publication dans MySNSTopic

à l'aide de l'action d'API Publish ou de la commande sns-publish. Le modèle crée les utilisateurs IAM MyPublishUser et MyQueueUser, et leur affecte des profils de connexion et des clés d'accès. L'utilisateur qui crée une pile avec ce modèle spécifie les mots de passe des profils de connexion en tant que paramètres d'entrée. Le modèle crée des clés d'accès pour les deux utilisateurs IAM avec MyPublishUserKey et MyQueueUserKey. AddUserToMyPublishTopicGroup ajoute MyPublishUser à MyPublishTopicGroup afin que l'utilisateur dispose des autorisations affectées au groupe.

MyRDMessageQueueGroup est un groupe IAM dont les membres sont autorisés à lire et supprimer des messages dans les deux files d'attente Amazon SQS à l'aide des actions d'API ReceiveMessage et DeleteMessage. AddUserToMyQueueGroup ajoute MyQueueUser à MyRDMessageQueueGroup afin que l'utilisateur dispose des autorisations affectées au groupe. MyQueuePolicy autorise MySNSTopic à publier ses notifications dans les deux files d'attente.

La liste suivante présente le contenu du modèle AWS CloudFormation.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template SNSToSQS: This Template creates
 an SNS topic that can send messages to
  two SQS queues with appropriate permissions for one IAM user to publish to the topic
 and another to read messages from the queues.
  MySNSTopic is set up to publish to two subscribed endpoints, which are two SQS queues
 (MyQueue1 and MyQueue2). MyPublishUser is an IAM user
  that can publish to MySNSTopic using the Publish API. MyTopicPolicy assigns that
 permission to MyPublishUser. MyQueueUser is an IAM user
  that can read messages from the two SQS queues. MyQueuePolicy assigns those
 permissions to MyQueueUser. It also assigns permission for
  MySNSTopic to publish its notifications to the two queues. The template creates
 access keys for the two IAM users with MyPublishUserKey
  and MyQueueUserKey. ***Warning*** you will be billed for the AWS resources used if
 you create a stack from this template.",

  "Parameters": {
    "MyPublishUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description": "Password for the IAM user MyPublishUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern": "[a-zA-Z0-9]*",
      "ConstraintDescription": "must contain only alphanumeric characters."
```

```
    },
    "MyQueueUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description": "Password for the IAM user MyQueueUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern": "[a-zA-Z0-9]*",
      "ConstraintDescription": "must contain only alphanumeric characters."
    }
  },


  "Resources": {
    "MySNSTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "Subscription": [{
            "Endpoint": {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "Protocol": "sqs"
          },
          {
            "Endpoint": {
              "Fn::GetAtt": ["MyQueue2", "Arn"]
            },
            "Protocol": "sqs"
          }
        ]
      }
    },
    "MyQueue1": {
      "Type": "AWS::SQS::Queue"
    },
    "MyQueue2": {
      "Type": "AWS::SQS::Queue"
    },
    "MyPublishUser": {
      "Type": "AWS::IAM::User",
      "Properties": {
        "LoginProfile": {
          "Password": {
            "Ref": "MyPublishUserPassword"
```

```
        }
      }
    }
  },
  "MyPublishUserKey": {
    "Type": "AWS::IAM::AccessKey",
    "Properties": {
      "UserName": {
        "Ref": "MyPublishUser"
      }
    }
  },
  "MyPublishTopicGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
        "PolicyName": "MyTopicGroupPolicy",
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Action": [
              "sns:Publish"
            ],
            "Resource": {
              "Ref": "MySNSTopic"
            }
          }]
        }
      }]
    }
  },
  "AddUserToMyPublishTopicGroup": {
    "Type": "AWS::IAM::UserToGroupAddition",
    "Properties": {
      "GroupName": {
        "Ref": "MyPublishTopicGroup"
      },
      "Users": [{
        "Ref": "MyPublishUser"
      }]
    }
  },
  "MyQueueUser": {
    "Type": "AWS::IAM::User",
```

```
      "Properties": {
        "LoginProfile": {
          "Password": {
            "Ref": "MyQueueUserPassword"
          }
        }
      }
    },
    "MyQueueUserKey": {
      "Type": "AWS::IAM::AccessKey",
      "Properties": {
        "UserName": {
          "Ref": "MyQueueUser"
        }
      }
    },
    "MyRDMessageQueueGroup": {
      "Type": "AWS::IAM::Group",
      "Properties": {
        "Policies": [{
          "PolicyName": "MyQueueGroupPolicy",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
              ],
              "Resource": [{
                  "Fn::GetAtt": ["MyQueue1", "Arn"]
                },
                {
                  "Fn::GetAtt": ["MyQueue2", "Arn"]
                }
              ]
            }]
          }
        }]
      }
    },
    "AddUserToMyQueueGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
```

```
          "Ref": "MyRDMessageQueueGroup"
        },
        "Users": [{
          "Ref": "MyQueueUser"
        }]
      }
    },
    "MyQueuePolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": {
              "Service": "sns.amazonaws.com"
            },
            "Action": ["sqs:SendMessage"],
            "Resource": "*",
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": {
                  "Ref": "MySNSTopic"
                }
              }
            }
          }]
        },
        "Queues": [{
          "Ref": "MyQueue1"
        }, {
          "Ref": "MyQueue2"
        }]
      }
    }
  },
  "Outputs": {
    "MySNSTopicTopicARN": {
      "Value": {
        "Ref": "MySNSTopic"
      }
    },
    "MyQueue1Info": {
      "Value": {
        "Fn::Join": [
```

```
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "URL:",
            {
              "Ref": "MyQueue1"
            }
          ]
        ]
      }
    },
    "MyQueue2Info": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue2", "Arn"]
            },
            "URL:",
            {
              "Ref": "MyQueue2"
            }
          ]
        ]
      }
    },
    "MyPublishUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyPublishUser", "Arn"]
            },
            "Access Key:",
            {
              "Ref": "MyPublishUserKey"
            },
```

```
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyPublishUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    },
    "MyQueueUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueueUser", "Arn"]
            },
            "Access Key:",
            {
              "Ref": "MyQueueUserKey"
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyQueueUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    }
  }
}
```

Diffusion en éventail vers les points de terminaison HTTP(S)

Vous pouvez utiliser Amazon SNS pour envoyer des messages de notification à un ou plusieurs points de terminaison HTTP ou HTTPS. Lorsque vous abonnez un point de terminaison à une rubrique, vous pouvez publier une notification dans la rubrique. Amazon SNS envoie alors une demande HTTP POST diffusant le contenu de la notification au point de terminaison abonné. Lors de l'abonnement du point de terminaison, vous choisissez si Amazon SNS doit utiliser HTTP ou HTTPS pour envoyer la demande POST au point de terminaison. Si vous utilisez HTTPS, vous pouvez bénéficier de la prise en charge des éléments suivants dans Amazon SNS :

- Server Name Indication (SNI) : Cette option permet à Amazon SNS de prendre en charge des points de terminaison HTTPS qui requièrent un SNI, tel qu'un serveur nécessitant plusieurs certificats pour héberger plusieurs domaines. Pour plus d'informations sur SNI, consultez Server Name Indication (Indication du nom du serveur).

- Basic and Digest Access Authentication : Cette option vous permet de spécifier un nom d'utilisateur et un mot de passe dans l'URL HTTPS pour la demande HTTP POST, par exemple `https://user:password@domain.com` ou `https://user@domain.com`. Le nom d'utilisateur et le mot de passe sont chiffrés via la connexion SSL établie lors de l'utilisation de HTTPS. Seul le nom de domaine est envoyé en texte brut. Pour plus d'informations sur l'authentification de base et de la valeur de hachage, consultez RFC-2617.

  > **Important**
  >
  > Amazon SNS ne prend actuellement pas en charge les points de terminaison HTTP(S) privés.
  > Les URL HTTPS ne peuvent être récupérées qu'à partir de l'action d'API Amazon SNS `GetSubscriptionAttributes`, pour les principaux auxquels vous avez accordé l'accès à l'API.

  > **Note**
  >
  > Le service client doit être en mesure de prendre en charge l'en-tête de réponse `HTTP/1.1 401 Unauthorized`

La demande contient l'objet et le message ayant été publiés dans la rubrique, ainsi que les métadonnées relatives à la notification dans un document JSON. La demande se présente comme la requête HTTP POST suivante. Pour plus d'informations sur l'en-tête HTTP et le format JSON du corps de la demande, consultez les pages En-têtes HTTP/HTTPS et Format JSON de notification HTTP/HTTPS.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
```

```
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
    Content-Length: 761
    Content-Type: text/plain; charset=UTF-8
    Host: ec2-50-17-44-49.compute-1.amazonaws.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" :
 "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

Rubriques

- Abonnement d'un point de terminaison HTTP/S à une rubrique
- Vérification des signatures des messages Amazon SNS
- Analyse des formats des messages

Abonnement d'un point de terminaison HTTP/S à une rubrique

Les pages de cette section décrivent comment abonner des points de terminaison HTTP/S aux rubriques Amazon SNS.

Rubriques

- Étape 1 : Vérification de la disponibilité de votre point de terminaison pour traiter des messages Amazon SNS
- Étape 2 : Abonner le point de terminaison HTTP/HTTPS à la rubrique Amazon SNS

- Étape 3 : Confirmer l'abonnement
- Étape 4 : Définir la politique de remise pour l'abonnement (facultatif)
- Étape 5 : Accorder aux utilisateurs les autorisations de publication dans la rubrique (facultatif)
- Étape 6 : Envoyer des messages au point de terminaison HTTP/HTTPS

Étape 1 : Vérification de la disponibilité de votre point de terminaison pour traiter des messages Amazon SNS

Avant d'abonner votre point de terminaison HTTP ou HTTPS à une rubrique, vous devez vous assurer qu'il est capable de traiter les demandes HTTP POST utilisées par Amazon SNS pour envoyer les messages de notification et de confirmation d'abonnement. En général, il s'agit de créer et déployer une application web (par exemple, un servlet Java si votre hôte de point de terminaison exécute Linux avec Apache et Tomcat) qui traite les demandes HTTP provenant d'Amazon SNS. Lorsque vous abonnez un point de terminaison HTTP, Amazon SNS envoie une demande de confirmation d'abonnement. Votre point de terminaison doit être préparé pour recevoir et traiter cette demande lorsque vous créez l'abonnement, car Amazon SNS envoie cette demande à ce moment-là. Amazon SNS n'envoie pas de notifications au point de terminaison tant que l'abonnement n'a pas été confirmé. Une fois que vous avez confirmé l'abonnement, Amazon SNS envoie des notifications au point de terminaison lorsqu'une action de publication est effectuée sur la rubrique abonnée.

Pour configurer votre point de terminaison pour le traitement des messages de notification et de confirmation d'abonnement

1. Votre code doit lire les en-têtes HTTP des demandes HTTP POST qu'Amazon SNS envoie à votre point de terminaison. Votre code doit rechercher le champ d'en-tête `x-amz-sns-message-type`, qui vous indique le type de message qu'Amazon SNS vous a envoyé. En examinant l'en-tête, vous pouvez déterminer le type de message sans avoir à analyser le corps de la demande HTTP. Vous devez traiter deux types de messages : `SubscriptionConfirmation` et `Notification`. Le message `UnsubscribeConfirmation` est utilisé uniquement lorsque l'abonnement est supprimé de la rubrique.

   Pour plus d'informations sur l'en-tête HTTP, consultez la page En-têtes HTTP/HTTPS. La demande HTTP POST suivante constitue un exemple de message de confirmation d'abonnement.

   ```
   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. Votre code doit analyser le document JSON dans le corps de la demande HTTP POST et content-type text/plain pour lire les paires nom-valeur qui constituent le message Amazon SNS. Utilisez un analyseur JSON capable de convertir la représentation avec séquence d'échappement des caractères de contrôle en valeurs de caractères ASCII (par exemple, conversion de \n en caractère de nouvelle ligne). Vous pouvez utiliser un analyseur JSON existant tel que Jackson JSON Processor ou écrire le vôtre. Pour envoyer le texte figurant dans les champs d'objet et de message dans un format JSON valide, Amazon SNS doit convertir certains caractères de contrôle en représentations avec séquence d'échappement qui peuvent être incluses dans le document JSON. Lorsque vous recevez le document JSON dans le corps de la demande POST envoyée à votre point de terminaison, vous devez restaurer les valeurs d'origine des caractères d'échappement si vous souhaitez une représentation exacte de l'objet d'origine et des messages publiés dans la rubrique. Cette opération est essentielle si vous souhaitez vérifier la signature d'une notification, car la signature utilise le message et l'objet dans leurs formats d'origine dans le cadre de la chaîne de connexion.

3. Votre code doit vérifier l'authenticité d'un message de notification, de confirmation d'abonnement ou de confirmation de désabonnement envoyé par Amazon SNS. À l'aide des informations contenues dans le message d'Amazon SNS, votre point de terminaison peut recréer la signature afin que vous puissiez vérifier le contenu du message en mettant en correspondance votre signature avec celle envoyée par Amazon SNS avec le message. Pour plus d'informations sur la vérification de la signature d'un message, consultez la page Vérification des signatures des messages Amazon SNS.
4. En fonction du type spécifié par le champ d'en-tête `x-amz-sns-message-type`, votre code doit lire le document JSON contenu dans le corps de la demande HTTP et traiter le message. Vous trouverez ci-dessous les instructions permettant de traiter les deux principaux types de messages.

   SubscriptionConfirmation

   Lisez la valeur du paramètre `SubscribeURL` et accédez à cette URL. Pour confirmer l'abonnement et commencer à recevoir des notifications sur le point de terminaison, vous devez accéder à l'`SubscribeURL`URL (par exemple, en envoyant une demande HTTP GET à l'URL). Consultez l'exemple de demande HTTP à l'étape précédente pour voir comment se présente `SubscribeURL`. Pour plus d'informations sur le format du message `SubscriptionConfirmation`, consultez la page Format JSON de confirmation d'abonnement HTTP/HTTPS. Lorsque vous accédez à l'URL, vous obtenez une réponse semblable au document XML suivant. Le document renvoie l'ARN de l'abonnement pour le point de terminaison dans l'élément `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>
   ```

   Au lieu d'accéder à `SubscribeURL`, vous pouvez confirmer l'abonnement à l'aide de l'action ConfirmSubscription avec le paramètre `Token` défini sur la valeur correspondante dans le message `SubscriptionConfirmation`. Si vous souhaitez autoriser uniquement le

propriétaire de la rubrique et le propriétaire de l'abonnement à désabonner le point de terminaison, appelez l'action `ConfirmSubscription` avec une signature AWS.

Notification

Lisez les valeurs de `Subject` et `Message` pour obtenir les informations de notification publiées dans la rubrique.

Pour plus d'informations sur le format du message `Notification`, consultez la page En-têtes HTTP/HTTPS. La demande HTTP POST suivante constitue un exemple de message de notification envoyé au point de terminaison 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. Assurez-vous que votre point de terminaison répond au message HTTP POST provenant d'Amazon SNS avec le code d'état approprié. La connexion expirera au bout de 15 secondes. Si votre point de terminaison ne répond pas dans le délai imparti ou renvoie un code d'état

en dehors de la plage 200–4xx, Amazon SNS considère la diffusion du message comme une tentative infructueuse.

6. Vérifiez que votre code peut traiter les relances de diffusion de messages à partir d'Amazon SNS. Si Amazon SNS ne reçoit pas de réponse positive de votre point de terminaison, il tente de diffuser à nouveau le message. Cela s'applique à tous les messages, notamment le message de confirmation d'abonnement. Par défaut, si la diffusion initiale du message échoue, Amazon SNS tente jusqu'à trois relances avec un délai de 20 secondes entre les échecs des tentatives.

   > **Note**
   >
   > La demande de message expire après 15 secondes. Cela signifie que si l'échec de la diffusion du message est provoqué par un dépassement du délai d'attente, Amazon SNS effectue une relance environ 35 secondes après la tentative de diffusion précédente. Vous pouvez définir une politique de livraison différente pour le point de terminaison.

   Amazon SNS utilise le champ d'en-tête `x-amz-sns-message-id` pour identifier de manière unique chaque message publié dans une rubrique Amazon SNS. En comparant les ID des messages que vous avez traités avec les messages entrants, vous pouvez déterminer si le message est une tentative de relance.

7. Si vous abonnez un point de terminaison HTTPS, assurez-vous que ce dernier possède un certificat de serveur provenant d'une autorité de certification (CA) approuvée. Amazon SNS envoie uniquement des messages aux points de terminaison HTTPS qui disposent d'un certificat de serveur signé par une autorité de certification qu'il a approuvée.
8. Déployez le code que vous avez créé pour recevoir des messages Amazon SNS. Lorsque vous abonnez le point de terminaison, il doit être prêt à recevoir au moins le message de confirmation d'abonnement.

Étape 2 : Abonner le point de terminaison HTTP/HTTPS à la rubrique Amazon SNS

Pour envoyer des messages à un point de terminaison HTTP ou HTTPS via une rubrique, vous devez abonner le point de terminaison à la rubrique Amazon SNS. Spécifiez le point de terminaison à l'aide de son URL. Pour vous abonner à une rubrique, vous pouvez utiliser la console Amazon SNS, la commande sns-subscribe ou l'action d'API Abonner. Avant de commencer, assurez-vous que vous disposez de l'URL du point de terminaison à abonner et que ce dernier est prêt à recevoir les messages de confirmation et de notification, comme décrit à l'étape 1.

Pour abonner un point de terminaison HTTP ou HTTPS à une rubrique à l'aide de la console Amazon SNS

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Choisissez Créer un abonnement.
4. Dans la liste déroulante Protocole, sélectionnez HTTP ou HTTPS.
5. Dans la zone Point de terminaison, collez l'URL du point de terminaison auquel la rubrique doit envoyer des messages, puis choisissez Créer un abonnement.
6. Le message de confirmation s'affiche. Choisissez Close (Fermer).

   Le champ ID de l'abonnement de votre nouvel abonnement affiche PendingConfirmation. Lorsque vous confirmez l'abonnement, le champ ID de l'abonnement affiche l'ID d'abonnement.

Étape 3 : Confirmer l'abonnement

Une fois que vous êtes abonné à votre point de terminaison, Amazon SNS lui envoie un message de confirmation d'abonnement. Le code qui exécute les actions décrites dans l'étape 1 doit déjà être déployé sur votre point de terminaison. En particulier, le code sur le point de terminaison doit récupérer la valeur `SubscribeURL` dans le message de confirmation d'abonnement, puis accéder à l'emplacement spécifié par l'URL `SubscribeURL` elle-même, ou le mettre à votre disposition afin que vous puissiez accéder manuellement à l'URL `SubscribeURL`, par exemple à l'aide d'un navigateur web. Amazon SNS n'envoie pas de messages au point de terminaison tant que l'abonnement n'a pas été confirmé. Lorsque vous accédez à `SubscribeURL`, la réponse contient un document XML comprenant un élément `SubscriptionArn` qui spécifie l'ARN de l'abonnement. Vous pouvez également utiliser la console Amazon SNS pour vérifier que l'abonnement est confirmé : le champ ID de l'abonnement affiche l'ARN de l'abonnement au lieu de la valeur `PendingConfirmation` que vous avez vue lors de l'ajout initial de l'abonnement.

Étape 4 : Définir la politique de remise pour l'abonnement (facultatif)

Par défaut, si la diffusion initiale du message échoue, Amazon SNS tente jusqu'à trois relances avec un délai de 20 secondes entre les échecs des tentatives. Comme indiqué dans l'étape 1, votre point de terminaison doit contenir du code capable de traiter les messages relancés. En définissant la politique de diffusion sur une rubrique ou un abonnement, vous pouvez contrôler la fréquence et l'intervalle auxquels Amazon SNS relancera les messages qui ont échoué. Vous pouvez également

spécifier le type de contenu pour vos notifications HTTP/S dans `DeliveryPolicy`. Pour de plus amples informations, veuillez consulter Création d'une politique de distribution HTTP/S.

Étape 5 : Accorder aux utilisateurs les autorisations de publication dans la rubrique (facultatif)

Par défaut, le propriétaire de la rubrique dispose d''autorisations de publication dans la rubrique. Pour permettre à d'autres utilisateurs ou applications d'effectuer une publication dans la rubrique, vous devez utiliser AWS Identity and Access Management (IAM) pour accorder des autorisations de publication dans la rubrique. Pour plus d'informations sur l'octroi d'autorisations d'actions Amazon SNS aux utilisateurs IAM, consultez la section Utilisation de politiques basées sur l'identité avec Amazon SNS.

Il existe deux façons de contrôler l'accès à une rubrique :

- Ajoutez une politique à un utilisateur ou un groupe IAM. La façon la plus simple d'accorder à des utilisateurs des autorisations d'accès à des rubriques consiste à créer un groupe et à lui ajouter la politique appropriée, puis à ajouter des utilisateurs à ce groupe. Il est beaucoup plus facile d'ajouter ou de supprimer des utilisateurs dans un groupe que de suivre les politiques que vous définissez pour des utilisateurs individuels.
- Ajoutez une politique à la rubrique. Si vous souhaitez accorder des autorisations d'accès à une rubrique à un autre compte AWS, la seule façon de procéder consiste à ajouter une politique dont le principal est Compte AWS auquel vous voulez accorder des autorisations.

Vous devez utiliser la première méthode dans la plupart des cas (appliquer des politiques à des groupes et gérer les autorisations accordées aux utilisateurs en ajoutant ou supprimant les utilisateurs appropriés dans les groupes). Si vous avez besoin d'accorder des autorisations à un utilisateur d'un autre compte, utilisez la seconde méthode.

Si vous avez ajouté la politique suivante à un utilisateur ou un groupe IAM, vous devez accorder à cet utilisateur ou aux membres de ce groupe l'autorisation d'effectuer l'action `sns:Publish` sur la rubrique MaRubrique.

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

```
  }]
}
```

L'exemple de politique suivant montre comment accorder à un autre compte des autorisations d'accès à une rubrique.

> **Note**
>
> Lorsque vous accordez à un autre Compte AWS l'accès à une ressource de votre compte, vous permettez également aux utilisateurs IAM disposant d'autorisations d'accès de niveau administrateur (accès générique) d'accéder à cette ressource. Tous les autres utilisateurs IAM de l'autre compte se voient automatiquement refuser l'accès à votre ressource. Si vous voulez accorder à des utilisateurs IAM spécifiques de ce compte Compte AWS l'accès à votre ressource, le compte ou un utilisateur IAM disposant d'un accès de niveau administrateur doit déléguer des autorisations pour la ressource à ces utilisateurs IAM. Pour plus d'informations sur la délégation entre comptes, consultez la section Activation d'accès entre comptes du guide Utilisation d'IAM.

Si vous avez ajouté la politique suivante à une rubrique MaRubrique dans le compte 123456789012, vous devez accorder au compte 111122223333 l'autorisation d'effectuer l'action `sns:Publish` sur cette rubrique.

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

Étape 6 : Envoyer des messages au point de terminaison HTTP/HTTPS

Vous pouvez envoyer un message aux abonnements d'une rubrique en effectuant une publication dans la rubrique. Pour publier une rubrique, vous pouvez utiliser la console Amazon SNS, la CLI `sns-publish` ou l'API `Publish`.

Si vous avez suivi l'étape 1, le code que vous avez déployé sur votre point de terminaison doit traiter la notification.

Pour effectuer une publication dans une rubrique à l'aide de la console Amazon SNS

1. En utilisant les informations d'identification de Compte AWS ou de l'utilisateur IAM disposant d'une autorisation de publication dans la rubrique, connectez-vous à AWS Management Console et ouvrez la console Amazon SNS à l'adresse https://console.aws.amazon.com/sns/.
2. Dans le panneau de navigation, choisissez Rubriques, puis sélectionnez une rubrique.
3. Cliquez sur le bouton Publier le message.
4. Dans la zone Sujet, entrez un objet (par exemple, **`Testing publish to my endpoint`**).
5. Dans la zone Message, entrez un texte (par exemple, **`Hello world!`**), puis choisissez Publier un message.

   Le message suivant s'affiche : Votre message a été publié.

Vérification des signatures des messages Amazon SNS

Pour contrôler l'authenticité d'un message envoyé à votre point de terminaison HTTP par Amazon SNS, vous pouvez vérifier la signature du message. Dans deux cas, nous recommandons de vérifier l'authenticité du message. Le premier cas est celui où Amazon SNS envoie un message à votre point de terminaison HTTP indiquant que vous êtes abonné à une rubrique. Le second, lorsqu'Amazon SNS vous envoie un message de confirmation à votre point de terminaison HTTP lors de l'exécution des actions d'API `Subscribe` ou `Unsubscribe`.

Vous devez exécuter les opérations suivantes lors de la vérification des messages envoyés par Amazon SNS :

- Utilisez toujours le protocole HTTPS lors de l'obtention du certificat à partir d'Amazon SNS.
- Validez l'authenticité du certificat.
- Vérifiez que le certificat a été reçu à partir d'Amazon SNS.
- Si possible, utilisez l'un des kits SDK AWS pris en charge pour qu'Amazon SNS valide et vérifie les messages.
- Vérifiez que les messages Amazon SNS ont bien été reçus par le `TopicArn` de votre choix.

Amazon SNS prend en charge deux versions de signature de message :

- `SignatureVersion1` : Amazon SNS crée la signature sur la base du hachage SHA1 du message.
- `SignatureVersion2` : Amazon SNS crée la signature sur la base du hachage SHA256 du message.

Pour configurer la version de signature des messages sur les rubriques Amazon SNS

Par défaut, les rubriques Amazon SNS utilisent `SignatureVersion` 1. Pour choisir l'algorithme de hachage sur votre rubrique Amazon SNS, `SignatureVersion` 1 (SHA1) ou `SignatureVersion` 2 (SHA256), vous pouvez utiliser l'action d'API `SetTopicAttributes`.

L'exemple de code suivant montre comment définir l'attribut de rubrique `SignatureVersion` avec le AWS CLI :

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

Pour vérifier la signature d'un message Amazon SNS lors de l'utilisation de demandes basées sur une requête HTTP

1. Extrayez les paires nom-valeur à partir du document JSON dans le corps de la requête HTTP POST envoyée par Amazon SNS à votre point de terminaison. Vous utiliserez les valeurs de certaines paires nom-valeur pour créer la chaîne de connexion. Lorsque vous vérifiez la signature d'un message Amazon SNS, il est essentiel de convertir les caractères de contrôle d'échappement en représentations des caractères d'origine dans `Message` et les valeurs `Subject`. Ces valeurs doivent être dans leur format d'origine lorsque vous les utilisez dans le cadre de la chaîne de connexion. Pour plus d'informations sur l'analyse du document JSON, consultez la page Étape 1 : Vérification de la disponibilité de votre point de terminaison pour traiter des messages Amazon SNS.

   La `SignatureVersion` vous indique la version de signature utilisée par Amazon SNS pour générer la signature du message. A partir de la version de la signature, vous pouvez déterminer les exigences requises pour la génération de la signature. Pour les notifications, Amazon SNS prend actuellement en charge la signature versions 1 et 2. Cette section indique les étapes de vérification d'une signature à l'aide de ces versions de signature.

2. Obtenez le certificat X509 utilisé par Amazon SNS pour signer le message. La valeur `SigningCertURL` pointe vers l'emplacement du certificat X509 utilisé pour créer la signature numérique du message. Récupérez le certificat à partir de cet emplacement.
3. Extrayez la clé publique à partir du certificat. La clé publique provenant du certificat spécifié par `SigningCertURL` est utilisée pour vérifier l'authenticité et l'intégrité du message.
4. Déterminez le type du message. Le format de la chaîne de connexion dépend du type de message, qui est spécifié par la valeur `Type`.
5. Créez la chaîne de connexion. La chaîne de connexion est une liste délimitée par des caractères de nouvelle ligne, qui est composée de paires nom-valeur provenant du message. Chaque paire nom-valeur est représentée par le nom, suivi d'un caractère de nouvelle ligne, suivi de la valeur et se terminant par un caractère de nouvelle ligne. Les paires nom-valeur doivent figurer dans l'ordre de tri par octet.

   En fonction du type de message, la chaîne de connexion doit avoir les paires nom-valeur suivantes.

   Notification

   Les messages de notification doivent contenir les paires nom-valeur suivantes :

   ```
   Message
   MessageId
   Subject (if included in the message)
   Timestamp
   TopicArn
   Type
   ```

   L'exemple suivant est une chaîne de connexion pour `Notification`.

   ```
   Message
   My Test Message
   MessageId
   4d4dc071-ddbf-465d-bba8-08f81c89da64
   Subject
   My subject
   Timestamp
   2019-01-31T04:37:04.321Z
   TopicArn
   arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
   Type
   ```

```
Notification
```

SubscriptionConfirmation et UnsubscribeConfirmation

Les messages `SubscriptionConfirmation` et `UnsubscribeConfirmation` doivent contenir les paires nom-valeur suivantes :

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

L'exemple suivant est une chaîne de connexion pour `SubscriptionConfirmation`.

```
Message
My Test Message
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-2:123456789012:s4-
MySNSTopic-1G1WEFCOXTC0P&Token=233...
Timestamp
2019-01-31T19:25:13.719Z
Token
233...
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

6. Décodez la valeur `Signature` à partir du format Base64. Le message diffuse la signature dans la valeur `Signature`, qui est codée en Base64. Avant de comparer la valeur de la signature avec la signature que vous avez calculée, assurez-vous que vous décodez la valeur `Signature` en Base64 afin d'utiliser le même format pour comparer les valeurs.
7. Générez la valeur de hachage dérivée du message Amazon SNS. Envoyez le message Amazon SNS au format canonique à l'algorithme de hachage qui a été utilisé pour générer la signature.

a. Si la `SignatureVersion` est 1, utilisez SHA1 comme algorithme de hachage.

b. Si la `SignatureVersion` est 2, utilisez SHA256 comme algorithme de hachage.

8. Générez la valeur de hachage déclarée du message Amazon SNS. La valeur de hachage déclarée résulte de l'utilisation de la valeur de clé publique (à l'étape 3) pour déchiffrer la signature diffusée avec le message Amazon SNS.
9. Vérifiez l'authenticité et l'intégrité du message Amazon SNS. Comparez la valeur de hachage dérivée (à l'étape 7) avec la valeur de hachage déclarée (à l'étape 8). Si les valeurs sont identiques, le destinataire est sûr que le message n'a pas été modifié pendant son transfert et qu'il provient d'Amazon SNS. Si les valeurs ne sont pas identiques, le message n'est pas digne de confiance pour le destinataire.

Analyse des formats des messages

Amazon SNS utilise les formats suivants.

Rubriques

En-têtes HTTP/HTTPS

Lorsque Amazon SNS envoie un message de confirmation d'abonnement, de notification ou de confirmation de désabonnement aux points de terminaison HTTP/HTTPS, il envoie un message POST avec un certain nombre de valeurs d'en-tête spécifiques à Amazon SNS. Vous pouvez utiliser les valeurs d'en-tête pour des opérations telles que l'identification du type de message sans avoir à analyser le corps du message JSON pour lire la valeur `Type`. Par défaut, Amazon SNS envoie toutes les notifications aux points de terminaison HTTP/S avec `Content-Type` défini sur `text/plain; charset=UTF-8`. Pour choisir un `Content-Type` autre que text/plain (par défaut), consultez `headerContentType` dans Création d'une politique de distribution HTTP/S.

`x-amz-sns-message-type`

Type du message. Les valeurs possibles sont `SubscriptionConfirmation`, `Notification` et `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Identifiant unique universel (UUID), propre à chaque message publié. Pour une notification qu'Amazon SNS renvoie au cours d'une nouvelle tentative, l'ID du message d'origine est utilisé.

`x-amz-sns-topic-arn`

Amazon Resource Name (ARN) de la rubrique dans laquelle ce message a été publié.

`x-amz-sns-subscription-arn`

ARN de l'abonnement à ce point de terminaison.

L'en-tête HTTP POST suivant est un exemple d'en-tête pour un message `Notification` adressé à un point de terminaison HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

Format JSON de confirmation d'abonnement HTTP/HTTPS

Une fois que vous avez abonné un point de terminaison HTTP/HTTPS, Amazon SNS envoie un message de confirmation d'abonnement au point de terminaison HTTP/HTTPS. Ce message contient une valeur `SubscribeURL` que vous devez visiter pour confirmer l'abonnement (sinon, vous pouvez utiliser la valeur `Token` avec `ConfirmSubscription`).

> **Note**
>
> Amazon SNS n'envoie pas de notifications à ce point de terminaison tant que l'abonnement n'est pas confirmé

Le message de confirmation d'abonnement est un message POST dont le corps contient un document JSON avec les paires nom-valeur suivantes.

`Type`

Type du message. Pour une confirmation d'abonnement, le type est `SubscriptionConfirmation`.

`MessageId`

Identifiant unique universel (UUID), propre à chaque message publié. Pour un message qu'Amazon SNS renvoie au cours d'une nouvelle tentative, l'ID de message d'origine est utilisé.

`Token`

Valeur que vous pouvez utiliser avec l'action `ConfirmSubscription` pour confirmer l'abonnement. Sinon, vous pouvez simplement visiter le paramètre `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) de la rubrique à laquelle ce point de terminaison est abonné.

`Message`

Chaîne qui décrit le message. Pour une confirmation d'abonnement, cette chaîne se présente comme suit :

```
You have chosen to subscribe to the topic arn:aws:sns:us-
east-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.
```

`SubscribeURL`

URL que vous devez visiter afin de confirmer l'abonnement. Sinon, vous pouvez utiliser à la place le `Token` avec l'action `ConfirmSubscription` pour confirmer l'abonnement.

`Timestamp`

Heure (GMT) à laquelle la confirmation d'abonnement a été envoyée.

`SignatureVersion`

Version de la signature Amazon SNS utilisée.

- Si `SignatureVersion` a pour valeur 1, `Signature` est une signature `SHA1withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Type`, `Timestamp` et `TopicArn`.
- Si `SignatureVersion` a pour valeur 2, `Signature` est une signature `SHA256withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Type`, `Timestamp` et `TopicArn`.

`Signature`

Signature `SHA1withRSA` ou `SHA256withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Type`, `Timestamp` et `TopicArn`.

`SigningCertURL`

URL permettant d'accéder au certificat utilisé pour signer le message.

Le message HTTP POST suivant est un exemple de message `SubscriptionConfirmation` à un point de terminaison HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
```

```
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

Format JSON de notification HTTP/HTTPS

Lorsqu'Amazon SNS envoie une notification à un point de terminaison HTTP ou HTTPS abonné, le corps du message POST envoyé au point de terminaison contient un document JSON avec les paires nom-valeur suivantes.

`Type`

Type du message. Pour une notification, le type est `Notification`.

`MessageId`

Identifiant unique universel (UUID), propre à chaque message publié. Pour une notification qu'Amazon SNS renvoie au cours d'une nouvelle tentative, l'ID du message d'origine est utilisé.

`TopicArn`

Amazon Resource Name (ARN) de la rubrique dans laquelle ce message a été publié.

`Subject`

Paramètre `Subject` spécifié quand la notification a été publiée dans la rubrique.

> **Note**
>
> Ce paramètre est facultatif. Si aucun paramètre `Subject` n'a été spécifié, cette paire nom-valeur n'apparaît pas dans ce document JSON.

`Message`

Valeur `Message` spécifiée quand la notification a été publiée dans la rubrique.

`Timestamp`

Heure (GMT) à laquelle la notification a été publiée.

`SignatureVersion`

Version de la signature Amazon SNS utilisée.

- Si `SignatureVersion` a pour valeur 1, `Signature` est une signature `SHA1withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Subject` (le cas échéant), `Type`, `Timestamp` et `TopicArn`.
- Si `SignatureVersion` a pour valeur 2, `Signature` est une signature `SHA256withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Subject` (le cas échéant), `Type`, `Timestamp` et `TopicArn`.

`Signature`

Signature `SHA1withRSA` ou `SHA256withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Subject` (le cas échéant), `Type`, `Timestamp` et `TopicArn`.

`SigningCertURL`

URL permettant d'accéder au certificat utilisé pour signer le message.

`UnsubscribeURL`

URL que vous pouvez utiliser pour désabonner le point de terminaison de cette rubrique. Si vous visitez cette URL, Amazon SNS désabonne le point de terminaison et cesse de lui envoyer des notifications.

Le message HTTP POST suivant est un exemple de message `Notification` à un point de terminaison HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
```

```
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
}
```

Format JSON de confirmation de désabonnement HTTP/HTTPS

Une fois qu'un point de terminaison HTTP/HTTPS est désabonné d'une rubrique, Amazon SNS envoie un message de confirmation de désabonnement au point de terminaison.

Le message de confirmation de désabonnement est un message POST dont le corps contient un document JSON avec les paires nom-valeur suivantes.

`Type`

Type du message. Pour une confirmation de désabonnement, le type est `UnsubscribeConfirmation`.

`MessageId`

Identifiant unique universel (UUID), propre à chaque message publié. Pour un message qu'Amazon SNS renvoie au cours d'une nouvelle tentative, l'ID de message d'origine est utilisé.

`Token`

Valeur que vous pouvez utiliser avec l'action `ConfirmSubscription` pour reconfirmer l'abonnement. Sinon, vous pouvez simplement visiter le paramètre `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) de la rubrique dont ce point de terminaison s'est désabonné.

`Message`

Chaîne qui décrit le message. Pour une confirmation de désabonnement, cette chaîne se présente comme suit :

```
You have chosen to deactivate subscription arn:aws:sns:us-
east-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.
```

`SubscribeURL`

URL que vous devez visiter afin de reconfirmer l'abonnement. Sinon, vous pouvez utiliser à la place le `Token` avec l'action `ConfirmSubscription` pour reconfirmer l'abonnement.

`Timestamp`

Heure (GMT) à laquelle la confirmation de désabonnement a été envoyée.

`SignatureVersion`

Version de la signature Amazon SNS utilisée.

- Si `SignatureVersion` a pour valeur 1, `Signature` est une signature `SHA1withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Type`, `Timestamp` et `TopicArn`.
- Si `SignatureVersion` a pour valeur 2, `Signature` est une signature `SHA256withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Type`, `Timestamp` et `TopicArn`.

`Signature`

Signature `SHA1withRSA` ou `SHA256withRSA` codée en Base64 des valeurs `Message`, `MessageId`, `Type`, `Timestamp` et `TopicArn`.

`SigningCertURL`

URL permettant d'accéder au certificat utilisé pour signer le message.

Le message HTTP POST suivant est un exemple de message `UnsubscribeConfirmation` à un point de terminaison 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"
}
```

Format JSON de politique de diffusion SetSubscriptionAttributes

Si vous envoyez une demande à l'action `SetSubscriptionAttributes` et que vous définissez le paramètre `AttributeName` sur la valeur `DeliveryPolicy`, la valeur du paramètre `AttributeValue` doit être un objet JSON valide. Par exemple, l'exemple suivant définit la politique de diffusion sur 5 tentatives au 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}}
...
```

Utilisez le format JSON suivant pour la valeur du paramètre `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"
    }
}
```

Pour plus d'informations sur l'action `SetSubscriptionAttribute`, accédez à SetSubscriptionAttributes dans la Référence d'API Amazon Simple Notification Service. Pour plus d'informations sur les en-têtes content-type HTTP pris en charge, consultez Création d'une politique de distribution HTTP/S.

Format JSON de politique de diffusion SetTopicAttributes

Si vous envoyez une demande à l'action `SetTopicAttributes` et que vous définissez le paramètre `AttributeName` sur la valeur `DeliveryPolicy`, la valeur du paramètre `AttributeValue` doit être un objet JSON valide. Par exemple, l'exemple suivant définit la politique de diffusion sur 5 tentatives au 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}}}
...
```

Utilisez le format JSON suivant pour la valeur du paramètre `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"
        }
    }
}
```

Pour de plus amples informations sur l'action `SetTopicAttribute`, accédez à SetTopicAttributes dans la Référence d'API Amazon Simple Notification Service. Pour plus d'informations sur les en-têtes content-type HTTP pris en charge, consultez Création d'une politique de distribution HTTP/S.

Diffusion dans les Event Fork Pipelines AWS

Pour l'archivage et l'analyse des événements, Amazon SNS recommande désormais d'utiliser son intégration native avec Amazon Data Firehose. Vous pouvez abonner les flux de diffusion Firehose aux rubriques SNS, ce qui vous permet d'envoyer des notifications aux points de terminaison d'archivage et d'analyse tels que les buckets Amazon Simple Storage Service (Amazon S3), les tables Amazon Redshift, Amazon Service (Service), etc. OpenSearch OpenSearch L'utilisation d'Amazon SNS avec les flux de diffusion Firehose est une solution entièrement gérée et sans code qui ne nécessite pas l'utilisation de fonctions. AWS Lambda Pour plus d’informations, consultez Streams de diffusion de Fanout to Firehose.

Vous pouvez utiliser Amazon SNS pour créer des applications qui reposent sur les événements et utilisent les services d'abonné afin d'exécuter automatiquement des tâches en réponse à des événements déclenchés par ses services d'éditeur. Ce modèle d'architecture peut rendre des services plus réutilisables, interopérables et évolutifs. Toutefois, il peut être fastidieux de traiter les événements via des pipelines qui répondent aux exigences courantes de gestion des événements, par exemple la sauvegarde, le stockage, la recherche, l'analytique et la relecture d'événements.

Pour accélérer le développement de vos applications basées sur les événements, vous pouvez abonner aux rubriques Amazon SNS des pipelines de gestion des événements, à technologie

Event Fork Pipelines AWS. AWS Event Fork Pipelines est une suite d'applications imbriquées open source, basée sur le modèle d'application sans serveur AWS (AWS SAM), que vous pouvez déployer directement à partir de la suite AWS Event Fork Pipelines (choisissez Show apps that create custom IAM roles or resource policies (Afficher les appli qui créent des rôles IAM personnalisés ou des politiques de ressource) dans votre compte AWS.

Pour un cas d'utilisation d'Event Fork Pipelines AWS, consultez la section Déploiement et test de l'AWSExemple d'application d'Event Fork Pipelines.

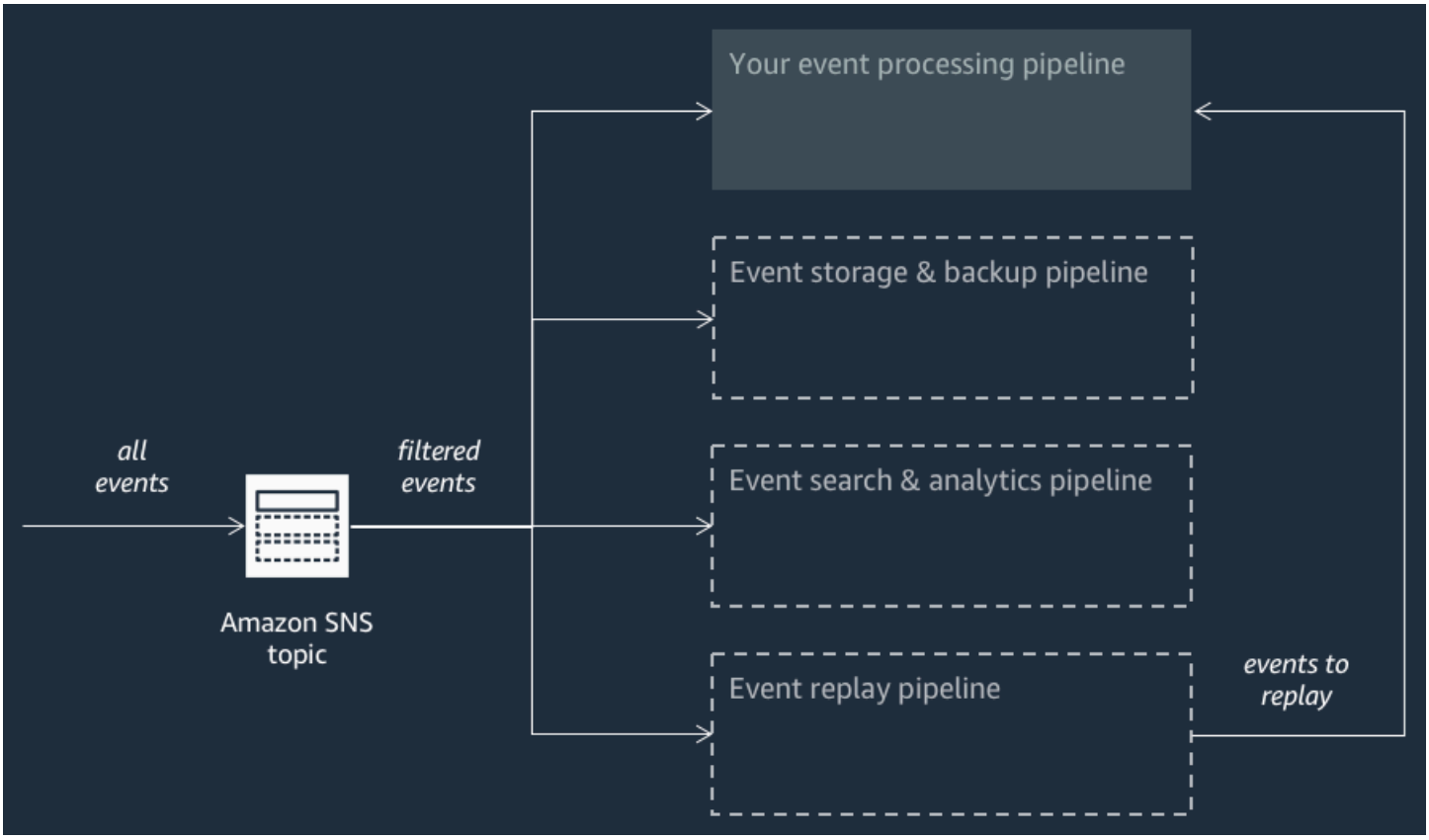
Rubriques

Fonctionnement des Event Fork PipelinesAWS

AWS Event Fork Pipelines est un modèle de conception sans serveur. Toutefois, il s'agit également d'une suite d'applications sans serveur imbriquées en fonction du AWS SAM (que vous pouvez déployer directement à partir de AWS Serverless Application Repository (AWS SAR) sur votre Compte AWS afin d'enrichir vos plateformes basées sur les événements). Vous pouvez déployer ces applications imbriquées individuellement, comme requis par votre architecture.

Rubriques

Le diagramme suivant présente une application d'Event Fork Pipelines AWS complétée par trois applications imbriquées. Vous pouvez déployer l'un des pipelines de la suite d'Event Fork Pipelines AWS sur AWS SAR indépendamment, comme le requiert votre architecture.



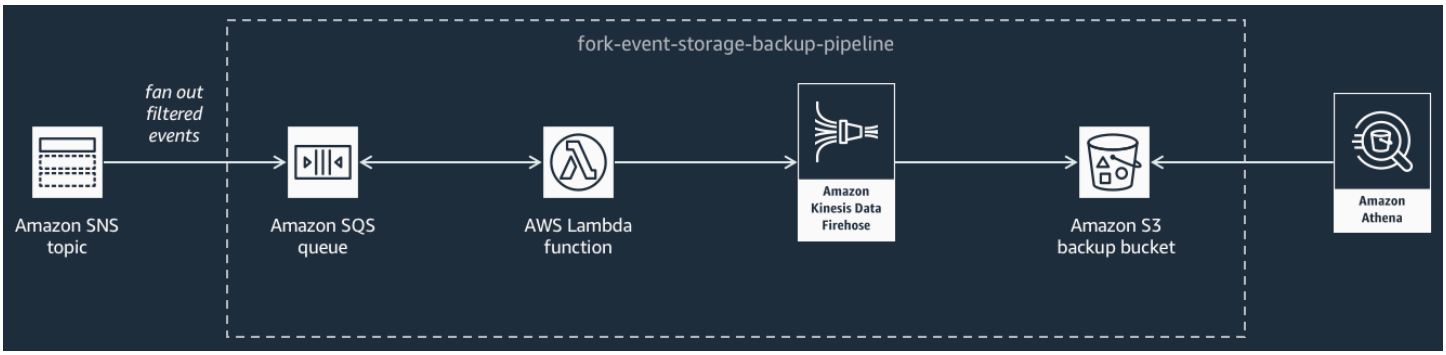
Chaque pipeline est abonné à la même rubrique Amazon SNS, ce qui lui permet de traiter les événements en parallèle au fur et à mesure de leur publication dans la rubrique. Chaque pipeline est indépendant et peut définir sa propre politique de filtre d'abonnement. Cela permet à un pipeline de traiter uniquement un sous-ensemble des événements qui l'intéressent (plutôt que tous les événements publiés dans la rubrique).

> **Note**
>
> Étant donné que vous placez les trois Event Fork Pipelines AWS avec vos pipelines de traitement des événements standard (peut-être déjà abonnés à votre rubrique Amazon SNS), vous n'avez pas besoin de modifier une partie de votre éditeur de messages actuel pour tirer parti des Event Fork Pipelines AWS dans vos applications existantes.

Pipeline de stockage et de sauvegarde d'événements

Le diagramme suivant montre le Pipeline de stockage et de sauvegarde d'événements. Vous pouvez abonner ce pipeline à votre rubrique Amazon SNS pour sauvegarder automatiquement les événements transitant par votre système.

Ce pipeline est composé d'une file d'attente Amazon SQS qui met en mémoire tampon les événements transmis par la rubrique Amazon SNS, d'une AWS Lambda fonction qui interroge automatiquement ces événements dans la file d'attente et les place dans un flux Amazon Data Firehose, et d'un compartiment Amazon S3 qui sauvegarde de manière durable les événements chargés par le flux.

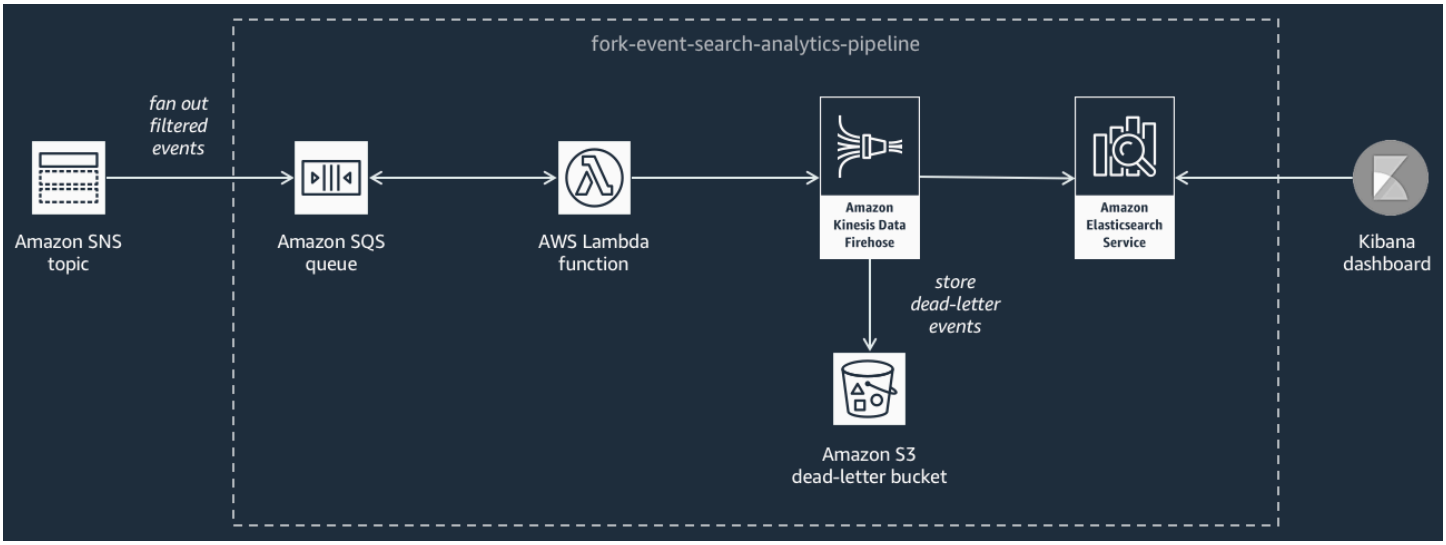


Pour affiner le comportement de votre flux Firehose, vous pouvez le configurer de façon à ce qu'il mette en tampon, transforme et compresse vos événements avant de les charger dans le compartiment. Pendant le chargement des événements , vous pouvez utiliser Amazon Athena pour interroger le compartiment à l'aide de requêtes SQL standard. Vous pouvez également configurer le pipeline de façon à réutiliser un compartiment Amazon S3 existant ou à en créer un nouveau.

Le pipeline de recherche et d'analyse d'événements

Le diagramme suivant montre le Pipeline de recherche et d'analyse d'événements. Vous pouvez abonner ce pipeline à votre rubrique Amazon SNS pour indexer les événements qui transitent via votre système dans un domaine de recherche, puis exécuter des analyses sur eux.

Ce pipeline est composé d'une file d'attente Amazon SQS qui met en mémoire tampon les événements fournis par la rubrique Amazon SNS, d'une AWS Lambda fonction qui extrait les événements de la file d'attente et les envoie dans un flux Amazon Data Firehose, d'un domaine OpenSearch Amazon Service qui indexe les événements chargés par le flux Firehose et d'un compartiment Amazon S3 qui stocke les événements en lettre morte qui ne peuvent pas être indexés dans le domaine de recherche.



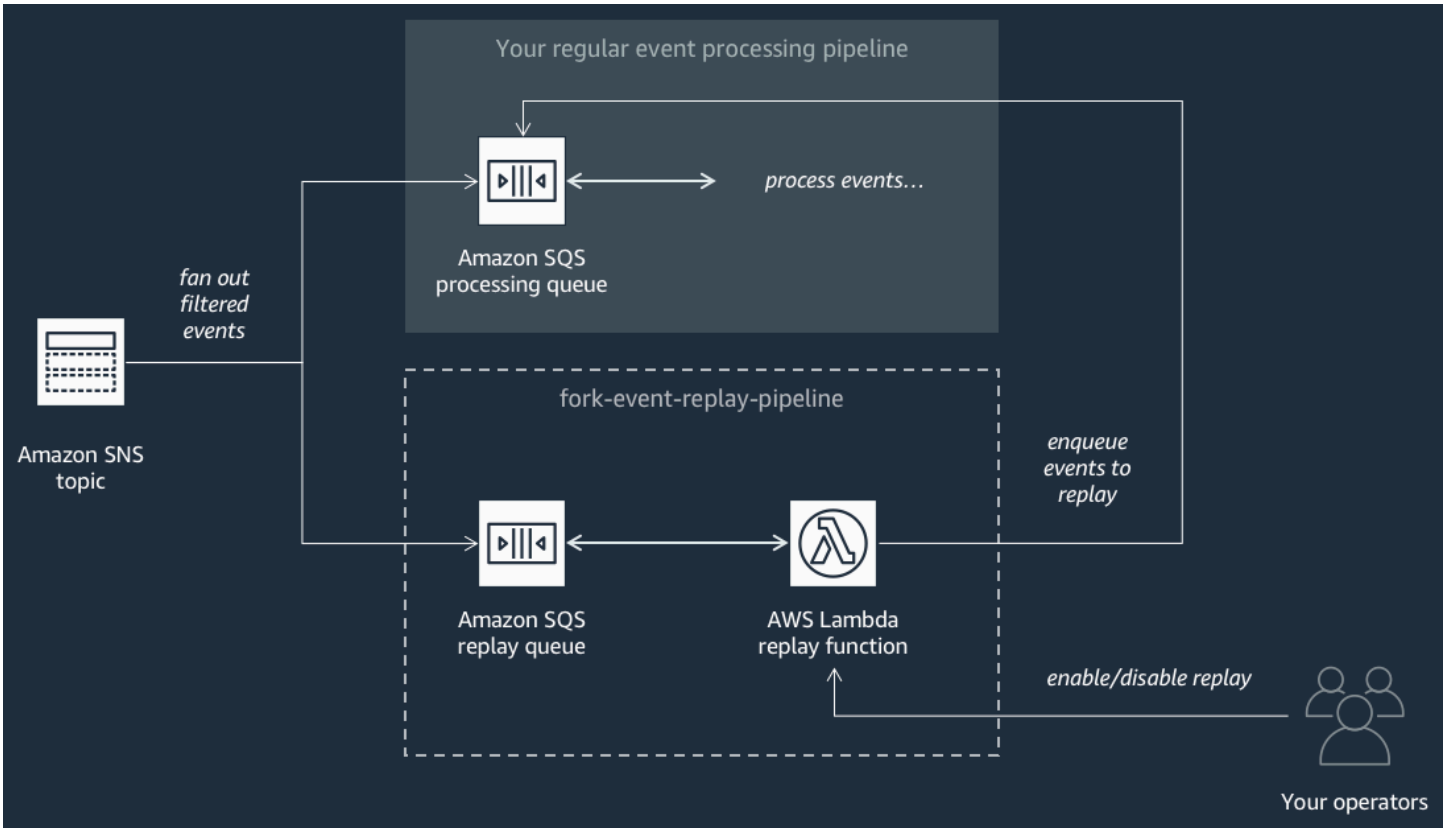
Pour affiner votre flux Firehose en termes de mise en mémoire tampon, de transformation et de compression des événements, vous pouvez configurer ce pipeline.

Vous pouvez également configurer si le pipeline doit réutiliser un OpenSearch domaine existant dans votre domaine Compte AWS ou en créer un nouveau pour vous. Lorsque les événements sont indexées dans le domaine de recherche, vous pouvez utiliser Kibana pour exécuter des analyses sur vos événements et mettre à jour les tableaux de bord visuels en temps réel.

Le pipeline de relecture d'événements

Le diagramme suivant montre le Pipeline de relecture d'événements. Pour enregistrer les événements qui ont été traités par votre système au cours des 14 derniers jours (par exemple, lorsque votre plateforme a besoin de récupérer après un incident), vous pouvez abonner ce pipeline à votre rubrique Amazon SNS, puis retraiter les événements.

Ce pipeline est composé d'une file d'attente Amazon SQS qui met en tampon les événements diffusés par la rubrique Amazon SNS, une fonction AWS Lambda qui interroge les événements à partir de la file d'attente et les redirige dans votre pipeline de traitement des événements standard, qui est également abonné à votre rubrique.



> **Note**
>
> Par défaut, la fonction de relecture est désactivée et ne relance pas vos événements. Si vous avez besoin de retraiter les événements, vous devez activer la file d'attente de relecture Amazon SQS en tant que source d'événement pour la fonction de relecture AWS Lambda.

Déploiement des AWSEvent Fork Pipelines

La AWSsuite d'Event Fork Pipelines (choisissez Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés) est disponible en tant que groupe d'applications publiques dans le AWS Serverless Application Repository, depuis lequel vous pouvez les déployer et les tester manuellement à l'aide de la AWS Lambdaconsole. Pour plus d'informations sur le déploiement des pipelines à l'aide de la console AWS Lambda, consultez la section Abonnement d'Event Fork Pipelines AWS à une rubrique Amazon SNS.

Dans un scénario de production, nous vous recommandons d'intégrer les Event Fork Pipelines AWS dans votre modèle d'application globale AWS SAM. La fonctionnalité d'application imbriquée vous permet de le faire en ajoutant la ressource `AWS::Serverless::Application` à votre modèle

AWS SAM, en référençant l AWS SAR `ApplicationId` et la `SemanticVersion` de l'application imbriquée.

Par exemple, vous pouvez utiliser le pipeline de stockage et de sauvegarde d'événements comme application imbriquée en ajoutant l'extrait de code YAML suivant à la section `Resources` de votre modèle AWS SAM.

```
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
```

Lorsque vous spécifiez des valeurs de paramètre, vous pouvez utiliser des fonctions intrinsèques AWS CloudFormation pour référencer d'autres ressources dans votre modèle. Par exemple, dans l'extrait YAML ci-dessus, le paramètre `TopicArn` référence la ressource `AWS::SNS::Topic` `MySNSTopic`, définie ailleurs dans le modèle AWS SAM. Pour plus d'informations, consultez la section de Référence des fonctions intrinsèques dans le AWS CloudFormationGuide de l'utilisateur.

> **Note**
>
> La page de la console AWS Lambda de votre application AWS SAR comprend le bouton Copier en tant que ressource SAM qui copie le fichier YAML requis pour l'imbrication d'une application SAR AWS dans le presse-papiers.

Déploiement et test de l'AWSExemple d'application d'Event Fork Pipelines

Pour accélérer le développement de vos applications basées sur les événements, vous pouvez abonner des pipelines de gestion des événements, à technologie Event Fork Pipelines AWS, vers les rubriques Amazon SNS.AWS Event Fork Pipelines est une suite d'applications imbriquées open source, basée sur le AWSmodèle d'application sans serveur (AWS SAM), que vous pouvez déployer directement à partir de la suite AWS Event Fork Pipelines (choisissez Show apps that create custom

IAM roles or resource policies (Afficher les appli qui créent des rôles IAM personnalisés ou des politiques de ressource) dans votre compte AWS. Pour de plus amples informations, consultez la section Fonctionnement des Event Fork PipelinesAWS.

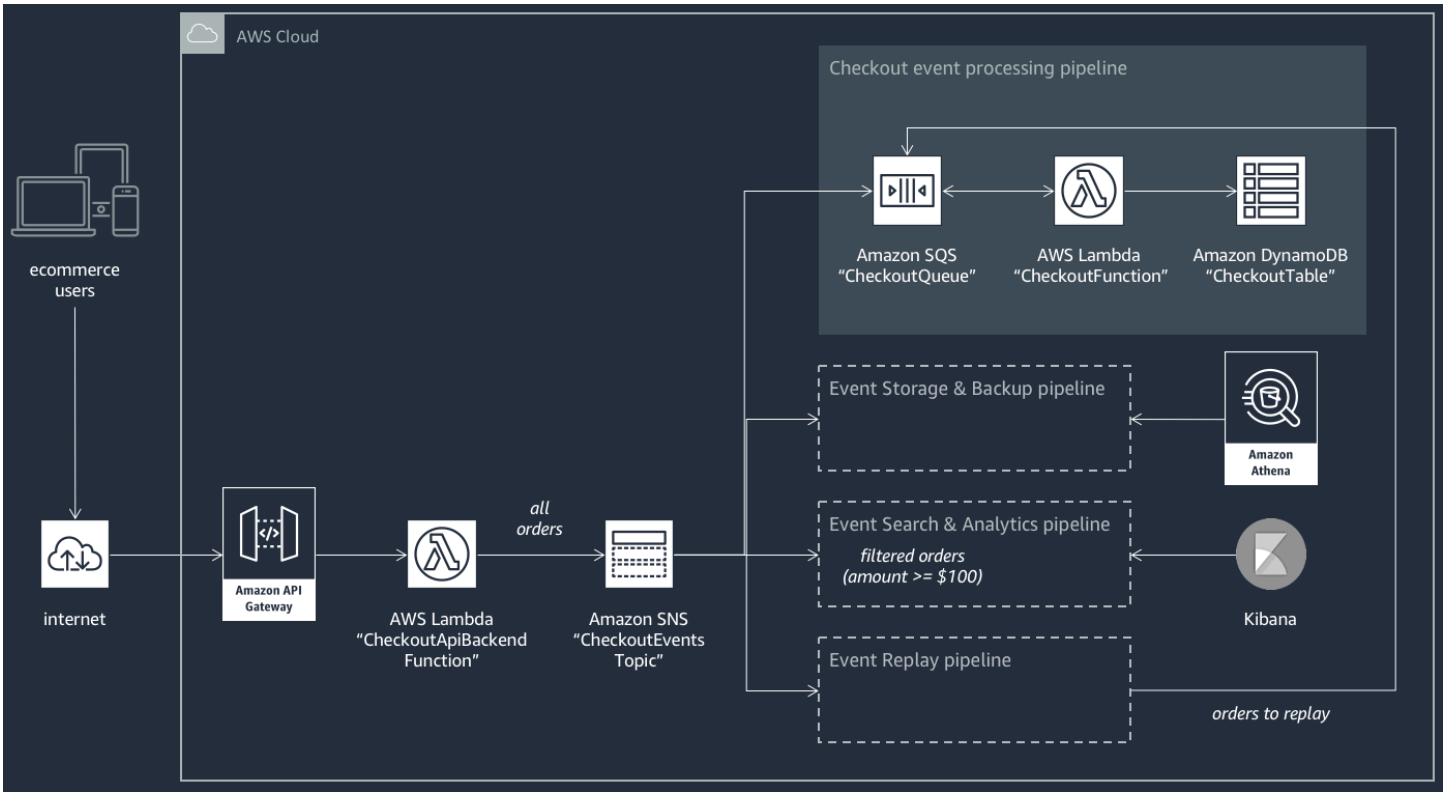
Cette page montre comment utiliser le AWS Management Console pour le déploiement et le test d'AWSexemple d'application d'Event Fork Pipelines.

> **Important**
>
> Afin de ne pas encourir des frais indésirables une fois que vous avez terminé le déploiement de l'exemple d'application d'Event Fork Pipelines AWS, supprimez sa pile AWS CloudFormation. Pour plus d'informations, consultez la section Suppression d'une pile dans la console AWS CloudFormation du Guide de l'utilisateur AWS CloudFormation.

Rubriques

Exemple AWS Cas d'utilisation d'Event Fork Pipelines

Le scénario suivant décrit une application d'e-commerce sans serveur dirigée par les événements qui utilise des Event Fork Pipelines AWS. Vous pouvez utiliser cet exemple d'application de commerce électronique dans le, AWS Serverless Application Repository puis le déployer dans votre Compte AWS AWS Lambda console, où vous pouvez le tester et examiner son code source GitHub.



Cette application d'e-commerce reçoit les commandes des acheteurs via une API RESTful hébergée par API Gateway et soutenue par la fonction AWS Lambda `CheckoutApiBackendFunction`. Cette fonction publie toutes les commandes reçues dans une rubrique Amazon SNS nommée `CheckoutEventsTopic`, qui transmet à son tour toutes les commandes à quatre différents pipelines.

Le premier pipeline est le pipeline de traitement standard des paiements conçu et mis en œuvre par le propriétaire de l'application d'e-commerce. Ce pipeline dispose de la file d'attente Amazon SQS `CheckoutQueue` qui met en mémoire tampon toutes les commandes reçues, une fonction AWS Lambda nommée `CheckoutFunction` qui interroge la file d'attente pour traiter ces commandes, et la table DynamoDB `CheckoutTable` qui enregistre en toute sécurité toutes les commandes passées.

Application d'Event Fork Pipelines AWS

Les composants de l'application d'e-commerce gèrent la logique d'entreprise de base. Toutefois, le propriétaire de l'application d'e-commerce propriétaire doit également prendre en compte les éléments suivants :

- Conformité - Sauvegardes sécurisées et compressées, chiffrées au repos et nettoyage des informations sensibles

- Résilience - Relecture de la plupart des commandes récentes en cas de défaillance du processus de réalisation
- Capacité de recherche - Exécution d'analyses et génération des métriques sur les commandes passées

Au lieu d'implémenter cette logique de traitement des événements, le propriétaire de l'application peut abonner des Event Fork Pipelines AWS à la rubrique Amazon SNS `CheckoutEventsTopic`

- Pipeline de stockage et de sauvegarde d'événements est configuré pour transformer les données afin de supprimer les détails des cartes de paiement, de mettre les données dans la mémoire tampon pendant 60 secondes, de les compresser à l'aide de GZIP et de les chiffrer à l'aide de la clé gérée par le client par défaut pour Amazon S3. Cette clé est gérée par AWS et optimisée par la technologie AWS Key Management Service (AWS KMS).

  Pour plus d'informations, consultez Choisir Amazon S3 pour votre destination, Amazon Data Firehose Data Transformation et Configurer les paramètres dans le manuel Amazon Data Firehose Developer Guide.
- Le pipeline de recherche et d'analyse d'événements est configuré avec un index de durée de nouvelle tentative de 30 secondes, un compartiment de stockage de commandes qui ne parviennent pas à être indexées dans le domaine de recherche et une politique de filtre permettant de limiter l'ensemble de commandes indexées.

  Pour plus d'informations, consultez Choisir le OpenSearch service pour votre destination dans le manuel Amazon Data Firehose Developer Guide.
- Le pipeline de relecture d'événements est configuré avec la partie de la file d'attente Amazon SQS du pipeline de traitement des commandes standard conçu et mis en œuvre par le propriétaire de l'application d'e-commerce.

  Pour de plus amples informations, consultez le Nom et l'URL d'une file d'attente dans Guide du développeur Amazon SQS.

La politique de filtre JSON suivante est définie dans la configuration du pipeline de recherche et d'analyse d'événements. Elle sélectionne uniquement les commandes entrantes dans lesquelles le montant total s'élève à 100 USD ou plus. Pour de plus amples informations, consultez la section Filtrage des messages Amazon SNS.

```
{
```

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

À l'aide du modèle d'Event Fork Pipelines AWS, le propriétaire de l'application d'e-commerce peut éviter la surcharge de développement qui suit généralement la logique de non différenciation du codage pour la gestion des événements. Au lieu de cela, elle peut déployer des Event Fork Pipelines AWS directement à partir de AWS Serverless Application Repository SAR dans son Compte AWS.

Étape 1 : Pour déployer l'exemple d'application

1. Connectez-vous à la AWS Lambdaconsole.
2. Dans le panneau de navigation, choisissez Fonctions, puis Créer une fonction.
3. Sur la page Créer une fonction, procédez de la façon suivante :

    a. Choisissez Parcourir le référentiel d'applications sans serveur, Applications publiques, Afficher les applications qui créent des politiques de rôles ou de ressources IAM personnalisées.

    b. Recherchez `fork-example-ecommerce-checkout-api` et choisissez l'application.
4. Sur la page fork-example-ecommerce-checkout-api, procédez comme suit :

    a. Dans la section Paramètres de l'application, entrez un Nom d'application (par exemple, `fork-example-ecommerce-my-app`).

    > **Note**
    >
    > - Pour rechercher vos ressources facilement plus tard, conservez le préfixe `fork-example-ecommerce`.
    > - Pour chaque déploiement, le nom de l'application doit être unique. Si vous réutilisez un nom d'application, le déploiement met à jour uniquement la pile AWS CloudFormation déployée précédemment (au lieu d'en créer une autre).

    b. (Facultatif) Entrez l'un des LogLevelparamètres suivants pour l'exécution de la fonction Lambda de votre application :

    - `DEBUG`
    - `ERROR`
    - `INFO` (default)

- `WARNING`

5. Choisissez I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Je comprends que cette application crée des rôles IAM et des politiques de ressources personnalisés, et déploie des applications imbriquées.) puis, en base de la page, choisissez Deploy (Déployer).

Sur la page État du déploiement pour fork-example-ecommerce - ***my-app***, Lambda affiche l'état Votre application est en cours de déploiement.

Dans la section Ressources, AWS CloudFormation commence à créer la pile et affiche l'état CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche l'état CREATE_COMPLETE.

> **Note**
>
> Le déploiement de toutes les ressources peut prendre 20-30 minutes.

Une fois le déploiement terminé, Lambda affiche l'état Votre application a été déployée.

Étape 2 : Pour exécuter l'exemple d'application

1. Dans la console AWS Lambda, sur le panneau de navigation, choisissez Applications.
2. Sur la page Applications, dans le champ de recherche, recherchez `serverlessrepo-fork-example-ecommerce-`*`my-app`*, puis choisissez l'application.
3. Dans la section Ressources, effectuez les opérations suivantes :

   a. Pour trouver la ressource dont le type est ApiGatewayRestApi, triez les ressources par type, par exemple`ServerlessRestApi`, puis développez la ressource.

   b. Deux ressources imbriquées sont affichées, de types ApiGatewayDeployment et ApiGatewayStage.

   c. Copiez le lien Point de terminaison de l'API de production et ajoutez-lui `/checkout`, par exemple :

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

4. Copiez le code JSON suivant dans un fichier nommé `test_event.json`.

```
{
   "id": 15311,
   "date": "2019-03-25T23:41:11-08:00",
   "status": "confirmed",
   "customer": {
      "id": 65144,
      "name": "John Doe",
      "email": "john.doe@example.com"
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
      "currency": "usd",
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. Pour envoyer une demande HTTPS au point de terminaison de votre API, transmettez la charge utile de l'exemple d'événement comme entrée en exécutant une commande `curl`, par exemple :

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

L'API renvoie la réponse vide suivante, indiquant une exécution réussie :

```
{ }
```

Étape 3 : Pour vérifier l'exécution de l'exemple d'application et de ses pipelines

Étape 1 : Pour vérifier l'exécution de l'exemple de pipeline de paiement

1. Connectez-vous à la console Amazon DynamoDB.
2. Dans le panneau de navigation, choisissez Tables.
3. Recherchez `serverlessrepo-fork-example` et choisissez `CheckoutTable`.
4. Dans la page des détails de la table, choisissez Éléments, puis choisissez l'élément créé.

   Les attributs stockés sont affichés.

Étape 2 : Pour vérifier l'exécution du pipeline de sauvegarde et de stockage d'événements

1. Connectez-vous à la console Amazon S3.
2. Dans le panneau de navigation, choisissez Compartiments.
3. Recherchez `serverlessrepo-fork-example` et choisissez `CheckoutBucket`.
4. Parcourez la hiérarchie de répertoires jusqu'à ce que vous trouviez un fichier avec l'extension `.gz`.
5. Pour télécharger le fichier, choisissez Actions, Ouvrir.
6. Le pipeline est configuré avec une fonction Lambda qui nettoie les informations de carte de paiement pour des raisons de conformité.

   Pour vérifier que la charge utile JSON stockée ne contient pas d'informations sur les cartes de paiement, décompressez le fichier.

Étape 3 : Pour vérifier l'exécution du pipeline de recherche et d'analyse d'événements

1. Connectez-vous à la console OpenSearch de service.
2. Dans le panneau de navigation, sous Mes domaines, choisissez le domaine préfixé avec `serverl-analyt`.
3. Le pipeline est configuré avec une politique de filtre d'abonnement Amazon SNS qui définit une condition de correspondance numérique.

   Pour vérifier que l'événement est indexé parce qu'il fait référence à une commande dont la valeur est supérieure à 100 USD, sur la page serverl-analyt-***abcdefgh1ijk***, choisissez Indices (Index), checkout_events.

Étape 4 : Pour vérifier l'exécution du pipeline de relecture d'événements

1. Connectez-vous à la console Amazon SQS.
2. Dans la liste des files d'attente, recherchez `serverlessrepo-fork-example` et choisissez `ReplayQueue`.
3. Choisissez Envoyer et recevoir des messages.
4. Dans la boîte de dialogue Envoyer et recevoir des messages dans fork-example-ecommerce - ***my-app***... ReplayP- ReplayQueue - ***123ABCD4E5F6***, choisissez Sondage pour les messages.
5. Pour vérifier que l'événement est mis en file d'attente, choisissez Plus de détails en regard du message qui s'affiche dans la file d'attente.

Étape 4 : Pour simuler un problème et relire les événements en vue d'une reprise

Étape 1 : Pour activer le problème simulé et envoyer une deuxième demande d'API

1. Connectez-vous à la AWS Lambdaconsole.
2. Dans le panneau de navigation, choisissez Fonctions.
3. Recherchez `serverlessrepo-fork-example` et choisissez `CheckoutFunction`.
4. Sur le fork-example-ecommerce- ***my-app*** - CheckoutFunction - ***ABCDEF***... page, dans la section Variables d'environnement, définissez la variable BUG_ENABLED sur true, puis choisissez Enregistrer.
5. Copiez le code JSON suivant dans un fichier nommé `test_event_2.json`.

```
{
```

```
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
    "payment": {
       "id": 3311,
       "amount": 75.00,
       "currency": "usd",
       "method": "credit",
       "card-network": "mastercard",
       "card-number": "1234 5678 9012 3456",
       "card-expiry": "12/2025",
       "card-owner": "Marcia Oliveira",
       "card-cvv": "321"
    },
    "shipping": {
       "id": 9900,
       "time": 20,
       "unit": "days",
       "method": "plane"
    },
    "items": [{
       "id": 9993,
       "product": 3120,
       "name": "Hockey Stick",
       "quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Pour envoyer une demande HTTPS au point de terminaison de votre API, transmettez la charge utile de l'exemple d'événement comme entrée en exécutant une commande `curl`, par exemple :

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

L'API renvoie la réponse vide suivante, indiquant une exécution réussie :

```
{ }
```

Étape 2 : Pour vérifier la simulation de corruption des données

1. Connectez-vous à la console Amazon DynamoDB.
2. Dans le panneau de navigation, choisissez Tables.
3. Recherchez `serverlessrepo-fork-example` et choisissez `CheckoutTable`.
4. Dans la page des détails de la table, choisissez Éléments, puis choisissez l'élément créé.

   Les attributs stockés sont affichés, certains étant marqués comme CORROMPU !

Étape 3 : Pour désactiver la simulation du problème

1. Connectez-vous à la AWS Lambdaconsole.
2. Dans le panneau de navigation, choisissez Fonctions.
3. Recherchez `serverlessrepo-fork-example` et choisissez `CheckoutFunction`.
4. Sur le fork-example-ecommerce- ***`my-app`*** - CheckoutFunction - ***`ABCDEF`***... page, dans la section Variables d'environnement, définissez la variable BUG_ENABLED sur false, puis choisissez Enregistrer.

Étape 4 : Pour activer la relecture en vue d'une reprise après un problème

1. Dans la console AWS Lambda, sur le panneau de navigation, choisissez Fonctions.
2. Recherchez `serverlessrepo-fork-example` et choisissez `ReplayFunction`.
3. Développez la section Concepteur, choisissez la vignette SQS puis, dans la section SQS, choisissez Activé.

   > **Note**
   >
   > Il faut compter environ 1 minute pour que le déclencheur source d'événements Amazon SQS soit activé.

4. Choisissez Enregistrer.
5. Pour afficher les attributs récupérées, revenez à la console Amazon DynamoDB.

6. Pour désactiver la relecture, revenez à la console AWS Lambda et désactivez le déclencheur source d'événements Amazon SQS pour `ReplayFunction`.

Abonnement d'Event Fork Pipelines AWS à une rubrique Amazon SNS

Pour accélérer le développement de vos applications basées sur les événements, vous pouvez abonner des pipelines de gestion des événements, à technologie d'Event Fork Pipelines AWS, aux rubriques Amazon SNS.AWS Event Fork Pipelines est une suite d'applications imbriquées open source, basée sur le AWSmodèle d'application sans serveur (AWS SAM), que vous pouvez déployer directement à partir de la suite AWS Event Fork Pipelines (choisissez Show apps that create custom IAM roles or resource policies (Afficher les appli qui créent des rôles IAM personnalisés ou des politiques de ressource) dans votre compte AWS. Pour de plus amples informations, consultez la section Fonctionnement des Event Fork PipelinesAWS.

Cette section montre comment vous pouvez utiliser le AWS Management Console pour déployer un pipeline, puis abonner des Event Fork Pipelines AWS à une rubrique Amazon SNS. Avant de commencer, créez une rubrique Amazon SNS.

Pour supprimer les ressources qui constituent un pipeline, recherchez le pipeline sur la page Applications de la AWS Lambda console, développez la section du modèle SAM, choisissez CloudFormationStack, puis choisissez Other Actions, Delete Stack.

Rubriques

- Pour déployer et abonner le pipeline de stockage et de sauvegarde d'événements
- Pour déployer et abonner le pipeline de recherche et d'analyse d'événements
- Pour déployer et abonner le pipeline de relecture d'événements

Pour déployer et abonner le pipeline de stockage et de sauvegarde d'événements

Pour l'archivage et l'analyse des événements, Amazon SNS recommande désormais d'utiliser son intégration native avec Amazon Data Firehose. Vous pouvez abonner les flux de diffusion Firehose aux rubriques SNS, ce qui vous permet d'envoyer des notifications aux points de terminaison d'archivage et d'analyse tels que les buckets Amazon Simple Storage Service (Amazon S3), les tables Amazon Redshift, Amazon Service (Service), etc. OpenSearch OpenSearch L'utilisation d'Amazon SNS avec les flux de diffusion Firehose est une solution

entièrement gérée et sans code qui ne nécessite pas l'utilisation de fonctions. AWS Lambda Pour plus d'informations, consultez Streams de diffusion de Fanout to Firehose.

Ce didacticiel montre comment déployer le Pipeline de stockage et de sauvegarde d'événements et l'abonner à une rubrique Amazon SNS. Ce processus transforme automatiquement le modèle AWS SAM associé au pipeline en pile AWS CloudFormation, puis déploie la pile dans votre Compte AWS. De plus, ce processus crée et configure l'ensemble de ressources qui comprennent le pipeline de stockage et de sauvegarde des événements, y compris ce qui suit :

- File d'attente Amazon SQS
- Fonction Lambda
- Flux de diffusion Firehose
- Compartiment de sauvegarde Amazon S3

Pour plus d'informations sur la configuration d'un flux avec un compartiment S3 comme destination, consultez le `S3DestinationConfiguration` manuel Amazon Data Firehose API Reference.

Pour plus d'informations sur la transformation des événements et sur la configuration de la mise en mémoire tampon des événements, de la compression des événements et du chiffrement des événements, consultez la section Création d'un flux de diffusion Amazon Data Firehose dans le manuel du développeur Amazon Data Firehose.

Pour plus d'informations sur le filtrage des événements, consultez la section Stratégies de filtre d'abonnement Amazon SNS dans le présent guide.

1. Connectez-vous à la AWS Lambdaconsole.
2. Dans le panneau de navigation, choisissez Fonctions, puis Créer une fonction.
3. Sur la page Créer une fonction, procédez de la façon suivante :
   a. Choisissez Parcourir le référentiel d'applications sans serveur, Applications publiques, Afficher les applications qui créent des politiques de rôles ou de ressources IAM personnalisées.
   b. Recherchez `fork-event-storage-backup-pipeline` et choisissez l'application.
4. Sur la page fork-event-storage-backup-pipeline, procédez comme suit :

a. Dans la section Paramètres de l'application, entrez un Nom d'application (par exemple, `my-app-backup`).

> **Note**
>
> - Pour chaque déploiement, le nom de l'application doit être unique. Si vous réutilisez un nom d'application, le déploiement met à jour uniquement la pile AWS CloudFormation déployée précédemment (au lieu d'en créer une autre).

b. (Facultatif) Pour BucketArn, entrez l'ARN du compartiment S3 dans lequel les événements entrants sont chargés. Si vous ne saisissez pas de valeur, un nouveau compartiment S3 est créé dans votre compte AWS.

c. (Facultatif) Pour DataTransformationFunctionArn, entrez l'ARN de la fonction Lambda par laquelle les événements entrants sont transformés. Si vous ne spécifiez pas de valeur, la transformation des données est désactivée.

d. (Facultatif) Entrez l'un des LogLevelparamètres suivants pour l'exécution de la fonction Lambda de votre application :

- `DEBUG`
- `ERROR`
- `INFO` (default)
- `WARNING`

e. Pour TopicArn, entrez l'ARN de la rubrique Amazon SNS à laquelle cette instance du pipeline de fork doit être abonnée.

f. (Facultatif) Pour StreamBufferingIntervalInSecondset StreamBufferingSizeInMBs, entrez les valeurs permettant de configurer la mise en mémoire tampon des événements entrants. Si vous n'entrez aucune valeur, 300 secondes et 5 Mo sont utilisés.

g. (Facultatif) Entrez l'un des StreamCompressionFormatparamètres suivants pour compresser les événements entrants :

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (default)
- `ZIP`

h. (Facultatif) Pour StreamPrefix, entrez le préfixe de chaîne pour nommer les fichiers stockés dans le compartiment de sauvegarde S3. Si vous ne spécifiez pas de valeur, aucun préfixe n'est utilisé.

i. (Facultatif) Pour SubscriptionFilterPolicy, entrez la politique de filtrage des abonnements Amazon SNS, au format JSON, à utiliser pour filtrer les événements entrants. La politique de filtrage détermine quels événements sont indexés dans l'index des OpenSearch services. Si vous ne spécifiez pas de valeur, aucun filtrage n'est utilisé (tous les événements sont indexés).

j. (Facultatif) Pour SubscriptionFilterPolicyScope, entrez la chaîne `MessageBody` ou `MessageAttributes` pour activer le filtrage des messages basé sur la charge utile ou basé sur les attributs.

k. Choisissez Je comprends que cette application crée des rôles IAM et des politiques de ressources personnalisés, et déploie des applications imbriquées., puis choisissez Déployer.

Sur la page État de déploiement pour ***my-app***, Lambda affiche l'état Votre application est en cours de déploiement.

Dans la section Ressources, AWS CloudFormation commence à créer la pile et affiche l'état CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche l'état CREATE_COMPLETE.

Une fois le déploiement terminé, Lambda affiche l'état Votre application a été déployée.

Les messages publiés dans votre rubrique Amazon SNS sont stockés dans le compartiment S3 de sauvegarde mis en service automatiquement par le pipeline de stockage et de sauvegarde d'événements.

Pour déployer et abonner le pipeline de recherche et d'analyse d'événements

Pour l'archivage et l'analyse des événements, Amazon SNS recommande désormais d'utiliser son intégration native avec Amazon Data Firehose. Vous pouvez abonner les flux de diffusion Firehose aux rubriques SNS, ce qui vous permet d'envoyer des notifications aux points de terminaison d'archivage et d'analyse tels que les buckets Amazon Simple Storage Service (Amazon S3), les tables Amazon Redshift, Amazon Service (Service), etc. OpenSearch OpenSearch L'utilisation d'Amazon SNS avec les flux de diffusion Firehose est une solution entièrement gérée et sans code qui ne nécessite pas l'utilisation de fonctions. AWS Lambda Pour plus d’informations, consultez Streams de diffusion de Fanout to Firehose.

Cette page montre comment déployer le Pipeline de recherche et d'analyse des événements et vous abonner à une rubrique Amazon SNS. Ce processus transforme automatiquement le modèle AWS SAM associé au pipeline en pile AWS CloudFormation, puis déploie la pile dans votre Compte AWS. De plus, ce processus crée et configure l'ensemble de ressources qui comprennent le pipeline de recherche et d'analyse des événements, y compris ce qui suit :

- File d’attente Amazon SQS
- Fonction Lambda
- Flux de diffusion Firehose
- Domaine Amazon OpenSearch Service
- Compartiment de lettres mortes Amazon S3

Pour plus d'informations sur la configuration d'un flux avec un index comme destination, consultez le `ElasticsearchDestinationConfiguration` manuel Amazon Data Firehose API Reference.

Pour plus d'informations sur la transformation des événements et sur la configuration de la mise en mémoire tampon des événements, de la compression des événements et du chiffrement des événements, consultez la section Création d'un flux de diffusion Amazon Data Firehose dans le manuel du développeur Amazon Data Firehose.

Pour plus d'informations sur le filtrage des événements, consultez la section Stratégies de filtre d'abonnement Amazon SNS dans le présent guide.

1. Connectez-vous à la AWS Lambdaconsole.
2. Dans le panneau de navigation, choisissez Fonctions, puis Créer une fonction.
3. Sur la page Créer une fonction, procédez de la façon suivante :
    a. Choisissez Parcourir le référentiel d'applications sans serveur, Applications publiques, Afficher les applications qui créent des politiques de rôles ou de ressources IAM personnalisées.
    b. Recherchez `fork-event-search-analytics-pipeline` et choisissez l'application.
4. Sur la page fork-event-search-analytics-pipeline, procédez comme suit :
    a. Dans la section Paramètres de l'application, entrez un Nom d'application (par exemple, `my-app-search`).

> **Note**
>
> Pour chaque déploiement, le nom de l'application doit être unique. Si vous réutilisez un nom d'application, le déploiement met à jour uniquement la pile AWS CloudFormation déployée précédemment (au lieu d'en créer une autre).

b. (Facultatif) Pour DataTransformationFunctionArn, entrez l'ARN de la fonction Lambda utilisée pour transformer les événements entrants. Si vous ne spécifiez pas de valeur, la transformation des données est désactivée.

c. (Facultatif) Entrez l'un des LogLevelparamètres suivants pour l'exécution de la fonction Lambda de votre application :

- `DEBUG`
- `ERROR`
- `INFO` (default)
- `WARNING`

d. (Facultatif) Pour SearchDomainArn, entrez l'ARN du domaine de OpenSearch service, un cluster qui configure les fonctionnalités de calcul et de stockage nécessaires. Si vous ne spécifiez pas de valeur, un nouveau domaine est créé avec la configuration par défaut.

e. Pour TopicArn, entrez l'ARN de la rubrique Amazon SNS à laquelle cette instance du pipeline de fork doit être abonnée.

f. Pour SearchIndexName, entrez le nom de l'index des OpenSearch services pour la recherche et l'analyse des événements.

> **Note**
>
> Les quotas suivants s'appliquent aux noms d'index :
>
> - Vous ne pouvez pas inclure de majuscules
> - Vous ne pouvez pas inclure les caractères suivants : `\ / * ? " < > | ` , #`
> - Vous ne pouvez pas commencer par les caractères suivants : `- + _`
> - Vous ne pouvez pas utiliser : `. ..`
> - Vous ne pouvez pas utiliser plus de 80 caractères
> - Vous ne pouvez pas utiliser plus de 255 octets

- Ne peut pas contenir de deux-points (depuis OpenSearch Service 7.0)

g. (Facultatif) Entrez l'un des SearchIndexRotationPeriodparamètres suivants pour la période de rotation de l'indice de OpenSearch service :

- `NoRotation` (default)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

La rotation d'index ajoute un horodatage au nom de l'index, ce qui facilite l'expiration des anciennes données.

h. Pour SearchTypeName, entrez le nom du type de OpenSearch service pour organiser les événements dans un index.

> **Note**
>
> - OpenSearch Les noms de type de service peuvent contenir n'importe quel caractère (à l'exception des octets nuls), mais ils ne peuvent pas commencer par_.
> - Pour le OpenSearch Service 6.x, il ne peut y avoir qu'un seul type par index. Si vous spécifiez un nouveau type pour un index existant qui possède déjà un autre type, Firehose renvoie une erreur d'exécution.

i. (Facultatif) Pour StreamBufferingIntervalInSecondset StreamBufferingSizeInMBs, entrez les valeurs permettant de configurer la mise en mémoire tampon des événements entrants. Si vous n'entrez aucune valeur, 300 secondes et 5 Mo sont utilisés.

j. (Facultatif) Entrez l'un des StreamCompressionFormatparamètres suivants pour compresser les événements entrants :

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (default)
- `ZIP`

k. (Facultatif) Pour StreamPrefix, entrez le préfixe de chaîne pour nommer les fichiers stockés dans le compartiment de lettres mortes S3. Si vous ne spécifiez pas de valeur, aucun préfixe n'est utilisé.

l. (Facultatif) Pour StreamRetryDurationInSecons, entrez la durée des nouvelles tentatives dans les cas où Firehose ne parvient pas à indexer les événements dans OpenSearch l'index des services. Si vous ne spécifiez pas de valeur, 300 secondes seront utilisées.

m. (Facultatif) Pour SubscriptionFilterPolicy, entrez la politique de filtrage des abonnements Amazon SNS, au format JSON, à utiliser pour filtrer les événements entrants. La politique de filtrage détermine quels événements sont indexés dans l'index des OpenSearch services. Si vous ne spécifiez pas de valeur, aucun filtrage n'est utilisé (tous les événements sont indexés).

n. Choisissez Je comprends que cette application crée des rôles IAM et des politiques de ressources personnalisés, et déploie des applications imbriquées., puis choisissez Déployer.

Sur la *my-app-search*page État du déploiement pour, Lambda affiche l'état Votre application est en cours de déploiement.

Dans la section Ressources, AWS CloudFormation commence à créer la pile et affiche l'état CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche l'état CREATE_COMPLETE.

Une fois le déploiement terminé, Lambda affiche l'état Votre application a été déployée.

Les messages publiés sur votre rubrique Amazon SNS sont indexés automatiquement dans l'index des OpenSearch services fourni par le pipeline Event Search and Analytics. Si le pipeline ne peut pas indexer un événement, il le stocke dans un compartiment S3 de lettres mortes.

Pour déployer et abonner le pipeline de relecture d'événements

Cette page montre comment déployer le Pipeline de relecture des événements et vous abonner à une rubrique Amazon SNS. Ce processus transforme automatiquement le modèle AWS SAM associé au pipeline en pile AWS CloudFormation, puis déploie la pile dans votre Compte AWS. De plus, ce processus crée et configure l'ensemble de ressources qui comprennent le pipeline de relecture d'événements, y compris une file d'attente Amazon SQS et une fonction Lambda.

Pour plus d'informations sur le filtrage des événements, consultez la section Stratégies de filtre d'abonnement Amazon SNS dans le présent guide.

1. Connectez-vous à la AWS Lambdaconsole.
2. Dans le panneau de navigation, choisissez Fonctions, puis Créer une fonction.
3. Sur la page Créer une fonction, procédez de la façon suivante :

   a. Choisissez Parcourir le référentiel d'applications sans serveur, Applications publiques, Afficher les applications qui créent des politiques de rôles ou de ressources IAM personnalisées.

   b. Recherchez `fork-event-replay-pipeline` et choisissez l'application.
4. Sur la page fork-event-replay-pipeline, procédez de la façon suivante :

   a. Dans la section Paramètres de l'application, entrez un Nom d'application (par exemple, `my-app-replay`).

   > **Note**
   >
   > Pour chaque déploiement, le nom de l'application doit être unique. Si vous réutilisez un nom d'application, le déploiement met à jour uniquement la pile AWS CloudFormation déployée précédemment (au lieu d'en créer une autre).

   b. (Facultatif) Entrez l'un des LogLevelparamètres suivants pour l'exécution de la fonction Lambda de votre application :

   - `DEBUG`
   - `ERROR`
   - `INFO` (default)
   - `WARNING`

   c. (Facultatif ReplayQueueRetentionPeriodInSeconds) Entrez la durée, en secondes, pendant laquelle la file de rediffusion Amazon SQS conserve le message. Si vous ne spécifiez pas de valeur, 1 209 600 secondes (14 jours) est utilisé.

   d. Pour TopicArn, entrez l'ARN de la rubrique Amazon SNS à laquelle cette instance du pipeline de fork doit être abonnée.

   e. Pour DestinationQueueName, entrez le nom de la file d'attente Amazon SQS à laquelle la fonction Lambda replay transmet les messages.

   f. (Facultatif) Pour SubscriptionFilterPolicy, entrez la politique de filtrage des abonnements Amazon SNS, au format JSON, à utiliser pour filtrer les événements entrants. La politique de filtre décide quels événements sont mis en mémoire tampon pour la relecture. Si vous

ne spécifiez pas de valeur, aucun filtrage n'est utilisé (tous les événements sont mis en mémoire tampon pour la relecture).

g. Choisissez Je comprends que cette application crée des rôles IAM et des politiques de ressources personnalisés, et déploie des applications imbriquées., puis choisissez Déployer.

Sur la *my-app-replay*page État du déploiement pour, Lambda affiche l'état Votre application est en cours de déploiement.

Dans la section Ressources, AWS CloudFormation commence à créer la pile et affiche l'état CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche l'état CREATE_COMPLETE.

Une fois le déploiement terminé, Lambda affiche l'état Votre application a été déployée.

Les messages publiés dans votre rubrique Amazon SNS sont mis en mémoire tampon dans la file d'attente Amazon SQS mise en service automatiquement par le pipeline de relecture d'événements.

> **Note**
>
> Par défaut, la relecture est désactivée. Pour activer la relecture, accédez à la page de la fonction sur la console Lambda, développez la section Concepteur, choisissez la vignette SQS puis, dans la section SQS, choisissez Activé.

Utilisation du planificateur Amazon EventBridge avec Amazon SNS

Le planificateur Amazon EventBridge est un planificateur sans serveur qui vous permet de créer, d'exécuter et de gérer des tâches à partir d'un service central et géré. Avec le planificateur EventBridge, vous pouvez créer des planifications en utilisant des expressions Cron et rate pour les modèles récurrents, ou configurer des invocations ponctuelles. Vous pouvez configurer des fenêtres de temps flexibles pour la livraison, définir des limites de nouvelles tentatives ainsi que la durée de conservation maximale pour les invocations d'API en échec.

Cette page explique comment utiliser le planificateur EventBridge pour publier un message depuis une rubrique Amazon SNS selon une planification.

Rubriques

- Configurer le rôle d'exécution

- Créer une planification
- Ressources connexes

Configurer le rôle d'exécution

Lorsque vous créez une planification, le planificateur EventBridge doit être autorisé à invoquer son opération d'API cible en votre nom. Vous accordez ces autorisations au planificateur EventBridge à l'aide d'un rôle d'exécution. La politique d'autorisation que vous associez au rôle d'exécution de votre planification définit les autorisations requises. Ces autorisations dépendent de l'API cible que vous souhaitez que le planificateur EventBridge invoque.

Lorsque vous utilisez la console du planificateur EventBridge pour créer une planification, comme dans la procédure suivante, le planificateur EventBridge définit automatiquement un rôle d'exécution en fonction de la cible que vous avez sélectionnée. Si vous souhaitez créer une planification à l'aide de l'un des kits SDK du planificateur EventBridge, de la AWS CLI ou de AWS CloudFormation, vous devez disposer d'un rôle d'exécution existant qui accorde les autorisations dont le planificateur EventBridge a besoin pour invoquer une cible. Pour plus d'informations sur la configuration manuelle d'un rôle d'exécution pour votre planification, voir Configuration d'un rôle d'exécution dans le Guide de l'utilisateur du planificateur EventBridge.

Créer une planification

Pour créer une planification à l'aide de la console

1. Ouvrez la console du planificateur Amazon EventBridge à l'adresse https://console.aws.amazon.com/scheduler/home.
2. Sur la page Planifications, choisissez Créer une planification.
3. Sur la page Spécifier le détail de la planification, dans la section Nom et description de la planification, procédez comme suit :

    a. Pour Nom de la planification, saisissez un nom à attribuer à votre planification. Par exemple, **`MyTestSchedule`**.

    b. (Facultatif) Dans le champ Description, saisissez une description de la planification. Par exemple, **`My first schedule`**.

    c. Pour Groupe de planifications, choisissez un groupe de planifications dans la liste déroulante. Si vous n'avez pas de groupe, choisissez par défaut. Pour créer un groupe de planifications, choisissez Crée votre propre planification.

Vous utilisez des groupes de planifications pour leur ajouter des balises.

4. • Choisissez vos options de planification.

Occurrence	Faites ceci...	
Planification ponctuelle Une planification ponctuelle n'invoque un objectif qu'une seule fois à la date et à l'heure que vous indiquez.	Pour Date et heure, procédez comme suit : • Entrez une date valide au format `YYYY/MM/DD` . • Entrez un horodatage au format `hh:mm` de 24 heures. • Dans le champ Fuseau horaire, choisissez le fuseau horaire.	
Planification récurrente Une planification récurrente invoque un objectif à un taux que vous spécifiez à l'aide d'une expression cron ou d'une expression rate.	a. Pour Schedule type (Planifier le type), effectuez l'une des étapes suivantes : • Pour utiliser une expression cron afin de définir la planification, choisissez Planification basée sur cron et entrez l'expression cron. • Pour utiliser une expression de rythme pour définir la planification, choisissez Planification basée sur le rythme.	

Occurrence	Faites ceci...	
	Pour plus d'informations sur les expressions cron et rate, consultez Types de planifications sur le planificateur EventBridge dans le Guide de l'utilisateur du planificateur Amazon EventBridge. b. Pour Fenêtre temporelle flexible, choisissez Désactivé pour désactiver cette option ou choisir l'une des fenêtres temporelles prédéfinies. Par exemple, si vous choisissez15 minutes et que vous définissez une planification récurrente pour invoquer son objectif une fois par heure, la planification s'exécute dans les 15 minutes suivant le début de chaque heure.	

5. (Facultatif) Si vous avez choisi Planification récurrente à l'étape précédente, dans la section Délai, procédez comme suit :

   a. Dans le champ Fuseau horaire, choisissez un fuseau horaire.

   b. Pour Date et heure de début, entrez une date valide au format `YYYY/MM/DD`, puis spécifiez un horodatage au format hh`:mm` de 24 heures.

c. Pour Date et heure de fin, entrez une date valide au format `YYYY/MM/DD`, puis spécifiez un horodatage au format `hh:mm` de 24 heures.

6. Choisissez Next (Suivant).
7. Sur la page Sélectionner la cible, choisissez l'opération d'API AWS invoquée par le planificateur EventBridge :

   a. Sélectionnez Publication Amazon SNS.

   b. Dans la section Publier, sélectionnez une rubrique SNS ou choisissez Créer une rubrique SNS.

   c. (Facultatif) Saisissez une charge utile JSON. Si vous n'entrez aucune charge utile, le planificateur EventBridge utilise un événement vide pour invoquer la fonction.

8. Choisissez Next (Suivant).
9. Sur la page Settings (Paramètres), procédez comme suit :

   a. Pour activer la planification, sous État de la planification, activez Activer la planification.

   b. Pour configurer une stratégie de nouvelles tentatives pour votre planification, sous Politique de nouvelle tentative et file d'attente de lettres mortes (DLQ), procédez comme suit :

      - Activez Réessayer.
      - Pour Âge maximum de l'événement, entrez le nombre maximum d'heures et de minutes de conservation d'un événement non traité par le planificateur EventBridge.
      - La durée maximale est 24 heures.
      - Pour Nombre maximum de tentatives, entrez le nombre maximum de tentatives de renvoi d'une erreur par le planificateur EventBridge.

        La valeur maximale est 185 nouvelles tentatives.

      Avec les stratégies de nouvelles tentatives, si une planification ne parvient pas à invoquer sa cible, le planificateur EventBridge la réexécute. Si elle est configurée, vous devez définir la durée de rétention maximale et les nouvelles tentatives pour la planification.

   c. Choisissez où le planificateur EventBridge stocke les événements non livrés.

Option File d'attente de lettres mortes (DLQ)	Faites ceci...
Ne stockez pas	Sélectionnez Aucun.
Stocker l'événement dans le même Compte AWS où vous créez la planification	a. Choisissez Sélectionnez une file d'attente Amazon SQS dans mon Compte AWS en tant que DLQ. b. Choisissez l'Amazon Resource Name (ARN) de la file d'attente Amazon SQS.
Stocker l'événement dans un autre Compte AWS que celui où vous créez la planification	a. Choisissez Spécifier une file d'attente Amazon SQS dans un autre Comptes AWS en tant que DLQ. b. Entrez l'Amazon Resource Name (ARN) de la file d'attente Amazon SQS.

d. Pour utiliser une clé gérée par le client afin de chiffrer votre entrée cible, sous Chiffrement, choisissez Personnaliser les paramètres de chiffrement (avancé).

   Si vous choisissez cette option, entrez un ARN de clé KMS existant ou choisissez Créez un AWS KMS keypour accéder à la console AWS KMS. Pour plus d'informations sur la façon dont le planificateur EventBridge chiffre vos données au repos, voir Chiffrement au repos dans le Guide de l'utilisateur du planificateur Amazon EventBridge.

e. Pour que le planificateur EventBridge crée un rôle d'exécution pour vous, choisissez Créer un rôle pour cette planification. Ensuite, saisissez un nom pour Nom du rôle. Si vous choisissez cette option, le planificateur EventBridge associe au rôle les autorisations requises pour votre cible modélisée.

10. Choisissez Next (Suivant).
11. Sur la page Examiner et créer une planification, examinez les détails de votre planification. Dans chaque section, choisissez Modifier pour revenir à cette étape et modifier ses détails.
12. Choisissez Créer une planification.

    Vous pouvez consulter la liste de vos planifications nouvelles et existantes sur la page Planifications. Sous la colonne État, vérifiez que votre nouvelle planification est activée.

Ressources connexes

Pour de plus amples informations sur le planificateur EventBridge, veuillez consulter les ressources suivantes :

- Guide de l'utilisateur du planificateur EventBridge
- Référence de l'API du planificateur EventBridge
- Tarification du planificateur EventBridge

Utilisation d'Amazon SNS pour la messagerie d'application à personne (A2P)

Cette section fournit des informations sur l'utilisation d'Amazon SNS pour les notifications utilisateur avec les abonnés, à travers des applications mobiles, des numéros de téléphone mobile et les adresses électroniques.

Rubriques

Messagerie texte mobile (SMS)

Vous pouvez utiliser pour envoyer des messages texte, ou des messages SMS, à des appareils compatibles SMS. Vous pouvez envoyer un message directement à un numéro de téléphone, ou vous pouvez envoyer un message à plusieurs numéros de téléphone simultanément en abonnant ces numéros de téléphone à une rubrique et en envoyant votre message à la rubrique.

Vous pouvez définir des préférences SMS pour votre compte AWS afin d'adapter vos distributions SMS à vos cas d'utilisation et votre budget. Par exemple, vous pouvez choisir d'optimiser la distribution de vos messages pour réduire les coûts ou améliorer la fiabilité. Vous pouvez aussi spécifier des quotas de dépenses pour les distributions de messages individuels et des quotas de dépenses mensuelles pour votre Compte AWS.

Selon les lois et réglementations locales (par exemple, aux États-Unis et au Canada), les destinataires de SMS peuvent refuser, ce qui signifie qu'ils choisissent d'arrêter de recevoir des SMS provenant de votre Compte AWS. Une fois qu'un destinataire refuse, vous pouvez, avec certaines restrictions, réactiver le numéro de téléphone afin de pouvoir à nouveau lui envoyer des messages.

Amazon SNS prend en charge les SMS dans plusieurs régions et vous pouvez envoyer des messages dans plus de 200 pays et régions. Pour de plus amples informations, veuillez consulter Pays et régions pris en charge.

Rubriques

Environnement de test (sandbox) pour SMS

Lorsque vous commencez à utiliser Amazon SNS pour envoyer des messages SMS, votre compte AWS se trouve dans l'environnement de test (sandbox) pour SMS. L'environnement de test (sandbox) pour SMS offre un environnement sûr pour vous permettre d'essayer les fonctionnalités d'Amazon SNS sans risquer votre réputation d'expéditeur SMS. Même si votre compte est dans l'environnement de test (sandbox) pour SMS, vous pouvez utiliser toutes les fonctionnalités d'Amazon SNS, avec les restrictions suivantes :

- Vous pouvez envoyer des messages SMS uniquement aux numéros de téléphone de destination vérifiés.
- Vous pouvez avoir jusqu'à 10 numéros de téléphone de destination vérifiés.
- Vous ne pouvez supprimer les numéros de téléphone de destination qu'après 24 heures ou plus se sont écoulées depuis la vérification ou la dernière tentative de vérification.

Lorsque votre compte est déplacé de l'environnement de test (sandbox), ces restrictions sont supprimées et vous pouvez envoyer des messages SMS à n'importe quel destinataire.

Rubriques

Ajout et vérification de numéros de téléphone dans l'environnement de test (sandbox) pour SMS

Pour commencer à envoyer des SMS alors que votre AWS compte est dans le sandbox SMS, créez une identité d'origine, ajoutez les numéros de téléphone de destination, puis vérifiez-les.

> **Note**
>
> Comme pour les comptes qui ne se trouvent pas dans l'environnement de test (sandbox) pour SMS, une identité d'origine est requise pour envoyer des messages SMS à des destinataires situés dans certains pays ou régions. Pour plus d'informations, consultez Pays et régions pris en charge.
> Les ID d'origine incluent les ID d'expéditeur et différents types de numéros d'origine.
> Pour afficher vos numéros d'origine existants, dans le panneau de navigation de la Console Amazon SNS, choisissez Numéros d'origine. Actuellement, les ID de l'expéditeur n'apparaissent pas dans cette liste.

Pour ajouter et vérifier des numéros de téléphone de destination

1. Connectez-vous à la console Amazon SNS.
2. Créez une identité d'origine pour le numéro de téléphone.
3. Dans le menu de la console, choisissez une Région AWS qui prend en charge la messagerie SMS.
4. Dans le panneau de navigation, sélectionnez Text messaging (SMS).
5. Sur la page Messagerie texte mobile (SMS), sous Numéros de téléphone de destination de l'environnement de test (sandbox), choisissez Ajouter un numéro de téléphone.
6. Souus Détails de destination, saisissez le code de pays et le numéro de téléphone, spécifiez la langue à utiliser pour le message de vérification, puis choisissez Ajouter un numéro de téléphone.

   Amazon SNS envoie un mot de passe unique (OTP) au numéro de téléphone de destination. Si le numéro de téléphone de destination ne reçoit pas l'OTP dans les 15 minutes, choisissez Renvoyer le code de vérification. Vous pouvez envoyer l'OTP au même numéro de téléphone de destination jusqu'à cinq fois toutes les 24 heures.
7. Dans la case Code de vérification, saisissez l'OTP envoyé au numéro de téléphone de destination, puis choisissez Vérifier le numéro de téléphone.

Le numéro de téléphone de destination et son statut de vérification apparaissent dans la section Numéros de téléphone de destination de l'environnement de test (sandbox). Si le statut de vérification est En suspens, la vérification a échoué. Cela peut se produire, par exemple, si vous n'avez pas saisi le code du pays avec le numéro de téléphone. Vous ne pouvez supprimer les numéros de téléphone de destination en suspens ou vérifiés qu'après 24 heures ou plus écoulées depuis la vérification ou la dernière tentative de vérification.

8. Répétez ces étapes dans chaque région où vous souhaitez utiliser ce numéro de téléphone de destination.

Résolution des problèmes de non-réception d'un texte OTP

Résolvez les problèmes courants susceptibles d'empêcher un numéro de téléphone de recevoir des SMS OTP.

- Limite de dépenses par SMS Amazon SNS : si vous avez Compte AWS dépassé la limite de dépenses pour l'envoi de SMS, d'autres messages, y compris des SMS OTP, risquent de ne pas être envoyés tant que la limite n'est pas augmentée ou que le problème de facturation n'est pas résolu.
- Numéro de téléphone non activé pour les notifications par SMS : dans certains pays ou régions, les destinataires doivent choisir de recevoir des SMS sous forme de codes courts, couramment utilisés pour les textes OTP. Si le numéro de téléphone du destinataire n'est pas activé, il ne recevra pas le texte OTP.
- Restrictions ou filtrage des opérateurs : certains opérateurs de téléphonie mobile peuvent avoir mis en place des restrictions ou des mécanismes de filtrage qui empêchent la livraison de certains types de messages SMS, y compris les textes OTP. Cela peut être dû aux politiques de sécurité ou aux mesures anti-spam mises en œuvre par le transporteur.
- Numéro de téléphone non valide ou incorrect : si le numéro de téléphone fourni par le destinataire est incorrect ou invalide, le texte OTP ne sera pas délivré.
- Problèmes de réseau : des problèmes ou des pannes de réseau temporaires peuvent empêcher l'envoi de messages SMS, y compris les SMS OTP, sur le téléphone du destinataire.
- Retard de livraison : dans certains cas, les messages SMS peuvent connaître des retards de livraison en raison de la congestion du réseau ou d'autres facteurs. Le texte OTP pourrait éventuellement être livré, mais il pourrait être retardé au-delà du délai prévu.

- Suspension ou résiliation du compte : en cas de problème avec votre compte Compte AWS, tel qu'un non-paiement ou une violation des AWS conditions d'utilisation, les fonctionnalités de messagerie Amazon SNS, y compris les SMS OTP, peuvent être suspendues ou résiliées.

Suppression des numéros de téléphone de l'environnement de test (sandbox) pour SMS

Vous pouvez supprimer les numéros de téléphone de destination en suspens ou vérifiés de l'environnement de test (sandbox) pour SMS.

Pour supprimer les numéros de téléphone de destination de l'environnement de test (sandbox) pour SMS

1. Attendez 24 heures après la vérification du numéro de téléphone, ou 24 heures après votre dernière tentative de vérification.
2. Connectez-vous à la console Amazon SNS.
3. Dans le menu de la console, choisissez une région AWS qui prend en charge la messagerie SMS où vous avez ajouté un numéro de téléphone de destination.
4. Dans le panneau de navigation, sélectionnez Text messaging (SMS).
5. Sur la page Messagerie texte mobile (SMS), sous Numéros de téléphone de destination de l'environnement de test (sandbox), choisissez le numéro de téléphone à supprimer, puis choisissez Supprimer le numéro de téléphone.
6. Pour confirmer que vous souhaitez supprimer le numéro de téléphone, saisissez **`delete me`**, puis choisissez Supprimer.

   Si 24 heures ou plus se sont écoulées depuis que vous avez vérifié ou tenté de vérifier le numéro de téléphone de destination, il est supprimé et Amazon SNS met à jour la liste de vos numéros de téléphone de destination.
7. Répétez ces étapes dans chaque région où vous avez ajouté le numéro de téléphone de destination et où vous n'avez plus l'intention de l'utiliser.

Déplacement de l'environnement de test (sandbox) pour SMS

Pour Compte AWS sortir du sandbox SMS, vous devez d'abord ajouter, vérifier et tester les numéros de téléphone de destination. Ensuite, vous devez créer un dossier avec AWS Support.

Pour demander que votre AWS compte soit retiré du sandbox SMS

1. Vérifiez les numéros de téléphone

   a. Lorsque vous Compte AWS êtes dans le sandbox SMS, ouvrez la console Amazon SNS.

   b. Dans le volet de navigation, sous Mobile, choisissez Messagerie texte (SMS).

   c. Dans la section Numéros de téléphone de destination du Sandbox, ajoutez et vérifiez un ou plusieurs numéros de téléphone de destination. Cette vérification garantit que vous pouvez envoyer et recevoir des messages avec succès.

2. Tester la publication de SMS

   - Vérifiez que vous êtes en mesure d'envoyer et de recevoir des messages à au moins un numéro de téléphone vérifié. Pour des instructions plus détaillées sur la façon de publier des SMS, consultezPublication sur un téléphone mobile.

3. Lancer la modification du bac à sable

   - Sur la page Messagerie texte mobile (SMS) de la console Amazon SNS, sous Informations de compte, choisissez Quitter l'environnement de test (sandbox) pour SMS. Cette action vous redirige vers le centre de support Amazon et crée automatiquement un dossier d'assistance avec l'option d'augmentation du quota de service sélectionnée.

4. Remplissez le formulaire

   - Dans le formulaire d'assistance, sous Augmentation du quota de service, procédez comme suit :

     i. Choisissez Choisir la messagerie texte SNS comme service.

     ii. Indiquez l'URL du site Web ou le nom de l'application à partir duquel vous souhaitez envoyer des SMS.

     iii. Spécifiez le type de message que vous allez envoyer : mot de passe à usage unique, promotionnel ou transactionnel.

     iv. Choisissez celui Région AWSà partir duquel vous allez envoyer des SMS.

     v. Répertoriez les pays ou régions dans lesquels vous prévoyez d'envoyer des SMS.

     vi. Décrivez comment vos clients acceptent de recevoir des messages.

     vii. Incluez tous les modèles de message que vous avez l'intention d'utiliser.

5. Spécifiez le quota et la région

- Sous Demandes, procédez comme suit :

  i. Choisissez l'Région AWSendroit où vous souhaitez déplacer votre Compte AWS.

  ii. Choisissez Limites générales pour le type de ressource.

  iii. Choisissez Exit SMS Sandbox for Quota.

  iv. (Facultatif) Pour demander des augmentations supplémentaires ou d'autres ajustements, choisissez Ajouter une autre demande et spécifiez les détails nécessaires.

  v. Pour Nouvelle valeur de quota, entrez la limite en dollars américains que vous demandez.

6. Détails supplémentaires

   a. Dans la description du dossier, fournissez tous les détails supplémentaires relatifs à votre demande.

   b. Sous Options de contact, choisissez la langue de contact de votre choix.

7. Soumettre la demande

   - Choisissez Soumettre pour envoyer votre demande à AWS Support.

L' AWS Support équipe fournit une première réponse à votre demande dans les 24 heures.

Pour empêcher que nos systèmes soient utilisés pour envoyer des contenus indésirables ou malveillants, chaque demande est traitée avec soin. Si cela est possible, nous accepterons votre demande dans ce délai de 24 heures. En revanche, si nous avons besoin de plus amples informations, le traitement de votre demande peut prendre plus de temps.

Nous pourrions ne pas être en mesure de donner suite à votre demande si votre cas d'utilisation n'est pas conforme à nos politiques.

Identités d’origine des messages SMS

Lorsque vous envoyez des messages SMS à l'aide d'Amazon SNS, vous pouvez vous identifier auprès de vos destinataires à l'aide des types suivants d'Identités d'origine :

- ID d'expéditeur
- Numéros d'origine

> **Note**
>
> La messagerie SMS Amazon SNS est disponible dans les régions où Amazon Pinpoint n'est pas actuellement pris en charge. Si vous opérez en Europe (Stockholm), au Moyen-Orient (Bahreïn), en Europe (Paris), en Amérique du Sud (São Paulo) ou aux USA Ouest (Californie du Nord), ouvrez la console Amazon Pinpoint dans la région USA Est (Virginie du Nord) pour inscrire votre entreprise et votre campagne 10DLC, mais ne demandez pas de numéro 10DLC. Utilisez plutôt la AWS console Service Quotas pour créer un cas visant à augmenter la limite de service tout en demandant le numéro 10DLC de cette région. Pour en savoir plus sur la procédure de demande d'identités d'origine, consultez Demande de prise en charge de la messagerie SMS avec Amazon SNS.

ID d'expéditeur

Un ID d'expéditeur est un nom alphabétique est un nom qui identifie l'expéditeur d'un message SMS. Lorsque vous envoyez un message SMS à l'aide d'un ID d'expéditeur et que le destinataire se trouve dans une zone où l'authentification de l'ID d'expéditeur est prise en charge, votre ID d'expéditeur s'affiche sur l'appareil du destinataire à la place d'un numéro de téléphone. Un ID d'expéditeur fournit aux destinataires de SMS plus d'informations sur l'expéditeur qu'un numéro de téléphone, un code long ou un code court.

Les ID d'expéditeur sont pris en charge dans plusieurs pays et régions à travers le monde. Dans certains cas, si vous êtes une entreprise qui envoie des messages SMS aux clients individuels, vous devez utiliser un ID d'expéditeur préalablement enregistré auprès d'un organisme de réglementation ou d'un groupe industriel. Pour obtenir la liste complète des pays et régions qui prennent en charge ou nécessitent des ID d'expéditeur, consultez Pays et régions pris en charge.

L'utilisation de l'ID d'expéditeur n'entraîne aucun coût supplémentaire. Toutefois, la prise en charge et les exigences en matière d'authentification d'ID d'expéditeur varient. Plusieurs marchés majeurs (y compris le Canada, la Chine et les États-Unis) ne prennent pas en charge les ID d'expéditeur. Certaines zones exigent que les entreprises qui envoient des messages SMS aux clients individuels doivent utiliser un ID d'expéditeur préalablement enregistré auprès d'un organisme de réglementation ou d'un groupe industriel.

> **Important**
>
> AWS interdit l'usurpation de SMS, dans le cadre de laquelle l'ID d'expéditeur est utilisé pour emprunter l'identité d'une autre personne, d'une autre société ou d'un autre produit. Utilisez uniquement un ID d'expéditeur qui représente une marque ou une marque commerciale que vous possédez.

Avantages

Un ID d'expéditeur fournir au destinataire plus d'informations sur l'expéditeur du message. Il est plus facile d'établir votre identité de marque à l'aide d'un ID d'expéditeur qu'à l'aide d'un code long ou court. L'utilisation d'un ID d'expéditeur n'entraîne aucun coût supplémentaire.

Inconvénients

La prise en charge et les exigences en matière d'authentification d'ID d'expéditeur ne sont pas identiques dans tous les pays ou régions. Plusieurs marchés majeurs (y compris le Canada, la Chine et les États-Unis) ne prennent pas en charge l'identifiant d'expéditeur. Dans certains zones, vous devez avoir votre ID d'expéditeur préalablement approuvé par un organisme de réglementation avant de pouvoir l'utiliser.

Enregistrement de l'ID d'expéditeur par pays

Vous devez ouvrir un dossier auprès de l'assistance AWS afin d'enregistrer les identifiants d'expéditeur pour les messages SMS. Après avoir fait votre demande d'assistance, AWS partagera les documents supplémentaires requis. Vous devez également fournir les informations suivantes pour le pays approprié dans lequel vous enregistrez l'identifiant de l'expéditeur.

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Australie (AU)	Transactionnel et promotionnel	• Alphanumérique • 11 caractères maximum • Aucun espace	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisa

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
		• Aucun caractère spécial • L'ID d'expéditeur doit être le nom de marque de la société qui envoie le SMS	teur appellera l'API/ le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom de la société et l'ID d'expéditeur • Numéro de licence commerciale officiel de la société ou numéro de TVA

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement

			• Modèles de messages que vous prévoyez d'envoyer
			• Une licence d'immatriculation d'entreprise. Des exemples incluent, sans s'y limiter :
			  • Numéro d'entreprise australien (ABN)
			  • Numéro de société australien (ACN)
			  • Numéro d'enregistrement australien (ARBN)
			  • Numéro d'entreprise autochtone (ICN)
			• Lettre d'autorisation (LOA)

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Biélorussie (BY)	Messages transactionnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial • L'ID d'expéditeur doit être le nom de marque de la société qui envoie le SMS	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Numéro de licence commerciale officiel de la société ou numéro de TVA • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Chine (CN) La Chine n'exige pas l'enregistrement de l'ID d'expéditeur, mais exige l'enregistrement du contenu/du modèle avec une signature corporelle préattachée (par exemple, [Amazon]).	Les mots-clés suivants ne sont pas autorisés : • Falung Gong • SB • Place Tian'anmen Messages des catégories suivantes : • Cartes bancaires • Paiements numériques (y compris les cryptomonnaies) • URL de trafic frauduleuses (hameçonnage ou spam) • Jeu • Contenu inapproprié (pour adultes, violence, drogues, alcool) • Emprunts • Chirurgie plastique • Politique • Religion • Négociation d'actions	• 11 caractères maximum • Aucun espace • Aucun caractère spécial	• Nom de la société • Adresse de la société • État/province • Pays • Numéro de téléphone de la société • Site Web de la société • Volume mensuel estimé • Type de message : promotionnel/transactionnel • Explication du cas d'utilisation • Les modèles que vous souhaitez enregistrer (les SMS envoyés ne doivent pas différer des modèles fournis pour garantir la conformité et une livraison réussie). Par exemple, [CompanyName] Votre mot de passe à usage unique est {OTP}. Ce

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
	• Monnaie virtuelle Les signatures entre crochets doivent être ajoutées au corps du message SMS. Pour une livraison réussie en Chine, vous devez enregistrer une signature de message avec votre modèle de contenu de message. Cette signature de message doit être ajoutée au contenu du message sur chaque SMS envoyé. Le fait de ne pas ajouter une signature enregistrée au corps du message SMS peut entraîner le blocage ou le filtrage des SMS. Les signatures doivent être ajoutées au corps du SMS entre crochets.		code expirera dans 10 minutes. • Confirmation que vous n'enverrez pas de contenu promotionnel sous forme de message transactionnel • Signature du message. Consultez Restrictions et exigences relatives au format de la signature.

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Égypte (EG)	Messages transacti onnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisa teur appellera l'API/ le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Votre société possède-t-elle une entité commercia le/un bureau en Égypte ? Ou s'agit-il d'une société non basée en Égypte qui envoie vers l'Égypte ? • Confirmation écrite que le cas d'utilisa tion couvre tous les messages à envoyer par cet ID d'expéditeur • Si ce n'est pas évident, fournissez le lien entre le nom de la société et l'ID d'expéditeur • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Inde (IN)	Messages transactionnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	Consultez Exigences relatives aux ID d'expéditeur pour l'Inde.

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Jordanie (JO)	Messages transacti onnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisa teur appellera l'API/ le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Certificat d'enregistrement de la société • Le type de message que vous prévoyez d'envoyer (par exemple, OTP, alertes) • Confirmation écrite que le cas d'utilisation décrit tous les messages à envoyer par cet ID d'expéditeur • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Koweït (KW)	Messages transactionnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Le type de message que vous prévoyez d'envoyer (par exemple, OTP, alertes) • Confirmation écrite que le cas d'utilisation décrit tous les messages à envoyer par cet ID d'expéditeur • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Philippines (PH)	Messages transacti onnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisa teur appellera l'API/ le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Le type de message que vous prévoyez d'envoyer (par exemple, OTP, alertes) • Confirmation écrite que le cas d'utilisation décrit tous les messages à envoyer par cet ID d'expéditeur • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Qatar (QA)	Messages transactionnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Type de SMS envoyé • (Transactionnel/promotionnel/OTP) Notez que seuls les messages transactionnels/OTP peuvent être utilisés avec des ID d'expéditeur destinés au Qatar par souci de conformité. • Confirmation écrite que le cas d'utilisation décrit tous les messages à envoyer par cet ID d'expéditeur • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Russie (RU)	Messages transacti onnels uniquement	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisa teur appellera l'API/ le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Numéro d'identif ication fiscale ou numéro de licence • Certificat d'enregis trement de la société • Adresse électroni que du contact

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			• Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom de la société et l'ID d'expéditeur • Confirmation que ce cas d'utilisation s'appliquera à tous les messages envoyés en Russie depuis ce compte • Confirmation que les messages non transactionnels doivent être envoyés à l'aide d'un ID d'expéditeur distinct • 272 $/mois, veuillez confirmer que vous approuvez ces frais mensuels récurrents : Oui/Non

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			• Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Arabie saoudite (SA)	Chaque ID d'expéditeur doit être transactionnel ou promotionnel. Il n'est pas possible d'utiliser un seul ID d'expéditeur pour les deux types de trafic. Si vous envoyez du trafic OTP ou 2FA, l'ID d'expéditeur doit être utilisé uniquement à cette fin. Les ID d'expéditeur promotionnels seront soumis à la liste « Ne pas déranger » (DND) de l'Arabie saoudite.	• Longueur de l'ID d'expéditeur promotionnel : 2 à 8 caractères, ID d'expéditeur précédé de « -AD » • Longueur de l'ID d'expéditeur transactionnel : 2 à 11 caractères • L'ID d'expéditeur doit représenter l'identité de marque de l'expéditeur • Incluez au moins une lettre • N'utilisez pas de caractères spéciaux ASCII (par exemple, #, @) • Vous pouvez inclure des lettres majuscules et minuscules, ainsi que des chiffres de 0 à 9	La prise en charge de l'enregistrement de l'ID d'expéditeur en Arabie Saoudite est réservée aux sociétés internationales uniquement. Nous n'aidons pas actuellement les entreprises saoudiennes locales à enregistrer des ID d'expéditeur • L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom de la société et l'ID d'expéditeur • Modèles de messages que vous prévoyez d'envoyer. • Cet ID d'expédit eur sera-t-il utilisé pour du contenu transactionnel ou promotionnel ? • Confirmation que les messages non transacti onnels doivent être envoyés à l'aide d'un ID d'expéditeur distinct • Si vous envoyez du trafic 2FA ou

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			OTP, confirmation que l'ID d'expédit eur doit être utilisé UNIQUEMENT à cette fin
Singapour (SG)	Messages transacti onnels uniquement	• 11 caractères maximum • Aucun espace • Aucun caractère spécial	Consultez Exigences relatives aux ID d'expéditeur pour Singapour.

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Sri Lanka (LK)	Aucune restriction ni exigence particulière	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Type de société (locale/internationale) • URL de la société (lien vers votre application ou le site Web de votre société) • Explication du cas d'utilisation et de l'objectif des messages • Modèles de messages que vous prévoyez d'envoyer • Cet ID d'expéditeur sera-t-il utilisé pour du contenu transactionnel ou promotionnel ? • Confirmation que les messages non transacti

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			onnels doivent être envoyés à l'aide d'un ID d'expéditeur distinct • Si vous envoyez du trafic 2FA ou OTP, confirmation que l'ID d'expédit eur doit être utilisé uniquement à cette fin

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Thaïlande (TH)	Aucune restriction ni exigence particulière	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Type de SMS envoyé (transact ionnel/promotionne l/OTP) • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Turquie (TR)	Aucune restriction ni exigence particulière	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • URL de la société (lien vers votre application ou le site Web de votre société) • Si votre société est locale ou internationale en Turquie • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Type de SMS envoyé (transact ionnel/promotionne l/OTP) • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Ukraine (UA)	Aucune restriction ni exigence particulière	• Alphanumérique • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • URL de la société (lien vers votre application ou le site Web de votre société) • Numéro de TVA • Société locale ou internationale • Volume mensuel estimé • Explication du cas d'utilisation et de l'objectif des messages • Si ce n'est pas évident, expliquez le lien entre le nom

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de la société et l'ID d'expéditeur • Type de SMS envoyé (transact ionnel/promotionne l/OTP) • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Émirats arabes unis (AE)	Messages transacti onnels uniquement	• Alphanumérique • Les ID d'expédit eur génériques ne sont pas autorisés et doivent identifier la marque • 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisa teur appellera l'API/ le service • Nom de la société • Adresse de la société (y compris la ville, l'état/la province, le code postal) • Pays de la société • Numéro de téléphone de la société • URL de la société (lien vers votre application ou le site Web de votre société) • Volume mensuel estimé • Si ce n'est pas évident, expliquez le lien entre le nom de la société et l'ID d'expéditeur • Explication du cas d'utilisation et

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
			de l'objectif des messages • Numéro de licence commerciale officiel de la société ou numéro de TVA • Confirmation écrite que tout le trafic envoyé par cet ID d'expéditeur est décrit par le cas d'utilisation donné • Modèles de messages que vous prévoyez d'envoyer

Nom du pays	Type de message	Restrictions et exigences relatives au format	Exigences en matière d'enregistrement
Vietnam (VN)	Messages transactionnels uniquement. Les messages marketing et promotionnels ne sont pas autorisés. Le contenu interdit inclut : • Contenu pour adultes • Services caritatifs • Cryptomonnaie • Loterie • Jeux d'argent/ casinos mobiles • Paris sportifs • Votes	• 11 caractères maximum • Aucun espace • Aucun caractère spécial	• L'ID d'expéditeur que vous souhaitez enregistrer • À partir de quelle région AWS l'utilisateur appellera l'API/ le service • Volume mensuel estimé • Si ce n'est pas évident, expliquez le lien entre le nom de la société et l'ID d'expéditeur • Explication du cas d'utilisation et de l'objectif des messages • Confirmation écrite que tout le trafic envoyé par cet ID d'expéditeur est décrit par le cas d'utilisation donné

Restrictions et exigences relatives au format de la signature

Pour une livraison réussie en Chine, vous devez enregistrer une signature de message avec votre modèle de contenu de message. Cette signature de message doit être ajoutée au contenu du message sur chaque SMS envoyé. Le fait de ne pas ajouter une signature enregistrée au corps du message SMS peut entraîner le blocage ou le filtrage des SMS.

- La signature doit être ajoutée au corps du SMS entre crochets
- Le texte standard doit figurer entre crochets
- Le texte Unicode doit utiliser des crochets lenticulaires pour contenir la signature : U+3010 CROCHET LENTICULAIRE GAUCHE et U+3011 CROCHET LENTICULAIRE DROIT. Exemple : 【Notice】
- Doit comporter entre 3 et 11 caractères
- Les caractères chinois/anglais sont pris en charge

Exigences relatives à l'identification de l'expéditeur pour la France

Ce guide fournit les étapes et les directives nécessaires pour créer un identifiant d'expéditeur dédié requis par les opérateurs de téléphonie mobile français pour envoyer des SMS en France.

Rubriques

- Configuration d'un identifiant d'expéditeur dédié pour la France
- Directives d'attribution de nom pour l'identifiant de l'expéditeur

Configuration d'un identifiant d'expéditeur dédié pour la France

Vous pouvez utiliser l'une des méthodes suivantes pour configurer un identifiant d'expéditeur dédié. Amazon SNS utilisera l'identifiant de l'expéditeur en votre nom pour les SMS publiés à l'aide de l'API `Publish`.

- Vous pouvez utiliser la console Amazon SNS pour configurer l'identifiant de l'expéditeur par défaut à utiliser pour tous les SMS publiés. Pour en savoir plus, consultez Configuration des préférences de messagerie SMS à l'aide du AWS Management Console.
- Vous pouvez utiliser l'API `Publish` pour définir l'identifiant de l'expéditeur à l'aide de l'attribut de message `AWS.SNS.SMS.SenderID` lorsque vous demandez à Amazon SNS de publier un message SMS. Pour en savoir plus, consultez Envoi d'un message (console).

Directives d'attribution de nom pour l'identifiant de l'expéditeur

- Le nom de l'identifiant de l'expéditeur doit être alphanumérique et doit comporter un maximum de 11 caractères.
- Le nom de l'identifiant de l'expéditeur ne doit pas contenir de caractères spéciaux ni d'espaces.

- Nous vous recommandons d'utiliser le même nom pour l'identifiant de l'expéditeur et le nom de marque de l'entreprise qui envoie le SMS.

Exigences relatives aux ID d'expéditeur pour l'Inde

Par défaut, lorsque vous envoyez des messages à des destinataires en Inde, Amazon SNS utilise les connexions ILDO (International Long Distance Operator) pour transmettre ces messages. Lorsque les destinataires voient un message envoyé via une connexion ILDO, celui-ci apparaît comme ayant été envoyé à partir d'un ID numérique aléatoire (sauf si vous achetez un code court dédié).

> **Note**
>
> Le prix pour l'envoi de messages via des routes locales est affiché sur la page de tarification SMS mondiale d'Amazon SNS. Le prix pour l'envoi de messages via des connexions ILDO est supérieur au prix pour l'envoi de messages via des routes locales.

Si vous préférez utiliser un ID d'expéditeur alphabétique pour vos messages SMS, vous devez envoyer ces messages via des routes locales plutôt que des routes ILDO. Pour envoyer des messages via des acheminements locaux, vous devez d'abord enregistrer votre cas d'utilisation et vos modèles de message auprès de l'Autorité de régulation des télécommunications de l'Inde (TRAI) via les portails de Technologie des registres distribués (DLT). Ces exigences d'enregistrement sont conçues de manière à réduire le nombre de messages non sollicités que les consommateurs indiens reçoivent et à protéger les consommateurs contre les messages potentiellement nuisibles. Ce processus d'enregistrement est géré par Vodafone India via son service Vilpower.

Rubriques

- Étape 1 : enregistrement auprès de la TRAI
- Étape 2 : demande d'un ID d'expéditeur
- Étape 3 : envoi de messages SMS
- Dépannage des messages SMS envoyés à des destinataires en Inde

Étape 1 : enregistrement auprès de la TRAI

Avant de pouvoir envoyer des messages SMS à des destinataires en Inde, vous devez enregistrer votre organisation auprès de la Telecom Regulatory Authority of India (TRAI). Pendant le processus d'enregistrement, soyez prêt à fournir les informations suivantes :

- Le numéro de compte permanent (PAN) de votre organisation.
- Le numéro de compte de déduction fiscale (TAN) de votre organisation.
- Le numéro d'identification de la taxe sur les produits et services (GSTIN) de votre organisation.
- Le numéro d'identité d'entreprise (CIN) de votre organisation.
- Une lettre d'autorisation qui vous donne le pouvoir d'enregistrer votre organisation.

Voici un exemple de liste de quelques sites d'enregistrement DLT (Distributed Ledger Technology) que vous pouvez utiliser pour enregistrer votre organisation auprès de la TRAI (des frais peuvent s'appliquer). Le processus d'enregistrement varie selon le site. Contactez les équipes de support respectives pour obtenir de l'aide.

- BSNL DLT – Enregistrement gratuit.
- Jio Trueconnect – Facture des frais pour finaliser le processus d'enregistrement.
- Solutions d'une entreprise intelligente – Facture des frais pour finaliser le processus d'enregistrement.
- Vilpower - Inclut un modèle que vous pouvez télécharger et modifier en fonction de vos besoins. Vilpower facture des frais pour finaliser le processus d'enregistrement.

Pour enregistrer votre organisation auprès de l'Autorité de régulation des télécommunications de l'Inde (TRAI)

Les informations suivantes expliquent comment enregistrer votre organisation auprès de la TRAI à l'aide de Vilpower.

1. Dans un navigateur web, accédez au site web de Vilpower à l'adresse https://www.vilpower.in.
2. Choisissez Signup (Inscription) pour créer un autre compte. Pendant le processus d'enregistrement, procédez comme suit :
   - Pour le type d'entité suivant lequel s'enregistrer, choisissez En tant qu'entreprise.
   - Pour le Nom du télévendeur, utilisez Infobip Private Limited - ALL. À l'invite, commencez à taper **`Infobip`** puis choisissez Infobip Private Limited – ALL dans la liste déroulante.
   - Pour Enter Telemarketer ID (Entrer l'ID de télévendeur), indiquez **`110200001152`**.
   - Lorsque vous êtes invité à fournir vos ID d'en-tête, entrez les ID d'expéditeur que vous souhaitez enregistrer.

> **Note**
>
> L'Inde exige que les ID d'expéditeur comportent précisément six caractères.

- Lorsque vous êtes invité à fournir vos modèles de contenu, entrez le contenu du message que vous prévoyez d'envoyer à vos destinataires. Incluez un modèle pour chaque message que vous prévoyez d'envoyer.

> **Note**
>
> Les sites web des fournisseurs d'enregistrement DLT ne sont pas gérés par Amazon Web Services. Les étapes présentées de leurs sites web sont susceptibles d'être modifiées.

Étape 2 : demande d'un ID d'expéditeur

Pour demander un ID d'expéditeur en Inde, vous devez déposer une demande AWS Support. Suivez les étapes indiquées à Demande d'ID expéditeur. Dans votre demande, fournissez les informations requies suivantes :

- La région AWS à partir de laquelle l'expéditeur envisage d'envoyer des messages SMS.
- Nom de l'entreprise utilisé lors du processus d'enregistrement de DLT.
- L'ID de l'entité principale (PEID) que vous avez reçu après l'enregistrement réussi de l'entité DLT.
- Volume mensuel estimé.
- Une explication de votre cas d'utilisation.
- Une description des flux d'abonnement de l'utilisateur final.
- Confirmation que les inclusions de l'utilisateur final sont collectés et enregistrés.

Étape 3 : envoi de messages SMS

Après avoir enregistré votre organisation auprès de la TRAI, vous pouvez envoyer des messages SMS à des destinataires en Inde.

1. Connectez-vous à la console Amazon SNS.

2. Dans le menu de la console, définissez le sélecteur de région sur une région prenant en charge la messagerie SMS.
3. Dans le panneau de navigation, choisissez Messages texte (SMS).
4. Sur la page Message texte mobile (SMS), choisissez Publier un message texte. La fenêtre Publier un message SMS s'ouvre.
5. Dans le champ Message type, choisissez l'une des valeurs suivantes :

   - Promotional – Messages non stratégiques, tels que les messages marketing.

     Lorsque vous utilisez des ID d'expéditeur numériques, choisissez cette option.
   - Transactional – Messages stratégiques qui prennent en charge les transactions clients, comme des codes secrets uniques pour l'authentification multifacteur.

     Lorsque vous utilisez des ID d'expéditeur alphabétiques ou alphanumériques, choisissez cette option.

   Ce paramètre au niveau du message remplace le type de message par défaut, que vous définissez sur la page Préférences de SMS.

   Pour la tarification des messages promotionnels et transactionnels, consultez la page Tarifs SMS internationaux.
6. Dans le champ Numéro, saisissez le numéro de téléphone auquel vous souhaitez envoyer le message.
7. Pour Message, saisissez le message à envoyer.

   Lorsque vous ajoutez du contenu à des messages SMS, assurez-vous qu'il correspond exactement au contenu du modèle enregistré DLT. Les opérateurs bloquent les messages SMS si le contenu de leur message inclut des retours de caractères supplémentaires, des espaces, des ponctuations ou des casses de phrase qui ne correspondent pas. Les variables d'un modèle peuvent comporter 30 caractères ou moins.
8. Dans la section Identités d'origine, pour l'ID de l'expéditeur, saisissez un ID personnalisé contenant entre 3 et 11 caractères.

   Les ID d'expéditeur peuvent être numériques pour les messages promotionnels, ou alphabétiques ou alphanumériques pour les messages transactionnels. L'ID expéditeur s'affiche en tant qu'expéditeur du message sur l'appareil de réception.

Pour les ID d'expéditeur promotionnels numériques enregistrés pour l'Inde, spécifiez l'ID de l'expéditeur en tant que paramètre du Numéro d'origine dans la demande d'envoi de SMS.

9. Développez la section Attributs spécifiques au pays et spécifiez les attributs requis suivants pour l'envoi de messages SMS à des destinataires en Inde :

   - ID de l'entité – L'ID de l'entité ou l'ID de l'entité principale (PE) que vous avez reçu de l'organisme de réglementation pour l'envoi de messages SMS à des destinataires en Inde.

     Il s'agit d'une chaîne personnalisée de 1 à 50 caractères fournie par la TRAI qui identifie de manière unique l'entité que vous avez enregistrée auprès de la TRAI.

   - ID de modèle – L'ID de modèle que vous avez reçu de l'organisme de réglementation pour l'envoi de messages SMS à des destinataires en Inde.

     Il s'agit d'une chaîne personnalisée de 1 à 50 caractères fournie par la TRAI qui identifie de manière unique le modèle que vous avez enregistré auprès de la TRAI. L'ID du modèle doit être associé à l'ID de l'expéditeur que vous avez spécifié à l'étape précédente, ainsi qu'au contenu du message.

10. Choisissez Publier le message.

Pour plus d'informations sur l'envoi de messages SMS à des destinataires situés dans d'autres pays, consultez Publication sur un téléphone mobile.

Dépannage des messages SMS envoyés à des destinataires en Inde

Voici quelques raisons pour lesquelles les opérateurs peuvent bloquer les messages SMS :

- Aucun modèle correspondant au contenu envoyé n'a été trouvé.

  Contenu envoyé :**<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  Modèle correspondant : Aucun

  Problème : Il n'y a pas de modèles DLT qui incluent <#> ou `{#var#}` au début du modèle enregistré DLT.

- La valeur d'une variable dépasse 30 caractères.

Contenu envoyé : **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

Modèle correspondant : **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

Problème : La valeur de « ABC Company - India Private Limited » dans le contenu envoyé dépasse une seule limite de caractères `{#var#}` de 30.

- La casse de phrase de message ne correspond pas à la casse de phrase dans le modèle.

  Contenu envoyé : **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  Modèle de correspondance :**{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.**

  Problème : le nom de la société ajouté au modèle de correspondance DLT est majuscule tandis que le contenu envoyé a changé des parties du nom en minuscules — « ABC Pvt. Ltd. » vs. « ABC PVT. LTD. »

Exigences relatives aux ID d'expéditeur pour Singapour

Les clients Amazon SNS peuvent envoyer du trafic SMS à Singapour à l'aide d'un ID d'expéditeur enregistré via le registre SSIR (SMS Sender ID Registry) de Singapour. Le registre SSIR a été lancé en mars 2022 par le centre SGNIC (Singapore Network Information Centre), propriété du conseil IMDA (Info-Communications Media Development Authority) de Singapour, qui permet aux organisations d'enregistrer leur ID d'expéditeur lorsqu'elles envoient des SMS à des téléphones mobiles à Singapour.

Pour utiliser un ID d'expéditeur pour Singapour enregistré, vous devez obtenir un numéro d'entité unique (UEN), puis soumettre une demande à Amazon pour inscrire votre compte sur une liste verte qui vous permettra d'utiliser votre ID d'expéditeur et terminer le processus d'enregistrement auprès du registre SSIR.

Si vous n'enregistrez pas votre ID au plus tard le 30/01/2023, l'ID des messages envoyés à l'aide d'un ID d'expéditeur deviendra LIKELY-SCAM, conformément aux règles des organismes de réglementation. Après cette date, les régulateurs continueront à filtrer ou à bloquer le trafic non enregistré à leur discrétion.

Important

Si vous demandez l'ID d'expéditeur dans les régions Amazon Pinpoint, utilisez la console Amazon Pinpoint pour l'enregistrer. Pour terminer le processus d'enregistrement manuellement pour les régions autres que les régions Amazon Pinpoint, utilisez l'enregistrement d'ID d'expéditeur pour Singapour.
Pour pouvoir continuer à envoyer des messages à Singapour, votre enregistrement doit être terminé au plus tard le 30/01/2023.
Il est très important que vous suiviez les étapes d'enregistrement dans l'ordre suivant. Si vous réalisez ces étapes dans un autre ordre, votre ID d'expéditeur peut être bloqué par le service ou ne pas être conservé sur l'appareil mobile.
Étape 1. Enregistrement pour obtenir un numéro d'entité unique (UEN) pour Singapour
Étape 2. Si vous demandez l'ID d'expéditeur dans les régions Amazon Pinpoint, utilisez les instructions d'enregistrement de l'ID d'expéditeur Amazon Pinpoint pour l'enregistrer.

- Pour enregistrer un ID d'expéditeur lorsque le compte ne se trouve pas dans une région Amazon Pinpoint, suivez les instructions d'enregistrement de l'ID d'expéditeur pour Singapour pour enregistrer manuellement l'ID d'expéditeur.
- Lorsque vous envoyez des SMS pour le compte d'une autre société, une lettre d'autorisation (LOA) de la société est requise.
- N'attendez pas d'approbation ni de changement de statut après avoir soumis l'enregistrement de votre ID d'expéditeur AWS. Passez immédiatement à l'étape 3.

Étape 3. Enregistrement d'un ID d'expéditeur auprès du centre SGNIC (Singapore Network Information Centre)

Rubriques

- Modification de l'enregistrement d'un ID d'expéditeur pour Singapour
- Suppression de l'enregistrement d'un ID d'expéditeur pour Singapour
- Problèmes d'enregistrement pour Singapour
- Questions fréquentes sur l'enregistrement d'ID d'expéditeur pour Singapour

Enregistrement pour obtenir un numéro d'entité unique (UEN) pour Singapour

Pour commencer un enregistrement auprès du registre SSIR, vous devez d'abord obtenir un numéro d'entité unique (UEN) pour Singapour. Vous recevez un numéro d'entité unique (UEN) lorsque vous enregistrez votre entreprise auprès de l'Account and Corporate Registry Authority (ACRA : Autorité de réglementation de la comptabilité et des entreprises). Pour plus d'informations, consultez Who Must Register with ACRA? (langue française non garantie). Le temps de traitement peut varier en fonction de la facilité de validation de votre demande par l'ACRA.

Enregistrement de votre ID d'expéditeur pour Singapour avec Amazon Pinpoint

Une fois que vous avez enregistré votre numéro d'entité unique (UEN) pour Singapour, vous pouvez terminer le processus d'enregistrement de l'ID d'expéditeur dans la console Amazon Pinpoint (uniquement pour les régions Amazon Pinpoint). Lorsque vous enregistrez votre ID d'expéditeur, assurez-vous que les informations sont complètes et exactes. Dans le cas contraire, votre enregistrement pourrait être refusé.

> **Important**
>
> Les informations que vous soumettez via la console Amazon Pinpoint seront transmises à nos partenaires opérateurs pour finaliser l'enregistrement.

Pour enregistrer un ID d'expéditeur pour Singapour

Suivez ces étapes pour enregistrer un ID d'expéditeur lorsque le compte se trouve dans une région Amazon Pinpoint. Si votre compte ne se trouve pas dans une région Amazon Pinpoint, consultez Enregistrement manuel pour terminer le processus d'enregistrement d'ID d'expéditeur pour Singapour.

1. Connectez-vous à la console de gestion AWS et ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.

2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet Sender ID registrations (Enregistrements d'ID d'expéditeur), choisissez Create registration (Créer un enregistrement).
4. Choisissez Singapour comme pays de destination.
5. Dans la section Company Information (Informations sur l'entreprise), saisissez les informations suivantes :
   - Dans le champ Company Name (Nom de l'entreprise), saisissez le nom de votre entreprise exactement tel qu'il apparaît sur votre enregistrement UEN.
   - Pour Tax ID (Numéro d'identification fiscale), saisissez le numéro UEN que vous avez reçu de l'ACRA.
   - Pour Company Website (Site web de l’entreprise), saisissez l’URL du site web de votre entreprise.
   - Pour Address 1 (Adresse 1), saisissez l'adresse postale du siège social de votre entreprise.
   - Le champ Address 2 (Adresse 2) est facultatif. Vous pouvez y saisir le numéro de local du siège social de votre entreprise.
   - Pour City (Ville), saisissez la ville du siège social de votre entreprise.
   - Pour State (État), saisissez l'état du siège social de votre entreprise.
   - Pour Zip Code (Code postal), saisissez le code postal du siège social de votre entreprise.
   - Pour Country (Pays), saisissez le code pays ISO à deux chiffres.
6. Dans la section Contact Information (Informations de contact), saisissez les informations suivantes :
   - Pour First Name (Prénom), saisissez le prénom de la personne qui sera le point de contact de votre entreprise.
   - Pour Last Name (Nom), saisissez le nom de la personne qui sera le point de contact de votre entreprise.
   - Pour Support Email (E-mail de support), saisissez l'adresse e-mail de la personne qui sera le point de contact de votre entreprise.
   - Pour Support Phone Number (Numéro de téléphone de support), saisissez le numéro de téléphone de la personne qui sera le point de contact de votre entreprise.
7. Dans la section Sender ID Information (Informations sur l'ID d'expéditeur), entrez les informations suivantes :

- Pour Sender ID (ID d'expéditeur), saisissez l'identifiant de l'expéditeur que vous souhaitez afficher pour vos messages.
- Sélectionnez True (Vrai) pour Registering on behalf of another brand/entity? (Enregistrement au nom d'une autre marque/entité ?) si c'est le cas. Si vous n'êtes pas l'utilisateur final qui envoie les messages, vous êtes considéré comme un « représentant » de l'autre marque/entité.
- Dans Letter of authorization image – optional (Image de la lettre d'autorisation - facultatif), téléchargez une image de la lettre d'autorisation (LOA) complète si vous avez coché la case Registering on behalf of another brand/entity? (Enregistrement au nom d'une autre marque/entité ?). Le type de fichier pris en charge est PNG et la taille maximale est de 400 Ko. À titre indicatif, vous pouvez télécharger un modèle de LOA ici.
- Pour Sender ID connection – optional (Connexion ID d'expédieteur - facultatif), vous pouvez ajouter des informations sur la connexion entre le SenderID demandé et le nom de l'entreprise.

8. Dans Messaging Use Case (Cas d'utilisation de messagerie), procédez comme suit :
   - Pour Monthly SMS Volume (Volume de SMS mensuel), sélectionnez le nombre de SMS qui seront envoyés chaque mois.
   - Pour Use Case Category (Catégorie de cas d'utilisation), sélectionnez l'un des types de cas d'utilisation suivants pour le numéro :
     - Two-factor authentication (Authentification à deux facteurs) :pour envoyer des codes d'authentification à deux facteurs.
     - One-time passwords (Mots de passe à usage unique) : pour envoyer un mot de passe à usage unique à un utilisateur.
     - Notifications : si vous avez uniquement l'intention d'envoyer des notifications importantes à vos utilisateurs.
     - Polling and surveys (Sondages et enquêtes) : pour interroger les utilisateurs sur leurs préférences.
     - Info on demand (Informations à la demande) : pour envoyer des messages aux utilisateurs après l'envoi d'une demande.
     - Promotions and Marketing (Promotions et marketing) : si vous avez uniquement l'intention d'envoyer des messages marketing à vos utilisateurs.
     - Other (Autre) : si votre cas d'utilisation n'entre dans aucune autre catégorie. Assurez-vous de remplir le champ Use Case Details (Détails du cas d'utilisation).
   - Renseignez Use Case Details (Détails du cas d'utilisation) : champ facultatif qui permet d'indiquer un contexte supplémentaire à la catégorie de cas d'utilisation sélectionnée.

9. Dans la section Messaging Samples (Exemples de messagerie), procédez comme suit :
   - Pour Message Sample 1 (Exemple de message 1), saisissez un exemple de message d'un corps de message SMS qui sera envoyé à vos utilisateurs finaux.
   - Pour Message Sample 2 – optional (Exemple de message 2 - facultatif) et Message Sample 3 – optional (Exemple de message 3 - facultatif), vous pouvez, si nécessaire, saisir d'autres exemples de messages du corps de message SMS qui seront envoyés.
   - Chaque zone de texte Message Sample (Exemple de message) a une limite maximale de 306 caractères.
10. Lorsque vous avez terminé, sélectionnez Submit registration (Envoyer l'enregistrement).

> **Important**
>
> Vous pouvez vérifier le statut de votre enregistrement en suivant les instructions ici : État d'enregistrement de l'ID d'expéditeur pour Singapour.
> N'attendez pas d'approbation ni de changement de statut après avoir soumis l'enregistrement de votre ID d'expéditeur. Passez immédiatement à Enregistrement d'un ID d'expéditeur auprès du centre SGNIC (Singapore Network Information Centre).

Enregistrement manuel pour terminer le processus d'enregistrement d'ID d'expéditeur pour Singapour

Suivez ces étapes pour enregistrer un ID d'expéditeur lorsque le compte ne se trouve pas dans une région Amazon Pinpoint. Si votre compte se trouve dans une région Amazon Pinpoint, consultez Enregistrement de votre ID d'expéditeur pour Singapour avec Amazon Pinpoint.

1. Téléchargez le fichier Singapore_Sender_ID_Registration_LOA_Template.zip et renseignez les informations requises.
2. Créez un cas auprès d'AWS Support.
3. Dans l'onglet Open support cases (Cas de support ouverts), choisissez Create case (Créer un cas).
4. Choisissez Looking for service limit increases (Rechercher une augmentation de la limite de service) et, pour le type de limite, choisissez SNS Text Messaging (Messages SMS SNS).
5. Pour Resource Type (Type de ressource), choisissez Sender ID Registration (Enregistrement d'ID d'expéditeur).
6. Joignez le document LOA et envoyez la demande.

Enregistrement d'un ID d'expéditeur auprès du centre SGNIC (Singapore Network Information Centre)

> **Warning**
>
> Si vous réalisez ces étapes dans un autre ordre, votre ID d'expéditeur peut être bloqué par le service ou ne pas être conservé sur l'appareil mobile.

1. Vous devez d'abord enregistrer votre ID d'expéditeur pour Singapour (SG) pour votre compte avec AWS (console Amazon Pinpoint ou procéder à un enregistrement manuel pour les régions autres qu'Amazon Pinpoint). Une fois cette étape terminée, vous pouvez passer à l'étape suivante.
2. Rapprochez-vous du centre SGNIC pour enregistrer votre ID d'expéditeur en utilisant le processus du registre SSIR (SMS Sender ID Registry) du centre SGNIC.
   - Lorsque vous terminez le processus, assurez-vous de répertorier tous les éléments suivants en tant qu'agrégateurs participants :
     - AMCS SG Private Limited (Amazon Media Communications Services)
     - Nexmo PTE LTD
     - Sinch Singapore PTE LTD
     - Telesign Singapore PTE LTD
     - Twilio Singapore PTD LTD

> **Note**
>
> Vous êtes tenu de soumettre l'enregistrement d'ID d'expéditeur à partir de chaque compte AWS individuel avec lequel vous envisagez d'utiliser l'ID d'expéditeur.

État d'enregistrement de l'ID d'expéditeur pour Singapour

Lorsque vous enregistrez votre ID d'expéditeur pour Singapour avec Amazon SNS, le statut de votre enregistrement sera l'un des cinq statuts suivants :

- Created (Créé) : votre enregistrement est créé mais n'a pas été envoyé.
- Submitted (Envoyé) : votre enregistrement a été envoyé et est en cours de validation.

- Reviewing (Vérification en cours) : votre enregistrement a été accepté et est en cours de vérification. La vérification peut prendre entre 1 et 3 semaines. Dans certains cas, elle peut prendre plus de temps.
- Complete (Terminé) : votre enregistrement a été approuvé et vous pouvez commencer à utiliser l'ID d'identifiant.
- Requires Updates (Mises à jour requises) : vous devez corriger votre enregistrement et le soumettre à nouveau. Pour en savoir plus, consultez Modification de l'enregistrement d'un ID d'expéditeur pour Singapour. Les champs qui exigent des mises à jour sont accompagnés d'une icône d'avertissement et d'une brève description du problème.

Pour toutes les régions autres que les régions Amazon Pinpoint, AWS Support enverra une confirmation par e-mail lors de l'inscription ou créera un dossier auprès d'AWS Support.

- Dans l'onglet Open support cases (Cas de support ouverts), choisissez Create case (Créer un cas).
- Sélectionnez Service Limit increase (Augmentation des limites de service).
- Pour Type de ressource, choisissez Sender ID Registration (Enregistrement de l'ID d'expéditeur) et pour limite, choisissez General Inquiry (Demande générale).

Vérification du statut de votre enregistrement

1. Connectez-vous à la console de gestion AWS et ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet Sender ID registrations (Enregistrements de l'ID d'expéditeur), choisissez SenderID.
4. Vous pouvez alors afficher le statut d'enregistrement de chaque SenderID.

Modification de l'enregistrement d'un ID d'expéditeur pour Singapour

Après avoir soumis votre enregistrement avec Amazon Pinpoint, le statut d'enregistrement est défini sur Requires Updates (Mises à jour requises) en cas de problème avec l'enregistrement. Le formulaire d'inscription est alors modifiable. Les champs qui exigent des mises à jour sont accompagnés d'une icône d'avertissement et d'une brève description du problème.

Pour modifier un ID d'expéditeur

1. Ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet SenderID Registration (Enregistrement ID d'expéditeur), choisissez le numéro que vous souhaitez modifier et sélectionnez le Registration ID (ID d'enregistrement).
4. Choisissez Update registration (Mettre à jour l'enregistrement) pour modifier le formulaire et corriger les champs comportant une icône d'avertissement.
5. Si vous vous enregistrez au nom d'une autre marque/entité, vous devrez télécharger à nouveau les fichiers précédemment envoyés dans Letter of authorization image – optional (Image de la lettre d'autorisation - facultatif).
6.
> **Important**
>
> Vérifiez à nouveau tous les champs pour vous assurer qu'ils sont corrects.

7. Choisissez Submit registration (Envoyer l'enregistrement) pour envoyer à nouveau l'enregistrement lorsque que vous avez terminé.

Suppression de l'enregistrement d'un ID d'expéditeur pour Singapour

Si vous ne souhaitez plus utiliser votre enregistrement d'ID d'expéditeur pour Singapour, vous pouvez le supprimer. Les enregistrements ne peuvent être supprimés que si leur statut est Created (Créé) ou Requires Updates (Doit être mis à jour).

Pour supprimer un enregistrement

1. Ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet Sender ID (ID d'expéditeur), choisissez l'ID que vous souhaitez modifier et sélectionnez Delete Registration (Supprimer l'enregistrement).

Problèmes d'enregistrement pour Singapour

Si votre ID d'expéditeur pour Singapour n'est pas accepté par Amazon Pinpoint, un message explique pourquoi il a été refusé. Si vous avez des questions concernant ces refus et que vous ne

trouvez pas de réponse sur la page Bonnes pratiques, vous pouvez envoyer une demande à nos équipes de support.

Pour soumettre une demande d'informations à propos d'un ID d'expéditeur pour Singapour refusé

1. Ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Choisissez Support, puis Centre de support.
3. Sur la page de Support, choisissez Création d'un cas.
4. Pour Case type (Type de cas), choisissez Service limit increase (Augmentation des limites de service).
5. Pour Limit type (Type de limite), choisissez Pinpoint SMS (SMS Pinpoint).
6. Dans la section Requests (Demandes), procédez de la manière suivante :
   - Pour Resource Type (Type de ressource), choisissez Sender ID Registration (Enregistrement d'ID d'expéditeur).
   - Pour le champ Limit (Limit), choisissez Registration Rejection Query (Demande concernant un refus d'enregistrement).
7. Dans Use case description (Description du cas d'utilisation), entrez l'ID d'expéditeur pour Singapour refusé et le motif du refus.
8. Sous Options de contact, pour Langue de contact préférée, choisissez la langue que vous préférez utiliser lors de la communication avec l'équipe de support AWS.
9. Pour Méthode de contact, choisissez le mode de communication que vous souhaitez utiliser pour échanger avec l'équipe de support AWS.
10. Choisissez Envoyer.

L'équipe AWS Support fournira des informations sur les raisons pour lesquelles votre enregistrement d'ID d'expéditeur a été refusé dans votre cas de support AWS.

Questions fréquentes sur l'enregistrement d'ID d'expéditeur pour Singapour

Questions fréquentes sur le processus d'enregistrement d'ID d'expéditeur pour Singapour avec Amazon Pinpoint Pinpoint.

Ai-je déjà un ID d'expéditeur pour Singapour ?

Pour vérifier si vous possédez un ID d'expéditeur pour Singapour

1. Ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.

2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet SenderID Registration (Enregistrement ID d'expéditeur), choisissez l'ID d'expéditeur que vous souhaitez afficher et sélectionnez l'ID d'enregistrement.

Combien de temps dure le processus d'enregistrement ?

La vérification prend généralement entre 1 et 3 semaines. Dans certains cas, elle peut prendre 5 semaines, voire plus, afin que vos informations soient vérifiées auprès des agences gouvernementales.

Qu'est-ce qu'un numéro d'entité unique (UEN) et comment en obtenir un ?

Un UEN est un identifiant d'entreprise à Singapour. Il est délivré par l'ACRA (Autorité de réglementation de la comptabilité et des entreprises). Les entreprises locales et les entreprises de Singapour peuvent obtenir un UEN auprès de l'ACRA. Une fois que vous aurez suivi la procédure d'enregistrement et d'incorporation standard, l'UEN sera émis. Vous pouvez demander un UEN auprès de l'ACRA via Bizfile.

Dois-je enregistrer un ID d'expéditeur pour Singapour ?

Oui. Si vous n'avez pas enregistré votre ID d'expéditeur pour Singapour au plus tard le 30/01/2023, l'ID de tout message envoyé à l'aide d'un ID d'expéditeur sera remplacé par LIKELY-SCAM.

Comment enregistrer mon ID d'expéditeur pour Singapour avec Amazon Pinpoint ?

Suivez les instructions de la section Enregistrement de votre ID d'expéditeur pour Singapour avec Amazon Pinpoint pour enregistrer un ID d'expéditeur.

Quel est le statut d'enregistrement de mon ID d'expéditeur pour Singapour et que signifie-t-il ?

Suivez les instructions de la rubrique État d'enregistrement de l'ID d'expéditeur pour Singapour pour vérifier votre enregistrement et son statut.

Quelles informations dois-je fournir ?

Vous devez fournir l'adresse de votre entreprise, un contact professionnel et un cas d'utilisation. Pour obtenir les informations requises, consultez la rubrique Enregistrement de votre ID d'expéditeur pour Singapour avec Amazon Pinpoint.

Que se passe-t-il si l'enregistrement de mon ID d'expéditeur pour Singapour est refusé ?

Si votre enregistrement est refusé, son statut devient Requires Updates (Mises à jour requises) et vous pouvez effectuer des mises à jour en suivant les instructions de la rubrique Modification de l'enregistrement d'un ID d'expéditeur pour Singapour.

De quelles autorisations ai-je besoin ?

L'utilisateur/rôle IAM que vous utilisez pour accéder à la console Amazon Pinpoint doit être activé avec l'autorisation *"sms-voice:*"*.

Numéros d'origine

Un numéro d'origine est une chaîne numérique qui identifie le numéro de téléphone de l'expéditeur de message SMS. Lorsque vous envoyez un message SMS à l'aide d'un numéro d'origine, l'appareil du destinataire affiche le numéro d'origine comme le numéro de téléphone de l'expéditeur. Vous pouvez spécifier différents numéros d'origine par cas d'utilisation.

> **Tip**
>
> Pour afficher une liste de tous les numéros d'origine existants dans votre compte AWS, dans le panneau de navigation de la console Amazon SNS, choisissez Numéros d'origine.

La prise en charge des numéros d'origine n'est pas disponible dans les pays où les lois locales exigent l'utilisation des ID d'expéditeur au lieu de numéros d'origine.

Rubriques

- 10DLC
- Numéros gratuits
- Codes courts
- Codes longs de personne à personne (P2P)
- Comparaison des numéros de produits américains

10DLC

Les opérateurs américains ne prennent plus en charge l'utilisation de messagerie SMS d'application à personne (A2P) sur des codes longs locaux et non enregistrés. Pour la messagerie SMS A2P à

volume élevé, les opérateurs américains offrent plutôt un nouveau type de code long appelé codes longs à 10 chiffres (10DLC).

> **Important**
>
> Le 26 janvier 2023, les fournisseurs de SMS d'Amazon SNS ont introduit de nouveaux processus d'évaluation manuelle pour les campagnes 10DLC, afin de répondre aux préoccupations en matière de spam par SMS soulevées par les opérateurs américains. Vous pouvez utiliser des codes courts et des numéros gratuits comme alternatives aux 10DLC pour envoyer des SMS aux États-Unis.
> Actuellement, nos fournisseurs de SMS n'ont pas fourni d'objectifs de niveau de service concernant la durée prévue des évaluations des campagnes 10DLC. Les évaluations sont déclenchées lorsqu'un numéro est associé à une campagne 10DLC. Les évaluations prennent plus de temps que le délai estimé de 14 jours précédemment communiqué par Amazon SNS.
> Amazon SNS travaille quotidiennement avec les fournisseurs de SMS pour garantir que :
>
> - les fournisseurs terminent les évaluations de campagnes 10DLC en attente dès que possible ;
> - les fournisseurs classent les demandes AWS par ordre de priorité dans leurs dossiers en suspens.
>
> Vous pouvez vérifier le statut des campagnes 10DLC en suivant les instructions fournies dans Campagnes 10DLC. Si des informations supplémentaires sont nécessaires pour approuver une campagne 10DLC, l'équipe d'assistance AWS vous en informera. L'enregistrement d'un numéro gratuit aux États-Unis pourra éventuellement être plus rapide que l'obtention de numéros 10DLC. Pour plus d'informations sur les numéros gratuits américains et le processus d'enregistrement, consultez Exigences relatives à l'enregistrement d'un numéro gratuit et processus d'enregistrement.

Qu'est-ce que 10DLC ?

10DLC est un type de code long qui est enregistré auprès des opérateurs pour prendre en charge la messagerie SMS A2P à volume élevé en utilisant le format de numéro de téléphone à 10 chiffres. Amazon SNS n'offre plus de codes longs locaux en tant que produit SMS et propose plutôt 10DLC. 10DLC ne vous affecte pas si vous n'utilisez que des codes courts et des numéros sans frais.

10DLC est un numéro de téléphone à 10 chiffres utilisé uniquement aux États-Unis. Les messages envoyés à partir d'un 10DLC aux destinataires affichent un numéro à 10 chiffres comme expéditeur. Contrairement aux numéros sans frais, 10DLC prend en charge à la fois la messagerie transactionnelle et la messagerie promotionnelle et peut inclure n'importe quel indicatif régional américain.

Si vous avez des codes longs locaux existants, vous pouvez demander que leurs codes longs locaux soient activés pour 10DLC. Pour ce faire, complétez le processus d'enregistrement 10DLC, puis soumettez un ticket de support. En cas de problème avec l'activation de votre code long pour 10DLC, vous êtes informé et invité à demander un nouveau 10DLC via la console Amazon Pinpoint (et non Amazon SNS). Pour plus d'informations sur la façon de déposer un ticket de support pour convertir un code long, consultez Associer un code long à une campagne 10DLC.

Afin d'utiliser un numéro 10DLC, inscrivez d'abord votre entreprise et créez une campagne 10DLC à l'aide de la console Amazon Pinpoint (et non Amazon SNS). AWS partage ces informations avec The Campaign Registry, un tiers qui approuve ou refuse votre enregistrement sur la base de ces informations. Dans certains cas, l'enregistrement se produit immédiatement. Par exemple, si vous vous êtes déjà inscrit à The Campaign Registry, il se peut qu'il dispose déjà de vos informations. Toutefois, l'approbation de certaines campagnes peut durer une semaine ou plus. Une fois votre entreprise et votre campagne 10DLC approuvées, vous pouvez acheter un numéro 10DLC et l'associer à votre campagne. L'approbation d'une demande d'un 10DLC peut également durer jusqu'à une semaine. Bien que vous puissiez associer plusieurs 10DLC à une seule campagne, vous ne pouvez pas utiliser le même 10DLC sur plusieurs campagnes. Pour chaque campagne que vous créez, vous devez disposer d'un 10DLC unique.

Capacités de 10DLC

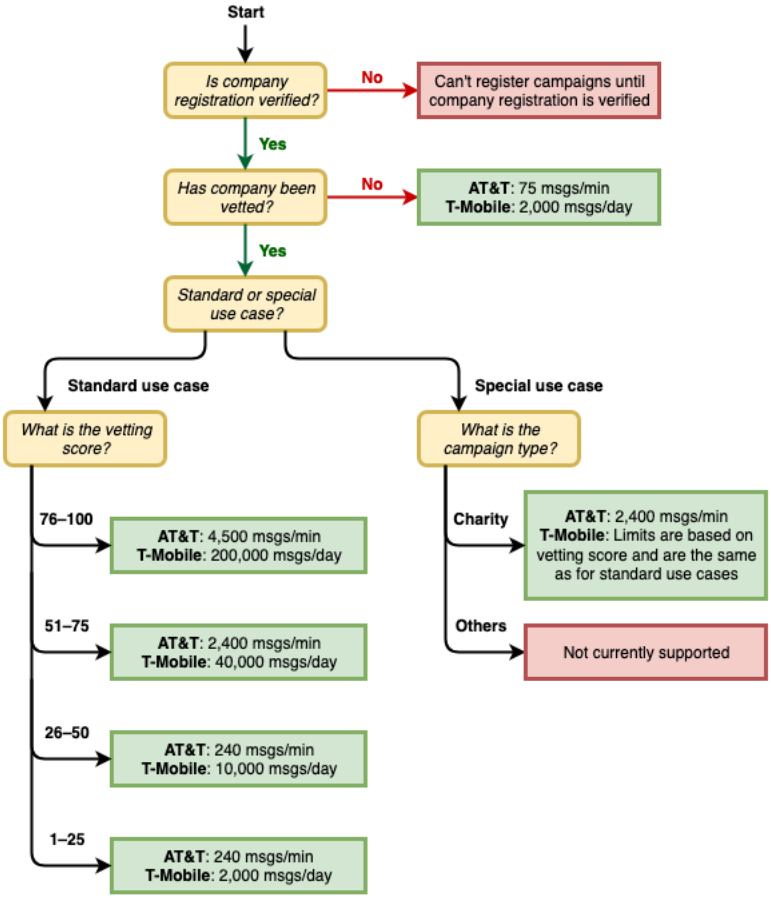
Les capacités des numéros de téléphone 10DLC dépendent des opérateurs mobiles utilisés par vos destinataires. AT&T fournit un nombre limite de parties de message pouvant être envoyées chaque minute pour chaque campagne. T-Mobile offre une limite journalière de messages pouvant être envoyés pour chaque entreprise, sans limite du nombre de parties de message pouvant être envoyées par minute. Verizon n'a pas publié de limites de débit, mais utilise un système de filtrage pour 10DLC conçu pour supprimer le courrier indésirable, les messages non sollicités et le contenu abusif, en mettant moins l'accent sur le débit réel des messages.

Les nouvelles campagnes 10DLC associées à des entreprises non vérifiées peuvent envoyer 75 parties de messages par minute aux destinataires qui utilisent AT&T, et 2 000 messages par jour aux destinataires qui utilisent T-Mobile. La limite de l'entreprise est partagée entre l'ensemble de

vos campagnes 10DLC. Par exemple, si vous avez enregistré une entreprise et deux campagnes, l'allocation quotidienne de 2 000 messages aux clients T-Mobile est partagée entre ces campagnes. De même, si vous enregistrez la même entreprise dans plusieurs comptes AWS, l'allocation quotidienne est partagée entre ces comptes.

Si vos besoins en termes de débit dépassent ces limites, vous pouvez demander que l'enregistrement de votre entreprise soit vérifié. Lorsque vous vérifiez l'enregistrement de votre entreprise, un fournisseur de vérification tiers analyse les détails de votre entreprise. Le fournisseur de vérification fournit ensuite un score de vérification, qui détermine les capacités de vos campagnes 10DLC. Des frais ponctuels s'appliquent pour le service de vérification. Pour de plus amples informations, veuillez consulter Vérification de votre enregistrement 10DLC Amazon SNS.

Votre débit réel varie en fonction de divers facteurs, tels que le fait que votre entreprise ait été vérifiée ou non, vos types de campagnes et votre score de vérification. Le diagramme suivant illustre les débits pour diverses situations.



Les débits pour 10DLC sont déterminés par les opérateurs mobiles américains en coopération avec The Campaign Registry. Ni Amazon SNS ni aucun autre service d'envoi de SMS ne peuvent augmenter le débit 10DLC au-delà de ces tarifs. Si vous avez besoin de débits élevés et de taux de

délivrabilité élevés pour l'ensemble des opérateurs américains, nous vous recommandons d'utiliser un code court. Pour plus d'informations sur l'obtention d'un code court, consultez Demande de codes courts dédiés pour la messagerie SMS avec Amazon SNS .

Mise en route avec 10DLC

Utilisez la console Amazon Pinpoint (et non Amazon SNS) pour demander votre 10DLC. Procédez comme suit pour configurer 10DLC pour vos campagnes 10DLC.

1. Enregistrez votre entreprise.

   Avant de pouvoir demander un 10DLC, votre entreprise doit être inscrite auprès de The Campaign Registry ; pour plus d'informations, consultez Enregistrement d'une entreprise. L'enregistrement est généralement instantané, à moins que The Campaign Registry ne nécessite plus d'informations. Il y a des frais d'enregistrement uniques pour enregistrer votre entreprise, affichés sur la page d'enregistrement. Ces frais uniques sont payés séparément de vos frais mensuels pour la campagne et 10DLC.

   > **Note**
   >
   > La messagerie SMS Amazon SNS est disponible dans les régions où Amazon Pinpoint n'est pas actuellement pris en charge. Il existe deux cas différents :
   >
   > a. Si vous utilisez un compte cloud commercial, vous devez ouvrir la console Amazon Pinpoint dans la région USA Est (Virginie du Nord) pour enregistrer votre entreprise et votre campagne 10DLC. Ne demandez pas de numéro 10DLC.
   >
   > b. Utilisez la console AWS Service Quotas pour créer une demande d'augmentation de la limite de service tout en demandant le numéro 10DLC pour cette région. Pour plus d'informations sur les régions où Amazon Pinpoint est disponible, consultez Points de terminaison et quotas Amazon Pinpoint dans la Références générales AWS.
   >
   > c. Si vous utilisez un compte AWS GovCloud (US), ouvrez la console Amazon Pinpoint dans la région USA Ouest pour enregistrer votre entreprise et votre campagne 10DLC. Ne demandez pas de numéro 10DLC. Utilisez plutôt la AWS console Service Quotas pour créer un cas visant à augmenter la limite de service tout en demandant le numéro 10DLC de cette région. Pour plus d'informations sur les régions où Amazon Pinpoint est disponible, consultez Points de terminaison et quotas Amazon Pinpoint dans la Références générales AWS.

2. (Facultatif, mais recommandé) Demandez une vérification

Si l'enregistrement de votre entreprise aboutit, vous pouvez commencer à créer des campagnes 10DLC à faible volume et à usage mixte. Ces campagnes peuvent envoyer 75 messages par minute aux destinataires qui utilisent AT&T, et votre entreprise enregistrée peut envoyer 2 000 messages par jour aux destinataires qui utilisent T-Mobile. Si votre cas d'utilisation nécessite un débit supérieur à ces valeurs, vous pouvez demander une vérification de l'enregistrement de votre entreprise. La vérification de l'enregistrement de votre entreprise peut augmenter les débits de vos entreprises et campagnes, sans toutefois être garanti. Pour plus d'informations sur la vérification, consultez Vérification de votre enregistrement 10DLC Amazon SNS.

3. Enregistrez votre campagne.

   Une fois votre entreprise enregistrée, créez une campagne 10DLC et associez-la à l'une de vos entreprises enregistrées. Cette campagne est soumise à The Campaign Registry pour approbation. Dans la plupart des cas, l'approbation de la campagne 10DLC est instantanée, à moins que The Campaign Registry ne nécessite plus d'informations. Pour de plus amples informations, veuillez consulter Enregistrement d'une campagne 10DLC.

4. Demandez votre numéro 10DLC.

   Une fois votre campagne 10DLC approuvée, vous pouvez demander un 10DLC et associer ce numéro à la campagne approuvée. Votre campagne 10DLC ne peut utiliser qu'un numéro approuvé pour elle. Consultez Demande de numéros 10DLC, de numéros sans frais et de codes longs P2P pour la messagerie SMS avec Amazon SNS.

Frais d'enregistrement et mensuels 10DLC

Il y a des frais d'enregistrement et mensuels associés à l'utilisation de 10DLC, tels que l'enregistrement de votre entreprise et de votre campagne 10DLC. Ces frais sont distincts de tous les autres frais mensuels facturés par AWS. Pour de plus amples informations, veuillez consulter la page Tarification SMS mondiale d'Amazon SNS.

Enregistrement d'une entreprise

Avant de pouvoir demander un 10DLC, vous devez enregistrer votre entreprise auprès de The Campaign Registry.

> **Note**
>
> La messagerie SMS Amazon SNS est disponible dans les régions où Amazon Pinpoint n'est pas actuellement pris en charge. Dans ces cas, ouvrez la console Amazon Pinpoint

dans la région USA Est (Virginie du Nord) pour enregistrer votre entreprise et votre campagne 10DLC, mais ne demandez pas de numéro 10DLC. Utilisez plutôt la AWS console Service Quotas pour créer un cas visant à augmenter la limite de service tout en demandant le numéro 10DLC de cette région. Pour plus d'informations sur les régions où Amazon Pinpoint est disponible, consultez Points de terminaison et quotas Amazon Pinpoint dans la Références générales AWS.

Statuts d'enregistrement d'une entreprise 10DLC

Lorsque vous enregistrez votre entreprise ou votre marque, l'un des deux statuts suivants est renvoyé : Unverified (Non vérifié) ou Verified (Vérifié). Si le statut de l'enregistrement de votre entreprise est Unverified (Non vérifié), cela signifie qu'un problème s'est produit avec votre enregistrement. Par exemple, le nom de l'entreprise enregistrée que vous avez fourni peut ne pas correspondre exactement au nom enregistré de l'entreprise associée au numéro d'identification fiscale que vous avez fourni. Si vous identifiez un problème dans les détails d'enregistrement de votre entreprise, vous pouvez les corriger. Pour plus d'informations sur la modification des détails d'enregistrement de votre entreprise, consultez Modification ou suppression d'une entreprise enregistrée.

Si le statut de l'enregistrement de votre entreprise est Verified (Vérifié), les détails d'enregistrement que vous avez fournis étaient exacts et vous pouvez commencer à créer des campagnes 10DLC.

Enregistrement de votre entreprise ou de votre marque

Vous n'avez besoin d'enregistrer votre entreprise qu'une seule fois. Une fois enregistrée, vous pouvez modifier votre entreprise et vos coordonnées. Pour supprimer une entreprise enregistrée, créez un cas auprès du support AWS. Pour plus d'informations sur la modification ou la suppression des détails de l'entreprise, consultez Modification ou suppression d'une entreprise enregistrée.

Pour enregistrer une entreprise

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet 10DLC campaigns (Campagnes 10DLC), choisissez Register company (Enregistrer une entreprise).

> **Note**
>
> La page Register your company (Enregistrement de votre entreprise) affiche les frais d'enregistrement. Il s'agit de frais ponctuels associés à l'enregistrement de votre entreprise. Ce coût est distinct de tout autre coût ou frais mensuel. Il vous est facturé lorsque vous enregistrez votre entreprise ou lorsque vous modifiez les détails d'enregistrement d'une entreprise existante.

4. Dans la section Company info (Informations sur l'entreprise), procédez comme suit :

   - Pour Legal company name (Nom légal de l'entreprise), saisissez le nom sous lequel l'entreprise est enregistrée. Le nom que vous saisissez doit correspondre exactement au nom de l'entreprise associé au numéro d'identification fiscale que vous fournissez.

   > **Important**
   >
   > Assurez-vous d'utiliser le nom légal exact de votre entreprise. Une fois soumises, vous ne pouvez pas modifier ces informations. Des informations incorrectes ou incomplètes peuvent entraîner le retard ou le refus de votre enregistrement.

   - Pour What type of legal form is this organization (Quel est le type de forme juridique de cette entreprise), choisissez l'option qui décrit le mieux votre entreprise.

   > **Note**
   >
   > Les options US government (Gouvernement américain) et Not-for-profit (But non lucratif) ne peuvent être utilisées que pour enregistrer les organisations basées aux États-Unis. Si votre organisation est basée dans un pays autre que les États-Unis, vous devez vous enregistrer en tant que Private for-profit (Organisation privée à but lucratif), quelle que soit la forme juridique réelle de votre organisation.

   - Si vous avez choisi Public for profit (Organisation publique à but lucratif) à l'étape précédente, entrez le symbole boursier de l'entreprise et la bourse à laquelle elle est cotée.

- Pour Country of registration (Pays d’enregistrement), choisissez le pays dans lequel l'entreprise est enregistrée.
- Pour Doing Business As (DBA) or brand name (Faire des affaires en tant que (DBA) ou nom de marque), saisissez tous les autres noms sous lesquels votre entreprise exerce ses activités.
- Pour Tax ID (Numéro d'identification fiscale), saisissez le numéro d'identification fiscale de votre entreprise. Le numéro que vous saisissez dépend du pays dans lequel votre entreprise est enregistrée.
  - Si vous enregistrez une entité américaine ou non américaine qui possède un numéro EIN de l'IRS, saisissez votre EIN à neuf chiffres. Le nom légal de l’entreprise, l'EIN et l'adresse physique que vous saisissez doivent tous correspondre aux informations de l’entreprise enregistrées auprès de l'IRS.
  - Si vous enregistrez une entité canadienne, saisissez votre numéro d’entreprise du gouvernement fédéral ou provincial. Ne saisissez pas le numéro d'entreprise (NE) fourni par l'ARC. Le nom légal de l’entreprise, le numéro d’entreprise et l'adresse physique que vous saisissez doivent tous correspondre aux informations de l’entreprise enregistrées auprès de Corporations Canada.
  - Si vous enregistrez une entité basée dans un autre pays, saisissez le numéro d’identification fiscale principal pour votre pays. Dans de nombreux pays, il s'agit de la partie numérique de votre numéro de TVA.
- Pour Vertical, choisissez la catégorie qui décrit le mieux l'entreprise que vous enregistrez.

5. Dans la section Contact info (Coordonnées), procédez comme suit :

- Pour Address/Street (Adresse/Rue), saisissez l'adresse postale physique associée à votre entreprise.
- Pour City (Ville), saisissez la ville dans laquelle se trouve l'adresse physique.
- Pour State or region (État ou région), saisissez l’état ou la région dans laquelle figure l’adresse.
- Pour Zip Code/Postal Code (Code postal), saisissez le code postal de l’adresse.
- Pour Company website (Site web de l’entreprise), saisissez l’URL complète du site web de votre entreprise. Incluez « http:// » ou « https:// » au début de l'adresse.
- Pour Support email (E-mail de contact), saisissez votre adresse e-mail.
- Pour Support phone number (Numéro de téléphone de contact), saisissez un numéro de téléphone et son code pays.

> **Note**
>
> The Campaign Registry exige une adresse e-mail et un numéro de téléphone de contact au cas où ils doivent vérifier les informations d'enregistrement auprès d'un représentant de votre entreprise.

6. Lorsque vous avez terminé, choisissez Create (Créer). L'enregistrement de votre entreprise est soumis à The Campaign Registry. Dans la plupart des cas, votre enregistrement est accepté immédiatement et un statut est fourni.

Si le statut d’enregistrement de votre entreprise est Verified (Vérifié), vous pouvez commencer à créer des campagnes 10DLC à faible volume et à usage mixte. Vous pouvez utiliser ce type de campagne pour envoyer jusqu’à 75 messages par minute aux destinataires qui utilisent AT&T, et votre entreprise enregistrée peut envoyer 2 000 messages par jour aux destinataires qui utilisent T-Mobile. Vous pouvez également envoyer des messages aux destinataires qui utilisent d'autres opérateurs américains, tels que Verizon et US Cellular. Ces opérateurs n'appliquent pas strictement les limites de débit, mais ils surveillent fortement les messages 10DLC à la recherche de signes de courrier indésirable et d'abus.

Si votre cas d'utilisation nécessite un débit supérieur à ces valeurs, vous pouvez demander une vérification supplémentaire de l'enregistrement de votre entreprise. Pour plus d'informations sur la vérification de l'enregistrement de votre marque, consultez Vérification de votre enregistrement 10DLC Amazon SNS.

Si le statut de l'enregistrement de votre entreprise est Unverified (Non vérifié), les informations que vous avez fournies étaient incorrectes. Vérifiez les informations que vous avez fournies et confirmez que tous les champs contiennent des informations correctes. Vous pouvez modifier certaines parties de l'enregistrement de votre entreprise dans la console Amazon Pinpoint. Pour plus d'informations sur la modification des détails d’enregistrement de votre entreprise, consultez Modification d'un enregistrement d'entreprise 10DLC.

Vérification de votre enregistrement 10DLC Amazon SNS

Si l'enregistrement de votre entreprise aboutit et que vous souhaitez enregistrer une campagne avec des capacités de débit plus élevées, vous devez vérifier l'enregistrement de votre entreprise.

Lorsque vous vérifiez votre enregistrement, une organisation tierce analyse les détails de l'entreprise que vous avez fournis et renvoie un score de vérification. Un score de vérification élevé peut entraîner des débits plus élevés pour votre entreprise 10DLC et les campagnes qui y sont associées. Cependant, il n'est pas garanti que la vérification augmente votre débit.

Les scores de vérification ne sont pas appliqués rétroactivement. En d'autres termes, si vous avez déjà créé une campagne 10DLC et que vous vérifiez ultérieurement l'enregistrement de votre entreprise, votre score de vérification n'est pas automatiquement appliqué à votre campagne existante. Il est donc recommandé de vérifier votre entreprise ou votre marque avant de créer l'une de vos campagnes 10DLC.

> **Note**
>
> Des frais non remboursables de 40 $ sont facturés pour vérifier votre entreprise ou votre marque.

Pour vérifier l'enregistrement de votre entreprise

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet 10DLC campaigns (Campagnes 10DLC), choisissez l'entreprise 10DLC que vous voulez vérifier.
4. En bas de la page des détails de l'entreprise, choisissez Apply for vetting (Demander une vérification).
5. Dans la fenêtre Apply for additional vetting (Demander une vérification supplémentaire), choisissez Submit (Soumettre).

   Pour les entreprises américaines, le processus de vérification prend généralement une minute environ. Pour les entreprises non américaines, le processus de vérification peut prendre beaucoup plus de temps, selon la disponibilité des données pour ce pays.

Une fois que vous avez soumis une demande de vérification, vous revenez à la page des détails de l'entreprise. La section Company vetting results (Résultats de la vérification de l'entreprise) affiche le statut et les résultats de votre demande de vérification. Lorsque le processus de vérification est

terminé, ce tableau affiche un score de vérification dans la colonne Score. Votre score de vérification détermine vos capacités de débit 10DLC. Votre débit varie en fonction du type de campagne que vous créez. Si vous créez des campagnes 10DLC à usage mixte ou marketing, vous devez obtenir un score de vérification supérieur à celui dont vous avez besoin pour les autres types de campagnes afin d'atteindre des débits élevés. Pour plus d'informations sur les capacités des numéros de téléphone 10DLC, consultez Capacités de 10DLC.

Si vous modifiez les détails de l'enregistrement de votre entreprise après avoir terminé le processus de vérification, vous pouvez demander de vérifier à nouveau votre enregistrement. Si vous modifiez uniquement le paramètre Vertical pour l'enregistrement de votre entreprise, votre score de vérification ne changera pas. Si vous modifiez des détails autres que le paramètre Vertical, le résultat de votre vérification peut changer. Dans les deux cas, les frais de vérification ponctuels vous sont à nouveau facturés.

Modification ou suppression d'une entreprise enregistrée

Vous pouvez modifier certaines informations d'enregistrement 10DLC de votre entreprise directement dans la console Amazon Pinpoint. Vous pouvez également supprimer un enregistrement d'entreprise 10DLC en créant un cas dans le Centre de support AWS .

Modification d'un enregistrement d'entreprise 10DLC

Une fois que vous avez terminé le processus d'enregistrement 10DLC d’une entreprise, vous pouvez modifier les détails de votre enregistrement.

Si un message d’erreur s’affiche après avoir modifié les informations d’enregistrement de votre entreprise, il peut y avoir d’autres problèmes avec votre enregistrement. Vous pouvez ouvrir un ticket auprès du AWS Support pour demander plus d'informations.

Pour modifier l’enregistrement d’une entreprise

1. Ouvrez la AWS SMS console à l'adresse https://console.aws.amazon.com/sms-voice/.
2. Suivez les instructions relatives à la modification de votre inscription dans le guide de l'utilisateur Amazon Pinpoint SMS.

Suppression d’un enregistrement d’entreprise 10DLC

Pour supprimer l'enregistrement d'une entreprise

1. Ouvrez la AWS SMS console à l'adresse https://console.aws.amazon.com/sms-voice/.

2. Suivez les instructions relatives à la suppression de votre inscription dans le guide de l'utilisateur Amazon Pinpoint SMS.

Enregistrement d'une campagne 10DLC

Lorsque vous enregistrez une campagne 10DLC, vous fournissez une description de votre cas d'utilisation, ainsi que les modèles de message que vous prévoyez d'utiliser. Avant de pouvoir créer et enregistrer une campagne 10DLC, vous devez commencer par enregistrer votre entreprise. Pour plus d'informations sur l'enregistrement de votre entreprise, consultez Enregistrement d’une entreprise.

> **Note**
>
> Après avoir enregistré votre entreprise, Amazon Pinpoint affiche l'un des deux statuts d’enregistrement suivants : Verified (Vérifié) ou Unverified (Non vérifié). Vous ne pouvez terminer le processus d'enregistrement d’une campagne 10DLC que si le statut de l'enregistrement de votre entreprise est Verified (Vérifié). Vous serez en mesure de créer des campagnes à faible volume et à usage mixte.
>
> Si le statut est Unverified (Non vérifié), cela signifie généralement que certaines des données que vous avez fournies lors de l'enregistrement de votre entreprise étaient incorrectes. Vous ne pourrez pas créer de campagnes 10DLC tant que votre entreprise possède ce statut. Vous pouvez modifier l'enregistrement de votre entreprise pour tenter de résoudre les problèmes liés à son enregistrement. Pour plus d'informations sur la modification des enregistrements d'entreprise 10DLC, consultez Modification ou suppression d'une entreprise enregistrée.

Sur cette page, vous devez d'abord fournir les détails sur l'entreprise pour laquelle vous créez la campagne 10DLC, puis fournir les détails du cas d'utilisation de la campagne elle-même. Les informations figurant sur cette page sont ensuite fournies à The Campaign Registry pour approbation.

Dans cette section, vous choisirez l'entreprise pour laquelle vous créez la campagne 10DLC et fournirez des détails supplémentaires.

Pour enregistrer une campagne 10DLC

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpoint à l’adresse https://console.aws.amazon.com/pinpoint/.

2. Sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet 10DLC campaigns (Campagnes 10DLC), choisissez Create a 10DLC campaign (Créer une campagne 10DLC).
4. Sur la page Create a 10DLC campaign (Créer une campagne 10DLC), dans la section Campaign info (Informations sur la campagne), procédez comme suit :

   a. Pour Nom de l'entreprise, choisissez l'entreprise pour laquelle vous créez cette campagne. Si vous n'avez pas encore enregistré l'entreprise, faites-le d'abord. Pour plus d'informations sur l'enregistrement d'une entreprise, consultez Enregistrement d'une entreprise.

   b. Dans 10DLC campaign name (Nom de la campagne 10DLC), attribuez un nom à la campagne.

   c. Pour Vertical, choisissez l'option qui représente le mieux votre entreprise.

   d. Dans Help message (Message d'aide), saisissez le message que vos clients reçoivent s'ils envoient le mot-clé « HELP » à votre numéro de téléphone 10DLC.

   e. Dans Stop message (Message d'arrêt), saisissez le message que vos clients reçoivent s'ils envoient le mot-clé « STOP » à votre numéro de téléphone 10DLC.

   > **Tip**
   >
   > Pour en savoir plus sur les messages qu'ils reçoivent de votre part, vos clients peuvent y répondre avec le mot « HELP ». Ils peuvent également répondre « STOP » pour refuser de recevoir des messages de votre part. Les opérateurs mobiles américains exigent que vous fournissiez des réponses à ces deux mots-clés.
   >
   > Voici un exemple de réponse HELP conforme aux exigences des opérateurs mobiles américains :
   >
   > **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
   >
   > Voici un exemple de réponse STOP conforme :
   >
   > **`You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.`**
   >
   > Vos réponses à ces mots-clés doivent contenir 160 caractères ou moins.

5. Dans la section Campaign use case (Cas d'utilisation de campagne), procédez comme suit :

a. Pour Use case type (Type de cas d'utilisation), si votre cas d'utilisation est lié à un organisme de bienfaisance, choisissez Special (Spécial). Sinon, choisissez Standard.

b. Pour Cas d'utilisation, choisissez un cas d'utilisation qui ressemble le plus à votre campagne dans la liste prédéfinie des cas d'utilisation. Les frais mensuels pour chaque cas d'utilisation apparaissent à côté du nom du cas d'utilisation.

> **Note**
>
> Les frais mensuels d'enregistrement de la campagne 10DLC sont affichés en regard de chaque type de cas d'utilisation. La plupart des types de campagnes 10DLC ont les mêmes frais mensuels. Les frais d'enregistrement des cas d'utilisation mixtes à faible volume sont inférieurs à ceux des autres types de cas d'utilisation. Toutefois, les campagnes mixtes à faible volume prennent en charge des débits inférieurs aux autres types de campagnes.

c. Saisissez au moins un exemple de message SMS. Voici l'exemple de message que vous prévoyez d'envoyer à vos clients. Si vous envisagez d'utiliser plusieurs modèles de messages pour cette campagne 10DLC, incluez-les également.

> **Important**
>
> N'utilisez pas d’espace réservé pour vos exemples de messages. Les exemples de messages que vous fournissez doivent refléter aussi précisément que possible les messages réels que vous envisagez d'envoyer.

6. La section Campaign and content attributes (Attributs de campagne et de contenu) contient une série de questions auxquelles il est possible de répondre par Oui ou par Non relatives aux caractéristiques particulières de la campagne. Certains attributs sont obligatoires, vous ne pouvez donc pas modifier la valeur par défaut.

   Assurez-vous que les attributs que vous choisissez sont appropriés à votre campagne.

   Indiquez si chacun des attributs suivants s'applique à la campagne que vous enregistrez :

   - Inclusion des abonnés – Les abonnés peuvent choisir de recevoir des messages relatifs à cette campagne.

- Exclusion des abonnés – Les abonnés peuvent refuser de recevoir les messages relatifs à cette campagne.
- Aide aux abonnés – Les abonnés peuvent contacter l'expéditeur du message après avoir envoyé le mot-clé HELP.
- Mise en commun des numéros – Cette campagne 10DLC utilise plus de 50 numéros de téléphone.
- Prêt direct ou accord de prêt – La campagne comprend des informations sur les prêts directs ou d'autres accords de prêt.
- Lien intégré – La campagne 10DLC comprend un lien intégré. Les liens générés par les raccourcisseurs d'URL courants, tels que TinyUrl ou Bit.ly, ne sont pas autorisés. Vous pouvez toutefois utiliser des raccourcisseurs d'URL qui proposent des domaines personnalisés.
- Numéro de téléphone intégré – La campagne inclut un numéro de téléphone intégré qui n'est pas un numéro de support client.
- Marketing d'affiliation – La campagne 10DLC inclut des informations provenant du marketing d'affiliation.
- Contenu classé par âge – La campagne 10DLC comprend du contenu classé par âge tel que défini par les lignes directrices de l'opérateur et de la Cellular Telecommunications and Internet Association (CTIA).

7. Sélectionnez Créer.

   Une fois que vous avez soumis les détails d'enregistrement de votre campagne, la page des messages SMS et vocaux s'ouvre. Un message s'affiche pour indiquer que votre campagne a été soumise et qu'elle est en cours d'examen. Vous pouvez voir le statut de votre demande sur l'onglet Campagnes DLC. Vous pouvez vérifier le statut de votre enregistrement sur l'onglet 10DLC, qui peut être l'un des éléments suivants :

   - Active – Votre campagne 10DLC a été approuvée. Vous pouvez demander un numéro de téléphone 10DLC et l'associer à votre campagne. Pour de plus amples informations, veuillez consulter Demande de numéros 10DLC, de numéros sans frais et de codes longs P2P pour la messagerie SMS avec Amazon SNS.
   - En suspens – Votre campagne 10DLC n'a pas encore été approuvée. Dans certains cas, l'approbation peut durer une semaine ou plus. S'il y a changement de statut, la console Amazon Pinpoint reflète ce changement. Nous ne vous informons pas des changements de statut.

- Rejetée – Votre campagne 10DLC a été rejetée. Pour obtenir plus d'informations, soumettez une demande de support qui inclut l'ID de campagne de la campagne rejetée.
- Suspendue – Un ou plusieurs opérateurs ont suspendu votre campagne 10DLC. Pour obtenir plus d'informations, soumettez une demande de support qui inclut l'ID de campagne de la campagne suspendue. Amazon Pinpoint n'inclut pas les motifs de la suspension sur la console. En outre, nous ne vous avertissons pas si votre campagne est suspendue.

8. Si votre 10DLC est approuvé, vous pouvez demander un numéro 10DLC à associer à cette campagne. Pour plus d'informations sur la façon de demander un numéro 10DLC, consultez Demande de numéros 10DLC, de numéros sans frais et de codes longs P2P pour la messagerie SMS avec Amazon SNS.

Utilisation de campagnes 10DLC dans plusieurs régions AWS

Lorsque vous enregistrez une entreprise, celle-ci est disponible dans votre compte AWS dans l’ensemble des régions AWS. Cependant, il n'en va pas de même pour les campagnes 10DLC. Une campagne 10DLC ne peut être utilisée que dans la région AWS dans laquelle elle a été enregistrée.

Si vous envisagez d'utiliser 10DLC dans plusieurs régions AWS, vous devez enregistrer séparément les campagnes 10DLC dans chacune de ces régions. Cette étape est nécessaire pour satisfaire aux exigences de l’opérateur. Vous êtes facturé pour chaque campagne que vous enregistrez, même si le cas d'utilisation est exactement le même.

L'enregistrement de plusieurs campagnes présente l'avantage supplémentaire d'augmenter vos débits pour les messages que vous envoyez aux destinataires qui utilisent AT&T comme opérateur mobile, car AT&T fournit des débits 10DLC pour chaque campagne. Comparez ceci à la façon dont T-Mobile gère le débit 10DLC, en fonction d’une allocation quotidienne de messages pour chaque entreprise (quel que soit le nombre de campagnes).

Modification ou suppression d'une campagne 10DLC

Vous pouvez modifier la réponse HELP, la réponse STOP et les exemples de messages pour une campagne 10DLC à l'aide de la console Amazon Pinpoint. Vous pouvez également supprimer des campagnes 10DLC à l'aide de la console.

Modification d'une campagne 10DLC

Une fois votre campagne approuvée, vous pouvez modifier les messages HELP, STOP et les exemples de messages. Vous pouvez également ajouter d'autres exemples de messages. Les modifications apportées à ces champs ne nécessitent pas de réapprobation de The Campaign

Registry ou des opérateurs. Vous ne pouvez pas modifier d'autres champs une fois la campagne 10DLC approuvée.

Vous pouvez avoir un maximum de cinq exemples de messages. Vous ne pouvez pas réduire le nombre d'exemples de messages que vous avez enregistrés à l'origine. Par exemple, si vous avez enregistré votre campagne avec trois exemples de messages SMS, vous ne pouvez pas réduire le nombre d'exemples de messages SMS à moins de trois.

> **Note**
>
> Si vous souhaitez modifier des champs autres que les messages HELP, STOP et les exemples de messages, vous devez commencer par supprimer la campagne 10DLC, puis recréer la campagne pour y inclure les informations à jour.

Pour modifier une campagne 10DLC

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpont à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet 10DLC campaigns (Campagnes 10DLC), choisissez la campagne 10DLC que vous souhaitez modifier.
4. Dans la section Campaign messages (Messages de campagne) de la page des détails de la campagne, choisissez Edit (Modifier).
5. Mettez à jour les champs suivants :

   - Help message (Message d'aide)
   - Stop message (Message d'arrêt)
   - Sample SMS message (Exemple de message SMS)

   Vous ne pouvez pas supprimer un exemple de message ajouté précédemment ou supprimer le contenu d'un exemple de message pour que le champ soit vide. Si vous supprimez le contenu d'un message sans le remplacer, le message d'origine sera utilisé lors de la mise à jour.
6. Sélectionnez Update (Mettre à jour). Une bannière de confirmation apparaît pour vous informer que les messages de campagne ont été mis à jour.

Suppression d'une campagne 10DLC

Vous pouvez supprimer une campagne 10DLC à l'aide de la console Amazon Pinpoint. Avant de supprimer une campagne 10DLC, vous devez supprimer tous les numéros de téléphone associés à cette campagne.

> **Important**
>
> Lorsque vous supprimez un numéro 10DLC d'une campagne, vous n'avez plus accès à ce numéro. En outre, les campagnes 10DLC supprimées ne peuvent pas être restaurées.

Pour supprimer une campagne 10DLC

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpont à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS and voice (Messages SMS et vocaux), choisissez Phone numbers (Numéros de téléphone).
3. Dans l'onglet 10DLC campaigns (Campagnes 10DLC), choisissez la campagne 10DLC que vous souhaitez modifier.
4. Dans la section Phone numbers (Numéros de téléphone), notez les numéros de téléphone associés à la campagne.
5. Dans l'onglet Phone numbers (Numéros de téléphone), choisissez le numéro 10DLC que vous voulez supprimer, puis choisissez Remove phone number (Supprimer le numéro de téléphone).

   > **Note**
   >
   > Cette étape n'est obligatoire que si plusieurs numéros de téléphone 10DLC sont associés à la campagne. Si vous n'avez qu'un seul numéro de téléphone associé à la campagne 10DLC, ce numéro apparaîtra dans l'onglet 10DLC campaigns (Campagnes 10DLC). Notez le numéro qui s'affiche dans l'onglet.

6. Saisissez **`delete`** dans la zone de confirmation, puis choisissez Confirm (Confirmer). Un message de réussite s'affiche en haut de la page des messages SMS et vocaux.
7. Répétez les deux étapes précédentes pour chaque numéro 10DLC associé à la campagne.
8. Une fois que vous avez supprimé tous les numéros associés à la campagne 10DLC, cliquez sur l'onglet 10DLC campaigns (Campagnes 10DLC).

9. Choisissez la campagne 10DLC que vous souhaitez supprimer.
10. Dans le coin supérieur droit de la page 10DLC campaign details (Détails de la campagne 10DLC), choisissez Delete (Supprimer).
11. Saisissez **`delete`** dans la zone de confirmation, puis choisissez Confirm (Confirmer). Un message de réussite s'affiche en haut de la page des messages SMS et vocaux.

Associer un code long à une campagne 10DLC

Si vous disposez d'un code long existant, vous pouvez l'associer à l'une de vos campagnes 10DLC actuelles en déposant une demande de support. Le code long que vous associez à la campagne 10DLC ne peut être utilisé qu'avec cette campagne et ne peut être utilisé pour aucune autre campagne 10DLC. Pendant que votre code long est migré vers 10DLC, vous serez toujours en mesure de l'utiliser. Cependant, tant que le code n'est pas approuvé, vous ne pourrez pas l'utiliser pour une campagne 10DLC.

Lors de la présentation de la demande, vous avez besoin des éléments suivants :

- Les longs codes à associer à une campagne 10DLC
- ID de campagne 10DLC à associer au code long

> **Note**
>
> Avant de pouvoir associer des codes longs à une campagne, vous devez enregistrer cette campagne 10DLC. Si vous n'avez pas encore créé et enregistré une campagne 10DLC, consultez Enregistrement d'une campagne 10DLC.

Pour affecter un code long à 10DLC

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/.
2. Sous Paramètres, puis sous SMS et services vocaux, choisissez l'onglet Numéros de téléphone.
3. Choisissez le code long à convertir en 10DLC.
4. Pour ouvrir le Centre de support, choisissez Affecter à la campagne 10DLC.
5. Pour le type de cas, choisissez Augmentation des limites de service.
6. Pour Type de limite, choisissez Pinpoint.

7. Dans la section Requests (Demandes), choisissez une région dans Region (Région), puis pour Limit (Limite), choisissez 10DLC - Associate existing US long code to 10DLC campaign (10DLC - Associer le code long américain existant à la campagne 10DLC).
8. Sous Description du cas, pour Description du cas d'utilisation, veillez à inclure l'ID de campagne 10DLC et les numéros de code longs que vous souhaitez associer à cette campagne. Vous pouvez inclure plusieurs codes longs dans la demande, mais vous ne devez inclure qu'un seul ID de campagne.
9. Sous Options de contact, pour Langue de contact préférée, choisissez la langue que vous préférez utiliser lors de la communication avec l'équipe de support AWS.
10. Pour Méthode de contact, choisissez le mode de communication que vous souhaitez utiliser pour échanger avec l'équipe de support AWS.
11. Choisissez Envoyer.

Accès entre comptes 10DLC

Chaque numéro de téléphone 10DLC est associé à un compte unique dans une seule région AWS. Si vous souhaitez utiliser le même numéro de téléphone 10DLC pour envoyer des messages dans plusieurs comptes ou régions, deux options s'offrent à vous :

1. Vous pouvez enregistrer la même entreprise et la même campagne dans chacun de vos comptes AWS. Ces enregistrements sont gérés et facturés séparément. Si vous enregistrez la même entreprise dans plusieurs comptes AWS, le nombre de messages quotidiens que vous pouvez envoyer aux clients T-Mobile est partagé entre chacun de ces comptes.
2. Vous pouvez terminer le processus d'enregistrement 10DLC dans un compte AWS, puis utiliser AWS Identity and Access Management (IAM) pour accorder à d'autres comptes l'autorisation d'envoyer des messages avec votre numéro 10DLC.

> **Note**
>
> Cette option permet un véritable accès entre comptes à vos numéros de téléphone 10DLC. Toutefois, notez que les messages envoyés depuis vos comptes secondaires sont traités comme s'ils avaient été envoyés depuis votre compte principal. Les quotas et la facturation sont imputés sur le compte principal et non sur les comptes secondaires.

Configuration d'un accès entre comptes à l'aide de stratégies IAM

Vous pouvez utiliser des rôles IAM pour associer d'autres comptes à votre compte principal. Vous pouvez ensuite déléguer les autorisations d'accès de votre compte principal à vos comptes secondaires en leur accordant l'accès aux numéros 10DLC du compte principal.

Pour accorder l'accès à un numéro 10DLC de votre compte principal

1. Si ce n'est pas déjà fait, terminez le processus d'enregistrement 10DLC dans le compte principal. Ce processus implique trois étapes :
   - Enregistrez votre entreprise. Pour plus d'informations, consultez Enregistrement de votre entreprise ou de votre marque pour utilisation avec 10DLC.
   - Enregistrez votre campagne 10DLC (cas d'utilisation). Pour plus d'informations, consultez Enregistrement d'une campagne 10DLC.
   - Associez un numéro de téléphone à votre campagne 10DLC. Pour plus d'informations, consultez Associer un code long à une campagne 10DLC.
2. Créez un rôle IAM dans votre compte principal qui permet à un autre compte d'appeler l'opération d'API `Publish` pour votre numéro de téléphone 10DLC. Pour plus d'informations sur la création de rôles, consultez Création de rôles IAM dans le Guide de l'utilisateur IAM.
3. Déléguez et testez l'autorisation d'accès à partir de votre compte principal à l'aide de rôles IAM avec n'importe quel autre compte qui doit utiliser vos numéros 10DLC. Par exemple, vous pouvez déléguer l'autorisation d'accès de votre compte de production à votre compte de développement. Pour plus d'informations sur la délégation et le test des autorisations, consultez Déléguer l'accès entre des comptes AWS à l'aide des rôles IAM dans le Guide de l'utilisateur IAM.
4. À l'aide du nouveau rôle, envoyez un message en utilisant un numéro 10DLC depuis le compte principal. Pour en savoir plus sur l'utilisation d'un rôle, consultez Utilisation des rôles IAM dans le Guide de l'utilisateur IAM.

Obtention d'informations sur les problèmes d'enregistrement 10DLC

Dans certains cas, vous pouvez recevoir un message d'erreur lorsque vous tentez d'enregistrer votre entreprise ou votre campagne 10DLC.

Problèmes d'enregistrement d'entreprise

Lorsque vous enregistrez votre entreprise, l'un des deux statuts d’enregistrement suivants s’affiche : Verified (Vérifié) ou Unverified (Non vérifié). Si le statut d'enregistrement de l'entreprise est Verified (Vérifié), l'enregistrement de votre entreprise a réussi. Vous pouvez commencer à créer des campagnes 10DLC.

Si le statut de l'enregistrement de votre entreprise est Unverified (Non vérifié), les informations que vous avez fournies étaient incorrectes. La console Amazon Pinpoint fournit des informations sur les raisons pour lesquelles l'enregistrement de votre entreprise a reçu ce statut.

Pour afficher les problèmes liés à l'enregistrement de votre entreprise 10DLC

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon Pinpoint à l’adresse https://console.aws.amazon.com/pinpoint/.
2. Dans le panneau de navigation, sous SMS, choisissez Numéros de téléphone.
3. Dans l’onglet 10DLC campaigns (Campagnes 10DLC), dans la liste des campagnes, choisissez le nom de l'entreprise pour laquelle vous souhaitez obtenir des informations.
4. La page détaillée de l'entreprise contient des informations sur les problèmes identifiés lors de votre enregistrement. Si un champ de la section Company info (Informations sur l’entreprise) contient un symbole d'avertissement, le problème d'enregistrement est lié aux informations contenues dans ce champ.

   Vérifiez les informations que vous avez fournies et confirmez que tous les champs contiennent des informations correctes. Vous pouvez modifier l'enregistrement de votre entreprise dans la console Amazon Pinpoint. Pour plus d'informations sur la modification des détails d’enregistrement de votre entreprise, consultez Modification ou suppression d'une entreprise enregistrée.

Problèmes d'enregistrement de campagne

Lorsque vous enregistrez votre campagne 10DLC, un message d'erreur peut s’afficher dans certaines situations.

Si vous ne parvenez pas à identifier le problème lié à votre enregistrement, vous pouvez créer un cas auprès du AWS Support Center pour demander des informations supplémentaires. Utilisez les procédures suivantes pour créer un cas AWS Support L’équipe AWS Support fournira des

informations sur les raisons pour lesquelles votre enregistrement de campagne 10DLC a été rejeté dans votre cas de support .

Pour soumettre une demande d'informations sur une campagne 10DLC rejetée

1. Connectez-vous à AWS Management Console via https://console.aws.amazon.com/.
2. Dans le menu Support, choisissez Centre de support.
3. Dans le panneau Vos cas de support, choisissez Créer un cas.
4. Cliquez sur le lien Vous cherchez à augmenter la limite de service ?, puis procédez comme suit :
   - Pour Type de limite, choisissez Pinpoint SMS.
5. Sous Requests (Demandes), complétez les sections suivantes :
   - Pour Région, choisissez la Région AWS dans laquelle vous avez tenté d'enregistrer la campagne.

     > **Note**
     >
     > La région est obligatoire dans la section Demandes. Même si vous avez fourni ces informations dans la section Détails du cas, vous devez également les inclure ici.

   - Pour Resource Type (Type de ressource), choisissez 10DLC Registration (Enregistrement 10DLC).
   - Pour Limite, choisissez Rejet de l'enregistrement d'une entreprise ou d'une campagne 10DLC.
6. Pour Nouvelle valeur limite, choisissez l'augmentation de limite pour le type de limite. Par défaut, cette valeur est **1**.
7. Sous Description du cas, saisissez l'ID de la campagne 10DLC rejetée.
8. (Facultatif) Si vous souhaitez soumettre d'autres demandes, choisissez Ajouter une autre demande. Si vous incluez plusieurs demandes, fournissez les informations requises pour chacune d'elles. Pour en savoir plus sur les informations requises, consultez les autres sections dans Demande de prise en charge de la messagerie SMS avec Amazon SNS.
9. Sous Options de contact, pour Langue de contact préférée, choisissez la langue dans laquelle vous souhaitez recevoir les communications pour ce cas.
10. Lorsque vous avez terminé, choisissez Submit (Soumettre).

Numéros gratuits

Un numéro sans frais est un numéro à 10 chiffres commençant par l'un des indicatifs régionaux suivants : 800, 888, 877, 866, 855, 844 ou 833. Vous pouvez utiliser des numéros sans frais pour envoyer des messages transactionnels uniquement.

Important

Les opérateurs mobiles américains ont récemment modifié leur réglementation et exigeront que tous les numéros gratuits soient enregistrés auprès d'un organisme de réglementation avant le 30 septembre 2022. Vérifiez le statut de votre TFN en vous rendant sur the section called “Statut d’enregistrement d’un numéro gratuit”. Pour plus d'informations sur l'enregistrement de votre entreprise, consultez the section called “Enregistrement de votre numéro gratuit”.

Le traitement de votre enregistrement peut prendre jusqu'à 15 jours ouvrés après sa soumission.

Mise à jour du 3 mars 2023 : à compter du 1er avril 2023, les opérateurs de téléphonie mobile appliqueront les seuils suivants à l'échelle du secteur pour les messages envoyés via n'importe quel numéro gratuit non enregistré :

- Limite quotidienne : 500 messages par jour, réinitialisation à 0 h, heure PT
- Limite hebdomadaire : 1 000 messages, réinitialisation le dimanche à 0 h
- Limite mensuelle : 2 000 messages, réinitialisation à la fin du mois civil à 0 h

Mise à jour le 19 septembre 2022 : à compter du 1er octobre 2022, les opérateurs de téléphonie mobile appliqueront les seuils suivants à l'échelle du secteur pour les messages envoyés via n'importe quel numéro gratuit non enregistré :

- Limite quotidienne : 2 000 messages
- Limite quotidienne : 12 000 messages
- Limite mensuelle : 25 000 messages

Nous vous encourageons vivement à effectuer votre inscription dès que possible. Les messages envoyés via des numéros gratuits non enregistrés seront traités dans la mesure du possible. Les messages seront soumis à un filtrage et à un blocage accrus au fil du temps, les opérateurs continuant à restreindre le trafic non enregistré.

Rubriques

Lignes directrices pour l'utilisation des numéros sans frais

Les numéros gratuits sont généralement utilisés pour les messages transactionnels, tels que la confirmation d'enregistrement ou pour l'envoi de mots de passe à usage unique. Ils peuvent être utilisés à la fois pour les messages vocaux et SMS. Le débit moyen est de trois parties de message par seconde (MPS). Cependant, ce débit est affecté par le codage de caractères. Pour plus d'informations sur la façon dont l'encodage des caractères affecte les parties du message, consultez Limites de caractères des SMS dans Amazon SNS. Pour plus d'informations sur l'enregistrement d'un numéro gratuit, consultez Exigences relatives à l'enregistrement d'un numéro gratuit et processus d'enregistrement.

Chaque compte client peut avoir jusqu'à cinq numéros gratuits. Si vous envoyez entre 15 et 100 SMS par seconde, nous vous recommandons d'enregistrer un ou plusieurs ID d'origine 10DLC. Si vos cas d'utilisation exigent l'envoi de plus de 100 SMS par seconde, nous vous recommandons d'acheter et d'enregistrer un ou plusieurs codes courts.

Lorsque vous utilisez un numéro gratuit comme numéro d'origine, suivez les instructions suivantes :

- N'utilisez pas d'URL raccourcies créées à partir de raccourcisseurs d'URL tiers, car ces messages sont plus susceptibles d'être marqués comme courriers indésirables.

  Si vous devez utiliser une URL raccourcie, envisagez d'utiliser unnuméro 10DLC ou un code court. L'utilisation de codes courts et de 10DLC nécessite que vous enregistriez votre modèle de message, où vous pouvez spécifier une URL raccourcie.
- Sachez que les mots-clés des réponses d'exclusion (STOP) et d'inclusion (UNSTOP) sont définies au niveau de l'opérateur. Vous ne pouvez pas modifier ces mots-clés ou d'autres mots-clés. Vous

ne pouvez pas non plus modifier les messages envoyés lorsque les utilisateurs répondent avec STOP et UNSTOP.

- N'envoyez pas le même contenu de message ou un contenu similaire à l'aide de plusieurs numéros gratuits. Les opérateurs appellent cette pratique raquettes ou mise en commun de numéros et ciblent ces messages pour le filtrage.
- Tous les messages associés aux industries suivantes peuvent être considérés comme restreints et font l'objet d'un filtrage intensif ou d'un blocage pur et simple. Cela peut inclure des mots de passe uniques (OTP) et l'authentification multifactorielle (MFA) pour les services liés à des catégories restreintes.

  Si votre enregistrement a été refusé parce qu'il s'agit d'un cas d'utilisation non conforme et que vous pensez que cette désignation est incorrecte, vous pouvez envoyer une demande via le support. Pour plus d'informations sur la manière de procéder, consultez Problèmes d'inscription.

  Le tableau suivant décrit les types de contenu restreint :

Catégorie	Exemples
Jeu	• Applications/sites Web • Casinos • Loteries promotionnelles
Services financiers à haut risque	• Prêts automobiles • Cryptomonnaie • Recouvrement des créances • Prêts sur salaire • Prêts à court terme et à taux d’intérêt élevé • Prêts hypothécaires • Prêts étudiants • Alertes boursières
Annulation de la dette	• Consolidation des dettes • Réduction de la dette • Programmes de réparation de crédits

Catégorie	Exemples
et-rich-quick Schémas G	• ork-from-home Programmes W • Opportunités d'investissement à faible risque • Schémas de marketing pyramidaux ou multi-niveaux
Substances interdites/contrôlées	• Cannabis/CBD
Hameçonnage	• Tentatives visant à amener les utilisateurs à révéler des informations personnelles ou des informations de connexion au site Web
S.H.A.F.T.	• Sexe • Haine • Alcool • Armes à feu • Tabac/vapotage

Acheter un numéro gratuit

Pour acheter des TFN, utilisez la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/sms-voice/. Pour de plus amples informations, veuillez consulter Exigences relatives à l'enregistrement d'un numéro gratuit et processus d'enregistrement.

Actuellement, Amazon Pinpoint SMS prend en charge les numéros gratuits pour les messages vocaux et SMS. Amazon SNS prend uniquement en charge la messagerie SMS.

Exigences relatives à l'enregistrement d'un numéro gratuit et processus d'enregistrement

> **Important**
>
> Si un numéro gratuit est utilisé pour autre chose que le cas d'utilisation spécifié, il peut être révoqué.

Cas d’utilisation interdits d’un numéro gratuit

Amazon SNS a une capacité limitée à envoyer des messages dans les cas où les messages sont bloqués (par exemple, cas d'utilisation liés à une substance contrôlée ou au phishing) ou lorsque des niveaux de filtrage élevés sont attendus (par exemple, des messages financiers à haut risque). Il se peut que vous ne puissiez pas enregistrer les TFN associés aux cas d'utilisation de contenus restreints définis dans Lignes directrices pour l'utilisation des numéros sans frais.

Enregistrement de votre numéro gratuit

Après avoir acheté un TFN, vous devez enregistrer le numéro. Pour savoir comment procéder, consultez la procédure d'enregistrement d'un numéro gratuit dans le guide de l'utilisateur Amazon Pinpoint SMS.

Enregistrement en libre-service pour les numéros gratuits dans les régions où Amazon Pinpoint envoie des SMS

Si vous avez demandé le TFN dans les régions Amazon Pinpoint SMS, complétez le processus d'enregistrement de l'entreprise directement dans la console SMS Amazon Pinpoint en suivant les instructions figurant sur le formulaire d'enregistrement du numéro gratuit américain figurant dans le guide de l'utilisateur Amazon Pinpoint SMS.

Lorsque vous enregistrez votre numéro gratuit, assurez-vous que les informations sont complètes et exactes, car votre enregistrement peut être refusé dans le cas contraire. Les informations que vous saisissez doivent correspondre exactement au siège social de votre entreprise.

Processus d'enregistrement manuel basé sur un formulaire pour les numéros gratuits situés dans des régions autres que les régions Amazon Pinpoint SMS

1. Téléchargez ce fichier US_TFN_Registration.zip et utilisez l'exemple de formulaire d'inscription (AWS US Toll-Free Registration Form-Business - Final.docx) pour compléter les informations requises dans le fichier CSV d'enregistrement TFN (BulkustFN - Final.csv).

   Chaque demande d'enregistrement ou chaque cas d'utilisation peut uniquement avoir jusqu'à cinq numéros gratuits. Si vous pensez pouvoir bénéficier d'une exemption à cette règle, veuillez fournir une explication détaillée pour examen. Répertoriez tous les numéros de téléphone associés à l'enregistrement ou au cas d'utilisation.
2. Créez un cas auprès d’AWS Support. Joignez votre fichier CSV complété au dossier et envoyez la demande d'enregistrement du numéro gratuit.

3. Choisissez Créer un cas, puis Vous cherchez à augmenter la limite de service ?
4. Pour Type de limite, choisissez SMS SNS.
5. Pour Resource Type (Type de ressource), choisissez 10DLC or Toll-free number registration (Enregistrement 10DLC ou d'un numéro gratuit).
6. Joignez le document US_TFN_registration et soumettez la demande.

Point clé à noter

1. Le traitement des inscriptions peut prendre jusqu'à deux semaines une fois que toutes les informations requises ont été soumises. Si des informations sont manquantes ou incomplètes, le processus d'enregistrement sera retardé. Si votre inscription est refusée, nous vous aiderons à trouver la raison pour laquelle elle a été refusée et nous vous proposerons des méthodes pour améliorer votre campagne afin qu'elle puisse être enregistrée.
2. Les numéros gratuits fonctionnent bien pour les cas d'utilisation transactionnels tels que l'authentification multifactorielle (MFA) où un débit limité est requis. Chaque numéro gratuit peut envoyer jusqu'à trois parties de SMS par seconde, et chaque compte client peut avoir jusqu'à cinq numéros gratuits. Si vous envoyez entre 15 et 100 SMS par seconde, nous vous recommandons d'enregistrer un ou plusieurs ID d'origine 10DLC. Si vos cas d'utilisation exigent l'envoi de plus de 100 SMS par seconde, nous vous recommandons d'acheter et d'enregistrer un ou plusieurs codes courts. Pour en savoir plus, consultez Lignes directrices pour l'utilisation des numéros sans frais.

Statut d’enregistrement d’un numéro gratuit

Pour vérifier le statut de votre inscription, consultez Vérifier le statut de votre inscription dans le guide de l'utilisateur Amazon Pinpoint SMS.

Modifier, supprimer et supprimer votre enregistrement

Utilisez le guide de l'utilisateur Amazon Pinpoint SMS pour effectuer les tâches suivantes :

- Modifiez votre inscription
- Annuler votre inscription
- Supprimer votre inscription
- Consultez vos ressources d'inscription

Problèmes d'inscription

Si l'enregistrement de votre numéro gratuit n'est pas accepté, un message explique pourquoi il a été refusé.

Pour soumettre une demande d'informations sur un numéro gratuit rejeté

1. Connectez-vous à l' AWS Management Console adresse https://console.aws.amazon.com/.
2. Dans le menu Support, choisissez Centre de support.
3. Dans le panneau Vos cas de support, choisissez Créer un cas.
4. Cliquez sur le lien Vous cherchez à augmenter la limite de service ?, puis procédez comme suit :
   - Pour Limit type (Type de limite), choisissez Pinpoint SMS (SMS Pinpoint).
5. Sous Requests (Demandes), complétez les sections suivantes :
   - Pour Région, choisissez la région dans laquelle vous avez tenté d'enregistrer la campagne.

     > **Note**
     >
     > La région est obligatoire dans la section Demandes. Même si vous avez fourni ces informations dans la section Détails du cas, vous devez également les inclure ici.

   - Pour Resource Type (Type de ressource), choisissez 10DLC or TFN Registration (Enregistrement 10DLC ou TFN).
   - Pour Limite, choisissez Rejet de l'enregistrement d'une entreprise ou d'une campagne.
6. Pour Nouvelle valeur limite, choisissez l'augmentation de limite pour le type de limite. Par défaut, cette valeur est **1**.
7. (Facultatif) Si vous souhaitez soumettre d'autres demandes, choisissez Ajouter une autre demande. Pour en savoir plus sur les informations requises, consultez les autres sections dans Demande de prise en charge de la messagerie SMS avec Amazon SNS.
8. Pour Description du cas d'utilisation, saisissez le numéro gratuit refusé.
9. Sous Options de contact, pour Langue de contact préférée, choisissez la langue dans laquelle vous souhaitez recevoir les communications pour ce cas.
10. Lorsque vous avez terminé, choisissez Submit (Soumettre).

Questions fréquentes sur les numéros gratuits

Questions fréquentes sur le processus d'enregistrement des numéros gratuits.

Est-ce que je possède actuellement un numéro gratuit ?

Pour vérifier si vous possédez un numéro gratuit

- Ouvrez la console SMS Amazon Pinpoint à l'adresse https://console.aws.amazon.com/sms-voice/.
- Dans le panneau de navigation, sous SMS, choisissez Numéros de téléphone.
- Le type de numéro gratuit est répertorié comme gratuit.

Dois-je enregistrer mon numéro gratuit ?

Oui. Pour continuer à utiliser un numéro gratuit que vous possédez actuellement, vous devez l'enregistrer avant le 30 septembre 2022. Si vous achetez un nouveau numéro gratuit après le 30 septembre 2022, vous devez l'enregistrer pour pouvoir envoyer des messages.

Comment acheter un numéro gratuit ?

Suivez les instructions de la section Demande d'un numéro de téléphone à l'aide de la console SMS Amazon Pinpoint pour acheter un TFN.

Comment enregistrer mon numéro gratuit ?

Suivez les indications de the section called “Enregistrement de votre numéro gratuit” pour enregistrer un numéro gratuit.

Quel est le statut d’enregistrement de mon numéro gratuit et que signifie-t-il ?

Suivez les instructions de la rubrique the section called “Statut d’enregistrement d’un numéro gratuit” pour vérifier votre enregistrement et son statut.

Quelles informations dois-je fournir ?

Vous devrez fournir l'adresse de votre entreprise, un contact professionnel et le cas d'utilisation du numéro gratuit. Les informations requises sont disponibles à la rubrique the section called “Enregistrement de votre numéro gratuit”.

Que se passe-t-il si mon enregistrement est refusé ?

Si votre inscription est refusée , le statut passe àExige des mises à jour. Pour effectuer des mises à jour, consultez the section called “Modifier, supprimer et supprimer votre enregistrement”.

De quelles autorisations ai-je besoin ?

L'utilisateur/le rôle IAM que vous utilisez pour accéder à la console SMS Amazon Pinpoint doit disposer de l'autorisation *« sms-voice : * »*, sinon vous recevrez un message d'erreur de refus d'accès.

Avantages et les inconvénients des numéros gratuits

Avantages

Les créateurs de numéros gratuits ont un débit d’envoi de messages par seconde plus élevé que les codes longs, ainsi qu’une bonne délivrabilité.

Inconvénients

Les consentements et les refus ne sont pas contrôlés car ils sont gérés au niveau de l'opérateur.

Vous ne devez ni inclure d'URL raccourcies dans votre message, ni utiliser le numéro pour envoyer un message promotionnel. Utilisez plutôt un numéro 10DLC ou un code court. Lorsque vous utilisez un code court ou un numéro 10DLC, vous devez enregistrer vos modèles de message, qui peuvent contenir des URL raccourcies et peuvent être des messages promotionnels. Pour en savoir plus sur les codes courts, veuillez consulter Codes courts. Pour de plus amples informations sur 10DLC, veuillez consulter 10DLC.

Codes courts

Les codes courts sont des séquences numériques plus courtes qu'un numéro de téléphone normal. Par exemple, aux États-Unis et au Canada, les numéros de téléphone standard (codes longs) contiennent 11 chiffres, tandis que les codes courts contiennent 5 ou 6 chiffres. Amazon SNS prend en charge les codes courts dédiés.

Codes courts dédiés

Si vous envoyez un large volume de messages SMS aux destinataires aux États-Unis ou au Canada, vous pouvez acheter un code court dédié. Contrairement aux codes courts du groupe partagé, les codes courts dédiés sont réservés à votre usage exclusif.

Avantages

L'utilisation d'un code court facile à se souvenir peut contribuer à établir la confiance. Si vous avez besoin d'envoyer des informations sensibles, telles que des mots de passe ponctuels, il est conseillé

de les adresser à l'aide d'un code court afin que votre client puisse rapidement déterminer si le message provient réellement de vous.

Si vous exécutez une nouvelle campagne de prospection de clients, vous pouvez inviter les clients potentiels à envoyer un mot-clé à votre code court (par exemple, « Envoyez FOOTBALL au 10987 pour recevoir les actualités et informations du football »). Les codes courts sont plus faciles à mémoriser que les codes longs, et il est plus facile pour les clients de saisir des codes courts sur leurs appareils. En réduisant la quantité des difficultés que les clients rencontrent lorsqu'ils s'inscrivent à vos programmes marketing, vous pouvez augmenter l'efficacité de vos campagnes.

Étant donné que les opérateurs mobiles doivent approuver les nouveaux codes courts avant de les rendre actifs, ils sont moins susceptibles de marquer les messages envoyés à partir de codes courts comme indésirables.

Lorsque vous utilisez des codes courts pour envoyer des messages SMS, vous pouvez envoyer un plus grand volume de messages par période de 24 heures que vous ne le pouvez lorsque vous utilisez d'autres types de provenance d'origine des identités. En d'autres termes, vous disposez d'un quota d'envoi beaucoup plus élevé. Vous pouvez également envoyer un plus grand volume de messages par seconde. En d'autres termes, vous disposez d'un taux d'envoi beaucoup plus élevé.

Inconvénients

Il existe des coûts supplémentaires pour acquérir des codes courts, et ils peuvent prendre beaucoup de temps pour être mis en œuvre. Par exemple, aux États-Unis, il y a un droit d'installation unique de 650.00 USD pour chaque code court, plus un coût mensuel de 995.00 USD pour chaque code court. Cette opération peut prendre entre 8 et 12 semaines avant que les codes courts ne soient actifs sur tous les réseaux des opérateurs. Pour connaître le prix et la durée de provisionnement d'un pays ou d'une région différents, suivez la procédure décrite à la section Demande de codes courts dédiés pour la messagerie SMS avec Amazon SNS .

Codes longs de personne à personne (P2P)

> **Important**
>
> À compter du 31 août 2023, un numéro dédié, tel qu'un numéro 10DLC ou un numéro gratuit est requis pour envoyer des SMS aux États-Unis et dans leurs territoires (Porto Rico, Guam, îles Samoa américaines et îles Vierges américaines). Votre demande de code long sera rejetée si vous utilisez les États-Unis comme emplacement pour ces régions.

> **Important**
>
> À compter du 1er juin 2021, les fournisseurs de télécommunications américains ne prennent plus en charge l'utilisation de codes longs de personne à personne (P2P) pour les communications d'application à personne (A2P) vers des destinations américaines. Au lieu de cela, vous devez utiliser un autre type d'ID d'origine pour ces messages. Pour de plus amples informations, veuillez consulter 10DLC.

Les codes longs P2P sont des numéros de téléphone qui utilisent le format numérique du pays ou de la région où vos destinataires sont situés. Les codes longs P2P sont également appelés numéros longs ou numéros mobiles virtuels. Par exemple, aux États-Unis et au Canada, les codes longs P2P contiennent 11 chiffres : le chiffre 1 (le code du pays), un préfixe à 3 chiffres et un numéro de téléphone à 7 chiffres.

Pour plus d'informations sur la demande de codes longs P2P, consultez Demande de numéros 10DLC, de numéros sans frais et de codes longs P2P pour la messagerie SMS avec Amazon SNS.

Avantages

Les codes longs P2P dédiés sont réservés à une utilisation par votre compte Amazon SNS uniquement. Ils ne sont pas partagés avec les autres utilisateurs. Lorsque vous utilisez des codes longs P2P dédiés, vous pouvez spécifier le code long P2P que vous souhaitez utiliser lors de l'envoi de chaque message. Si vous envoyez plusieurs messages au même client, vous pouvez vous assurer que chaque message est bien envoyé depuis le même numéro de téléphone. Pour cette raison, les codes longs P2P dédiés peuvent être utiles dans l'établissement de votre marque ou de votre identité.

Inconvénients

Les codes longs P2P ne sont pas pris en charge pour les communications A2P vers des destinations américaines.

Si vous envoyez plusieurs centaines de messages par jour à partir d'un code long P2P dédié, les opérateurs mobiles peuvent identifier votre numéro comme celui qui envoie des messages non sollicités. Si votre code long P2P est signalé, vos messages peuvent ne pas être transmis à vos destinataires.

Les codes longs P2P ont aussi un débit limité. Les tarifs d'envois maximaux varient selon les pays. Pour plus d'informations, consultez Support AWS. Si vous prévoyez d'envoyer un grand nombre de messages SMS, ou que vous prévoyez d'en envoyer à un rythme supérieur à celui d'un message par seconde, vous devez acheter un code court dédié.

Certains opérateurs ne vous permettent pas d'utiliser des codes longs P2P pour envoyer des messages SMS A2P, y compris aux États-Unis. Un message SMS A2P est un message envoyé à l'appareil mobile d'un client lorsque ce client remet son numéro de mobile à une application. Les messages A2P sont des conversations unidirectionnelles, telles que les messages marketing, les mots de passe uniques et les rappels de rendez-vous. Si vous prévoyez d'envoyer des messages A2P, vous devez acheter un code court dédié (si vos clients sont situés aux États-Unis ou au Canada), ou utiliser un ID d'expéditeur (si vos destinataires se trouvent dans un pays ou une région où les ID d'expéditeur sont pris en charge).

Un numéro 10DLC est utilisé uniquement pour envoyer des messages sur le territoire américain. L'utilisation d'un numéro 10DLC exige d'enregistrer la marque de votre entreprise et la campagne à laquelle vous souhaitez associer ce numéro. Un fois l'enregistrement approuvé, vous pouvez demander un numéro de téléphone 10DLC sur la page SMS and voice (Messages SMS et vocaux) de la console Amazon Pinpoint à l'adresse https://console.aws.amazon.com/pinpoint/. Une fois la demande effectuée, le délai de réception de l'approbation est de 7 à 10 jours. Le numéro ne peut être utilisé avec aucune autre campagne.

Comparaison des numéros de produits américains

Ce tableau montre la comparaison du support pour les types de numéros de téléphone américains.

Caractéristique du produit	Code court	Numéro gratuit	10DLC	
Format de nombre	5 à 6 chiffres	Numéro à 10 chiffres	Numéro à 10 chiffres	
Canal de support	SMS	SMS	SMS	
Type de trafic SMS	Promotionnel et transactionnel	Transactionnel	Promotionnel et transactionnel	
Nécessite une évaluation	Oui	Non	Oui	

Caractéristique du produit	Code court	Numéro gratuit	10DLC	
Temps d'approvisionnement estimé	12 semaines[1]	15 jours ouvrés	1 semaine	
Débit SMS (nombre de messages SMS par seconde) [2]	100 parties de messages par seconde ; débit plus élevé disponible moyennant des frais supplémentaires.	3 parties de message par seconde	Varie en fonction de votre enregistrement 10DLC. Prend en charge jusqu'à 100 parties de message par seconde.	
Mots-clés requis	Opt-in, opt-out et HELP	STOP, UNSTOP. Ceux-ci sont gérés par le réseau. Vous ne pouvez pas modifier l'exclusion et l'inclusion dans les messages.	Opt-in, opt-out et HELP	

[1] L'estimation de l'approvisionnement n'inclut pas le temps d'approbation.

[2] Pour plus d'informations sur la taille maximale des messages SMS, consultez Publication sur un téléphone mobile.

Demande de prise en charge de la messagerie SMS avec Amazon SNS

Certaines options SMS avec Amazon SNS ne sont pas disponibles pour votre compte AWS jusqu'à ce que vous contactiez AWS Support. Créez un cas dans le AWS Support Centre pour formuler une demande relative à une des opérations suivantes :

- Une augmentation du seuil de vos dépenses mensuelles pour l'envoi de SMS

  Par défaut, le seuil des dépenses mensuelles est fixé à 1 USD. Votre seuil de dépenses détermine le volume de messages que vous pouvez envoyer avec Amazon SNS. Vous pouvez demander un seuil de dépenses correspondant au volume mensuel de messages prévu pour votre cas d'utilisation des SMS.
- Un déplacement de l'environnement de test (sandbox) pour SMS afin que vous puissiez envoyer des messages SMS sans restrictions. Pour de plus amples informations, veuillez consulter Déplacement de l'environnement de test (sandbox) pour SMS.
- Un numéro d'origine dédié
- Un ID expéditeur dédié

  Un ID expéditeur est un ID personnalisé qui s'affiche en tant qu'expéditeur sur l'appareil du destinataire. Par exemple, vous pouvez utiliser votre marque d'entreprise pour faciliter la reconnaissance de la source du message. La prise en charge des ID expéditeur varie selon les pays ou les régions. Pour de plus amples informations, veuillez consulter Pays et régions pris en charge.

Lorsque vous créez votre cas dans le Centre AWS Support, veillez à inclure toutes les informations requises pour le type de demande que vous soumettez. Dans le cas contraire, AWS Support doit vous contacter pour obtenir ces informations avant de continuer. En soumettant un cas détaillé, vous contribuez à ce qu'il soit satisfait sans retard. Pour les détails requis pour des types spécifiques de demande SMS, consultez les rubriques suivantes.

Rubriques

- Demande de codes courts dédiés pour la messagerie SMS avec Amazon SNS
- Demande de numéros 10DLC, de numéros sans frais et de codes longs P2P pour la messagerie SMS avec Amazon SNS
- Demande d’ID expéditeur pour les SMS avec Amazon SNS
- Demande d’augmentations de votre quota de dépenses mensuelles pour l’envoi de SMS pour Amazon SNS

Demande de codes courts dédiés pour la messagerie SMS avec Amazon SNS

Un code court est un nombre que vous pouvez utiliser pour l'envoi d'un volume élevé de messages SMS. Les codes courts sont souvent utilisés pour les messages A2P (envoyés par une

application à une personne), l'authentification à deux facteurs (2FA) et le marketing. Un code court contient généralement entre 3 et 7 chiffres, selon le pays ou la région dans lequel il se trouve.

Vous pouvez uniquement utiliser des codes courts pour envoyer des messages aux destinataires situés dans le pays où le code court concerné est utilisé. Si votre cas d'utilisation requiert le recours à des codes courts dans plusieurs pays, vous devez demander un code court distinct pour chaque pays dans lequel se trouvent vos destinataires.

Pour de plus amples informations sur la tarification des codes courts, consultez Tarification Amazon SNS.

> **Important**
>
> Si vous n'êtes pas encore familiarisé avec l'envoi de messagerie SMS avec Amazon SNS, demandez un seuil de dépenses mensuel pour l'envoi de SMS qui réponde à votre cas d'utilisation prévu des SMS. Par défaut, votre seuil de dépenses mensuel est fixé à 1 USD. Vous pouvez demander une augmentation de votre seuil de dépenses dans le dossier de support incluant votre demande de code court. Vous pouvez également utiliser un dossier distinct. Pour de plus amples informations, veuillez consulter Demande d'augmentations de votre quota de dépenses mensuelles pour l'envoi de SMS pour Amazon SNS.
> De plus, si vous demandez un code court dédié pour envoyer des messages contenant ou pouvant contenir des données de santé protégées, vous devez identifier cet usage dans votre description de cas lorsque vous ouvrez un dossier de support, comme indiqué ci-dessous.

Ouverture d'un cas de support à code court Amazon SNS

Ouvrez un cas avec AWS Support en effectuant les étapes suivantes.

> **Note**
>
> Certains champs du formulaire de demande sont marqués comme « facultatifs ». Toutefois, AWS Support nécessite toutes les informations mentionnées dans les étapes suivantes afin de traiter votre demande. Si vous ne fournissez pas toutes les informations requises, le traitement de votre demande peut être retardé.

Pour demander un code court dédié

1. Accédez au AWSCentre de support.
2. Connectez-vous à AWS Management Console via https://console.aws.amazon.com/.
3. Dans le menu Support, choisissez Centre de support.
4. Dans le panneau Vos cas de support, choisissez Créer un cas.
5. Cliquez sur le lien Vous cherchez à augmenter la limite de service ?, puis procédez comme suit :

   - Pour Type de limite, choisissez SMS SNS, puis procédez comme suit :
   - (Facultatif) Pour Fournir un lien vers le site ou l'application qui enverra les SMS, fournissez un lien vers le site ou le nom de l'application où les membres de votre public accepteront de recevoir vos SMS.
   - (Facultatif) Pour Quel type de message envisagez-vous d'envoyer, choisissez le type de message que vous prévoyez d'envoyer à l'aide de votre code long :
     - Mot de passe ponctuel – Messages fournissant des mots de passe que vos clients utilisent pour s'authentifier sur votre site web ou votre application.
     - Promotionnel – Messages non stratégiques faisant la promotion de votre entreprise ou de votre service, comme des offres spéciales ou des annonces.
     - Transactionnel – Messages d'information importants qui prennent en charge les transactions clients, comme des confirmations de commande ou des alertes de compte. Les messages transactionnels ne peuvent pas comporter de contenu promotionnel ni marketing.
   - (Facultatif) Pour À partir de quelle région AWS allez-vous envoyer des messages, choisissez la région à partir de laquelle vous allez envoyer des messages.
   - (Facultatif) Pour Vers quels pays prévoyez-vous d'envoyer des messages, saisissez les pays ou les régions vers lesquels vous prévoyez d'envoyer des SMS.
   - (Facultatif) Dans Comment vos clients acceptent-ils de recevoir des messages de votre part, saisissez une description de la façon dont vos clients acceptent de recevoir des messages de votre part.
   - (Facultatif) Dans le champ Veuillez fournir le modèle de message que vous comptez utiliser pour envoyer des messages à vos clients, incluez le modèle que vous allez utiliser.
6. Sous Requests (Demandes), complétez les sections suivantes :

   - Pour Région, choisissez la Région AWS pour votre demande de code court.

> **Note**
>
> La région est obligatoire dans la section Demandes. Même si vous avez fourni ces informations dans la section Détails du cas, vous devez également les inclure ici.

- Pour Type de ressources, sélectionnez Codes courts SMS dédiés.
- Pour Limite, sélectionnez l'option la plus proche de votre cas d'utilisation.

7. Pour Valeur de la nouvelle limite, choisissez le nombre d'ID expéditeur que vous demandez. Par défaut, cette valeur est **1**.
8. (Facultatif) Si vous souhaitez soumettre d'autres demandes, choisissez Ajouter une autre demande. Pour en savoir plus sur les informations requises, consultez les autres sections dans Demande de prise en charge de la messagerie SMS avec Amazon SNS.
9. Sous Description du cas, résumez votre cas d'utilisation, indiquez la façon dont vos destinataires s'abonneront aux messages envoyés avec votre code court et fournissez les informations suivantes :

   - Informations sur la société :
     - Nom de la société
     - L'adresse de messagerie de la société
     - Le nom et le numéro de téléphone du contact principal pour votre demande
     - L'adresse électronique et le numéro d'appel gratuit dans votre entreprise pour la prise en charge
     - L'identifiant fiscal de la société
     - Le nom de votre produit ou service

   - Processus de connexion des utilisateurs :
     - Le site web de l'entreprise, ou le site Web sur lequel vos clients devront s'inscrire pour recevoir des messages en provenance de votre code court.
     - Comment les utilisateurs devront-ils s'inscrire pour recevoir des messages en provenance de votre code court. Spécifiez une ou plusieurs des options suivantes :
       - **`Text messages`**
       - **`Website`**

    - **Mobile app**
    - **Other** Sinon, fournissez une explication.
  - Le texte pour l'option permettant de s'inscrire aux messages sur votre site Web, votre application ou ailleurs.
  - La séquence de messages que vous avez l'intention d'utiliser pour la confirmation de l'acceptation. Fournissez toutes les informations suivantes :
    1. Le SMS que vous avez l'intention d'envoyer lors de l'inscription d'un utilisateur. Ce message demande à l'utilisateur s'il accepte les messages récurrents. Par exemple : ExampleCorp : Répondez OUI pour recevoir des alertes de transaction de compte. Des tarifs de message et de données peuvent s'appliquer.
    2. La réponse d'acceptation que vous attendez de la part de l'utilisateur. Il s'agit généralement d'un mot-clé, par exemple OUI.
    3. Le message de confirmation que vous souhaitez envoyer lorsque les clients envoient ce mot-clé à votre code court. Par exemple : Vous êtes désormais inscrit aux alertes de compte de ExampleCorp. Des tarifs de message et de données peuvent s'appliquer. STOP pour annuler ou AIDE pour plus d'informations.

- L'objectif de vos messages :
  - L'objectif des messages que vous avez l'intention d'envoyer avec votre code court. Spécifiez l'une des options suivantes :
    - **Promotions and marketing**
    - **Location-based services**
    - **Notifications**
    - **Information on demand**
    - **Group chat**
    - **Two-factor authentication (2FA)**
    - **Polling and surveys**
    - **Sweepstakes or contests**
    - **Other** Sinon, fournissez une explication.
  - Indiquez si vous avez l'intention d'utiliser votre code court afin d'envoyer des messages promotionnels ou marketing pour une entreprise autre que la vôtre.

- Si vous prévoyez d'utiliser votre code court pour envoyer des messages contenant ou pouvant contenir des données de santé protégées, tels que définis par le Health Insurance Portability and Accountability Act (HIPAA) et les lois et règlements connexes.

- Contenu des messages:
  - Message que vous avez l'intention d'envoyer lorsque des clients s'inscrivent pour recevoir vos messages en envoyant un mot-clé spécifique. Soyez vigilant lorsque vous spécifiez ce mot-clé et ce message, toute modification du message peut prendre plusieurs semaines. Lorsque nous créons votre code court, nous enregistrons le mot-clé et le message auprès des opérateurs de téléphonie mobile du pays dans lequel vous utilisez le code court. Votre message peut ressembler à l'exemple suivant : Bienvenue dans les alertes de *ProductName* ! Les tarifs Msg&data s'appliquent. *2* msgs par mois. Pour obtenir de l'aide, répondez AIDE, STOP pour annuler.
  - La réponse que vous souhaitez envoyer lorsque les clients répondent à vos messages avec le mot-clé AIDE. Ce message doit inclure les informations de contact pour le support client. Par exemple : Alertes de *ProductName* : Aide à l'adresse *exemple.com/aide* ou composez le *(800) 555-0199*. Les tarifs Msg&data peuvent s'appliquer. *2* msgs par mois. Répondez STOP pour annuler.
  - La réponse que vous souhaitez envoyer lorsque les clients répondent à vos messages avec le mot-clé STOP. Ce message doit confirmer que l'utilisateur ne recevra plus de messages de votre part. Par exemple : vous vous êtes désabonné des alertes de *ProductName*. Aucun autre message ne sera envoyé. Pour obtenir de l'aide, répondez AIDE ou composez le *(800) 555-0199*.
  - Le texte que vous avez l'intention d'envoyer pour rappeler périodiquement que l'utilisateur est abonné à vos messages. Par exemple : Rappel : vous êtes abonné aux alertes de compte de ExampleCorp. Des tarifs de message et de données peuvent s'appliquer. STOP pour annuler ou AIDE pour plus d'informations.
  - Un exemple de chaque type de message que vous avez l'intention d'envoyer à l'aide de votre code court. Fournissez au moins trois exemples. Si vous avez l'intention d'envoyer plus de trois types de messages, fournissez des exemples de tous ces messages.

> **Important**
>
> Les opérateurs de téléphonie mobile nous demandent de fournir toutes les informations mentionnées ci-dessus afin de mettre en service les codes courts. Nous ne pouvons pas traiter votre demande tant que vous n'avez pas fourni toutes ces informations.

10. (Facultatif) Si vous souhaitez soumettre d'autres demandes, choisissez Ajouter une autre demande. Si vous incluez plusieurs demandes, fournissez les informations requises pour chacune d'elles. Pour en savoir plus sur les informations requises, consultez les autres sections dans Demande de prise en charge de la messagerie SMS avec Amazon SNS.
11. Sous Options de contact, pour Langue de contact préférée, choisissez la langue dans laquelle vous souhaitez recevoir les communications pour ce cas.
12. Lorsque vous avez terminé, choisissez Submit (Soumettre).

Après réception de votre demande, nous envoyons une première réponse dans un délai de 24 heures. Nous sommes susceptibles de vous contacter pour demander des informations supplémentaires. Si nous sommes en mesure de vous fournir un code court, nous vous envoyons les informations sur les coûts inhérents à l'obtention du code court dans le pays ou la région que vous avez spécifié dans votre demande. Nous fournissons également une estimation de la durée nécessaire pour mettre en service un code court dans votre pays ou région. Il faut généralement plusieurs semaines pour mettre en service un code court, bien que ce délai puisse être nettement plus long ou plus court selon le pays ou la région associé au code court.

> **Note**
>
> Les frais associés à l'utilisation de codes courts sont applicables immédiatement après que nous avons envoyé votre demande de code court aux opérateurs. Vous devez payer ces frais, même si le code court n'a pas été encore entièrement alloué.

Pour empêcher que nos systèmes soient utilisés pour envoyer des contenus indésirables ou malveillants, chaque demande est traitée avec soin. Nous pourrions ne pas être en mesure de traiter votre demande si votre cas d'utilisation n'est pas conforme à nos stratégies.

Étapes suivantes

Vous avez enregistré un code court auprès d'opérateurs sans fil et passé en revue vos paramètres dans la console Amazon SNS. Vous pouvez désormais utiliser Amazon SNS pour envoyer des SMS avec votre code court en tant que numéro d'origine.

Demande de numéros 10DLC, de numéros sans frais et de codes longs P2P pour la messagerie SMS avec Amazon SNS

> **Important**
>
> À compter du 1er juin 2021, les fournisseurs de télécommunications américains ne prennent plus en charge l'utilisation de codes longs person-to-person (P2P) pour les communications application-to-person (A2P) vers des destinations américaines. Au lieu de cela, vous devez utiliser un autre type d'ID d'origine pour ces messages. Pour de plus amples informations, veuillez consulter 10DLC.

Pour demander des numéros 10DLC, des numéros gratuits, et des codes longs P2P, utilisez la console Amazon Pinpoint. Pour obtenir des instructions détaillées, consultez Demande d'un numéro dans le Guide de l'utilisateur Amazon Pinpoint.

> **Important**
>
> Les opérateurs mobiles américains ont récemment modifié leur réglementation et exigeront que tous les numéros gratuits soient enregistrés auprès d'un organisme de réglementation avant le 30 septembre 2022. Pour plus d'informations sur l'enregistrement d'un numéro gratuit, consultez Enregistrement de votre numéro gratuit.
> Si vous avez acheté votre numéro gratuit le 30 septembre 2022 ou avant, il sera à l'état Active (Actif) jusqu'au 1er octobre 2022, sauf si vous avez terminé votre enregistrement et qu'il a été retourné à l'état Completed (Terminé). Sinon, il sera à l'état Pending (En attente) et vous ne pourrez pas envoyer de messages avec celui-ci tant que vous n'aurez pas enregistré le numéro, que l'enregistrement n'aura pas été retourné ou qu'il sera à l'état Active (Actif). L'enregistrement peut prendre jusqu'à 15 jours ouvrables.

Pour enregistrer votre société de 10 DLC et votre campagne, ouvrez la console Amazon Pinpoint dans la région USA Est (Virginie du Nord). Au lieu de demander un numéro de 10 DLC, utilisez la

console AWS Service Quotas pour créer un cas d'augmentation de limite de service lorsque vous demandez le numéro de 10 DLC pour cette région. Pour plus d'informations sur les régions où Amazon Pinpoint est disponible, consultez Points de terminaison et quotas Amazon Pinpoint dans la Références générales AWS.

> **Note**
>
> Si vous n'êtes pas encore familiarisé avec l'envoi de SMS avec Amazon SNS, vous devez également demander un seuil de dépenses mensuelles pour l'envoi de SMS qui réponde aux demandes attendues de votre cas d'utilisation des SMS. Par défaut, votre seuil de dépenses mensuel est fixé à 1 USD. Pour de plus amples informations, veuillez consulter Demande d'augmentations de votre quota de dépenses mensuelles pour l'envoi de SMS pour Amazon SNS.

Demande d'ID expéditeur pour les SMS avec Amazon SNS

> **Important**
>
> Si vous n'êtes pas encore familiarisé avec l'envoi de SMS avec Amazon SNS, demandez un seuil de dépenses mensuel pour l'envoi de SMS qui réponde à votre cas d'utilisation prévu des SMS. Par défaut, votre seuil de dépenses mensuel est fixé à 1 USD. Vous pouvez demander une augmentation de votre limite de dépenses dans la demande de support incluant votre demande d'ID expéditeur. Ou, si vous préférez, vous pouvez soumettre un cas distinct. Pour de plus amples informations, veuillez consulter Demande d'augmentations de votre quota de dépenses mensuelles pour l'envoi de SMS pour Amazon SNS.

Dans la messagerie SMS, un ID expéditeur est un nom qui apparaît en tant qu'expéditeur du message sur les appareils des destinataires. Les ID expéditeur sont un moyen utile de vous identifier auprès des destinataires de vos messages.

La prise en charge des ID expéditeur varie selon les pays . Par exemple, les transporteurs aux États-Unis ne prennent pas du tout en charge les ID expéditeur, mais les transporteurs en Inde exigent que les expéditeurs utilisent les ID expéditeur. Pour obtenir la liste complète des pays qui prennent en charge les ID expéditeur, consultez Pays et régions pris en charge.

> **Important**
>
> Certains pays exigent que vous enregistriez les ID expéditeur avant de les utiliser pour envoyer des messages. En fonction du pays, ce processus d'inscription peut prendre plusieurs semaines. Les pays qui nécessitent des ID expéditeur pré-enregistrés sont indiqués dans le tableau de la page Pays pris en charge.
> Vous pouvez utiliser et enregistrer le même ID d'expéditeur dans plusieurs comptes AWS pour les messages SMS. Si vous bénéficiez d'un support d'entreprise et que vous enregistrez plusieurs modèles dans plusieurs comptes, suivez les étapes ci-dessous et travaillez avec votre gestionnaire technique de compte pour vous assurer que votre expérience d'onboarding est coordonnée.

Si vous envoyez des messages à des destinataires dans un pays où les ID expéditeur sont pris en charge, et que ce pays ne nécessite pas que vous enregistriez votre ID expéditeur, vous n'avez pas à effectuer d'autres étapes. Vous pouvez commencer immédiatement à envoyer des messages qui incluent des valeurs d'ID expéditeur.

Vous n'avez besoin de suivre les procédures de cette page que si vous prévoyez d'envoyer des messages vers un pays où l'enregistrement des ID expéditeur est obligatoire.

Étape 1 : ouvrir une demande de SMS Amazon SNS

Si vous prévoyez d'envoyer les messages à des destinataires dans un pays où les ID expéditeur sont obligatoires, vous pouvez demander un ID expéditeur en créant une nouvelle demande dans le Centre de support AWS.

> **Note**
>
> Si vous prévoyez d'envoyer des messages à des destinataires dans un pays où les ID expéditeur sont autorisés, mais pas obligatoires, vous n'avez pas besoin d'ouvrir une demande dans le Centre de support. Vous pouvez commencer immédiatement à envoyer des messages qui utilisent les ID expéditeur.

Pour demander un ID expéditeur

1. Connectez-vous à AWS Management Console via https://console.aws.amazon.com/.

2. Dans le menu Support, choisissez Centre de support.
3. Dans le panneau Vos cas de support, choisissez Créer un cas.
4. Cliquez sur le lien Vous cherchez à augmenter la limite de service ?, puis procédez comme suit :

   - Pour Limit type (Type de limite), choisissez Pinpoint SMS (SMS Pinpoint).
   - (Facultatif) Pour Fournir un lien vers le site ou l'application qui enverra les SMS, identifiez le site Web ou l'application où les membres de votre public accepteront de recevoir vos SMS.
   - (Facultatif) Pour Quel type de message envisagez-vous d'envoyer, choisissez le type de message que vous prévoyez d'envoyer à l'aide de votre code long :
     - Mot de passe ponctuel – Messages fournissant des mots de passe que vos clients utilisent pour s'authentifier sur votre site web ou votre application.
     - Promotionnel – Messages non stratégiques faisant la promotion de votre entreprise ou de votre service, comme des offres spéciales ou des annonces.
     - Transactionnel – Messages d'information importants qui prennent en charge les transactions clients, comme des confirmations de commande ou des alertes de compte. Les messages transactionnels ne peuvent pas comporter de contenu promotionnel ni marketing.
   - (Facultatif) Pour À partir de quelle région AWS allez-vous envoyer des messages, choisissez la Région AWS à partir de laquelle vous allez envoyer des SMS.
   - (Facultatif) Pour À quels pays comptez-vous envoyer des messages, saisissez les pays dans lesquels vous souhaitez enregistrer un ID d'expéditeur. La prise en charge des ID expéditeur et des exigences d'enregistrement d'ID expéditeur peut varier selon les pays. Pour de plus amples informations, veuillez consulter Pays et régions pris en charge.

     Si la liste des pays dépasse la limite de caractères pour cette zone de texte, vous pouvez plutôt répertorier les pays dans la section Description de la demande.
   - (Facultatif) Dans Comment vos clients acceptent-ils de recevoir des messages de votre part, saisissez une description de la façon dont vos clients acceptent de recevoir des messages de votre part.
   - (Facultatif) Dans le champ Veuillez fournir le modèle de message que vous comptez utiliser pour envoyer des messages à vos clients, incluez tous les modèles de message que vous allez utiliser.
5. Sous Requests (Demandes), complétez les sections suivantes :

   - Pour Région, choisissez la Région AWS de votre ID d'expéditeur.

> **Note**
>
> La région est obligatoire dans la section Demandes. Même si vous avez fourni ces informations dans la section Détails du cas, vous devez également les inclure ici.

- Pour Type de ressource, choisissez Limites générales.
- Pour le champ Limite, sélectionnez Accès de production SMS.

6. Pour Valeur de la nouvelle limite, choisissez le nombre d'ID expéditeur que vous demandez. Par défaut, cette valeur est **1**.
7. (Facultatif) Si vous souhaitez soumettre d'autres demandes, choisissez Ajouter une autre demande. Pour en savoir plus sur les informations requises, consultez les autres sections dans Demande de prise en charge de la messagerie SMS avec Amazon SNS.
8. Sous Case description (Description de cas), pour Use case description (Description du cas d'utilisation), saisissez les informations suivantes :
   - ID expéditeur que vous souhaitez enregistrer.
   - Modèle que vous prévoyez d'utiliser pour vos SMS.
   - Nombre de messages que vous prévoyez d'envoyer à chaque destinataire par mois.
   - Informations sur la façon dont vos clients acceptent de recevoir des messages de votre part.
   - Nom de votre entreprise ou organisation.
   - Adresse associée à votre entreprise ou organisation.
   - Pays dans lequel votre entreprise ou organisation est basée.
   - Numéro de téléphone de votre entreprise ou organisation.
   - URL du site Web de votre entreprise ou organisation.
9. Sous Options de contact, pour Langue de contact préférée, choisissez la langue dans laquelle vous souhaitez recevoir les communications pour ce cas.
10. Lorsque vous avez terminé, choisissez Submit (Soumettre).

Après réception de votre demande, nous envoyons une première réponse dans un délai de 24 heures. Nous sommes susceptibles de vous contacter pour demander des informations supplémentaires. Si nous sommes en mesure de vous fournir un ID expéditeur, nous vous envoyons une estimation du temps nécessaire pour permettre sa mise en service.

Pour empêcher que nos systèmes soient utilisés pour envoyer des contenus indésirables ou malveillants, chaque demande est traitée avec soin. Nous pourrions ne pas être en mesure de traiter votre demande si votre cas d'utilisation n'est pas conforme à nos stratégies.

Étape 2 : mettre à jour vos paramètres de SMS dans la console Amazon SNS

Lorsque nous aurons terminé le processus d'obtention de votre ID expéditeur, nous vous adresserons une notification. Lorsque vous recevez cette notification, procédez comme suit dans cette section pour configurer Amazon SNS et utiliser votre ID d'expéditeur comme ID d'expéditeur par défaut pour tous les messages envoyés à l'aide de votre compte. Vous pouvez également choisir de spécifier l'ID expéditeur à utiliser lors de la publication du message.

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Mobile, puis SMS.
3. Dans la section Préférences de SMS, choisissez Modifier.
4. Dans la section Détails, dans le champ ID d'expéditeur par défaut, entrez l'ID d'expéditeur fourni à utiliser comme valeur par défaut pour tous les messages envoyés depuis votre compte.
5. Lorsque vous avez terminé, choisissez Save changes (Enregistrer les modifications).

Étapes suivantes

Vous avez enregistré un ID d'expéditeur et mis à jour vos paramètres dans la console Amazon SNS. Vous pouvez désormais utiliser Amazon SNS pour envoyer des SMS avec votre ID d'expéditeur. Les destinataires des SMS dans les pays pris en charge verront votre ID expéditeur en tant qu'expéditeur du message sur leurs appareils. Si un ID expéditeur différent est utilisé lors de la publication des messages, il remplacera l'ID par défaut configuré ici.

Demande d'augmentations de votre quota de dépenses mensuelles pour l'envoi de SMS pour Amazon SNS

Amazon SNS propose des quotas de dépenses destinés à vous aider à gérer le coût mensuel maximal lié à l'envoi de SMS à l'aide de votre compte. Le quota de dépenses limite vos risques en cas d'attaque malveillante et empêche votre application en amont d'envoyer plus de messages que prévu. Vous pouvez configurer Amazon SNS pour arrêter la publication de SMS lorsqu'il détermine que l'envoi d'un SMS entraînera un coût dépassant votre quota de dépenses pour le mois en cours.

Pour éviter que vos activités ne soient impactées, nous vous recommandons de demander un quota de dépenses suffisamment élevé pour faire face à vos charges de travail de production. Pour de

plus amples informations, consultez Étape 1 : ouvrir une demande de SMS Amazon SNS. Une fois le quota reçu, vous pouvez gérer votre risque en appliquant tout le quota, ou seulement une partie, comme le décrit la section Étape 2 : Mettre à jour les paramètres de SMS. L'application d'une valeur inférieure à la totalité du quota vous permet de contrôler vos dépenses mensuelles tout en gardant la possibilité d'augmenter en fonction de vos besoins.

> **Important**
>
> Amazon SNS étant un système distribué, il cesse d'envoyer des SMS après quelques minutes si le quota de dépenses est dépassé. Si vous continuez à envoyer des SMS au cours de cette période, vous risquez de devoir payer des frais au-delà de votre quota.

Le quota des dépenses pour tous les nouveaux comptes est défini à 1 USD par mois. Ce quota doit vous permettre de tester les capacités d'envoi de messages d'Amazon SNS. Si vous souhaitez demander une augmentation du quota de dépenses en SMS pour votre compte, ouvrez une demande d'augmentation de quota dans le Centre de support AWS.

Étape 1 : ouvrir une demande de SMS Amazon SNS

Vous pouvez demander une augmentation de votre quota de dépenses mensuelles en envoyant une demande dans le Centre de support AWS.

> **Note**
>
> Certains champs du formulaire de demande sont marqués comme « facultatifs ». Toutefois, AWS Support nécessite toutes les informations mentionnées dans les étapes suivantes afin de traiter votre demande. Si vous ne fournissez pas toutes les informations requises, le traitement de votre demande peut être retardé.

1. Connectez-vous à AWS Management Console via https://console.aws.amazon.com/.
2. Dans le menu Support, choisissez Centre de support.
3. Dans le panneau Vos cas de support, choisissez Créer un cas.
4. Cliquez sur le lien Vous cherchez à augmenter la limite de service ?, puis procédez comme suit :
   - Pour Type de limite, choisissez SMS SNS.

- (Facultatif) Pour Fournir un lien vers le site ou l'application qui enverra les SMS, entrez les informations sur le site Web, l'application ou le service qui enverra des SMS.
- (Facultatif) Pour Quel type de message envisagez-vous d'envoyer, choisissez le type de message que vous prévoyez d'envoyer à l'aide de votre code long :
  - Mot de passe ponctuel – Messages fournissant des mots de passe que vos clients utilisent pour s'authentifier sur votre site web ou votre application.
  - Promotionnel – Messages non stratégiques faisant la promotion de votre entreprise ou de votre service, comme des offres spéciales ou des annonces.
  - Transactionnel – Messages d'information importants qui prennent en charge les transactions clients, comme des confirmations de commande ou des alertes de compte. Les messages transactionnels ne peuvent pas comporter de contenu promotionnel ni marketing.
- (Facultatif) Pour À partir de quelle région AWS allez-vous envoyer des messages, choisissez la région à partir de laquelle vous allez envoyer des messages.
- (Facultatif) Pour Vers quels pays comptez-vous envoyer des messages, entrez le pays ou la région dans lesquels vous souhaitez acheter des codes courts.
- (Facultatif) Dans Comment vos clients acceptent-ils de recevoir messages de votre part, fournissez des détails sur votre processus d'acceptation.
- (Facultatif) Dans le champ Veuillez fournir le modèle de message que vous comptez utiliser pour envoyer des messages à vos clients, incluez le modèle que vous allez utiliser.

5. Sous Requests (Demandes), complétez les sections suivantes :

   - Pour Région, choisissez la région à partir de laquelle vous allez envoyer des messages.

     > **Note**
     >
     > La région est obligatoire dans la section Demandes. Même si vous avez fourni ces informations dans la section Détails du cas, vous devez également les inclure ici.

   - Pour Type de ressource, choisissez Limites générales.
   - Pour Limit (Limite), choisissez Account Spend Threshold Increase (Augmenter la limite de dépenses du compte).

6. Pour Nouvelle valeur limite, saisissez le montant maximal (en USD) que vous pouvez dépenser en SMS chaque mois civil.

7. Sous Case description (Description de cas), pour Use case description (Description du cas d'utilisation), saisissez les informations suivantes :

   - Le site Web ou l'application de l'entreprise ou du service qui envoie les SMS.
   - Le service fourni par votre site Web ou votre application, ainsi que la façon dont vos SMS contribuent à ce service.
   - La façon dont les utilisateurs s'inscrivent volontairement pour recevoir vos SMS sur votre site web, votre application ou tout autre emplacement.

   Si le quota de dépenses que vous demandez (c'est-à-dire la valeur que vous spécifiez pour New quota value (Nouvelle valeur de quota), dépasse 10 000 USD, fournissez les informations supplémentaires suivantes pour chaque pays vers lequel vous envoyez des SMS :

   - Si vous utilisez un ID expéditeur ou un code court. Si vous utilisez un ID expéditeur, indiquez :
     - L'ID expéditeur.
     - Si l'ID expéditeur est enregistré auprès des opérateurs sans fil dans le pays.
   - Le nombre de transactions par seconde (TPS) attendu pour votre SMS.
   - La taille moyenne des SMS.
   - Le modèle pour les messages que vous envoyez vers le pays.
   - (Facultatif) L'encodage de caractères nécessaire, le cas échéant.
8. (Facultatif) Si vous souhaitez soumettre d'autres demandes, choisissez Ajouter une autre demande. Si vous incluez plusieurs demandes, fournissez les informations requises pour chacune d'elles. Pour en savoir plus sur les informations requises, consultez les autres sections dans Demande de prise en charge de la messagerie SMS avec Amazon SNS.
9. Sous Options de contact, pour Langue de contact préférée, choisissez la langue dans laquelle vous souhaitez recevoir les communications pour ce cas.
10. Lorsque vous avez terminé, choisissez Submit (Soumettre).

L'équipe AWS Support répond à votre demande dans un délai de 24 heures.

Pour empêcher que nos systèmes soient utilisés pour envoyer des contenus indésirables ou malveillants, chaque demande est traitée avec soin. Si cela est possible, nous accepterons votre demande dans ce délai de 24 heures. En revanche, si nous avons besoin de plus amples informations, le traitement de votre demande peut prendre plus de temps.

Nous pourrions ne pas être en mesure de donner suite à votre demande si votre cas d'utilisation n'est pas conforme à nos politiques.

Étape 2 : mettre à jour vos paramètres de SMS dans la console Amazon SNS

Lorsque vous aurez été informé que votre quota de dépenses mensuelles a été augmenté, vous devez ajuster ce quota pour votre compte dans la console Amazon SNS.

> **Important**
>
> Vous devez suivre ces étapes ou votre limite de dépenses en SMS ne sera pas augmentée.

Pour ajuster votre quota de dépenses sur la console

1. Connectez-vous à la console Amazon SNS.
2. Ouvrez le menu de navigation de gauche, développez Mobile, puis choisissez SMS.
3. Sur la pageSMS mobile, dans la section Préférences de SMS, choisissez Modifier.
4. Sur la page Modifier les préférences de SMS, dans la section Détails, entrez votre nouvelle limite de dépenses en SMS dans le champ Limite de dépenses du compte.

   > **Note**
   >
   > Il se peut que vous receviez un avertissement indiquant que la valeur entrée est supérieure à la limite de dépenses par défaut. Vous pouvez ignorer ce texte.

5. Sélectionnez Save Changes (Enregistrer les modifications).

   > **Note**
   >
   > Si vous obtenez une erreur « Paramètre non valide », consultez le contact du support AWS et confirmez que vous avez saisi la nouvelle limite de dépenses SMS correcte. Si vous rencontrez toujours un problème, ouvrez un dossier dans le Centre de support AWS.

Définition des préférences de messagerie SMS

Utilisez Amazon SNS pour spécifier des préférences pour la messagerie SMS. Par exemple, vous pouvez spécifier si vous souhaitez optimiser les distributions à des fins de coût ou de fiabilité, votre limite de dépenses mensuelles, comment les distributions sont consignées et si vous souhaitez vous abonner à des rapports d'utilisation quotidiens des SMS.

Ces préférences prennent effet pour chaque SMS que vous envoyez depuis votre compte, mais vous pouvez en remplacer certaines lorsque vous envoyez un message individuel. Pour plus d'informations, consultez Publication sur un téléphone mobile.

Rubriques

Configuration des préférences de messagerie SMS à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Choisissez une région qui prend en charge les SMS.
3. Dans le panneau de navigation, choisissez Mobile, SMS.
4. Sur la pageSMS mobile, dans la section Préférences de SMS, choisissez Modifier.
5. Sur la page Modifier les préférences SMS, dans la section Détails, procédez comme suit :

   a. Pour Type de message par défaut, sélectionnez l'une des options suivantes :

      - Promotionnel – Messages non stratégiques (par exemple, marketing). Amazon SNS optimise la distribution de messages pour générer le coût le plus bas.
      - Transactionnel (par défaut) – Messages stratégiques qui prennent en charge les transactions clients, comme des codes secrets uniques pour l'authentification multifacteur. Amazon SNS optimise la distribution de messages pour obtenir la meilleure fiabilité possible.

      Pour la tarification des messages promotionnels et transactionnels, consultez la page Tarifs SMS internationaux.

b. Dans le champ Limite de dépense du compte, saisissez le montant maximal (en USD) que vous voulez dépenser pour les SMS par mois calendaire.

> **Important**
>
> - Par défaut, le quota de dépenses est définie sur 1,00 USD. Si vous souhaitez augmenter le quota de service, soumettez une demande.
> - Si le montant défini dans la console dépasse le quota de votre service, Amazon SNS cesse la publication de SMS.
> - Amazon SNS étant un système distribué, il cesse d'envoyer des SMS quelques minutes après le dépassement du quota de dépense. Si vous continuez à envoyer des SMS au cours de cet intervalle, vous risquez de devoir payer des coûts au-delà de votre quota.

6. (Facultatif) Pour ID de l'expéditeur par défaut, entrez un ID personnalisé, par exemple votre marque, qui s'affichera en tant qu'expéditeur de l'appareil de réception.

> **Note**
>
> La prise en charge des ID expéditeur varie selon les pays.

7. (Facultatif) Saisissez le nom du compartiment Amazon S3 pour les rapports d'utilisation.

> **Note**
>
> La politique de compartiment S3 doit accorder l'accès en écriture à Amazon SNS.

8. Sélectionnez Enregistrer les modifications.

Configuration des préférences (AWS SDK)

Pour définir vos préférences en matière de SMS à l'aide de l'un AWS des SDK, utilisez l'action de ce SDK qui correspond à la `SetSMSAttributes` demande dans l'API Amazon SNS. Cette demande vous permet d'affecter des valeurs aux différents attributs SMS, tels que votre quota de dépenses mensuelles et votre type de SMS par défaut (promotionnel ou transactionnel). Pour tous les attributs SMS, consultez SetSMSAttributes dans la Référence de l'API Amazon Simple Notification Service.

Les exemples de code suivants montrent comment utiliser`SetSMSAttributes`.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Comment utiliser Amazon SNS pour définir l’attribut DefaultSMSType.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour définir les attributs des SMS

L'exemple `set-sms-attributes` suivant définit l'ID d'expéditeur par défaut des SMS sur `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Cette commande ne produit aucun résultat.

- Pour plus d'informations sur l'API, consultez SetSMSAttributes dans la Référence des commandes AWS CLI .

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for JavaScript .

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for PHP .

Envoi de messages SMS

Cette section décrit l'envoi de messages SMS.

Rubriques

- Publication dans une rubrique
- Publication sur un téléphone mobile

Publication dans une rubrique

Vous pouvez publier simultanément un seul message SMS à plusieurs numéros de téléphone en abonnant ces numéros de téléphone à une rubrique Amazon SNS. Une rubrique SNS est un canal de communication auquel vous pouvez ajouter des abonnés. Vous pouvez ensuite publier des messages à tous les abonnés. Un abonné reçoit tous les messages publiés dans la rubrique jusqu'à ce que vous annuliez l'abonnement ou jusqu'à ce qu'il refuse de recevoir des SMS provenant de votre compte AWS.

Rubriques

- Envoi d'un message à une rubrique (console)
- Envoi d'un message à une rubrique (kits SDK AWS)

Envoi d'un message à une rubrique (console)

Pour créer une rubrique

Procédez comme suit si vous ne disposez pas encore d'une rubrique à laquelle envoyer des SMS.

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, choisissez une région AWS qui prend en charge la messagerie SMS.
3. Dans le volet de navigation, choisissez Rubriques.
4. Sur la page Rubriques, choisissez Créer une rubrique.
5. Sur la page Créer une rubrique sous Détails, procédez comme suit :

   a. Pour Type, choisissez Standard.

   b. Pour Nom de la rubrique, saisissez un nom de rubrique.

   c. (Facultatif) Dans le champ Nom d'affichage, saisissez un préfixe personnalisé pour vos messages SMS. Lorsque vous envoyez un message à la rubrique, Amazon SNS ajoute le nom d'affichage suivi d'un signe supérieur (>) et d'un espace. Les noms d'affichage ne sont pas sensibles à la casse et Amazon SNS les convertit en majuscules. Par exemple, si le nom d'affichage d'une rubrique est `MyTopic` et que le message est `Hello World!`, le message se présente comme suit :

   ```
   MYTOPIC> Hello World!
   ```

6. Choisissez Créer une rubrique. Le nom de la rubrique et l'Amazon Resource Name (ARN) apparaissent sur la page Rubriques.

Pour créer un abonnement aux SMS

Vous pouvez utiliser les abonnements pour envoyer un SMS à plusieurs destinataires en publiant le message une seule fois dans votre rubrique.

> **Note**
>
> Lorsque vous commencez à utiliser Amazon SNS pour envoyer des messages SMS, votre compte AWS se trouve dans l'environnement de test (sandbox) pour SMS. L'environnement de test (sandbox) pour SMS offre un environnement sûr pour vous permettre d'essayer les fonctionnalités d'Amazon SNS sans risquer votre réputation d'expéditeur SMS. Même si votre compte se trouve dans l'environnement de test (sandbox) pour SMS, vous pouvez utiliser toutes les fonctionnalités d'Amazon SNS, mais vous pouvez envoyer des SMS uniquement à des numéros de téléphone de destination vérifiés. Pour de plus amples informations, veuillez consulter Environnement de test (sandbox) pour SMS.

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Abonnements.
3. Sur la page Abonnements, choisissez Créer un abonnement.
4. Sur la page Créer un abonnement, sous Détails, procédez comme suit :
   a. Pour ARN de rubrique, saisissez ou choisissez l'ARN (Amazon Resource Name) de la rubrique à laquelle vous souhaitez envoyer des SMS.
   b. Pour Protocole, choisissez SMS.
   c. Pour Point de terminaison, saisissez le numéro de téléphone que vous souhaitez abonner à votre rubrique.
5. Choisissez Create subscription (Créer un abonnement). Les informations d'abonnement s'affichent sur la page Abonnements.

   Pour ajouter d'autres numéros de téléphone, répétez ces étapes. Vous pouvez également ajouter d'autres types d'abonnements, tels que la messagerie électronique.

Pour envoyer un message

Lorsque vous publiez un message dans une rubrique, Amazon SNS tente de livrer ce message à chaque numéro de téléphone qui est abonné à la rubrique.

1. Dans la console Amazon SNS, sur la page Rubriques, choisissez le nom de la rubrique à laquelle envoyer des SMS.
2. Sur la page des détails de la rubrique, sélectionnez Publier le message.
3. Sur la page Publier un message dans la rubrique, sous Détails du message, procédez comme suit :
   a. Pour Objet, laissez le champ vide, sauf si votre rubrique contient des abonnements par e-mail et que vous voulez effectuer une publication à la fois dans des abonnements par e-mail et par SMS. Amazon SNS utilise l'Objet que vous saisissez en tant que ligne d'objet de l'e-mail.
   b. (Facultatif) Pour time-to-live (TTL), saisissez un certain nombre de secondes pendant lesquelles Amazon SNS doit envoyer votre message SMS à tous les abonnés aux points de terminaison d'une application mobile.
4. Sous Corps du message, procédez comme suit :

a. Pour Structure des messages, choisissez Charge utile identique pour tous les protocoles de distribution pour envoyer le même message à tous les types de protocoles abonnés à votre rubrique. Ou, choisissez Charge utile personnalisée pour chaque protocole de distribution pour personnaliser le message pour les abonnés de différents types de protocole. Par exemple, vous pouvez saisir un message par défaut pour les abonnés à un numéro de téléphone et un message personnalisé pour les abonnés à un courrier électronique.

b. Pour Corps du message à envoyer au point de terminaison, saisissez votre message ou vos messages personnalisés par protocole de distribution.

   Si votre rubrique possède un nom d'affichage, Amazon SNS l'ajoute au message, augmentant ainsi sa longueur. La longueur du nom affiché représente le nombre de caractères dans le nom plus deux caractères pour le signe supérieur (**>**) et l'espace ajoutés par Amazon SNS.

   Pour plus d'informations sur les quotas de taille pour les SMS, consultez Publication sur un téléphone mobile.

5. (Facultatif) Pour Attributs de message, ajoutez des métadonnées de message telles que des horodatages, des signatures et des ID.
6. Choisissez Publier le message. Amazon SNS envoie le message SMS et affiche un message de réussite.

Envoi d'un message à une rubrique (kits SDK AWS)

Pour utiliser un kit SDK AWS, vous devez le configurer avec vos informations d'identification. Pour de plus amples informations, veuillez consulter Les fichiers de configuration et d'informations d'identification partagés dans le AWSGuide de référence des kits SDK et des outils.

L'exemple de code suivant illustre comment :

- Créer une rubrique Amazon SNS.
- Abonner des numéros de téléphone à la rubrique.
- Publier des messages SMS dans la rubrique afin que tous les numéros de téléphone abonnés reçoivent le message en même temps.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créer une rubrique et renvoyez son 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 "";
    }
}
```

Abonner un point de terminaison à une rubrique

```
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);
        }
    }
}
```

Définir des attributs sur le message, tels que l'ID de l'expéditeur, le prix maximal et son type. Les attributs de message sont facultatifs.

```
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);
        }
    }
}
```

Publier un message dans une rubrique Le message est envoyé à chaque abonné.

```
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);
        }
    }
}
```

Publication sur un téléphone mobile

Vous pouvez utiliser Amazon SNS pour envoyer des messages SMS directement à un téléphone mobile sans abonner le numéro de téléphone à une rubrique Amazon SNS.

> **Note**
>
> L'abonnement de numéros de téléphone à une rubrique est utile si vous voulez envoyer un message à plusieurs numéros de téléphone à la fois. Pour obtenir des instructions sur la publication d'un SMS dans une rubrique, consultez Publication dans une rubrique.

Lorsque vous envoyez un message, vous pouvez contrôler s'il est optimisé en matière de coût ou de fiabilité de distribution. Vous pouvez également spécifier un ID de l'expéditeur ou numéro d'origine. Si vous envoyez le message par programmation à l'aide de l'API Amazon SNS ou AWS des SDK, vous pouvez spécifier un prix maximum pour la livraison du message.

Chaque SMS peut contenir jusqu'à 140 octets et le quota de caractères dépend du schéma de codage. Par exemple, un SMS peut contenir :

- 160 caractères GSM
- 140 caractères ASCII
- 70 caractères UCS-2

Si vous publiez un message qui dépasse ce quota de taille, Amazon SNS le fractionne en plusieurs messages, chacun n'excédant pas ce quota. Les messages ne sont pas coupés au milieu d'un mot, mais plutôt sur la base d'un mot entier. Le quota de taille totale pour une action de publication de SMS est de 1600 octets.

Lorsque vous envoyez un SMS, vous spécifiez le numéro de téléphone au format E.164, structure de numérotation standard utilisée pour les télécommunications internationales. Les numéros qui respectent ce format peuvent comporter 15 chiffres au maximum, avec pour préfixe le signe plus (+) et le code pays. Par exemple, un numéro de téléphone américain au format E.164 se présente sous la forme suivante : +1XXX5550100.

Rubriques

- Envoi d'un message (console)

- Envoi d'un message (AWS SDK)

Envoi d'un message (console)

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, choisissez une région AWS qui prend en charge la messagerie SMS.
3. Dans le panneau de navigation, sélectionnez Text messaging (SMS).
4. Sur la page Messagerie texte mobile (SMS), choisissez Publier un message texte.
5. Sur la page Publier un message SMS, pour Type de message choisissez l'une des options suivantes :

   - Promotional – Messages non stratégiques, tels que les messages marketing.
   - Transactional – Messages stratégiques qui prennent en charge les transactions clients, comme des codes secrets uniques pour l'authentification multifacteur.

   > **Note**
   >
   > Ce paramètre au niveau des messages remplace le type de message par défaut au niveau du compte. Vous pouvez définir un type de message par défaut au niveau du compte dans la section Préférences de messagerie SMS de la page Messagerie texte mobile (SMS).

   Pour obtenir des informations sur la tarification des messages promotionnels et transactionnels, consultez Tarification mondiale SMS.
6. Pour Numéro de téléphone de destination, saisissez le numéro de téléphone auquel vous souhaitez envoyer le message.
7. Pour Message, saisissez le message à envoyer.
8. (Facultatif) Sous Identités d'origine, spécifiez comment vous identifier auprès de vos destinataires :

   - Pour spécifier un ID de l'expéditeur, saisissez un ID personnalisé contenant entre 3 et 11 caractères alphanumériques, dont au moins une lettre et aucun espace. L'ID expéditeur s'affiche en tant qu'expéditeur du message sur l'appareil de réception. Par exemple, vous

pouvez utiliser votre marque d’entreprise pour faciliter la reconnaissance de la source du message.

La prise en charge des ID expéditeur varie selon les pays et/ou les régions. Par exemple, les messages diffusés à des numéros de téléphone américains n'afficheront pas l'ID expéditeur. Pour les pays et régions qui prennent en charge les ID expéditeur, consultez Pays et régions pris en charge.

Si vous ne spécifiez pas d'ID d'expéditeur, l'un des éléments suivants s'affiche comme identité d'origine :

- Dans les pays qui prennent en charge les codes longs, le code long apparaît.
- Dans les pays où seuls les ID d'expéditeur sont pris en charge, NOTICE apparaît.

Cet ID expéditeur au niveau du message remplace l'ID expéditeur par défaut, que vous définissez sur la page Préférences de SMS.

- Pour spécifier un Numéro d'origine, saisissez une chaîne de 5 à 14 numéros à afficher comme numéro de téléphone de l'expéditeur sur l'appareil du destinataire. Cette chaîne doit correspondre à un numéro d'origine configuré dans votre Compte AWS pour le pays de destination. Le numéro d'origine peut être un numéro 10DLC, un numéro gratuit, un code person-to-person long ou un code court. Pour plus d’informations, consultez Identités d’origine des messages SMS.

  Si vous ne spécifiez pas de numéro d'origine, Amazon SNS sélectionne un numéro d'origine à utiliser pour le message texte SMS, en fonction de votre configuration. Compte AWS

9. Si vous envoyez des messages SMS à des destinataires en Inde, développez Attributs spécifiques au pays, et spécifiez les attributs suivants :

   - ID d'entité – ID d'entité ou ID d'entité principale (PE) pour l'envoi de messages SMS à des destinataires en Inde. Cet ID est une chaîne unique de 1 à 50 caractères fournie par la Telecom Regulatory Authority of India (TRAI) pour identifier l'entité que vous avez enregistrée auprès de la TRAI.
   - ID de modèle – ID de modèle pour l'envoi de messages SMS à des destinataires en Inde. Cet ID est une chaîne unique de 1 à 50 caractères fournie par la TRAI qui identifie le modèle que vous avez enregistré auprès de la TRAI. L'ID du modèle doit être associé à l'ID de l'expéditeur que vous avez spécifié pour le message.

Pour plus d'informations sur l'envoi de messages SMS à des destinataires en Inde, consultez Exigences relatives aux ID d'expéditeur pour l'Inde.

10. Choisissez Publier le message.

> **Tip**
>
> Pour envoyer des messages SMS à partir d'un numéro d'origine, vous pouvez également choisir Numéros d'origine dans le panneau de navigation de la console Amazon SNS. Choisissez un numéro d'origine qui inclut SMS dans la colonne Capacités, puis choisissez Publier un message texte.

Envoi d'un message (AWS SDK)

Pour envoyer un message SMS à l'aide de l'un des AWS SDK, utilisez l'opération d'API de ce SDK qui correspond à la `Publish` demande de l'API Amazon SNS. Cette demande vous permet d'envoyer un SMS directement à un numéro de téléphone. Vous pouvez également utiliser le paramètre `MessageAttributes` pour définir les valeurs des noms d'attributs suivants :

`AWS.SNS.SMS.SenderID`

Un ID personnalisé contenant entre 3 et 11 caractères alphanumériques ou des traits d'union (-), dont au moins une lettre et aucun espace. L'ID de l'expéditeur s'affiche en tant qu'expéditeur du message sur l'appareil de réception. Par exemple, vous pouvez utiliser votre marque d'entreprise pour permettre de faciliter la reconnaissance de la source du message.

La prise en charge des ID expéditeur varie selon les pays ou les régions. Par exemple, les messages diffusés à des numéros de téléphone américains n'affichent pas l'ID d'expéditeur. Pour les pays et régions qui prennent en charge les ID de l'expéditeur, consultez Pays et régions pris en charge.

Si vous ne spécifiez pas d'ID d'expéditeur, un code long s'affiche comme l'ID d'expéditeur dans les régions ou pays pris en charge. Pour les pays ou régions qui requièrent un ID d'expéditeur sous forme alphabétique, AVIS s'affiche en tant qu'ID d'expéditeur.

Cet attribut au niveau du message remplace l'attribut au niveau du compte `DefaultSenderID`, que vous pouvez définir à l'aide de la demande `SetSMSAttributes`.

`AWS.MM.SMS.OriginationNumber`

Une chaîne personnalisée de 5 à 14 nombres, qui peut inclure un signe plus initial en option (+). Cette chaîne de nombres apparaît comme le numéro de téléphone de l'expéditeur sur le périphérique de réception. La chaîne doit correspondre à un numéro d'origine configuré dans votre AWS compte pour le pays de destination. Le numéro d'origine peut être un numéro 10DLC, un numéro gratuit, un code long person-to-person (P2P) ou un code court. Pour plus d'informations, consultez Numéros d'origine.

Si vous ne spécifiez pas de numéro d'origine, Amazon SNS choisit un numéro d'origine en fonction AWS de la configuration de votre compte.

`AWS.SNS.SMS.MaxPrice`

Le prix maximum en USD que vous êtes prêt à payer pour envoyer le message SMS. Si Amazon SNS détermine que l'envoi du message entraînerait un coût supérieur à votre prix maximum, il n'envoie pas le message.

Cet attribut n'a aucun effet si le coût de vos month-to-date SMS a déjà dépassé le quota défini pour l'`MonthlySpendLimit`attribut. Vous pouvez définir l'attribut `MonthlySpendLimit` à l'aide de la demande `SetSMSAttributes`.

Si vous envoyez le message à une rubrique Amazon SNS, le prix maximum s'applique à chaque distribution de message envoyée sur chaque numéro de téléphone abonné à la rubrique.

`AWS.SNS.SMS.SMSType`

Le type de message que vous envoyez :

- `Promotional` (par défaut) – Messages non stratégiques, tels que messages marketing.
- `Transactional` – Messages stratégiques qui prennent en charge les transactions clients, comme des codes secrets uniques pour l'authentification multifacteur.

Cet attribut au niveau du message remplace l'attribut au niveau du compte `DefaultSMSType`, que vous pouvez définir à l'aide de la demande `SetSMSAttributes`.

`AWS.MM.SMS.EntityId`

Cet attribut est requis uniquement pour l'envoi de messages SMS à des destinataires en Inde.

Il s'agit de votre ID d'entité ou ID d'entité principale (PE) pour l'envoi de messages SMS aux destinataires en Inde. Cet ID est une chaîne unique de 1 à 50 caractères fournie par la Telecom Regulatory Authority of India (TRAI) pour identifier l'entité que vous avez enregistrée auprès de la TRAI.

`AWS.MM.SMS.TemplateId`

Cet attribut est requis uniquement pour l'envoi de messages SMS à des destinataires en Inde.

Ceci est votre modèle pour l'envoi de messages SMS à des destinataires en Inde. Cet ID est une chaîne unique de 1 à 50 caractères fournie par TRAI qui identifie le modèle que vous avez enregistré auprès de la TRAI. L'ID du modèle doit être associé à l'ID de l'expéditeur que vous avez spécifié pour le message.

Envoi d'un message

Les exemples de code suivants montrent comment publier des messages SMS à l'aide d'Amazon SNS.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}");
            }
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for .NET Référence de l'API.

C++

Kit SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for C++ Référence de l'API.

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for Java 2.x .

Kotlin

Kit SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section Publier de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for PHP Référence de l'API.

Python

Kit SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence du kit SDK AWS de l'API Python (Boto3).

Surveillance de l'activité SMS

En surveillant votre activité SMS, vous pouvez suivre les numéros de téléphone de destination, les distributions ayant réussi ou échoué, les motifs d'échec, les coûts et autres informations. Amazon SNS vous aide en récapitulant les statistiques dans la console, envoyant des informations à Amazon CloudWatch et en envoyant des rapports d'utilisation quotidiens des SMS à un compartiment Amazon S3 que vous spécifiez.

Rubriques

- Affichage des statistiques de distribution SMS
- Affichage des journaux et des métriques Amazon CloudWatch pour les distributions de SMS
- Affichage des rapports d'utilisation quotidiens des SMS

Affichage des statistiques de distribution SMS

Vous pouvez utiliser la console Amazon SNS pour afficher les statistiques concernant vos distributions SMS récentes.

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, définissez le sélecteur de région sur une région prenant en charge la messagerie SMS.
3. Dans le panneau de navigation, choisissez Messages texte (SMS).
4. Sur la page SMS, dans la section Statistiques du compte, affichez les graphiques pour vos SMS transactionnels et promotionnels délivrés. Chaque graphique présente les données suivantes pour les 15 jours précédents :
   - Delivery rate (pourcentage de distributions réussies)
   - Sent (nombre de tentatives de distribution)
   - Failed (nombre d'échecs de distribution)

Sur cette page, vous pouvez également cliquer sur le bouton Utilisation pour accéder au compartiment Amazon S3 où vous stockez vos rapports d'utilisation quotidiens. Pour de plus amples informations, veuillez consulter . Affichage des rapports d'utilisation quotidiens des SMS.

Affichage des journaux et des métriques Amazon CloudWatch pour les distributions de SMS

Vous pouvez utiliser Amazon CloudWatch et Amazon CloudWatch Logs pour contrôler les distributions de vos messages SMS.

Rubriques

- Affichage des métriques Amazon CloudWatch
- Affichage des CloudWatch Logs
- Exemple de journal pour une distribution SMS réussie
- Exemple de journal pour une distribution SMS ayant échoué
- Causes d'échec de la distribution SMS

Affichage des métriques Amazon CloudWatch

Amazon SNS collecte automatiquement des métriques sur vos distributions de SMS et les envoie à Amazon CloudWatch. Vous pouvez utiliser CloudWatch pour contrôler ces métriques et créer des alarmes pour vous avertir lorsqu'une métrique dépasse un seuil. Par exemple, vous pouvez contrôler les métriques CloudWatch afin de connaître votre vitesse de distribution de messages SMS et les frais relatifs aux messages SMS du mois actuel.

Pour plus d'informations sur la surveillance des métriques CloudWatch, la définition d'alarmes CloudWatch et les types de métriques disponibles, consultez Surveillance des rubriques Amazon SNS à l'aide de CloudWatch.

Affichage des CloudWatch Logs

Vous pouvez rassembler des informations sur les distributions de SMS qui ont réussi et celles qui ont échoué, en autorisant Amazon SNS à écrire dans Amazon CloudWatch Logs. Pour chaque SMS que vous envoyez, Amazon SNS écrit un journal qui inclut le prix du message, l'état de réussite ou d'échec, la cause d'échec (le cas échéant), la durée de conservation du message et d'autres informations.

Pour activer et afficher les CloudWatch Logs pour vos messages SMS

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, définissez le sélecteur de région sur une région prenant en charge la messagerie SMS.
3. Dans le panneau de navigation, choisissez Messages texte (SMS).
4. Sur la pageSMS mobile, dans la section Préférences de SMS, choisissez Modifier.
5. Sur la page suivante, développez la section Journalisation de l'état de distribution.
6. Pour Taux d'échantillonnage de réussite, spécifiez le pourcentage de distributions de SMS réussies pour lesquelles Amazon SNS écrira des journaux dans CloudWatch Logs. Par exemple :

   - Pour écrire des journaux uniquement pour les distributions qui ont échoué, définissez cette valeur sur 0.
   - Pour écrire des journaux pour 10 % des distributions qui ont réussi, définissez-la sur 10.

   Si vous ne spécifiez pas de pourcentage, Amazon SNS écrit des journaux pour toutes les distributions réussies.

7. Pour fournir les autorisations requises, utilisez une des méthodes suivantes :

   - Pour créer un nouveau rôle de service, choisissez Créer un nouveau rôle de service puis Créer de nouveaux rôles. Sur la page suivante, choisissez Autoriser pour permettre à Amazon SNS d'accéder en écriture aux ressources de votre compte.
   - Pour utiliser un rôle de service existant, choisissez Utiliser le rôle de service existant, puis collez le nom ARN dans la case Rôle IAM pour les distributions ayant réussi et échoué.

     Le rôle de service que vous spécifiez doit autoriser l'accès en écriture aux ressources de votre compte. Pour de plus amples informations sur la création de rôles IAM, veuillez consulter la section Création d'un rôle pour un AWS service dans le Guide de l'utilisateur IAM.
8. Choisissez Enregistrer les modifications.
9. De retour sur la page Messagerie texte mobile (SMS), accédez à la section Journaux de l'état de distribution pour afficher tous les journaux disponibles.

> **Note**
>
> Selon l'opérateur du numéro de téléphone de destination, l'affichage des journaux de distribution peut durer jusqu'à 72 heures dans la console Amazon SNS.

Exemple de journal pour une distribution SMS réussie

Le journal de l'état de distribution pour une distribution SMS réussie se présente comme l'exemple suivant :

```
{
    "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"
}
```

Exemple de journal pour une distribution SMS ayant échoué

Le journal de l'état de distribution pour une distribution SMS ayant échoué se présente comme l'exemple suivant :

```
{
    "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"
}
```

Causes d'échec de la distribution SMS

La cause d'un échec est fournie avec l'attribut `providerResponse`. La distribution des SMS risque d'échouer pour les raisons suivantes :

- Il est bloqué en tant que courrier indésirable par l'opérateur de téléphonie
- La destination est sur une liste bloquée
- Le numéro de téléphone n'est pas valide
- Le corps du message n'est pas valide
- L'opérateur de téléphonie a bloqué ce message

- L'opérateur de téléphonie est actuellement inaccessible/indisponible
- Le téléphone a bloqué le SMS
- Le téléphone est sur une liste bloquée
- Le téléphone est actuellement inaccessible/indisponible
- Le numéro de téléphone est désactivé
- Cette distribution entraînerait un dépassement du prix maximum
- Erreur inconnue lors de la tentative d'accès au téléphone

Affichage des rapports d'utilisation quotidiens des SMS

Vous pouvez contrôler vos distributions SMS en vous abonnant à des rapports d'utilisation quotidiens à partir d'Amazon SNS. Pour chaque jour où vous envoyez au moins un SMS, Amazon SNS diffuse un rapport d'utilisation sous forme de fichier CSV au compartiment Amazon S3 spécifié. Il faut compter 24 heures pour que le rapport d'utilisation des SMS soit disponible dans le compartiment S3.

Rubriques

Informations sur le rapport d'utilisation quotidien

Le rapport d'utilisation inclut les informations suivantes pour chaque SMS que vous envoyez à partir de votre compte.

Le rapport n'inclut pas les messages envoyés aux destinataires qui se sont désabonnés.

- Heure de publication du message (au format UTC)
- ID de message
- Numéro de téléphone de destination
- Type de message
- Statut de distribution
- Prix du message (en USD)
- Référence (un message est fractionné en plusieurs parties s'il est trop long pour constituer un seul message)
- Nombre total de parties

> **Note**
>
> Si Amazon SNS n'a pas reçu le nombre de parties, nous définissons sa valeur à zéro.

Abonnement à des rapports d'utilisation quotidiens

Pour vous abonner à des rapports d'utilisation quotidiens, vous devez créer un compartiment Amazon S3 avec les autorisations appropriées.

Pour créer un compartiment Amazon S3 pour vos rapports d'utilisation quotidiens

1. À partir de Compte AWS qui envoie des messages SMS, connectez-vous à laConsole Amazon S3.
2. Choisissez Créer un compartiment.
3. Pour Nom du compartiment, nous vous recommandons de saisir un nom unique pour votre compte et votre organisation. Par exemple, utilisez le modèle `<my-bucket-prefix>-<account_id>-<org-id>`.

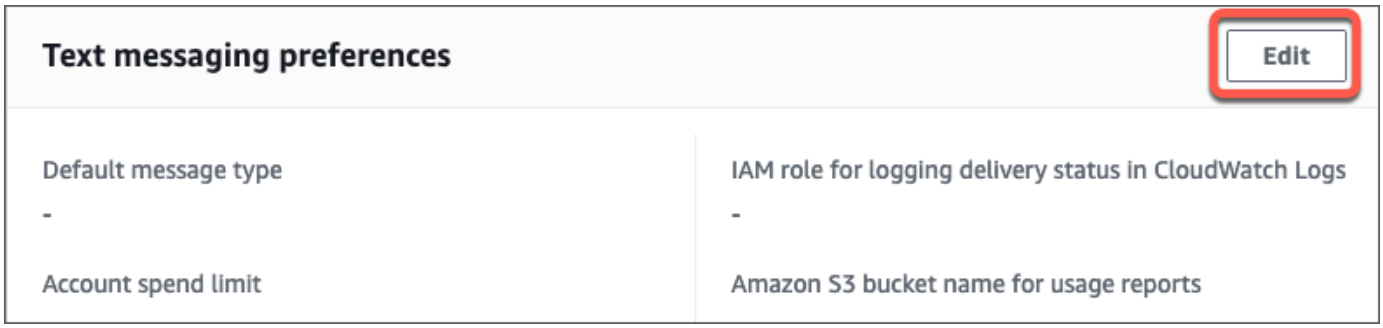
   Pour de plus amples informations sur les conventions et les restrictions pour les noms de compartiments, consultez Règles relatives à l'attribution des noms de compartiments dans le Guide de l'utilisateur du service d'Amazon Simple Storage.
4. Choisissez Create (Créer).
5. Dans le tableau Tous les compartiments, sélectionnez le compartiment.
6. Dans l'onglet Autorisations, sélectionnez politique de compartiment.
7. Dans la fenêtre Éditeur de politique de compartiment, indiquez une politique qui permet au principal du service Amazon SNS d'écrire dans votre compartiment. Pour voir un exemple, consultez Exemple de politique de compartiment.

   Si vous utilisez l'exemple de politique, n'oubliez pas de remplacer *`my-s3-bucket`* par le nom de compartiment que vous avez choisi à l'étape 3.
8. Choisissez Enregistrer.

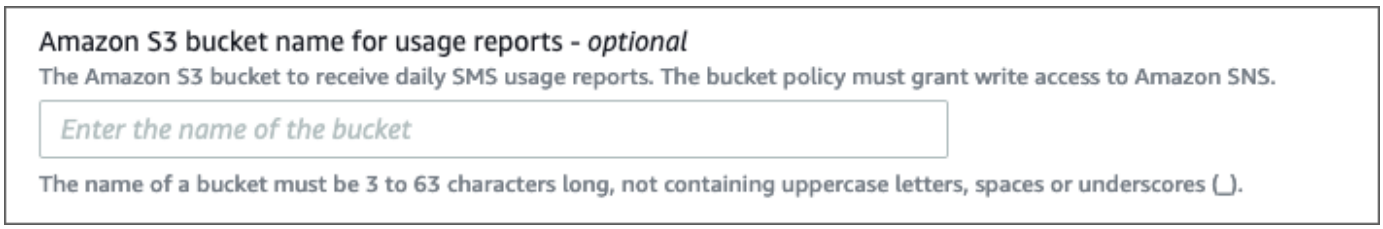
Pour vous abonner à des rapports d'utilisation quotidiens

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Messages texte (SMS).

3. Sur la pageMobile text messaging (SMS) (SMS mobile), dans la section Text messaaging preferences (Préférences de SMS), choisissez Edit (Modifier).



4. Sur la page Edit text messaging preferences (Modifier les préférences de SMS) dans la section Details (Détails), spécifiez le Amazon S3 bucket name for usage reports (Nom du compartiment Amazon S3 pour les rapports d'utilisation).



5. Choisissez Enregistrer les modifications.

Exemple de politique de compartiment

La politique suivante permet au principal du service Amazon SNS d'exécuter les actions `s3:PutObject`, `s3:GetBucketLocation` et `s3:ListBucket`.

AWS fournit des outils pour tous les services avec des principaux de service qui ont reçu l'accès aux ressources dans votre compte. Lorsque le principal dans une déclaration de politique d'un compartiment Amazon S3 est un principal de service AWS, vous pouvez utiliser les clés de condition globales `aws:SourceArn` ou `aws:SourceAccount` pour vous protéger contre le problème de député confus. Pour limiter la région et le compte à partir desquels le compartiment peut recevoir des rapports d'utilisation quotidiens, utilisez `aws:SourceArn` comme indiqué dans l'exemple ci-dessous. Si vous ne souhaitez pas limiter les régions qui peuvent générer ces rapports, utilisez `aws:SourceAccount` pour limiter en fonction du compte qui génère les rapports. Si vous ne connaissez pas l'ARN de la ressource, utilisez `aws:SourceAccount`.

Utilisez l'exemple suivant qui inclut une protection du député confus lorsque vous créez un compartiment Amazon S3 pour recevoir les rapports quotidiens d'utilisation des SMS d'Amazon SNS.

```
{
 "Version": "2008-10-17",
 "Statement": [{
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::my-s3-bucket/*",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowGetBucketLocation",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:GetBucketLocation",
   "Resource": "arn:aws:s3:::my-s3-bucket",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowListBucket",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:ListBucket",
```

```
   "Resource": "arn:aws:s3:::my-s3-bucket",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  }
 ]
}
```

> **Note**
>
> Vous pouvez publier des rapports d'utilisation sur des compartiments Amazon S3 appartenant au Compte AWS qui est spécifié dans l'élément `Condition` de la politique Amazon S3. Pour publier des rapports d'utilisation dans un compartiment Amazon S3 qu'un autre Compte AWS possède, consultez Comment puis-je copier des objets S3 à partir d'un autre Compte AWS?.

Exemple de rapport d'utilisation quotidien

Une fois que vous êtes abonné à des rapports d'utilisation quotidiens, Amazon SNS place chaque jour un fichier CSV avec les données d'utilisation à l'emplacement suivant :

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Chaque fichier peut contenir jusqu'à 50 000 enregistrements. Si les enregistrements d'une journée dépassent ce quota, Amazon SNS ajoute plusieurs fichiers.

L'illustration suivante présente un exemple de rapport :

```
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
```

Gestion des numéros de téléphone et des abonnements SMS

Amazon SNS fournit plusieurs options pour gérer les personnes autorisées à recevoir des messages SMS de votre compte. A une fréquence limitée, vous pouvez réactiver des numéros de téléphone qui ont refusé de recevoir des SMS de votre compte. Pour arrêter l'envoi de messages à des abonnements SMS, vous pouvez supprimer des abonnements ou les rubriques qui effectuent des publications dedans.

Rubriques

Désactivation de la réception des SMS

Selon les lois et réglementations locales en vigueur (par exemple, aux États-Unis et au Canada), les destinataires de SMS peuvent utiliser leurs appareils pour refuser d'en recevoir en répondant au message avec l'une des options suivantes.

- ARRÊTER (français)
- ANNULER
- FIN
- REFUSER
- REFUSER
- SORTIR
- SUPPRIMER
- ARRÊTER
- TD
- SE DÉSABONNER

Pour refuser, le destinataire doit répondre au même numéro d'origine qu'Amazon SNS a utilisé pour envoyer le message. Une fois que vous vous êtes désinscrit, le destinataire ne recevra plus de SMS de votre part, Compte AWS sauf si vous avez indiqué le numéro de téléphone.

Si le numéro de téléphone est abonné à une rubrique Amazon SNS, la désactivation ne supprime pas l'abonnement, mais les SMS ne seront pas diffusés à cet abonnement, sauf si vous réactivez le numéro de téléphone.

Gestion des numéros de téléphone et des abonnements (console)

Vous pouvez utiliser la console Amazon SNS pour contrôler quels numéros de téléphone recevront des SMS de votre compte.

Réactivation d'un numéro de téléphone désactivé

Vous pouvez afficher les numéros de téléphone qui ont refusé de recevoir des SMS de votre compte et vous pouvez les réactiver pour reprendre l'envoi de messages.

Vous ne pouvez réactiver un numéro de téléphone qu'une fois tous les 30 jours.

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, définissez le sélecteur de région sur une région prenant en charge la messagerie SMS.
3. Dans le panneau de navigation, choisissez Messages texte (SMS).
4. Sur la page Messages texte (SMS), choisissez Afficher les numéros de téléphone désabonnés. La page Opted out phone numbers affiche les numéros de téléphone désactivés.
5. Cochez la case correspondant au numéro de téléphone que vous voulez réactiver, puis sélectionnez Opt in. Le numéro de téléphone n'est plus désactivé et recevra les SMS que vous lui envoyez.

Suppression d'un abonnement SMS

Supprimez un abonnement SMS pour arrêter l'envoi de SMS à ce numéro de téléphone lorsque vous effectuez une publication dans vos rubriques.

1. Dans le panneau de navigation, choisissez Abonnements.
2. Cochez les cases correspondant aux abonnements à supprimer. Choisissez ensuite Actions, puis Delete Subscriptions.

3. Dans la fenêtre Supprimer, sélectionnez Supprimer. Amazon SNS supprime l'abonnement et affiche un message de de réussite.

Suppression d'une rubrique

Supprimez une rubrique lorsque vous ne souhaitez plus publier de messages dans ses points de terminaison abonnés.

1. Dans le panneau de navigation, choisissez Rubriques.
2. Cochez les cases correspondant aux rubriques à supprimer. Choisissez ensuite Actions, puis Supprimer des rubriques.
3. Dans la fenêtre Supprimer, sélectionnez Supprimer. Amazon SNS supprime la rubrique et affiche un message de réussite.

Gestion des numéros de téléphone et des abonnements (kits SDK AWS)

Vous pouvez utiliser les AWS SDK pour envoyer des demandes programmatiques à Amazon SNS et gérer les numéros de téléphone autorisés à recevoir des SMS depuis votre compte.

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez la section Fichiers de configuration et d'informations d'identification partagés dans le Guide de référence AWS des SDK et des outils.

Affichage de tous les numéros de téléphone désactivés

Pour afficher tous les numéros de téléphone désactivés, soumettez une demande `ListPhoneNumbersOptedOut` avec l'API Amazon SNS.

Les exemples de code suivants montrent comment utiliser`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Pour répertorier les désactivations des SMS

L'exemple `list-phone-numbers-opted-out` suivant répertorie les numéros de téléphone qui ont désactivé la réception de SMS.

```
aws sns list-phone-numbers-opted-out
```

Sortie :

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK for Java 2.x API.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK for PHP API.

Vérification de la désactivation d'un numéro de téléphone

Pour vérifier si un numéro de téléphone est désactivé, soumettez une demande `CheckIfPhoneNumberIsOptedOut` avec l'API Amazon SNS.

Les exemples de code suivants montrent comment utiliser`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}");
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Pour vérifier la désactivation d'un numéro de téléphone aux SMS

L'`check-if-phone-number-is-opted-out`exemple suivant vérifie si le numéro de téléphone spécifié est désactivé pour ne pas recevoir de SMS en provenance du AWS compte courant.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Sortie :

```
{
    "isOptedOut": false
}
```

- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for JavaScript API.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for PHP API.

Réactivation d'un numéro de téléphone désactivé

Pour réactiver un numéro de téléphone, soumettez une demande `OptInPhoneNumber` avec l'API Amazon SNS.

Vous ne pouvez réactiver un numéro de téléphone qu'une fois tous les 30 jours.

Suppression d'un abonnement SMS

Pour supprimer un abonnement SMS à partir d'une rubrique Amazon SNS, obtenez l'ARN de l'abonnement en soumettant une demande `ListSubscriptions` avec l'API Amazon SNS API, puis transmettez l'ARN à une demande `Unsubscribe`.

Les exemples de code suivants montrent comment utiliser`Unsubscribe`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Désabonnez-vous d'une rubrique à l'aide d'un ARN d'abonnement.

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

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for .NET .

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour se désabonner d'une rubrique

L'exemple `unsubscribe` suivant supprime l'abonnement spécifié d'une rubrique.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Cette commande ne produit aucun résultat.

- Pour plus d'informations sur l'API, consultez Unsubscribe dans la Référence des commandes AWS CLI .

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for JavaScript .

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section Se désabonner de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for PHP .

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence du kit SDK AWS de l'API Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus d'informations sur l'API, consultez Unsubscribe dans la Référence d'API du kit SDK AWS pour SAP ABAP.

Suppression d'une rubrique

Pour supprimer une rubrique ainsi que la totalité de ses abonnements, obtenez l'ARN de la rubrique en soumettant une demande `ListTopics` avec l'API Amazon SNS, puis transmettez l'ARN à la demande `DeleteTopic`.

Les exemples de code suivants montrent comment utiliser`DeleteTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Supprimez une rubrique à l'aide de son ARN de rubrique.

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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour supprimer une rubrique SNS

L'exemple `delete-topic` suivant supprime la rubrique SNS spécifiée.

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

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

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS CLI commandes.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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
  //   }
  // }
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for PHP API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez DeleteTopicle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section de référence du AWS SDK pour l'API SAP ABAP.

Pays et régions pris en charge

> **Important**
>
> À compter du 31 août 2023, un numéro dédié, tel qu'un numéro 10DLC ou un numéro gratuit est requis pour envoyer des SMS aux États-Unis et dans leurs territoires (Guam, Porto Rico, îles Samoa américaines et îles Vierges américaines).

Amazon SNS prend actuellement en charge la messagerie SMS dans les régions suivantes : AWS

Nom de la région	Région	Point de terminaison	Protocole
USA Est (Ohio)	us-east-2	sns.us-east-2.amazonaws.com	HTTP et HTTPS
USA Est (Virginie du Nord)	us-east-1	sns.us-east-1.amazonaws.com	HTTP et HTTPS
USA Ouest (Californie du Nord)	us-west-1	sns.us-west-1.amazonaws.com	HTTP et HTTPS
USA Ouest (Oregon)	us-west-2	sns.us-west-2.amazonaws.com	HTTP et HTTPS
Afrique (Le Cap)	af-south-1	sns.af-south-1.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Hyderabad)	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP et HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Asie-Pacifique (Jakarta)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Melbourne)	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Mumbai)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Osaka)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Singapore)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Sydney)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP et HTTPS
Asie-Pacifique (Tokyo)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP et HTTPS
Canada (Centre)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP et HTTPS
Europe (Francfort)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP et HTTPS
Europe (Irlande)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP et HTTPS
Europe (Londres)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP et HTTPS
Europe (Milan)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP et HTTPS
Europe (Paris)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP et HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Europe (Espagne)	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP et HTTPS
Europe (Stockholm)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP et HTTPS
Europe (Zurich)	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP et HTTPS
Israël (Tel Aviv)	il-central-1	sns.il-central-1.amazonaws.com	HTTP et HTTPS
Moyen-Orient (Bahreïn)	me-south-1	sns.me-south-1.amazonaws.com	HTTP et HTTPS
Moyen-Orient (EAU)	me-central-1	sns.me-central-1.amazonaws.com	HTTP et HTTPS
Amérique du Sud (Sao Paulo)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP et HTTPS
AWS GovCloud (USA Est)	us-gov-east-1	sns. us-gov-east-1. amazonaws.com	HTTP et HTTPS
AWS GovCloud (US-Ouest)	us-gov-west-1	sns. us-gov-west-1. amazonaws.com	HTTP et HTTPS

Vous pouvez utiliser Amazon SNS pour envoyer des SMS dans les pays et régions suivants :

> **Note**
>
> L'utilisation des identifiants d'expéditeur dans les pays pris en charge peut améliorer la distribution des SMS.

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
A						
Afghanistan	AF	93	Non	Non	Oui	Non
Albanie	AL	355	Non	Non	Oui	Non
Algérie	DZ	213	Non	Non	Oui	Non
Andorre	AD	376	Non	Non	Oui	Non
Anguilla	AI	1-264	Non	Non	Oui	Non
Antigua et Barbuda	AG	1-268	Non	Non	Oui	Non
Argentine	AR	54	Oui	Non	Non	Non
Arménie	AM	374	Non	Non	Oui	Non
Aruba	AW	297	Non	Non	Oui	Non
Australie	AU	61	Non	Oui	Enregistrement obligatoire[1]	Oui
Autriche	AT	43	Oui	Oui	Oui	Oui
Azerbaïdjan	AZ	994	Non	Non	Oui	Non
B						
Bahamas	BS	1-242	Non	Non	Non	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Bahreïn	BH	973	Non	Non	Oui	Non
Bangladesh	BD	880	Non	Non	Oui	Non
Barbade	BB	1-246	Non	Non	Oui	Non
Biélorussie	BY	375	Non	Non	Enregistrement obligatoire[1]	Non
Belgique	BE	32	Non	Oui	Non	Oui
Belize	BZ	501	Non	Non	Oui	Non
Bermudes	BM	1-441	Non	Non	Oui	Non
Bhoutan	BT	975	Non	Non	Oui	Non
Bolivie	BO	591	Non	Non	Oui	Non
Bosnie-Herzégovine	BA	387	Non	Non	Oui	Non
Botswana	BW	267	Non	Non	Oui	Non
Brésil	BR	55	Oui	Non	Non	Oui
Brunei	BN	673	Non	Non	Oui	Non
Bulgarie	BG	359	Oui	Non	Oui	Oui
Burkina Faso	BF	226	Non	Non	Oui	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Burundi	BL	257	Non	Non	Oui	Non
C						
Cambodge	KH	855	Non	Non	Oui	Non
Cameroun	CM	237	Non	Non	Oui	Non
Canada	CA	1	Oui	Oui	Non	Oui
Cap-Vert	CV	238	Non	Non	Oui	Non
Iles Caïmans	KY	1-345	Non	Non	Non	Non
République centrafricaine	CF	236	Non	Non	Oui	Non
Tchad	TD	235	Non	Non	Oui	Non
Chili	CL	56	Oui	Oui	Non	Oui
Chine	CN	86	Oui	Non	Non [2]	Oui
Colombie	CO	57	Oui	Oui	Non	Oui
Comores	KM	269	Non	Non	Oui	Non
Iles Cook	CK	682	Non	Non	Oui	Oui
Costa Rica	CR	506	Non	Non	Non	Non
Croatie	HR	385	Non	Non	Oui	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Chypre	CY	357	Non	Non	Oui	Non
Tchéquie (République tchèque)	CZ	420	Non	Oui	Oui	Oui
D						
République démocratique du Congo	CD	243	Non	Non	Oui	Non
Danemark	DK	45	Oui	Oui	Oui	Oui
Djibouti	DJ	253	Non	Non	Oui	Non
Dominique	DM	1-767	Non	Non	Oui	Non
République Dominicaine	DO	1-809, 1-829, 1-849	Oui	Non	Non	Oui
E						
Equateur	EC	593	Oui	Non	Non	Oui
Egypte	EG	20	Oui	Non	Enregistrement obligatoire[1]	Oui
El Salvador	SV	503	Non	Non	Non	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Guinée équatoriale	GQ	240	Non	Non	Oui	Non
Érythrée	ER	291	Non	Non	Oui	Non
Estonie	EE	372	Non	Oui	Oui	Oui
Ethiopie	ET	251	Non	Non	Oui	Non
F						
Iles Féroé	FO	298	Non	Non	Oui	Non
Fidji	FJ	679	Non	Non	Oui	Non
Finlande	FI	358	Oui	Oui	Oui	Oui
France	FR	33	Oui	Non	Oui	Oui
Guyane française	GF	594	Non	Non	Oui	Non
Polynésie française	PF	689	Non	Non	Oui	Non
G						
Gabon	GA	241	Non	Non	Oui	Non
Gambie	GM	220	Non	Non	Oui	Non
Géorgie	GE	995	Non	Non	Oui	Non
Allemagne	DE	49	Oui	Oui	Oui	Oui

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Ghana	GH	233	Non	Non	Oui	Non
Gibraltar	GI	350	Non	Non	Oui	Non
Grèce	GR	30	Non	Non	Oui	Non
Groenland	GL	299	Non	Non	Oui	Non
Grenade	GD	1-473	Non	Non	Oui	Non
Guadeloupe	GP	590	Non	Non	Oui	Non
Guam	GU	1-671	Non	Non	Non	Oui
Guatemala	GT	502	Non	Non	Non	Non
Guernesey	GG	44-1481	Non	Non	Oui	Non
Guinée	GN	224	Non	Non	Oui	Non
Guinée-Bissau	GW	245	Non	Non	Oui	N/A
Guyane	GY	592	Non	Non	Oui	Non
H						
Haïti	H	509	Non	Non	Oui	Non
Honduras	HN	504	Non	Non	Oui	Non
Hong Kong	HK	852	Non	Oui	Oui	Oui
Hongrie	HU	36	Non	Oui	Non	Oui

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
I						
Islande	IS	354	Non	Non	Oui	Non
Inde	IN	91	Oui	Oui[4]	Enregistrement obligatoire[3]	Oui
Indonésie	ID	62	Non	Non	Oui	Non
Irak	IQ	964	Non	Non	Oui	Non
Irlande	IE	353	Non	Oui	Oui	Oui
Île de Man	IM	44-1624	Non	Non	Oui	Non
Israël	IL	972	Non	Oui	Oui	Oui
Italie	IT	39	Oui	Oui	Oui	Oui
Côte d’Ivoire	CI	225	Non	Non	Oui	Non
J						
Jamaïque	JM	1-876	Non	Non	Oui	Non
Japon	JP	81	Oui	Oui	Oui	Oui
Jersey	JE	44-1534	Non	Oui	Oui	Oui
Jordanie	JO	962	Non	Non	Enregistrement obligatoire[1]	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
K						
Kazakhstan	KZ	7	Non	Non	Oui	Non
Kenya	KE	254	Non	Non	Oui	Non
Kosovo	XK	383	Non	Non	Oui	Non
Koweït	KW	965	Non	Non	Enregistrement obligatoire[1]	Non
Kirghizstan	KG	996	Non	Non	Oui	Non
L						
Laos	LA	856	Non	Non	Oui	Non
Lettonie	LV	371	Non	Non	Oui	Non
Liban	LB	961	Non	Non	Oui	Non
Lesotho	LS	266	Non	Non	Oui	Non
Liberia	LR	231	Non	Oui	Non	
Libye	LY	218	Non	Non	Oui	Non
Liechtenstein	LI	423	Non	Non	Oui	Non
Lituanie	LT	370	Non	Oui	Oui	Oui

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Luxembourg	LU	352	Non	Oui	Oui	Oui
M						
Macao	MO	853	Non	Non	Oui	Non
Macédoine	MK	389	Non	Non	Oui	Non
Madagascar	MG	261	Non	Non	Oui	Non
Malawi	MW	265	Non	Non	Oui	Non
Malaisie	MY	60	Oui	Non	Non	Oui
Maldives	MV	960	Non	Non	Oui	Non
Mali	ML	223	Non	Non	Oui	Non
Malte	MT	356	Non	Non	Oui	Non
Îles Marshall	MH	692	Non	Non	Non	Non
Martinique	MQ	596	Non	Non	Oui	Non
Mauritanie	MR	222	Non	Non	Oui	Non
Maurice	MU	230	Non	Non	Oui	Non
Mayotte	YT	262	Non	Non	Oui	Non
Mexique	MX	52	Oui	Non	Non	Oui

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
États fédérés de Micronésie	FM	691	Non	Non	Non	Non
Moldavie	MD	373	Non	Non	Oui	Non
Monaco	MC	377	Non	Non	Non	Non
Mongolie	MN	976	Non	Non	Oui	Non
Monténégro	ME	382	Non	Non	Oui	Non
Montserrat	MS	1-664	Non	Non	Oui	Non
Maroc	MA	212	Oui	Non	Oui	Oui
Mozambique	MZ	258	Non	Non	Non	Non
Birmanie	MM	95	Non	Oui	Oui	Oui
N						
Namibie	NA	264	Non	Non	Oui	Non
Népal	NP	977	Non	Non	Oui	Non
Pays-Bas	NL	31	Oui	Oui	Oui	Oui
Antilles néerlandaises	AN	599	Non	Non	Oui	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Nouvelle-Calédonie	NC	687	Non	Non	Oui	Non
Nouvelle-Zélande [6]	NZ	64	Oui	Non	Non	Oui
Nicaragua	NI	505	Non	Non	Non	Non
Niger	NE	227	Non	Non	Oui	Non
Nigeria	NG	234	Non	Non	Oui	Non
Niué	NU	683	Non	Non	Oui	Non
Norvège	NO	47	Non	Oui	Oui	Oui
O						
Oman	OM	968	Non	Non	Non	N/A
P						
Pakistan	PK	92	Non	Non	Oui	N/A
Palestine	PS	970	Non	Non	Oui	Non
Panama	PA	507	Non	Non	Oui	Non
Papouasie-Nouvelle-Guinée	PG	675	Non	Non	Oui	Non
Paraguay	PY	595	Non	Non	Non	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Pérou	PE	51	Oui	Non	Non	Oui
Philippines	PH	63	Non	Oui	Enregistrement obligatoire[1]	Oui
Pologne	PL	48	Non	Oui	Oui	Oui
Portugal	PT	351	Non	Oui	Oui	Oui
Porto Rico	PR	1-797, 1-939	Non	Non	Non	Oui
Q						
Qatar	QA	974	Non	Non	Enregistrement obligatoire[1]	Non
R						
République du Congo	CG	242	Non	Non	Non	Non
La Réunion (France)	RE	262	Non	Non	Oui	Non
Roumanie	RO	40	Non	Oui	Oui	Oui
Russie	RU	7	Oui	Non	Enregistrement obligatoire[1]	Oui

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Rwanda	RW	250	Non	Non	Oui	Non
S						
Saint Kitts et Nevis	KN	1-869	Non	Non	Non	Non
Sainte-Lucie	LC	1-758	Non	Non	Non	Non
Samoa	WS	685	Non	Non	Oui	Non
Saint-Marin	SM	378	Non	Non	Oui	Non
Sao Tomé-et-Principe	ST	239	Non	Non	Oui	Non
Arabie saoudite	SA	966	Non	Oui[4]	Enregistrement obligatoire[1]	Non
Sénégal	SN	221	Non	Non	Oui	Non
Serbie	RS	381	Non	Non	Oui	Non
Seychelles	SC	248	Non	Non	Oui	Non
Sierra Leone	SL	232	Non	Non	Oui	Non
Singapour	SG	65	Oui	Oui	Oui[5]	Oui
Slovaquie	SK	421	Non	Oui	Oui	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Slovénie	SI	386	Non	Non	Oui	Non
Iles Salomon	SB	677	Non	Non	Oui	Non
Somalie	SO	252	Non	Non	Oui	Non
Afrique du Sud	ZA	27	Oui	Oui	Non	Oui
Corée du Sud	KR	82	Non	Non	Non	Non
Soudan du Sud	SS	211	Non	Non	Oui	Non
Espagne	ES	34	Oui	Oui	Oui	Oui
Sri Lanka	LK	94	Non	Non	Enregistrement obligatoire[1]	Non
Suriname	SR	597	Non	Non	Oui	Non
Swaziland	SZ	268	Non	Non	Oui	Non
Suède	SE	46	Oui	Oui	Oui	Oui
Suisse	CH	41	Non	Oui	Oui	Oui
T						
Taïwan	TW	886	Non	Oui	Non	Oui

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Tadjikistan	TJ	992	Non	Non	Oui	Non
Tanzanie	TX	255	Non	Non	Oui	Non
Thaïlande	TH	66	Non	Oui	Enregistrement obligatoire[1]	Oui
Timor-Leste	TL	670	Non	Non	Oui	Non
Togo	TG	228	Non	Non	Oui	Non
Tonga	TO	676	Non	Non	Oui	Non
Trinidad et Tobago	TT	1-868	Non	Non	Oui	Non
Tunisie	TN	216	Non	Non	Oui	Non
Turquie	TR	90	Non	Non	Enregistrement obligatoire[1]	Non
Turkménistan	TM	993	Non	Non	Non	Non
Iles Turks et Caicos	TC	1-649	Non	Non	Oui	Non
Tuvalu	TC	688	Non	Non	Oui	Non
U						

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Ouganda	UG	256	Non	Non	Oui	Non
Ukraine	UA	380	Non	Oui	Oui	Oui
Émirats arabes unis (EAU)	AE	971	Oui	Oui	Enregistrement obligatoire[1]	Oui
Royaume-Uni	GB	44	Oui	Oui	Oui	Oui
États-Unis	US	1	Oui	Oui	Non	Oui
Uruguay	UY	598	Oui	Non	Non	Oui
Ouzbékistan	UZ	998	Non	Non	Oui	Non
V						
Vanuatu	VU	678	Non	Non	Oui	Non
Venezuela	VE	58	Non	Non	Non	Non
Vietnam	VN	84	Non	Non	Enregistrement obligatoire[1]	Non
Iles Vierges britanniques	VG	1-284	Non	Non	Oui	Non

Pays ou région	Code ISO	Code de numérotation	Codes courts pris en charge	Codes longs pris en charge	Prend en charge les ID expéditeur	Prend en charge les SMS bidirectionnels
Îles Vierges, US	VI	1-340	Non	Non	Non	Oui
W						
X						
Y						
Yémen	YE	967	Non	Non	Oui	Non
Z						
Zambie	ZM	260	Non	Non	Oui	Non
Zimbabwe	ZW	263	Non	Non	Oui	Non

Remarques

1. Les expéditeurs doivent utiliser un ID d'expéditeur alphabétique préenregistré. Pour demander un ID d'expéditeur à AWS Support, voirDemande d'ID expéditeur pour les SMS avec Amazon SNS. Certains pays exigent que les expéditeurs répondent à des exigences spécifiques ou respectent certaines restrictions afin d'obtenir une approbation. Dans ces cas, AWS Support vous pouvez vous contacter pour obtenir des informations supplémentaires après avoir soumis votre demande d'identifiant d'expéditeur.
2. Les expéditeurs doivent utiliser un modèle préenregistré pour chaque type de message qu'ils ont l'intention d'envoyer. Si un expéditeur ne répond pas à cette exigence, ses messages seront bloqués. Pour enregistrer un modèle, ouvrez un dossier SMS Amazon SNS auprès de. AWS Support Lorsque vous créez la demande, fournissez les mêmes informations que pour une demande d'ID d'expéditeur. Pour plus d'informations, consultez Demande d'ID expéditeur pour les

SMS avec Amazon SNS. Certains pays exigent que les expéditeurs répondent à des exigences supplémentaires spécifiques ou respectent certaines restrictions afin d'obtenir une approbation. Dans ces cas, il est AWS Support possible que vous soyez invité à fournir des informations supplémentaires.

> **Note**
>
> Pour envoyer des messages en Chine, vous devez d'abord enregistrer vos modèles AWS Support pour approbation.

3. Les expéditeurs doivent utiliser un ID d'expéditeur alphabétique préenregistré. Des étapes d'enregistrement supplémentaires sont obligatoires. Pour plus d'informations, consultez Exigences relatives aux ID d'expéditeur pour l'Inde.

4. Dans ces pays, les codes longs ne prennent en charge que les messages entrants. En d'autres termes, vous ne pouvez pas utiliser ces codes longs pour envoyer des messages à vos destinataires, mais vous pouvez les utiliser pour recevoir des messages de vos destinataires. Ces codes longs sont utiles pour permettre à vos destinataires de refuser les messages, si vous envoyez des messages à l'aide d'un ID d'expéditeur alphabétique, car les ID d'expéditeur ne prennent en charge que les messages sortants.

5. Amazon SNS peut envoyer du trafic SMS à Singapour à l'aide d'un ID d'expéditeur enregistré auprès du registre SSIR (SMS Sender ID Registry) de Singapour, qui est un registre créé par le conseil IMDA (Info-Communications Media Development Authority) de Singapour. Pour plus d'informations sur les exigences relatives à l'utilisation d'un ID d'expéditeur de Singapour, consultez Exigences relatives aux ID d'expéditeur pour Singapour.

   Vous pouvez également envoyer du trafic SMS à Singapour à l'aide d'ID d'expéditeur non enregistrés ou d'autres types d'identités d'origine tels que des codes courts ou des codes longs.

6. En l'absence d'un code court dédié, Amazon SNS tente toujours d'envoyer les messages aux destinataires néo-zélandais en utilisant un groupe partagé de codes courts. Compte tenu des restrictions des opérateurs locaux concernant les numéros partagés, la délivrabilité avec ces numéros partagés est soumis au principe du meilleur effort. Par conséquent, Amazon SNS recommande vivement de se procurer un code court dédié pour l'ensemble du trafic envoyé vers la Nouvelle-Zélande. Les messages contenant des URL doivent être ajoutés à une liste blanche

via le processus de code court dédié. Pour en savoir plus sur l'achat d'un code court, consultez Demande de codes courts dédiés pour la messagerie SMS avec Amazon SNS .

Bonnes pratiques pour les SMS

Les utilisateurs de téléphonie mobile ont tendance à avoir une tolérance très faible pour les SMS non sollicités. Les taux de réponse aux campagnes SMS non sollicitées sera presque toujours faible, et, par conséquent, le retour sur investissement sera médiocre.

En outre, les opérateurs de téléphonie mobile soumettent les expéditeurs de SMS à des audits permanents. Ils restreignent ou bloquent les messages en provenance des numéros qu'ils considèrent comme envoyant des messages non sollicités.

L'envoi de contenu non sollicité constitue également une violation de la politique d'utilisation acceptable d'AWS. L'équipe Amazon SNS procède à des audits réguliers des campagnes SMS. Elle peut restreindre ou bloquer votre capacité à envoyer des messages s'il s'avère que vous envoyez des messages non sollicités.

Enfin, dans de nombreux pays, régions et juridictions, de lourdes pénalités peuvent être appliquées pour l'envoi de SMS non sollicités. Par exemple, aux États-Unis, le TCPA (Telephone Consumer Protection Act) stipule que les consommateurs peuvent bénéficier de 500 à 1 500 USD de dommages et intérêts (payés par l'expéditeur) pour chaque message non sollicité reçu.

Cette section décrit plusieurs bonnes pratiques qui peuvent vous aider à améliorer votre engagement client et à éviter de lourdes pénalités. Cependant, notez que cette section ne contient pas de conseils juridiques. Consultez toujours un avocat pour obtenir des conseils juridiques.

Rubriques

- Éviter la fatigue multicanal
- Utiliser des codes courts dédiés
- Vérifiez vos numéros de téléphone de destination
- Conception axée sur la redondance
- Limites et restrictions pour les SMS
- Gestion des mots clés de désabonnement
- CreatePool
- PutKeyword
- Gestion des paramètres des numéros
- Limites de caractères des SMS dans Amazon SNS

Respectez les lois, les réglementations et les exigences des opérateurs

Vous pouvez avoir à vous acquitter d'amendes et de pénalités significatives si vous enfreignez les législations et les réglementations des pays ou régions où résident vos clients. Pour cette raison, il est essentiel de comprendre les lois liées à la messagerie SMS de chaque pays ou région dans lesquels vous travaillez.

La liste suivante contient des liens vers les lois essentielles qui s'appliquent aux communications SMS sur les principaux marchés à travers le monde.

- États-Unis : le Telephone Consumer Protection Act de 1991, également connu sous le nom de TCPA, s'applique à certains types de messages SMS. Pour de plus amples informations, veuillez consulter les règles et réglementations sur le site Web de la Federal Communications Commission.
- Royaume-Uni : la Privacy and Electronic Communications (EC Directive) Regulations 2003, également connue sous le nom de PECR, s'applique à certains types de messages SMS. Pour plus d'informations, consultez What are PECR? sur le site Web de l'UK Information Commissioner's Office.
- Union européenne : la directive sur la vie privée et les communications électroniques (2002), parfois connue sous le nom d'ePrivacy Directive, s'applique à certains types de messages SMS. Pour de plus amples informations, veuillez consulter le texte complet de la loi (format PDF) sur le site Web Europa.eu.
- Canada : le Fighting Internet and Wireless Spam Act, plus communément appelé Canada's Antispam Law ou CASL, s'applique à certains types de messages SMS. Pour de plus amples informations, veuillez consulter le texte complet de la loi sur le site Web du Parlement canadien.

- Japon : la loi sur le règlement de transmission des courriers électroniques spécifiques peut s’appliquer à certains types de messages SMS. Pour plus d'informations, consultez la rubrique Japan's Countermeasures Against Spam du site web du ministère japonais de l'intérieur et des communications.

En tant qu'expéditeur, ces lois peuvent s'appliquer à votre cas, même si votre entreprise ou votre organisation n'est pas basée dans l'un de ces pays. Certaines législations dans cette liste ont été initialement créées pour traiter les e-mails ou les appels téléphoniques indésirables, mais ont été interprétées ou étendues pour s’appliquer aussi aux messages SMS. D’autres pays et régions peuvent avoir leurs propres lois liées à la transmission des messages SMS. Consultez un avocat dans chaque pays ou région où vos clients sont situés pour obtenir des conseils juridiques.

Dans de nombreux pays, les opérateurs locaux ont en fin de compte le pouvoir de déterminer le type de trafic qui circule sur leurs réseaux. Cela signifie que les opérateurs peuvent imposer des restrictions sur le contenu des SMS qui dépassent les exigences minimales des lois locales.

Obtenir une autorisation

N'envoyez pas de messages aux destinataires qui n'ont pas demandé explicitement à recevoir les types spécifiques de messages que vous prévoyez d'envoyer. Ne partagez pas de listes d'inscription, y compris entre des organisations de la même entreprise.

Si les destinataires peuvent s'inscrire pour recevoir vos messages via un formulaire en ligne, ajoutez des systèmes pour empêcher les scripts automatisés d'inscrire des personnes à leur insu. Vous devez également limiter le nombre de fois qu'un utilisateur peut soumettre un numéro de téléphone au cours d'une même session.

Lorsque vous recevez une demande d’acceptation de SMS, envoyez au destinataire un message pour l’inviter à confirmer qu’il souhaite recevoir des messages de votre part. N’envoyez aucun message supplémentaire à ce destinataire tant qu'il n’a pas confirmé son abonnement. Un message de confirmation d’abonnement peut se présenter comme l’exemple ci-dessous :

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

Conservez les enregistrements incluant la date, l’heure et la source de chaque demande d’acceptation et de chaque confirmation. Cela peut être utile si un opérateur ou un organisme de régulation les demande ; cela peut également vous aider à effectuer des audits réguliers sur votre liste de clients.

Flux de travaux d'abonnement

Dans certains cas (comme l'enregistrement gratuit ou par code abrégé aux États-Unis), les opérateurs de téléphonie mobile vous demandent de fournir des maquettes ou des captures d'écran de l'ensemble de votre flux de travaux d’abonnement. Les maquettes ou captures d'écran doivent ressembler étroitement au flux de travaux d'abonnement que vos destinataires suivront.

Vos maquettes ou captures d'écran doivent inclure toutes les informations requises indiquées ci-dessous afin de maintenir le plus haut niveau de conformité.

Informations requises

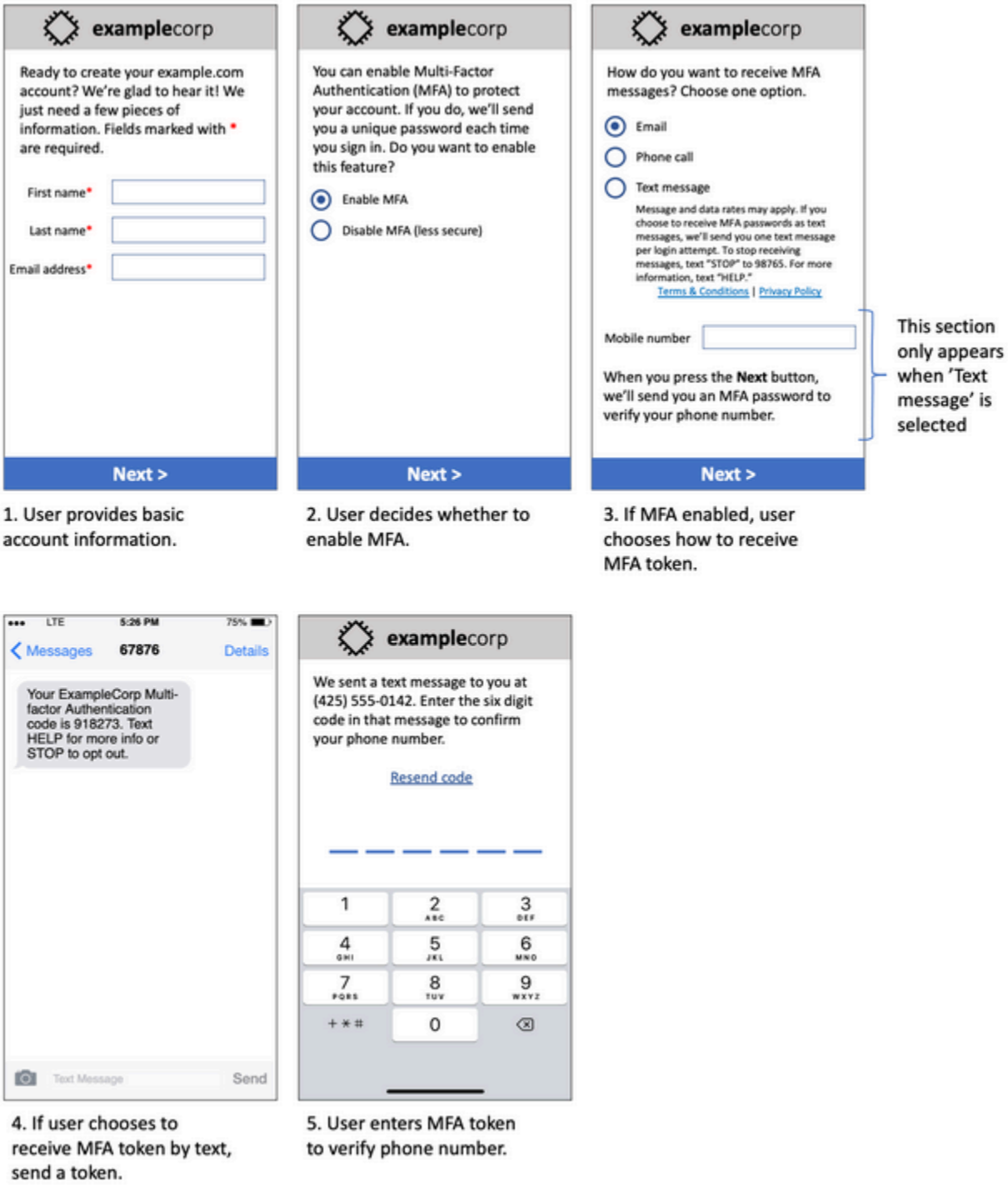
- Une description du cas d'utilisation de la messagerie que vous allez envoyer via votre programme.
- La phrase « Des frais peuvent s'appliquer pour les messages et données. »
- Une indication de la fréquence à laquelle les destinataires recevront des messages de votre part. Par exemple, un programme de messagerie récurrent peut indiquer « un message par semaine ». Un exemple d'utilisation d'un mot de passe unique ou d'une authentification multifactorielle peut indiquer « la fréquence des messages varie » ou « un message par tentative de connexion ».
- Liens vers vos conditions générales et vos documents de politique de confidentialité.

Motifs de rejet courants pour les abonnements non conformes

- Si le nom de l'entreprise fourni ne correspond pas à celui indiqué sur la maquette ou la capture d'écran. Toute relation non évidente doit être expliquée dans la description du flux de travaux d’abonnement.
- S'il apparaît qu'un message sera envoyé au destinataire, mais qu'aucun consentement n'est explicitement recueilli avant de le faire. Le consentement explicite est obligatoire pour tous les messages.
- S'il apparaît que la réception d'un SMS est requise pour s'inscrire à un service. Cela n'est pas conforme si le flux de travaux ne fournit aucune alternative à la réception d'un message d’abonnement sous une autre forme, comme un e-mail ou un appel vocal.
- Si la langue d’abonnement est entièrement présentée dans les conditions d'utilisation. Les informations doivent toujours être présentées au destinataire au moment de l'abonnement plutôt que dans un document de politique associé.
- Si un client a consenti à recevoir un type de message de votre part et que vous lui envoyez d'autres types de SMS. Par exemple, il accepte de recevoir des mots de passe à usage unique, mais reçoit également des messages de sondage.

- Si les informations requises (énumérées ci-dessus) ne sont pas présentées aux destinataires.

L'exemple suivant est conforme aux exigences des opérateurs de téléphonie mobile pour un cas d'utilisation d'authentification multifactorielle.



Maquette d'un cas d'utilisation d'authentification multifactorielle

Elle contient du texte et des images finalisés, et affiche l'intégralité du flux d'abonnement, avec des annotations. Dans le flux de travaux d'abonnement, le client doit prendre des mesures distinctes et intentionnelles pour donner son consentement à recevoir des SMS. Ce flux contient toutes les informations requises.

Autres types de flux de travaux d'abonnement

Les opérateurs de téléphonie mobile accepteront également les flux de travaux d'abonnement en dehors des applications et des sites Web, comme l'abonnement verbal ou écrit, s'ils sont conformes aux éléments décrits ci-dessus. Un flux de travaux d'abonnement conforme et un script verbal ou écrit recueilleront le consentement explicite du destinataire pour recevoir un type de message spécifique. À titre d'exemple, citons le script verbal qu'un agent de support utilise pour recueillir le consentement avant de l'enregistrer dans une base de données de service ou un numéro de téléphone indiqué sur un dépliant promotionnel. Pour fournir une maquette de ces types de flux de travaux d'abonnement, vous pouvez fournir une capture d'écran de votre script d'abonnement, de vos supports marketing ou de votre base de données dans laquelle les chiffres sont collectés. Les opérateurs de téléphonie mobile peuvent avoir des questions supplémentaires concernant ces cas d'utilisation si l'abonnement n'est pas clair ou si le cas d'utilisation dépasse certains volumes.

Ne pas envoyer à d'anciennes listes

Les utilisateurs changent souvent de numéro de téléphone. Un numéro de téléphone dont le propriétaire a accepté d'être contacté il y a deux ans pourrait appartenir à quelqu'un d'autre aujourd'hui. N'utilisez pas une ancienne liste de numéros de téléphone pour un nouveau programme de messages ; si vous le faites, certains messages risquent d'échouer car le numéro n'est plus en service, et certaines personnes se désinscriront parce qu'elles ne se souviennent pas de vous avoir donné leur consentement.

Effectuer un audit de vos listes de clients

Si vous envoyez des campagnes SMS récurrentes, effectuez un audit de vos listes de clients de façon régulière. Effectuer un audit de vos listes de clients permet de vous assurer que seuls les clients intéressés par la réception de vos messages reçoivent ces derniers.

Lorsque vous effectuez un audit de votre liste, envoyez à chaque client un message lui rappelant qu'il s'est abonné, et fournissez-lui les informations requises pour un désabonnement. Un message de rappel peut se présenter comme l'exemple ci-dessous :

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

Archivage des enregistrements

Conservez les enregistrements indiquant à quel moment chaque client a demandé de recevoir des messages SMS de votre part, et quels messages vous avez envoyés à chaque client. De nombreux pays et régions du monde entier imposent aux expéditeurs de SMS de conserver ces enregistrements de manière facilement consultable. Les opérateurs de téléphonie mobile peuvent également vous demander ces informations à tout moment. Les informations exactes que vous devez fournir varient en fonction du pays ou de la région. Pour plus d'informations sur les exigences en matière d'archivage des enregistrements, consultez les réglementations relatives aux messages SMS commerciaux de chaque pays ou région où vos clients se trouvent.

Un opérateur ou un organisme de réglementation nous demande parfois de prouver qu'un client s'est inscrit pour recevoir des messages de votre part. Le cas échéant, AWS Support vous contacte avec une liste des informations requises par l'opérateur ou l'organisme. Si vous ne pouvez pas fournir les informations demandées, nous sommes susceptibles de suspendre votre capacité à envoyer de nouveaux messages SMS.

Faites en sorte que vos messages soient clairs, honnêtes et concis

Le SMS est un support unique. La limite de 160 caractères par message vous impose de rédiger des messages concis. Les techniques que vous pouvez utiliser sur d'autres canaux de communication, comme les e-mails, peuvent ne pas s'appliquer au canal SMS et peuvent même sembler malhonnêtes ou trompeuses lorsqu'elles sont utilisées avec des SMS. Si le contenu de vos messages ne respecte pas les bonnes pratiques, les destinataires peuvent ignorer vos messages ; dans le pire des cas, les opérateurs de téléphonie mobile peuvent identifier vos messages comme du spam et bloquer les futurs messages provenant de votre numéro de téléphone.

Cette section fournit des conseils et des idées pour créer un message SMS efficace.

Identifiez-vous en tant qu'expéditeur

Vos destinataires doivent pouvoir déterminer immédiatement que vous êtes l'auteur du message. Les expéditeurs qui suivent cette bonne pratique incluent un nom d'identification (« nom du programme ») au début de chaque message.

Ne le faites pas :

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Essayez plutôt ceci :

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

N'essayez pas de faire en sorte que votre message ressemble à un message personnel

Certains spécialistes du marketing sont tentés d'ajouter une touche personnelle à leurs SMS, en faisant croire que leurs messages ont été envoyés par une personne spécifique. Cependant, cette technique peut donner l'impression que votre message est une tentative d’hameçonnage.

Ne le faites pas :

```
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.
```

Essayez plutôt ceci :

```
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.
```

Soyez prudent lorsque vous parlez d'argent

Les fraudeurs exploitent souvent la volonté des gens d'économiser et de recevoir de l'argent. Ne faites pas d'offres trop belles pour être vraies. N'utilisez pas l'appât de l'argent pour tromper les gens. N'utilisez pas de symboles monétaires pour parler d’argent.

Ne le faites pas :

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Essayez plutôt ceci :

```
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.
```

Utilisez uniquement les caractères nécessaires

Les marques ont souvent envie de protéger leurs produits en ajoutant des symboles de marque tels que ™ ou ® dans leurs messages. Toutefois, ces symboles ne font pas partie de l'ensemble de caractères standard (appelé alphabet GSM) qui peut être inclus dans un SMS de 160 caractères. Lorsque vous envoyez un message contenant l'un de ces caractères, votre message est automatiquement envoyé à l'aide d'un système de codage de caractères différent, qui ne prend en charge que 70 caractères par message. Par conséquent, votre message peut être divisé en plusieurs parties. Étant donné que vous êtes facturé pour chaque partie que vous envoyez, l'envoi de l'intégralité du message peut vous coûter plus cher. En outre, vos destinataires peuvent recevoir plusieurs messages d'affilée de votre part, au lieu d'un seul message. Pour plus d'informations sur l'encodage de caractères SMS, consultez Limites de caractères des SMS dans Amazon SNS.

Ne le faites pas :

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Essayez plutôt ceci :

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> Les deux exemples précédents sont presque identiques, mais le premier contient un symbole de marque déposée (®), qui ne fait pas partie de l'alphabet GSM. Par conséquent, le premier exemple est envoyé en deux messages, tandis que le second est envoyé en un message.

Utilisez des liens valides et sûrs

Si votre message comporte des liens, vérifiez-les pour vous assurer qu'ils fonctionnent. Testez vos liens sur un appareil en dehors du réseau de votre entreprise pour vous assurer qu'ils sont corrects. En raison de la limite de 160 caractères des SMS, les URL très longues peuvent être réparties entre plusieurs messages. Vous devez utiliser des domaines de redirection pour ajouter des URL abrégées. Toutefois, n'utilisez pas de services gratuits pour raccourcir des liens, comme tinyurl.com ou bitly.com, car les opérateurs ont tendance à filtrer les messages qui contiennent des liens sur

ces domaines. Cependant, vous pouvez utiliser des services payants pour raccourcir vos liens, à condition que ces derniers soient redirigés vers un domaine dont l'usage est exclusivement réservé à votre entreprise ou organisation.

Ne le faites pas :

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Essayez plutôt ceci :

```
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.
```

Limitez le nombre d'abréviations employé

La limite de 160 caractères du canal SMS pousse certains expéditeurs à utiliser des abréviations de manière intensive dans leurs messages. Cependant, l'utilisation excessive d'abréviations peut sembler peu professionnelle à de nombreux lecteurs et peut amener certains utilisateurs à signaler votre message comme spam. Il est tout à fait possible d'écrire un message cohérent sans utiliser trop d'abréviations.

Ne le faites pas :

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Essayez plutôt ceci :

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Répondre de manière appropriée

Lorsqu’un destinataire répond à vos messages, veillez à lui répondre en lui fournissant des informations utiles. Par exemple, lorsqu'un client répond à l'un de vos messages avec le mot-clé « AIDE », envoyez-lui des informations sur le programme auquel il est abonné, le nombre de messages que vous envoyez chaque mois, et les méthodes qui lui permettent de vous contacter afin d’obtenir plus d'informations. La réponse à une demande AIDE peut se présenter comme l’exemple ci-dessous :

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Lorsqu'un client répond avec le mot-clé « STOP », indiquez-lui qu'il ne recevra plus d'autres messages. La réponse à une demande STOP peut se présenter comme l’exemple ci-dessous :

```
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.
```

Ajuster votre envoi en fonction de l’implication

Les priorités de vos clients peuvent évoluer au fil du temps. Si les clients ne considèrent plus vos messages comme utiles, ils peuvent les refuser complètement, voire même les signaler comme messages indésirables. Pour ces raisons, il est important que vous ajustiez vos pratiques d'envoi en fonction de l'implication du client.

Pour les clients qui s’impliquent rarement par rapport à vos messages, vous devez ajuster la fréquence de ces derniers. Par exemple, si vous envoyez des messages hebdomadaires aux clients impliqués, vous pouvez créer un récapitulatif mensuel distinct pour les clients qui sont moins impliqués.

Enfin, supprimez de vos listes de clients ceux qui ne sont pas impliqués du tout. Cette étape permet de prévenir toute frustration provoquée par vos messages de la part des clients. Elle vous permet d’économiser de l’argent et contribue à protéger votre réputation d’expéditeur.

Envoyer à des heures appropriées

Envoyez des messages uniquement pendant les heures normales d’ouverture de bureau. Si vous envoyez des messages à l'heure du dîner ou au milieu de la nuit, il y a de grandes chances que vos clients se désinscrivent de vos listes afin d'éviter d'être dérangés. De plus, cela n'a aucun sens d'envoyer des SMS lorsque vos clients ne peuvent pas y répondre immédiatement.

Si vous envoyez des campagnes ou des parcours à de très grandes audiences, vérifiez les taux de débit pour vos numéros de création. Divisez le nombre de destinataires par votre débit pour déterminer le temps qu'il faudra pour envoyer des messages à tous vos destinataires.

Éviter la fatigue multicanal

Dans vos campagnes, si vous utilisez plusieurs canaux de communication (e-mail, SMS et messages push), n'envoyez pas le même message dans chaque canal. Lorsque vous envoyez le même

message simultanément dans plusieurs canaux, vos clients percevront probablement votre comportement d'envoi comme agaçant plutôt qu'utile.

Utiliser des codes courts dédiés

Si vous utilisez des codes courts, conservez un code court distinct pour chaque marque et chaque type de message. Par exemple, si votre entreprise possède deux marques, utilisez un code court distinct pour chacune d'elles. De même, si vous envoyez des messages promotionnels et des messages transactionnels, utilisez un code court distinct pour chaque type de message. Pour en savoir plus sur la demande de codes courts, consultez Demande de codes courts dédiés pour la messagerie SMS avec Amazon SNS .

Vérifiez vos numéros de téléphone de destination

Lorsque vous envoyez des SMS via Amazon SNS, vous êtes facturé pour chaque partie de message que vous envoyez. Le prix que vous payez par partie de message varie en fonction du pays ou de la région du destinataire. Pour plus d'informations sur la tarification Amazon SNS, consultez Tarifs Amazon SNS.

Lorsqu'Amazon SNS accepte une demande d'envoi d'un message SMS (suite à un appel à l'API SendMessages, ou à la suite du lancement d'une campagne ou d'un parcours), l'envoi de ce message vous est facturé. Cette instruction est vraie même si le destinataire prévu ne reçoit pas réellement le message. Par exemple, si le numéro de téléphone du destinataire n'est plus en service ou si le numéro auquel vous avez envoyé le message n'était pas un numéro de téléphone portable valide, l'envoi du message vous sera toujours facturé.

Amazon SNS accepte les demandes valides d'envoi de SMS et tente de les distribuer. Pour cette raison, vous devez vérifier que les numéros de téléphone auxquels vous envoyez des messages sont des numéros de téléphone mobiles valides. Vous pouvez utiliser le service de validation des numéros de téléphone Amazon SNS pour déterminer si un numéro de téléphone est valide et de quel type de numéro il s'agit (mobile, fixe ou VoIP, par exemple). Pour plus d'informations, consultez la rubrique Validation des numéros de téléphone dans Amazon Pinpoint dans le Guide du développeur Amazon Pinpoint.

Conception axée sur la redondance

Pour les programmes de messagerie stratégiques, nous vous recommandons de configurer Amazon SNS dans plusieurs Région AWS. Amazon SNS est disponible dans plusieurs Régions AWS. Pour obtenir la liste des régions où Amazon SNS est disponible, consultez les Références générales AWS.

Les numéros de téléphone que vous utilisez pour les messages SMS, y compris les codes courts, les codes longs, les numéros gratuits et les numéros 10DLC, ne peuvent pas être répliqués dans les Régions AWS. Par conséquent, pour utiliser Amazon SNS dans plusieurs régions, vous devez demander des numéros de téléphone distincts dans chaque région où vous souhaitez utiliser Amazon SNS. Par exemple, si vous utilisez un code court pour envoyer des SMS à des destinataires aux États-Unis, vous devez demander des codes courts distincts pour chaque Région AWS que vous prévoyez d'utiliser.

Dans certains pays, vous pouvez également utiliser plusieurs types de numéros de téléphone pour une redondance accrue. Par exemple, aux États-Unis, vous pouvez demander des codes courts, des numéros 10DLC et des numéros gratuits. Chacun de ces types de numéros de téléphone emprunte un chemin différent vers le destinataire. Le fait de disposer de plusieurs types de numéros de téléphone, soit dans la même Région AWS, soit sur plusieurs Régions AWS, fournit une couche de redondance supplémentaire, qui peut contribuer à améliorer la résilience.

Limites et restrictions pour les SMS

Pour connaître les limites et les restrictions relatives aux SMS, consultez Limites et restrictions relatives aux SMS dans Amazon Pinpoint dans le Guide de l'utilisateur Amazon Pinpoint.

Gestion des mots clés de désabonnement

Les destinataires de SMS peuvent utiliser leurs appareils pour se désinscrire des messages en répondant avec un mot clé. Pour de plus amples informations, veuillez consulter Désactivation de la réception des SMS.

CreatePool

Utilisez l'action d'API `CreatePool` pour créer un nouveau pool et associer une identité d'origine spécifiée au pool. Pour plus d'informations, consultez CreatePool dans API Amazon Pinpoint SMS and Voice.

PutKeyword

Utilisez l'action d'API `PutKeyword` pour créer ou mettre à jour une configuration de mot clé sur un groupe ou numéro de téléphone d'origine. Pour plus d'informations, consultez PutKeyword dans API Amazon Pinpoint SMS and Voice.

Gestion des paramètres des numéros

Vous pouvez utiliser les options de la section Number settings (Paramètres des numéros) de la page des paramètres des messages SMS et vocaux, afin de gérer les paramètres des codes courts et des codes longs dédiés que vous avez demandés auprès d'AWS Support et attribués à votre compte. Pour plus d'informations, consultez Gestion des paramètres des numéros dans le Guide de l'utilisateur Amazon Pinpoint.

Limites de caractères des SMS dans Amazon SNS

Chaque SMS peut contenir jusqu'à 140 octets d'informations. Le nombre de caractères que vous pouvez inclure dans un seul message SMS dépend du type de caractères figurant dans le message.

Si votre message utilise uniquement des caractères du jeu de caractères GSM 03.38, également connu sous le nom d'alphabet GSM 7 bits, il peut contenir jusqu'à 160 caractères. Si votre message contient des caractères n'appartenant pas au jeu de caractères GSM 03.38, il peut contenir jusqu'à 70 caractères. Lorsque vous envoyez un message SMS, Amazon SNS détermine automatiquement l'encodage le plus efficace à utiliser.

Lorsqu'un message contient plus que le nombre maximal de caractères, il est fractionné. Lorsqu'un messages est fractionné, chacune de ses parties contient des informations complémentaires sur la partie qui la précède. Lorsque l'appareil du destinataire reçoit des parties de message ainsi fractionnées, il utilise ces informations complémentaires pour s'assurer que toutes les parties du message sont affichées dans le bon ordre. Selon l'opérateur mobile et l'appareil du destinataire, plusieurs messages peuvent être affichés comme un seul message ou comme une séquence de messages séparés. Par conséquent, le nombre de caractères dans chaque partie d'un message est réduit à 153 (pour les messages qui contiennent uniquement des caractères GSM 03.38) ou 67 (pour les messages qui contiennent d'autres caractères). Vous pouvez estimer le nombre de parties que contient votre message avant de l'envoyer à l'aide des outils de calcul de la longueur des SMS, dont plusieurs sont disponibles en ligne. La taille maximale prise en charge pour tout message est de 1 600 caractères GSM ou de 630 caractères non GSM. Pour plus d'informations sur le débit et la taille des messages, consultez SMS character limits in Amazon Pinpoint (Limites de caractères des SMS dans Amazon Pinpoint) dans le Guide de l'utilisateur Amazon Pinpoint.

Pour afficher le nombre de parties pour chaque message que vous envoyez, vous devez d'abord activer Event stream settings (Paramètres du flux d'événements). Ce faisant, Amazon SNS produit un événement `_SMS.SUCCESS` lorsque le message est remis au fournisseur mobile du destinataire. L'enregistrement d'événement `_SMS.SUCCESS` contient un attribut appelé

`attributes.number_of_message_parts`. Cet attribut spécifie le nombre de parties contenues dans le message.

> **Important**
>
> Lorsque vous envoyez un message avec plusieurs parties, vous êtes facturé en fonction du nombre de parties contenues dans le message.

Jeu de caractères GSM 03.38

Le tableau suivant répertorie tous les caractères qui sont présents dans le jeu de caractères GSM 03.38. Si vous envoyez un message qui inclut uniquement des caractères indiqués dans le tableau suivant, le message peut contenir jusqu'à 160 caractères.

Caractères standard 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	h/24, j/7	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						

Le jeu de caractères GSM 03.38 inclut plusieurs symboles en plus de ceux indiqués dans le tableau précédent. Cependant, chacun de ces caractères est considéré comme deux caractères, car il comprend également un caractère d'échappement invisible :

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Enfin, le jeu de caractères GSM 03.38 inclut également les caractères non imprimés suivants :

- Un caractère espace.
- Une commande de saut de ligne, ce qui signifie la fin d'une ligne de texte et le début d'une autre.
- Une commande de retour chariot, qui se déplace au début d'une ligne de texte (généralement après un caractère de saut de ligne).
- Une commande d'échappement, qui est automatiquement ajoutée pour les caractères de la liste précédente.

Exemples de messages

Cette section contient plusieurs exemples de messages SMS. Pour chaque exemple, elle affiche le nombre total de caractères, ainsi que le nombre de parties de message pour le message.

Exemple 1 : message long qui ne contient que des caractères de l'alphabet GSM 03.38

Le message suivant contient uniquement des caractères figurant dans l'alphabet 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.
```

Le message précédent contient 180 caractères, il doit donc être fractionné en plusieurs parties. Lorsqu'un message est fractionné, chacune de ses parties peut contenir 153 caractères GSM 03.38. Par conséquent, ce message est envoyé en 2 parties.

Exemple 2 : message contenant des caractères multi-octets

Le message suivant contient plusieurs caractères chinois, dont aucun n’appartient à l'alphabet GSM 03.38.

```
##################################################·####1994#7#########
```

Le message précédent contient 71 caractères. Cependant, étant donné que la majorité des caractères du message n’appartiennent pas à l'alphabet GSM 03.38, il est envoyé en deux parties. Chacune d’elles peut contenir un maximum de 67 caractères.

Exemple 3 : message contenant un seul caractère non GSM

Le message suivant contient un seul caractère qui n’appartient pas à l'alphabet GSM 03.38. Dans cet exemple, le caractère est un guillemet simple (’), c'est-à-dire un caractère différent d'une apostrophe ordinaire ('). Les applications de traitement de texte telles que Microsoft Word remplacent souvent automatiquement les apostrophes par des guillemets simples fermants. Si vous rédigez vos messages SMS dans Microsoft Word et que vous les collez dans Amazon SNS, vous devez supprimer ces caractères spéciaux et les remplacer par des apostrophes.

```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

Le message précédent contient 130 caractères. Cependant, comme il contient le caractère guillemet simple fermant, qui ne fait pas partie de l'alphabet GSM 03.38, il est envoyé en deux parties.

Si vous remplacez le caractère guillemet simple fermant dans ce message par une apostrophe (qui fait partie de l'alphabet GSM 03.38), le message est envoyé en une seule partie.

Notifications push mobile

Amazon SNS vous donne la possibilité d'envoyer des messages de notification push directement à des applications sur des appareils mobiles. Les messages de notification push envoyés à un point de terminaison mobile peuvent apparaître dans l'application mobile en tant que messages d'alerte, mises à jour de badge ou même alertes sonores.

Rubriques

Fonctionnement des notifications utilisateur

Vous envoyez des messages de notification push aux appareils mobiles et aux ordinateurs de bureau, à l'aide de l'un des services de notification push pris en charge suivants :

- Amazon Device Messaging (ADM)
- Apple Push Notification Service (APNs) pour iOS et Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Service de notification push Microsoft pour Windows Phone (MPNS)
- Services de notification push Windows (WNS)

Les services de notification push, tels qu'APNs et FCM, maintiennent une connexion avec chaque application et appareil mobile associé inscrits pour utiliser leur service. Lorsqu'une application et un appareil mobile s'inscrivent, le service de notification push renvoie un jeton d'appareil. Amazon SNS utilise ce jeton d'appareil pour créer un point de terminaison mobile, auquel il peut envoyer directement des messages de notification push. Afin qu'Amazon SNS communique avec les différents services de notification push, soumettez vos informations d'identification de service

de notification push à Amazon SNS pour qu'il les utilise en votre nom. Pour plus d'informations, consultez Présentation du processus de notification utilisateur.

Outre l'envoi direct de messages de notification push, vous pouvez également utiliser Amazon SNS pour envoyer des messages aux points de terminaison mobiles abonnés à une rubrique. Le concept est le même que l'abonnement d'autres types de point de terminaison, comme Amazon SQS, HTTP/S, e-mail et SMS, à une rubrique, comme décrit dans Qu'est-ce qu'Amazon SNS ?. La différence réside dans le fait qu'Amazon SNS communique à l'aide des services de notification push, afin que les points de terminaison mobiles abonnés reçoivent les messages de notification push envoyés à la rubrique.

Présentation du processus de notification utilisateur

1. Obtenez les informations d'identification et le jeton de périphérique pour les plates-formes mobiles que vous souhaitez prendre en charge.
2. Utilisez les informations d'identification pour créer un objet d'application de plate-forme (`PlatformApplicationArn`) à l'aide d'Amazon SNS. Pour plus d'informations, consultez Création d'une application de plateforme.
3. Utilisez les informations d'identification renvoyées pour demander aux plateformes mobiles un jeton de périphérique pour votre application mobile et l'appareil. Le jeton que vous recevez représente votre application mobile et l'appareil.
4. Utilisez le jeton de périphérique et le `PlatformApplicationArn` pour créer un objet de point de terminaison de plate-forme (`EndpointArn`) avec Amazon SNS. Pour plus d'informations, consultez Création d'un point de terminaison de plateforme.
5. Utilisez le `EndpointArn` pour publier un message dans une application sur un appareil mobile. Pour plus d'informations, consultez Publication sur un appareil mobile et l'API Publish dans la Référence d'API Amazon Simple Notification Service.

Configuration d'une application mobile

Cette section explique comment utiliser le AWS Management Console avec les informations décrites dans Conditions préalables pour les notifications utilisateur Amazon SNS pour configurer des applications mobiles.

Rubriques

- Conditions préalables pour les notifications utilisateur Amazon SNS

- Création d'une application de plateforme
- Création d'un point de terminaison de plateforme
- Ajout de jetons d'appareil ou d'ID d'enregistrement
- Méthodes d'authentification Apple
- Méthodes d'authentification Firebase Cloud Messaging (FCM)
- Gestion des terminaux Firebase Cloud Messaging (FCM)

Conditions préalables pour les notifications utilisateur Amazon SNS

Pour commencer à utiliser des notifications push mobile Amazon SNS, vous devez disposer des éléments suivants :

- Un ensemble d'informations d'identification pour la connexion à l'un des services de notification push pris en charge : ADM, APNs, Baidu, FCM, MPNS ou WNS.
- Un jeton d'appareil ou un ID d'enregistrement pour l'application mobile et l'appareil.
- Amazon SNS configuré pour envoyer des messages de notification push aux points de terminaison mobiles.
- Une application mobile enregistrée et configurée pour utiliser l'un des services de notification push pris en charge.

L'enregistrement de votre application auprès d'un service de notification push nécessite plusieurs étapes. Amazon SNS a besoin de certaines des informations que vous fournissez au service de notification push pour envoyer des messages de notification push directs au point de terminaison mobile. En général, vous avez besoin des informations d'identification requises pour la connexion au service de notification push, d'un jeton d'appareil ou d'un ID d'enregistrement (représentant votre appareil mobile et votre application mobile), que vous avez reçu du service de notification push, et de l'application mobile inscrite auprès du service de notification push.

La forme exacte des informations d'identification diffère selon les plateformes mobiles, mais dans tous les cas, ces informations d'identification doivent être soumises lors de l'établissement d'une connexion à la plateforme. Un ensemble d'informations d'identification est émis pour chaque application mobile et doit être utilisé pour envoyer un message à une instance de cette application.

Les noms spécifiques varient en fonction du service de notification push utilisé. Par exemple, lorsque vous utilisez APNs comme service de notifications push, vous avez besoin d'un jeton d'appareil. Parallèlement, avec FCM, l'équivalent du jeton d'appareil est appelé ID d'enregistrement. Le jeton

d'appareil ou l'ID d'enregistrement est une chaîne qui est envoyée à l'application par le système d'exploitation de l'appareil mobile. Ils identifient de façon unique une instance d'une application mobile en cours d'exécution sur un appareil mobile spécifique et peuvent être considérés comme les identifiants uniques de cette paire application-appareil.

Amazon SNS stocke les informations d'identification (plus quelques autres paramètres) sous forme de ressource d'application de plateforme. Les jetons d'appareil (là encore avec certains paramètres supplémentaires) sont représentés en tant qu'objets appelés points de terminaison de plateforme. Chaque point de terminaison de plateforme appartient à une application de plateforme spécifique, et la communication avec chaque point de terminaison de plateforme est possible à l'aide des informations d'identification qui sont stockées dans son application de plateforme correspondante.

Les sections suivantes incluent les conditions préalables pour chacun des services de notification push pris en charge. Une fois que vous avez obtenu les informations prérequises, vous pouvez envoyer un message de notification push à l'aide d'AWS Management Console ou des API push mobile Amazon SNS. Pour plus d'informations, consultez Présentation du processus de notification utilisateur.

Création d'une application de plateforme

Pour qu'Amazon SNS envoie des messages de notification à des points de terminaison mobiles, que ce soit directement ou via des abonnements à une rubrique, vous devez commencer par créer une application de plateforme. Après avoir enregistré l’appli auprès d'AWS, l'étape suivante consiste à créer un point de terminaison pour l'application et l'appareil mobile. Amazon SNS utilise ensuite le point de terminaison pour l'envoi de messages de notification à l'appli et à l'appareil.

Pour créer une application de plateforme

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Mobile (Applications mobiles), puis Push notifications (Notifications push).
3. Dans la section Platform applications (Applications de plateforme), choisissez Create platform application (Créer une application de plateforme).

   Pour obtenir une liste des Régions AWS dans lesquelles vous pouvez créer des applications mobiles, consultez la section Régions prises en charge pour les applications mobiles.
4. Dans Application name (Nom de l’application), saisissez un nom permettant de représenter votre appli.

Les noms d'appli doivent être constitués uniquement de lettres majuscules et minuscules ASCII, de chiffres, de traits de soulignement, de traits d'union et de points. Les noms doivent également contenir entre 1 et 256 caractères.

5. Dans Push notification platform (Plateforme de notification push), sélectionnez la plateforme auprès de laquelle l'appli est enregistrée, puis saisissez les informations d'identification appropriées.

> **Note**
>
> Si vous réutilisez l'une des plateformes Apple Push Notification Service (APN), vous pouvez choisir entre une authentification basée sur un jeton ou un certificat, puis sélectionner Choose file (Choisir un fichier) pour charger le fichier .p8 ou .p12 (exporté à partir de Keychain Access) dans Amazon SNS.

6. Choisissez Créer une application de plateforme.

   L'application est alors inscrite auprès d'Amazon SNS, ce qui crée un objet d'application de plateforme pour la plateforme sélectionnée et renvoie le PlatformApplicationArn correspondant.

Création d'un point de terminaison de plateforme

Lorsqu'une application et un appareil mobile s'inscrivent auprès d'un service de notification push, le service de notification push renvoie un jeton d'appareil. Amazon SNS utilise ce jeton d'appareil pour créer un point de terminaison mobile, auquel il peut envoyer directement des messages de notification push. Pour plus d'informations, veuillez consulter Conditions préalables pour les notifications utilisateur Amazon SNS et Présentation du processus de notification utilisateur.

Cette section décrit l'approche recommandée pour la création d'un point de terminaison de plateforme.

Rubriques

- Création d'un point de terminaison de plateforme
- Code fictif
- AWS Exemple de SDK
- Résolution des problèmes

Création d'un point de terminaison de plateforme

Pour envoyer des notifications à une application à l'aide d'Amazon SNS, le jeton d'appareil de cette application doit d'abord être inscrit auprès d'Amazon SNS grâce à l'appel de l'action de création d'un point de terminaison de plateforme. Cette action utilise l'Amazon Resource Name (ARN) de l'application de plateforme et le jeton d'appareil en tant que paramètres et renvoie l'ARN du point de terminaison de plateforme créé.

L'`CreatePlatformEndpoint`action effectue les opérations suivantes :

- Si le point de terminaison de plateforme existe déjà, ne le recréez pas. Renvoyez à l'appelant l'ARN du point de terminaison de plateforme existant.
- Si le point de terminaison de plateforme existe déjà avec le même jeton d'appareil, mais des paramètres différents, ne le recréez pas. Envoyez une exception à l'appelant.
- Si le point de terminaison de plateforme n'existe pas, créez-le. Renvoyez à l'appelant l'ARN du point de terminaison de plateforme nouvellement créé.

Vous ne devez pas appeler l'action de création de point de terminaison de plateforme à chaque démarrage d'une application, car cette approche ne fournit pas toujours un point de terminaison en état de fonctionnement. Cela peut se produire, par exemple, lorsqu'une application est désinstallée et réinstallée sur le même appareil et que son point de terminaison existe déjà mais est désactivé. Un processus d'inscription réussi doit effectuer les opérations suivantes :

1. S'assurer qu'un point de terminaison de plateforme existe pour cette combinaison application-appareil.
2. Vérifier que le jeton d'appareil dans le point de terminaison de plateforme est le jeton d'appareil valide le plus récent.
3. S'assurer que le point de terminaison de plateforme est activé et prêt à être utilisé.

Code fictif

Le pseudo-code suivant décrit une pratique recommandée pour la création d'un point de terminaison de plateforme en état de fonctionnement, actuel et activé dans une large gamme de conditions de démarrage. Cette approche fonctionne s'il s'agit de l'enregistrement initial de l'application, si le point de terminaison de plateforme pour cette application existe déjà et si le point de terminaison de plateforme est activé, possède le jeton d'appareil correct, etc. Il n'est pas risqué de l'appeler plusieurs

fois d'affilée, car il ne créera pas de points de terminaison de plateforme en double ou ne modifiera pas un point de terminaison de plateforme existant s'il est déjà à jour et activé.

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
  # this is a first-time registration
  call create platform endpoint
  store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
  # the platform endpoint was deleted
  call create platform endpoint with the latest device token
  store the returned platform endpoint ARN
else
  if (the device token in the endpoint does not match the latest one) or
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

Cette approche peut servir chaque fois que l'application veut s'enregistrer ou se ré-enregistrer. Elle peut également être utilisée lors de la notification d'une modification du jeton d'appareil à Amazon SNS. Dans ce cas, il vous suffit d'appeler l'action avec la dernière valeur de jeton d'appareil. Voici quelques points à prendre en compte à propos de cette approche :

- Il existe deux cas dans lesquels elle peut appeler l'action de création de point de terminaison de plateforme. Elle peut être appelée au tout début, lorsque l'application ne connaît pas son propre ARN de point de terminaison de plateforme, comme cela se produit lors d'une inscription initiale. Elle est également appelée si l'action initiale d'obtention des attributs de point de terminaison échoue avec une exception introuvable, comme cela se produit si l'application connaît son ARN de point de terminaison, mais qu'il a été supprimé.
- L'action d'obtention d'attributs de point de terminaison est appelée pour vérifier l'état du point de terminaison de plateforme, même si ce dernier vient d'être créé. Cela se produit lorsque le point de terminaison de plateforme existe déjà mais est désactivé. Dans ce cas, l'action de création de point de terminaison de plateforme réussit, mais n'active pas le point de terminaison de plateforme. Vous devez donc vérifier l'état du point de terminaison de plateforme avant d'indiquer la réussite.

AWS Exemple de SDK

Le code suivant montre comment implémenter le pseudo-code précédent à l'aide des clients Amazon SNS fournis par les AWS SDK.

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

CLI

AWS CLI

Pour créer un point de terminaison d'application de plateforme

L'exemple `create-platform-endpoint` suivant crée un point de terminaison pour l'application de plateforme spécifiée à l'aide du jeton spécifié.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Sortie :

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

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

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Pour plus d’informations, consultez Actions d'API push mobile.

Résolution des problèmes

Appel répété de la création du point de terminaison de plateforme avec un jeton d'appareil obsolète

En particulier pour les points de terminaison FCM, vous pensez peut-être qu'il est préférable de stocker le premier jeton d'appareil émis par l'application, puis d'appeler le point de terminaison de la plateforme de création avec ce jeton d'appareil à chaque fois que l'application démarre. Cela peut sembler correct dans la mesure où l'application n'a pas à gérer l'état du jeton d'appareil et où Amazon SNS met automatiquement à jour le jeton d'appareil à l'aide de sa valeur la plus récente. Cependant, cette solution présente un certain nombre de problèmes graves :

- Amazon SNS s'appuie sur les retours de FCM pour remplacer les jetons d'appareil expirés par de nouveaux jetons d'appareil. FCM conserve les informations concernant les anciens jetons d'appareil un certain temps, mais pas indéfiniment. Une fois que FCM a oublié la connexion entre l'ancien jeton d'appareil et le nouveau, Amazon SNS ne peut plus mettre à jour le jeton d'appareil stocké dans le point de terminaison de plateforme avec la valeur correcte ; au lieu de cela, il désactive simplement le point de terminaison de plateforme.
- L'application de plateforme contient plusieurs points de terminaison de plateforme correspondant au même jeton d'appareil.
- Amazon SNS impose un quota au nombre de points de terminaison de plateforme qui peuvent être créés en commençant par le même jeton d'appareil. Enfin, la création de nouveaux points de terminaison échoue avec une exception de paramètre non valide et le message d'erreur suivant : « This endpoint is already registered with a different token (Ce point de terminaison est déjà enregistré avec un jeton différent) ».

Pour plus d'informations sur la gestion des points de terminaison FCM, consultez. Gestion des terminaux Firebase Cloud Messaging (FCM)

Réactivation d'un point de terminaison de plateforme associé à un jeton d'appareil non valide

Lorsqu'une plateforme mobile (par exemple, APNs ou FCM) informe Amazon SNS que le jeton d'appareil utilisé dans la demande de publication n'était pas valide, Amazon SNS désactive le point de terminaison de plateforme associée à ce jeton d'appareil. Amazon SNS rejettera ensuite les publications ultérieures sur ce jeton d'appareil. Bien que vous puissiez penser qu'il est préférable de simplement réactiver le point de terminaison de plateforme et de continuer la publication, dans la plupart des cas, cela ne fonctionnera pas : les messages qui sont publiés ne sont pas diffusés et le point de terminaison de plateforme est à nouveau désactivé peu de temps après.

En effet, cela vient du fait que le jeton d'appareil associé au point de terminaison de plateforme est réellement non valide. Les diffusions à ce point de terminaison ne peuvent pas réussir, car il ne correspond plus à aucune application installée. Lors de la prochaine publication, la plateforme mobile informera à nouveau Amazon SNS que le jeton d'appareil n'est pas valide et Amazon SNS désactivera à nouveau le point de terminaison de plateforme.

Pour réactiver un point de terminaison de plateforme désactivé, ce dernier doit être associé à un jeton d'appareil valide (avec un appel d'action de définition d'attributs de point de terminaison), puis activé. Les diffusions à ce point de terminaison de plateforme ne réussiront qu'à ce moment-là. La seule fois où la réactivation d'un point de terminaison de plateforme fonctionne sans mise à jour de son

jeton d'appareil est lorsqu'un jeton d'appareil associé à ce point de terminaison qui était non valide redevient valide. Cela peut se produire, par exemple, lorsqu'une application a été désinstallée, puis réinstallée sur le même appareil mobile et reçoit le même jeton d'appareil. L'approche présentée ci-dessus effectue cette opération, en s'assurant de ne réactiver un point de terminaison de plateforme qu'après avoir vérifié que le jeton d'appareil qui lui est associé est le dernier en date disponible.

Ajout de jetons d'appareil ou d'ID d'enregistrement

Lors de l'inscription initiale d'une application et d'un appareil mobile auprès d'un service de notification, tel qu'Apple Push Notification Service (APNs) et Firebase Cloud Messaging (FCM), des jetons d'appareil ou ID d'enregistrement sont renvoyés par le service de notification. Lorsque vous ajoutez les jetons d'appareil ou ID d'enregistrement à Amazon SNS, ils sont utilisés avec l'API `PlatformApplicationArn` pour créer un point de terminaison pour l'application et l'appareil. Lorsqu'Amazon SNS crée le point de terminaison, un paramètre `EndpointArn` est renvoyé. Le paramètre `EndpointArn` permet à Amazon SNS de savoir vers quelle application et quel appareil mobile envoyer le message de notification.

Vous pouvez ajouter des jetons d'appareil et des ID d'enregistrement à Amazon SNS à l'aide des méthodes suivantes :

- Ajout manuel d'un jeton unique à AWS à l'aide d'AWS Management Console
- Chargement de plusieurs jetons à l'aide de l'API `CreatePlatformEndpoint`
- Enregistrement de jetons à partir des appareils qui installeront vos applications ultérieurement

Pour ajouter manuellement un jeton d'appareil ou un ID d'enregistrement

1. Connectez-vous à la console Amazon SNS.
2. Choisissez Mobile, puis Notifications push.
3. Dans la section Applications de plate-forme, sélectionnez votre application, puis choisissez Modifier. Si vous n'avez pas encore créé d'application de plateforme, faites-le dès à présent. Pour obtenir des instructions sur la façon de procéder, veuillez consulter Création d'une application de plateforme.
4. Choisissez Ajouter des points de terminaison.
5. Dans la zone Jeton de point de terminaison, saisissez l'ID de jeton ou d'enregistrement, en fonction du service de notification choisi. Par exemple, avec ADM et FCM, vous devez entrer l'ID d'enregistrement.

6. (Facultatif) Dans la zone User Data (Données d'utilisateur), entrez des informations arbitraires à associer au point de terminaison. Amazon SNS n'utilise pas ces données. Les données doivent être au format UTF-8 et inférieures à 2 Ko.
7. Enfin, choisissez Ajouter des points de terminaison.

   Une fois le point de terminaison créé, vous pouvez soit envoyer des messages directement à un appareil mobile, soit envoyer des messages à des appareils mobiles qui sont abonnés à une rubrique.

Pour charger plusieurs jetons à l'aide de l'API **`CreatePlatformEndpoint`**

Les étapes suivantes montrent comment utiliser l'exemple d'application Java (package `bulkupload`) fourni par AWS pour télécharger plusieurs jetons (jetons d'appareil ou ID d'enregistrement) vers Amazon SNS. Vous pouvez utiliser cet exemple d'application pour vous familiariser avec le chargement des jetons existants.

> **Note**
>
> Les étapes suivantes utilisent l'IDE Eclipse Java. Elles partent du principe que vous avez installé le kit AWS SDK for Java et que vous disposez des informations d'identification de sécurité AWS pour votre compte Compte AWS. Pour de plus amples informations, veuillez consulter AWS SDK for Java. Pour plus d'informations sur les informations d'identification, consultez la section Comment puis-je obtenir les informations d'identification de sécurité ? du document Références générales AWS.

1. Téléchargez et décompressez le fichier snsmobilepush.zip.
2. Créez un projet Java dans Eclipse.
3. Importez le dossier `SNSSamples` dans le répertoire de niveau supérieur du projet Java nouvellement créé. Dans Eclipse, choisissez avec le bouton droit de la souris le nom du projet Java, puis Importer. Développez Général, choisissez Système de fichiers, puis Suivant, naviguez jusqu'au dossier `SNSSamples`, choisissez OK, puis Terminer.
4. Téléchargez une copie de la bibliothèque OpenCSV et ajoutez-la au chemin de génération du package `bulkupload`.
5. Ouvrez le fichier `BulkUpload.properties` contenant le package `bulkupload`.
6. Ajoutez ce qui suit à `BulkUpload.properties` :

- L'`ApplicationArn` auquel vous souhaitez ajouter des points de terminaison.
- Le chemin d'accès absolu de l'emplacement du fichier CSV contenant les jetons.
- Les noms des fichiers CSV (par exemple, `goodTokens.csv` et `badTokens.csv`) qui seront créés pour consigner les jetons analysés correctement par Amazon SNS et ceux qui ont échoué.
- (Facultatif) Caractères qui spécifient le délimiteur et la citation dans le fichier CSV contenant les jetons.
- (Facultatif) Nombre de threads à utiliser pour créer simultanément des points de terminaison. La valeur par défaut est 1 thread.

Votre `BulkUpload.properties` terminé devrait ressembler à ce qui suit :

```
applicationarn:arn:aws:sns:us-west-2:111122223333:app/FCM/fcmpushapp
csvfilename:C:\\mytokendirectory\\mytokens.csv
goodfilename:C:\\mylogfiles\\goodtokens.csv
badfilename:C:\\mylogfiles\\badtokens.csv
delimiterchar:'
quotechar:"
numofthreads:5
```

7. Exécutez l’application BatchCreatePlatformEndpointSample.java pour charger les jetons vers Amazon SNS.

   Dans cet exemple, les points de terminaison créés pour les jetons qui ont été téléchargés correctement vers Amazon SNS sont consignés dans `goodTokens.csv`, tandis que les jetons incorrects sont consignés dans `badTokens.csv`. En outre, des journaux STD OUT doivent être écrits sur la console d'Eclipse, avec un contenu similaire au suivant :

```
<1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-
west-2:111122223333:app/FCM/fcmpushapp/165j2214-051z-3176-b586-138o3d420071
<2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/mytokens.csv
```

Pour enregistrer des jetons à partir des appareils qui installeront vos applications ultérieurement

Vous pouvez utiliser l'une des deux options suivantes :

- Utiliser le service Amazon Cognito : votre application mobile aura besoin d'informations d'identification pour créer les points de terminaison associés à votre application de plateforme Amazon SNS. Nous vous conseillons d'utiliser des informations d'identification temporaires qui expirent après une période donnée. Pour la plupart des scénarios, nous recommandons d'utiliser Amazon Cognito pour créer des informations d'identification de sécurité temporaires. Pour plus d'informations, consultez le Guide du développeur Amazon Cognito. Si vous souhaitez être informé lorsqu'une application s'enregistre auprès d'Amazon SNS, vous pouvez vous inscrire pour recevoir un évènement Amazon SNS, qui fournira le nouvel ARN de point de terminaison. Vous pouvez également utiliser l'API `ListEndpointByPlatformApplication` pour obtenir la liste complète des points de terminaison inscrits auprès d'Amazon SNS.
- Utilisez un serveur proxy : si votre infrastructure d'application est déjà configurée pour que vos applications mobiles soient appelées et enregistrées à chaque installation, vous pouvez continuer à utiliser cette configuration. Votre serveur se comporte comme un proxy et transmet le jeton d'appareil aux notifications push mobile Amazon SNS, ainsi que des données utilisateur que vous souhaitez stocker. Dans cette optique, le serveur proxy se connecte à Amazon SNS à l'aide de vos informations d'identification AWS et utilise l'appel d'API `CreatePlatformEndpoint` pour télécharger les informations de jeton. L'Amazon Resource Name (ARN) de point de terminaison nouvellement créé est renvoyé et votre serveur peut le stocker pour les appels de publication suivants envoyés à Amazon SNS.

Méthodes d'authentification Apple

Vous pouvez autoriser Amazon SNS à envoyer des notifications push à votre appli iOS ou macOS en communiquant des informations qui vous identifient en tant que développeur de l'application. Pour vous authentifier, fournissez une clé ou un certificat lors de la création d'une application de plateforme, que vous pouvez obtenir à partir de votre compte Apple Developer.

Clé de signature de jeton

Une clé de signature privée utilisée par Amazon SNS pour signer les jetons d'authentification du service Apple Push Notification (APN).

Si vous fournissez une clé de signature, Amazon SNS utilise un jeton pour s'authentifier auprès du service APNs chaque fois que vous envoyez une notification push. Avec votre clé de signature, vous pouvez envoyer des notifications push aux environnements de production et de test (sandbox) du service APNS.

Votre clé de signature n'expire pas, et vous pouvez l'utiliser pour plusieurs appli. Pour plus d'informations, consultez la page Communiquer avec le service APNs à l'aide de jetons d'authentification de la section Aide du compte Developer du site Web Apple.

Certificat

Certificat TLS utilisé par Amazon SNS pour s'authentifier auprès du service APNs lorsque vous envoyez des notifications push. Procurez-vous le certificat à partir de votre compte Apple Developer.

Les certificats prennent fin au bout d'un an. Dans ce cas, vous devez créer un certificat et le fournir à Amazon SNS. Pour plus d'informations, consultez la section Établissement d'une connexion basée sur un certificat au service APNs sur le site Web Apple Developer.

Pour gérer les paramètres du service APNs à l'aide de la console de gestion AWS

1. Connectez-vous à la console Amazon SNS.
2. Sous Mobile (Applications mobiles), choisissez Push notifications (Notifications push).
3. Sélectionnez l'Application pour laquelle vous souhaitez modifier les paramètres du service APNs, puis choisissez Edit (Modifier).
4. Sur la page Edit (Modifier), pour Authentication type (Type d'authentification), choisissez Token (Jeton) ou Certificate (Certificat).
5. Chargez les informations d'identification appropriées pour le certificat ou la clé de signature du jeton. Vous pouvez obtenir ces informations à partir de votre compte Apple Developer.
6. Selon le type d'authentification que vous choisissez, procédez comme suit :
   - Si vous choisissez Token (Jeton), fournissez les informations suivantes à partir de votre compte Apple Developer. Amazon SNS a besoin de ces informations pour créer des jetons d'authentification.
     - Signing key (Clé de signature) : clé de signature du jeton d'authentification de votre compte Apple Developer, que vous téléchargez sous forme de fichier .p8. Apple vous autorise à ne télécharger votre clé de signature qu'une seule fois.
     - Signing key ID (ID de clé de signature) : ID attribué à votre clé de signature. Amazon SNS a besoin de ces informations pour créer des jetons d'authentification. Pour trouver cette valeur dans votre compte Apple Developer, choisissez Certificates, IDs & Profiles (Certificats, ID et profils), puis votre clé dans la section Keys (Clés).

- Team identifier (Identifiant d'équipe) : ID attribué à l'équipe chargée de votre compte Apple Developer. Vous trouverez cette valeur sur la page Membership (Adhésion).
- Bundle identifier (Identifiant de solution groupée) : ID attribué à votre appli. Pour trouver cette valeur, choisissez Certificates, IDs & Profiles (Certificats, ID et profils), puis App IDs (ID d'application) dans la section Identifiers (Identifiants), puis choisissez votre appli.

- Si vous choisissez Certificate (Certificat), fournissez les informations suivantes :
  - SSL certificate (Certificat SSL) : fichier .p12 de votre certificat TLS. Vous pouvez exporter ce fichier depuis Keychain Access après avoir téléchargé et installé votre certificat depuis votre compte Apple Developer.
  - Certificate password (Mot de passe du certificat) : si vous avez attribué un mot de passe à votre certificat, spécifiez-le ici.

7. Lorsque vous avez terminé, choisissez Save changes (Enregistrer les modifications).

Méthodes d'authentification Firebase Cloud Messaging (FCM)

Cette rubrique explique comment obtenir les informations d'identification de l'API FCM (HTTP v1) requises auprès de Google pour les utiliser avec l' AWS API, AWS CLI et le AWS Management Console.

Rubriques

> **Important**
>
> 20 juin 2023 — Google a déconseillé son ancienne API HTTP Firebase Cloud Messaging (FCM). Amazon SNS prend désormais en charge la livraison vers tous les types d'appareils à l'aide de l'API FCM HTTP v1. Nous vous recommandons de migrer vos applications push mobiles existantes vers la dernière API HTTP v1 de FCM au plus tard le 1er juin 2024 pour éviter toute interruption.
> 18 janvier 2024 — Amazon SNS a introduit la prise en charge de l'API FCM HTTP v1 pour l'envoi de notifications push mobiles aux appareils Android.

> 26 mars 2024 — Amazon SNS prend en charge l'API FCM HTTP v1 pour les appareils Apple et les destinations Webpush. Nous vous recommandons de migrer vos applications push mobiles existantes vers la dernière API HTTP v1 de FCM au plus tard le 1er juin 2024 afin d'éviter toute interruption des applications.

Vous pouvez autoriser Amazon SNS à envoyer des notifications push à vos applications en communiquant des informations qui vous identifient en tant que développeur de l'application. Pour vous authentifier, fournissez une clé d'API ou un jeton lors de la création d'une application de plateforme. Vous pouvez obtenir les informations suivantes depuis votre console d'application Firebase :

Clé d'API

> La clé d'API est une information d'identification utilisée lors de l'appel de l'ancienne API de Firebase. Les anciennes API FCM seront supprimées par Google le 20 juin 2024. Si vous utilisez actuellement une clé d'API comme information d'dentification de plateforme, vous pouvez mettre à jour les informations d'identification de la plateforme en sélectionnant l'option Jeton et en téléchargeant le fichier JSON associé pour votre application Firebase.

Jeton

> Un jeton d'accès de courte durée est utilisé lors de l'appel de l'API HTTP v1. Il s'agit de l'API suggérée par Firebase pour envoyer des notifications push. Afin de générer des jetons d'accès, Firebase fournit aux développeurs un ensemble d'informations d'identification sous la forme d'un fichier de clé privée (également appelé fichier service.json).

Prérequis

Vous devez obtenir vos informations d'identification FCM service.json pour pouvoir gérer les paramètres FCM dans Amazon SNS. Pour obtenir vos informations d'identification service.json, consultez Migrate from legacy FCM APIs to HTTP v1 dans la documentation de Google Firebase.

Gestion des paramètres FCM (API)

Vous pouvez créer des notifications push FCM à l'aide de l' AWS API. Le nombre et la taille des ressources Amazon SNS d'un AWS compte sont limités. Pour plus d'informations, consultez la section Points de terminaison et quotas Amazon Simple Notification Service dans le Références générales AWS Guide.

Pour créer une notification push FCM associée à une rubrique AWS Amazon SNS (API)

Lorsque vous utilisez des informations d'identification de type clé, les données `PlatformCredential` se présentent sous forme d'une `API key`. Lorsque vous utilisez des informations d'identification de type jeton, les données `PlatformCredential` se présentent sous forme d'un fichier de clé privée au format JSON :

- `CreatePlatformApplication`

Pour récupérer un type d'identifiant FCM pour une rubrique Amazon SNS (API) existante AWS

Récupère le type d'information d'identification `"AuthenticationMethod": "Token"` ou `"AuthenticationMethod": "Key"` :

- GetPlatformApplicationAttributes

Pour définir un attribut FCM pour une rubrique Amazon SNS existante (API AWS)

Définit l'attribut FCM :

- SetPlatformApplicationAttributes

Gestion des paramètres FCM (interface CLI)

Vous pouvez créer des notifications push FCM à l'aide de la AWS Command Line Interface (CLI). Le nombre et la taille des ressources Amazon SNS d'un AWS compte sont limités. Pour plus d'informations, consultez Points de terminaison et quotas Amazon Simple Notification Service.

Pour créer une notification push FCM avec une rubrique Amazon SNS (AWS CLI)

Lorsque vous utilisez des informations d'identification de type clé, les données `PlatformCredential` se présentent sous forme d'une `API key`. Lorsque vous utilisez des informations d'identification de type jeton, les données `PlatformCredential` se présentent sous forme d'un fichier de clé privée au format JSON. Lorsque vous utilisez la AWS CLI, le fichier doit être au format chaîne et les caractères spéciaux doivent être ignorés. Pour formater correctement le fichier, Amazon SNS recommande d'utiliser la commande suivante : `SERVICE_JSON=`jq @json <<< cat service.json``

- create-platform-application

Pour récupérer un type d'information d'identification FCM pour une rubrique Amazon SNS existante (AWS CLI)

Récupère le type d'information d'identification `"AuthenticationMethod": "Token"` ou `"AuthenticationMethod": "Key"` :

- get-platform-application-attributes

Pour définir un attribut FCM pour une rubrique Amazon SNS existante (AWS CLI)

Définit l'attribut FCM :

- set-platform-application-attributes

Gestion des paramètres FCM (console)

Suivez les étapes ci-dessous pour saisir les informations d'identification utilisées par votre application pour se connecter à FCM.

1. Connectez-vous à la console Amazon SNS.
2. Sous Mobile (Applications mobiles), choisissez Push notifications (Notifications push).
3. Sélectionnez une application FCM existante et choisissez Modifier. Si vous n'avez pas encore créé d'application de plateforme, consultez Création d'une application de plateforme.
4. Sur la page Modifier, pour Informations d'indentification Firebase Cloud Messaging, choisissez Jeton ou Clé. Vous pouvez obtenir les informations suivantes à partir de votre console d'application Firebase.
   - Si vous choisissez Jeton, téléchargez un fichier de clé privée valide. Le contenu de ce fichier est utilisé pour générer des jetons d'accès de courte durée lors de l'envoi de notifications.
   - Si vous choisissez Clé, entrez la clé d'API Google.
5. Lorsque vous avez terminé, choisissez Enregistrer les modifications.

Voir aussi

- Utilisation des charges utiles de Google Firebase Cloud Messaging (FCM) v1 dans Amazon SNS

Gestion des terminaux Firebase Cloud Messaging (FCM)

Rubriques

Gestion et maintenance des jetons d'appareils

Vous pouvez garantir la délivrabilité des notifications push de votre application mobile en suivant ces étapes :

1. Stockez tous les jetons de l'appareil, les ARN des points de terminaison Amazon SNS correspondants et les horodatages sur votre serveur d'applications.
2. Supprimez tous les jetons périmés et supprimez les ARN des points de terminaison Amazon SNS correspondants.

Au démarrage initial de votre application, vous recevrez un jeton d'appareil (également appelé jeton d'enregistrement) pour l'appareil. Ce jeton d'appareil est émis par le système d'exploitation de l'appareil et est lié à votre application FCM. Une fois que vous avez reçu ce jeton d'appareil, vous pouvez l'enregistrer auprès d'Amazon SNS en tant que point de terminaison de plateforme. Nous vous recommandons de stocker le jeton de l'appareil, l'ARN du point de terminaison de la plateforme Amazon SNS et l'horodatage en les enregistrant sur votre serveur d'applications ou sur un autre magasin persistant. Pour configurer votre application FCM afin de récupérer et de stocker les jetons d'appareil, consultez la section Récupérer et stocker les jetons d'enregistrement dans la documentation Firebase de Google.

Il est important que vous conserviez up-to-date les jetons. Les jetons de l'appareil de votre utilisateur peuvent changer dans les conditions suivantes :

1. L'application mobile est restaurée sur un nouvel appareil.
2. L'utilisateur désinstalle ou met à jour l'application.
3. L'utilisateur efface les données de l'application.

Lorsque le jeton de votre appareil change, nous vous recommandons de mettre à jour le point de terminaison Amazon SNS correspondant avec le nouveau jeton. Cela permet à Amazon SNS de

poursuivre la communication avec l'appareil enregistré. Vous pouvez le faire en implémentant le pseudo code suivant dans votre application mobile. Il décrit une pratique recommandée pour créer et gérer des points de terminaison de plateforme activés. Cette approche peut être exécutée à chaque démarrage des applications mobiles ou sous forme de tâche planifiée en arrière-plan.

Code fictif

Utilisez le pseudo-code FCM suivant pour gérer et gérer les jetons de l'appareil.

```
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
```

Pour en savoir plus sur les exigences relatives à la mise à jour des jetons, consultez la section Mettre à jour régulièrement les jetons dans la documentation Firebase de Google.

Détection de jetons non valides

Lorsqu'un message est envoyé à un point de terminaison FCM v1 avec un jeton d'appareil non valide, Amazon SNS reçoit l'une des exceptions suivantes :

- `UNREGISTERED`(HTTP 404) — Lorsqu'Amazon SNS reçoit cette exception, vous recevez un événement d'échec de livraison avec un identifiant `FailureType` de`InvalidPlatformToken`, et un jeton de plateforme *`FailureMessage`* of associé au point de terminaison n'est pas valide.

Amazon SNS désactivera le point de terminaison de votre plateforme en cas d'échec d'une livraison, à cette exception près.

- `INVALID_ARGUMENT`(HTTP 400) — Lorsqu'Amazon SNS reçoit cette exception, cela signifie que le jeton de l'appareil ou la charge utile du message n'est pas valide. Pour plus d'informations, consultez ErrorCodela documentation Firebase de Google.

Comme il `INVALID_ARGUMENT` peut être renvoyé dans l'un ou l'autre de ces cas, Amazon SNS renverra un « de »`InvalidNotification`, et le corps *`FailureType`* de notification « *`FailureMessage`* of » n'est pas valide. Lorsque vous recevez cette erreur, vérifiez que votre charge utile est correcte. Si c'est correct, vérifiez que le jeton de l'appareil l'est up-to-date. Amazon SNS ne désactive pas le point de terminaison de votre plateforme en cas d'échec d'une livraison, à cette exception près.

Un autre cas où vous rencontrerez un échec de `InvalidPlatformToken` livraison est lorsque le jeton de l'appareil enregistré n'appartient pas à l'application qui tente d'envoyer ce message. Dans ce cas, Google renverra une erreur SENDER_ID_MISMATCH. Amazon SNS désactivera le point de terminaison de votre plateforme en cas d'échec d'une livraison, à cette exception près.

Tous les codes d'erreur observés reçus de l'API FCM v1 sont disponibles CloudWatch lorsque vous configurez l'enregistrement du statut de livraison pour votre application.

Pour recevoir les événements de livraison de votre application, consultezÉvénements d'application disponibles.

Supprimer les jetons périmés

Les jetons sont considérés comme périmés une fois que les envois de messages au terminal commencent à échouer. Amazon SNS définit ces jetons périmés comme des points de terminaison désactivés pour votre application de plateforme. Lorsque vous publiez sur un point de terminaison désactivé, Amazon SNS renvoie un `EventDeliveryFailure` événement avec le point de terminaison `FailureType` of`EndpointDisabled`, et le point *`FailureMessage`* de terminaison est désactivé. Pour recevoir les événements de livraison de votre application, consultezÉvénements d'application disponibles.

Lorsque vous recevez cette erreur d'Amazon SNS, vous devez supprimer ou mettre à jour le jeton périmé dans l'application de votre plateforme.

Envoi de notifications push mobile

Cette section décrit comment envoyer un message de notification push mobile.

Rubriques

- Publier dans une rubrique
- Publication sur un appareil mobile
- Publication avec des charges utiles spécifiques à la plate-forme

Publier dans une rubrique

Vous pouvez également utiliser Amazon SNS pour envoyer des messages à des points de terminaison mobiles abonnés à une rubrique. Le concept est le même que l'abonnement d'autres types de point de terminaison, comme Amazon SQS, HTTP/S, e-mail et SMS, à une rubrique, comme décrit dans Qu'est-ce qu'Amazon SNS ?. La différence est qu'Amazon SNS communique via des services de notification tels que Apple Push Notification Service (APNS) et Google Firebase Cloud Messaging (FCM). Les points de terminaison mobiles abonnés reçoivent les notifications envoyées à la rubrique grâce au service de notification.

Publication sur un appareil mobile

Vous pouvez envoyer des messages de notification push Amazon SNS directement à un point de terminaison qui représente une application sur un appareil mobile.

Pour envoyer directement un message

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Push notifications (Notifications Push).
3. Sur la page des notifications push mobiles, dans la section Applications de la plateforme, choisissez le nom de l'application, par exemple ***MyApp***.
4. Sur la ***MyApp***page, dans la section Points de terminaison, choisissez un point de terminaison, puis choisissez Publier le message.
5. Sur la page Publier un message vers un point de terminaison, saisissez le message qui s'affiche dans l'application sur l'appareil mobile, puis choisissez Publier un message.

   Amazon SNS envoie le message de notification à la plateforme de notification qui, à son tour, envoie le message à l'application.

Publication avec des charges utiles spécifiques à la plate-forme

Vous pouvez utiliser les API AWS Management Console ou Amazon SNS pour envoyer des messages personnalisés avec des charges utiles spécifiques à la plate-forme vers des appareils mobiles. Pour plus d'informations sur l'utilisation des API Amazon SNS, consultez l'Actions d'API push mobile et le fichier `SNSMobilePush.java` dans `snsmobilepush.zip`.

Rubriques

- Envoi de messages au format JSON
- Envoi de messages propres à une plateforme
- Envoi de messages à une application sur plusieurs plateformes
- Envoi de messages à APNs en tant qu'alertes ou notifications en arrière-plan
- Utilisation des charges utiles de Google Firebase Cloud Messaging (FCM) v1 dans Amazon SNS

Envoi de messages au format JSON

Lors de l'envoi de charges utiles propres à une plateforme, les données doivent être formatées en tant que chaînes de paire valeur clé JSON, avec les guillemets dans une séquence d'échappement.

Les exemples suivants montrent un message personnalisé pour la plateforme FCM.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

Envoi de messages propres à une plateforme

En plus d'envoyer des données personnalisées en tant que paires valeur clé, vous pouvez envoyer des paires valeur clé propres à une plateforme.

L'exemple suivant illustre l'inclusion des paramètres FCM `time_to_live` et `collapse_key` après les paires valeur clé de données personnalisées dans le paramètre FCM `data`.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Pour une liste des paires valeur clé prises en charge dans chacun des services de notification push pris en charge dans Amazon SNS, consultez les liens suivants :

> **Important**
>
> Amazon SNS prend désormais en charge l'API HTTP v1 de Firebase Cloud Messaging (FCM) pour envoyer des notifications push mobiles aux appareils Android.
> 26 mars 2024 — Amazon SNS prend en charge l'API FCM HTTP v1 pour les appareils Apple et les destinations Webpush. Nous vous recommandons de migrer vos applications push mobiles existantes vers la dernière API HTTP v1 de FCM au plus tard le 1er juin 2024 afin d'éviter toute interruption des applications.

- Référence des clés de charge utile dans la documentation APNs
- Protocole HTTP de Firebase Cloud Messaging dans la documentation FCM
- Envoi d'un message dans la documentation ADM

Envoi de messages à une application sur plusieurs plateformes

Pour envoyer un message à une application installée sur les appareils de plusieurs plateformes, comme FCM et APNs, vous devez commencer par abonner les points de terminaison mobiles à une rubrique dans Amazon SNS, puis publier le message dans la rubrique.

L'exemple suivant montre un message à envoyer aux points de terminaison mobiles abonnés sur APNs, FCM et 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\"}}"
}
```

Envoi de messages à APNs en tant qu'alertes ou notifications en arrière-plan

Amazon SNS peut envoyer des messages à APNs en tant que notifications `alert` ou `background` (pour plus d'informations, reportez-vous à la section Transmettre les mises à jour en arrière-plan à votre application dans la documentation APNs).

- Une notification APNs d'`alert` informe l'utilisateur en affichant un message d'alerte, en émettant un avertissement sonore ou en ajoutant un badge à l'icône de votre application.
- Une notification APNs d'`background` réveille votre application ou lui indique d'agir sur le contenu de la notification, sans en informer l'utilisateur.

Spécification des valeurs d'en-tête APNs personnalisées

Nous vous recommandons de spécifier des valeurs personnalisées pour l'attribut de message `AWS.SNS.MOBILE.APNS.PUSH_TYPE` réservé à l'aide de l'action d'`Publish`API Amazon SNS, AWS des SDK ou du. AWS CLI L'exemple DE CLI suivant définit `content-available` pour 1 et `apns-push-type` `background` pour la rubrique spécifiée.

```
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
```

Inférence de l'en-tête de type push APNs à partir de la charge utile

Si vous ne définissez pas l'en-tête APNs `apns-push-type`, Amazon SNS définit l'en-tête sur `alert` ou `background` en fonction de la clé `content-available` dans le dictionnaire `aps` de votre configuration de charge utile APNs au format JSON.

> **Note**
>
> Amazon SNS est capable de déduire uniquement les en-têtes `alert` ou `background`, bien que l'en-tête `apns-push-type` puisse être défini sur d'autres valeurs.

- `apns-push-type` a la valeur `alert`
  - Si le `aps` dictionnaire contient `content-available` défini sur `1` et une ou plusieurs clés qui déclenchent des interactions utilisateur.
  - Si le dictionnaire `aps` contient `content-available` défini sur `0` ou si la clé `content-available` est absente.
  - Si la valeur de la clé `content-available` n'est pas un entier ou un booléen.
- `apns-push-type` a la valeur `background`
  - Si le dictionnaire `aps` contient uniquement la valeur `content-available` définie sur `1` et aucune clé qui déclenchent des interactions utilisateur.

> **Important**
>
> Si Amazon SNS envoie un objet de configuration brut pour APNs en tant que notification en arrière-plan uniquement, vous devez inclure `content-available` défini sur `1` dans le dictionnaire `aps`. Bien que vous puissiez inclure des clés personnalisées, le dictionnaire `aps` ne doit contenir aucune clé qui déclenche des interactions utilisateur (par exemple, des alertes, des badges ou des sons).

Voici un exemple d'objet de configuration brut.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

Dans cet exemple, Amazon SNS définit l'en-tête APNs `apns-push-type` sur `background` pour le message. Lorsqu'Amazon SNS détecte que le dictionnaire `apn` contient la clé `content-available` définie sur `1` et ne contient aucune autre clé pouvant déclencher des interactions utilisateur, il définit l'en-tête sur `background`.

Utilisation des charges utiles de Google Firebase Cloud Messaging (FCM) v1 dans Amazon SNS

Amazon SNS prend en charge l'utilisation de l'API FCM HTTP v1 pour envoyer des notifications vers des destinations Android, iOS et Webpush. Cette rubrique fournit des exemples de structure de charge utile lors de la publication de notifications push mobiles à l'aide de la CLI ou de l'API Amazon SNS.

Vous pouvez inclure les types de messages suivants dans votre charge utile lorsque vous envoyez une notification FCM :

- Message de données : un message de données est géré par votre application cliente et contient des paires clé-valeur personnalisées. Lorsque vous créez un message de données, vous devez inclure la `data` clé avec un objet JSON comme valeur, puis saisir vos paires clé-valeur personnalisées.
- Message de notification ou message d'affichage : un message de notification contient un ensemble prédéfini de touches gérées par le SDK FCM. Ces clés varient en fonction du type d'appareil sur lequel vous livrez. Pour plus d'informations sur les clés de notification spécifiques à la plate-forme, consultez les rubriques suivantes :
  - Clés de notification Android
  - Clés de notification APNS
  - Clés de notification Webpush

Pour plus d'informations sur les types de messages FCM, consultez la section Types de messages dans la documentation Firebase de Google.

Table des matières

Utilisation de la structure de charge utile FCM v1 pour envoyer des messages

Si vous créez une application FCM pour la première fois, ou si vous souhaitez profiter des fonctionnalités de FCM v1, vous pouvez choisir d'envoyer une charge utile au format FCM v1. Pour ce faire, vous devez inclure la clé `fcmV1Message` de niveau supérieur. Pour plus d'informations sur la création de charges utiles FCM v1, consultez Migrer des anciennes API FCM vers HTTP v1

et Personnalisation d'un message sur toutes les plateformes dans la documentation Firebase de Google.

Exemple de charge utile FCM v1 envoyée à Amazon SNS :

> **Note**
>
> La valeur `GCM` clé utilisée dans l'exemple suivant doit être codée sous forme de chaîne lors de la publication d'une notification via 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\",
                }
            }
        }
    }
  }"
}
```

Lorsque vous envoyez une charge utile JSON, veillez à inclure l'`message-structure`attribut dans votre demande et à `json` le définir sur.

Exemple de 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
```

Pour plus d'informations sur l'envoi de charges utiles au format FCM v1, consultez les informations suivantes dans la documentation Firebase de Google :

- Migrer des anciennes API FCM vers HTTP v1
- À propos des messages de la FCM
- Ressource REST : projects.messages

Utilisation de l'ancienne structure de charge utile pour envoyer des messages à l'API FCM v1

Lors de la migration vers FCM v1, il n'est pas nécessaire de modifier la structure de charge utile que vous utilisiez pour vos anciennes informations d'identification. Amazon SNS transforme votre charge utile dans la nouvelle structure de charge utile FCM v1 et l'envoie à Google.

Format de charge utile du message d'entrée :

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

Message envoyé à 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"
    }
  }
}
```

Risques potentiels

- Le mappage de l'ancienne version à la version v1 ne prend pas en charge le service de notification push (APNS) d'Apple `headers` ni les `fcm_options` touches. Si vous souhaitez utiliser ces champs, envoyez une charge utile FCM v1.
- Dans certains cas, les en-têtes de message sont requis par FCM v1 pour envoyer des notifications silencieuses à vos appareils APNs. Si vous envoyez actuellement des notifications silencieuses à vos appareils APNs, elles ne fonctionneront pas avec l'ancienne approche. Nous vous recommandons plutôt d'utiliser la charge utile FCM v1 pour éviter des problèmes inattendus. Pour obtenir la liste des en-têtes APN et leur utilité, consultez la section Communiquer avec les APN dans le manuel Apple Developer Guide.
- Si vous utilisez l'attribut TTL Amazon SNS lors de l'envoi de votre notification, celui-ci ne sera mis à jour que dans le `android` champ. Si vous souhaitez définir l'attribut TTL APNS, utilisez la charge utile FCM v1.
- Les `webpush` touches `androidapns`, et seront mappées et renseignées avec toutes les clés pertinentes fournies. Par exemple, si vous fournissez `title` une clé partagée entre les trois plateformes, le mappage FCM v1 renseignera les trois plateformes avec le titre que vous avez fourni.

- Certaines clés partagées entre plateformes supposent des types de valeur différents. Par exemple, la `badge` clé transmise à `apns` attend une valeur entière, tandis que la `badge` clé transmise à `webpush` attend une valeur String. Dans les cas où vous fournissez la `badge` clé, le mappage FCM v1 ne renseigne que la clé pour laquelle vous avez fourni une valeur valide.

Événements d'échec de livraison du FCM

Le tableau suivant indique le type de défaillance Amazon SNS correspondant aux codes d'erreur/d'état reçus de Google pour les demandes de notification FCM v1. Tous les codes d'erreur observés reçus de l'API FCM v1 sont disponibles CloudWatch lorsque vous configurez l'enregistrement du statut de livraison pour votre application.

Code d'erreur/d'état FCM	Type de panne Amazon SNS	Message d'échec	Cause et atténuation
`UNREGISTERED`	`InvalidPlatformToken`	Le jeton de plateforme associé au point de terminaison n'est pas valide.	Le jeton d'appareil attaché à votre terminal est périmé ou non valide. Amazon SNS a désactivé votre point de terminaison. Mettez à jour le point de terminaison Amazon SNS avec le jeton d'appareil le plus récent.
`INVALID_ARGUMENT`	`InvalidNotification`	Le corps de notification n'est pas valide.	Le jeton ou la charge utile du message de l'appareil n'est peut-être pas valide. Vérifiez que la charge utile de votre message est valide. Si la charge utile du message est valide, mettez à jour

Code d'erreur/d'état FCM	Type de panne Amazon SNS	Message d'échec	Cause et atténuation
			le point de terminaison Amazon SNS avec le jeton d'appareil le plus récent.
SENDER_ID_MISMATCH	InvalidPlatformToken	Le jeton de plateforme associé au point de terminaison n'est pas valide.	L'application de plateforme associée au jeton d'appareil n'est pas autorisée à envoyer au jeton d'appareil. Vérifiez que vous utilisez les informations d'identification FCM correctes dans votre application de plateforme Amazon SNS.
UNAVAILABLE	DependencyUnavailable	La dépendance n'est pas disponible.	FCM n'a pas pu traiter la demande à temps. Toutes les nouvelles tentatives exécutées par Amazon SNS ont échoué. Vous pouvez stocker ces messages dans une file d'attente de lettres mortes (DLQ) et les rediffuser ultérieurement.

Code d'erreur/d'état FCM	Type de panne Amazon SNS	Message d'échec	Cause et atténuation
`INTERNAL`	`UnexpectedFailure`	Défaillance inattendue ; veuillez contacter Amazon. Phrase d'échec [Erreur interne].	Le serveur FCM a rencontré une erreur lors de la tentative de traitement de votre demande. Toutes les nouvelles tentatives exécutées par Amazon SNS ont échoué. Vous pouvez stocker ces messages dans une file d'attente de lettres mortes (DLQ) et les rediffuser ultérieurement.
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	Les informations d'identification de l'application de plateforme ne sont pas valides.	Impossible d'envoyer un message destiné à un appareil iOS ou à un appareil Webpush. Vérifiez que vos informations de développement et de production sont valides.

Code d'erreur/d'état FCM	Type de panne Amazon SNS	Message d'échec	Cause et atténuation
`QUOTA_EXCEEDED`	`Throttled`	Demande limitée par [gcm].	Le quota de débit de messages, le quota de débit de messages de l'appareil ou le quota de débit de messages thématiques a été dépassé. Pour plus d'informations sur la manière de résoudre ce problème, consultez ErrorCode la documentation Firebase de Google.
`PERMISSION_DENIED`	`InvalidNotification`	Le corps de notification n'est pas valide.	Dans le cas d'une `PERMISSION_DENIED` exception, l'appelant (votre application FCM) n'est pas autorisé à exécuter l'opération spécifiée dans la charge utile. Accédez à votre console FCM et vérifiez que les actions d'API requises sont activées sur vos informations d'identification.

Attributs des applications mobile

Amazon Simple Notification Service (Amazon SNS) prend en charge la journalisation de l'état de diffusion des messages de notification push. Une fois que vous avez configuré les attributs d'application, les entrées de journal seront envoyées à CloudWatch Logs pour les messages envoyés aux points de terminaison mobiles à partir d'Amazon SNS. La journalisation du statut de distribution du message permet de fournir des informations opérationnelles plus précises, par exemple :

- Savoir si un message de notification push a été diffusé au service de notification push à partir d'Amazon SNS.
- Identifier la réponse envoyée par le service de notification push à Amazon SNS.
- Déterminer la durée de conservation du message (la durée entre l'horodatage de publication et juste avant sa remise à un service de notification push).

Pour configurer les attributs d'application pour l'état de diffusion du message, vous pouvez utiliser la AWS Management Console, les kits de développement logiciel (SDK) AWS ou l'API de requête.

Rubriques

Configuration des attributs de l'état de diffusion du message à l'aide de AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Mobile, puis Notifications Push.
3. Depuis la section Applications de plateforme, choisissez l'application qui contient les points de terminaison pour lesquels vous voulez recevoir les CloudWatch Logs.
4. Choisissez Actions sur l'application, puis Statut de diffusion.
5. Dans la boîte de dialogue État de la distribution, sélectionnez Créer des rôles IAM.

   Vous serez ensuite redirigé vers la console IAM.

6. Sélectionnez Autoriser pour permettre à Amazon SNS d'accéder en écriture aux CloudWatch Logs en votre nom.
7. Maintenant, de retour dans la boîte de dialogue Statut de diffusion, saisissez un nombre dans le champ Pourcentage de réussites à échantillonner (0 à 100) pour indiquer le pourcentage de messages envoyés pour lequel vous souhaitez recevoir des CloudWatch Logs.

> **Note**
>
> Une fois que vous avez configuré les attributs d'application pour l'état de diffusion du message, toutes les diffusions de message ayant échoué génèrent des CloudWatch Logs.

8. Enfin, sélectionnez Enregistrer la configuration. Vous êtes maintenant en mesure d'afficher et d'analyser les CloudWatch Logs contenant l'état de diffusion du message. Pour plus d'informations sur l'utilisation de CloudWatch, consultez la documentation CloudWatch.

Exemples de journal CloudWatch d'état de remise des messages Amazon SNS

Une fois que vous avez configuré les attributs de l'état de diffusion du message pour un point de terminaison d'application, des CloudWatch Logs sont générés. Les exemples de journaux, au format JSON, se présentent comme suit :

SUCCÈS


```
{
  "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"
  }
}
```

ÉCHEC

```
{
  "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"
  }
}
```

Pour obtenir une liste des codes de réponse du service de notification push, consultez la page Codes de réponse de la plateforme.

Configuration des attributs de l'état de diffusion du message avec les kits SDK AWS

Les kits SDK AWS fournissent des API dans plusieurs langages pour utiliser les attributs de l'état de diffusion du message avec Amazon SNS.

L'exemple Java suivant montre comment utiliser l'API `SetPlatformApplicationAttributes` pour configurer les attributs d'application pour l'état de diffusion des messages de notification push. Vous pouvez utiliser les attributs suivants pour l'état de diffusion du message : `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn` et `SuccessFeedbackSampleRate`. Les attributs `SuccessFeedbackRoleArn` et `FailureFeedbackRoleArn` sont utilisés pour permettre à Amazon SNS d'accéder en écriture aux CloudWatch Logs en votre nom. L'attribut `SuccessFeedbackSampleRate` permet de spécifier le pourcentage de la fréquence d'échantillonnage (0-100) des messages diffusés avec succès. Une fois que vous avez configuré

l'attribut `FailureFeedbackRoleArn`, toutes les diffusions de message qui ont échoué génèrent des CloudWatch Logs.

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
 SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-
west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

Pour plus d'informations sur le SDK pour Java, consultez Mise en route avec le kit AWS SDK for Java.

Codes de réponse de la plateforme

Voici une liste de liens pour les codes de réponse du service de notification push :

Service de notification push	Codes de réponse
Amazon Device Messaging (ADM)	Consultez Format de la réponse dans la documentation ADM.
Apple Push Notification service (APNs).	Consultez HTTP/2 Response from APNs dans la section Communicating with APNs du manuel Local and Remote Notification Programming Guide.
Firebase Cloud Messaging (FCM)	Consultez Codes de réponse aux messages d'erreur en aval dans la documentation Firebase Cloud Messaging.
Service de notification push Microsoft pour Windows Phone (MPNS)	Consultez Codes de réponse du service de notification push pour Windows Phone 8 dans la documentation Windows 8 Development.

Service de notification push	Codes de réponse
Services de notification push Windows (WNS)	Consultez « Codes de réponse » dans En-têtes de demande et de réponse du service de notification Push (applications d'exécution Windows) dans la documentation Windows 8 Development.

Évènements d'application mobile

Amazon SNS prend en charge le déclenchement de notifications lorsque certains évènements d'application se produisent. Vous pouvez ensuite effectuer une action par programmation sur cet événement. Votre application doit prendre en charge un service de notification push tel qu'Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) et Windows Push Notification Services (WNS). Vous définissez les notifications d'événements d'application à l'aide de la console Amazon SNS ou AWS CLI des AWS SDK.

Rubriques

- Événements d'application disponibles
- Envoi de notifications push mobile

Événements d'application disponibles

Les notifications d'événements d'application suivent la création, la suppression et la mise à jour de points de terminaison de plateforme individuels, ainsi que les échecs de diffusion. Voici les noms d'attributs pour les événements d'application.

Nom d'attribut	Déclencheur de notification
`EventEndpointCreated`	Un nouveau point de terminaison de plateforme est ajouté à votre application.
`EventEndpointDeleted`	Tous les points de terminaison de plateforme associés à votre application sont supprimés.

Nom d'attribut	Déclencheur de notification
`EventEndpointUpdated`	Tous les attributs des points de terminaison de plateforme associés à votre application sont modifiés.
`EventDeliveryFailure`	Une diffusion vers n'importe quel point de terminaison de plateforme associé à votre application fait l'objet d'une utilisation hors limites permanente. **Note** Pour suivre les échecs de diffusion du côté application de plateforme, abonnez-vous aux événements d'état de diffusion du message pour l'application. Pour plus d'informations, consultez la page Utilisation des attributs d'application Amazon SNS pour le statut de distribution du message.

Vous pouvez associer n'importe quel attribut avec une application, qui peut alors recevoir ces notifications d'événements.

Envoi de notifications push mobile

Pour envoyer des notifications d'événements d'application, spécifiez une rubrique qui recevra les notifications pour chaque type d'événement. Lorsqu'Amazon SNS envoie les notifications, la rubrique peut les acheminer à des points de terminaison qui effectueront l'action par programmation.

Important

Des applications à volume élevé créeront un grand nombre de notifications d'événements d'application (par exemple, des dizaines de milliers), qui envahiront les points de terminaison destinés à une utilisation humaine, tels que les adresses e-mail, les numéros de téléphone et les applications mobiles. Respectez les consignes suivantes lorsque vous envoyez des notifications d'événements d'application à une rubrique :

- Chaque rubrique recevant des notifications ne doit contenir que des abonnements pour les points de terminaison programmatiques, tels que les points de terminaison HTTP ou HTTPS, les files d'attente Amazon SQS ou les fonctions. AWS Lambda

- Pour réduire le volume de traitement qui est déclenché par les notifications, limitez les abonnements de chaque rubrique à un petit nombre (par exemple, cinq ou moins).

Vous pouvez envoyer des notifications d'événements d'application à l'aide de la console Amazon SNS, du AWS Command Line Interface (AWS CLI) ou des AWS SDK.

AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Mobile, Notifications Push.
3. Sur la page des notifications push mobiles, dans la section Applications de la plateforme, choisissez une application, puis sélectionnez Modifier.
4. Développez la section Notifications d'événement.
5. Choisissez Actions, Configurer les évènements.
6. Saisissez les ARN pour les rubriques à utiliser pour les événements suivants :
   - Point de terminaison créé
   - Point de terminaison supprimé
   - Point de terminaison mis à jour
   - Échec de diffusion
7. Sélectionnez Enregistrer les modifications.

AWS CLI

Exécutez la commande set-platform-application-attributes.

L'exemple suivant définit la même rubrique Amazon SNS pour les quatre évènements d'application :

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
```

```
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDK

Définissez les notifications d'événements d'application en soumettant une `SetPlatformApplicationAttributes` demande à l'aide de l'API Amazon SNS à l'aide d'un AWS SDK.

Pour obtenir la liste complète des guides du développeur et des exemples de code du AWS SDK, y compris l'aide au démarrage et des informations sur les versions précédentes, consultezUtilisation d'Amazon SNS avec un SDK AWS.

Actions d'API push mobile

Pour utiliser les API push mobile Amazon SNS, vous devez d'abord satisfaire aux conditions préalables requises pour le service de notification push, tel que Apple Push Notification Service (APNs) ou Firebase Cloud Messaging (FCM). Pour obtenir plus d'informations sur les conditions préalables, consultez la page Conditions préalables pour les notifications utilisateur Amazon SNS.

Pour envoyer un message de notification push à une application mobile et un appareil à l'aide des API, vous devez d'abord utiliser l'action `CreatePlatformApplication`, qui renvoie un attribut `PlatformApplicationArn`. L'attribut `PlatformApplicationArn` est ensuite utilisé par `CreatePlatformEndpoint`, qui renvoie un attribut `EndpointArn`. Vous pouvez alors utiliser l'attribut `EndpointArn` avec l'action `Publish` pour envoyer un message de notification à une application mobile et un appareil, ou vous pouvez utiliser l'attribut `EndpointArn` avec l'action `Subscribe` pour l'abonnement à une rubrique. Pour de plus amples informations, veuillez consulter . Présentation du processus de notification utilisateur.

Les API push mobile Amazon SNS sont les suivantes :

`CreatePlatformApplication`

> Crée un objet d'application de plateforme pour l'un des services de notification push pris en charge, tels qu'APNs et FCM, auxquels les appareils et les applications mobiles peuvent s'inscrire. Renvoie un attribut `PlatformApplicationArn`, qui est utilisé par l'action `CreatePlatformEndpoint`.

`CreatePlatformEndpoint`

Crée un point de terminaison pour un appareil et une application mobile sur l'un des services de notification push pris en charge. `CreatePlatformEndpoint` utilise l'attribut `PlatformApplicationArn` renvoyé par l'action `CreatePlatformApplication`. L'attribut `EndpointArn`, qui est renvoyé lors de l'utilisation de `CreatePlatformEndpoint`, est ensuite utilisé avec l'action `Publish` pour envoyer un message de notification à une application mobile et un appareil.

`CreateTopic`

Crée une rubrique dans laquelle des messages peuvent être publiés.

`DeleteEndpoint`

Supprime le point de terminaison pour un appareil et une application mobile sur l'un des services de notification push pris en charge.

`DeletePlatformApplication`

Supprime un objet d'application de plateforme.

`DeleteTopic`

Supprime une rubrique et tous ses abonnements.

`GetEndpointAttributes`

Récupère les attributs de point de terminaison pour un appareil et une application mobile.

`GetPlatformApplicationAttributes`

Récupère les attributs de l'objet d'application de plateforme.

`ListEndpointsByPlatformApplication`

Répertorie les points de terminaison et leurs attributs pour des appareils et des applications mobiles dans un service de notification push pris en charge.

`ListPlatformApplications`

Répertorie les objets d'application de plateforme pour les services de notification push pris en charge.

`Publish`

Envoie un message de notification à tous les points de terminaison abonnés à une rubrique.

`SetEndpointAttributes`

Définit les attributs d'un point de terminaison pour un appareil et une application mobile.

`SetPlatformApplicationAttributes`

Définit les attributs de l'objet d'application de plateforme.

`Subscribe`

Prépare l'abonnement d'un point de terminaison en lui envoyant un message de confirmation. Pour créer réellement un abonnement, le propriétaire du point de terminaison doit appeler l'action ConfirmSubscription avec le jeton provenant du message de confirmation.

`Unsubscribe`

Supprime un abonnement.

Erreurs d'API push mobile

Les erreurs qui sont renvoyées par les API Amazon SNS pour push mobile sont répertoriées dans le tableau suivant. Pour plus d'informations sur les API Amazon SNS pour push mobile, consultez la page Actions d'API push mobile.

Erreur	Description	Code d'état HTTPS	Action d'API
Le nom de l'application est une chaîne nulle	Le nom d'application requis est défini sur null.	400	`CreatePlatformApplication`
Platform Name is null string	Le nom de plateforme requis est défini sur null.	400	`CreatePlatformApplication`
Platform Name is invalid	Une valeur non valide ou hors plage a été fournie pour le nom de plateforme.	400	`CreatePlatformApplication`
APNs — Principal is not a valid certificate	Un certificat non valide a été fourni pour le principal	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
	APNs, qui est le certificat SSL. Pour plus d'informations, consultez CreatePlatformApplication dans la Référence de l'API d'Amazon Simple Notification Service.		
APNs — Principal is a valid cert but not in a .pem format	Un certificat valide qui n'est pas au format .pem a été fourni pour le principal APNs, qui est le certificat SSL.	400	`CreatePlatformApplication`
APNs — Principal is an expired certificate	Un certificat arrivé à expiration a été fourni pour le principal APNs, qui est le certificat SSL.	400	`CreatePlatformApplication`
APNs — Principal is not an Apple issued certificate	Un certificat non émis par Apple a été fourni pour le principal APNs qui est le certificat SSL.	400	`CreatePlatformApplication`
APNs — Principal is not provided	Le principal APNs, qui est le certificat SSL, n'a pas été fourni.	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
APNs — Credential is not provided	Les informations d'identification APNs, qui sont la clé privée, n'ont pas été fournies. Pour de plus amples informations, consultez CreatePlatformApplication dans la Référence de l'API d'Amazon Simple Notification Service.	400	`CreatePlatformApplication`
APNs — Credential are not in a valid .pem format	Les informations d'identification APNs, qui sont la clé privée, ne sont pas dans un format .pem valide.	400	`CreatePlatformApplication`
FCM — serverAPIKey is not provided	Les informations d'identification FCM, qui sont la clé API, n'ont pas été fournies. Pour plus d'informations, consultez CreatePlatformApplication dans la Référence de l'API d'Amazon Simple Notification Service.	400	`CreatePlatformApplication`
FCM — serverAPIKey is empty	Les informations d'identification FCM, qui sont la clé API, sont vides.	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
FCM — serverAPIKey is a null string	Les informations d'identification FCM, qui sont la clé API, sont null.	400	`CreatePlatformApplication`
FCM — serverAPIKey is invalid	Les informations d'identification FCM, qui sont la clé API, ne sont pas valides.	400	`CreatePlatformApplication`
ADM — clientsecret is not provided	Le code secret du client requis n'est pas fourni.	400	`CreatePlatformApplication`
ADM — clientsecret is a null string	La chaîne requise pour le code secret du client est null.	400	`CreatePlatformApplication`
ADM — client_secret is empty string	La chaîne requise pour le code secret du client est vide.	400	`CreatePlatformApplication`
ADM — client_secret is not valid	La chaîne requise pour le code secret du client n'est pas valide.	400	`CreatePlatformApplication`
ADM — client_id is empty string	La chaîne requise pour l'ID client est vide.	400	`CreatePlatformApplication`
ADM — clientId is not provided	La chaîne requise pour l'ID client n'est pas fournie.	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
ADM — clientid is a null string	La chaîne requise pour l'ID client est null.	400	`CreatePlatformApplication`
ADM — client_id is not valid	La chaîne requise pour l'ID client n'est pas valide.	400	`CreatePlatformApplication`
EventEndpointCreated has invalid ARN format	Le format d'ARN pour EventEndpointCreated n'est pas valide.	400	`CreatePlatformApplication`
EventEndpointDeleted has invalid ARN format	Le format d'ARN pour EventEndpointDeleted n'est pas valide.	400	`CreatePlatformApplication`
EventEndpointUpdated has invalid ARN format	Le format d'ARN pour EventEndpointUpdated n'est pas valide.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure has invalid ARN format	Le format d'ARN pour EventDeliveryAttemptFailure n'est pas valide.	400	`CreatePlatformApplication`
EventDeliveryFailure has invalid ARN format	Le format d'ARN pour EventDeliveryFailure n'est pas valide.	400	`CreatePlatformApplication`
EventEndpointCreated is not an existing Topic	La rubrique EventEndpointCreated n'existe pas.	400	`CreatePlatformApplication`
EventEndpointDeleted is not an existing Topic	La rubrique EventEndpointDeleted n'existe pas.	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
EventEndpointUpdated is not an existing Topic	La rubrique EventEndpointUpdated n'existe pas.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure is not an existing Topic	La rubrique EventDeliveryAttemptFailure n'existe pas.	400	`CreatePlatformApplication`
EventDeliveryFailure is not an existing Topic	La rubrique EventDeliveryFailure n'existe pas.	400	`CreatePlatformApplication`
Platform ARN is invalid	L'ARN de plateforme n'est pas valide.	400	`SetPlatformAttributes`
Platform ARN is valid but does not belong to the user	L'ARN de plateforme est valide, mais n'appartient pas à l'utilisateur.	400	`SetPlatformAttributes`
APNs — Principal is not a valid certificate	Un certificat non valide a été fourni pour le principal APNs, qui est le certificat SSL. Pour plus d'informations, consultez CreatePlatformApplication dans la Référence de l'API d'Amazon Simple Notification Service.	400	`SetPlatformAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
APNs — Principal is a valid cert but not in a .pem format	Un certificat valide qui n'est pas au format .pem a été fourni pour le principal APNs, qui est le certificat SSL.	400	`SetPlatformAttributes`
APNs — Principal is an expired certificate	Un certificat arrivé à expiration a été fourni pour le principal APNs, qui est le certificat SSL.	400	`SetPlatformAttributes`
APNs — Principal is not an Apple issued certificate	Un certificat non émis par Apple a été fourni pour le principal APNs qui est le certificat SSL.	400	`SetPlatformAttributes`
APNs — Principal is not provided	Le principal APNs, qui est le certificat SSL, n'a pas été fourni.	400	`SetPlatformAttributes`
APNs — Credential is not provided	Les informations d'identification APNs, qui sont la clé privée, n'ont pas été fournies. Pour de plus amples informations, consultez CreatePlatformApplication dans la Référence de l'API d'Amazon Simple Notification Service.	400	`SetPlatformAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
APNs — Credential are not in a valid .pem format	Les informations d'identification APNs, qui sont la clé privée, ne sont pas dans un format .pem valide.	400	`SetPlatformAttributes`
FCM — serverAPIKey is not provided	Les informations d'identification FCM, qui sont la clé API, n'ont pas été fournies. Pour plus d'informations, consultez CreatePlatformApplication dans la Référence de l'API d'Amazon Simple Notification Service.	400	`SetPlatformAttributes`
FCM — serverAPIKey is a null string	Les informations d'identification FCM, qui sont la clé API, sont null.	400	`SetPlatformAttributes`
ADM — clientId is not provided	La chaîne requise pour l'ID client n'est pas fournie.	400	`SetPlatformAttributes`
ADM — clientid is a null string	La chaîne requise pour l'ID client est null.	400	`SetPlatformAttributes`
ADM — clientsecret is not provided	Le code secret du client requis n'est pas fourni.	400	`SetPlatformAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
ADM — clientsecret is a null string	La chaîne requise pour le code secret du client est null.	400	SetPlatformAttributes
EventEndpointUpdated has invalid ARN format	Le format d'ARN pour EventEndpointUpdated n'est pas valide.	400	SetPlatformAttributes
EventEndpointDeleted has invalid ARN format	Le format d'ARN pour EventEndpointDeleted n'est pas valide.	400	SetPlatformAttributes
EventEndpointUpdated has invalid ARN format	Le format d'ARN pour EventEndpointUpdated n'est pas valide.	400	SetPlatformAttributes
EventDeliveryAttemptFailure has invalid ARN format	Le format d'ARN pour EventDeliveryAttemptFailure n'est pas valide.	400	SetPlatformAttributes
EventDeliveryFailure has invalid ARN format	Le format d'ARN pour EventDeliveryFailure n'est pas valide.	400	SetPlatformAttributes
EventEndpointCreated is not an existing Topic	La rubrique EventEndpointCreated n'existe pas.	400	SetPlatformAttributes
EventEndpointDeleted is not an existing Topic	La rubrique EventEndpointDeleted n'existe pas.	400	SetPlatformAttributes
EventEndpointUpdated is not an existing Topic	La rubrique EventEndpointUpdated n'existe pas.	400	SetPlatformAttributes

Erreur	Description	Code d'état HTTPS	Action d'API
EventDeliveryAttemptFailure is not an existing Topic	La rubrique EventDeliveryAttemptFailure n'existe pas.	400	SetPlatformAttributes
EventDeliveryFailure is not an existing Topic	La rubrique EventDeliveryFailure n'existe pas.	400	SetPlatformAttributes
Platform ARN is invalid	L'ARN de plateforme n'est pas valide.	400	GetPlatformApplicationAttributes
Platform ARN is valid but does not belong to the user	L'ARN de plateforme est valide, mais n'appartient pas à l'utilisateur.	403	GetPlatformApplicationAttributes
Token specified is invalid	Le jeton spécifié n'est pas valide.	400	ListPlatformApplications
Platform ARN is invalid	L'ARN de plateforme n'est pas valide.	400	ListEndpointsByPlatformApplication
Platform ARN is valid but does not belong to the user	L'ARN de plateforme est valide, mais n'appartient pas à l'utilisateur.	404	ListEndpointsByPlatformApplication
Token specified is invalid	Le jeton spécifié n'est pas valide.	400	ListEndpointsByPlatformApplication

Erreur	Description	Code d'état HTTPS	Action d'API
Platform ARN is invalid	L'ARN de plateforme n'est pas valide.	400	`DeletePlatformApplication`
Platform ARN is valid but does not belong to the user	L'ARN de plateforme est valide, mais n'appartient pas à l'utilisateur.	403	`DeletePlatformApplication`
Platform ARN is invalid	L'ARN de plateforme n'est pas valide.	400	`CreatePlatformEndpoint`
Platform ARN is valid but does not belong to the user	L'ARN de plateforme est valide, mais n'appartient pas à l'utilisateur.	404	`CreatePlatformEndpoint`
Token is not specified	Le jeton n'est pas spécifié.	400	`CreatePlatformEndpoint`
Token is not of correct length	La longueur du jeton est incorrecte.	400	`CreatePlatformEndpoint`
Les données utilisateurs du client sont trop volumineuses.	Les données utilisateur du client ne doivent pas dépasser 2 048 octets en encodage UTF-8.	400	`CreatePlatformEndpoint`
Endpoint ARN is invalid	L'ARN du point de terminaison n'est pas valide.	400	`DeleteEndpoint`

Erreur	Description	Code d'état HTTPS	Action d'API
Endpoint ARN is valid but does not belong to the user	L'ARN du point de terminaison est valide, mais n'appartient pas à l'utilisateur.	403	`DeleteEndpoint`
Endpoint ARN is invalid	L'ARN du point de terminaison n'est pas valide.	400	`SetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	L'ARN du point de terminaison est valide, mais n'appartient pas à l'utilisateur.	403	`SetEndpointAttributes`
Token is not specified	Le jeton n'est pas spécifié.	400	`SetEndpointAttributes`
Token is not of correct length	La longueur du jeton est incorrecte.	400	`SetEndpointAttributes`
Les données utilisateurs du client sont trop volumineuses.	Les données utilisateur du client ne doivent pas dépasser 2 048 octets en encodage UTF-8.	400	`SetEndpointAttributes`
Endpoint ARN is invalid	L'ARN du point de terminaison n'est pas valide.	400	`GetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	L'ARN du point de terminaison est valide, mais n'appartient pas à l'utilisateur.	403	`GetEndpointAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
Target ARN is invalid	L'ARN cible n'est pas valide.	400	`Publish`
Target ARN is valid but does not belong to the user	L'ARN cible est valide, mais n'appartient pas à l'utilisateur.	403	`Publish`
Message format is invalid	Le format du message n'est pas valide.	400	`Publish`
Message size is larger than supported by protocol/end-service	La taille du message est supérieure à la taille prise en charge par le protocole/service final.	400	`Publish`

Utilisation de l'attribut de message de time-to-live (TTL) Amazon SNS pour les notifications push mobile

Amazon Simple Notification Service (Amazon SNS) prend en charge la définition d'un attribut de message de time-to-live (TTL) pour les messages de notification push mobile. Cela s'ajoute à la capacité existante de définir le TTL dans le corps du message Amazon SNS pour les services de notification push mobiles qui le prennent en charge, tels qu'Amazon Device Messaging (ADM) et Firebase Cloud Messaging (FCM) lors de l'envoi vers Android.

L'attribut de message TTL est utilisé pour spécifier des métadonnées d'expiration relatives à un message. Cela vous permet d'indiquer le temps mis par le service de notification push, tel qu'Apple Push Notification Service (APNs) ou FCM, pour diffuser le message au point de terminaison. Si, pour une raison quelconque (telle que la mise hors tension de l'appareil mobile), le message ne peut pas être diffusé dans la durée de vie spécifiée, il est supprimé et aucune autre tentative de diffusion n'est effectuée. Pour spécifier le TTL dans les attributs des messages, vous pouvez utiliser les AWS Management Console kits de développement AWS logiciel (SDK) ou l'API de requête.

Rubriques

- Attributs de message TTL pour les services de notification push

- Ordre de priorité pour déterminer la durée de vie
- Spécifier le TTL à l'aide du AWS Management Console

Attributs de message TTL pour les services de notification push

Voici une liste des attributs de message TTL pour les services de notification push que vous pouvez utiliser pour définir lors de l'utilisation des AWS SDK ou de l'API de requête :

Service de notification push	Attribut de message TTL
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple Push Notification service (APNs).	`AWS.SNS.MOBILE.APNS.TTL`
Apple Push Notification Service Sandbox (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCM lors de l'envoi vers Android)	`AWS.SNS.MOBILE.FCM.TTL`
Services de notification push Windows (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Chacun des services de notification push gère la durée de vie différemment. Amazon SNS fournit une vue abstraite de la durée de vie pour tous les services de notification push, ce qui facilite la spécification de la durée de vie. Lorsque vous utilisez le AWS Management Console pour spécifier le TTL (en secondes), vous ne devez saisir la valeur TTL qu'une seule fois et Amazon SNS calculera ensuite le TTL pour chacun des services de notification push sélectionnés lors de la publication du message.

La durée de vie est relative à l'heure de publication. Avant de remettre un message de notification push à un service de notification push spécifique, Amazon SNS calcule la durée de conservation (la durée entre l'horodatage de publication et juste avant sa remise à un service de notification push) pour la notification push et transmet la durée de vie restante au service de notification push spécifique. Si la durée de vie est inférieure à la durée de conservation, Amazon SNS ne tente pas d'effectuer la publication.

Si vous spécifiez un TTL pour un message de notification push, la valeur TTL doit être un entier positif, sauf si la valeur de `0` a une signification spécifique pour le service de notification push, comme dans le cas des APN et du FCM (lors de l'envoi vers Android). Si la valeur de durée de vie est définie sur`0` et que le service de notification push n'a pas de signification spécifique pour `0`, Amazon SNS supprime le message. Pour plus d'informations sur la définition du paramètre de durée de vie sur `0` lors de l'utilisation d'APNs, consultez le Tableau A-3 Identificateurs d'élément pour les notifications à distance dans la documentation Binary Provider API.

Ordre de priorité pour déterminer la durée de vie

La priorité utilisée par Amazon SNS pour déterminer la durée de vie pour un message de notification push se fonde sur l'ordre suivant, où le plus petit nombre a la priorité la plus élevée :

1. Durée de vie de l'attribut de message
2. Durée de vie du corps du message
3. Durée de vie par défaut du service de notification push (varie selon le service)
4. Durée de vie par défaut d'Amazon SNS (4 semaines)

Si vous définissez des valeurs de durée de vie différentes (une dans les attributs de message et une autre dans le corps du message) pour le même message, Amazon SNS modifie la durée de vie dans le corps du message pour qu'elle corresponde à celle spécifiée dans l'attribut du message.

Spécifier le TTL à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Mobile, Notifications Push.
3. Sur la page Notifications Push mobile, dans la section Applications de plateforme, sélectionnez une application.
4. Sur la ***MyApplication***page, dans la section Points de terminaison, choisissez un point de terminaison d'application, puis choisissez Publier le message.
5. Dans la section Détails du message, saisissez la valeur de la durée de vie (la durée, en secondes, pendant laquelle le service de notification push doit diffuser le message au point de terminaison).
6. Choisissez Publier le message.

Régions prises en charge pour les applications mobiles

Actuellement, vous pouvez créer des applications mobiles dans les régions suivantes :

- USA Est (Ohio)
- USA Est (Virginie du Nord)
- USA Ouest (Californie du Nord)
- USA Ouest (Oregon)
- Africa (Cape Town)
- Asie-Pacifique (Hong Kong)
- Asie-Pacifique (Jakarta)
- Asie-Pacifique (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Asie-Pacifique (Tokyo)
- Canada (Centre)
- Europe (Francfort)
- Europe (Irlande)
- Europe (Londres)
- Europe (Milan)
- Europe (Paris)
- Europe (Stockholm)
- Moyen-Orient (Bahreïn)
- Moyen-Orient (EAU)
- Amérique du Sud (São Paulo)
- AWS GovCloud (US-West)

Bonnes pratiques en matière de notifications push mobiles

Cette section décrit plusieurs bonnes pratiques qui peuvent vous aider à améliorer votre engagement client.

Gestion des points de terminaison

Des problèmes de livraison peuvent survenir au cas où des jetons d'appareil changent en raison de l'action d'un utilisateur sur l'appareil (par exemple, une application est réinstallée sur l'appareil) ou de mises à jour de certificats affectant les appareils exécutant une version iOS particulière. Une bonne pratique recommandée par Apple consiste à s'enregistrer avec des APN chaque fois que votre application est lancée.

Étant donné que le jeton d'appareil ne change pas chaque fois qu'une application est ouverte par un utilisateur, l'API `CreatePlatformEndpoint` idempotente peut être utilisée. Toutefois, cela peut créer des doublons pour le même appareil dans les cas où le jeton lui-même n'est pas valide ou si le point de terminaison est valide mais désactivé (par exemple, une incompatibilité entre les environnements de production et de sandbox).

Un mécanisme de gestion des jetons d'appareil tel que celui présent dans le pseudo-code peut être utilisé.

Pour plus d'informations sur la gestion et la maintenance des jetons de périphérique FCM v1, consultezGestion des terminaux Firebase Cloud Messaging (FCM).

Journalisation de l'état de distribution

Pour surveiller l'état de distribution des notifications push, nous vous recommandons d'activer la journalisation de l'état de distribution pour votre application de plateforme Amazon SNS. Cela vous aide à résoudre les échecs de livraison, car les journaux contiennent des codes de réponse fournisseur renvoyés par le service push de plateforme. Pour plus d'informations sur l'activation de la journalisation de l'état de distribution, consultez Comment puis-je accéder aux journaux de diffusion des rubriques Amazon SNS pour les notifications push ?.

Notifications d’événements

Pour gérer les points de terminaison d'une manière axée sur les événements, vous pouvez utiliser la fonctionnalité de notification d'événement. Elle permet à la rubrique Amazon SNS configurée de distribuer les événements aux abonnés, tels qu'une fonction Lambda, pour des événements

d'application de plateforme relatifs à la création de points de terminaison, à des suppressions, à des mises à jour ou à des échecs de livraison.

Notifications par e-mail

Cette page explique comment abonner une adresse e-mail à une rubrique Amazon SNS à l'aide du AWS Management Console AWS SDK for Java, ou. AWS SDK for .NET

> **Remarques**
>
> - Vous ne pouvez pas personnaliser le corps de l'e-mail. La fonction de remise des e-mails est destinée à fournir des alertes système internes, et non des messages marketing.
> - L'abonnement direct de points de terminaison d'e-mail n'est pris en charge que pour les rubriques standard.
> - Le débit de livraison des e-mails est limité en fonction des quotas Amazon SNS.

> **Important**
>
> Pour empêcher les destinataires des listes de diffusion de désabonner tous les destinataires des e-mails de la rubrique Amazon SNS, consultez Configurer une inscription par e-mail qui nécessite de s'authentifier pour se désinscrire dans le support AWS .

Pour abonner une adresse e-mail à une rubrique Amazon SNS à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation de gauche, choisissez Subscriptions (Abonnements).
3. Sur la page Abonnements, choisissez Créer un abonnement.
4. Sur la page Créer un abonnement, dans la section Détails, procédez comme suit :
   a. Pour ARN de rubrique, choisissez l'Amazon Resource Name (ARN) d'une rubrique.
   b. Pour Protocole, choisissez E-mail.
   c. Saisissez l'adresse e-mail pour le point de terminaison.

d. (Facultatif) Pour configurer une politique de filtrage, développez la section Politique de filtrage d'abonnement. Pour plus d'informations, consultez Stratégies de filtre d'abonnement Amazon SNS.

e. (Facultatif) Pour activer le filtrage basé sur la charge utile, configurez `Filter Policy Scope` sur `MessageBody`. Pour plus d'informations, consultez Étendue de la politique de filtre d'abonnement Amazon SNS.

f. (Facultatif) Pour configurer une file d'attente de lettres mortes pour l'abonnement, développez la section Politique de reconduite (File d'attente de lettres mortes). Pour de plus amples informations, consultez la section Files d'attente de lettres mortes (DLQ)) d'Amazon SNS.

g. Choisissez Créer un abonnement.

   La console crée l'abonnement et ouvre la page Détails de l'abonnement.

Vous devez confirmer l'abonnement avant que l'adresse e-mail puisse recevoir des messages.

Pour confirmer un abonnement

1. Vérifiez votre boîte de réception et choisissez Confirmer l'abonnement dans l'e-mail d'Amazon SNS.
2. Amazon SNS ouvre votre navigateur web et affiche une confirmation d'abonnement avec votre ID d'abonnement.

Pour abonner une adresse e-mail à une rubrique Amazon SNS à l'aide d'un SDK AWS

Pour utiliser un AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez Les fichiers de configuration et d'informations d'identification partagés dans le AWS Guide de référence des kits SDK et des outils.

Les exemples de code suivants montrent comment utiliser`Subscribe`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

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

Abonnez une file d'attente à une rubrique avec des filtres facultatifs.

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

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for .NET .

C++

Kit SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

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

Abonnez une application mobile à un sujet.

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

Abonne une fonction Lambda à une rubrique.

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

Abonne une file d'attente SQS à une rubrique.

```
        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;
           }
```

Abonnez-vous à un sujet avec un filtre.

```
       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;
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour s'abonner à une rubrique

La commande `subscribe` suivante abonne une adresse e-mail à la rubrique spécifiée.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Sortie :

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Pour plus d'informations sur l'API, consultez Subscribe dans la Référence des commandes AWS CLI .

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une file d'attente à une rubrique avec des filtres facultatifs.

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }
```

```
 return subscriptionArn, err
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for Go .

Java

Kit SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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);
        }
    }
}
```

Abonnez un point de terminaison HTTP à une rubrique.

```
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);
        }
    }
}
```

Abonne une fonction Lambda à une rubrique.

```
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 "";
```

```
    }
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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'
  // }
};
```

Abonnez une application mobile à un sujet.

```
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;
};
```

Abonne une fonction Lambda à une rubrique.

```
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;
};
```

Abonne une file d'attente SQS à une rubrique.

```
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;
};
```

Abonnez-vous à un sujet avec un filtre.

```
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;
};
```

- Pour plus d'informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK for JavaScript Référence de l'API.

Kotlin

Kits SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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()
    }
}
```

Abonne une fonction Lambda à une rubrique.

```
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}")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section S'abonner de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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());
}
```

Abonnez un point de terminaison HTTP à une rubrique.

```
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());
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK for PHP Référence de l'API.

Python

Kit SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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
```

- Pour de plus amples informations sur l'API, consultez Abonner dans Référence du kit SDK AWS pour l'API Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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
```

- Pour plus d'informations, consultez le AWS SDK for Ruby Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK for Ruby Référence de l'API.

Rust

Kit SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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(())
}
```

- Pour de plus amples informations sur l'API, consultez Abonner dans Référence du kit SDK AWS pour l'API Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
   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.
```

- Pour plus d'informations sur l'API, consultez Subscribe dans la Référence d'API du kit SDK AWS pour SAP ABAP.

Exemples de code pour Amazon SNS à l'aide de kits SDK AWS

Les exemples de code suivants montrent comment utiliser Amazon SNS avec un kit de développement AWS logiciel (SDK).

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Les Exemples de services croisés sont des exemples d'applications fonctionnant sur plusieurs Services AWS.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Mise en route

Hello Amazon SNS

Les exemples de code suivants montrent comment démarrer avec Amazon SNS.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Code pour le MakeLists fichier CMake C.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code pour le fichier source 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;
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for C++ API.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
```

```
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Initialisez un client SNS et répertoriez les rubriques figurant dans votre compte.

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

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}")
            }
    }
}
```

- Pour plus de détails sur l'API, consultez ListTopicsla section AWS SDK pour la référence de l'API Kotlin.

Exemples de code

- Actions pour Amazon SNS à l'aide de kits SDK AWS
  - Utilisation CheckIfPhoneNumberIsOptedOut avec un AWS SDK ou une CLI
  - Utilisation ConfirmSubscription avec un AWS SDK ou une CLI
  - Utilisation CreateTopic avec un AWS SDK ou une CLI
  - Utilisation DeleteTopic avec un AWS SDK ou une CLI
  - Utilisation GetSMSAttributes avec un AWS SDK ou une CLI
  - Utilisation GetTopicAttributes avec un AWS SDK ou une CLI
  - Utilisation ListPhoneNumbersOptedOut avec un AWS SDK ou une CLI
  - Utilisation ListSubscriptions avec un AWS SDK ou une CLI

- Utiliser API Gateway pour invoquer une fonction Lambda
- Utilisent des événements planifiés pour invoquer une fonction Lambda

Actions pour Amazon SNS à l'aide de kits SDK AWS

Les exemples de code suivants montrent comment effectuer des actions Amazon SNS individuelles avec AWS des SDK. Ces extraits appellent l'API Amazon SNS et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la Référence de l'API Amazon Simple Notification Service (Amazon SNS).

Exemples

- Utilisation CheckIfPhoneNumberIsOptedOut avec un AWS SDK ou une CLI
- Utilisation ConfirmSubscription avec un AWS SDK ou une CLI
- Utilisation CreateTopic avec un AWS SDK ou une CLI
- Utilisation DeleteTopic avec un AWS SDK ou une CLI
- Utilisation GetSMSAttributes avec un AWS SDK ou une CLI
- Utilisation GetTopicAttributes avec un AWS SDK ou une CLI
- Utilisation ListPhoneNumbersOptedOut avec un AWS SDK ou une CLI
- Utilisation ListSubscriptions avec un AWS SDK ou une CLI
- Utilisation ListTopics avec un AWS SDK ou une CLI
- Utilisation Publish avec un AWS SDK ou une CLI
- Utilisation SetSMSAttributes avec un AWS SDK ou une CLI
- Utilisation SetSubscriptionAttributes avec un AWS SDK ou une CLI
- Utilisation SetSubscriptionAttributesRedrivePolicy avec un AWS SDK ou une CLI
- Utilisation SetTopicAttributes avec un AWS SDK ou une CLI
- Utilisation Subscribe avec un AWS SDK ou une CLI
- Utilisation TagResource avec un AWS SDK ou une CLI
- Utilisation Unsubscribe avec un AWS SDK ou une CLI

Utilisation **CheckIfPhoneNumberIsOptedOut** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}");
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Pour vérifier la désactivation d'un numéro de téléphone aux SMS

L'`check-if-phone-number-is-opted-out`exemple suivant vérifie si le numéro de téléphone spécifié est désactivé pour ne pas recevoir de SMS du AWS compte courant.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Sortie :

```
{
    "isOptedOut": false
}
```

- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
```

```
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for JavaScript API.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK for PHP API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`ConfirmSubscription`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`ConfirmSubscription`.

CLI

AWS CLI

Pour confirmer un abonnement

La commande `confirm-subscription` suivante termine le processus de confirmation lancé lorsque vous vous êtes abonné à une rubrique SNS nommée `my-topic`. Le paramètre --token provient du message de confirmation envoyé au point de terminaison de notification spécifié dans l'appel d'abonnement.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --token
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

Sortie :

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Pour plus de détails sur l'API, reportez-vous ConfirmSubscriptionà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ConfirmSubscriptionà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous ConfirmSubscriptionà la section Référence des AWS SDK for JavaScript API.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous ConfirmSubscriptionà la section Référence des AWS SDK for PHP API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`CreateTopic`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`CreateTopic`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- Créer et publier dans une rubrique FIFO
- Publier des messages dans des files d'attente

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une rubrique avec un nom spécifique.

```
    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;
        }
```

```
    }
```

Créez une nouvelle rubrique avec un nom et des attributs FIFO et de déduplication spécifiques.

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

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour créer une rubrique SNS

L'exemple `create-topic` suivant crée une rubrique SNS nommée `my-topic`.

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

Sortie :

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

Pour plus d'informations, consultez la section Utilisation de l'interface de ligne de AWS commande avec Amazon SQS et Amazon SNS dans le Guide de l'utilisateur de AWS l'interface de ligne de commande.

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS CLI commandes.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
```

```
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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 "";
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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()
    }
}
```

- Pour plus de détails sur l'API, consultez CreateTopicla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for PHP API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez CreateTopicle AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK for Ruby API.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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(())
}
```

- Pour plus de détails sur l'API, voir CreateTopicla section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, consultez CreateTopicla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DeleteTopic`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`DeleteTopic`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Publier des messages dans des files d'attente

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Supprimez une rubrique à l'aide de son ARN de rubrique.

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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour supprimer une rubrique SNS

L'exemple `delete-topic` suivant supprime la rubrique SNS spécifiée.

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

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS CLI commandes.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



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

```
 }
 return err
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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
  //   }
```

```
  // }
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus de détails sur l'API, consultez DeleteTopicla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK for PHP API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez DeleteTopicle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, consultez DeleteTopicla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`GetSMSAttributes`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`GetSMSAttributes`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
```

```
}
```

- Pour de plus amples informations sur l'API, consultez GetSMSAttributes dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour répertorier les attributs des SMS par défaut

L'exemple `get-sms-attributes` suivant répertorie les attributs par défaut pour l'envoi de SMS.

```
aws sns get-sms-attributes
```

Sortie :

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Pour plus d'informations sur l'API, consultez GetSMSAttributes dans la Référence des commandes AWS CLI .

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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");
    }
}
```

- Pour de plus amples informations sur l'API, consultez GetSMSAttributes dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez GetSMSAttributes dans Référence de l'API AWS SDK for JavaScript .

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez GetSMSAttributes dans Référence de l'API AWS SDK for PHP .

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`GetTopicAttributes`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`GetTopicAttributes`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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");
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour récupérer les attributs d'une rubrique

L'exemple `get-topic-attributes` suivant affiche les attributs de la rubrique spécifiée.

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

Sortie :

```
{
    "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"
    }
}
```

- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK for JavaScript API.

SDK pour JavaScript (v2)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Importez le kit SDK et les modules client et appelez l'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);
  });
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}")
    }
}
```

- Pour plus de détails sur l'API, consultez GetTopicAttributesla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK for PHP API.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, consultez GetTopicAttributesla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`ListPhoneNumbersOptedOut`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Pour répertorier les désactivations des SMS

L'exemple `list-phone-numbers-opted-out` suivant répertorie les numéros de téléphone qui ont désactivé la réception de SMS.

```
aws sns list-phone-numbers-opted-out
```

Sortie :

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK for Java 2.x API.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK for PHP API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`ListSubscriptions`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`ListSubscriptions`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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;
}
```

- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour répertorier vos abonnements SNS

L'`list-subscriptions`exemple suivant affiche la liste des abonnements SNS de votre AWS compte.

```
aws sns list-subscriptions
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez ListSubscriptionsla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK for PHP API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez ListSubscriptionsle AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, consultez ListSubscriptionsla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`ListTopics`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`ListTopics`.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}");
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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;
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Pour répertorier vos rubriques SNS

L'`list-topics`exemple suivant répertorie toutes les rubriques SNS de votre AWS compte.

```
aws sns list-topics
```

Sortie :

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS CLI commandes.

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez ListTopicsla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for PHP API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez ListTopicsle AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end
```

```
# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK for Ruby API.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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(())
}
```

- Pour plus de détails sur l'API, voir ListTopicsla section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, consultez ListTopicsla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`Publish`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`Publish`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- Créer et publier dans une rubrique FIFO
- Publier un message texte SMS
- Publier des messages dans des files d'attente

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Publier un message dans une rubrique

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

Publiez un message dans une rubrique avec des options de groupe, de duplication et d'attribut.

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

Appliquez les sélections de l'utilisateur à l'action de publication.

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

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for .NET Référence de l'API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

Publiez un message avec un attribut.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
```

```
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

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

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Exemple 1 : pour publier un message dans une rubrique

L'exemple `publish` suivant publie le message spécifié sur la rubrique SNS spécifiée. Le message provient d'un fichier texte qui vous permet d'inclure des sauts de ligne.

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

```
    --message file://message.txt
```

Contenu de `message.txt` :

```
Hello World
Second Line
```

Sortie :

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

Exemple 2 : pour publier un SMS sur un numéro de téléphone

L'exemple `publish` suivant publie le message `Hello world!` sur le numéro de téléphone `+1-555-555-0100`.

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

Sortie :

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

- Pour plus d'informations sur l'API, consultez Publish dans la Référence des commandes AWS CLI .

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for Go Référence de l'API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

Publiez un message dans une rubrique avec des options de groupe, de duplication et d'attribut.

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

    let groupId, deduplicationId, choices;

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

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

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
```

```
      });
    }

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

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

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

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for JavaScript.

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for JavaScript .

Kotlin

Kit SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section Publier de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for PHP .

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple montre la publication d'un message avec un seul élément MessageAttribute déclaré en ligne.

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

Exemple 2 : Cet exemple montre la publication d'un message dont plusieurs MessageAttributes ont été déclarés à l'avance.

```
$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
```

- Pour plus de détails sur l'API, consultez la section Publication dans la référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Publiez un message avec des attributs afin qu'un abonnement puisse filtrer en fonction des attributs.

```
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
```

Publiez un message qui prend différentes formes en fonction du protocole de l'abonné.

```
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
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS Référence de l'API du kit SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for Ruby .

Rust

Kit SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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(())
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence du kit SDK AWS pour l'API Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus d'informations sur l'API, consultez Publish dans la Référence du kit SDK AWS pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`SetSMSAttributes`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`SetSMSAttributes`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Comment utiliser Amazon SNS pour définir l’attribut DefaultSMSType.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
```

```
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour définir les attributs des SMS

L'exemple `set-sms-attributes` suivant définit l'ID d'expéditeur par défaut des SMS sur `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Cette commande ne produit aucun résultat.

- Pour plus d'informations sur l'API, consultez SetSMSAttributes dans la Référence des commandes AWS CLI .

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
```

```
    }
}
```

- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for JavaScript .

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK for PHP .

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`SetSubscriptionAttributes`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`SetSubscriptionAttributes`.

CLI

AWS CLI

Pour définir des attributs d'abonnement

L'exemple `set-subscription-attributes` suivant définit l'attribut `RawMessageDelivery` sur un abonnement 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
```

Cette commande ne produit aucun résultat.

L'exemple `set-subscription-attributes` suivant définit un attribut `FilterPolicy` sur un abonnement 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\"] }"
```

Cette commande ne produit aucun résultat.

L'exemple `set-subscription-attributes` suivant supprime l'attribut `FilterPolicy` d'un abonnement 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 "{}"
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous SetSubscriptionAttributesà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous SetSubscriptionAttributesà la section Référence des AWS SDK for Java 2.x API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour plus de détails sur l'API, consultez SetSubscriptionAttributesle AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **SetSubscriptionAttributesRedrivePolicy** avec un AWS SDK ou une CLI

L'exemple de code suivant montre comment utiliser`SetSubscriptionAttributesRedrivePolicy`.

Java

Kit SDK pour Java 1.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`SetTopicAttributes`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`SetTopicAttributes`.

CLI

AWS CLI

Pour définir un attribut pour une rubrique

L'exemple `set-topic-attributes` suivant définit l'attribut `DisplayName` pour la rubrique spécifiée.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus de détails sur l'API, consultez SetTopicAttributesla section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for PHP API.

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for Ruby.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus de détails sur l'API, consultez SetTopicAttributesla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`Subscribe`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`Subscribe`.

Les exemples d’actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- Créer et publier dans une rubrique FIFO
- Publier des messages dans des files d'attente

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

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

Abonnez une file d'attente à une rubrique avec des filtres facultatifs.

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

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for .NET .

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

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

Abonnez une application mobile à un sujet.

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

Abonne une fonction Lambda à une rubrique.

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

Abonne une file d'attente SQS à une rubrique.

```
        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;
           }
```

Abonnez-vous à un sujet avec un filtre.

```
       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;
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour s'abonner à une rubrique

La commande `subscribe` suivante abonne une adresse e-mail à la rubrique spécifiée.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Sortie :

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Pour plus d'informations sur l'API, consultez Subscribe dans la Référence des commandes AWS CLI .

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une file d'attente à une rubrique avec des filtres facultatifs.

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }
```

```
 return subscriptionArn, err
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for Go .

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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);
        }
    }
}
```

Abonnez un point de terminaison HTTP à une rubrique.

```
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);
        }
    }
}
```

Abonne une fonction Lambda à une rubrique.

```
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 "";
```

```
    }
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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'
  // }
};
```

Abonnez une application mobile à un sujet.

```
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;
};
```

Abonne une fonction Lambda à une rubrique.

```
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;
};
```

Abonne une file d'attente SQS à une rubrique.

```
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;
};
```

Abonnez-vous à un sujet avec un filtre.

```
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;
};
```

- Pour plus d'informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK for JavaScript Référence de l'API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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()
    }
}
```

Abonne une fonction Lambda à une rubrique.

```
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}")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section S'abonner de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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());
}
```

Abonnez un point de terminaison HTTP à une rubrique.

```
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());
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK for PHP Référence de l'API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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
```

- Pour de plus amples informations sur l'API, consultez Abonner dans Référence du kit SDK AWS pour l'API Python (Boto3).

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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
```

- Pour plus d'informations, consultez le AWS SDK for Ruby Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK for Ruby Référence de l'API.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
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(())
}
```

- Pour de plus amples informations sur l'API, consultez Abonner dans Référence du kit SDK AWS pour l'API Rust.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Abonnez une adresse e-mail à un sujet.

```
   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.
```

- Pour plus d'informations sur l'API, consultez Subscribe dans la Référence du kit SDK AWS pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`TagResource`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`TagResource`.

CLI

AWS CLI

Pour ajouter une balise à une rubrique

L'exemple `tag-resource` suivant ajoute une balise de métadonnées à la rubrique Amazon SNS spécifiée.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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")
    }
```

```
}
```

- Pour plus de détails sur l'API, consultez TagResourcela section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`Unsubscribe`** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser`Unsubscribe`.

Les exemples d’actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l’exemple de code suivant :

- Publier des messages dans des files d'attente

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Désabonnez-vous d'une rubrique à l'aide d'un ARN d'abonnement.

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

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for .NET .

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for C++ .

CLI

AWS CLI

Pour se désabonner d'une rubrique

L'exemple `unsubscribe` suivant supprime l'abonnement spécifié d'une rubrique.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Cette commande ne produit aucun résultat.

- Pour plus d'informations sur l'API, consultez Unsubscribe dans la Référence des commandes AWS CLI .

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for Java 2.x .

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez le client dans un module séparé et exportez-le.

```
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({});
```

Importez le kit SDK et les modules client et appelez l'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;
};
```

- Pour de plus amples informations, consultez le AWS SDK for JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for JavaScript .

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section Se désabonner de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK for PHP .

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence du kit SDK AWS de l'API Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.
```

- Pour plus d'informations sur l'API, consultez Unsubscribe dans la Référence du kit SDK AWS de l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Scénarios pour Amazon SNS utilisant des SDK AWS

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans Amazon SNS avec AWS des SDK. Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions dans Amazon SNS. Chaque scénario inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code.

Exemples

- Création d'un point de terminaison de plateforme pour les notifications push Amazon SNS à l'aide d'un SDK AWS

- Création et publication sur une rubrique FIFO Amazon SNS à l'aide d'un SDK AWS
- Publier des SMS sur une rubrique Amazon SNS à l'aide d'un SDK AWS
- Publiez un message volumineux sur Amazon SNS avec Amazon S3 à l'aide d'un SDK AWS
- Publier un SMS Amazon SNS à l'aide d'un SDK AWS
- Publiez des messages Amazon SNS dans les files d'attente Amazon SQS à l'aide d'un SDK AWS

Création d'un point de terminaison de plateforme pour les notifications push Amazon SNS à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment créer un point de terminaison de plateforme pour les notifications push Amazon SNS.

CLI

AWS CLI

Pour créer un point de terminaison d'application de plateforme

L'exemple `create-platform-endpoint` suivant crée un point de terminaison pour l'application de plateforme spécifiée à l'aide du jeton spécifié.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Sortie :

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

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

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
```

```
                    platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Création et publication sur une rubrique FIFO Amazon SNS à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment créer une rubrique FIFO Amazon SNS et y publier des informations.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Cet exemple

- crée une rubrique FIFO Amazon SNS, deux files d'attente FIFO Amazon SQS et une file d'attente standard.
- abonne les files d’attente à la rubrique et publie un message dans la rubrique.

Le test vérifie que le message a bien été reçu pour chaque file d'attente. L'exemple complet montre également l'ajout de stratégies d'accès et supprime les ressources à la fin.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
```

```
                "   fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
```

```
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

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

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

            return topicArn;

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

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

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
```

```
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

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

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

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

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

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

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

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
  - CreateTopic
  - Publish
  - Subscribe

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une rubrique FIFO Amazon SNS, abonnez les files d'attentes standard et FIFO Amazon SQS à la rubrique et publiez un message dans la rubrique.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
```

```
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
```

```
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

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

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
```

```
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
```

```
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
```

```
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la Référence du kit SDK AWS pour l'API Python (Boto3).
  - CreateTopic
  - Publish
  - Subscribe

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une rubrique FIFO, abonnez une file d'attente FIFO Amazon SQS à la rubrique et publiez un message dans une rubrique 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.
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la Référence du kit SDK AWS de l'API SAP ABAP.
  - CreateTopic
  - Publish
  - Subscribe

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Publier des SMS sur une rubrique Amazon SNS à l'aide d'un SDK AWS

L'exemple de code suivant illustre comment :

- Créer une rubrique Amazon SNS.
- Abonner des numéros de téléphone à la rubrique.
- Publier des messages SMS dans la rubrique afin que tous les numéros de téléphone abonnés reçoivent le message en même temps.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créer une rubrique et renvoyez son 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 "";
    }
}
```

Abonner un point de terminaison à une rubrique

```
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);
        }
    }
}
```

Définir des attributs sur le message, tels que l'ID de l'expéditeur, le prix maximal et son type. Les attributs de message sont facultatifs.

```
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);
        }
    }
```

```
}
```

Publier un message dans une rubrique Le message est envoyé à chaque abonné.

```
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);
        }
    }
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Publiez un message volumineux sur Amazon SNS avec Amazon S3 à l'aide d'un SDK AWS

L’exemple de code suivant montre comment publier un message volumineux sur Amazon SNS à l’aide d’Amazon S3 pour stocker la charge utile du message.

Java

Kit SDK pour Java 1.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Pour publier un message volumineux, vous pouvez utiliser la bibliothèque client étendue Amazon SNS pour Java. Le message que vous envoyez fait référence à un objet Amazon S3 contenant le contenu réel du message.

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

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Publier un SMS Amazon SNS à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment publier des messages SMS à l'aide d'Amazon SNS.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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}");
            }
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for .NET Référence de l'API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for C++ Référence de l'API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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);
        }
    }
}
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence de l'API AWS SDK for Java 2.x .

Kotlin

Kit SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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.")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section Publier de la référence du kit SDK AWS pour l'API Kotlin.

PHP

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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());
}
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK for PHP.
- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK for PHP Référence de l'API.

Python

SDK pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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
```

- Pour de plus amples informations sur l'API, consultez Publier dans Référence du kit SDK AWS de l'API Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Publiez des messages Amazon SNS dans les files d'attente Amazon SQS à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment :

- Créer une rubrique (FIFO ou non FIFO).
- Abonner plusieurs files d'attente à la rubrique avec la possibilité d'appliquer un filtre.
- Publier des messages dans la rubrique.
- Interroger les files d'attente pour les messages reçus.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario interactif à une invite de commande.

```
/// <summary>
/// Console application to run a workflow scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
```

```
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
```

```
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues workflow is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the workflow.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this workflow, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
```

```
                        $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                        $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
```

```
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
```

```
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
```

```
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
```

```
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }
```

```
        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
```

```
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
```

```
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Créez une classe qui englobe les opérations 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;
```

```
    }
}
```

Créez une classe qui englobe les opérations 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;
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code 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();
}
```

- Pour plus d’informations sur l’API, consultez les rubriques suivantes dans la référence de l’API AWS SDK for C++ .

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publish
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Go

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario interactif à une invite de commande.

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
```

```
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic() (string, string, bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }
```

```
 topicArn, err := runner.snsActor.CreateTopic(topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ordinal string, isFifoTopic bool)
 (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)
```

```
err = runner.sqsActor.AttachSendMessagePolicy(queueUrl, queueArn, topicArn)
if err != nil {
 panic(err)
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n"
+
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following
tones: %v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(topicArn, queueArn,
filterPolicy)
```

```
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(topicArn string, isFifoTopic bool,
 contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }
```

```
  err := runner.snsActor.Publish(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }
```

```
  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
```

```
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic()
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
```

```
  resources.Cleanup()
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Définissez une structure qui englobe les actions Amazon SNS utilisées dans cet exemple.

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



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

```
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
```

```
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
```

```
 return err
}
```

Définissez une structure qui englobe les actions Amazon SQS utilisées dans cet exemple.

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
```

```
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(queueUrl string, queueArn string,
 topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(context.TODO(),
 &sqs.SetQueueAttributesInput{
```

```
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
```

```
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(queueUrl string, messages []types.Message)
 error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(context.TODO(),
 &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

JavaScript

SDK pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Il s'agit du point d'entrée de ce flux de travail.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
```

```
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Le code précédent fournit les dépendances nécessaires et démarre le flux de travail. La section suivante contient l'essentiel de l'exemple.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;
```

```
/**
 * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
 * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
 * @param {import('../../libs/prompter.js').Prompter} prompter
 * @param {import('../../libs/logger.js').Logger} logger
 */
constructor(snsClient, sqsClient, prompter, logger) {
  this.snsClient = snsClient;
  this.sqsClient = sqsClient;
  this.prompter = prompter;
  this.logger = logger;
}

async welcome() {
  await this.logger.log(MESSAGES.description);
}

async confirmFifo() {
  await this.logger.log(MESSAGES.snsFifoDescription);
  this.isFifo = await this.prompter.confirm({
    message: MESSAGES.snsFifoPrompt,
  });

  if (this.isFifo) {
    this.logger.logSeparator(MESSAGES.headerDedup);
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }
```

```
    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );
```

```
    const { Attributes } = await this.sqsClient.send(
      new GetQueueAttributesCommand({
        QueueUrl: response.QueueUrl,
        AttributeNames: ["QueueArn"],
      }),
    );

    this.queues.push({
      queueName,
      queueArn: Attributes.QueueArn,
      queueUrl: response.QueueUrl,
    });

    await this.logger.log(
      MESSAGES.queueCreatedNotice
        .replace("${QUEUE_NAME}", queueName)
        .replace("${QUEUE_URL}", response.QueueUrl)
        .replace("${QUEUE_ARN}", Attributes.QueueArn),
    );
  }
}

async attachQueueIamPolicies() {
  for (const [index, queue] of this.queues.entries()) {
    const policy = JSON.stringify(
      {
        Statement: [
          {
            Effect: "Allow",
            Principal: {
              Service: "sns.amazonaws.com",
            },
            Action: "sqs:SendMessage",
            Resource: queue.queueArn,
            Condition: {
              ArnEquals: {
                "aws:SourceArn": this.topicArn,
              },
            },
          },
        ],
      },
      null,
```

```
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
```

```
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
```

```
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

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

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

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

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

```
              },
            },
          }
        : {}),
    }),
  );

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

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

async receiveAndDeleteMessages() {
  for (const queue of this.queues) {
    const { Messages } = await this.sqsClient.send(
      new ReceiveMessageCommand({
        QueueUrl: queue.queueUrl,
      }),
    );

    if (Messages) {
      await this.logger.log(
        MESSAGES.messagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      );
      console.log(Messages);

      await this.sqsClient.send(
        new DeleteMessageBatchCommand({
          QueueUrl: queue.queueUrl,
          Entries: Messages.map((message) => ({
            Id: message.MessageId,
            ReceiptHandle: message.ReceiptHandle,
          })),
        }),
      );
    } else {
      await this.logger.log(
```

```
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

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

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
```

```
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Exemples de solutions sans serveur pour Amazon AWS SNS utilisant des kits de développement logiciel

Les exemples de code suivants montrent comment utiliser Amazon SNS avec AWS des SDK.

Exemples

- Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

Les exemples de code suivants montrent comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages à partir d’une rubrique SNS. La fonction extrait les messages du paramètre d’événement et consigne le contenu de chaque message.

.NET

AWS SDK for .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l’exemple complet et découvrez comment le configurer et l’exécuter dans le référentiel d’exemples sans serveur.

Consommation d’un événement SNS avec Lambda à l’aide de .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

Kit SDK for Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement SNS avec Lambda à l'aide de 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 pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Utilisation d'un événement SNS avec Lambda à l'aide de 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 pour JavaScript (v3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement SNS avec JavaScript Lambda en utilisant.

```
// 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;
  }
}
```

Consommation d'un événement SNS avec TypeScript Lambda en utilisant.

```
// 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

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Utilisation d'un événement SNS avec Lambda à l'aide de 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 pour Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Utilisation d'un événement SNS avec Lambda à l'aide de 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

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement SNS avec Lambda à l'aide de 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 pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Utilisation d'un événement S3 avec Lambda à l'aide de 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
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Exemples multiservices pour Amazon AWS SNS à l'aide de kits SDK

Les exemples d'applications suivants utilisent des AWS SDK pour associer Amazon SNS à d'autres applications. Services AWS Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter l'application.

Exemples

- Créer une application pour soumettre des données à une table DynamoDB
- Créer une application de publication et d'abonnement qui traduit les messages
- Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes
- Créer une application Amazon Textract Explorer
- Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un SDK AWS
- Publiez des messages Amazon SNS dans les files d'attente Amazon SQS à l'aide d'un SDK AWS
- Utiliser API Gateway pour invoquer une fonction Lambda
- Utilisent des événements planifiés pour invoquer une fonction Lambda

Créer une application pour soumettre des données à une table DynamoDB

Les exemples de code suivants montrent comment créer une application qui soumet des données à une table Amazon DynamoDB et vous avertit lorsqu'un utilisateur met à jour la table.

Java

SDK pour Java 2.x

Indique comment créer une application Web dynamique qui envoie des données à l'aide de l'API Java Amazon DynamoDB et envoie un message texte à l'aide de l'API Java Amazon Simple Notification Service.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- DynamoDB
- Amazon SNS

JavaScript

SDK pour JavaScript (v3)

Cet exemple montre comment créer une application qui permet aux utilisateurs de soumettre des données à une table Amazon DynamoDB et d'envoyer un message texte à l'administrateur à l'aide d'Amazon Simple Notification Service (Amazon SNS).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Cet exemple est également disponible dans le AWS SDK for JavaScript guide du développeur v3.

Les services utilisés dans cet exemple

- DynamoDB
- Amazon SNS

Kotlin

SDK pour Kotlin

Indique comment créer une application Android native qui envoie des données à l'aide de l'API Kotlin Amazon DynamoDB et envoie un message texte à l'aide de l'API Kotlin Amazon SNS.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- DynamoDB
- Amazon SNS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Créer une application de publication et d’abonnement qui traduit les messages

Les exemples de code suivants montrent comment créer une application dotée de fonctionnalités d’abonnement et de publication et qui traduit les messages.

.NET

AWS SDK for .NET

Indique comment utiliser l’API .NET Amazon Simple Notification Service pour créer une application Web dotée de fonctionnalités d’abonnement et de publication. De plus, cet exemple d’application traduit également des messages.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon Translate

Java

SDK pour Java 2.x

Indique comment utiliser l’API Java Amazon Simple Notification Service pour créer une application Web dotée de fonctionnalités d’abonnement et de publication. De plus, cet exemple d’application traduit également des messages.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour obtenir le code source complet et les instructions sur la façon de configurer et d'exécuter l'exemple utilisant l'API Java Async, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon Translate

Kotlin

SDK pour Kotlin

Indique comment utiliser l'API Kotlin Amazon SNS pour créer une application dotée de fonctionnalités d'abonnement et de publication. De plus, cet exemple d'application traduit également des messages.

Pour obtenir le code source complet et les instructions relatives à la création d'une application Web, consultez l'exemple complet sur GitHub.

Pour obtenir le code source complet et les instructions sur la façon de créer une application Android native, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon Translate

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes

Les exemples de code suivants montrent comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

.NET

AWS SDK for .NET

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK pour C++

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK pour Java 2.x

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK pour JavaScript (v3)

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK pour Kotlin

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK pour Rust

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Créer une application Amazon Textract Explorer

Les exemples de code suivants expliquent comment explorer la sortie Amazon Textract via une application interactive.

JavaScript

SDK pour JavaScript (v3)

Montre comment utiliser le AWS SDK for JavaScript pour créer une application React qui utilise Amazon Textract pour extraire des données d'une image de document et les afficher sur une page Web interactive. Cet exemple s’exécute dans un navigateur Web et nécessite une identité Amazon Cognito authentifiée pour les informations d’identification. Il utilise Amazon Simple Storage Service (Amazon S3) pour le stockage et, pour les notifications, il interroge une file d’attente Amazon Simple Queue Service (Amazon SQS) abonnée à une rubrique Amazon Simple Notification Service (Amazon SNS).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK pour Python (Boto3)

Montre comment utiliser Amazon Textract pour détecter des éléments de texte, de formulaire et de tableau dans une image de document. AWS SDK for Python (Boto3) L'image d'entrée

et la sortie d'Amazon Textract sont affichées dans une application Tkinter qui vous permet d'explorer les éléments détectés.

- Soumettez une image de document à Amazon Textract et explorez la sortie des éléments détectés.
- Soumettez des images directement à Amazon Textract ou via un compartiment Amazon Simple Storage Service (Amazon S3).
- Utilisez des API asynchrones pour démarrer une tâche qui publie une notification dans une rubrique Amazon Simple Notification Service (Amazon SNS) lorsque le travail est terminé.
- Interrogez un service Amazon Simple Queue Service (Amazon SQS) pour obtenir un message de fin de tâche et affichez les résultats.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment détecter des personnes et des objets dans une vidéo avec Amazon Rekognition.

Python

SDK pour Python (Boto3)

Utilisez Amazon Rekognition pour détecter des visages, des objets et des personnes dans des vidéos en démarrant des tâches de détection asynchrone. Cet exemple montre

également comment configurer Amazon Rekognition pour notifier une rubrique Amazon Simple Notification Service (Amazon SNS) lorsque les tâches sont terminées et abonner une file d'attente Amazon Simple Queue Service (Amazon SQS) à la rubrique. Lorsque la file d'attente reçoit un message concernant une tâche, elle est récupérée et les résultats sont affichés.

Il est préférable de visionner cet exemple sur GitHub. Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Publiez des messages Amazon SNS dans les files d'attente Amazon SQS à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment :

- Créer une rubrique (FIFO ou non FIFO).
- Abonner plusieurs files d'attente à la rubrique avec la possibilité d'appliquer un filtre.
- Publier des messages dans la rubrique.
- Interroger les files d'attente pour les messages reçus.

Java

SDK pour Java 2.x

Présente la messagerie avec des rubriques et des files d'attente avec Amazon Simple Notification Service (Amazon SNS) et Amazon Simple Queue Service (Amazon SQS).

Pour obtenir le code source complet et les instructions illustrant la messagerie avec des rubriques et des files d'attente dans Amazon SNS et Amazon SQS, consultez l'exemple complet sur. GitHub

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon SQS

Kotlin

SDK pour Kotlin

Présente la messagerie avec des rubriques et des files d'attente avec Amazon Simple Notification Service (Amazon SNS) et Amazon Simple Queue Service (Amazon SQS).

Pour obtenir le code source complet et les instructions illustrant la messagerie avec des rubriques et des files d'attente dans Amazon SNS et Amazon SQS, consultez l'exemple complet sur. GitHub

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon SQS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utiliser API Gateway pour invoquer une fonction Lambda

Les exemples de code suivants montrent comment créer une AWS Lambda fonction invoquée par Amazon API Gateway.

Java

SDK pour Java 2.x

Montre comment créer une AWS Lambda fonction à l'aide de l'API d'exécution Lambda Java. Cet exemple fait appel à différents AWS services pour réaliser un cas d'utilisation spécifique. Cet exemple montre comment créer une fonction Lambda invoquée par Amazon API Gateway qui analyse une table Amazon DynamoDB à la recherche d’anniversaires professionnels et utilise Amazon Simple Notification Service (Amazon SNS) pour envoyer un message texte à vos employés qui les félicitent à leur date d’anniversaire.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK pour JavaScript (v3)

Montre comment créer une AWS Lambda fonction à l'aide de l'API JavaScript d'exécution Lambda. Cet exemple fait appel à différents AWS services pour réaliser un cas d'utilisation spécifique. Cet exemple montre comment créer une fonction Lambda invoquée par Amazon API Gateway qui analyse une table Amazon DynamoDB à la recherche d’anniversaires professionnels et utilise Amazon Simple Notification Service (Amazon SNS) pour envoyer un message texte à vos employés qui les félicitent à leur date d’anniversaire.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Cet exemple est également disponible dans le AWS SDK for JavaScript guide du développeur v3.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisent des événements planifiés pour invoquer une fonction Lambda

Les exemples de code suivants montrent comment créer une AWS Lambda fonction invoquée par un événement EventBridge planifié par Amazon.

Java

SDK pour Java 2.x

Montre comment créer un événement EventBridge planifié Amazon qui invoque une AWS Lambda fonction. Configurez EventBridge pour utiliser une expression cron afin de planifier le moment où la fonction Lambda est invoquée. Dans cet exemple, vous créez une fonction Lambda à l'aide de l'API d'exécution Lambda. Cet exemple fait appel à différents AWS services pour réaliser un cas d'utilisation spécifique. Cet exemple montre comment créer une application qui envoie un message texte mobile à vos employés pour les féliciter à leur date d'anniversaire.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK pour JavaScript (v3)

Montre comment créer un événement EventBridge planifié Amazon qui invoque une AWS Lambda fonction. Configurez EventBridge pour utiliser une expression cron afin de planifier le moment où la fonction Lambda est invoquée. Dans cet exemple, vous créez une fonction Lambda à l'aide de l'API d'exécution JavaScript Lambda. Cet exemple fait appel à différents AWS services pour réaliser un cas d'utilisation spécifique. Cet exemple montre comment créer une application qui envoie un message texte mobile à vos employés pour les féliciter à leur date d'anniversaire.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Cet exemple est également disponible dans le AWS SDK for JavaScript guide du développeur v3.

Les services utilisés dans cet exemple

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation d'Amazon SNS avec un SDK AWS. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Sécurité Amazon SNS

Cette section fournit des informations sur la sécurité, l'authentification et le contrôle d'accès dans Amazon SNS, ainsi que sur le langage de la politique d'accès Amazon SNS.

Rubriques

Protection des données

Le modèle de responsabilité partagée AWS s'applique à la protection des données dans Amazon Simple Notification Service. Comme décrit dans ce modèle, AWS est responsable de la protection de l'infrastructure globale sur laquelle l'ensemble du AWS Cloud s'exécute. La gestion du contrôle de votre contenu hébergé sur cette infrastructure est de votre responsabilité. Ce contenu comprend les tâches de configuration et de gestion de la sécurité des services AWS que vous utilisez. Pour en savoir plus sur la confidentialité des données, veuillez consulter FAQ sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWSet RGPD sur le Blog de sécurité AWS.

À des fins de protection des données, nous vous recommandons de protéger les informations d'identification Compte AWS et de configurer les comptes utilisateur individuels avec AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multi-facteur (MFA) avec chaque compte.
- Utilisez SSL/TLS pour communiquer avec les ressources AWS. Nous recommandons TLS 1.2 ou une version ultérieure.
- Configurez une API et la journalisation des activités utilisateur avec AWS CloudTrail.

- Utilisez des solutions de chiffrement AWS, ainsi que tous les contrôles de sécurité par défaut au sein des services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données personnelles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés FIPS 140-2 lorsque vous accédez à AWS via une CLI ou une API, utilisez un point de terminaison FIPS. Pour en savoir plus sur les points de terminaison FIPS disponibles, consultez Norme de traitement de l'information fédérale (Federal Information Processing Standard (FIPS)) 140-2.
- Message Data Protection
  - Message Data Protection est une nouvelle fonctionnalité majeure d'Amazon SNS
  - Utilisez MDP pour scanner les messages à la recherche d'informations confidentielles ou sensibles
  - Proposez un audit des messages pour tout le contenu circulant dans la rubrique
  - Fournissez des contrôles d'accès au contenu pour les messages publiés sur le sujet et les messages diffusés par la rubrique

> **Important**
>
> Nous vous recommandons vivement de ne jamais placer d'informations confidentielles ou sensibles, telles que des adresses électroniques, dans des balises ou des champs de format libre tels qu'un champ Nom. Cela s'applique aussi lorsque vous utilisez Amazon SNS ou d'autres services Amazon Web Services à l'aide de la console, de l'API, de la AWS CLI ou des SDK AWS. Toutes les données que vous saisissez dans des identifications ou des champs de format libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'URL pour valider votre demande au serveur.

Les sections suivantes fournissent des informations supplémentaires sur la protection des données dans Amazon SNS.

Rubriques

- Sécurité Message Data Protection

Chiffrement des données

La protection des données fait référence au fait de protéger les données pendant leur transit (lorsqu'elles sont transmises en direction ou en provenance d'Amazon SNS) et au repos (lorsqu'elles sont stockées sur des disques dans les centres de données Amazon SNS). Vous pouvez protéger les données en transit à l'aide de la technologie Secure Sockets Layer (SSL) ou du chiffrement côté client. Par défaut, Amazon SNS stocke les messages et les fichiers à l'aide du chiffrement du disque. Vous pouvez protéger les données inactives en demandant à Amazon SNS de chiffrer vos messages avant de les enregistrer dans le système de fichiers chiffré de ses centres de données. Amazon SNS recommande d'utiliser SSE pour optimiser le chiffrement des données.

Rubriques

- Chiffrement au repos
- Gestion des clés
- Activation du chiffrement côté serveur (SSE) pour une rubrique Amazon SNS
- Activation du chiffrement côté serveur (SSE) pour une rubrique Amazon SNS avec une file d'attente Amazon SQS chiffrée abonnée

Chiffrement au repos

Le chiffrement côté serveur (SSE) vous permet de stocker des données sensibles dans des rubriques chiffrées en protégeant le contenu des messages des rubriques Amazon SNS à l'aide de clés gérées AWS Key Management Service dans (). AWS KMS

SSE chiffre les messages une fois reçus par Amazon SNS. Les messages sont stockés sous forme cryptée et ne sont déchiffrés que lorsqu'ils sont envoyés.

- Pour plus d'informations sur la gestion du chiffrement SSE à l'aide d'AWS Management Console ou du AWS SDK for Java (en définissant l'attribut `KmsMasterKeyId` avec les actions d'API `CreateTopic` et `SetTopicAttributes`), consultez la section Activation du chiffrement côté serveur (SSE) pour une rubrique Amazon SNS.
- Pour plus d'informations sur la création de rubriques chiffrées à l'aide de AWS CloudFormation (en définissant la propriété `KmsMasterKeyId` à l'aide de la ressource `AWS::SNS::Topic`), consultez le Guide de l'utilisateur AWS CloudFormation.

> **Important**
>
> Toutes les requêtes adressées aux rubriques avec le chiffrement SSE activé doivent utiliser HTTPS et Signature version 4.
> Pour plus d'informations sur la compatibilité d'autres services avec les rubriques chiffrées, consultez la documentation de votre service.
> Amazon SNS prend uniquement en charge que les clés KMS de chiffrement symétriques. Vous ne pouvez utiliser aucun autre type de clé KMS pour crypter les ressources de votre service. Pour savoir si une clé KMS est symétrique, consultez Identification de clés asymétriques KMS.

AWS KMS combine du matériel et des logiciels sécurisés et hautement disponibles pour fournir un système de gestion de clés à l'échelle du cloud. Lorsque vous utilisez Amazon SNS avec AWS KMS, les clés de données qui chiffrent les données de vos messages sont également chiffrées et stockées avec les données qu'elles protègent.

L'utilisation d'AWS KMS offre les avantages suivants :

- Vous pouvez créer et gérer la AWS KMS key vous-même.
- Vous pouvez également utiliser des clés KMS gérées par AWS pour Amazon SNS, qui sont uniques pour chaque compte et région.
- Les normes de sécurité AWS KMS peuvent vous aider à respecter les exigences de conformité liées au chiffrement.

Pour plus d'informations, veuillez consulter Présentation de AWS Key Management Service dans le Manuel du développeur AWS Key Management Service.

Rubriques

- Portée du chiffrement
- Termes clés

Portée du chiffrement

Le chiffrement SSE chiffre le corps d'un message dans une rubrique Amazon SNS.

Le chiffrement SSE ne chiffre pas les éléments suivants :

- Métadonnées de rubrique (nom et attributs de la rubrique)
- Métadonnées de message (objet, ID, horodatage et attributs du message)
- Politique de protection des données
- Métriques par rubrique

> **Note**
>
> - Un message est chiffré uniquement s'il est envoyé après l'activation du chiffrement d'une rubrique. Amazon SNS ne chiffre pas les messages en attente.
> - Tout message chiffré reste chiffré même si le chiffrement de sa rubrique est désactivé.

Termes clés

Les termes clés suivants peuvent vous aider à mieux comprendre les fonctionnalités de chiffrement SSE. Pour des descriptions détaillées, consultez la Référence de l'API Amazon Simple Notification Service.

Clé de données

Clé de chiffrement des données (DEK) permettant de chiffrer le contenu des messages Amazon SNS.

Pour plus d'informations, consultez Clés de données dans le Guide du développeur AWS Key Management Service et Chiffrement d'enveloppe dans le Guide du développeur AWS Encryption SDK.

ID AWS KMS key

Alias, ARN d'alias, ID de clé ou ARN de clé AWS KMS key ou de AWS KMS personnalisée – dans votre compte ou un autre compte. Alors que l'alias de la AWS KMS gérée par AWS pour Amazon SNS est toujours `alias/aws/sns`, l'alias d'une AWS KMS personnalisée peut, par exemple, être `alias/MyAlias`. Vous pouvez utiliser ces clés AWS KMS pour protéger les messages des rubriques Amazon SNS.

> **Note**
>
> Gardez à l'esprit les points suivants :

- La première fois que vous utilisez la AWS Management Console pour spécifier la KMS gérée par AWS pour Amazon SNS pour une rubrique, AWS KMS crée la KMS gérée par AWS pour Amazon SNS.
- Alternativement, la première fois que vous utilisez l'action `Publish` sur une rubrique avec SSE activé, AWS KMS crée la KMS gérée par AWS pour Amazon SNS.

Vous pouvez créer des clés AWS KMS, définir les politiques qui contrôlent la façon dont les clés AWS KMS peuvent être utilisées et vérifier l'utilisation des AWS KMS à l'aide de la section AWS KMS keys de la console AWS KMS ou de l'action `CreateKey` AWS KMS. Pour en savoir plus, consultez AWS KMS keys et Création des clés dans le Manuel du développeur AWS Key Management Service. Pour plus d'exemples d'AWS KMSidentifiants, consultez la référence KeyIdde l'AWS Key Management ServiceAPI. Pour plus d'informations sur la recherche d'identifiants de AWS KMS, consultez la section Recherche de l'ID et de l'ARN d'une clé dans le Guide du développeur AWS Key Management Service.

Important

Des frais supplémentaires sont facturés pour l'utilisation de AWS KMS. Pour plus d'informations, veuillez consulter les sections Estimation des coûts AWS KMS et Tarification d'AWS Key Management Service.

Gestion des clés

Les sections suivantes fournissent des informations sur l'utilisation des clés gérées dans AWS Key Management Service (AWS KMS). Pour en savoir plus sur

Note

Amazon SNS prend uniquement en charge que les clés KMS de chiffrement symétriques. Vous ne pouvez utiliser aucun autre type de clé KMS pour crypter les ressources de votre service. Pour savoir si une clé KMS est symétrique, consultez Identification de clés asymétriques KMS.

Rubriques

- Estimation des coûts AWS KMS
- Configuration des autorisations AWS KMS
- Erreurs AWS KMS

Estimation des coûts AWS KMS

Pour prévoir les coûts et mieux comprendre votre facture AWS, vous souhaiterez peut-être connaître la fréquence à laquelle Amazon SNS utilise votre AWS KMS key.

> **Note**
>
> Bien que la formule ci-après puisse vous donner une très bonne idée des coûts à prévoir, les coûts réels risquent d'être plus élevés en raison de la nature distribuée d'Amazon SNS.

Pour calculer le nombre de demandes d'API (`R`) par rubrique, utilisez la formule suivante :

```
R = B / D * (2 * P)
```

`B` est la période de facturation (en secondes).

`D` est la période de réutilisation des clés de données (en secondes – Amazon SNS réutilise une clé de données pour une durée maximale de 5 minutes).

`P` est le nombre de principaux de publication qui envoient des messages à la rubrique Amazon SNS.

Voici des exemples de calcul. Pour obtenir des informations précises sur la tarification, consultez Tarification AWS Key Management Service.

Exemple 1 : calcul du nombre d'appels d'API AWS KMS pour 1 éditeur et 1 rubrique

Cet exemple suppose que :

- La période de facturation va du 1er au 31 janvier (2 678 400 secondes).
- La période de réutilisation des clés de données est de 5 minutes (300 secondes).
- Il y a 1 rubrique.
- Il y a 1 principal de publication.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Exemple 2 : calcul du nombre d'appels d'API AWS KMS pour plusieurs éditeurs et 2 rubriques

Cet exemple suppose que :

- La période de facturation va du 1er au 28 février (2 419 200 secondes).
- La période de réutilisation des clés de données est de 5 minutes (300 secondes).
- Il y a 2 rubriques.
- La première rubrique comporte 3 principals de publication.
- La deuxième en compte 5.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Configuration des autorisations AWS KMS

Avant de pouvoir utiliser le chiffrement SSE, vous devez configurer les politiques d'AWS KMS key pour autoriser le chiffrement de rubriques, ainsi que le chiffrement et le déchiffrement de messages. Pour obtenir des exemples et des informations supplémentaires sur les autorisations AWS KMS, consultez Autorisations de l'API AWS KMS : référence des actions et ressources dans le Guide du développeurAWS Key Management Service. Pour plus d'informations sur la configuration d'une rubrique Amazon SNS avec le chiffrement côté serveur, consultez Configurer une rubrique Amazon SNS avec le chiffrement côté serveur.

> **Note**
>
> Vous pouvez également gérer les autorisations pour les clés KMS à chiffrement symétrique à l'aide de politiques IAM. Pour plus d'informations, consultez Utilisation de politiques IAM avec AWS KMS.
>
> Alors que vous pouvez configurer des autorisations globales pour envoyer et recevoir des messages à destination et en provenance d'Amazon SNS, AWS KMS exige que l'ARN complet des clés KMS de régions spécifiques soit explicitement nommé dans la section `Resource` d'une politique IAM.

Vous devez aussi vous assurer que les stratégies de clés de la AWS KMS key accordent les autorisations nécessaires. Pour cela, nommez les mandataires qui produisent et consomment des messages chiffrés dans Amazon SNS en tant qu'utilisateurs dans la stratégie de clé KMS.

Vous pouvez également spécifier les actions AWS KMS requises et l'ARN KMS dans une politique IAM affectée aux principaux qui publient des messages chiffrés dans Amazon SNS et s'y abonnent. Pour en savoir plus, consultez la section Gestion de l'accès à AWS KMS dans le Guide du développeur AWS Key Management Service.

Si vous sélectionnez une clé gérée par le client pour votre rubrique Amazon SNS et que vous utilisez des alias pour contrôler l'accès aux clés KMS à l'aide de politiques IAM ou de stratégies de clés KMS avec la clé de condition `kms:ResourceAliases`, veillez à ce qu'un alias soit également associé à la clé gérée par le client sélectionnée. Pour plus d'informations sur l'utilisation d'alias pour contrôler l'accès aux clés KMS, consultez Utilisation d'alias pour contrôler l'accès aux clés KMS dans le Guide du développeur AWS Key Management Service.

Autoriser un utilisateur à envoyer des messages à une rubrique avec chiffrement SSE

Le diffuseur de publication doit disposer des autorisations `kms:GenerateDataKey*` et `kms:Decrypt` pour la 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"
  }]
}
```

Activer la compatibilité entre des sources d'événements à partir de services AWS et de rubriques chiffrées

Plusieurs services AWS publient des évènements dans des rubriques Amazon SNS. Pour permettre à ces sources d'événements d'utiliser des rubriques chiffrés, vous devez effectuer les opérations suivantes.

1. Utilisez une clé gérée par le client. Pour en savoir plus, consultez Création des clés dans le Guide du développeur AWS Key Management Service.
2. Pour permettre au service AWS d'avoir les autorisations `kms:GenerateDataKey*` et `kms:Decrypt`, ajoutez l'instruction suivante à la politique de la clé KMS.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Source de l'événement	principal du service
Amazon CloudWatch	`cloudwatch.amazonaws.com`
Amazon CloudWatch Events	`events.amazonaws.com`
AWS CodeCommit	`codecommit.amazonaws.com`
AWS CodeStar	`codestar-notifications.amazonaws.com`
AWS Database Migration Service	`dms.amazonaws.com`
AWS Directory Service	`ds.amazonaws.com`

Source de l'événement	principal du service
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	`importexport.amazonaws.com`
AWSSystems Manager Incident Manager	AWS Systems Manager Incident Manager comprend deux principes de service : `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

Note

Certaines sources d'évènements Amazon SNS nécessitent que vous fournissiez un rôle IAM (plutôt que le principal du service) dans la politique AWS KMS key :

- Amazon EC2 Auto Scaling
- Amazon Elastic Transcoder
- AWS CodePipeline
- AWS Config
- AWS Elastic Beanstalk
- AWS IoT
- EC2 Image Builder

3. Ajoutez les clés de condition `aws:SourceAccount` et `aws:SourceArn` à la politique basée sur les ressources KMS pour protéger davantage la clé KMS contre les attaques de l'adjoint confus. Reportez-vous à la liste des documents spécifiques au service (ci-dessus) pour connaître les détails précis de chaque cas.

> **Important**
>
> L'ajout du `aws:SourceAccount` et de `aws:SourceArn` à une politique AWS KMS n'est pas pris en charge pour les rubriques EventBridge-to-encrypted.

```
{
  "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. Activez le chiffrement SSE pour votre rubrique à l'aide de votre clé KMS.
5. Fournissez l'ARN de la rubrique chiffré à la source de l'événement.

Erreurs AWS KMS

Lorsque vous utilisez Amazon SNS et AWS KMS, vous risquez de rencontrer des erreurs. La liste suivante décrit les erreurs et les solutions de dépannage possibles.

KMSAccessDeniedException

Le texte chiffré désigne une clé qui n'existe pas ou à laquelle vous n'avez pas accès.

Code d'état HTTP : 400

KMSDisabledException

La demande a été rejetée, car la clé KMS spécifiée n'est pas activée.

Code d'état HTTP : 400

KMSInvalidStateException

La demande a été rejetée, car l'état de la ressource spécifiée n'est pas valide pour cette demande. Pour plus d'informations, consultez États de clé de AWS KMS keys dans le Manuel du développeur AWS Key Management Service.

Code d'état HTTP : 400

KMSNotFoundException

La requête a été rejetée, car l'entité ou la ressource spécifiée est introuvable.

Code d'état HTTP : 400

KMSOptInRequired

L'ID de clé d'accès AWS a besoin d'un abonnement pour le service.

Code d'état HTTP : 403

KMSThrottlingException

La demande a été refusée suite à une limitation des demandes. Pour plus d'informations sur cette limitation, consultez Quotas dans le Manuel du développeur AWS Key Management Service.

Code d'état HTTP : 400

Activation du chiffrement côté serveur (SSE) pour une rubrique Amazon SNS

Le chiffrement côté serveur (SSE) vous permet de stocker des données sensibles dans des rubriques chiffrées. Le chiffrement SSE protège le contenu des messages présents dans les rubriques

Amazon SNS à l'aide de clés gérées dans AWS Key Management Service (AWS KMS). Pour en savoir plus sur le chiffrement côté serveur avec Amazon SNS, consultez Chiffrement au repos. Pour en savoir plus sur la création de clés AWS KMS, consultez Création de clés dans le Guide du développeur AWS Key Management Service.

> **Important**
>
> Toutes les requêtes adressées aux rubriques avec le chiffrement SSE activé doivent utiliser HTTPS et Signature version 4.

Activer le chiffrement côté serveur (SSE) pour une rubrique Amazon SNS à l'aide de la AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sur la page Topics (Rubriques), sélectionnez une rubrique, puis choisissez Actions, Edit (Modifier).
4. Développez la section Encryption (Chiffrement) et effectuez les opérations suivantes :

   a. Choisissez Activer le chiffrement.

   b. Spécifiez la clé AWS KMS. Pour de plus amples informations, veuillez consulter Termes clés.

      Pour chaque type de KMS, les champs Description, Account (Compte) et KMS ARN (ARN KMS) s'affichent.

      > **Important**
      >
      > Si vous n'êtes pas le propriétaire de la clé KMS, ou si vous vous connectez avec un compte n'ayant pas les autorisations `kms:ListAliases` et `kms:DescribeKey`, vous ne pouvez pas afficher les informations relatives à la KMS sur la console Amazon SNS.
      > Demandez au propriétaire de la KMS de vous accorder ces autorisations. Pour plus d'informations, consultez la rubrique Autorisations d'API AWS KMS : référence des actions et ressources dans le Guide du développeur AWS Key Management Service.

- La clé KMD gérée par AWS pour Amazon SNS (Défaut) alias/aws/sns est sélectionnée par défaut.

  > **Note**
  >
  > Gardez à l'esprit les points suivants :
  >
  > - La première fois que vous utilisez la AWS Management Console pour spécifier la KMS gérée par AWS pour Amazon SNS pour une rubrique, AWS KMS crée la KMS gérée par AWS pour Amazon SNS.
  > - Alternativement, la première fois que vous utilisez l'action `Publish` sur une rubrique avec SSE activé, AWS KMS crée la KMS gérée par AWS pour Amazon SNS.

- Pour utiliser une KMS personnalisée à partir de votre compte AWS, choisissez le champ Clé KMS, puis choisissez la KMS personnalisée dans la liste.

  > **Note**
  >
  > Pour obtenir des instructions pour la création de KMS personnalisées, consultez la section Création de clés dans le Guide du développeur AWS Key Management Service.

- Pour utiliser un ARN KMS personnalisé à partir de votre compte AWS ou à partir d'un autre compte AWS, saisissez-le dans le champ Clé KMS.

5. Sélectionnez Save Changes (Enregistrer les modifications).

   SSE est activé pour votre rubrique et la page ***MaRubrique*** s'affiche.

   Le statut Encryption (Chiffrement) de la rubrique, Account (Compte) AWS, Customer master key (CMK) (Clé principale client [CMK]), CMK ARN (ARN CMK) et Description s’affichent dans l’onglet Encryption (Chiffrement).

Configurer une rubrique Amazon SNS avec le chiffrement côté serveur

Lorsque vous créez votre clé KMS, utilisez la stratégie de clé KMS suivante :

```
{
```

```
    "Effect": "Allow",
    "Principal": {
        "Service": "service.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

Activation du chiffrement côté serveur (SSE) pour une rubrique Amazon SNS avec une file d'attente Amazon SQS chiffrée abonnée

Vous pouvez activer le chiffrement côté serveur (SSE) pour une rubrique afin de protéger ses données. Pour permettre à Amazon SNS d'envoyer des messages à des files d'attente Amazon SQS chiffrées, la clé gérée par le client associée à la file d'attente Amazon SQS doit avoir un énoncé de politique qui accorde l'accès au principal du service Amazon SNS aux actions d'API AWS KMS `GenerateDataKey` et `Decrypt`. Pour plus d'informations sur l'utilisation du chiffrement SSE, consultez la section Chiffrement au repos.

Cette page vous montre comment activer le chiffrement SSE à l'aide de la AWS Management Console pour une rubrique Amazon SNS à laquelle une file d'attente Amazon SQS chiffrée est abonnée.

Étape 1 : Créer une clé KMS personnalisée

1. Connectez-vous à la console AWS KMS avec un utilisateur disposant au minimum de la politique `AWSKeyManagementServicePowerUser`.
2. Sélectionnez Create a key (Créer une clé).

3. Pour créer une clé KMS de chiffrement symétrique, pour Key type (Type de clé), choisissez Symmetric (Symétrique).

   Pour de plus amples informations sur la création d'une clé KMS asymétrique dans la console AWS KMS, consultez Création de clés KMS asymétriques (console).

4. Dans Key usage (Utilisation de la clé), l'option Encrypt and decrypt (Chiffrer et déchiffrer) est sélectionnée pour vous.

   Pour plus d'informations sur la création de clés KMS qui génèrent et vérifient des codes MAC, consultez Création de clés KMS HMAC.

   Pour de plus amples informations sur les Options avancées, consultez Clés à usage spécial.

5. Choisissez Next (Suivant).

6. Saisissez un alias pour la clé KMS. Le nom d'alias ne peut pas commencer par **`aws/`**. Le préfixe **`aws/`** est réservé par Amazon Web Services pour représenter Clés gérées par AWS dans votre compte.

   > **Note**
   >
   > L'ajout, la suppression ou la mise à jour d'un alias peut permettre d'accorder ou de refuser l'autorisation d'utiliser la clé KMS. Pour plus de détails, consultez ABAC pour AWS KMS et Utilisation d'alias pour contrôler l'accès aux clés KMS.

   Un alias est un nom d'affichage que vous pouvez utiliser pour identifier facilement la clé KMS. Nous vous conseillons de choisir un alias qui indique le type de données que vous envisagez de protéger ou l'application que vous pensez utiliser avec la clé KMS.

   Les alias sont requis lorsque vous créez une clé KMS dans la AWS Management Console. Ils sont facultatifs lorsque vous utilisez l'opération CreateKey.

7. (Facultatif) Saisissez une description pour la clé KMS.

   Vous pouvez ajouter une description maintenant ou la mettre à jour à tout moment, sauf si l'état de la clé est `Pending Deletion` ou `Pending Replica Deletion`. Pour ajouter, modifier ou supprimer la description d'une clé gérée par un client existante, modifiez la description dans la AWS Management Console ou utilisez l'opération UpdateKeyDescription.

8. (Facultatif) Saisissez une clé de balise et une valeur de balise facultative. Pour ajouter plus d'une balise à la clé KMS, sélectionnez Add tag (Ajouter une balise).

   > **Note**
   >
   > L'ajout ou la suppression d'une balise sur une KMS permet d'accorder ou de refuser l'autorisation d'utilisation de cette clé KMS. Pour plus de détails, consultez ABAC pour AWS KMS et Utilisation d'étiquettes pour contrôler l'accès aux clés KMS.

   Lorsque vous ajoutez des balises à vos ressources AWS, AWS génère un rapport de répartition des coûts faisant apparaître la consommation et les coûts regroupés par balises. Les balises peuvent également être utilisées pour contrôler l'accès à une clé KMS. Pour de plus amples informations sur l'étiquetage des clés KMS, veuillez consulter Étiquetage de clés et ABAC pour AWS KMS.
9. Choisissez Next (Suivant).
10. Sélectionnez les utilisateurs et les rôles IAM qui peuvent administrer la clé KMS.

    > **Note**
    >
    > Cette politique de clé donne au Compte AWS le contrôle total de cette clé KMS. Il permet aux administrateurs de compte d'utiliser des politiques IAM pour autoriser d'autres principals à gérer la clé KMS. Pour plus d'informations, consultez Stratégie de clé par défaut.
    >
    > Les bonnes pratiques IAM déconseillent d'avoir recours à des utilisateurs IAM dotés d'informations d'identification à long terme. Dans la mesure du possible, utilisez des rôles IAM, qui fournissent des informations d'identification temporaires. Pour plus d'informations, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.
11. (Facultatif) Pour empêcher les utilisateurs et les rôles IAM sélectionnés de supprimer cette clé KMS, dans la section Key deletion (Suppression de la clé) en bas de la page, décochez la case Allow key administrators to delete this key (Autoriser les administrateurs de clé à supprimer cette clé).
12. Choisissez Next (Suivant).

13. Sélectionnez les utilisateurs et les rôles IAM qui peuvent utiliser la clé dans les opérations de chiffrement. Choisissez Next (Suivant).
14. Sur la page Review and edit key policy (Vérifier et modifier la politique de clé), ajoutez l'instruction suivante dans la politique de clé, puis choisissez Finish (Terminer).

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

Votre nouvelle clé gérée par le client s'affiche dans la liste des clés.

Étape 2 : Création d'une une rubrique Amazon SNS chiffrée

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Choisissez Create topic (Créer une rubrique).
4. Sur la page Create new topic (Créer une rubrique), dans Name (Nom), tapez le nom de la rubrique (par exemple, `MyEncryptedTopic`), puis choisissez Create topic (Créer une rubrique).
5. Développez la section Encryption (Chiffrement) et effectuez les opérations suivantes :

    a. Choisissez Enable server-side encryption (Activer le chiffrement côté serveur).

    b. Spécifiez la clé gérée par le client. Pour de plus amples informations, veuillez consulter Termes clés.

    Pour chaque type de clé gérée par le client, la Description, le Compte et l'ARN de la clé gérée par le client sont affichés.

> **Important**
>
> Si vous n'êtes pas le propriétaire de la clé gérée par le client, ou si vous vous connectez avec un compte n'ayant pas les autorisations `kms:ListAliases` et `kms:DescribeKey`, vous ne pouvez pas afficher les informations relatives à la clé gérée par le client sur la console Amazon SNS.
> Demandez au propriétaire de la clé gérée par le client de vous accorder ces autorisations. Pour plus d'informations, consultez la rubrique Autorisations d'API AWS KMS : référence des actions et ressources dans le Guide du développeur AWS Key Management Service.

c. Pour Clé gérée par le client, choisissez MyCustomKey que vous avez créée précédemment, puis choisissez Activer le chiffrement côté serveur.

6. Sélectionnez Save Changes (Enregistrer les modifications).

   SSE est activé pour votre rubrique et la page MaRubrique s'affiche.

   Le statut Chiffrement de la rubrique, Compte AWS, Clé gérée par le client, l'ARN et la Description de la clé gérée par le client s’affichent dans l’onglet Chiffrement.

Votre nouvelle rubrique chiffrée apparaît dans la liste des rubriques.

Étape 3 : Création et abonnement de files d'attente Amazon SQS chiffrées

1. Connectez-vous à la console Amazon SQS.
2. Choisissez Create New Queue (Créer une file d'attente).
3. Sur la page Create New Queue (Créer une file d'attente), procédez comme suit :

   a. Saisissez un nom de file d'attente (par exemple, `MyEncryptedQueue1`).

   b. Sélectionnez Standard Queue (File d'attente standard), puis Configure Queue (Configurer la file d'attente).

   c. Sélectionnez Use SSE (Utiliser le chiffrement SSE).

   d. Pour AWS KMS key, choisissez MyCustomKey que vous avez créée précédemment, puis choisissez Créer une file d'attente.

4. Répétez l'opération pour créer une deuxième file d'attente (par exemple, `MyEncryptedQueue2`).

Vos nouvelles files d'attente chiffrées s'affichent dans la liste des files d'attente.

5. Sur la console Amazon SQS, sélectionnez `MyEncryptedQueue1` et `MyEncryptedQueue2`, puis choisissez Actions sur la file d'attente, Abonner la file d'attente à la rubrique SNS.
6. Dans la boîte de dialogue S'abonner à une rubrique, pour Choisir une rubrique, sélectionné MyEncryptedTopic, puis choisissez S'abonner.

   Les abonnements de vos files d'attente chiffrées à votre rubrique chiffrée s'affichent dans la boîte de dialogue Résultat de l'abonnement à la rubrique.
7. Sélectionnez OK.

Étape 4 : Publication d'un message dans votre rubrique chiffrée

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Dans la liste des rubriques, sélectionnez MyEncryptedTopic, puis choisissez Publish message (Publier un message).
4. Sur la page Publier un message, procédez comme suit :

   a. (Facultatif) Dans la section Message details (Détails du message), saisissez le Subject (Objet) (for example, `Testing message publishing`).

   b. Dans la section Message body (Corps du message), saisissez le corps du message (par exemple, `My message body is encrypted at rest.`).

   c. Choisissez Publier le message.

Votre message est publié dans vos files d'attente chiffrées souscrites.

Étape 5 : Vérification de la livraison du message

1. Connectez-vous à la console Amazon SQS.
2. Dans la liste des files d'attente, choisissez MyEncryptedQueue1, puis Send and receive messages (Envoyer et recevoir des messages).
3. Sur la page Send and receive messages in MyEncryptedQueue1 (Envoyer et recevoir des messages dans MyEncryptedQueue1), choisissez Poll for messages (Rechercher des messages).

Le message que vous avez envoyé précédemment s'affiche.

4. Choisissez Plus de détails pour afficher votre message.
5. Lorsque vous avez terminé, choisissez Fermer.
6. Répétez l'opération pour MyEncryptedQueue2.

Confidentialité du trafic inter-réseau

Un point de terminaison d'Amazon Virtual Private Cloud (Amazon VPC) pour Amazon SNS est une entité logique au sein d'un VPC qui autorise la connectivité uniquement à Amazon SNS. Le VPC achemine les demandes vers Amazon SNS et les réponses en retour vers le VPC. Les sections suivantes fournissent des informations sur l'utilisation des points de terminaison de VPC et la création de politiques de point de terminaison de VPC.

Si vous utilisez Amazon Virtual Private Cloud (Amazon VPC) pour héberger vos ressources AWS, vous pouvez établir une connexion entre votre VPC et Amazon SNS. Avec cette connexion, vous pouvez publier des messages dans vos rubriques Amazon SNS sans les envoyer via le réseau Internet public.

Amazon VPC est un service AWS que vous pouvez utiliser pour lancer des ressources AWS dans un réseau virtuel que vous définissez. Avec un VPC, vous contrôlez des paramètres réseau, tels que la plage d'adresses IP, les sous-réseaux, les tables de routage et les passerelles réseau. Pour connecter votre VPC à Amazon SNS, vous définissez un point de terminaison de VPC d'interface. Ce type de point de terminaison vous permet de connecter votre VPC à des services AWS. Le point de terminaison assure une connectivité scalable et fiable à Amazon SNS, sans qu'une passerelle Internet, une instance NAT (Network Address Translation) ou une connexion VPN ne soit nécessaire. Pour plus d'informations, consultez Points de terminaison VPC dans le Guide de l'utilisateur Amazon VPC.

Les informations de cette section sont destinées aux utilisateurs d'Amazon VPC. Pour plus d'informations et pour commencer à créer un VPC, consultez Mise en route avec Amazon VPC dans le Guide de l'utilisateur Amazon VPC.

> **Note**
>
> Les points de terminaison d'un VPC ne vous permettent pas d'abonner une rubrique Amazon SNS à une adresse IP privée.

Rubriques

Création d'un point de terminaison Amazon VPC pour Amazon SNS

Pour publier des messages dans vos rubriques Amazon SNS à partir d'un Amazon VPC, créez un point de terminaison de VPC d'interface. Vous pouvez publier ensuite des messages dans vos rubriques tout en conservant le trafic au sein du réseau que vous gérez avec le VPC.

Utilisez les informations suivantes pour créer le point de terminaison et tester la connexion entre votre VPC et Amazon SNS. Ou, pour obtenir une procédure qui vous aidera à démarrer à partir de zéro, consultez Publication d'un message Amazon SNS à partir d'Amazon VPC.

Création du point de terminaison

Vous pouvez créer un point de terminaison Amazon SNS dans votre VPC à l'aide du AWS Management Console AWS SDK AWS CLI, de l'API Amazon SNS ou. AWS CloudFormation

Pour plus d'informations sur la création et la configuration d'un point de terminaison à l'aide de la console Amazon VPC ou de la AWS CLI, consultez la section Création d'un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

> **Important**
>
> Vous pouvez utiliser Amazon Virtual Private Cloud uniquement avec des points de terminaison HTTPS Amazon SNS.
> Lorsque vous créez un point de terminaison, spécifiez que Amazon SNS est le service auquel vous voulez que votre VPC se connecte. Dans la console Amazon VPC, les noms de service varient en fonction de la région choisie. Par exemple, si vous choisissez US Est (Virginie du Nord), le nom du service est com.amazonaws.us-east-1.sns.
> Lorsque vous configurez Amazon SNS pour que les messages soient envoyés depuis Amazon VPC, vous devez activer le DNS privé et spécifier les points de terminaison au format `sns.us-east-2.amazonaws.com`.
> Le DNS privé ne prend pas en charge les points de terminaison existants, tels que `queue.amazonaws.com` ou `us-east-2.queue.amazonaws.com`.

Pour plus d'informations sur la création et la configuration d'un point de terminaison à l'aide de cette `AWS::EC2::VPCEndpoint`ressource AWS CloudFormation, consultez la ressource du Guide de AWS CloudFormation l'utilisateur.

Test de la connexion entre votre VPC et Amazon SNS

Une fois que vous avez créé un point de terminaison pour Amazon SNS, vous pouvez publier des messages de votre VPC dans vos rubriques Amazon SNS. Pour tester cette connexion, procédez comme suit :

1. Connectez-vous à une instance Amazon EC2 qui se trouve dans votre VPC. Pour plus d'informations sur la connexion, consultez Connexion à votre instance Linux ou Connexion à votre instance Windows dans la documentation Amazon EC2.

   Par exemple, pour vous connecter à une instance Linux à l'aide d'un client SSH, exécutez la commande suivante à partir d'un terminal :

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Où :

   - ec2-key-pair.pem est le fichier qui contient la paire de clés fournie par Amazon EC2 lorsque vous avez créé l'instance.
   - instance-hostname est le nom d'hôte public de l'instance. Pour obtenir le nom d'hôte dans la console Amazon EC2 : choisissez Instances, sélectionnez votre instance et recherchez la valeur de DNS public (IPv4).
2. À partir de votre instance, utilisez la commande Amazon SNS `publish` avec la AWS CLI. Vous pouvez envoyer un message simple à une rubrique à l'aide de la commande suivante :

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   Où :

   - aws-region est AWS la région dans laquelle se trouve le sujet.
   - sns-topic-arnest le nom de ressource Amazon (ARN) du sujet. Pour obtenir l'ARN à partir de la console Amazon SNS : choisissez Topics (Rubriques), trouvez votre rubrique, puis recherchez la valeur dans la colonne ARN.

Si le message est correctement reçu par Amazon SNS, le terminal affiche un ID de message, semblable à ce qui suit :

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Création d'une politique de point de terminaison d'Amazon VPC pour Amazon SNS

Vous pouvez créer une politique pour les points de terminaison Amazon VPC pour Amazon SNS dans laquelle vous spécifiez les éléments suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, veuillez consulter Contrôle de l'accès aux services avec des points de terminaison d'un VPC dans le Amazon VPC Guide de l'utilisateur.

L'exemple suivant de politique de point de terminaison d'un VPC spécifie que l'utilisateur IAM `MyUser` est autorisé à publier dans la rubrique Amazon SNS `MyTopic`.

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

Ce qui suit est refusé :

- Autres actions d'API Amazon SNS, telles que `sns:Subscribe` et `sns:Unsubscribe`.
- Autres utilisateurs et règles IAM qui tentent d'utiliser ce point de terminaison de VPC.

- `MyUser` publiant dans une autre rubrique Amazon SNS.

> **Note**
>
> L'utilisateur IAM peut toujours utiliser d'autres actions d'API Amazon SNS depuis l'extérieur du VPC.

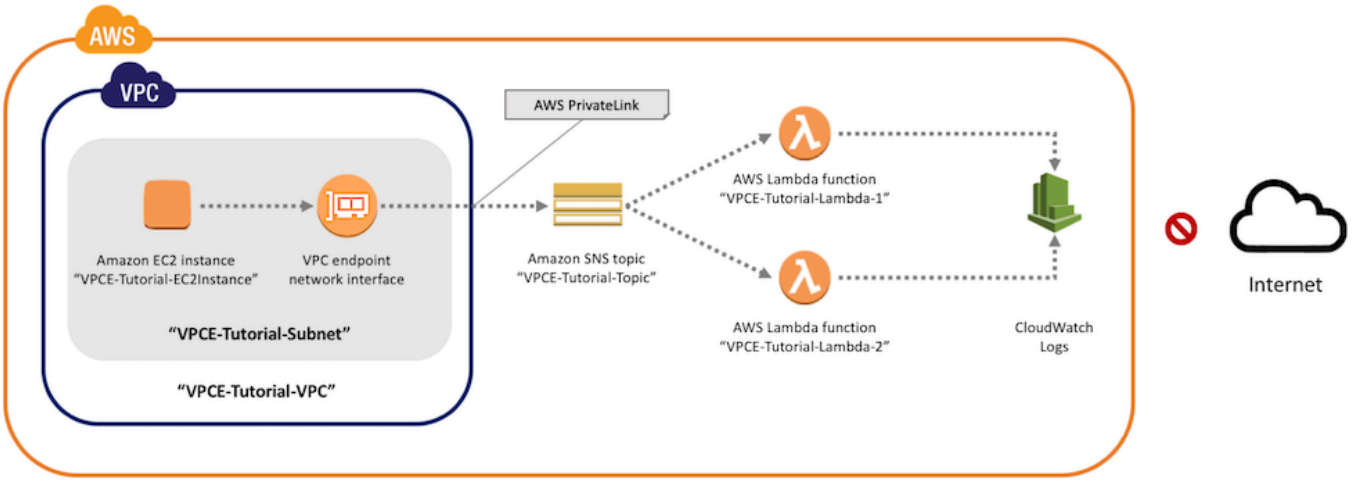
Publication d'un message Amazon SNS à partir d'Amazon VPC

Cette section vous explique comment publier dans une rubrique Amazon SNS tout en gardant les messages sécurisés dans un réseau privé. Vous publiez un message à partir d'une instance Amazon EC2 qui est hébergée dans Amazon Virtual Private Cloud (Amazon VPC). Le message reste au sein du réseau AWS sans passer par l'Internet public. En publiant des messages de façon privée à partir d'un VPC, vous pouvez améliorer la sécurité du trafic entre vos applications et Amazon SNS. Cette sécurité est importante lorsque vous publiez des informations personnelles identifiables (PII) à propos de vos clients, ou lorsque votre application est soumise à des réglementations du marché. Par exemple, la publication en mode privé est utile si vous disposez d'un système de santé qui doit se conformer à la loi HIPAA (Health Insurance Portability and Accountability Act), ou d'un système financier qui doit se conformer à la norme PCI DSS (Payment Card Industry Data Security Standard).

Les étapes générales à suivre sont les suivantes :

- Utilisez un modèle AWS CloudFormation pour créer automatiquement un réseau privé temporaire dans votre Compte AWS.
- Créez un point de terminaison de VPC qui connecte le VPC avec Amazon SNS.
- Connectez-vous à une instance Amazon EC2 et publiez un message en mode privé dans une rubrique Amazon SNS.
- Vérifier que le message a été remis avec succès.
- Supprimez les ressources que vous avez créées au cours de ce processus afin qu'elles ne restent pas dans votre Compte AWS.

Le schéma suivant illustre le réseau privé que vous créerez dans votre compte AWS quand vous aurez terminé ce didacticiel :



Ce réseau est composé d'un VPC qui contient une instance Amazon EC2. L'instance se connecte à Amazon SNS via un point de terminaison VPC d'interface. Ce type de point de terminaison se connecte à des services à technologie AWS PrivateLink. Une fois la connexion établie, vous pouvez vous connecter à l'instance Amazon EC2 et publier des messages dans la rubrique Amazon SNS, même si le réseau est déconnecté de l'Internet public. La rubrique diffuse les messages qu'elle reçoit à deux fonctions AWS Lambda abonnées. Ces fonctions journalisent les messages qu'elles reçoivent dans Amazon CloudWatch Logs.

Il vous faudra environ 20 minutes pour réaliser ces étapes.

Rubriques

Avant de commencer

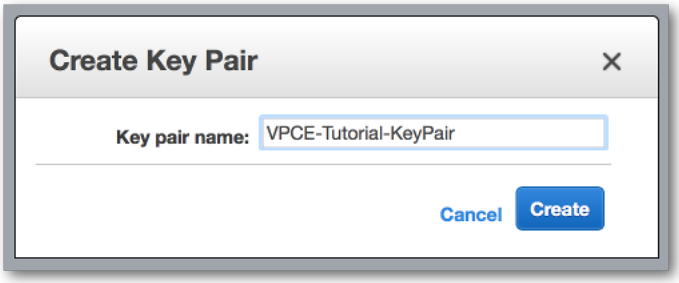
Avant de commencer, vous avez besoin d'un compte Amazon Web Services (AWS). Lorsque vous vous inscrivez, votre compte est automatiquement inscrit à tous les services dans AWS, notamment Amazon SNS et Amazon VPC. Si vous n'avez pas encore créé de compte, accédez à https://aws.amazon.com/, puis choisissez Créer un compte gratuit.

Étape 1 : Créer une paire de clés Amazon EC2

Une paire de clés est utilisée pour vous connecter à une instance Amazon EC2. Elle se compose d'une clé publique utilisée pour chiffrer vos informations de connexion et d'une clé privée utilisée pour les déchiffrer. Lorsque vous créez une paire de clés, vous téléchargez une copie de la clé privée. Par la suite, vous utiliserez la paire de clés pour vous connecter à une instance Amazon EC2. Pour vous connecter, vous spécifiez le nom de la paire de clés et vous fournissez la clé privée.

Pour créer la paire de clés

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le menu de navigation de gauche, recherchez la section Network & Security (Réseau et sécurité). Choisissez ensuite Paires de clés.
3. Choisissez Create Key Pair.
4. Dans la fenêtre Créer une paire de clés, pour le champ Nom de la paire de clés, tapez **`VPCE-Tutorial-KeyPair`**. Ensuite, choisissez Créer.



5. Le fichier de clé privée est automatiquement téléchargé dans votre navigateur. Enregistrez-la dans un emplacement sûr. Amazon EC2 attribut au fichier une extension `.pem`.
6. (Facultatif) Si vous utilisez un client SSH sur un ordinateur Mac ou Linux pour vous connecter à votre instance Linux, utilisez la commande `chmod` pour définir les autorisations de votre fichier de clé privée afin d'être la seule personne autorisée à le lire :

   a. Ouvrez un terminal et accédez au répertoire contenant la clé privée :

```
$ cd /filepath_to_private_key/
```

b. Définissez les autorisations à l'aide de la commande suivante :

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

Étape 2 : Créer les ressources AWS

Pour configurer l'infrastructure, vous utilisez un modèle AWS CloudFormation. Un modèle est un fichier qui tient lieu de plan pour créer des ressources AWS, comme des instances Amazon EC2 et des rubriques Amazon SNS. Le modèle pour ce processus est fourni sur GitHub pour que vous le téléchargiez.

Vous fournissez le modèle à AWS CloudFormation, puis AWS CloudFormation alloue les ressources dont vous avez besoin sous la forme d'une pile dans votre Compte AWS. Une pile est un ensemble de ressources que vous gérez comme une seule unité. Lorsque vous aurez terminé le didacticiel, vous pourrez utiliser AWS CloudFormation pour supprimer toutes les ressources de la pile en même temps. Ces ressources ne sont pas conservées dans votre Compte AWS, sauf si vous le souhaitez.

La pile pour ce processus inclut les ressources suivantes :

- Un VPC et les ressources de mise en réseau associées, notamment un sous-réseau, un groupe de sécurité, une passerelle Internet et une table de routage.
- Une instance Amazon EC2 qui est lancée dans le sous-réseau au sein du VPC.
- Une rubrique Amazon SNS
- Deux fonctions AWS Lambda. Ces fonctions reçoivent les messages qui sont publiés dans la rubrique Amazon SNS et journalisent des évènements dans CloudWatch Logs.
- Métriques et journaux Amazon CloudWatch.
- Un rôle IAM qui autorise l'instance Amazon EC2 à utiliser Amazon SNS et un rôle IAM qui autorise les fonctions Lambda à écrire dans les journaux CloudWatch.

Pour créer les ressources AWS

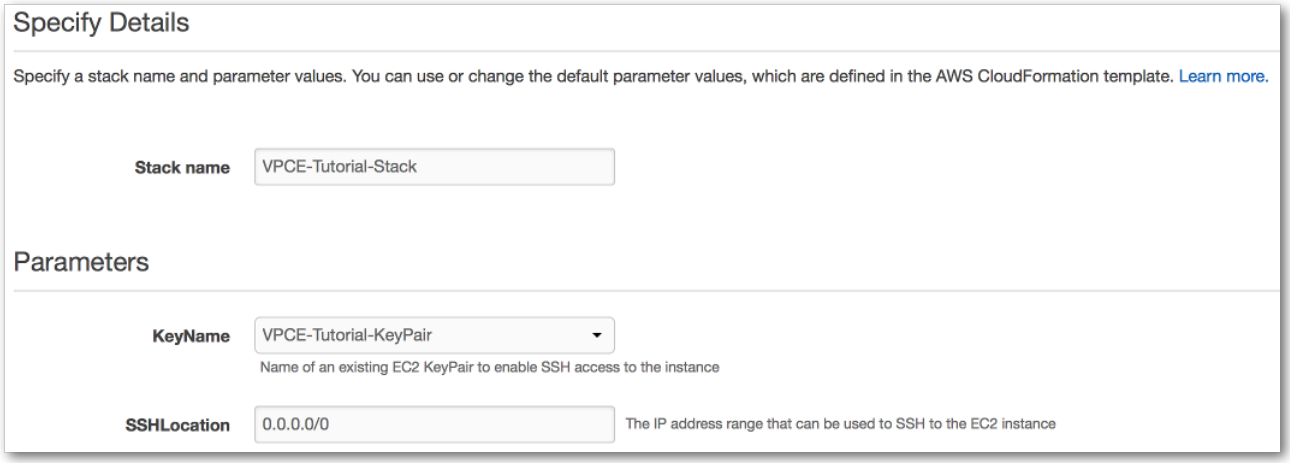
1. Téléchargez le modèle de fichier à partir du site web GitHub.
2. Connectez-vous à la AWS CloudFormationconsole.

3. Choisissez Create Stack (Créer une pile).
4. Sur la page Select Template (Sélectionner un modèle), choisissez Upload a template to Amazon S3 (Charger un modèle dans Amazon S3), sélectionnez le fichier, puis Next (Suivant).
5. Sur la page Specify Details (Spécifier les détails), spécifiez les noms de pile et de clé :

   a. Pour le Nom de la pile, tapez **`VPCE-Tutorial-Stack`**.

   b. Pour KeyName, choisissez VPCE-Tutorial-KeyPair.

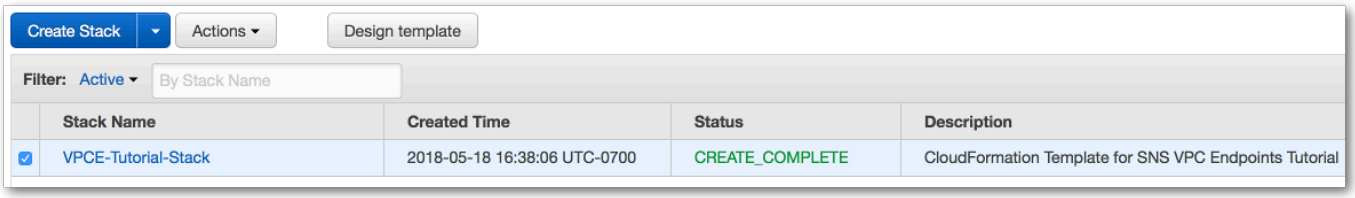
   c. Pour SSHLocation, conservez la valeur par défaut **`0.0.0.0/0`**.



   d. Choisissez Suivant.

6. Dans la page Options, conservez toutes les valeurs par défaut, puis choisissez Suivant.
7. Sur la page Review (Vérification), vérifiez les détails de la pile.
8. Sous Capacités, acceptez que AWS CloudFormation peut créer des ressources IAM avec des noms personnalisés.
9. Choisissez Create (Créer).

   La console AWS CloudFormation ouvre la page Piles. La pile VPCE-Tutorial-Stack a l'état CREATE_IN_PROGRESS. Après quelques minutes, une fois le processus de création terminé, l'état devient CREATE_COMPLETE.



> **Tip**
>
> Choisissez le bouton Actualiser pour afficher l'état le plus récent de la pile.

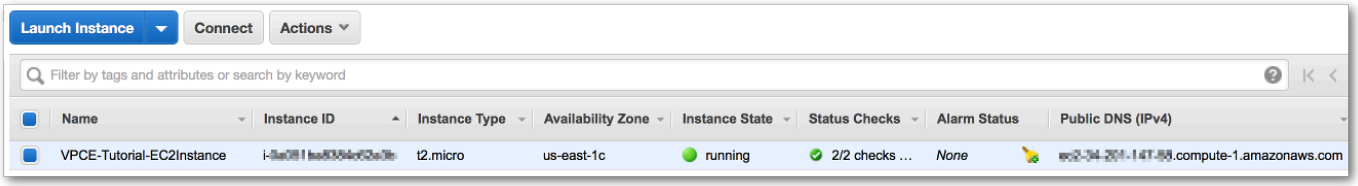
Étape 3 : Confirmation que votre instance Amazon EC2 n'a pas accès à Internet

L'instance Amazon EC2 qui a été lancée dans votre VPC à l'étape précédente n'a pas accès à Internet. Elle interdit le trafic sortant, et elle ne peut pas publier des messages dans Amazon SNS. Vous pouvez vérifier cela en vous connectant à l'instance. Essayez ensuite de vous connecter à un point de terminaison public et tentez d'envoyer un message à Amazon SNS.

À ce stade, la tentative de publication échoue. Dans une étape ultérieure, une fois que vous aurez créé un point de terminaison de VPC pour Amazon SNS, votre tentative de publication réussira.

Pour vous connecter à votre instance Amazon EC2

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le menu de navigation de gauche, recherchez la section Instances. Choisissez ensuite Instances.
3. Dans la liste des instances, sélectionnez VPCE-Tutorial-EC2Instance.
4. Copiez le nom d'hôte qui est fourni dans la colonne Public DNS (IPv4) (DNS public (IPv4)).



5. Ouvrez un terminal. Depuis le répertoire qui contient la paire de clés, connectez-vous à l'instance à l'aide de la commande suivante, où *instance-hostname* est le nom d'hôte que vous avez copié à partir de la console Amazon EC2 :

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

Pour vérifier que l'instance n'a pas de connectivité Internet

- Dans votre terminal, essayez de vous connecter à n'importe quel point de terminaison public, comme amazon.com :

```
$ ping amazon.com
```

Comme la tentative de connexion échoue, vous pouvez annuler à tout moment (Ctrl+C sur Windows ou Commande + C sur macOS).

Pour vérifier que l'instance n'a pas de connectivité vers Amazon SNS

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de navigation de gauche, choisissez Rubriques.
3. Sur la page Rubrique, copiez l'Amazon Resource Name (ARN) de la rubrique VPCE-Tutorial-Topic.
4. Dans votre terminal, essayez de publier un message dans la rubrique :

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Comme la tentative de publication échoue, vous pouvez annuler à tout moment.

Étape 4 : Création d'un point de terminaison Amazon VPC pour Amazon SNS

Pour connecter le VPC à Amazon SNS, vous définissez un point de terminaison de VPC d'interface. Une fois que vous avez ajouté le point de terminaison, vous pouvez vous connecter à l'instance Amazon EC2 dans votre VPC et à partir de là, vous pouvez utiliser l'API Amazon SNS. Vous pouvez publier des messages dans la rubrique, et les messages sont publiés en mode privé. Ils restent dans le réseau AWS et ils ne transitant pas par l'Internet public.

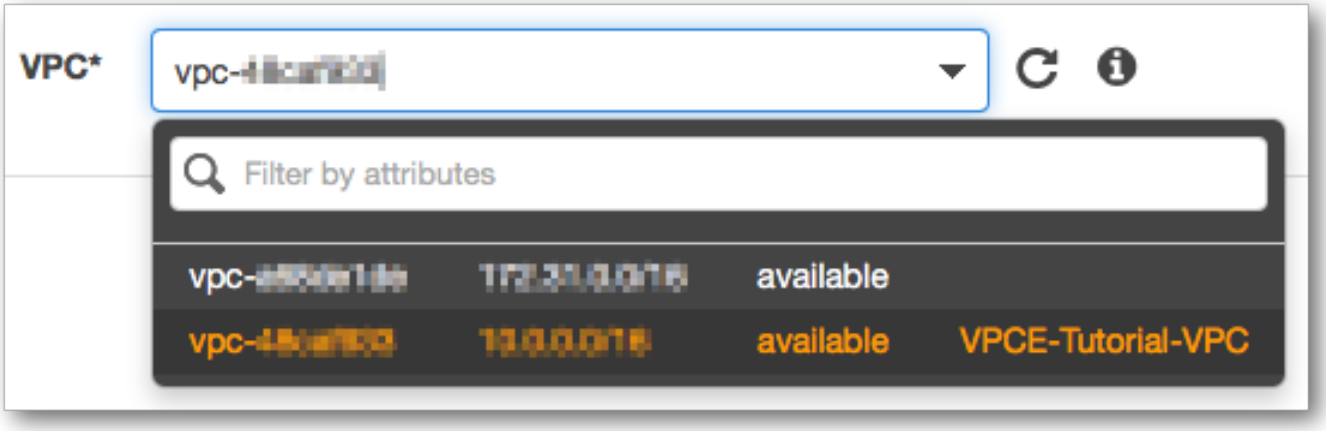
> **Note**
>
> L'instance n'a toujours pas accès à d'autres services AWS et points de terminaison sur Internet.

Pour créer le point de terminaison

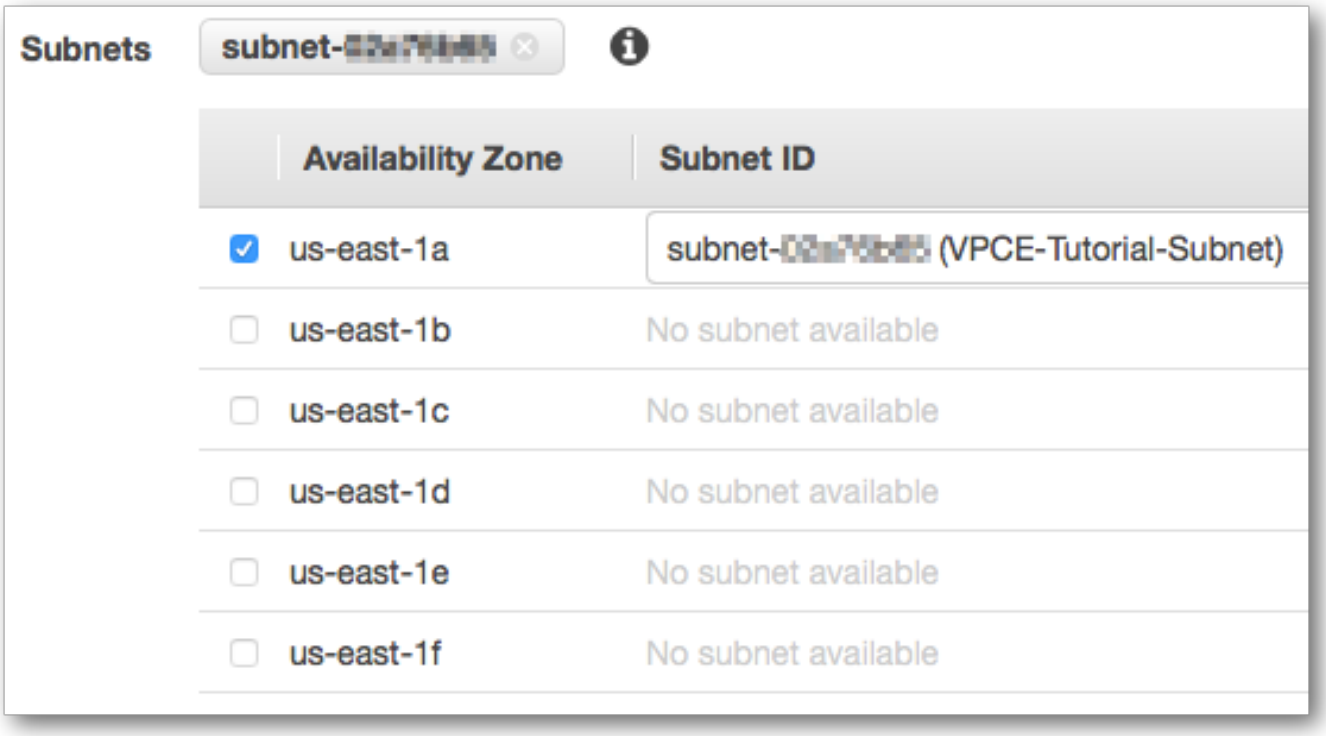
1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Dans le menu de navigation de gauche, choisissez Endpoints (Points de terminaison).
3. Choisissez Create Endpoint (Créer un point de terminaison).

4. Sur la page Créer un point de terminaison, pour Catégorie de services, conservez la valeur par défaut services AWS.
5. Pour Nom du service, choisissez le nom de service pour Amazon SNS.

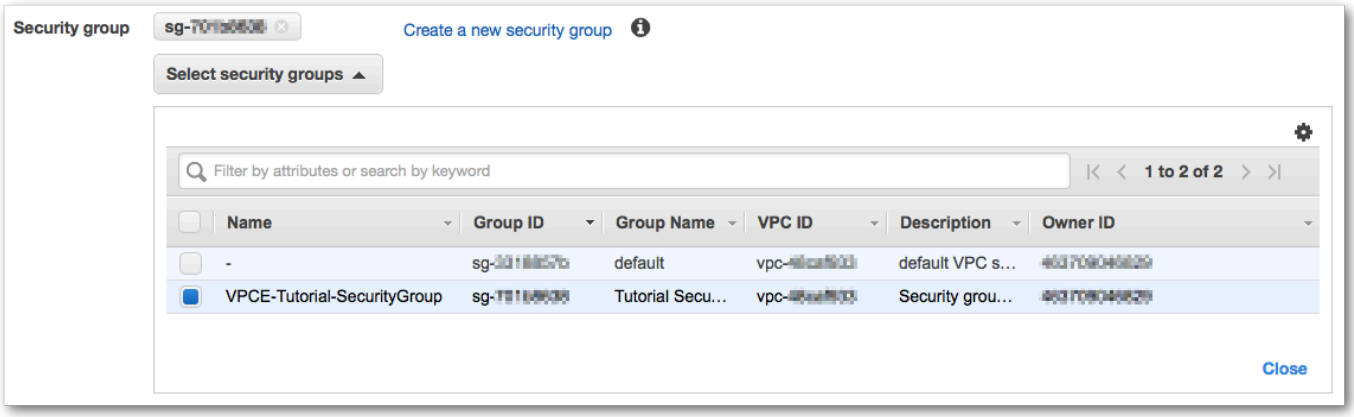
   Les noms de service varient en fonction de la région choisie. Par exemple, si vous choisissez US Est (Virginie du Nord), le nom du service est com.amazonaws.***us-east-1***.sns.
6. Pour VPC, choisissez le VPC qui a le nom VPCE-Tutorial-VPC.



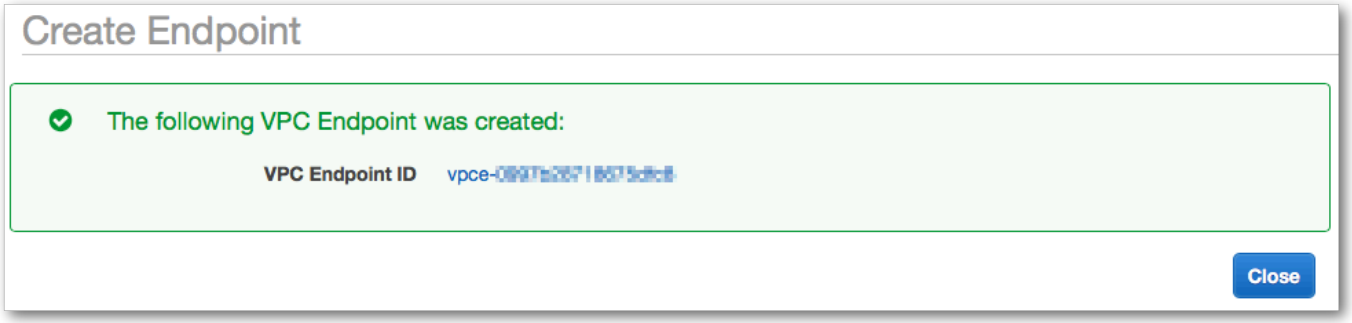
7. Pour Subnets (Sous-réseaux), choisissez le sous-réseau qui a VPCE-Tutorial-Subnet dans l'ID de sous-réseau.



8. Pour Enable Private DNS Name (Activer le nom DNS privé), sélectionnez Enable for this endpoint (Activer pour ce point de terminaison).
9. Pour Security group (Groupe de sécurité), choisissez Select security group (Sélectionner un groupe de sécurité) et choisissez VPCE-Tutorial-SecurityGroup.

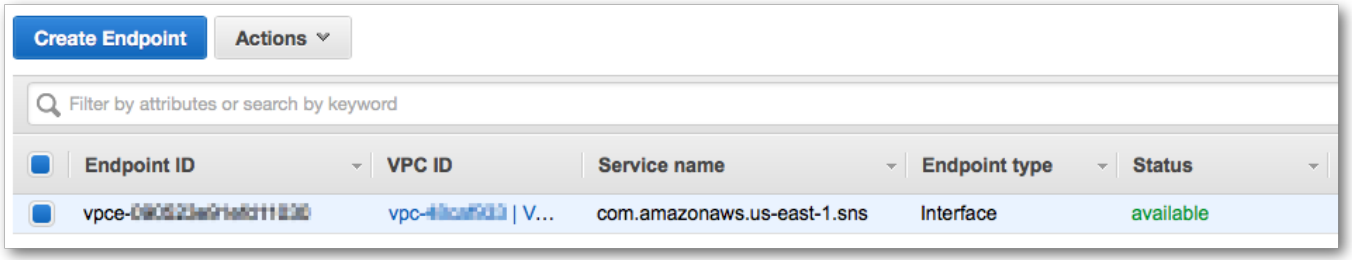


10. Choisissez Create endpoint (Créer un point de terminaison). La console Amazon VPC confirme qu'un point de terminaison de VPC a été créé.



11. Choisissez Fermer.

    La console Amazon VPC ouvre la page Points de terminaison. Le nouveau point de terminaison a l'état pending (en attente). Après quelques minutes, une fois le processus de création terminé, l'état devient available (disponible).



Étape 5 : Publication d'un message dans votre rubrique Amazon SNS

Maintenant que votre VPC inclut un point de terminaison pour Amazon SNS, vous pouvez vous connecter à l'instance Amazon EC2 et publier des messages dans la rubrique.

Pour publier un message

1. Si votre terminal n'est plus connecté à votre instance Amazon EC2, reconnectez-vous :

   ```
   $ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
   ```

2. Exécutez la même commande que celle que vous avez utilisée précédemment pour publier un message dans votre rubrique Amazon SNS. Cette fois, la tentative de publication aboutit et Amazon SNS renvoie un ID de message :

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

   {
       "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
   }
   ```

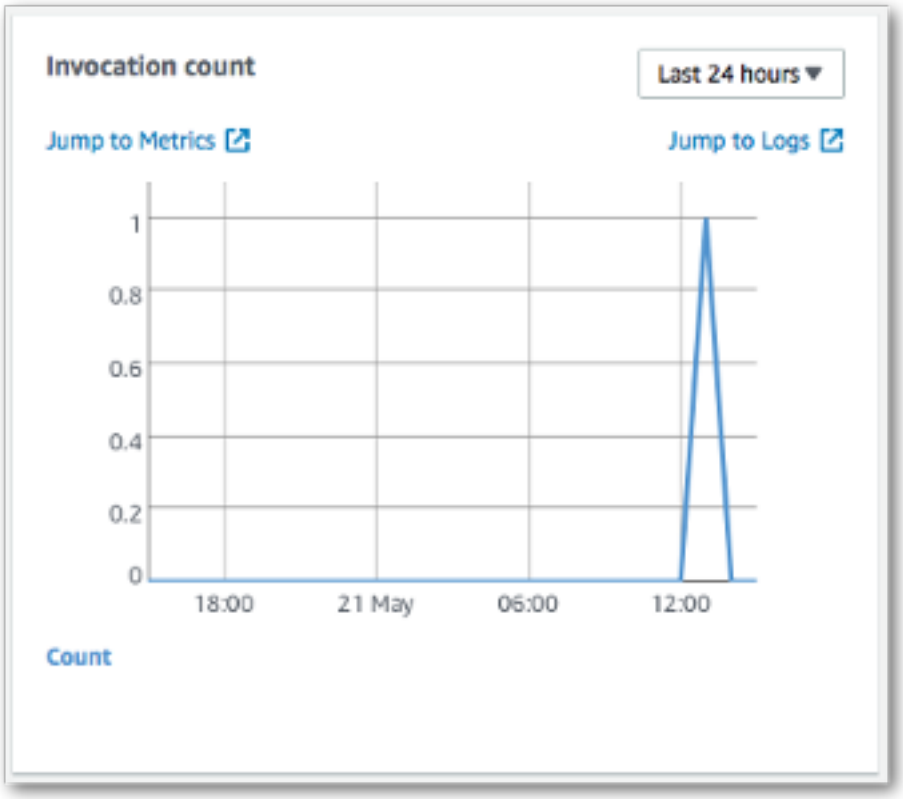
Étape 6 : Vérifier la livraison de vos messages

Lorsque la rubrique Amazon SNS reçoit un message, elle diffuse le message en l'envoyant aux deux fonctions Lambda abonnées. Lorsque ces fonctions reçoivent le message, elles journalisent l'évènement dans des journaux CloudWatch. Pour vérifier que la remise de messages a réussi, vérifiez que les fonctions ont été appelées et assurez-vous que les journaux CloudWatch ont été mis à jour.

Pour vérifier que les fonctions Lambda ont été appelées

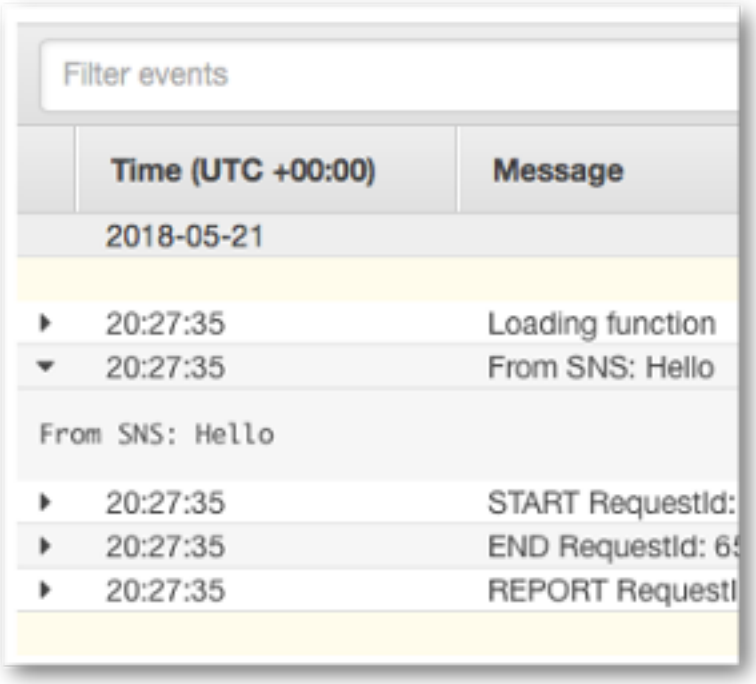
1. Ouvrez la console AWS Lambda à l'adresse https://console.aws.amazon.com/lambda/.
2. Sur la page Fonctions, choisissez VPCE-Tutorial-Lambda-1.
3. Choisissez Surveillance.
4. Vérifiez le graphique Nombre d'appels. Ce graphique montre le nombre de fois que la fonction Lambda a été exécutée.

   Le nombre d'appels correspond au nombre de fois où vous avez publié un message dans la rubrique.



Pour vérifier que les journaux CloudWatch ont été mis à jour

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/ CloudWatch/.
2. Dans le menu de navigation de gauche, choisissez Journaux.
3. Vérifiez les journaux qui ont été écrits par les fonctions Lambda :
   a. Choisissez le groupe de journaux /aws/lambda/VPCE-Tutorial-Lambda-1/.
   b. Choisissez le flux de journaux.
   c. Vérifiez que le journal inclut l'entrée `From SNS: Hello`.



d. Choisissez Log Groups (Groupes de journaux) en haut de la console pour renvoyer la page Log Groups (Groupes de journaux). Ensuite, répétez les étapes précédentes pour le groupe de journaux /aws/lambda/VPCE-Tutorial-Lambda-2/.

Félicitations ! En ajoutant un point de terminaison pour Amazon SNS à un VPC, vous avez pu publier un message dans une rubrique depuis le réseau qui est géré par le VPC. Le message a été publié en mode privé sans être exposé sur l'Internet public.

Étape 7 : Nettoyer

À moins que vous souhaitiez conserver les ressources que vous avez créées, vous pouvez les supprimer maintenant. En supprimant des ressources AWS que vous n'utilisez plus, vous évitez les frais superflus pour votre Compte AWS.

Tout d'abord, supprimez votre point de terminaison de VPC à l'aide de la console Amazon VPC. Ensuite, supprimez les autres ressources que vous avez créées en supprimant la pile dans la console AWS CloudFormation. Lorsque vous supprimez une pile, AWS CloudFormation supprime les ressources de la pile de votre Compte AWS.

Pour supprimer votre point de terminaison de VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

2. Dans le menu de navigation de gauche, choisissez Endpoints (Points de terminaison).
3. Sélectionnez le point de terminaison que vous avez créé.
4. Choisissez Actions, puis Delete Endpoint (Supprimer le point de terminaison).
5. Dans la fenêtre Delete Endpoint (Supprimer le point de terminaison), sélectionnez Yes, Delete (Oui, supprimer).

   L'état du point de terminaison passe à deleting (en cours de suppression). Une fois la suppression terminée, le point de terminaison est supprimé de la page.

Pour supprimer votre pile AWS CloudFormation

1. Ouvrez la console AWS CloudFormation à l'adresse https://console.aws.amazon.com/cloudformation.
2. Sélectionnez la pile VPCE-Tutorial-Stack.
3. Choisissez Actions, puis Supprimer pile.
4. Dans la fenêtre Delete Stack (Supprimer la pile), sélectionnez Yes, Delete (Oui, supprimer).

   L'état de la pile passe à DELETE_IN_PROGRESS. Une fois la suppression terminée, la pile est supprimée de la page.

Ressources connexes

Pour plus d'informations, veuillez consulter les ressources suivantes.

- Blog sur la sécurité AWS : sécurisation de messages publiés sur Amazon SNS à l’aide d’AWS PrivateLink
- Qu'est-ce qu'Amazon VPC?
- Points de terminaison d'un VPC
- Qu'est-ce qu'Amazon EC2?
- AWS CloudFormation Concepts

Sécurité Message Data Protection

- Message Data Protection est une fonctionnalité d'Amazon SNS utilisée pour définir vos propres règles et politiques, afin d'auditer et de contrôler le contenu des données en mouvement, par opposition aux données au repos.
- Message Data Protection fournit des services de gouvernance, de conformité et d'audit pour les applications d'entreprise centrées sur les messages, afin que l'entrée et la sortie des données puissent être contrôlées par le propriétaire de la rubrique Amazon SNS et que les flux de contenu puissent être suivis et enregistrés.
- Vous pouvez rédiger des règles de gouvernance basées sur la charge utile pour éviter que le contenu non autorisé de la charge utile n’entre dans vos flux de messages.
- Vous pouvez accorder différentes autorisations d'accès au contenu à différents abonnés et auditer l'ensemble du processus de flux de contenu.

Gestion des identités et des accès dans Amazon SNS

L'accès à Amazon SNS requiert des informations d'identification qu'AWS peut utiliser pour authentifier vos demandes. Ces informations d'identification doivent disposer d'autorisations pour accéder aux ressources AWS, telles que les rubriques et les messages Amazon SNS. Les sections suivantes vous expliquent comment utiliser AWS Identity and Access Management (IAM) et Amazon SNS pour sécuriser vos ressources en contrôlant qui peut y accéder.

AWS Identity and Access Management (IAM) est un Service AWS qui aide un administrateur à contrôler en toute sécurité l’accès aux ressources AWS. Les administrateurs IAM contrôlent les personnes qui peuvent être authentifiées (connectées) et autorisées (disposant d'autorisations) pour utiliser des ressources Amazon SNS. IAM est un Service AWS que vous pouvez utiliser sans frais supplémentaires.

Public ciblé

Votre utilisation de AWS Identity and Access Management (IAM) diffère selon la tâche que vous effectuez dans Amazon SNS.

Utilisateur du service : si vous utilisez le service Amazon SNS pour accomplir votre tâche, votre administrateur vous fournira les informations d'identification et les autorisations nécessaires. Vous pourrez avoir besoin d'autorisations supplémentaires si vous utilisez davantage de fonctions Amazon SNS. Si vous comprenez la gestion des accès, vous pourrez demander les

autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonction dans Amazon SNS, consultez Résolution des problèmes d'accès et d'identité Amazon Simple Notification Service (Gestion des identités et des accès dans Amazon SNS).

Administrateur du service : si vous êtes le responsable des ressources Amazon SNS de votre entreprise, vous bénéficiez probablement d'un accès total à ce service. C'est à vous de déterminer les fonctions et les ressources Amazon SNS auxquelles vos utilisateurs des services pourront accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour découvrir la façon dont votre entreprise peut utiliser IAM avec Amazon SNS, consultez Fonctionnement d'Amazon Simple Notification Service avec IAM (Fonctionnement d'Amazon Simple Notification Service avec IAM).

Administrateur IAM : Si vous êtes un administrateur IAM, vous pouvez souhaiter obtenir des informations sur l'écriture de politiques pour gérer l'accès à Amazon SNS. Pour afficher des exemples de politiques basées sur l'identité Amazon SNS que vous pouvez utiliser dans IAM, consultez Exemples de politiques basées sur une identité pour Amazon Simple Notification Service (Exemples de politiques basées sur l'identité pour Amazon Simple Notification Service).

Authentification par des identités

L'authentification correspond au processus par lequel vous vous connectez à AWS avec vos informations d'identification. Vous devez vous authentifier (être connecté à AWS) en tant qu'utilisateur root du Compte AWS, utilisateur IAM ou en endossant un rôle IAM.

Vous pouvez vous connecter à AWS en tant qu'identité fédérée à l'aide des informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification de connexion unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS en utilisant la fédération, vous endossez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter à la AWS Management Console ou au portail d'accès AWS. Pour plus d'informations sur la connexion à AWS, consultez Connexion à votre Compte AWSdans le Guide de l'utilisateurConnexion à AWS.

Si vous accédez à AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes en

utilisant vos informations d'identification. Si vous n'utilisez pas les outils AWS, vous devez signer les requêtes vous-même. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez Signature des demandes d'API AWS dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, AWS vous recommande d'utiliser l'authentification multifactorielle (MFA) pour améliorer la sécurité de votre compte. Pour en savoir plus, veuillez consulter Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) dans l'interface AWS dans le Guide de l'utilisateur IAM.

Utilisateur root Compte AWS

Lorsque vous créez un Compte AWS, vous commencez avec une seule identité de connexion disposant d'un accès complet à tous les Services AWS et ressources du compte. Cette identité est appelée utilisateur root du Compte AWS. Vous pouvez y accéder en vous connectant à l'aide de l'adresse électronique et du mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur root pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur root et utilisez-les pour effectuer les tâches que seul l'utilisateur root peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez Tâches nécessitant des informations d'identification d'utilisateur root dans le Guide de l'utilisateur IAM.

Identité fédérée

Demandez aux utilisateurs humains, et notamment aux utilisateurs qui nécessitent un accès administrateur, d'appliquer la bonne pratique consistant à utiliser une fédération avec fournisseur d'identité pour accéder à Services AWS en utilisant des informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, un fournisseur d'identité Web, l'AWS Directory Service, l'annuaire Identity Center ou tout utilisateur qui accède à Services AWS en utilisant des informations d'identification fournies via une source d'identité. Quand des identités fédérées accèdent à Comptes AWS, elles endossent des rôles, ces derniers fournissant des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous connecter et vous synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre

source d'identité pour une utilisation sur l'ensemble de vos applications et de vos Comptes AWS. Pour obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité dans votre Compte AWS qui dispose d'autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une entité au sein de votre Compte AWS qui dispose d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez temporairement endosser un rôle IAM dans la AWS Management Console en changeant de rôle. Vous pouvez obtenir un rôle en appelant une opération d'API AWS CLI ou AWS à l'aide d'une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie,

l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center.

- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, certains Services AWS vous permettent d'attacher une politique directement à une ressource (au lieu d'utiliser un rôle en tant que proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.
- Accès interservices : certains Services AWS utilisent des fonctions dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, une fonction du service ou un rôle lié au service.
  - Forward access sessions (FAS) – Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions dans AWS, vous êtes considéré comme un principal. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui déclenche une autre action dans un autre service. FAS utilise les autorisations du principal appelant Service AWS, combinées à la demande Service AWS pour effectuer des demandes aux services en aval. Les demandes FAS ne sont formulées que lorsqu'un service reçoit une demande qui, pour aboutir, a besoin d'interagir avec d'autres ressources ou Services AWS. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez Sessions de transmission d'accès.
  - Fonction du service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

- Rôle lié au service : un rôle lié au service est un type de fonction de service lié à un Service AWS. Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre Compte AWS et sont détenus par le service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications s'exécutant sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer des informations d'identification temporaires pour les applications s'exécutant sur une instance EC2 et effectuant des demandes d'API AWS CLI ou AWS. Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un rôle AWS à une instance EC2 et le rendre disponible à toutes les applications associées, vous pouvez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez les accès dans AWS en créant des politiques et en les attachant à des identités AWS ou à des ressources. Une politique est un objet dans AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit les autorisations de ces dernières. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur racine ou séance de rôle) envoie une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées dans AWS en tant que documents JSON. Pour plus d'informations sur la structure et le contenu des déclarations de politique JSON, consultez Présentation des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent endosser les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur avec cette politique peut obtenir des informations utilisateur à partir de la AWS Management Console, de la AWS CLI ou de l'API AWS.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des déclarations de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez attacher à plusieurs utilisateurs, groupes et rôles dans votre Compte AWS. Les politiques gérées incluent les politiques gérées par AWS et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou des Services AWS.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques gérées AWS depuis IAM dans une politique basée sur une ressource.

Listes de contrôle d’accès (ACL)

Les listes de contrôle d’accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l’autorisation d’accéder à une ressource. Les listes de contrôle d’accès sont similaires aux politiques basées sur les ressources, bien qu’elles n’utilisent pas le format de document de politique JSON.

Amazon S3, AWS WAF et Amazon VPC sont des exemples de services prenant en charge les ACL. Pour en savoir plus sur les listes de contrôle d’accès, consultez Présentation des listes de contrôle d’accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d’autres types de politiques moins courantes. Ces types de politiques peuvent définir le nombre maximum d’autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d’autorisations : une limite d’autorisations est une fonction avancée dans laquelle vous définissez le nombre maximal d’autorisations qu’une politique basée sur l’identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d’autorisations pour une entité. Les autorisations qui en résultent représentent la combinaison des politiques basées sur l’identité d’une entité et de ses limites d’autorisation. Les politiques basées sur les ressources qui spécifient l’utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d’autorisations. Un refus explicite dans l’une de ces politiques remplace l’autorisation. Pour plus d’informations sur les limites d’autorisations, consultez Limites d’autorisations pour des entités IAM dans le Guide de l’utilisateur IAM.
- Politiques de contrôle des services (SCP) - les SCP sont des politiques JSON qui spécifient le nombre maximal d’autorisations pour une organisation ou une unité d’organisation (OU) dans AWS Organizations. AWS Organizations est un service qui vous permet de regrouper et de gérer de façon centralisée plusieurs Comptes AWS détenus par votre entreprise. Si vous activez toutes les fonctions d'une organisation, vous pouvez appliquer les politiques de contrôle de service (SCP) à l'un ou à l'ensemble de vos comptes. Les politiques de contrôle des services (SCP) limitent les autorisations pour les entités dans les comptes membres, y compris chaque utilisateur racine de compte Compte AWS. Pour plus d'informations sur les organisations et les SCP, consultez Fonctionnement des SCP dans le Guide de l'utilisateur AWS Organizations.
- politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la séance obtenue sont une combinaison des politiques

basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations, consultez Politiques de séance dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations obtenues sont plus compliquées à comprendre. Pour découvrir la façon dont AWS détermine s'il convient d'autoriser une demande en présence de plusieurs types de politiques, veuillez consulter Logique d'évaluation de politiques dans le Guide de l'utilisateur IAM.

Contrôle d'accès

Amazon SNS dispose de son propre système d'autorisations basées sur les ressources pour utiliser des politiques écrites dans le même langage que celui des politiques AWS Identity and Access Management (IAM). Cela signifie que vous pouvez obtenir le même type de résultat avec des politiques Amazon SNS et des politiques IAM.

> **Note**
>
> Il est important de comprendre que tous les Comptes AWS peuvent déléguer leurs autorisations à des utilisateurs possédant leurs propres comptes. L'accès intercomptes vous permet de partager l'accès à vos ressources AWS sans avoir à gérer d'utilisateurs supplémentaires. Pour plus d'informations sur l'accès entre comptes, consultez Activation de l'accès entre comptes dans le Guide de l'utilisateur IAM.

Présentation de la gestion de l'accès dans Amazon SNS

Cette section décrit les concepts de base que vous devez comprendre pour utiliser le langage de la politique d'accès pour écrire des politiques. Elle décrit également le processus général de fonctionnement du contrôle d'accès avec le langage de la politique d'accès, ainsi que le mode d'évaluation des politiques.

Rubriques

- Quand utiliser un contrôle d'accès
- Concepts clés

- Présentation de l'architecture
- Utilisation du langage de la politique d'accès
- Logique d'évaluation
- Cas d'exemple pour le contrôle d'accès Amazon SNS

Quand utiliser un contrôle d'accès

Vous disposez d'une grande latitude dans la façon dont vous pouvez autoriser ou refuser l'accès à une ressource. Cependant, les cas d'utilisation classiques sont assez simples :

- Vous souhaitez accorder à un autre compte Compte AWS l'autorisation d'effectuer un type particulier d'action sur une rubrique (par exemple, Publier). Pour plus d’informations, consultez Autoriser l'accès Compte AWS à une rubrique.
- Vous souhaitez limiter les abonnements à votre rubrique au seul protocole HTTPS. Pour plus d’informations, consultez Limiter les abonnements à HTTPS.
- Vous souhaitez autoriser Amazon SNS à publier des messages dans votre file d'attente Amazon SQS. Pour plus d’informations, consultez Publier des messages dans une file d'attente Amazon SQS.

Concepts clés

Les sections suivantes décrivent les concepts que vous devez connaître pour utiliser le langage de la politique d'accès. Ils vous sont présentés dans un ordre logique, avec les premiers termes que vous devez savoir en haut de la liste.

Rubriques

- Autorisation
- Instruction
- Politique
- Emetteur
- Principal
- Action
- Ressource
- Conditions et clés

- Demandeur
- Evaluation
- Effet
- Refus par défaut
- Autorisation
- Refus explicite

Autorisation

Une autorisation est le concept consistant à autoriser ou à refuser un type d'accès à une ressource spécifique. Les autorisations suivent fondamentalement cette forme : « A est/n'est pas autorisé à faire B pour C lorsque D s'applique ». Par exemple, Jane (A) est autorisée à publier (B) dans TopicA (C) pour autant qu'elle utilise le protocole HTTP (D). Chaque fois que Jane effectue une publication dans TopicA, le service vérifie si elle possède l'autorisation appropriée et si la demande est conforme aux conditions définies dans l'autorisation.

Instruction

Une instruction est la description formelle d'une autorisation unique, écrite dans le langage de la politique d'accès. Vous écrivez toujours une instruction dans le cadre d'un document conteneur plus vaste connu sous le nom de politique (voir le concept suivant).

Politique

Une politique est un document (écrit dans le langage de la politique d'accès) qui joue le rôle de conteneur pour une ou plusieurs instructions. Par exemple, une politique peut comporter deux instructions : une indiquant que Jane peut s'abonner à l'aide du protocole de messagerie et une autre indiquant que Bob ne peut pas publier dans la rubrique A. Comme illustré dans la figure suivante, un scénario équivalent impliquerait deux politiques : une indiquant que Jane peut s'abonner à l'aide du protocole de messagerie, et l'autre indiquant que Bob ne peut pas publier dans la rubrique A.

Seuls les caractères ASCII sont autorisés dans les documents de politique. Vous pouvez utiliser `aws:SourceAccount` et `aws:SourceOwner` pour contourner le scénario dans lequel vous devez brancher d'autres ARN de services AWS contenant des caractères non-ASCII. Reportez-vous à la différence entre `aws:SourceAccount` par rapport à `aws:SourceOwner`.

Emetteur

L'émetteur est la personne qui écrit une politique pour accorder des autorisations pour une ressource. Par définition, l'auteur est toujours le propriétaire de la ressource. AWS ne permet pas à des utilisateurs de service AWS de créer des politiques pour les ressources qui ne leur appartiennent pas. Si John est le propriétaire de la ressource, AWS authentifie son identité lorsqu'il envoie la politique qu'il a écrite pour accorder des autorisations d'accès à cette ressource.

Principal

Le principal est la ou les personnes qui reçoivent l'autorisation dans la politique. Le principal correspond à A dans l'instruction « A est autorisé à faire B pour C lorsque D s'applique ». Vous pouvez configurer une politique pour que le principal s'applique à tout le monde. En d'autres termes, vous pouvez spécifier un caractère générique pour représenter tous les utilisateurs. Cette approche est notamment utile si vous ne voulez pas limiter l'accès en fonction de l'identité réelle du demandeur, mais plutôt en fonction de certaines autres caractéristiques d'identification telles que son adresse IP.

Action

L'action est l'activité que le principal est autorisé à effectuer. L'action correspond à B dans l'instruction « A est autorisé à faire B pour C lorsque D s'applique ». En général, l'action est simplement l'opération dans la demande envoyée à AWS. Par exemple, Jane envoie une demande à Amazon SNS avec `Action=Subscribe`. Vous pouvez spécifier une ou plusieurs actions dans une politique.

Ressource

La ressource est l'objet pour lequel le principal demande l'accès. La ressource correspond à C dans l'instruction « A est autorisé à faire B pour C lorsque D s'applique ».

Conditions et clés

Les conditions sont des restrictions ou des détails relatifs à l'autorisation. La condition correspond à D dans l'instruction « A est autorisé à faire B pour C lorsque D s'applique ». La partie de la politique qui spécifie les conditions est parfois la plus détaillée et la plus complexe. Les conditions typiques sont liées aux éléments suivants :

- Date et heure (par exemple, la demande doit arriver avant une date spécifique)
- Adresse IP (par exemple, l'adresse IP du demandeur doit faire partie d'une plage d'adresses CIDR particulière)

Une clé est la caractéristique spécifique qui est la base d'une restriction d'accès. Par exemple, la date et l'heure de la demande.

Vous utilisez à la fois les conditions et les clés afin d'exprimer la restriction. La méthode la plus simple pour comprendre comment mettre en œuvre une restriction est un exemple concret : si vous souhaitez limiter l'accès à avant le 30 mai 2010, utilisez la condition appelée `DateLessThan`. Vous utilisez la clé appelée `aws:CurrentTime` et vous la définissez à la valeur `2010-05-30T00:00:00Z` . AWS définit les conditions et les clés que vous pouvez utiliser. Le service AWS lui-même (par exemple, Amazon SQS ou Amazon SNS) peut aussi définir des clés qui lui sont spécifiques. Pour plus d'informations, consultez Autorisations d'API Amazon SNS : référence des actions et ressources.

Demandeur

Le demandeur est la personne qui envoie une demande à un service AWS afin d'accéder à une ressource donnée. Le demandeur envoie une demande à AWS qui équivaut fondamentalement à dire : « M'autorisez-vous à faire B pour C lorsque D s'applique ? »

Evaluation

L'évaluation est le processus que le service AWS utilise pour déterminer si une demande entrante doit être refusée ou autorisée en fonction des politiques applicables. Pour plus d'informations sur la logique d'évaluation, consultez la section Logique d'évaluation.

Effet

L'effet est le résultat que vous voulez qu'une instruction de politique renvoie au moment de l'évaluation. Vous spécifiez cette valeur lorsque vous écrivez les instructions dans une politique. Les valeurs possibles sont deny (refuser) et allow (autoriser).

Par exemple, vous pouvez écrire une stratégie dont l'instruction refuse toutes les demandes qui viennent de l'Antarctique (effet = refus étant donné que la demande utilise une adresse IP attribuée à l'Antarctique). Vous pouvez également écrire une politique dont l'instruction autorise toutes les demandes qui ne proviennent pas de l'Antarctique (effet=autorisation, étant donné que la demande ne vient pas de l'Antarctique). Même si les deux instructions semblent avoir le même résultat, selon la logique du langage de la politique d'accès, elles sont différentes. Pour plus d’informations, consultez Logique d'évaluation.

Bien que vous ne puissiez définir que deux valeurs distinctes pour l'effet (autorisation ou refus), trois résultats différents sont possibles lors de l'évaluation de politique : refus par défaut, autorisation ou refus explicite. Pour plus d'informations, reportez-vous aux concepts ci-dessous, ainsi qu'à la section Logique d'évaluation.

Refus par défaut

Un refus par défaut est le résultat par défaut d'une politique en l'absence d'autorisation ou de refus explicite.

Autorisation

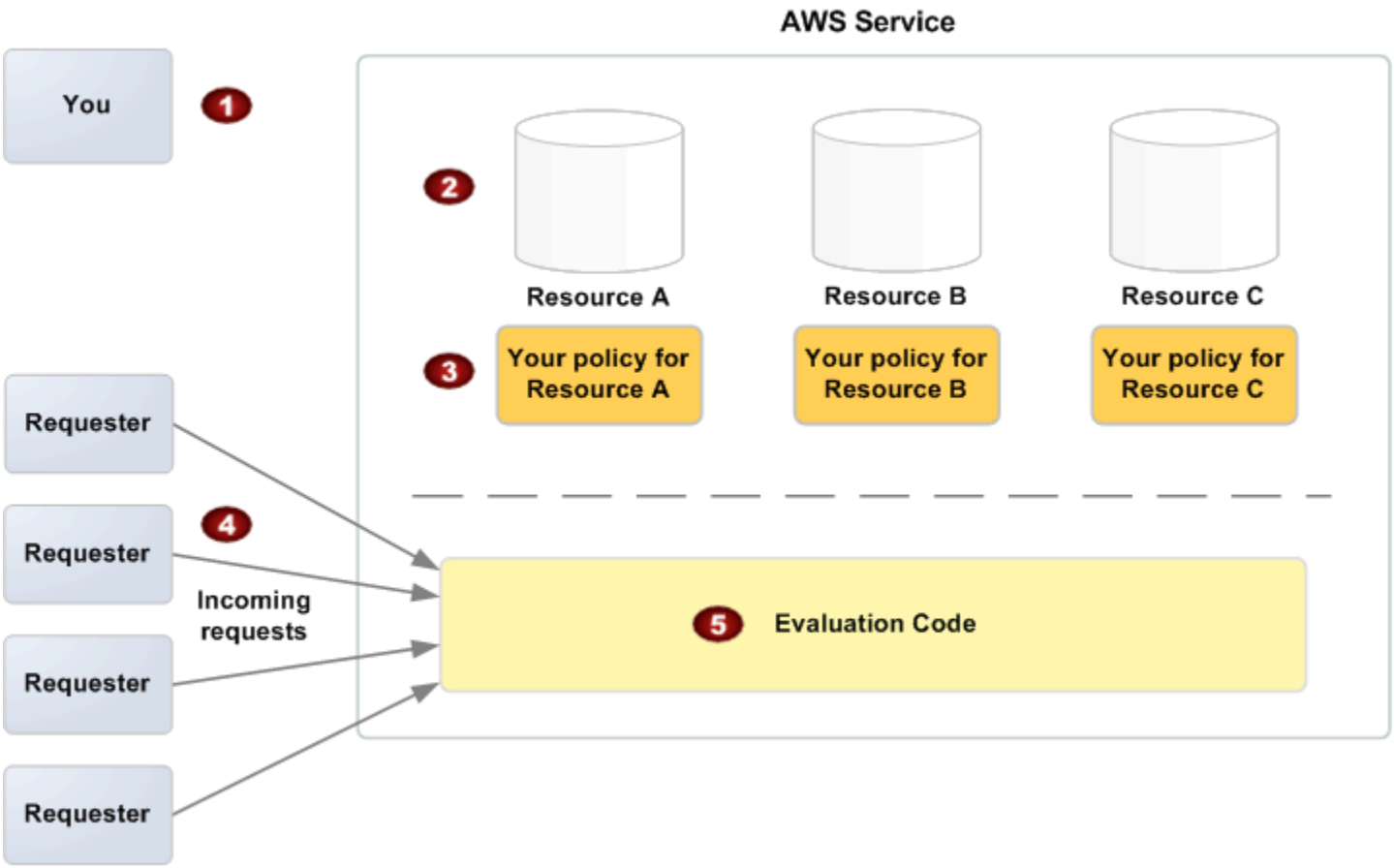
Une autorisation est le résultat d'une instruction avec l'effet=allow, sous réserve que toutes les conditions soient remplies. Exemple : autoriser les demandes si elles sont reçues avant 13 h le 30 avril 2010. Une autorisation prévaut sur tous les refus par défaut, mais jamais sur un refus explicite.

Refus explicite

Un refus explicite est le résultat d'une instruction qui avec l'effet=refuser, sous réserve que toutes les conditions soient remplies. Exemple : refuser toutes les demandes si elles viennent de l'Antarctique. Toute demande qui vient de l'Antarctique sera toujours refusée, indépendamment des autorisations octroyées par les autres politiques.

Présentation de l'architecture

La figure et le tableau suivants décrivent les principaux composants qui interagissent pour assurer un contrôle d'accès pour vos ressources.

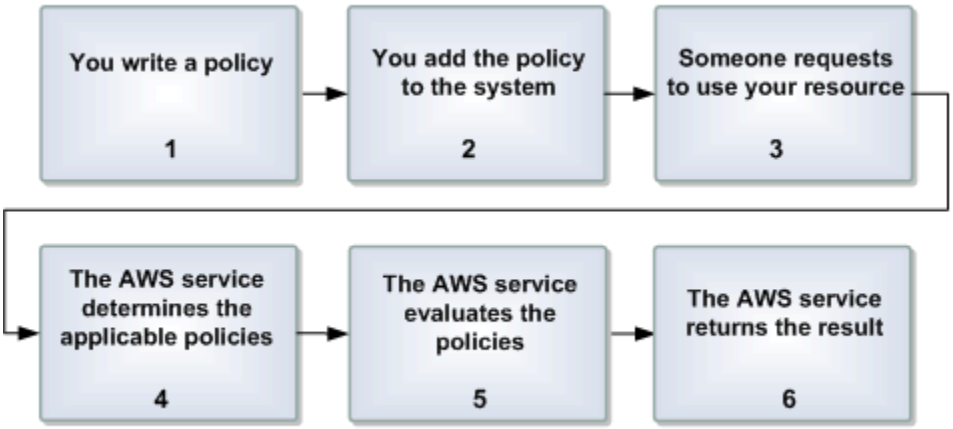


1	Vous-même, le propriétaire de la ressource.
2	Vos ressources contenues dans le service AWS (par exemple, les files d'attente Amazon SQS).
3	Vos politiques. Généralement, vous utilisez une politique par ressource, même s'il est possible d'en avoir plusieurs. Le service AWS lui-même fournit une API que vous utilisez pour importer et gérer vos politiques.
4	Les demandeurs et leurs demandes entrantes au service AWS.
5	Code d'évaluation du langage de la politique d'accès. Il s'agit du code du service AWS qui évalue les demandes entrantes par rapport aux politiques applicables et de détermine si le demandeur est autorisé à accéder à la

ressource. Pour plus d'informations sur la façon dont le service prend la décision, consultez la section Logique d'évaluation.

Utilisation du langage de la politique d'accès

La figure et le tableau suivants décrivent le fonctionnement général du contrôle d'accès avec le langage de la politique d'accès.



Procédure d'utilisation du contrôle d'accès avec le langage de la politique d'accès

1	Vous écrivez une politique pour la ressource. Par exemple, vous en écrivez une pour spécifier les autorisations des rubriques Amazon SNS.
2	Vous importez la politique dans AWS. Le service AWS lui-même fournit une API que vous utilisez pour importer vos politiques. Par exemple, vous utilisez l'action Amazon SNS `SetTopicAttributes` pour télécharger une politique pour une politique Amazon SNS spécifique.
3	Quelqu'un envoie une demande d'utilisation de la ressource. Par exemple, un utilisateur envoie à Amazon SNS une demande d'utilisation de l'une de vos rubriques.
4	Le service AWS détermine quelles politiques sont applicables à la demande.

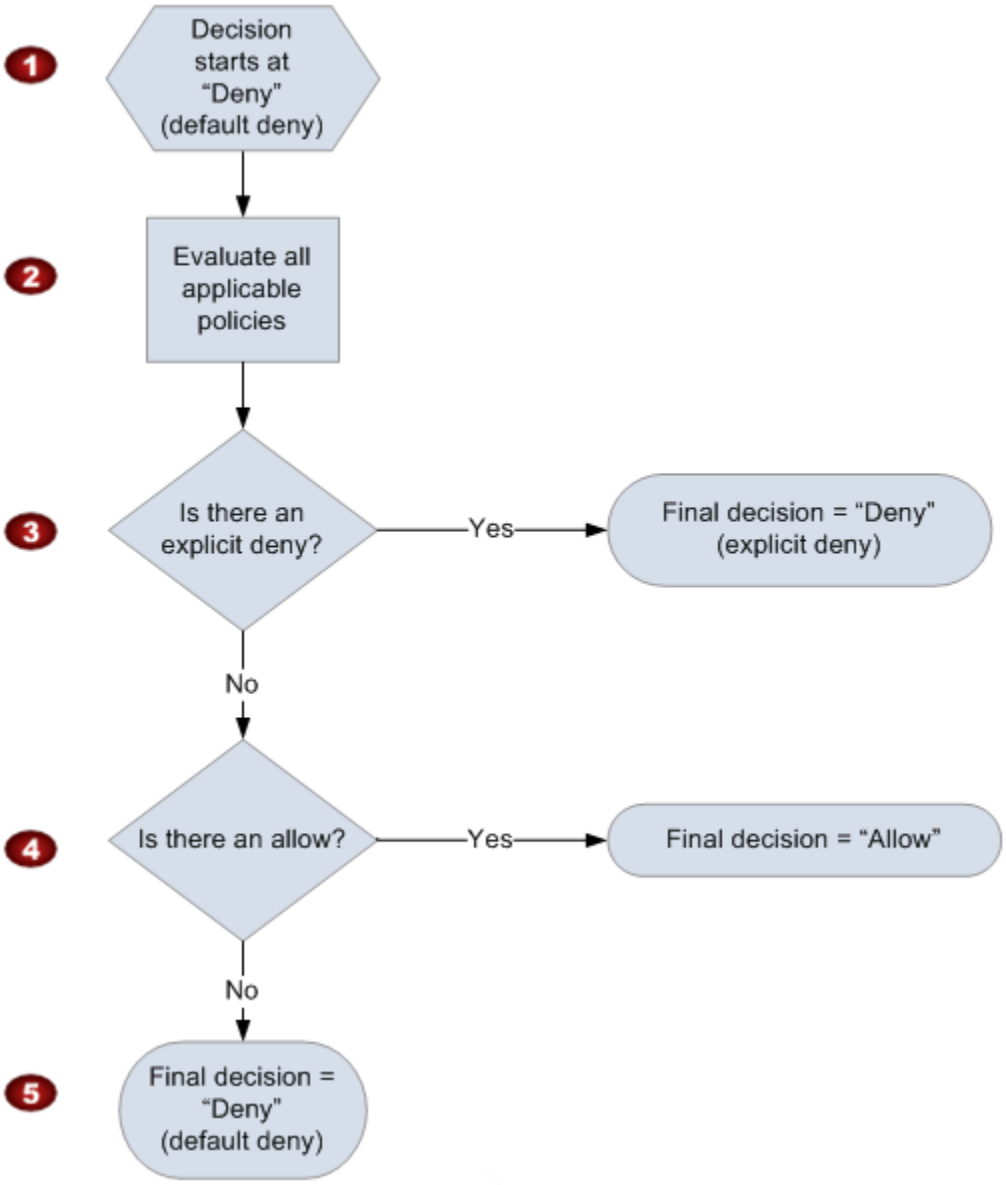
	Par exemple, Amazon SNS examine toutes les politiques Amazon SNS disponibles et détermine celles qui sont applicables (en fonction de la ressource, du demandeur, etc.).
5	Le service AWS évalue les politiques. Par exemple, Amazon SNS évalue les politiques et détermine si le demandeur est autorisé à utiliser votre rubrique ou non. Pour plus d'informations sur la logique de décision, consultez la section Logique d'évaluation.
6	Le service AWS refuse la demande ou continue de la traiter. Par exemple, selon le résultat de l'évaluation de la politique, le service renvoie une erreur « Accès refusé » pour le demandeur ou continue à traiter la demande.

Logique d'évaluation

L'objectif au moment de l'évaluation consiste à déterminer si une demande donnée doit être autorisée ou refusée. La logique d'évaluation suit plusieurs règles de base :

- Par défaut, toutes les demandes d'utilisation de votre ressource ne provenant pas de vous sont refusées.
- Une autorisation prévaut sur les refus par défaut.
- Un refus explicite remplace toute autorisation
- L'ordre dans lequel les politiques sont évaluées n'est pas important.

Le graphique et la discussion ci-dessous décrivent plus en détail comment la décision est prise.



1	La décision commence par un refus par défaut.
2	Le code d'application évalue ensuite toutes les politiques qui s'appliquent à la demande (en se basant sur la ressource, le principal, l'action et les conditions). L'ordre dans lequel le code d'application évalue les politiques n'a pas d'importance.
3	Dans toutes ces politiques, le code d'application recherche une instruction de refus explicite susceptible de s'appliquer à la demande.

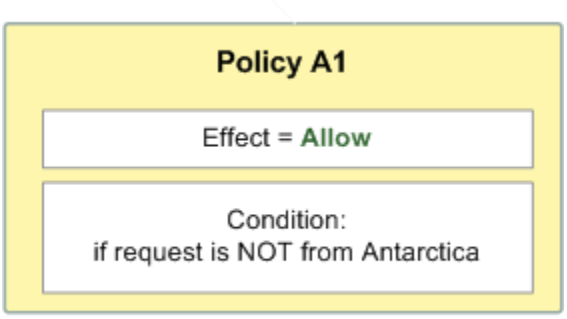
	S'il en détecte une, le code de l'application retourne une décision de « refus » et le processus se termine. Il s'agit d'un refus explicite. Pour plus d'informations, veuillez consulter la section . Refus explicite).
4	En l'absence de refus explicite, le code d'application recherche des instructions d'« autorisation » pouvant s'appliquer à la demande. S'il en trouve une, il retourne une décision d'« autorisation » et le service continue à traiter la demande.
5	Si aucune autorisation n'est détectée, la décision finale est un « refus ». En l'absence d'autorisation ou de refus explicite, nous parlons d'un refus par défaut (pour plus d'informations, consultez la page Refus par défaut).

Interaction entre les refus explicites et les refus par défaut

Une politique génère un refus par défaut si celle-ci ne s'applique pas directement à la demande. Par exemple, si un utilisateur demande à utiliser Amazon SNS, mais que la politique sur la rubrique ne fait pas référence au compte Compte AWS de l'utilisateur, cette politique a pour résultat un refus par défaut.

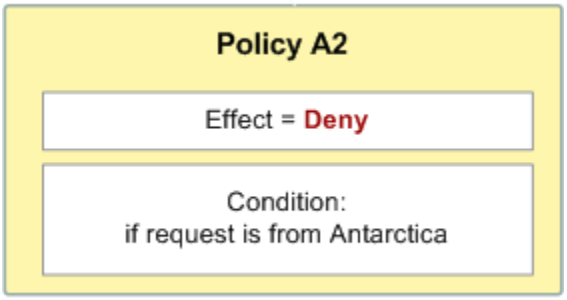
Une politique génère également un refus par défaut si une condition de l'instruction n'est pas respectée. Si toutes les conditions de l'instruction sont respectées, la politique génère une autorisation ou un refus explicite, selon la valeur de l'élément Effect qu'elle indique. Les politiques ne spécifient pas comment procéder si une condition n'est pas respectée. Le résultat est donc un refus dans ce cas.

Par exemple, supposons que vous souhaitiez refuser les demandes provenant de l'Antarctique. Vous créez une politique appelée Policy A1 qui autorise une demande uniquement si elle ne provient pas de l'Antarctique. Le schéma suivant illustre la politique.



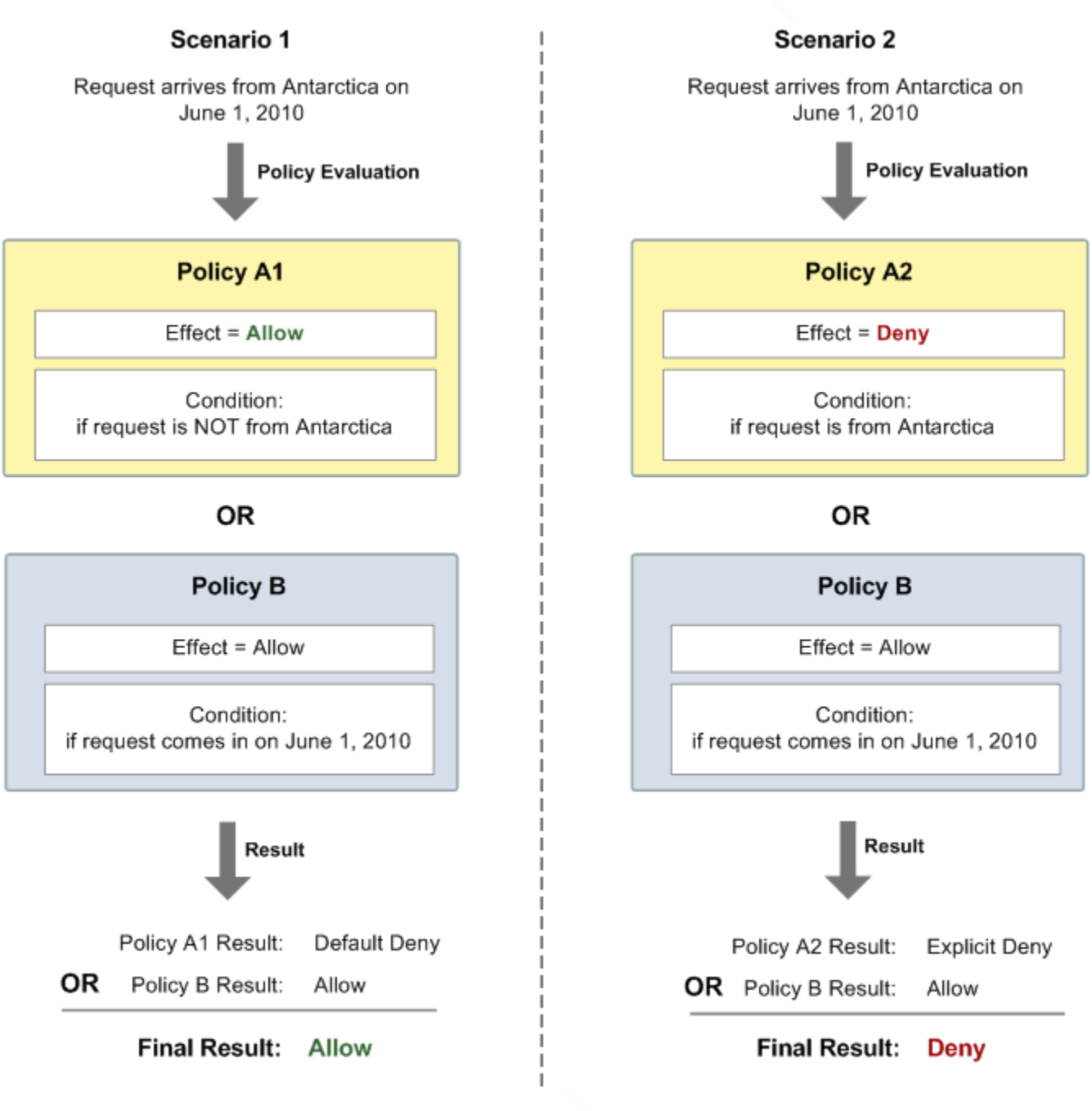
Si une personne envoie une demande depuis les États-Unis, la condition est respectée (la demande ne provient pas de l'Antarctique). Par conséquent, la demande est autorisée. En revanche, si un utilisateur effectue une demande depuis l'Antarctique, la condition n'est pas respectée et la politique génère un refus par défaut.

Vous pouvez convertir ce résultat par un refus explicite. Pour ce faire, réécrivez la politique (nommée Policy A2) comme dans le schéma suivant. Ici, la politique refuse explicitement une demande si elle vient de l'Antarctique.



Si un utilisateur effectue une demande depuis l'Antarctique, la condition est respectée et la politique génère alors un refus explicite.

La distinction entre un refus par défaut et un refus explicite est importante, car un refus par défaut peut être remplacé par une autorisation, ce qui n'est pas le cas d'un refus explicite. Par exemple, prenons le cas d'une autre politique qui autorise les demandes si elles sont reçues le 1er juin 2010. Comment cette politique modifie-t-elle le résultat global lorsqu'elle est associée à la politique qui restreint l'accès depuis l'Antarctique ? Nous allons comparer le résultat global en cas d'association de la politique basée sur la date (que nous appellerons Policy B) avec les politiques précédentes (A1 et A2). Le scénario 1 associe les politiques Policy A1 et Policy B, tandis que le scénario 2 combine les politiques Policy A2 et Policy B. L'illustration et la discussion suivantes montrent les résultats obtenus lorsqu'une demande provient de l'Antarctique le 1er juin 2010.



Dans le scénario 1, la politique Policy A1 génère un refus par défaut, comme décrit plus haut dans cette section. La politique Policy B renvoie une autorisation, car par définition, elle autorise les demandes le 1er juin 2010. L'autorisation de Policy B remplace le refus par défaut de Policy A1 et, par conséquent, la demande est autorisée.

Dans le scénario 2, la politique Policy A2 génère un refus explicite, comme décrit plus haut dans cette section. Ainsi, Policy B génère également une autorisation. Le refus explicite de Policy A2 remplace l'autorisation de Policy B et, par conséquent, la demande est refusée.

Cas d'exemple pour le contrôle d'accès Amazon SNS

Cette section décrit quelques exemples de cas d'utilisation standard du contrôle d'accès.

Rubriques

Autoriser l'accès Compte AWS à une rubrique

Supposons que vous disposiez d'une rubrique dans le système Amazon SNS. Dans le cas le plus simple, vous voulez autoriser un ou plusieurs comptes Comptes AWS à accéder à une action de rubrique spécifique (par exemple, Publish).

Pour ce faire, utilisez l'action d'API Amazon SNS `AddPermission`. Elle prend une rubrique, une liste d'ID de Compte AWS, une liste des actions et une étiquette, et crée automatiquement une nouvelle instruction dans la politique de contrôle d'accès de la rubrique. Dans ce cas, vous n'écrivez pas de politique vous-même, car Amazon SNS génère automatiquement la nouvelle instruction de politique. Vous pouvez supprimer l'instruction de politique ultérieurement en appelant `RemovePermission` avec son étiquette.

Par exemple, si vous avez appelé le `AddPermission` sujet arn:aws:sns:us-east-2:444455556666 :MyTopic, avec l'ID 1111-2222-3333, l'action et l'étiquette, Amazon SNS générera `Publish` et insérera la déclaration de Compte AWS politique de contrôle d'accès suivante : `grant-1234-publish`

```
{
```

```
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Une fois que cette instruction est ajoutée, l'utilisateur avec Compte AWS 1111-2222-3333 peut publier des messages dans la rubrique.

Limiter les abonnements à HTTPS

Dans l'exemple suivant, vous limitez le protocole de transmission des notifications à HTTPS.

Vous devez savoir comment écrire votre propre politique pour la rubrique, car l'action Amazon SNS `AddPermission` ne vous permet pas de spécifier une restriction de protocole lorsque vous autorisez un utilisateur à accéder à votre rubrique. Dans ce cas, vous souhaitez écrire votre propre politique, puis utiliser l'action `SetTopicAttributes` pour définir l'attribut `Policy` de la rubrique sur votre nouvelle politique.

L'exemple suivant de politique complète donne à l'ID de compte Compte AWS 1111-2222-3333 la possibilité de s'abonner à des notifications à partir d'une rubrique.

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

```
}
```

Publier des messages dans une file d'attente Amazon SQS

Dans ce cas d'utilisation, vous souhaitez publier des messages provenant de votre rubrique dans la file d'attente Amazon SQS. A l'instar d'Amazon SNS, Amazon SQS utilise le langage de politique de contrôle d'accès d'Amazon. Pour autoriser Amazon SNS à envoyer des messages, vous devez utiliser l'action Amazon SQS `SetQueueAttributes` pour définir une politique sur la file d'attente.

Là encore, vous aurez besoin de savoir comment écrire votre propre politique, car l'action Amazon SQS `AddPermission` ne crée pas d'instructions de politique avec des conditions.

> **Note**
>
> L'exemple présenté ci-dessous est une politique Amazon SQS (contrôlant l'accès à votre file d'attente), et non une politique Amazon SNS (contrôlant l'accès à votre rubrique). Les actions sont des actions Amazon SQS et la ressource est l'Amazon Resource Name (ARN) de la file d'attente. Vous pouvez déterminer l'ARN de la file d'attente en récupérant l'attribut `QueueArn` de la file d'attente avec l'action `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"
      }
    }
  }]
}
```

Cette politique utilise la condition `aws:SourceArn` pour limiter l'accès à la file d'attente en fonction de la source du message envoyé à la file d'attente. Vous pouvez utiliser ce type de politique pour

permettre à Amazon SNS d'envoyer des messages à votre file d'attente uniquement si les messages proviennent de l'une de vos propres rubriques. Dans ce cas, vous spécifiez un de vos sujets en particulier, dont l'ARN est arn:aws:sns:us-east- 2:444455556666 :. MyTopic

La politique précédente constitue un exemple de politique Amazon SQS que vous pouvez écrire et ajouter à une file d'attente spécifique. Il accorderait l'accès à Amazon SNS et à d'autres services AWS. Amazon SNS accorde une politique par défaut à toutes les rubriques nouvellement créées. La politique par défaut accorde l'accès à votre rubrique à tous les autres services AWS. Cette politique par défaut utilise une condition `aws:SourceArn` pour s'assurer que les services AWS accèdent à votre rubrique uniquement pour le compte de ressources AWS que vous possédez.

Autoriser les notifications d'évènements Amazon S3 à publier dans une rubrique

Dans ce cas, vous souhaitez configurer une politique de rubrique afin que le compartiment Amazon S3 d'un autre compte Compte AWS puisse effectuer une publication dans votre rubrique. Pour plus d'informations sur la publication de notifications à partir d'Amazon S3, accédez à la page Configuration de la notification des évènements de compartiment.

Cet exemple suppose que vous écriviez votre propre politique et utilisiez ensuite l'action `SetTopicAttributes` pour définir l'attribut `Policy` de la rubrique sur votre nouvelle politique.

L'exemple de déclaration suivant utilise la condition `SourceAccount` pour s'assurer que seul le compte propriétaire Amazon S3 peut accéder à la rubrique. Dans cet exemple, le propriétaire de la rubrique est 111122223333 et le propriétaire Amazon S3 est 444455556666. L'exemple indique que tout compartiment Amazon S3 appartenant à 444455556666 est autorisé à publier sur. MyTopic

```
{
  "Statement": [{
    "Effect": "Allow",
     "Principal": {
      "Service": "s3.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      }
    }
  }]
}
```

Lors de la publication d'évènements sur Amazon SNS, les services suivants prennent en charge `aws:SourceAccount` :

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon ElastiCache
- Amazon GameLift
- API SMS et Voix Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

Autoriser Amazon SES à publier sur une rubrique appartenant à un autre compte

Vous pouvez autoriser un autre service AWS à publier sur une rubrique appartenant à un autre Compte AWS. Supposons que vous vous êtes connecté au compte 111122223333, que vous avez ouvert Amazon SES et que vous avez créé un e-mail. Pour publier des notifications concernant cet e-mail dans une rubrique Amazon SNS appartenant au compte 444455556666, vous devez créer une politique comme la suivante. Pour ce faire, vous devez fournir des informations sur le principal (l'autre service) et sur la propriété de chaque ressource. L'instruction `Resource` fournit l'ARN de rubrique, qui inclut l'ID de compte du propriétaire de la rubrique, 444455556666. L'instruction `"aws:SourceOwner": "111122223333"` spécifie que votre compte est propriétaire de l'e-mail.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
```

```
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
  ]
}
```

Lors de la publication d'évènements sur Amazon SNS, les services suivants prennent en charge `aws:SourceOwner` :

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon ElastiCache
- Amazon GameLift
- API SMS et Voix Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` par rapport à `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner` est obsolète et les nouveaux services peuvent s'intégrer à Amazon SNS uniquement via `aws:SourceArn` et `aws:SourceAccount`. Amazon SNS assure toujours la rétrocompatibilité des services existants qui prennent actuellement en charge `aws:SourceOwner`.

Les clés de condition `aws:SourceAccount` et `aws:SourceOwner` sont chacune définies par certains Services AWS lorsqu'ils publient dans une rubrique Amazon SNS. Lorsqu'elle est prise en charge, la valeur sera la valeur à 12 chiffres de l'ID de compte AWS au nom duquel le service publie des données. Certains services prennent en charge l'un et d'autres prennent en charge l'autre.

- Voir Autoriser les notifications d'évènements Amazon S3 à publier dans une rubrique pour savoir comment les notifications Amazon S3 utilisent `aws:SourceAccount` et une liste de services AWS qui prennent en charge cette condition.
- Voir Autoriser Amazon SES à publier sur une rubrique appartenant à un autre compte pour savoir comment Amazon SES utilisent `aws:SourceOwner` et une liste de services AWS qui prennent en charge cette condition.

Autoriser les comptes d'une organisation dans AWS Organizations à publier dans une rubrique d'un autre compte

Le service AWS Organizations vous aide à gérer la facturation, à contrôler l'accès et la sécurité et à partager les ressources de manière centralisée dans l'ensemble de vos comptes Comptes AWS.

Vous pouvez trouver votre ID d'organisation dans la console Organisations. Pour plus d'informations, consultez Affichage des détails d'une organisation à partir du compte de gestion.

Dans cet exemple, n'importe quel compte Compte AWS de l'organisation `myOrgId` peut publier dans la rubrique Amazon SNS `MyTopic` dans le compte `444455556666`. La politique vérifie la valeur de l'ID de l'organisation à l'aide de la clé de condition globale `aws:PrincipalOrgID`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

Autoriser la publication de n'importe quelle CloudWatch alarme dans un sujet d'un autre compte

Dans ce cas, toutes les CloudWatch alarmes associées au compte `111122223333` peuvent être publiées sur une rubrique Amazon SNS associée au compte. `444455556666`

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

Limiter la publication à une rubrique Amazon SNS uniquement à partir d'un point de terminaison de VPC spécifique

Dans ce cas, la rubrique dans le compte 444455556666 est autorisée à publier uniquement à partir du point de terminaison de VPC avec l'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"
      }
    }
  }]
}
```

Fonctionnement d'Amazon Simple Notification Service avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon SNS, découvrez les fonctions IAM qui peuvent être utilisées avec Amazon SNS.

Fonctions IAM que vous pouvez utiliser avec Amazon Simple Notification Service

Fonctionnalité IAM	Prise en charge d'Amazon SNS
Politiques basées sur l’identité	Oui
Politiques basées sur les ressources	Oui
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Oui
ACL	Non
ABAC (identifications dans les politiques)	Partielle
Informations d’identification temporaires	Oui
Autorisations de principal	Oui

Fonctionnalité IAM	Prise en charge d'Amazon SNS
Fonctions du service	Oui
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont Amazon SNS et d'autres services AWS fonctionnent avec IAM, consultez Services AWS qui fonctionnent avec IAM dans le Guide de l'utilisateur IAM.

Actions de politique pour Amazon SNS

Prend en charge les actions de politique	Oui

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de politique possèdent généralement le même nom que l'opération d'API AWS associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour afficher la liste des actions Amazon SNS, consultez Ressources définies par Amazon SNS dans Référence de l'autorisation de service.

Les actions de politique dans Amazon SNS utilisent le préfixe suivant avant l'action :

```
sns
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Pour consulter des exemples de politiques Amazon SNS basées sur l'identité, consultez Exemples de politiques basées sur une identité pour Amazon Simple Notification Service (Exemples de politiques basées sur l'identité pour Amazon Simple Notification Service).

Ressources de politique pour Amazon SNS

Prend en charge les ressources de politique	Oui

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets pour lesquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour afficher la liste des types de ressource Amazon SNS et leurs ARN, consultez Actions définies par Amazon SNS dans Référence de l'autorisation de service. Pour connaître les actions avec lesquelles vous pouvez spécifier l'ARN de chaque ressource, consultez Ressources définies par Amazon SNS.

Pour consulter des exemples de politiques Amazon SNS basées sur l'identité, consultez Exemples de politiques basées sur une identité pour Amazon Simple Notification Service (Exemples de politiques basées sur l'identité pour Amazon Simple Notification Service).

Clés de condition de politique pour Amazon SNS

Prise en charge des clés de condition de stratégie spécifiques au service	Oui

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une opération `OR` logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments des politiques IAM : variables et balises dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques à un service. Pour afficher toutes les clés de condition globales AWS, consultez Clés de contexte de condition globale AWS dans le Guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition Amazon SNS, consultez Clés de condition pour Amazon SNS dans la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez Ressources définies par Amazon SNS.

Pour consulter des exemples de politiques Amazon SNS basées sur l'identité, consultez Exemples de politiques basées sur une identité pour Amazon Simple Notification Service (Exemples de politiques basées sur l'identité pour Amazon Simple Notification Service).

Listes ACL dans Amazon SNS

Prend en charge les listes ACL	Non

Les listes de contrôle d'accès (ACL) vérifient quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec Amazon SNS

Prend en charge ABAC (identifications dans les politiques)	Partielle

Le contrôle d'accès basé sur les attributs (ABAC) est une politique d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés étiquettes. Vous pouvez attacher des étiquettes à des entités IAM (utilisateurs ou rôles), ainsi qu'à de nombreuses ressources AWS. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des balises, vous devez fournir les informations de balise dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur l'ABAC, consultez Qu'est-ce que le contrôle d'accès basé sur les attributs (ABAC) ? dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec Amazon SNS

Prend en charge les informations d'identification temporaires	Oui

Certains Services AWS ne fonctionnent pas quand vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, notamment sur les Services AWS qui fonctionnent avec des informations d'identification temporaires, consultez Services AWS qui fonctionnent avec IAM dans le Guide de l'utilisateur IAM.

Vous utilisez des informations d'identification temporaires quand vous vous connectez à la AWS Management Console en utilisant toute méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS en utilisant le lien d'authentification unique (SSO) de votre société, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d'identification temporaires lorsque vous vous connectez à la console en tant qu'utilisateur, puis changez de rôle. Pour plus d'informations sur le changement de rôle, consultez Changement de rôle (console) dans le Guide de l'utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide d'AWS CLI ou de l'API AWS. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour accéder à AWS. AWS recommande de générer des informations d'identification temporaires de façon dynamique au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez Informations d'identification de sécurité temporaires dans IAM.

Autorisations de principal entre services pour Amazon SNS

Prend en charge les sessions d'accès direct (FAS)	Oui

Lorsque vous vous servez d'un utilisateur IAM ou d'un rôle IAM pour accomplir des actions dans AWS, vous êtes considéré comme un principal. Lorsque vous utilisez certains services, l'action

que vous effectuez est susceptible de lancer une autre action dans un autre service. FAS utilise les autorisations du principal appelant Service AWS, combinées à la demande Service AWS pour effectuer des demandes aux services en aval. Les demandes FAS ne sont formulées que lorsqu'un service reçoit une demande qui, pour aboutir, a besoin d'interagir avec d'autres ressources ou Services AWS. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.

Fonctions du service pour Amazon SNS

Prend en charge les fonctions du service	Oui

Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations d'une fonction du service peut altérer la fonctionnalité d'Amazon SNS. Ne modifiez des rôles de service que quand Amazon SNS vous le conseille.

Rôles liés à un service pour Amazon SNS

Prend en charge les rôles liés à un service.	Non

Un rôle lié à un service est un type de rôle de service lié à un Service AWS. Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre Compte AWS et sont détenus par le service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur une identité pour Amazon Simple Notification Service

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier les ressources Amazon SNS. Ils ne peuvent pas non plus exécuter des tâches à l'aide de la AWS Management Console, de l'AWS Command Line Interface (AWS CLI) ou de l'API AWS. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM doit créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent endosser les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Amazon SNS, y compris le format des ARN pour chacun des types de ressources, consultez Actions, ressources et clés de condition pour Amazon SNS dans la Référence de l'autorisation de service.

Rubriques

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources Amazon SNS dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Démarrer avec AWS gérées et évoluez vers les autorisations de moindre privilège - Pour commencer à accorder des autorisations à vos utilisateurs et charges de travail, utilisez les politiques gérées AWS qui accordent des autorisations dans de nombreux cas d'utilisation courants. Elles sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client AWS qui sont spécifiques

à vos cas d'utilisation. Pour de plus amples informations, consultez AWS Politiques gérées ou AWS Politiques gérées pour les activités professionnelles dans le Guide de l'utilisateur IAM.

- Accorder les autorisations de moindre privilège - Lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez Politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utiliser des conditions dans les politiques IAM pour restreindre davantage l'accès - Vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées via un Service AWS spécifique, comme AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles - IAM Access Analyzer valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Authentification multifactorielle (MFA) nécessaire : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root dans votre Compte AWS, activez l'authentification multifactorielle pour une sécurité renforcée. Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Configuration de l'accès aux API protégé par MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon SNS

Pour accéder à la console Amazon Simple Notification Service, vous devez disposer d'un ensemble minimum d'autorisations. Ces autorisations doivent vous permettre de répertorier et consulter des informations sur les ressources Amazon SNS dans votre Compte AWS. Si vous créez une stratégie

basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette stratégie.

Vous n'avez pas besoin d'accorder les autorisations minimales de console pour les utilisateurs qui effectuent des appels uniquement à AWS CLI ou à l'API AWS. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour vous assurer que les utilisateurs et les rôles peuvent continuer à utiliser la console Amazon SNS, attachez également la politique Amazon SNS *ConsoleAccess* ou la politique gérée par *ReadOnly* AWS aux entités. Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courantes. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonction avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations qui en résultent représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) - les SCP sont des politiques JSON qui spécifient le nombre maximal d'autorisations pour une organisation ou une unité d'organisation (OU) dans AWS Organizations. AWS Organizations est un service qui vous permet de regrouper et de gérer de façon centralisée plusieurs Comptes AWS détenus par votre entreprise. Si vous activez toutes les fonctions d'une organisation, vous pouvez appliquer les politiques de contrôle de service (SCP) à l'un ou à l'ensemble de vos comptes. Les politiques de contrôle des services (SCP) limitent les autorisations pour les entités dans les comptes membres, y compris chaque utilisateur racine de compte Compte AWS. Pour plus d'informations sur les organisations et les SCP, consultez Fonctionnement des SCP dans le Guide de l'utilisateur AWS Organizations.
- politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou

un utilisateur fédéré. Les autorisations de la séance obtenue sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations, consultez Politiques de séance dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations obtenues sont plus compliquées à comprendre. Pour découvrir la façon dont AWS détermine s'il convient d'autoriser une demande en présence de plusieurs types de politiques, veuillez consulter Logique d'évaluation de politiques dans le Guide de l'utilisateur IAM.

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations nécessaires pour réaliser cette action sur la console ou par programmation à l'aide de l'AWS CLI ou de l'API 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": "*"
        }
    ]
}
```

Politiques basées sur l'identité pour Amazon SNS

Prend en charge les politiques basées sur une identité	Oui

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un Groupes d’utilisateurs IAM ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l’utilisateur IAM.

Avec les politiques IAM basées sur l’identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité car celle-ci s’applique à l’utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l’utilisateur IAM.

Exemples de politiques basées sur l'identité pour Amazon SNS

Pour consulter des exemples de politiques Amazon SNS basées sur l'identité, consultez Exemples de politiques basées sur une identité pour Amazon Simple Notification Service (Exemples de politiques basées sur l'identité pour Amazon Simple Notification Service).

Politiques basées sur les ressources au sein d'Amazon SNS

Prend en charge les politiques basées sur une ressource	Oui

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou des Services AWS.

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal entre comptes à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Quand le principal et la ressource se trouvent dans des Comptes AWS différents, un administrateur IAM dans le compte approuvé doit également accorder à l'entité principal (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez Différence entre les rôles IAM et les politiques basées sur une ressource dans le Guide de l'utilisateur IAM.

Utilisation de politiques basées sur l'identité avec Amazon SNS

Rubriques

- politiques IAM et Amazon SNS ensemble
- Format ARN des ressources Amazon SNS
- Actions d'API Amazon SNS
- Clés de politique Amazon SNS
- Exemples de politiques pour Amazon SNS

Amazon Simple Notification Service s'intègre à AWS Identity and Access Management (IAM) afin que vous puissiez spécifier les actions Amazon SNS qu'un utilisateur dans votre Compte AWS peut effectuer avec les ressources Amazon SNS. Vous pouvez indiquer une rubrique particulière dans la politique. Par exemple, vous pouvez utiliser des variables lorsque vous créez une politique IAM qui autorise certains utilisateurs de votre organisation à utiliser l'action `Publish` avec des rubriques spécifiques dans votre Compte AWS. Pour plus d'informations, consultez la section Variables de politique dans le guide Utilisation d'IAM.

> **Important**
>
> L'utilisation d'Amazon SNS avec IAM ne modifie pas l'utilisation d'Amazon SNS. Aucun changement n'est apporté aux actions Amazon SNS et aucune nouvelle action Amazon SNS associée aux utilisateurs et au contrôle d'accès n'est ajoutée.

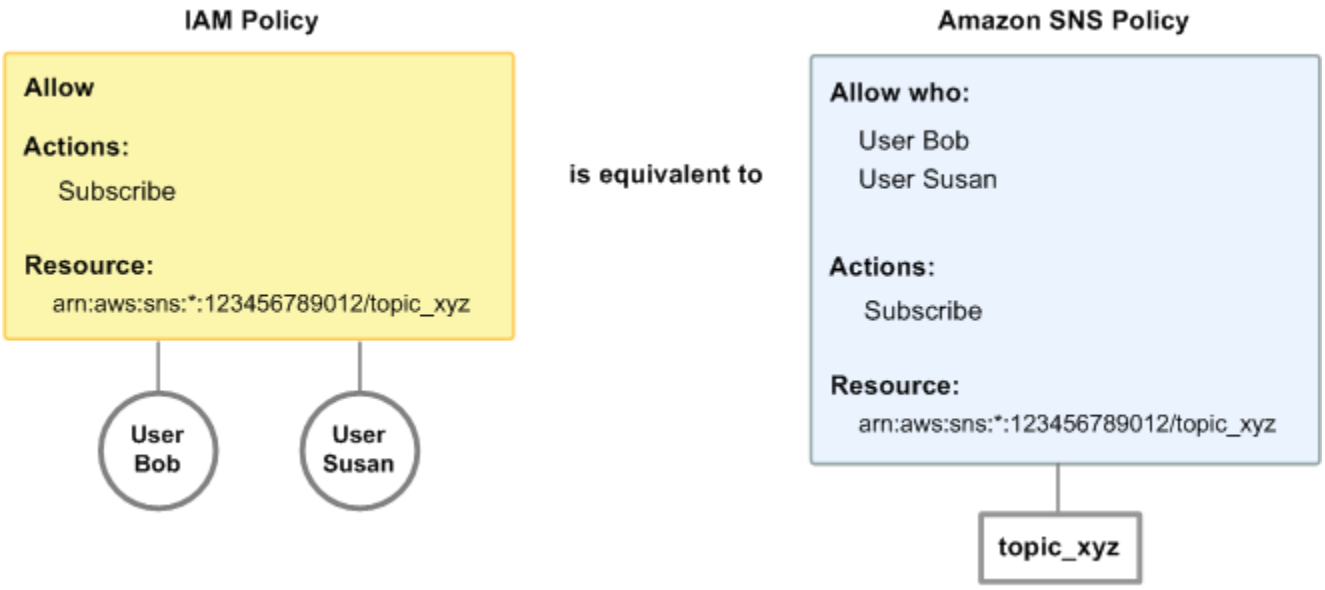
Pour obtenir des exemples de politiques couvrant les actions et ressources Amazon SNS, consultez Exemples de politiques pour Amazon SNS.

politiques IAM et Amazon SNS ensemble

Vous utilisez une politique IAM pour limiter l'accès de vos utilisateurs aux actions et rubriques Amazon SNS. Une politique IAM peut limiter l'accès uniquement aux utilisateurs de votre compte AWS, pas à d'autres comptes Comptes AWS.

Vous utilisez une politique Amazon SNS avec une rubrique particulière pour limiter les personnes autorisées à travailler avec cette rubrique (par exemple, qui peut y publier des messages, qui peut s'y abonner, etc.). Les politiques Amazon SNS peuvent accorder l'accès à d'autres Comptes AWS, ou à des utilisateurs au sein de votre propre Compte AWS.

Pour accorder à vos utilisateurs des autorisations d'accès à vos rubriques Amazon SNS, vous pouvez utiliser des politiques IAM, des politiques Amazon SNS, ou les deux. Dans l'ensemble, vous obtenez les mêmes résultats avec l'un ou l'autre. Par exemple, le schéma suivant illustre une politique IAM et une politique Amazon SNS équivalentes. La politique IAM autorise l'action Amazon SNS `Subscribe` pour la rubrique nommée topic_xyz dans votre compte AWS. La politique IAM est attachée aux utilisateurs Bob et Susan (ce qui signifie que Bob et Susan disposent des autorisations définies dans la politique). La politique Amazon SNS donne également à Bob et Susan l'autorisation d'accéder à `Subscribe` pour la rubrique topic_xyz.



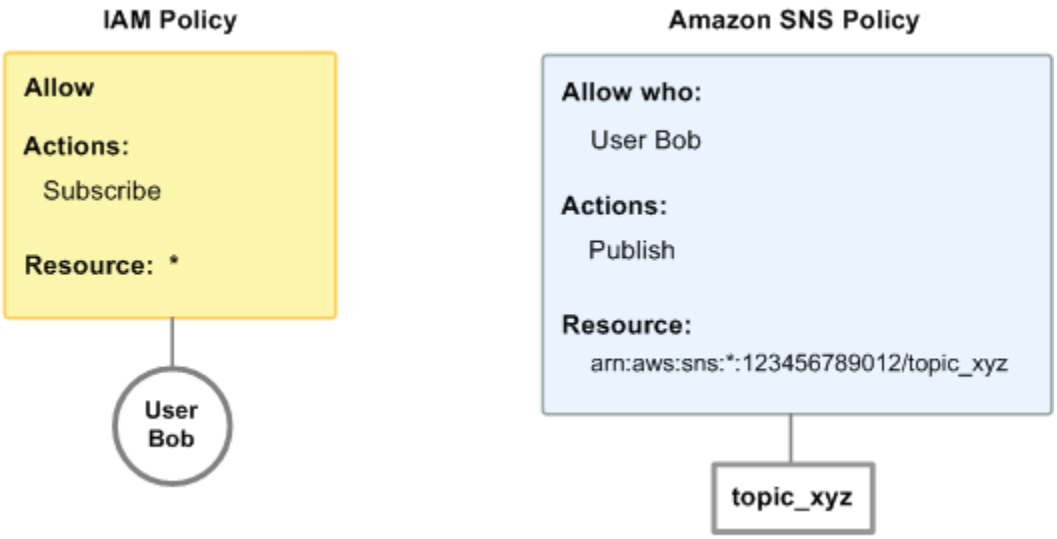
> **Note**
>
> L'exemple précédent montre des politiques simples, sans conditions. Vous pouvez spécifier une condition particulière dans l'une ou l'autre des politiques et aboutir au même résultat.

Il existe cependant une différence entre les politiques IAM et Amazon SNS d'AWS : le système de politique Amazon SNS vous permet d'accorder une autorisation à d'autres comptes Comptes AWS, contrairement à la politique IAM.

Il vous incombe de décider si vous voulez utiliser les deux systèmes conjointement pour gérer vos autorisations, en fonction de vos besoins. Les exemples suivants illustrent la façon dont les deux systèmes de politique interagissent.

Example 1

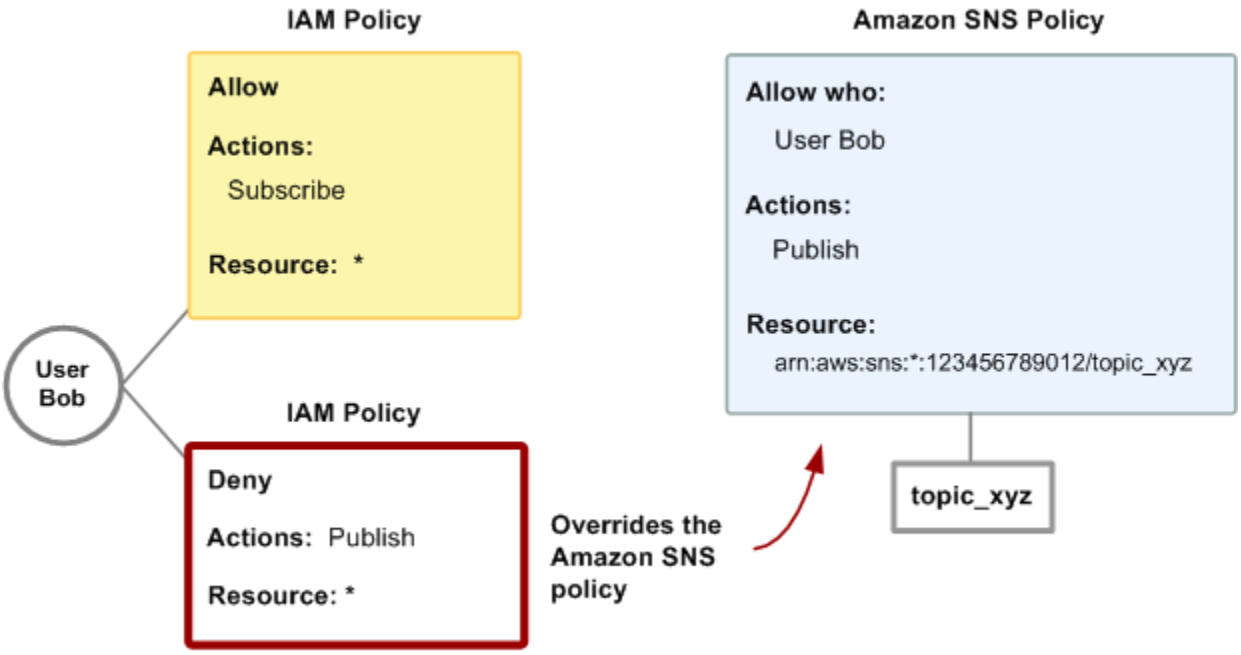
Dans cet exemple, une politique IAM et une politique Amazon SNS s'appliquent toutes deux à Bob. La politique IAM lui accorde une autorisation pour `Subscribe` sur n'importe quelle rubrique du compte Compte AWS, tandis que la politique Amazon SNS l'autorise à utiliser `Publish` sur une rubrique spécifique (topic_xyz). Le diagramme suivant illustre le concept.



Si Bob était sur le point d'envoyer une demande d'abonnement à une rubrique du compte AWS, la politique IAM autoriserait l'action. Si Bob était sur le point d'envoyer une demande de publication d'un message dans la rubrique topic_xyz, la politique Amazon SNS autoriserait l'action.

Example 2

Dans ce cas, nous nous appuyons sur l'exemple 1 (où deux politiques s'appliquent à Bob). Disons que Bob publie des messages dans la rubrique topic_xyz, ce qu'il ne devrait pas être autorisé à faire. Vous voulez complètement supprimer cette possibilité de publication dans des rubriques. La meilleure chose à faire consiste à ajouter une politique IAM qui lui refuse l'accès à l'action `Publish` sur toutes les rubriques. Cette troisième politique remplace la politique Amazon SNS qui lui donnait initialement l'autorisation d'effectuer une publication dans la rubrique topic_xyz, car un refus explicite remplace toujours une autorisation (pour plus d'informations sur la logique d'évaluation de politique, consultez Logique d'évaluation). Le diagramme suivant illustre le concept.



Pour obtenir des exemples de politiques couvrant les actions et ressources Amazon SNS, consultez Exemples de politiques pour Amazon SNS. Pour plus d'informations sur l'écriture de politiques Amazon SNS, consultez la documentation technique pour Amazon SNS.

Format ARN des ressources Amazon SNS

Pour Amazon SNS, les rubriques sont le seul type de ressource que vous pouvez spécifier dans une politique. Vous trouverez ci-dessous le format Amazon Resource Name (ARN) pour les rubriques.

```
arn:aws:sns:region:account_ID:topic_name
```

Pour plus d'informations sur les ARN, consultez ARN dans le Guide de l'utilisateur IAM.

Example

Ce qui suit est un ARN pour une rubrique nommée MyTopic dans la région us-east-2, appartenant à 123456789012. Compte AWS

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

Example

Si un sujet était nommé MyTopic dans chacune des différentes régions prises en charge par Amazon SNS, vous pourriez le spécifier avec l'ARN suivant.

```
arn:aws:sns:*:123456789012:MyTopic
```

Vous pouvez utiliser les caractères génériques * et ? dans le nom de la rubrique. Par exemple, la ligne suivante peut faire référence à toutes les rubriques créées par Bob, auxquelles il a ajouté le préfixe bob_.

```
arn:aws:sns:*:123456789012:bob_*
```

Pour vous faciliter les choses, lorsque vous créez une rubrique, Amazon SNS renvoie l'ARN de la rubrique dans la réponse.

Actions d'API Amazon SNS

Dans une politique IAM, vous pouvez spécifier les actions qu'Amazon SNS propose. Toutefois, les actions `ConfirmSubscription` et `Unsubscribe` ne nécessitent pas d'authentification, ce qui signifie que même si vous spécifiez ces actions dans une politique, IAM ne limitera pas l'accès des utilisateurs à ces actions.

Chaque action que vous spécifiez dans une politique doit être précédée de la chaîne en minuscules `sns:`. Pour spécifier toutes les actions Amazon SNS, par exemple, vous utiliseriez `sns:*`. Pour obtenir la liste des actions, consultez la Référence de l'API Amazon Simple Notification Service.

Clés de politique Amazon SNS

Amazon SNS implémente les clés de politique suivantes à l'échelle d'AWS, ainsi que des clés spécifiques aux services.

Pour obtenir la liste des clés de condition prises en charge par chaque Service AWS, consultez Actions, ressources et clés de condition pour les Services AWS dans le Guide de l'utilisateur IAM. Pour obtenir la liste des clés de condition qui peuvent être utilisées dans plusieurs Services AWS, consultez Clés de contexte de condition globales AWS dans le Guide de l'utilisateur IAM.

Amazon SNS utilise les clés spécifiques au service suivantes. Utilisez ces clés dans les politiques limitant l'accès aux demandes `Subscribe`.

- sns:endpoint – URL, adresse e-mail ou ARN provenant d'une demande `Subscribe` ou d'un abonnement confirmé au préalable. Il convient de l'utiliser avec des conditions de chaîne (consultez la section Exemples de politiques pour Amazon SNS) pour limiter l'accès à des points de terminaison spécifiques (par exemple, *@yourcompany.com).
- sns:protocol – valeur `protocol` provenant d'une demande `Subscribe` ou d'un abonnement confirmé au préalable. Il convient de l'utiliser avec des conditions de chaîne (consultez la section Exemples de politiques pour Amazon SNS) pour limiter la publication à des protocoles de diffusion spécifiques (par exemple, https).

Exemples de politiques pour Amazon SNS

Cette section illustre plusieurs exemples simples de politique pour contrôler l'accès utilisateur à Amazon SNS.

> **Note**
>
> À l'avenir, il se peut qu'Amazon SNS ajoute de nouvelles actions qui devraient être logiquement incluses dans l'une des politiques suivantes, en fonction des objectifs établis de la politique.

Example 1 : Autoriser un groupe à créer et gérer des rubriques

Dans cet exemple, nous créons une politique qui permet d'accéder à `CreateTopic`, `ListTopics`, `SetTopicAttributes` et `DeleteTopic`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2 : Autoriser le groupe informatique à publier des messages dans une rubrique particulière

Dans cet exemple, nous créons un groupe pour l'informatique et affectons une politique qui permet d'accéder à `Publish` sur le sujet qui vous intéresse spécifiquement.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3 : Donner aux utilisateurs du Compte AWS la possibilité de s'abonner à des rubriques

Dans cet exemple, nous créons une politique qui permet d'accéder à l'action `Subscribe`, avec des conditions de correspondance de chaîne pour les clés de politique `sns:Protocol` et `sns:Endpoint`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4 : Autoriser un partenaire à publier des messages dans une rubrique particulière

Vous pouvez utiliser une politique Amazon SNS ou une politique IAM pour autoriser un partenaire à effectuer une publication dans une rubrique spécifique. Si votre partenaire possède un Compte AWS, il sera peut-être plus facile d'utiliser une politique Amazon SNS. Cependant, toute personne dans l'entreprise du partenaire qui dispose des informations d'identification de sécurité AWS peut publier des messages dans la rubrique. Cet exemple part du principe que vous voulez limiter l'accès à une personne (ou application) spécifique. Pour ce faire, vous devez traiter le partenaire comme un utilisateur au sein de votre propre entreprise et utiliser une politique IAM au lieu d'une politique Amazon SNS.

Dans cet exemple, nous créons un groupe appelé WidgetCo qui représente l'entreprise partenaire ; nous créons un utilisateur pour la personne (ou l'application) spécifique de l'entreprise partenaire qui a besoin d'un accès ; puis nous plaçons l'utilisateur dans le groupe.

Nous joignons ensuite une politique qui accorde au groupe l'`Publish`accès au sujet spécifique nommé WidgetPartnerTopic.

Nous voulons également empêcher le WidgetCo groupe de faire quoi que ce soit d'autre avec les sujets. Nous avons donc ajouté une déclaration refusant l'autorisation d'effectuer des actions Amazon SNS autres que celles portant `Publish` sur des sujets autres que. WidgetPartnerTopic Cette opération est nécessaire uniquement s'il existe une politique large ailleurs dans le système, qui donne aux utilisateurs un large accès à 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"
    }
  ]
}
```

Utilisation d'informations d'identification de sécurité temporaires avec Amazon SNS

Outre la création d'utilisateurs IAM avec leurs propres informations d'identification de sécurité, IAM vous permet également d'accorder des informations d'identification de sécurité temporaires à n'importe quel utilisateur, en l'autorisant à accéder à vos services et ressources AWS. Vous pouvez gérer les utilisateurs qui possèdent des Comptes AWS : il s'agit des utilisateurs IAM. Vous pouvez également gérer les utilisateurs de votre système qui ne disposent pas de comptes Comptes AWS. Ces utilisateurs sont appelés utilisateurs fédérés. De plus, les « utilisateurs » peuvent également être des applications que vous créez pour accéder à vos ressources AWS.

Vous pouvez utiliser ces informations d'identification de sécurité temporaires pour effectuer des demandes auprès d'Amazon SNS. Les bibliothèques d'API calculent la valeur de signature

nécessaire en utilisant ces informations d'identification pour authentifier votre demande. Si vous envoyez des demandes en utilisant des informations d'identification expirées, Amazon SNS les rejette.

Pour plus d'informations sur la prise en charge des informations d'identification de sécurité temporaires par IAM, consultez la section Accès temporaire à vos ressources AWS dans Utilisation d'IAM.

Example Utilisation des informations d'identification de sécurité temporaires pour l'authentification d'une demande Amazon SNS

L'exemple suivant montre comment obtenir des informations d'identification de sécurité temporaires pour authentifier une demande Amazon SNS.

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Autorisations d'API Amazon SNS : référence des actions et ressources

La liste suivante fournit des informations spécifiques à l'implémentation Amazon SNS du contrôle d'accès :

- Chaque politique ne doit couvrir qu'une seule rubrique (lors de l'écriture d'une politique, n'incluez pas d'instructions qui couvrent différentes rubriques).
- Chaque politique doit avoir un ID de politique unique (). `Id`
- Chaque instruction dans une politique doit avoir un ID d'instruction unique (). `sid`

Quotas de politiques

Le tableau suivant répertorie les quotas maximaux pour une déclaration de politique.

Nom	Quota maximal
Octets	30 Ko
Instructions	100
principals	1 à 200 (0 n'est pas valide.)
Ressource	1 (0 n'est pas valide. La valeur doit correspondre à l'ARN de la rubrique de la politique).

Actions de politique Amazon SNS valides

Amazon SNS prend en charge les actions indiquées dans le tableau suivant.

Action	Description
sns : AddPermission	Accorde l'autorisation d'ajouter des autorisations à la politique de la rubrique.
sns : DeleteTopic	Accorde l'autorisation de supprimer une rubrique.
sns : GetDataProtectionPolicy	Accorde l'autorisation de récupérer la politique de protection des données d'une rubrique.
sns : GetTopicAttributes	Accorde l'autorisation de recevoir tous les attributs de la rubrique.
sns : ListSubscriptionsByTopic	Accorde l'autorisation de récupérer tous les abonnements à une rubrique spécifique.
sns : ListTagsForResource	Accorde l'autorisation de répertorier toutes les balises ajoutées à une rubrique spécifique.
sns:Publish	Accorde l'autorisation d'effectuer une publication et de publier un lot dans une rubrique ou un point de terminaison. Pour plus d'informations, consultez Publish et PublishBatchle manuel Amazon Simple Notification Service API Reference.

Action	Description
sns : PutDataProtectionPolicy	Accorde l'autorisation de définir la politique de protection des données d'une rubrique.
sns : RemovePermission	Accorde l'autorisation de supprimer des autorisations dans la politique de la rubrique.
sns : SetTopicAttributes	Accorde l'autorisation de définir les attributs d'une rubrique.
sns:Subscribe	Accorde l'autorisation de s'abonner à une rubrique.

Clés spécifiques aux services

Amazon SNS utilise les clés spécifiques au service suivantes. Vous pouvez les utiliser dans des politiques qui limitent l'accès aux demandes `Subscribe`.

- sns:endpoint – URL, adresse e-mail ou ARN provenant d'une demande `Subscribe` ou d'un abonnement confirmé au préalable. Il convient de l'utiliser avec des conditions de chaîne (consultez la section Exemples de politiques pour Amazon SNS) pour limiter l'accès à des points de terminaison spécifiques (par exemple, *@example.com).
- sns:protocol – valeur `protocol` provenant d'une demande `Subscribe` ou d'un abonnement confirmé au préalable. Il convient de l'utiliser avec des conditions de chaîne (consultez la section Exemples de politiques pour Amazon SNS) pour limiter la publication à des protocoles de diffusion spécifiques (par exemple, https).

> **Important**
>
> Lorsque vous utilisez une politique pour contrôler l'accès par sns:Endpoint, sachez que les problèmes DNS sont susceptibles d'affecter ultérieurement la résolution des noms du point de terminaison.

Résolution des problèmes d'accès et d'identité Amazon Simple Notification Service

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous utilisez Amazon SNS et IAM.

Rubriques

Je ne suis pas autorisé à effectuer une action dans Amazon SNS

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à effectuer une action, vos stratégies doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `sns:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

Dans ce cas, la stratégie de Mateo doit être mise à jour pour l'autoriser à accéder à la ressource *`my-example-widget`* à l'aide de l'action `sns:`*`GetWidget`*.

Si vous avez encore besoin d'aide, contactez votre administrateur AWS. Votre administrateur vous a fourni vos informations de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon SNS.

Certains Services AWS vous permettent de transmettre un rôle existant à ce service, au lieu de créer une nouvelle fonction du service ou rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon SNS. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez encore besoin d'aide, contactez votre administrateur AWS. Votre administrateur vous a fourni vos informations de connexion.

Je veux autoriser des personnes extérieures à mon Compte AWS à accéder à mes ressources Amazon SNS

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si Amazon SNS est compatible avec ces fonctionnalités, veuillez consulter Fonctionnement d'Amazon Simple Notification Service avec IAM (Fonctionnement d'Amazon Simple Notification Service avec IAM).
- Pour savoir comment octroyer l'accès à vos ressources à des Comptes AWS dont vous êtes propriétaire, consultez la section Fournir l'accès à un utilisateur IAM dans un autre Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment octroyer l'accès à vos ressources à des tiers Comptes AWS, consultez Fournir l'accès aux Comptes AWS appartenant à des tiers dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

Journalisation et surveillance dans Amazon SNS

Cette section fournit des informations sur la journalisation et la surveillance des rubriques Amazon SNS.

Rubriques

Journalisation des appels d'API Amazon SNS à l'aide de CloudTrail

Amazon SNS est intégré à AWS CloudTrail, un service qui enregistre les actions effectuées par un utilisateur, un rôle ou un service AWS dans Amazon SNS. CloudTrail enregistre les appels d'API pour Amazon SNS en tant qu'évènements. Les appels enregistrés incluent les appels de la console Amazon SNS et les appels de code vers les opérations d'API Amazon SNS. Si vous créez un journal de suivi, vous pouvez activer la livraison continue des évènements CloudTrail dans un compartiment Amazon S3, y compris les évènements pour Amazon SNS. Si vous ne configurez pas de journal d'activité, vous pouvez toujours afficher les événements les plus récents dans la console CloudTrail dans Event history (Historique des événements). Avec les informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été envoyée à Amazon SNS, l'adresse IP à partir de laquelle la demande a été effectuée, l'auteur et la date de la demande, ainsi que d'autres détails.

Pour en savoir plus sur CloudTrail, y compris la façon de le configurer et de l'activer, consultez le AWS CloudTrail Guide de l'utilisateur.

Informations relatives à Amazon SNS dans CloudTrail

CloudTrail est activé dans votre Compte AWS lors de la création de ce dernier. Quand une activité d'évènement prise en charge a lieu dans Amazon SNS, elle est enregistrée dans un évènement CloudTrail avec d'autres évènements de service AWS dans l'Historique des évènements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, veuillez consulter Affichage des évènements avec l'historique des évènements CloudTrail.

Pour un enregistrement continu des évènements dans votre Compte AWS, y compris les évènements pour Amazon SNS, créez un journal d'activité. Un journal de suivi permet à CloudTrail de livrer des fichiers journaux dans un compartiment Amazon S3. Par défaut, lorsque vous créez un journal

de suivi dans la console, il s'applique à toutes les régions AWS. Le journal de suivi consigne les événements de toutes les Régions dans la partition AWS et livre les fichiers journaux dans le compartiment Amazon S3 de votre choix. En outre, vous pouvez configurer d'autres services AWS pour analyser et agir sur les données d'événements collectées dans les journaux CloudTrail. Pour en savoir plus, consultez les ressources suivantes :

- Présentation de la création d'un journal d'activité
- Intégrations et services pris en charge par CloudTrail
- Configuration des Notifications de Amazon SNS pour CloudTrail
- Réception des fichiers journaux CloudTrail de plusieurs régions et Réception des fichiers journaux CloudTrail de plusieurs comptes

Contrôler les événements de plan dans CloudTrail

Amazon SNS prend en charge la journalisation des actions suivantes en tant qu'évènements dans des fichiers journaux CloudTrail :

- `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**
>
> Lorsque vous n'êtes pas connecté à Amazon Web Services (mode non authentifié) et que les actions ConfirmSubscription ou Unsubscribe sont appelées, elles ne sont pas journalisées dans CloudTrail. Ainsi, lorsque vous choisissez le lien fourni dans une notification par e-mail afin de confirmer un abonnement en attente à une rubrique, l'action `ConfirmSubscription`

est appelée en mode non authentifié. Dans cet exemple, l'action `ConfirmSubscription` n'est pas journalisée dans CloudTrail.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec les informations d'identification utilisateur racine ou AWS Identity and Access Management (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre service AWS.

Pour plus d'informations, consultez l'élément userIdentity CloudTrail.

Événements de plan de données dans CloudTrail

Pour activer la journalisation des actions d'API suivantes dans des fichiers CloudTrail, vous devez activer la journalisation de l'activité de l'API de plan de données dans CloudTrail. Pour plus d'informations, consultez Journalisation des événements de données dans le Guide de l'utilisateur AWS CloudTrail.

Les événements de plan de données peuvent également être filtrés par type de ressource pour un contrôle détaillé sur les appels d'API Amazon SNS que vous souhaitez journaliser et payer de manière sélective dans CloudTrail. Par exemple, en spécifiant le type de ressource `AWS::SNS::Topic`, vous pouvez journaliser les appels des actions d'API `Publish` et `PublishBatch` pour les rubriques. De même, en spécifiant le type de ressource `AWS::SNS::PlatformEndpoint`, vous pouvez journaliser les appels de l'action d'API Publish pour les points de terminaison de la plateforme. Pour plus d'informations, consultez `AdvancedEventSelector` dans la Référence d'API AWS CloudTrail.

> **Note**
>
> Le type de ressource Amazon SNS `AWS::SNS::PhoneNumber` n'est pas journalisé par CloudTrail.

API de plan de données Amazon SNS

- Publish
- PublishBatch

Exemple : entrées du fichier journal Amazon SNS

Un journal d'activité est une configuration qui permet d'envoyer des événements sous forme de fichiers journaux à un compartiment Simple Storage Service (Amazon S3) que vous spécifiez. Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs entrées. Un événement représente une demande unique provenant de n'importe quelle source et comprend des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la requête, etc. Les fichiers journaux CloudTrail ne constituent pas une série ordonnée retraçant les appels d'API publiques. Ils ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre les actions `ListTopics`, `CreateTopic` et `DeleteTopic`.

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

Les exemples suivants montrent des entrées de journal CloudTrail qui illustrent les actions `Publish` et `PublishBatch`.

Publier

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T16:44:05Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T16:48:37Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "Publish",
```

```
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "message": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "subject": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageStructure": "json",
  "messageAttributes": "HIDDEN_DUE_TO_SECURITY_REASONS"
 },
 "responseElements": {
  "messageId": "0787cd1e-d92b-521c-a8b4-90434e8ef840"
 },
 "requestID": "0a8ab208-11bf-5e01-bd2d-ef55861b545d",
 "eventID": "bb3496d4-5252-4660-9c28-3c6aebdb21c0",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

PublishBatch

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
```

```
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAIOSFODNN7EXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/Admin",
     "accountId": "123456789012",
     "userName": "ExampleUser"
    },
    "attributes": {
     "creationDate": "2023-08-21T19:20:49Z",
     "mfaAuthenticated": "false"
    }
   }
  },
  "eventTime": "2023-08-21T19:22:01Z",
  "eventSource": "sns.amazonaws.com",
  "eventName": "PublishBatch",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
  "requestParameters": {
   "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
   "publishBatchRequestEntries": [{
     "id": "1",
     "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    {
     "id": "2",
     "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
    }
   ]
  },
  "responseElements": {
   "successful": [{
     "id": "1",
     "messageId": "30d68101-a64a-5573-9e10-dc5c1dd3af2f"
    },
    {
     "id": "2",
     "messageId": "c0aa0c5c-561d-5455-b6c4-5101ed84de09"
    }
```

```
  ],
  "failed": []
 },
 "requestID": "e2cdf7f3-1b35-58ad-ac9e-aaaea0ace2f1",
 "eventID": "10da9a14-0154-4ab6-b3a5-1825b229a7ed",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

Surveillance des rubriques Amazon SNS à l'aide de CloudWatch

Amazon SNS et Amazon CloudWatch étant intégrés, vous pouvez collecter, afficher et analyser des métriques pour chaque notification Amazon SNS active. Une fois que vous avez configuré CloudWatch pour Amazon SNS, vous disposez d'un meilleur aperçu des performances de vos rubriques Amazon SNS, des notifications push et des diffusions de SMS. Par exemple, vous pouvez définir une alarme qui vous envoie une notification par e-mail si un seuil défini est atteint pour une métrique Amazon SNS, comme `NumberOfNotificationsFailed`. Pour obtenir la liste de toutes les métriques envoyées par Amazon SNS à CloudWatch, consultez Métriques Amazon SNS. Pour plus d'informations sur les notifications push Amazon SNS, consultez Notifications push mobile.

> **Note**
>
> Les métriques que vous configurez avec CloudWatch pour vos rubriques Amazon SNS sont automatiquement collectées et envoyées à CloudWatch à intervalles d'1 minute. Ces métriques sont rassemblées sur toutes les rubriques qui sont considérées comme actives par CloudWatch. Une rubrique est considérée comme active par CloudWatch pendant six heures au maximum depuis la dernière activité (c'est-à-dire, un appel d'API) sur la rubrique.

Aucun frais n'est facturé pour les métriques Amazon SNS présentées dans CloudWatch. Elles sont fournies dans le cadre du service Amazon SNS.

Afficher les métriques CloudWatch pour Amazon SNS

Vous pouvez surveiller les métriques pour Amazon SNS à l'aide de la console CloudWatch, de l'interface de ligne de commande (CLI) CloudWatch, ou par programmation à l'aide de l'API CloudWatch. Les procédures suivantes vous expliquent comment accéder aux métriques avec la AWS Management Console.

Pour afficher les métriques à l'aide de la console CloudWatch

1. Connectez-vous à la console CloudWatch.
2. Dans le volet de navigation, choisissez Métriques.
3. Sous l'onglet All metrics (Toutes les métriques), choisissez SNS, puis l'une des dimensions suivantes :
   - Country, SMS Type
   - PhoneNumber
   - Topic Metrics
   - Metrics with no dimensions
4. Pour afficher plus de détails, choisissez un élément spécifique. Par exemple, si vous choisissez Métriques de rubrique puis NumberOfMessagesPublished, le nombre moyen de messages Amazon SNS publiés pendant une période d'une minute sur la plage de temps de 6 heures s'affiche.
5. Pour consulter les métriques d'utilisation d'Amazon SNS, dans l'onglet All metrics (Toutes les métriques), choisissez Usage (Utilisation), puis sélectionnez la target Amazon SNS usage metric (métrique d'utilisation Amazon SNS cible), par exemple, `NumberOfMessagesPublishedPerAccount`.

Définir les alarmes CloudWatch pour les métriques Amazon SNS

CloudWatch vous permet également de définir des alarmes lorsqu'un seuil est atteint pour une métrique. Par exemple, vous pouvez définir une alarme pour la métrique

NumberOfNotificationsFailed, afin de recevoir une notification par e-mail lorsque la valeur définie pour ce seuil est atteinte pendant la période d'échantillonnage.

Pour définir des alarmes à l'aide de la console CloudWatch

1. Connectez-vous à AWS Management Console et ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Choisissez Alarmes, puis sélectionnez le bouton Créer une alarme. L'assistant Create Alarm démarre alors.
3. Faites défiler les métriques Amazon SNS afin de rechercher celle sur laquelle vous souhaitez placer une alarme. Sélectionnez la métrique sur laquelle créer une alarme sur et choisissez Continuer.
4. Remplissez les champs Nom, Description, Seuil et Time de la métrique, puis choisissez Continuer.
5. Choisissez Alarm comme état de l'alarme. Si vous voulez que CloudWatch vous envoie un e-mail lorsque l'état de l'alarme est atteint, choisissez une rubrique Amazon SNS préexistante ou Créer une rubrique e-mail. Si vous choisissez Create New Email Topic (Créer une nouvelle rubrique d'e-mail), vous pouvez définir le nom et les adresses e-mail d'une nouvelle rubrique. Cette liste sera enregistrée et s'affichera dans la zone de liste déroulante des futures alarmes. Choisissez Continue (Continuer).

   > **Note**
   >
   > Si vous utilisez Créer une rubrique e-mail pour créer une rubrique Amazon SNS, les adresses e-mail doivent être vérifiées avant de recevoir des notifications. Les e-mails sont envoyés uniquement lorsque l'alarme passe à un état défini. Si ce changement d'état de l'alarme se produit avant la vérification des adresses e-mail, elles ne reçoivent pas de notification.

6. A ce stade, l'assistant Create Alarm vous donne la possibilité de passer en revue l'alarme que vous allez créer. Si vous avez besoin d'apporter des modifications, vous pouvez utiliser les liens Edit situés à droite. Une fois que vous êtes satisfait, choisissez Create Alarm (Créer une alarme).

Pour plus d'informations sur l'utilisation de CloudWatch et des alarmes, consultez la documentation CloudWatch.

Métriques Amazon SNS

Amazon SNS envoie les métriques suivantes à CloudWatch.

Espace de noms	Métrique	Description
AWS/SNS	`NumberOfMessagesPublished`	Nombre de messages publiés dans vos rubriques Amazon SNS. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme
AWS/SNS	`NumberOfNotificationsDelivered`	Nombre de messages transmis avec succès à des points de terminaison d'abonnement à partir de vos rubriques Amazon SNS. Pour qu'une tentative de remise réussisse, l'abonnement du point de terminaison doit accepter le message. Un abonnement accepte un message si a.) il manque une politique de filtrage ou b.) sa politique de filtrage contient des attributs qui correspondent à ceux qui sont affectés au message. Si l'abonnement rejette le message, la tentative de remise n'est pas comptabilisée pour cette métrique. Unités : nombre

Espace de noms	Métrique	Description
		Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme

Espace de noms	Métrique	Description
AWS/SNS	`NumberOfNotificationsFailed`	Nombre de messages qu'Amazon SNS n'a pas pu diffuser. Pour les points de terminaison Amazon SQS, e-mail, SMS ou push mobile, la métrique est incrémentée de 1 quand Amazon SNS arrête les tentatives de remise de message. Pour les points de terminaison HTTP ou HTTPS, la métrique inclut toutes les tentatives de remise en échec, y compris les tentatives qui suivent la tentative initiale. Pour tous les autres points de terminaison, le nombre augmente de 1 lorsque le message ne peut pas être remis (quel que soit le nombre de tentatives). Cette métrique ne comprend pas les messages qui ont été rejetés par des stratégies de filtre d'abonnement. Vous pouvez contrôler le nombre de nouvelles tentatives pour les points de terminaison HTTP. Pour de plus amples informations, veuillez consulter Nouvelle tentative de distribution des messages Amazon SNS. Unités : nombre

Espace de noms	Métrique	Description
		Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`NumberOfNotificationsFilteredOut`	Nombre de messages qui ont été rejetés par des stratégies de filtre d'abonnement. Une politique de filtre rejette un message lorsque les attributs du message ne correspondent pas à ses attributs. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Nombre de messages qui ont été rejetés par des politiques de filtre d'abonnement pour le filtrage basé sur les attributs. Unités : CountValid Dimensions : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne

Espace de noms	Métrique	Description
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Nombre de messages qui ont été rejetés par des politiques de filtre d'abonnement pour le filtrage basé sur la charge utile. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Nombre de messages qui ont été rejetés par des stratégies de filtre d'abonnement, car les attributs des messages ne sont pas valides – par exemple, le JSON d'un attribut est mal formaté. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne

Espace de noms	Métrique	Description
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	Nombre de messages qui ont été rejetés par des stratégies de filtre d'abonnement, car les messages n'ont pas d'attribut. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Nombre de messages qui ont été rejetés par des politiques de filtre d'abonnement, car le corps de message ne convient pas pour le filtrage – par exemple, corps de message JSON non valide. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne

Espace de noms	Métrique	Description
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Nombre de messages qui ont été déplacés vers une file d'attente de lettres mortes. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Nombre de messages qui n'ont pas pu être déplacés vers une file d'attente de lettres mortes. Unités : nombre Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`PublishSize`	Taille des messages publiés. Unités : octets Dimensions valides : Application, PhoneNumber, Platform et TopicName Statistiques valides : minimum, maximum, moyenne et nombre

Espace de noms	Métrique	Description
AWS/SNS	`SMSMonthToDateSpentUSD`	Frais cumulés depuis le début du mois calendaire en cours pour l'envoi de messages SMS. Vous pouvez définir une alarme pour cette métrique afin d'être informé lorsque les frais mensuels sont proches du quota de dépenses mensuelles pour l'envoi de messages SMS pour votre compte. Lorsqu'Amazon SNS détermine que l'envoi d'un SMS entraînerait un coût supérieur à ce quota, le service cesse la publication de SMS en quelques minutes. Pour plus d'informations sur la configuration de votre quota de dépenses mensuelles pour l'envoi de SMS, ou pour obtenir des informations sur l'augmentation du quota de dépenses avec AWS, consultez Définition des préférences de messagerie SMS. Unités : USD Dimensions valides : PhoneNumber Statistiques valides : maximum

Espace de noms	Métrique	Description
AWS/SNS	`SMSSuccessRate`	Taux de diffusions SMS réussies. Unités : nombre Dimensions valides : PhoneNumber Statistiques valides : somme, moyenne, exemples de données

Dimensions pour les métriques Amazon SNS

Amazon Simple Notification Service envoie les dimensions suivantes à CloudWatch.

Dimension	Description
`Application`	Filtre sur les objets d'application, qui représentent une applicati on et l'appareil enregistrés auprès de l'un des services de notification push pris en charge, tels qu'APNs et FCM.
`Application,Platform`	Filtre sur les objets d'application et de plateforme, où les objets de plateforme sont destinés aux services de notification push pris en charge, tels qu'APNs et FCM.
`Country`	Filtre sur le pays ou la région de destination d'un SMS. Le pays ou la région est représenté par son code ISO 3166-1 alpha-2.
`PhoneNumber`	Filtre sur le numéro de téléphone lorsque vous publiez des SMS directement sur un numéro de téléphone (sans rubrique).
`Platform`	Filtre sur les objets de plateforme pour les services de notificat ions push, tels qu'APNs et FCM.
`TopicName`	Filtre sur les noms de rubrique Amazon SNS.
`SMSType`	Filtre sur le type de SMS. Il peut être promotionnel ou transacti onnel.

Métriques d'utilisation Amazon SNS

Amazon Simple Notification Service envoie les métriques d'utilisation suivantes à CloudWatch.

Espace de noms	Service	Métrique	Ressource	Type	Description
AWS/Utilisation	SNS	`ResourceCount`	`NumberOfMessagesPublishedPerAccount`	Ressource	• Nombre de messages publiés dans vos rubriques Amazon SNS dans votre compte AWS. • Unités : aucune • Statistiques valides : somme
AWS/Utilisation	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Ressource	• Nombre approximatif de rubriques dans votre compte AWS. • Unités : aucune • Statistiques valides : moyenne, minimum,

Espace de noms	Service	Métrique	Ressource	Type	Description
					maximum, somme
AWS/Utilisation	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Ressource	• Nombre approximatif de politiques de filtre dans votre compte AWS. • Unités : aucune • Statistiques valides : moyenne, minimum, maximum, somme

Espace de noms	Service	Métrique	Ressource	Type	Description
AWS/Utilisation	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Ressource	• Nombre approximatif d'abonnements en attente dans votre compte AWS. • Unités : aucune • Statistiques valides : moyenne, minimum, maximum, somme

Espace de noms	Service	Métrique	Ressource	Type	Description
AWS/Utilisation	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber`	API	• Nombre d'appels d'API pour l'API Amazon SNS sélectionnée sur votre compte AWS. • Unités : aucune • Statistiques valides : somme

Espace de noms	Service	Métrique	Ressource	Type	Description
			• DeleteTopic • GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut		

Espace de noms	Service	Métrique	Ressource	Type	Description
			• ListPlatformApplications • ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes		

Espace de noms	Service	Métrique	Ressource	Type	Description
			• SetSubscriptionAttributes • SetTopicAttributes • Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Validation de conformité pour Amazon SNS

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon SNS dans le cadre de plusieurs programmes de conformité AWS, y compris la la sur la portabilité et la responsabilité en matière d'assurance maladie (HIPAA).

Pour obtenir une liste des services AWS relevant de programmes de conformité spécifiques, consultez Services AWS relevant de programme de conformité. Pour obtenir des renseignements généraux, consultez Programmes de conformité AWS .

Vous pouvez télécharger les rapports de l'audit externe avec AWS Artifact. Pour plus d'informations, veuillez consulter Téléchargement de rapports dans AWS Artifact.

Votre responsabilité de conformité lors de l'utilisation d'Amazon SNS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise, ainsi que la législation et la

réglementation applicables. AWS fournit les ressources suivantes pour faciliter le respect de la conformité :

- Guides de Quick Start (démarrage rapide) de la sécurité et de la conformité. Ces guides de déploiement traitent des considérations architecturales et fournissent des étapes pour déployer des environnements de base axés sur la sécurité et la conformité sur AWS.
- Livre blanc sur l'architecture pour la sécurité et la conformité HIPAA – Le livre blanc décrit comment les entreprises peuvent utiliser AWS pour créer des applications conformes à HIPAA.
- Ressources de conformité AWS – Cet ensemble de manuels et de guides peut s'appliquer à votre secteur d'activité et à votre emplacement.
- Évaluation des ressources à l'aide de règles dans le Guide du développeur AWS Config : le service AWS Config évalue dans quelle mesure vos configurations de ressources sont conformes aux pratiques internes, aux directives sectorielles et aux réglementations.
- AWS Security Hub – Ce service AWS fournit une vue complète de votre état de sécurité au sein d'AWS qui vous permet de vérifier votre conformité aux normes du secteur et aux bonnes pratiques de sécurité.

Résilience dans Amazon SNS

La résilience d'Amazon SNS est garantie en tirant parti de l'infrastructure AWS mondiale, qui s'articule autour des zones Régions AWS de disponibilité. Régions AWS proposent des zones de disponibilité physiquement séparées et isolées connectées par un réseau à faible latence, à haut débit et hautement redondant. Cette architecture permet un basculement fluide entre les zones de disponibilité sans interruption, ce qui rend les applications et les bases de données intrinsèquement plus tolérantes aux pannes et évolutives par rapport aux infrastructures de centres de données traditionnelles. En utilisant les zones de disponibilité, les abonnés Amazon SNS bénéficient d'une disponibilité et d'une fiabilité accrues, garantissant ainsi la livraison des messages malgré les perturbations potentielles. Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

En outre, les abonnements aux rubriques Amazon SNS peuvent être configurés avec de nouvelles tentatives de livraison et des files d'attente de lettres mortes, ce qui permet de gérer automatiquement les défaillances transitoires et de garantir que les messages atteignent de manière fiable leurs destinations prévues.

Amazon SNS prend également en charge le filtrage des messages et les attributs des messages, ce qui vous permet d'adapter les stratégies de résilience à leurs cas d'utilisation spécifiques, améliorant ainsi la robustesse globale de vos applications.

Sécurité de l'infrastructure dans Amazon SNS

En tant que service géré, Amazon SNS est protégé par les procédures de sécurité du réseau AWS mondial décrites dans la documentation relative aux meilleures pratiques en matière de sécurité, d'identité et de conformité.

Utilisez les actions d' AWS API pour accéder à Amazon SNS via le réseau. Les clients doivent prendre en charge le protocole TLS (Transport Layer Security) 1.2 ou version ultérieure. Les clients doivent également prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)

Vous devez signer les demandes à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires afin de signer les demandes.

Vous pouvez appeler ces actions d'API à partir de n'importe quel emplacement sur le réseau, mais Amazon SNS prend en charge les politiques d'accès basées sur les ressources, ce qui peut inclure des restrictions en fonction de l'adresse IP source. Vous pouvez également utiliser des politiques Amazon SNS pour contrôler l'accès à partir de points de terminaison Amazon VPC ou de VPC spécifiques. Cela permet d'isoler efficacement l'accès réseau à une rubrique Amazon SNS donnée uniquement du VPC spécifique au sein du réseau. AWS Pour plus d'informations, voir Limiter la publication à une rubrique Amazon SNS uniquement à partir d'un point de terminaison de VPC spécifique.

Bonnes pratiques de sécurité pour Amazon SNS

AWS offre de nombreuses fonctionnalités de sécurité pour Amazon SNS. Déterminez en quoi ces fonctionnalités de sécurité peuvent être utiles dans le contexte de votre propre stratégie de sécurité.

> **Note**
>
> Les instructions relatives à ces fonctionnalités de sécurité s'appliquent aux cas d'utilisation et aux implémentations courants. Nous vous suggérons de consulter ces bonnes pratiques dans

le contexte de votre cas d'utilisation, de votre architecture et de votre modèle spécifique de lutte contre les menaces.

Bonnes pratiques en matière de prévention

Voici les bonnes pratiques de sécurité préventive pour Amazon SNS.

Rubriques

- Vérification que les rubriques ne sont pas accessibles publiquement
- Implémentation d'un accès sur la base du moindre privilège
- Utilisation de rôles IAM pour les applications et les services AWS nécessitant un accès à Amazon SNS
- Mise en œuvre du chiffrement côté serveur
- Application du chiffrement des données en transit
- Réflexion sur l'utilisation des points de terminaison de VPC pour accéder à Amazon SNS
- Assurez-vous que les abonnements ne sont pas configurés pour être livrés aux points de terminaison http bruts

Vérification que les rubriques ne sont pas accessibles publiquement

À moins que vous n'ayez explicitement demandé que quiconque sur Internet puisse lire ou écrire dans votre rubrique Amazon SNS, vous devez vous assurer que celle-ci n'est pas accessible publiquement (accessible par tout le monde ou par tout utilisateur AWS authentifié).

- Évitez de créer des stratégies avec `Principal` défini sur `""`.
- Évitez d'utiliser un caractère générique (`*`). Nommez plutôt un ou plusieurs utilisateurs spécifiques.

Implémentation d'un accès sur la base du moindre privilège

Lorsque vous accordez des autorisations, vous décidez qui les reçoit, à quelles rubriques celles-ci sont destinées et quelles actions d'API spécifiques vous souhaitez autoriser pour ces rubriques. La mise en œuvre du principe du moindre privilège est cruciale pour réduire les risques en matière de sécurité. Elle contribue également à réduire l'effet négatif des erreurs ou des intentions malveillantes.

Suivez les conseils de sécurité standard pour accorder le moindre privilège. Autrement dit, accordez uniquement les autorisations requises pour effectuer une tâche spécifique. Vous pouvez implémenter le moindre privilège en utilisant une combinaison de stratégies de sécurité relatives à l'accès des utilisateurs.

Amazon SNS utilise le modèle éditeur-abonné, nécessitant trois types d'accès de compte utilisateur :

- Administrateurs - Accès à la création, à la modification et à la suppression de rubriques. Les administrateurs contrôlent également les stratégies de rubrique.
- Éditeurs – Accès à l'envoi de messages à des rubriques.
- Abonnés – Accès à l'abonnement à des rubriques.

Pour plus d'informations, consultez les sections suivantes :

- Gestion des identités et des accès dans Amazon SNS
- Autorisations d'API Amazon SNS : référence des actions et ressources

Utilisation de rôles IAM pour les applications et les services AWS nécessitant un accès à Amazon SNS

Pour que des applications ou des services AWS comme Amazon EC2 accèdent à des rubriques Amazon SNS, ceux-ci doivent utiliser des informations d'identification AWS valides dans leurs demandes d'API AWS. Comme ces informations d'identification ne font pas l'objet d'une rotation automatique, vous ne devez pas stocker des informations d'identification AWS directement dans l'application ou l'instance EC2.

Vous devez utiliser un rôle IAM pour gérer des informations d’identification temporaires pour les applications ou services devant accéder à Amazon SNS. Quand vous utilisez un rôle, vous n'avez pas besoin de distribuer des informations d'identification à long terme (telles qu'un nom d'utilisateur, un mot de passe et des clé d'accès) à une instance EC2 ou à un service AWS comme AWS Lambda. À la place, le rôle fournit des autorisations temporaires que les applications peuvent utiliser lors d'appels à d'autres ressources AWS.

Pour de plus amples informations, veuillez consulter Rôles IAMet Scénarios courants pour les rôles : utilisateurs, applications et services dans le guide de l’utilisateur.

Mise en œuvre du chiffrement côté serveur

Pour atténuer les problèmes de fuite de données, utilisez le chiffrement au repos pour chiffrer vos messages à l'aide d'une clé stockée dans un emplacement différent de celui où les messages sont stockés. Le chiffrement côté serveur (SSE) fournit le chiffrement des données au repos. Amazon SNS chiffre vos données au niveau des messages lorsqu'il les stocke et déchiffre les messages pour vous lorsque vous y accédez. SSE utilise des clés gérées dans AWS Key Management Service. Tant que vous authentifiez votre demande et que vous avez des autorisations d'accès, il n'y a aucune différence entre l'accès aux rubriques chiffrées et aux rubriques déchiffrées.

Pour plus d'informations, consultez Chiffrement au repos et Gestion des clés.

Application du chiffrement des données en transit

Il est possible, mais non recommandé, de publier des messages qui ne sont pas chiffrés pendant le transit en utilisant HTTP. Cependant, vous ne pouvez pas utiliser HTTP lors de la publication dans une rubrique SNS chiffrée.

AWS recommande d'utiliser HTTPS au lieu de HTTP. Lorsque vous utilisez HTTPS, les messages sont automatiquement chiffrés pendant le transit, même si la rubrique SNS elle-même n'est pas chiffrée. Sans HTTPS, un pirate sur le réseau peut écouter le trafic réseau ou le manipuler, en utilisant une attaque telle qu'une attaque de l'homme du milieu.

Pour appliquer uniquement les connexions chiffrées via HTTPS, ajoutez la condition `aws:SecureTransport` dans la politique IAM attachée aux rubriques SNS non chiffrées. Cela force les éditeurs de messages à utiliser HTTPS au lieu de HTTP. Vous pouvez utiliser l'exemple de politique suivant à titre de guide :

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

Réflexion sur l'utilisation des points de terminaison de VPC pour accéder à Amazon SNS

Si vous avez des rubriques avec lesquelles vous devez pouvoir interagir mais qui ne doivent absolument pas être exposées à Internet, utilisez des points de terminaison de VPC afin d'en limiter l'accès aux seuls hôtes qui font partie de ce VPC. Vous pouvez utiliser des stratégies de rubrique pour contrôler l'accès aux rubriques à partir de points de terminaison Amazon point de terminaison d'un VPC ou de VPC spécifiques.

Les points de terminaison Amazon SNS VPC offrent deux façons de contrôler l'accès à vos messages :

- Vous pouvez contrôler les demandes, les utilisateurs ou les groupes autorisés à traverser un point de terminaison d'un VPC spécifique.
- Vous pouvez contrôler quels VPC ou points de terminaison de VPC ont accès à votre rubrique à l'aide d'une stratégie de rubrique.

Pour plus d'informations, consultez Création du point de terminaison et Création d'une politique de point de terminaison d'Amazon VPC pour Amazon SNS.

Assurez-vous que les abonnements ne sont pas configurés pour être livrés aux points de terminaison http bruts

Évitez de configurer les abonnements pour effectuer une livraison à des points de terminaison http bruts. Ayez toujours des abonnements livrés au nom de domaine d'un de point de terminaison. Par exemple, un abonnement configuré pour effectuer une livraison à un point de terminaison, `http://1.2.3.4/my-path`, doit être remplacé par `http://my.domain.name/my-path`.

Résolution des problèmes liés aux rubriques Amazon SNS

Cette section fournit des informations sur le dépannage des rubriques Amazon SNS.

Résolution des problèmes liés aux rubriques Amazon SNS avec AWS X-Ray

AWS X-Ray collecte des données sur des demandes servies par votre application, et vous permet d'afficher et de filtrer des données afin d'identifier les problèmes potentiels et les possibilités d'optimisation. Pour toutes les demandes suivies transmises à votre application, vous pouvez consulter des informations détaillées sur la demande et la réponse, mais également sur les appels que votre application effectue vers des bases de données, microservices, ressources AWS en aval et API web HTTP.

Vous pouvez utiliser X-Ray avec Amazon SNS pour suivre et analyser les messages qui transitent par le biais de votre application. Vous pouvez utiliser la AWS Management Console pour afficher la carte de connexions entre Amazon SNS et les autres services que votre application utilise. Vous pouvez également utiliser la console pour afficher des mesures comme la latence moyenne et les taux de défaillance. Pour plus d'informations, consultez Amazon SNS et AWS X-Ray dans le AWS X-RayGuide du développeur.

Suivi actif dans Amazon SNS

Vous pouvez l'utiliser AWS X-Ray pour suivre et analyser les demandes des utilisateurs lorsqu'elles passent par vos rubriques Amazon SNS jusqu'à vos abonnements Amazon Data Firehose, Amazon AWS LambdaSQS et HTTP/S endpoint. Dans la mesure où X-Ray vous donne un end-to-end aperçu de l'intégralité d'une demande, vous pouvez voir ce qui s'appelle votre rubrique Amazon SNS et ce qui se trouve en aval des abonnements à votre rubrique. Vous pouvez analyser les latences de vos messages et de leurs services backend (par exemple, combien de temps une demande passe dans une rubrique et combien de temps il a fallu pour transmettre le message à chacun des abonnements de la rubrique).

> **Important**
>
> Les rubriques Amazon SNS ayant de nombreux abonnements peuvent atteindre une taille limite et ne pas être entièrement suivies. Pour plus d'informations sur les limites de taille des documents de suivi, consultez Service Quotas de X-Ray dans la référence générale AWS.

Si vous appelez une API Amazon SNS à partir d'un service qui est déjà suivi, Amazon SNS transmet le suivi, même si le suivi X-Ray n'est pas activé sur l'API.

Amazon SNS prend en charge le suivi X-Ray pour les rubriques standard et FIFO. Vous pouvez activer X-Ray pour une rubrique Amazon SNS à l'aide de la console Amazon SNS, de l'API Amazon SNS `SetTopicAttributes`, de la Référence d'interface de ligne de commande Amazon Simple Notification Service ou d'AWS CloudFormation.

Pour en savoir plus sur l'utilisation d'Amazon SNS avec X-Ray, consultez Amazon SNS et AWS X-Ray dans le Guide du développeur AWS X-Ray.

Rubriques

Autorisations de suivi actif

Lorsque vous utilisez la console Amazon SNS, Amazon SNS tente de créer les autorisations nécessaires pour que la rubrique Amazon SNS appelle X-Ray. La tentative peut être rejetée si vous ne disposez pas des autorisations nécessaires pour utiliser la console Amazon SNS. Pour plus d'informations, consultez Gestion des identités et des accès dans Amazon SNS et Cas d'exemple pour le contrôle d'accès Amazon SNS.

Lorsque vous utilisez l'interface de ligne de commande, vous devez configurer les autorisations manuellement. Ces autorisations sont configurées à l'aide de politiques de ressources. Pour en

savoir plus sur l'utilisation des autorisations requises dans X-Ray, consultez Amazon SNS et AWS X-Ray.

Activation du suivi actif sur une rubrique Amazon SNS (console)

Lorsque le suivi actif est activé sur une rubrique Amazon SNS, il lit l'ID de suivi, envoie les données au client en fonction de l'ID de suivi et propage l'ID de suivi aux services en aval.

1. Connectez-vous à la console Amazon SNS.
2. Choisissez une rubrique ou créez-en une. Pour plus d'informations sur la création de rubriques, consultez Création d'une rubrique Amazon SNS.
3. Sur la page Créer une rubrique, dans la section Détails, choisissez un type de rubrique : FIFO ou Standard.
   a. Entrez un Nom pour la rubrique.
   b. (Facultatif) Entrez un Nom d'affichage pour votre rubrique.
4. Développez Active tracing (Suivi actif) et choisissez Use active tracing (Utiliser le suivi actif).

Une fois que vous avez activé X-Ray pour votre rubrique Amazon SNS, vous pouvez utiliser la carte des services X-Ray pour afficher les end-to-end traces et les cartes de service associées à cette rubrique.

Activation du suivi actif sur une rubrique Amazon SNS (AWS SDK)

L'exemple de code suivant montre comment activer le suivi actif sur une rubrique Amazon SNS à l'aide d'AWS SDK pour 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());
        }
    }
```

Activation du suivi actif sur une rubrique Amazon SNS (interface de ligne de commande AWS)

L'exemple de code suivant montre comment activer le suivi actif sur une rubrique Amazon SNS à l'aide de l'interface de ligne de commande AWS.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Activation du suivi actif sur une rubrique Amazon SNS (AWS CloudFormation)

La pile AWS CloudFormation suivante montre comment activer le suivi actif sur une rubrique Amazon SNS.

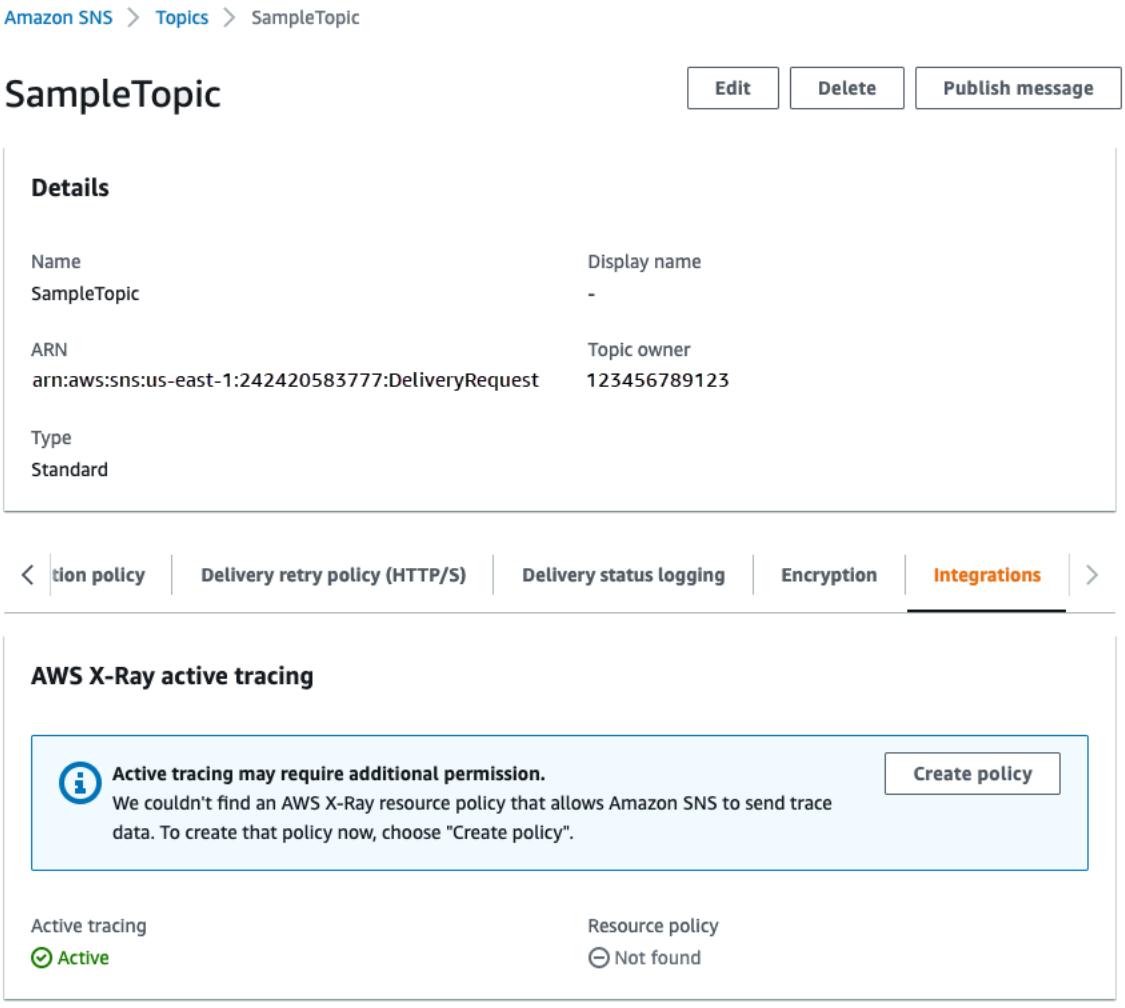
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Vérifier que le suivi actif est activé pour votre rubrique

Vous pouvez utiliser la console Amazon SNS pour vérifier si le suivi actif est activé pour votre rubrique ou quand la politique de ressources n'a pas pu être ajoutée.

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation de gauche, choisissez Rubriques.
3. Sur la page Topics (Rubriques), choisissez une rubrique.
4. Sélectionnez l'onglet Integrations (Intégrations).

   Lorsque le suivi actif est activé, une icône Active (Actif) verte s'affiche.
5. Si vous avez activé le suivi actif et que vous ne voyez pas que la politique de ressources a été ajoutée, choisissez Create policy (Créer une politique) pour ajouter les autorisations supplémentaires requises.



Tester le suivi actif

1. Connectez-vous à la console Amazon SNS.
2. Créer une rubrique Amazon SNS. Pour plus d'informations sur la manière de procéder, consultez Pour créer un sujet à l'aide du AWS Management Console.
3. Développez Active tracing (Suivi actif) et choisissez Use active tracing (Utiliser le suivi actif).
4. Publiez un message dans la rubrique Amazon SNS. Pour plus d'informations sur la manière de procéder, consultez Pour publier des messages dans une rubrique Amazon SNS à l'aide de la AWS Management Console.
5. Utilisez la carte des services X-Ray pour afficher les end-to-end traces et les cartes de service associées au sujet.

Historique de la documentation

Le tableau suivant décrit les modifications récentes apportées au Guide du développeur Amazon Simple Notification Service.

Les fonctionnalités du service sont parfois déployées progressivement AWS dans les régions où un service est disponible. Nous mettons à jour cette documentation pour la première version uniquement. Nous ne fournissons pas d'informations sur la disponibilité des régions et n'annonçons pas les déploiements régionaux ultérieurs. Pour plus d'informations sur la disponibilité des fonctionnalités du service dans les régions et pour vous abonner aux notifications concernant les mises à jour, voir Quelles sont les nouveautés AWS ? .

Modification	Description	Date
Soutien de Canada West (Calgary) pour les sujets du FIFO	Amazon SNS prend en charge le thème FIFO dans l'ouest du Canada (Calgary).	28 mars 2024
Support des SMS Amazon SNS dans cinq nouvelles régions	Amazon SNS a ajouté la prise en charge des SMS dans les régions suivantes : Asie-Pacifique (Hyderabad), Asie-Pacifique (Melbourne), Moyen-Orient (Émirats arabes unis), Europe (Zurich) et Europe (Espagne).	8 février 2024
Prise en charge du protocole HTTP v1 de Firebase Cloud Messaging (FCM)	Amazon SNS prend en charge les informations d'identification FCM v1.	18 janvier 2024
SMS Amazon SNS pris en charge dans la région Asie-Pacifique (Jakarta)	Amazon SNS prend en charge les SMS dans la région Asie-Pacifique (Jakarta).	14 décembre 2023
AWS CloudFormation support pour la configuration `DeliveryStatusLogg`	AWS CloudFormation une assistance est disponible pour la configuration `DeliveryS`	7 décembre 2023

ing des rubriques Amazon SNS	tatusLogging lors de la création ou de la mise à jour de rubriques Amazon SNS.	
Nouveaux opérateurs de filtrage des messages ajoutés	Vous pouvez désormais utiliser la correspondance de suffixes et les opérateur s equals-ignore case et OR pour filtrer les messages Amazon SNS.	16 novembre 2023
Ajout de la prise en charge de l'archivage-relecture des messages	Les propriétaires de rubrique peuvent archiver les messages dans une rubrique pour une durée maximale de 365 jours. Les abonnés à une rubrique peuvent relire les messages archivés sur un point de terminaison abonné afin de récupérer les messages suite à la défaillan ce d'une application en aval ou pour répliquer l'état d'une application existante.	26 octobre 2023
Ajout de la prise en charge de l'abonnement d'une file d'attente standard à une rubrique FIFO	Vous pouvez abonner une file d'attente FIFO Amazon SQS ou une file d’attente standard à une rubrique FIFO Amazon SNS. Seules les files d'attente FIFO Amazon SQS garantissent la réception des messages dans l'ordre et sans doublons.	14 septembre 2023

Ajout de la prise en charge des SMS pour Israël (Tel Aviv)	Les SMS Amazon SNS sont désormais pris en charge dans la région Israël (Tel Aviv).	28 août 2023
Prise en charge du suivi actif X-Ray ajoutée pour les rubriques FIFO	Auparavant uniquement compatible avec les rubriques standard Amazon SNS, elle permet AWS X-Ray désormais de suivre et d'analyser les demandes des utilisateurs lorsqu'elles passent par vos rubriques FIFO vers vos abonnements Amazon Data Firehose, Amazon AWS Lambda SQS et HTTP/S Endpoint.	31 mai 2023
Prise en charge améliorée des en-têtes Content-Type	Vous pouvez définir l'en-tête Content-Type dans la politique de demande pour spécifier le type de support de la notification.	23 mars 2023
Ajout de la prise en charge du suivi actif	AWS X-Ray suit et analyse les demandes des utilisateurs au fur et à mesure qu'elles passent par les rubriques standard Amazon SNS pour accéder à vos abonnements Amazon Data Firehose, Amazon AWS Lambda SQS et HTTP/S endpoint.	8 février 2023
Enregistrement de l'ID d'expéditeur pour Singapour	Instructions ajoutées pour l'enregistrement des ID d'expéditeur pour Singapour.	10 janvier 2023

Filtrage des messages basé sur la charge utile	Le filtrage basé sur la charge utile vous permet de filtrer les messages en fonction de la charge utile des messages et d'éviter les coûts associés au traitement des données indésirables.	22 novembre 2022
Un algorithme de hachage SHA256 a été ajouté pour la signature des messages Amazon SNS	Prise en charge ajoutée pour un algorithme de hachage SHA256 pour l'utilisation de la signature des messages Amazon SNS.	15 septembre 2022
Régions supplémentaires ajoutées aux SMS	Amazon SNS prend en charge la messagerie SMS dans les régions suivantes : Afrique (Le Cap), Asie-Pacifique (Osaka), Europe (Milan) et AWS GovCloud (États-Unis de l'Est).	9 septembre 2022
Prise en charge de message data protection ajoutée	La protection des données des messages protège les données publiées sur vos rubriques Amazon SNS en utilisant des politiques de protection des données pour auditer et bloquer les informations sensibles qui circulent entre les applications ou AWS les services.	8 septembre 2022

Nouveau processus d'enregistrement pour les numéros gratuits	Ajout de la prise en charge de l'envoi de messages Amazon SNS à l'aide de numéros de téléphone gratuits à des destinataires américains.	1er août 2022
Prise en charge des contrôles d'accès basés sur les attributs (ABAC)	Ajout de la prise en charge du contrôle d'accès basé sur les attributs (ABAC) pour les actions d'API, notamment `Publish` et `PublishBatch` . L'ABAC est une stratégie d'autorisation qui définit les autorisations d'accès en fonction de balises qui peuvent être associées aux ressources IAM, telles que les utilisateurs et les rôles IAM, et aux AWS ressources, telles que les rubriques Amazon SNS, afin de simplifier la gestion des autorisations.	10 janvier 2022
Prise en charge de l'authentification par jeton Apple pour les notifications push	Vous pouvez autoriser Amazon SNS à envoyer des notifications push à votre appli iOS ou macOS en communiquant des informations qui vous identifient en tant que développeur de l’application.	28 octobre 2021

Les nouveaux expéditeurs de messages SMS sont placés dans l'environnement de test (sandbox) SMS	L'environnement de test (sandbox) SMS existe pour prévenir les fraudes et les abus, et pour vous aider à protéger votre réputation en tant qu'expéditeur. Lorsque votre AWS compte est dans le sandbox SMS, vous ne pouvez envoyer des SMS qu'à des numéros de téléphone de destination vérifiés.	1er juin 2021
Les nouveaux expéditeurs de messages SMS sont placés dans l'environnement de test (sandbox) SMS	L'environnement de test (sandbox) SMS existe pour prévenir les fraudes et les abus, et pour vous aider à protéger votre réputation en tant qu'expéditeur. Lorsque votre AWS compte est dans le sandbox SMS, vous ne pouvez envoyer des SMS qu'à des numéros de téléphone de destination vérifiés.	1er juin 2021
Nouveaux attributs pour l'envoi de messages SMS à des destinataires en Inde	Deux nouveaux attributs, ID de l'entité et l'ID de modèle, sont maintenant requis pour l'envoi de messages SMS à des destinataires en Inde.	22 avril 2021

Mises à jour des opérateurs de filtrage des messages	Un nouvel opérateur, `cidr`, est disponible pour les adresses IP source de message et les sous-réseaux correspondants. Vous pouvez désormais également vérifier l'absence d'une clé d'attribu t et utiliser un préfixe avec l'opérateur `anything-but` pour la correspondance de chaîne d'attribut.	7 avril 2021
Fin de la prise en charge des codes longs P2P pour les destinations américaines	À compter du 1er juin 2021, les fournisseurs de télécommu nications américains ne prennent plus en charge l'utilisation de codes longs person-to-person (P2P) pour les communications applicati on-to-person (A2P) vers des destinations américaines. Au contraire, vous pouvez utiliser des codes courts, des numéros gratuits ou un nouveau type de numéro d'origine appelé 10DLC.	16 février 2021
Support pour les métriques Amazon CloudWatch d'une minute	La CloudWatch métrique d'une minute pour Amazon SNS est désormais disponible dans AWS toutes les régions.	28 janvier 2021

Support pour les points de terminaison Amazon Data Firehose	Vous pouvez abonner les flux de diffusion Firehose aux rubriques SNS. Cela vous permet d'envoyer des notifications aux points de terminaison d'archivage et d'analyse tels que les buckets Amazon Simple Storage Service (Amazon S3), les tables Amazon Redshift, Amazon Service (Service), OpenSearch etc. OpenSearch	12 janvier 2021
Les numéros d'origine sont disponibles	Vous pouvez utiliser des numéros d'origine lors de l'envoi de messages texte (SMS).	23 octobre 2020
Prise en charge des rubriques FIFO Amazon SNS	Pour intégrer des applications distribuées nécessitant la cohérence des données en temps quasi réel, vous pouvez utiliser les rubriques « First In, First Out » (FIFO) d'Amazon SNS avec les files d'attente FIFO d'Amazon SQS.	22 octobre 2020
La bibliothèque client étendue Amazon SNS pour Java est disponible	Vous pouvez utiliser cette bibliothèque pour publier des messages Amazon SNS volumineux.	25 août 2020
SSE est disponible dans les régions de Chine	Le chiffrement côté serveur (SSE) pour Amazon SNS est disponible dans les régions de Chine.	20 janvier 2020

Prise en charge de l'utilisation de DLQ pour capturer les messages non livrables	Pour récupérer les messages non distribuables, vous pouvez utiliser une file d'attente de lettres mortes (DLQ) Amazon SQS avec un abonnement Amazon SNS.	14 novembre 2019
Prise en charge de la spécification de valeurs d'en-tête APNs personnalisées	Vous pouvez spécifier une valeur d'en-tête APNs personnalisée.	18 octobre 2019
Prise en charge du champ d'en-tête « `apns-push-type` » pour APNs	Vous pouvez utiliser le champ d'en-tête `apns-push-type` pour les notifications mobiles envoyées via APNs.	10 septembre 2019
Support pour le dépannage des rubriques à l'aide AWS X-Ray	Vous pouvez utiliser X-Ray pour résoudre le problème de messages transitant par des rubriques SNS.	24 juillet 2019
Prise en charge de la correspondance d'attribut de clé avec l'opérateur « `exists` »	Vous pouvez utiliser l'opérateur `exists` pour vérifier si un message entrant possède un attribut dont la clé est répertoriée dans la stratégie de filtrage.	5 juillet 2019
Prise en charge de la correspondance « anything-but » (tout sauf) de plusieurs valeurs numériques	En plus de plusieurs chaînes, Amazon SNS permet la correspondance « anything-but » (tout sauf) de plusieurs valeurs numériques.	5 juillet 2019
Les notes de mise à jour d'Amazon SNS sont disponibles sous la forme d'un flux RSS	En suivant le titre sur cette page (Historique de la documentation), choisissez RSS.	22 juin 2019

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le Glossaire AWS dans la Référence Glossaire AWS.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.