Guide du développeur

Amazon Simple Notification Service

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

Amazon Simple Notification Service: Guide du développeur

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

Les marques commerciales et la présentation commerciale d’Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d’une manière susceptible d’entraîner une confusion chez les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce qu'Amazon SNS ? ... 1
Comment ça marche ... 1
Accès à Amazon SNS ... 1
Scénarios Amazon SNS courants ... 2
Intégration d'applications ... 2
Alertes d'application ... 3
Notifications utilisateur ... 3
Notifications push mobile ... 4
Tarification pour Amazon SNS ... 4
Fonctionnalités et fonctionnalités d'Amazon SNS ... 4
Services couramment partagés ... 6
Travailler avec AWS SDKs ... 9
Création d'un sujet et publication de messages ... 11
Configuration ... 11
Créer un compte et un utilisateur IAM ... 11
Étapes suivantes ... 13
Étape 1 : Création d'un sujet ... 14
AWS Management Console ... 14
AWS SDKs ... 17
Étape 2 : Création d'un abonnement à un sujet ... 33
Pour abonner un point de terminaison à une rubrique Amazon SNS ... 33
Étape 3 : publication d'un message ... 35
AWS Management Console ... 35
AWS SDKs ... 37
Charges utiles de messages volumineux ... 61
Attributs de message ... 73
Traitement par lots de messages ... 78
Étape 4 : Supprimer un abonnement et un sujet ... 81
AWS Management Console ... 82
AWS SDKs ... 82
Étapes suivantes ... 93
Ordre des messages et déduplication à l'aide des rubriques FIFO ... 94
Rubriques FIFO à haut débit ... 94
Cas d'utilisation ... 95

Partitions et distribution des données ... 95
Partitions et distribution des données ... 96
Permettre un débit élevé ... 96
Activer le mode haut débit pour les files d'attente Amazon SQS ... 97
Cas d'utilisation de la rubrique FIFO ... 97
Détails d'ordre des messages ... 98
Regroupement de messages ... 99
Répartition des données par groupe de messages IDs pour améliorer les performances ... 100
Distribution des messages ... 101
Filtrage de messages ... 102
Déduplication de message ... 103
Sécurité des messages ... 105
Durabilité des messages ... 106
Archivage-relecture des messages ... 107
Qu'est-ce que l'archivage-relecture des messages ... 107
Pour les propriétaires de rubrique ... 109
Pour les abonnés à une rubrique ... 114
Exemples de code ... 118
Exemple de FIFO (AWS SDKs) ... 119
Exemple de FIFO (AWS CloudFormation) ... 132
Filtrage de messages ... 136
Étendue de la politique de filtre d'abonnement ... 136
Stratégies de filtre d'abonnement ... 137
Exemples de politiques de filtrage Amazon SNS ... 138
Contraintes de politique de filtre ... 141
Logique AND/OR ... 145
Correspondance de clé ... 149
Correspondance de valeur numérique ... 152
Correspondance de valeur de chaîne ... 154
Application d'une politique de filtrage d'abonnement ... 162
AWS Management Console ... 163
AWS CLI ... 164
AWS SDKs ... 165
API Amazon SNS ... 170
AWS CloudFormation ... 170
Suppression d'une politique de filtrage d'abonnement ... 171

À l'aide du AWS Management Console ... 171
À l'aide du AWS CLI ... 171
Utilisation de l'API Amazon SNS ... 171
Message Data Protection ... 172
Qu'est-ce que Message Data Protection ? ... 172
Pourquoi utiliser Message Data Protection ? ... 173
Politiques de protection des données ... 173
En quoi consistent les politiques de protection des données ? ... 173
Vue d'ensemble de la structure de protection des données ... 174
Comment puis-je déterminer les principaux IAM ... 176
Opérations de la politique de protection des données ... 176
Exemples de politiques de protection des données ... 184
Création de politiques de protection des données ... 191
Suppression de politiques de protection des données ... 201
identifiants des données ... 202
Identificateurs de données gérés ... 202
Identificateurs des données personnalisés ... 243
Distribution des messages ... 246
Remise des messages bruts ... 246
Activation de la remise des messages bruts avec AWS Management Console ... 247
Exemples de format des messages ... 247
Attributs des messages et envoi de messages bruts pour les abonnements Amazon SQS ... 248
Diffusion entre comptes ... 248
Création de l'abonnement par le propriétaire de la file d'attente ... 249
Création de l'abonnement par un utilisateur qui n'est pas le propriétaire de la file d'attente ... 251
Comment forcer un abonnement à exiger une authentification sur les demandes de désabonnement ? ... 254
Livraison interrégionale ... 254
Régions d'activation ... 255
Statut de distribution des messages ... 258
Conditions préalables à l'enregistrement de l'état de livraison ... 258
Configuration de le statut de distribution à l'aide de AWS Management Console ... 259
Configuration de l'enregistrement de l'état de livraison à l'aide du AWS SDKs ... 261
AWS Exemples de SDK pour configurer les attributs des rubriques ... 263
Configuration de la journalisation du statut de distribution à l'aide d' AWS CloudFormation ... 272
Nouvelles tentatives de distribution des messages ... 273

Protocoles et politiques de distribution ... 273
Étapes de la politique de distribution ... 274
Création d'une politique HTTP/S de livraison ... 275
Files d'attente de lettres mortes ... 281
Pourquoi les distributions de messages échouent-elles ? ... 282
Fonctionnement des files d'attente de lettres mortes ... 283
Comment les messages sont-ils déplacés dans une file d'attente de lettres mortes ? ... 283
Comment puis-je déplacer des messages hors d'une file d'attente de lettres mortes ? ... 284
Comment puis-je contrôler et consigner les files d'attente de lettres mortes ? ... 284
Configuration d'une file d'attente de lettre morte ... 285
Archivage et analytique des messages ... 291
Gestion et optimisation des ressources ... 292
Identification ... 292
Identification pour l'allocation des coûts ... 292
Identification pour le contrôle d'accès ... 293
Identification pour la recherche et le filtrage des ressources ... 294
Configuration des identifications ... 295
Sources et destinations des évènements Amazon SNS ... 302
Sources des évènements ... 302
Analyse ... 302
Intégration d'applications ... 303
Facturation et gestion des coûts ... 304
Applications métier ... 305
Calcul ... 305
Conteneurs ... 307
Engagement client ... 307
Base de données ... 308
Outils pour développeurs ... 310
Web et mobile frontaux ... 311
Développement de jeux ... 312
Internet des objets ... 312
Machine learning ... 313
Gestion et gouvernance ... 315
Multimédia ... 317
Migration et transfert ... 318
Mise en réseau et diffusion de contenu ... 319

Sécurité, identité et conformité ... 320
Sans serveur ... 321
Stockage ... 322
Sources supplémentaires des évènements ... 324
Destinations d'évènement ... 325
Destinations A2A ... 326
Destinations A2P ... 327
Application-to-application messagerie ... 330
Streams de diffusion de Fanout to Firehose ... 330
Prérequis ... 331
Abonnement d'un flux de diffusion à une rubrique ... 332
Gestion des messages sur plusieurs destinations de flux de diffusion ... 333
Exemple de cas d'utilisation de l'archivage et de l'analyse des messages ... 346
Diffusion dans les fonctions Lambda ... 358
Prérequis ... 358
Abonnement d'une fonction à une rubrique ... 359
Distribution ramifiée vers des files d'attente Amazon SQS ... 360
Abonnement d'une file d'attente à une rubrique ... 361
Automatisez la messagerie Amazon SNS vers Amazon SQS avec AWS CloudFormation ... 369
Notifications de fanout envoyées aux points de terminaison HTTPS ... 376
Abonnement d'un point de terminaison à une rubrique ... 378
Vérification des signatures de message ... 388
Analyse des formats des messages ... 393
Événements Fanout pour AWS Event Fork Pipelines ... 403
Comment fonctionne AWS Event Fork Pipelines ... 404
Déploiement de pipelines AWS Event Fork ... 407
Déploiement et test de l'exemple d'application Event Fork Pipelines ... 408
Abonnement d'un pipeline d'événements à une rubrique ... 416
Utilisation du EventBridge planificateur ... 426
Configuration du rôle d'exécution ... 426
Créer une planification ... 426
Ressources connexes ... 431
Application-to-person messagerie ... 432
Messagerie texte mobile ... 432
Comment Amazon SNS livre-t-il mes SMS ? ... 433
Premiers pas ... 434

Identités d'origine ... 441
Configurations ... 442
Bonnes pratiques en matière de messagerie SMS ... 522
Envoi de notifications push mobile ... 538
Comment fonctionnent les notifications utilisateur Amazon SNS ... 538
Configuration des notifications push avec Amazon SNS ... 539
Configuration d'une application mobile ... 540
Utilisation d'Amazon SNS pour les notifications push mobiles ... 560
Attributs des applications mobile ... 575
Évènements d'application mobile ... 579
Actions d'API push mobile ... 582
Erreurs courantes de l'API push mobile ... 584
Durée de vie push mobile ... 596
Régions prises en charge ... 598
Bonnes pratiques en matière de notifications push mobiles ... 599
Configuration et gestion des abonnements aux e-mails ... 600
AWS Management Console ... 601
AWS SDKs ... 602
Exemples de code ... 634
Principes de base ... 648
Hello Amazon SNS ... 648
Actions ... 661
Scénarios ... 839
Créer une application pour soumettre des données à une table DynamoDB ... 840
Création d’une application Amazon SNS ... 841
Créer un point de terminaison de plateforme pour les notifications push ... 843
Création d’une application sans serveur pour gérer des photos ... 845
Créer une application Amazon Textract Explorer ... 850
Créer et publier dans une rubrique FIFO ... 851
Détecter des personnes et des objets dans une vidéo ... 864
Publier des messages SMS dans une rubrique ... 865
Publier un message volumineux ... 872
Publier un message texte SMS ... 875
Publier des messages dans des files d'attente ... 883
Utiliser API Gateway pour invoquer une fonction Lambda ... 999
Utilisent des événements planifiés pour invoquer une fonction Lambda ... 1001

Exemples sans serveur ... 1003
Invocation d’une fonction lambda à partir d’un déclencheur Amazon SNS ... 1003
Sécurité ... 1014
Chiffrement des données ... 1015
Sécurisation des données grâce au chiffrement côté serveur ... 1015
Gestion des clés ... 1018
Configuration du chiffrement des rubriques à l'aide du chiffrement côté serveur ... 1025
Configuration du chiffrement des rubriques avec un abonnement chiffré à la file d'attente Amazon SQS ... 1027
Sécurisation du trafic avec des points de terminaison VPC ... 1033
Création d'un point de terminaison d'un VPC ... 1034
Création d'une politique de VPC ... 1036
Publication d'un message à partir d'un VPC ... 1037
Utilisation de points de terminaison à double pile pour la connectivité ... 1046
Sécurité Message Data Protection ... 1047
Gestion des identités et des accès ... 1047
Public ciblé ... 1048
Authentification par des identités ... 1048
Gestion des accès à l’aide de politiques ... 1052
Contrôle d’accès ... 1056
Cas d'utilisation du contrôle d'accès ... 1056
Concepts clés de la politique d'accès ... 1056
Présentation de l'architecture ... 1060
Utilisation du langage de la politique d'accès ... 1061
Logique d'évaluation ... 1062
Cas d'exemple pour le contrôle d'accès Amazon SNS ... 1064
Comment Amazon SNS fonctionne avec IAM ... 1076
AWS politiques gérées ... 1077
Actions de politique ... 1083
Ressources de politique ... 1084
Clés de condition de politique ... 1085
ACLs ... 1086
ABAC ... 1086
Informations d’identification temporaires ... 1087
Autorisations de principal ... 1087
Rôles de service ... 1088

Rôles liés à un service ... 1088
Exemples de politiques basées sur l'identité ... 1089
Politiques basées sur l'identité ... 1093
Politiques basées sur les ressources ... 1094
Utilisation de politiques basées sur l'identité ... 1094
Gestion des politiques IAM personnalisées ... 1101
Utilisation d'informations d'identification temporaires ... 1102
Référence des autorisations d'API ... 1104
Journalisation et surveillance ... 1108
CloudTrail journaux ... 1109
Surveillance des sujets à l'aide de CloudWatch ... 1119
Validation de conformité ... 1137
Résilience ... 1138
Sécurité de l'infrastructure ... 1139
Bonnes pratiques de sécurité ... 1140
Bonnes pratiques en matière de prévention ... 1140
Rubriques de résolution des problèmes utilisant AWS X-Ray ... 1145
Suivi actif ... 1145
Autorisations ... 1146
Activation du suivi actif ... 1146
Activation du suivi actif sur une rubrique Amazon SNS à l'aide du SDK AWS ... 1147
Activation du suivi actif sur une rubrique Amazon SNS à l'aide de la CLI AWS ... 1147
Activation du suivi actif sur une rubrique Amazon SNS à l'aide de AWS CloudFormation ... 1148
Vérifier que le suivi actif est activé ... 1148
Test ... 1149
Historique de la documentation Amazon SNS ... 1150
... mclx

Qu'est-ce qu'Amazon SNS ?

Amazon Simple Notification Service (Amazon SNS) est un service entièrement géré qui fournit des messages des éditeurs (producteurs) aux abonnés (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.

Comment ça marche

Dans SNS, les éditeurs envoient des messages à un sujet, qui fait office de canal de communication. Le sujet agit comme un point d'accès logique, garantissant que les messages sont transmis à plusieurs abonnés sur différentes plateformes.

Les abonnés à une rubrique SNS peuvent recevoir des messages via différents points de terminaison, en fonction de leur cas d'utilisation, tels que :

- Amazon SQS
- Lambda
- Points de terminaison HTTP (S)
- E-mails
- Notifications push mobile
- Messages texte (SMS) mobiles
- Amazon Data Firehose
- Fournisseurs de services (par exemple, Datadog, MongoDB, Splunk)

Le SNS prend en charge la messagerie Application-to-Application (A2A) et Application-to-Person (A2P), ce qui permet d'envoyer des messages entre différentes applications ou directement vers des téléphones portables, des adresses e-mail, etc.

Accès à Amazon SNS

Vous pouvez accéder à Amazon SNS et le gérer via la console ou AWS CLI AWS SDKs, selon votre méthode d'interaction préférée. La console propose une interface graphique pour les tâches

de base, tandis que le AWS CLI et SDKs fournit des fonctionnalités avancées de configuration et d'automatisation pour les cas d'utilisation plus complexes.

- 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 est SDKs disponible dans différentes langues. Pour plus d'informations, voir SDKs et Boîtes à outils.

Scénarios Amazon SNS courants

Utilisez ces scénarios Amazon SNS courants pour mettre en œuvre des architectures évolutives axées sur les événements et garantir une communication fiable en temps réel entre les applications et les utilisateurs.

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 associer une autre instance de EC2 serveur Amazon à un entrepôt de données pour analyser 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
- Notifications Amazon SNS envoyées par Fanout aux fonctions Lambda pour un traitement automatisé
- Notifications Amazon SNS envoyées aux files d'attente Amazon SQS pour un traitement asynchrone
- Notifications Amazon SNS envoyées aux points de terminaison HTTPS
- 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 avec Amazon SNS.

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 Envoi de notifications push mobiles avec Amazon SNS.

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 la section Tarification des SMS dans le monde entier.

Fonctionnalités et fonctionnalités d'Amazon SNS

Amazon SNS propose un ensemble complet de fonctionnalités conçues pour améliorer la messagerie entre les applications et les utilisateurs. Ces fonctionnalités permettent une communication fluide, une transmission sécurisée des messages et une gestion robuste des messages, garantissant ainsi une disponibilité, une durabilité et une flexibilité élevées pour un large éventail de cas d'utilisation de la messagerie.

Application-to-application messagerie

Une pplication-to-application messagerie prend en charge les abonnés tels que les flux de diffusion, les fonctions Lambda, les files d'attente Amazon SQS, les points de terminaison et les pipelines HTTP/S Event Fork. AWS Cela permet une distribution efficace des messages dans les architectures pilotées par les événements.

Application-to-person notifications

pplication-to-personLes notifications fournissent des notifications aux utilisateurs aux abonnés, telles que les applications mobiles, les numéros de téléphone portable et les adresses e-mail.

Rubriques standard et FIFO

Les rubriques FIFO garantissent un ordre, un regroupement et une déduplication stricts des messages, ce qui permet aux FIFO et aux files d'attente standard de s'abonner pour le traitement

des messages. Les rubriques standard sont utilisées lorsque le classement des messages et les éventuelles duplications ne sont pas critiques, prenant en charge tous les protocoles de diffusion pour des cas d'utilisation plus larges.

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 et rediffusion des messages Amazon SNS pour les rubriques FIFO.

Attributs de message

Attributs de message Amazon SNSvous permet de fournir des métadonnées arbitraires concernant le 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 de plus amples informations, veuillez consulter 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 “Sécurisation des données grâce au chiffrement côté serveur” Vous pouvez également établir une connexion privée entre Amazon SNS et votre cloud privé virtuel (VPC). Pour plus d'informations, consultez. the section called “Sécurisation du trafic avec des points de terminaison VPC”

AWS services couramment utilisés avec Amazon SNS

Intégrez Amazon SNS à plusieurs AWS Cloud services pour améliorer le traitement des messages, améliorer le contrôle d'accès, permettre le traitement piloté par les événements et automatiser les ressources. Cette intégration optimise les performances, renforce la sécurité et rationalise les opérations.

Amazon CloudWatch

Amazon CloudWatch assure la surveillance et l'observabilité d'Amazon SNS, vous aidant ainsi à suivre la livraison des messages, à détecter les anomalies et à résoudre les problèmes. Grâce à CloudWatch, vous pouvez :

- Surveillez les indicateurs Amazon SNS tels que le nombre de messages publiés, délivrés ou ayant échoué selon les sujets et les abonnements.
- Configurez des CloudWatch alarmes pour déclencher des actions automatisées lorsque les métriques Amazon SNS dépassent des seuils prédéfinis, tels que des échecs de livraison élevés ou des ralentissements.
- Utilisez CloudWatch les journaux pour capturer l'état de livraison Amazon SNS des messages envoyés aux points de terminaison HTTP/S, Lambda et Amazon SQS à des fins de débogage et d'audit.

Pour de plus amples informations, veuillez consulter Surveillance des rubriques Amazon SNS à l'aide de CloudWatch.

Amazon SQS

Amazon SQS est un service de mise en file d'attente de messages entièrement géré qui permet une communication sécurisée, durable et évolutive entre les composants logiciels distribués. Il permet de dissocier l'architecture des applications en mettant les messages en mémoire tampon, en garantissant une distribution fiable et en empêchant les défaillances du système dues à la perte de messages. Amazon SQS s'intègre à Amazon SNS de la manière suivante :

- Files d'attente de lettres mortes : Amazon SNS peut acheminer les messages non distribuables vers une file d'attente de lettres mortes Amazon SQS à des fins de résolution des problèmes et de retraitement.
- Abonnements à des rubriques : vous pouvez abonner une file d'attente Amazon SQS à une rubrique Amazon SNS, ce qui permet à Amazon SNS de diffuser des messages à plusieurs clients à l'aide d'Amazon SQS.
- Support des files d'attente FIFO : les files d'attente Amazon SQS FIFO peuvent être abonnées aux rubriques FIFO d'Amazon SNS, ce qui garantit un ordre strict des messages et un traitement unique. Les files d'attente Amazon SQS standard peuvent également s'abonner aux rubriques Amazon SNS, mais ne garantissent pas la livraison ou la déduplication ordonnées des messages.

AWS CloudFormation

AWS CloudFormation automatise le provisionnement et la gestion des AWS ressources, y compris les rubriques Amazon SNS et les abonnements, en utilisant l'infrastructure en tant que code (IaC). Avec AWS CloudFormation, vous pouvez :

- Définissez les sujets, les abonnements et les autorisations Amazon SNS dans un modèle réutilisable et contrôlé par version.
- Garantissez un déploiement cohérent des ressources Amazon SNS dans plusieurs Comptes AWS régions.
- Mettez à jour ou modifiez les configurations Amazon SNS à l'aide d'ensembles de modifications sans intervention manuelle.

Pour plus d’informations, consultez le Guide de l’utilisateur AWS CloudFormation.

AWS CloudTrail

CloudTrail fournit une visibilité sur l'activité des API pour Amazon SNS, en vous aidant à surveiller et à auditer l'accès aux rubriques, abonnements et messages Amazon SNS. Avec CloudTrail, vous pouvez :

- Suivez les appels d'API passés à Amazon SNS, y compris les personnes qui ont accédé aux rubriques, aux abonnements et aux autorisations ou les ont modifiées.
- Détectez les activités non autorisées ou inattendues en analysant les journaux à des fins de sécurité et de conformité.
- Intégrez Amazon CloudWatch ou créez AWS Security Hub des alertes basées sur des actions Amazon SNS inhabituelles.

Pour de plus amples informations, veuillez consulter Journalisation des appels d'API AWS SNS à l'aide de AWS CloudTrail.

AWS Lambda

AWS Lambda est un service de calcul sans serveur qui exécute automatiquement votre code en réponse à des événements, éliminant ainsi le besoin de provisionner ou de gérer des serveurs. Il vous permet de créer des applications pilotées par des événements qui évoluent automatiquement et s'exécutent dans un environnement informatique à haute disponibilité.

Amazon SNS s'intègre à Lambda en vous permettant de souscrire une fonction Lambda à une rubrique Amazon SNS. Lorsqu'une rubrique Amazon SNS reçoit un message, elle peut déclencher la fonction Lambda, permettant le traitement en temps réel, l'automatisation et l'exécution de la logique d'application. Cette intégration est couramment utilisée pour :

- Traitement piloté par les événements — Déclenchez automatiquement des fonctions en réponse aux messages Amazon SNS.
- Transformation des données : modifiez ou filtrez les messages Amazon SNS avant de les transférer vers d'autres services.
- Flux de travail automatisés : traitez les notifications relatives aux alertes des applications, à la surveillance du système ou à l'orchestration d'événements.

AWS Identity and Access Management (JE SUIS)

IAM fournit un contrôle d'accès sécurisé aux AWS ressources, vous permettant de gérer qui peut accéder à vos rubriques Amazon SNS, quelles actions ils peuvent effectuer et dans quelles conditions. Avec IAM, vous pouvez :

- Authentifiez les utilisateurs et les services avant qu'ils ne puissent interagir avec les rubriques Amazon SNS.
- Définissez des autorisations détaillées pour spécifier les sujets Amazon SNS sur lesquels les utilisateurs ou les rôles peuvent publier, s'abonner ou gérer.

- Utilisez des politiques basées sur l'identité pour appliquer les meilleures pratiques de sécurité, telles que la restriction de l'accès à des adresses IP ou à des conditions spécifiques Comptes AWS.

Pour de plus amples informations, veuillez consulter Utilisation de politiques basées sur l'identité avec Amazon SNS.

AWS Key Management Service (AWS KMS)

AWS KMS améliore la sécurité d'Amazon SNS en activant le chiffrement côté serveur (SSE) pour garantir la confidentialité des messages. Avec AWS KMS, vous pouvez :

- Chiffrez les messages Amazon SNS au repos à l' AWS aide de clés de chiffrement gérées ou gérées par le client (). CMKs
- Contrôlez l'accès aux rubriques Amazon SNS en définissant des politiques clés précises qui limitent les personnes autorisées à publier ou à s'abonner.
- Garantissez le respect des exigences réglementaires et de sécurité en auditant l'utilisation des clés via AWS CloudTrail.

Pour de plus amples informations, veuillez consulter Gestion des clés de chiffrement et des coûts Amazon SNS.

AWS X-Ray

X-Ray assure le suivi pour Amazon SNS, vous aidant à analyser et à déboguer le flux de messages via votre architecture basée sur les événements. Avec X-Ray, vous pouvez :

- Suivez la diffusion des messages Amazon SNS sur plusieurs points de terminaison Services AWS, tels que Lambda, Amazon SQS et HTTP/S.
- Identifiez les problèmes de latence en visualisant le temps nécessaire à la publication, à la livraison et au traitement des messages.
- Détectez les erreurs et réessayez dans les flux de messages Amazon SNS pour résoudre les problèmes d'échec de livraison ou de lenteur des délais de traitement.

Pour de plus amples informations, veuillez consulter Suivi actif dans Amazon SNS.

Utilisation d'Amazon SNS avec un SDK AWS

AWS des kits de développement logiciel (SDKs) sont disponibles pour de nombreux langages de programmation courants. 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 pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils AWS pour PowerShell exemples de code
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour 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 AWS SDKs.

> **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.

Création d'une rubrique Amazon SNS et publication de messages

Cette rubrique décrit les étapes de base de la gestion des ressources Amazon SNS, notamment en ce qui concerne les sujets, les abonnements et la publication de messages. Tout d'abord, vous allez configurer les autorisations d'accès nécessaires pour Amazon SNS, en vous assurant de disposer des autorisations appropriées pour créer et gérer les ressources Amazon SNS. Vous allez ensuite créer une nouvelle rubrique Amazon SNS, qui servira de plateforme centrale pour la gestion et la distribution des messages aux abonnés. Après avoir créé le sujet, vous allez créer un abonnement à ce sujet, permettant à des points de terminaison spécifiques de recevoir les messages publiés sur ce sujet.

Une fois le sujet et l'abonnement définis, vous publierez un message sur le sujet, en observant comment Amazon SNS transmet efficacement le message à tous les points de terminaison abonnés. Enfin, vous apprendrez comment supprimer à la fois l'abonnement et le sujet, pour terminer le cycle de vie des ressources Amazon SNS que vous avez gérées. Cette approche vous permet de bien comprendre les opérations fondamentales d'Amazon SNS et d'acquérir les compétences pratiques nécessaires pour gérer les flux de travail de messagerie à l'aide de la console Amazon SNS.

Configuration de l'accès à Amazon SNS

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

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 l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

Une partie de la procédure d'inscription consiste à recevoir un appel téléphonique ou un message texte et à saisir un code de vérification sur le clavier 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. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte 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, octroyez 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.

Connexion en tant qu'utilisateur doté 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.

Attribution d'un accès à d'autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d'autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

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

   Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l'utilisateur AWS IAM Identity Center .

Étapes suivantes

Maintenant que vous êtes prêt à travailler avec Amazon SNS, commencez par :

1. Création d'une rubrique Amazon SNS
2. Création d'un abonnement à une rubrique Amazon SNS
3. Publication d'un message Amazon SNS
4. Supprimer un sujet et un abonnement 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 pour Java, le et le AWS SDK pour .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.

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

La création d'un sujet dans Amazon SNS pose les bases de la distribution de messages, vous permettant de publier des messages qui peuvent être transmis à plusieurs abonnés. Cette étape est essentielle pour configurer le type de sujet, les paramètres de chiffrement et les politiques d'accès, afin de garantir que le sujet répond aux exigences opérationnelles, de conformité et de sécurité de l'organisation.

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 de plus amples informations, veuillez consulter Déduplication des messages Amazon SNS pour les rubriques FIFO.

5. (Facultatif) Développez la section Chiffrement et effectuez les opérations suivantes. Pour de plus amples informations, veuillez consulter Sécurisation des données Amazon SNS grâce au chiffrement côté serveur.

   a. Choisissez Activer le chiffrement.

   b. Spécifiez la AWS KMS clé. 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 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 sur la création de clés personnalisées KMSs, voir Création de clés dans le guide du AWS Key Management Service développeur

- 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 de plus amples informations, veuillez consulter 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 de plus amples informations, veuillez consulter 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 de plus amples informations, veuillez consulter 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 du sujet 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 la section Les fichiers de configuration et d'identification partagés dans le guide de référence des outils AWS SDKs et des outils.

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

.NET

SDK pour .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 pour .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 pour 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.

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

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

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

```
 SnsClient *sns.Client
}



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

 return topicArn, err
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour PHP.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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 pour Ruby.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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, reportez-vous CreateTopicà la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

        let output = try await snsClient.createTopic(
```

```
            input: CreateTopicInput(name: name)
        )

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

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

Création d'un 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, adresses e-mail et AWS ressources HTTP (S) dans d'autres domaines Comptes AWS nécessitent une confirmation de l'abonnement avant de pouvoir recevoir des messages.

Pour abonner un point de terminaison à une rubrique Amazon SNS

L'abonnement d'un point de terminaison à une rubrique Amazon SNS permet de transmettre des messages au point de terminaison spécifié, garantissant ainsi que les bons systèmes ou utilisateurs reçoivent des notifications lorsqu'un message est publié sur le sujet. Cette étape est essentielle pour relier le sujet aux consommateurs, qu'il s'agisse d'applications, de destinataires d'e-mails ou d'autres services, afin d'assurer une communication fluide entre les systèmes.

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' AWS ARN généré lorsque vous avez créé 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 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 de plus amples informations, veuillez consulter Conditions préalables à l'abonnement aux flux de diffusion Firehose aux rubriques Amazon SNS.

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

f. (Facultatif) Pour configurer une politique de filtre, développez la section Politique de filtre d'abonnement. Pour de plus amples informations, veuillez consulter 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 de plus amples informations, veuillez consulter 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 pour les lettres mortes Amazon SNS.

i. Choisissez Créer un abonnement.

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

Publication d'un message 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.

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. L'identifiant du groupe de messages est requis pour les sujets FIFO. Les messages du même groupe de messages sont remis dans l'ordre dans lequel ils sont publiés.

   c. (Facultatif) Pour un sujet standard, entrez un ID de groupe de messages. Cet identifiant est transmis à tous les abonnements standard Amazon SQS et n'est ni utilisé ni envoyé à aucun autre type de point de terminaison.

   d. 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.

e. (Facultatif) Pour notifications push mobile, saisissez une valeur de Durée de vie (TTL) en secondes. Il s'agit du délai dont dispose un service de notification push, tel que Apple Push Notification Service (APNs) ou Firebase Cloud Messaging (FCM), pour transmettre le message au terminal.

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 de plus amples informations, veuillez consulter Publication de notifications Amazon SNS 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 de plus amples informations, veuillez consulter 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 la section Les fichiers de configuration et d'identification partagés dans le guide de référence AWS SDKs et Tools.

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

.NET

SDK pour .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 pour .NET Référence de l'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.

```
//! 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 pour 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.

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

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

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



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

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

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

```
    let deduplicationId;
    let choices;

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

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

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

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

```
            }
          : {}),
      }),
    );

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

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

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

PowerShell

Outils pour PowerShell V4

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, voir Publier dans la référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

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, voir Publier dans la référence des Outils AWS pour PowerShell applets de commande (V5).

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.

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

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

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

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

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

- Pour plus de détails sur l'API, voir Publier dans le AWS SDK pour la référence de l'API Swift.

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 client étendues Amazon SNS sont compatibles avec les rubriques standard et FIFO.

Bibliothèque client étendue Amazon SNS pour Java

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 SDK, consultez la section Configurer le AWS SDK pour Java dans le Guide AWS SDK pour Java du développeur.
- 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 pour 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 de taille de message personnalisé — Les messages dont la charge utile et les attributs dépassent cette taille sont automatiquement stockés dans Amazon S3.

- **`alwaysThroughS3`**flag — Définissez cette valeur sur pour `true` forcer le stockage de toutes les charges utiles des 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 pour stocker les charges utiles des 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 sur 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 Amazon SNS pour Python

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`**— Le nom du compartiment Amazon S3 qui stockera les messages volumineux.
- **`use_legacy_attribute`**— Dans `True` ce cas, tous les messages publiés utilisent l'attribut de message réservé Legacy (`SQSLargePayloadSize`) au lieu de l'attribut de message réservé actuel (`ExtendedPayloadSize`).
- **`message_size_threshold`** – Seuil de stockage du message dans le compartiment de messages volumineux. Ne peut pas être inférieur `0` ou supérieur à`262144`. L'argument par défaut est `262144`.
- **`always_through_s3`** – Si la valeur est `True`, tous les messages sont stockés dans Amazon S3. L'argument par défaut est `False`.
- **`s3_client`**— L'`client`objet Boto3 Amazon S3 à utiliser pour stocker des objets sur Amazon S3. Utilisez-le si vous souhaitez contrôler le client Amazon S3 (par exemple, configuration Amazon S3 personnalisée ou informations d'identification). La valeur par défaut est utilisée `boto3.client("s3")` lors de la première utilisation si ce n'est pas déjà le cas.

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.

- Joignez la politique à la file d'attente Amazon SQS pour recevoir le message de la rubrique Amazon SNS.
- Abonnez la file d'attente pour recevoir des messages de la rubrique.
- Publiez un message de test à l'aide du client étendu Amazon SNS, de la ressource Topic et PlatformEndpoint de la ressource.
- 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
from sns_extended_client import SNSExtendedClientSession
from json import loads

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

def allow_sns_to_write_to_sqs(topicarn, queuearn):
    policy_document = """{{
        "Version":"2012-10-17",
        "Statement":[
            {{
            "Sid":"MyPolicy",
            "Effect":"Allow",
            "Principal" : {{"AWS" : "*"}},
            "Action":"SQS:SendMessage",
            "Resource": "{}",
            "Condition":{{
                "ArnEquals":{{
                "aws:SourceArn": "{}"
                }}
            }}
            }}
        ]
        }}""".format(queuearn, topicarn)
```

```
    return policy_document

def get_msg_from_s3(body,sns_extended_client):
    """Handy Helper to fetch message from S3"""
    json_msg = loads(body)
    s3_object = sns_extended_client.s3_client.get_object(
        Bucket=json_msg[1].get("s3BucketName"), Key=json_msg[1].get("s3Key")
    )
    msg = s3_object.get("Body").read().decode()
    return msg


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

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

    # Delete the Processed Message
    sqs.delete_message(
        QueueUrl=queue_url,
        ReceiptHandle=sqs_msg['ReceiptHandle']
    )


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

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

```
)["Attributes"].get("QueueArn")

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

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

sns_extended_client.large_payload_support = s3_extended_payload_bucket

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


# Below is the example that all the messages will be sent to the S3 bucket
sns_extended_client.always_through_s3 = True
sns_extended_client.publish(
    TopicArn=sns_topic_arn, Message="This message should be published to S3"
)
print("\n\nPublished using SNS extended client:")
fetch_and_print_from_sqs(sqs, demo_queue_url,sns_extended_client)  # Prints message
 stored in s3

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

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

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


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

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

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

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

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

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

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

platform_endpoint.always_through_s3 = True
# Can Set custom S3 Keys to be used to store objects in S3
platform_endpoint.publish(
```

```
    Message="This message should be published to S3 using the PlatformEndpoint
 resource",
    MessageAttributes={
        "S3Key": {
            "DataType": "String",
            "StringValue": "247c11c4-a22c-42e4-a6a2-9b5af5b76587",
        }
    },
)
print("\nPublished using PlatformEndpoint Resource:")
fetch_and_print_from_sqs(sqs, demo_queue_url,platform_endpoint)
```

Sortie

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

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

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

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

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 du AWS Management Console ou du AWS SDK pour Java, consultez le Pour publier des messages dans une rubrique Amazon SNS à l'aide de la AWS Management Console didacticiel.

> **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. 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 types de données de type chaîne, l'attribut value doit respecter les mêmes restrictions de contenu que le corps du message. Toutefois, si l'attribut de message est utilisé pour le filtrage, la valeur doit être une chaîne JSON valide pour garantir la compatibilité avec les politiques de filtrage des abonnements Amazon SNS. 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 de code, consultez http://en.wikipedia. org/wiki/ASCII#ASCII_printable_characters.

  > **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 AWS Lambda abonnements. 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 AWS Lambda abonnements. 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 la section Envoi de demandes de notification APNs 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 globale totale de tous les messages que vous envoyez dans le cadre d'une seule demande d'`PublishBatch`API ne peut pas dépasser 262 144 octets (256 KiB).
> 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 software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishBatchRequest;
```

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

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

// Code
private static final int MAX_BATCH_SIZE = 10;

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

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

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

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

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

        // Handle failed messages
```

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

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

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

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

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

// Code
private static final int MAX_BATCH_SIZE = 10;

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

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

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

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

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

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

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 éditeur tente de publier un message sur le sujet supprimé, il recevra un message d'erreur indiquant que le sujet 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.

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

La suppression d'une rubrique ou d'un abonnement Amazon SNS garantit une gestion efficace des ressources, empêche toute utilisation inutile et permet d'organiser la console Amazon SNS. Cette étape permet d'éviter les coûts potentiels liés aux ressources inutilisées et de rationaliser l'administration en supprimant les sujets ou les abonnements qui ne sont plus nécessaires.

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 la section Les fichiers de configuration et d'identification partagés dans le guide de référence des outils AWS SDKs et des outils.

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

.NET

SDK pour .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 pour .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 pour 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.

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

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

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

```
}



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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour 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.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

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

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

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.

Stratégies de classement et de déduplication des messages à l'aide des rubriques FIFO d'Amazon SNS

Cette rubrique fournit des informations sur les caractéristiques et fonctionnalités des rubriques Amazon SNS FIFO (First-In-First-Out) et sur la manière dont elles s'intègrent aux files d'attente FIFO Amazon SQS. Vous apprendrez à utiliser ces services ensemble pour garantir un tri et une déduplication stricts des messages, essentiels pour les applications nécessitant une cohérence des données. Ce contenu couvre les cas d'utilisation spécifiques dans lesquels les rubriques FIFO d'Amazon SNS sont utiles, et fournit un aperçu des scénarios dans lesquels l'ordre et l'unicité des messages sont essentiels.

Vous découvrirez également les détails techniques de l'ordre des messages, du regroupement des messages et de la manière dont ceux-ci affectent la livraison des messages. La rubrique sur la déduplication des messages explique les mécanismes qui empêchent les messages dupliqués, en veillant à ce que chaque message ne soit traité qu'une seule fois. En outre, vous découvrirez le filtrage des messages, la sécurité et la durabilité, qui sont essentiels pour préserver l'intégrité et la fiabilité de votre système de messagerie. Ce contenu inclut également des informations sur l'archivage et la rediffusion des messages, proposant des stratégies pour gérer l'historique des messages. Des exemples de code pratiques sont également fournis pour vous aider à implémenter ces fonctionnalités dans vos propres applications, vous permettant ainsi d'acquérir une expérience pratique des sujets liés au FIFO Amazon SNS et de leur intégration aux files d'attente FIFO Amazon SQS.

Rubriques FIFO à haut débit sur Amazon SNS

Les rubriques FIFO à haut débit d'Amazon SNS gèrent efficacement le débit élevé des messages tout en maintenant un ordre strict des messages, garantissant ainsi la fiabilité et l'évolutivité des applications traitant de nombreux messages. Cette solution est idéale pour les scénarios exigeant à la fois un débit élevé et une livraison ordonnée des messages. Pour améliorer le débit des messages à l'aide de rubriques FIFO à haut débit, il est recommandé d'augmenter le nombre de groupes de messages. Pour plus d'informations sur les quotas de messages à haut débit, consultez la section Quotas du service Amazon SNS dans le. Référence générale d'Amazon Web Services

Cas d'utilisation du haut débit pour les rubriques FIFO d'Amazon SNS

Les cas d'utilisation suivants mettent en évidence les diverses applications des sujets FIFO à haut débit, en démontrant leur efficacité dans tous les secteurs et dans tous les scénarios :

- Traitement des données en temps réel : les applications traitant des flux de données en temps réel, tels que le traitement d'événements ou l'ingestion de données de télémétrie, peuvent tirer parti des sujets FIFO à haut débit pour gérer l'afflux continu de messages tout en préservant leur ordre pour une analyse précise.
- Traitement des commandes de commerce électronique : sur les plateformes de commerce électronique où le maintien de l'ordre des transactions clients est essentiel, les sujets FIFO à haut débit garantissent que les commandes sont livrées de manière séquentielle et sans délai, même pendant les périodes de pointe des achats.
- Services financiers : les institutions financières traitant des transactions ou des transactions à haute fréquence s'appuient sur des sujets FIFO à haut débit pour traiter les données de marché et les transactions avec une latence minimale, tout en respectant des exigences réglementaires strictes en matière de commande des messages.
- Streaming multimédia : les plateformes de streaming et les services de distribution multimédia utilisent des sujets FIFO à haut débit pour gérer la diffusion des fichiers multimédia et du contenu en streaming, garantissant ainsi des expériences de lecture fluides aux utilisateurs tout en maintenant le bon ordre de diffusion du contenu

Partitions et distribution de données pour un débit élevé pour les rubriques FIFO Amazon SNS

Avec les rubriques à haut débit, Amazon SNS distribue les données des rubriques FIFO entre les partitions. Une partition est une allocation de capacité pour un sujet qui est automatiquement répliquée sur plusieurs zones de disponibilité au sein d'un Région AWS. Vous ne gérez pas les partitions. Amazon SNS gère plutôt automatiquement les partitions en votre nom, en fonction du taux d'entrée.

Pour les rubriques FIFO, Amazon SNS modifie le nombre de partitions d'une rubrique dans les situations suivantes :

- Si le taux de publication actuel approche ou dépasse ce que les partitions existantes peuvent supporter, des partitions supplémentaires sont allouées jusqu'à ce que le sujet atteigne le quota

régional. Pour plus d'informations sur les quotas, consultez la section Quotas du service Amazon SNS dans le. Référence générale d'Amazon Web Services

- Si les partitions actuelles sont peu utilisées, le nombre de partitions peut être réduit.

La gestion de la partition s'effectue automatiquement à l'arrière-plan et est transparente pour vos applications. Votre sujet et vos messages sont disponibles à tout moment.

> **Note**
>
> Une limitation temporaire de l'API de publication peut se produire si vous augmentez soudainement et de manière significative le trafic vers votre sujet tout en envoyant plusieurs fois le volume habituel. Cette limitation peut durer jusqu'à la durée de la fenêtre de déduplication, le temps que le sujet évolue pour s'adapter à l'augmentation du trafic.

Distribution des données par groupe de messages IDs

Lors de la publication d'un message dans une rubrique FIFO, Amazon SNS utilise la valeur de l'ID de groupe de messages de chaque message comme entrée pour une fonction de hachage interne. La valeur de sortie de la fonction de hachage détermine quelle partition traite le message, un ou plusieurs groupes de messages IDs peuvent être gérés par une partition donnée.

> **Note**
>
> Amazon SNS est optimisé pour une distribution uniforme des éléments entre les partitions d'une rubrique FIFO, quel que soit le nombre de partitions. AWS recommande d'utiliser un groupe de messages IDs pouvant comporter un grand nombre de valeurs distinctes.

Activez le haut débit sur votre rubrique Amazon SNS FIFO

Par défaut, les rubriques Amazon SNS FIFO sont configurées pour la déduplication au niveau des sujets. Cette opération est contrôlée par l'attribut de rubrique `FifoThroughputScope`défini sur `Topic` et comporte des quotas de débit plus restreints. Consultez les quotas de service Amazon SNS dans le. Référence générale d'Amazon Web Services

Pour activer le haut débit pour votre rubrique Amazon SNS FIFO, mettez à jour `FifoThroughputScope` l'attribut sur. `MessageGroup` Cette modification peut être effectuée via la

console ou à l'aide du SDK AWS CLI et peut également être définie lors de la création de la rubrique, ce que recommande Amazon SNS pour optimiser l'expérience client et réduire les risques que votre sujet soit limité.

> **Important**
>
> Une fois que vous avez activé l'option « `FifoThroughputScope` to » d'un sujet`MessageGroup`, il ne peut pas être rétabli au `Topic` débit.

Activer le mode haut débit pour toute file d'attente FIFO Amazon SQS abonnée

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

Exemple de cas d'utilisation de la rubrique Amazon SNS 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 de commande des messages Amazon SNS pour les rubriques FIFO. Pour des extraits de code qui implémentent ce cas d'utilisation, consultez Exemples de code Amazon SNS pour les rubriques FIFO.

Détails de commande des messages Amazon SNS 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 Exemple de cas d'utilisation de la rubrique Amazon SNS 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 Amazon SNS pour les rubriques FIFO.

Regroupement de messages Amazon SNS 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 aucune limite au nombre de groupes IDs dans les rubriques SNS FIFO ou dans les files d'attente FIFO SQS. 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 Livraison de messages Amazon SNS pour les rubriques FIFO et Filtrage des messages Amazon SNS 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 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 m3. 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.

Répartition des données par groupe de messages IDs 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 de groupes IDs 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 groupes de messages IDs, quel que soit le nombre de groupes. AWS recommande d'utiliser un grand nombre de groupes de messages distincts IDs 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.

Livraison de messages Amazon SNS 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 plus grand effort de commande et de at-least-once livraison, l'abonnement des files d'attente standard Amazon SQS aux rubriques FIFO d'Amazon SNS permet de réduire les coûts, en plus de partager les files d'attente entre les charges de travail qui n'utilisent pas le FIFO.

> **Note**
>
> Pour répartir les messages des rubriques FIFO d'Amazon SNS vers les AWS Lambda fonctions, 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 Amazon SNS 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 Amazon SNS 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 des messages Amazon SNS pour les rubriques FIFO.

Déduplication des messages Amazon SNS 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'intégralité d'une rubrique Amazon SNS FIFO lorsque l'`FifoThroughputScope`attribut topic est défini sur. `Topic` Lorsque l'attribut topic `FifoThroughputScope` est défini sur`MessageGroup`, la déduplication des messages s'applique à chaque 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 sur une rubrique Amazon SNS FIFO, tout message publié avec le même ID de déduplication, dans l'intervalle de déduplication de cinq minutes, est accepté mais n'est pas livré. La rubrique Amazon SNS FIFO continue de suivre l'ID de déduplication du message, dans le périmètre de déduplication configuré par l'attribut topic`FifoThroughputScope`, 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 de plus amples informations, veuillez consulter Sécurité des messages Amazon SNS pour les rubriques FIFO.

Sécurité des messages Amazon SNS pour les rubriques FIFO

Vous pouvez activer le chiffrement pour les rubriques Amazon SNS FIFO et les files d'attente Amazon SQS FIFO en utilisant AWS Key Management Service () les clés principales du client ()AWS KMS. CMKs

- Vous pouvez créer de nouveaux sujets et files d'attente FIFO chiffrés ou activer le chiffrement pour ceux qui existent déjà.
- Seul le corps du message est crypté. Les attributs des messages, les métadonnées des ressources et les mesures des ressources ne sont pas chiffrés.

> **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 Amazon SNS 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 maintient votre messagerie au sein de l' AWS infrastructure et du réseau, ce qui améliore la sécurité globale de votre application. Lorsque vous l'utilisez AWS PrivateLink, il n'est pas nécessaire de configurer une passerelle Internet, une traduction d'adresses réseau (NAT) ou un réseau privé virtuel (VPN). Pour plus d'informations, consultez Sécurisation du trafic Amazon SNS avec des points de terminaison VPC 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 de plus amples informations, veuillez consulter Durabilité des messages Amazon SNS pour les rubriques FIFO.

Durabilité des messages Amazon SNS 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 pour les lettres mortes Amazon SNS l'article Designing durable serverless apps with DLQs for Amazon SNS, Amazon SQS AWS Lambda, sur le blog Compute.AWS

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 et rediffusion des messages Amazon SNS pour les rubriques FIFO.

Archivage et rediffusion des messages Amazon SNS pour les rubriques FIFO

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

Amazon SNS fournit une fonctionnalité d'archivage et de rediffusion des messages sans code, spécialement conçue pour les sujets FIFO (premier entré, premier sorti). Cette fonctionnalité permet aux propriétaires de sujets de stocker des messages directement dans les archives des sujets

pendant 365 jours au maximum et de les rediffuser aux abonnés en cas de besoin. L'archivage et la relecture des messages sont essentiels pour récupérer les messages perdus et synchroniser les applications entre les régions ou les systèmes en répliquant les états.

Cette fonctionnalité est accessible via l' AWS API, le SDK et AWS Management Console. AWS CloudFormation

Principaux cas d'utilisation

- Récupération des messages : récupérez les messages perdus en raison de défaillances d'applications en aval en les rejouant sur le terminal de l'abonné.
- Réplication d'état : répliquez l'état d'un système existant dans un nouvel environnement en rejouant les messages commençant par un horodatage spécifique.
- Correction d'erreur : renvoyez les messages manqués pendant les pannes pour vous assurer que tous les événements sont traités correctement.

Composantes de l'archivage et de la rediffusion des messages

Gérez l'archivage et la rediffusion des messages pour les sujets FIFO d'Amazon SNS, notamment en définissant des périodes de rétention, en surveillant l' CloudWatchutilisation des messages archivés, en initiant des rediffusions via des attributs d'abonnement et en comprenant les autorisations requises pour modifier et lancer des rediffusions.

Archivage des messages

- Le propriétaire de la rubrique active la fonction d'archivage et définit la période de conservation des messages, qui peut aller jusqu'à 365 jours. Pour en savoir plus, voir Archivage des messages Amazon SNS pour les propriétaires de rubriques FIFO
- CloudWatch les métriques permettent de surveiller les messages archivés.

Rediffusion du message

- Un abonné lance une rediffusion en sélectionnant la fenêtre temporelle pendant laquelle les messages doivent être retraités vers le point de terminaison abonné. Pour en savoir plus, consultez Rediffusion des messages Amazon SNS pour les abonnés à la rubrique FIFO.
- Vous gérez la rediffusion via les attributs d'abonnement à l'aide de `ReplayPolicy` cette fonctionnalité.

Autorisations pertinentes

- **`SetSubscriptionAttributes`**— Nécessaire pour configurer ou modifier les paramètres de rediffusion à l'aide de l'`ReplayPolicy`attribut d'un abonnement.
- **`Subscribe`**— Nécessaire pour souscrire un nouvel abonnement et lancer des rediffusions.
- **`GetTopicAttributes`**— Permet de consulter les propriétés du sujet, mais l'initiation de la rediffusion tourne principalement autour de la gestion des abonnements.

Archivage des messages Amazon SNS pour les propriétaires de rubriques 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.

Créez une politique d'archivage des messages à l'aide du 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**
   >
   > L'archivage et la rediffusion des messages Amazon SNS ne sont disponibles que pour les rubriques FIFO application-to-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. Sélectionnez 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 AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez la section Shared `config` and `credentials` files dans le guide de référence AWS SDKs and Tools.

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);
```

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

Pour créer une politique d'archivage à l'aide de la AWS CloudFormation section `AWS::SNS::Topic`du Guide de AWS CloudFormation l'utilisateur.

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 AWS KMS autorisations.

> **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": "*"
}
```

Surveillez les statistiques d'archivage des messages à l'aide d'Amazon CloudWatch

Vous pouvez surveiller les messages archivés à l'aide d'Amazon CloudWatch à l'aide des indicateurs suivants. Pour être informé des anomalies dans vos charges de travail et éviter tout impact, vous pouvez configurer les CloudWatch alarmes Amazon 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.
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>",
  ...
}
```

Rediffusion des messages Amazon SNS pour les abonnés à la rubrique FIFO

Amazon SNS Replay permet aux abonnés aux rubriques de récupérer et de redistribuer les messages archivés depuis le magasin de données des rubriques vers un point de terminaison abonné.

- Les messages peuvent être rejoués immédiatement après la création de l'abonnement.
- Un message rejoué conserve le même contenu et `MessageId` le même `Timestamp` que l'original.
- Le message inclut un `Replayed` attribut indiquant qu'il s'agit d'un message rejoué.
- Pour ne rejouer que des messages spécifiques, appliquez une politique de filtrage à votre abonnement.

Pour en savoir plus sur le filtrage des messages, consultez Filtrage des messages relus.

Créez une politique de rediffusion des messages à l'aide du 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 Création d'un 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. Choisir la date et l'heure de début de la réécoute : choisissez la date (YYYY/MM/DDformat) et l'heure (format hh:mm:ss sur 24 heures) à partir desquelles vous souhaitez commencer à lire les 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 réécoute : choisissez la date (YYYY/MM/DDformat) et l'heure (format hh:mm:ss sur 24 heures) auxquelles vous souhaitez arrêter de relire les 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 mesures de réexécution des messages depuis ce flux de travail à l'aide de CloudWatch, voirSurveillez les statistiques de rediffusion des messages à l'aide d'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 AWS SDK, vous devez le configurer avec vos informations d'identification. Pour plus d'informations, consultez la section Shared `config` and `credentials` files dans le guide de référence AWS SDKs and Tools.

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);
```

Comprendre le EndingPoint

Lorsque vous appliquez un `ReplayPolicy` à un abonnement Amazon SNS, la `EndingPoint` valeur est facultative. Si ce n'`EndingPoint`est pas le cas, la rediffusion démarrera à partir de la date spécifiée `StartingPoint` et se poursuivra jusqu'à ce qu'elle rattrape l'heure actuelle, y compris le traitement des messages récemment publiés. Une fois rattrapé, l'abonnement fonctionnera comme un abonnement normal et recevra les nouveaux messages au fur et à mesure de leur publication.

Si un `EndingPoint` est spécifié, le service rejouera les messages du `StartingPoint` haut vers le, `EndingPoint` puis s'arrêtera. Cette action interrompt effectivement l'abonnement. Lorsque l'abonnement est suspendu, les messages récemment publiés ne seront pas remis au point de terminaison abonné.

Pour reprendre la livraison des messages, appliquez-en un nouveau `ReplayPolicy` sans fournir de et définissez le point dans le temps souhaité `StartingPoint` à partir duquel vous souhaitez continuer à recevoir des messages. `EndingPoint` Par exemple, pour reprendre un abonnement à partir duquel une rediffusion précédente s'est terminée, réglez le nouveau `StartingPoint` sur le précédent. `EndingPoint`

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, veuillez consulter Stratégies de filtre d'abonnement Amazon SNS.

Surveillez les statistiques de rediffusion des messages à l'aide d'Amazon CloudWatch

Vous pouvez surveiller les messages en rediffusion à l'aide d'Amazon CloudWatch à l'aide des indicateurs suivants. Pour être informé des anomalies dans vos charges de travail et éviter tout impact, vous pouvez configurer les CloudWatch alarmes Amazon 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 Amazon SNS pour les rubriques FIFO

Utilisez les exemples de code suivants pour intégrer le cas d'utilisation de l'exemple de gestion des prix des pièces automobiles à une rubrique FIFO Amazon SNS et à une file d'attente FIFO Amazon SQS ou à une file d'attente standard.

Utilisation d'un AWS SDK

À l'aide d'un AWS SDK, vous créez une rubrique Amazon SNS FIFO en définissant `FifoTopic` son attribut 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 Amazon SNS pour les rubriques FIFO.

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.

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",
                "FifoThroughputScope", "MessageGroup");

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

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

            return topicArn;

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

```
    }

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

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

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

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

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

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

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

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

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

- 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 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),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


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

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

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


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

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

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

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


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

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
```

```
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- 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 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 du kit SDK AWS de l'API SAP ABAP.
  - CreateTopic
  - Publish
  - 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 lethe section called “Cas d'utilisation de la rubrique FIFO”, le point d'entrée de chaque application client est la file d'attente Amazon SQS, que la AWS Lambda fonction 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 la section Utilisation AWS Lambda avec Amazon SQS dans le manuel du AWS Lambda développeur. Pour plus d'informations sur la création de vos propres sondeurs de file d'attente, consultez les recommandations relatives aux files d'attente standard et FIFO Amazon SQS dans le guide du développeur Amazon Simple Queue Service ReceiveMessageet dans le manuel Amazon Simple Queue Service API Reference.

En utilisant AWS CloudFormation

AWS CloudFormation vous permet d'utiliser un fichier modèle pour créer et configurer un ensemble de AWS ressources en une seule unité. 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 AWS ressources à l'aide d'un AWS CloudFormation modèle, voir Get Started dans le guide de AWS CloudFormation l'utilisateur.

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 tous les attributs ou propriétés du corps du message répondent aux conditions spécifiées dans la politique de filtrage, 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.

Étendue de la politique de filtre d'abonnement Amazon SNS

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

- `MessageAttributes`— Applique la politique de filtrage aux attributs des messages (paramètre par défaut).
- `MessageBody`— Applique la politique de filtrage au corps du 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 de plus amples informations, veuillez consulter 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 politiques de filtrage Amazon SNS

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 liées à la politique de filtrage dans Amazon SNS

Lorsque vous configurez des politiques de filtrage dans Amazon SNS, vous devez tenir compte de quelques règles importantes. Ces règles permettent de garantir l'application efficace des politiques de filtrage tout en préservant les performances et la compatibilité du système.

Contraintes de politique de filtre

Lorsque vous configurez des politiques de filtrage dans Amazon SNS, suivez ces règles importantes pour vous assurer qu'elles fonctionnent efficacement tout en préservant les performances et la compatibilité du système :

- 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 filtrage — La combinaison totale de valeurs d'une politique de filtrage ne doit pas dépasser 150. Pour calculer la combinaison totale, multipliez le nombre de valeurs de chaque tableau dans la politique de filtrage.
- Limiter le nombre de clés : une politique de filtrage peut comporter un maximum de cinq clés.

Considérations supplémentaires

- 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.
- 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'empêchera 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**
>
> Lorsque vous utilisez le filtrage basé sur les attributs dans Amazon SNS, vous devez supprimer deux fois certains caractères spéciaux, notamment :
>
> - Guillemets doubles («)
> - Barres obliques inversées ()
>
> Si ces caractères ne sont pas supprimés deux fois, la politique de filtrage ne correspondra pas aux attributs d'un message publié et la notification ne sera pas envoyée.

Considérations supplémentaires

- 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`est pris en charge pour les valeurs d'attributs numériques.
- Amazon SNS ignore les attributs des messages dont le type de données est binaire.

Exemple de politique en matière de complexité :

Dans l'exemple de politique suivant, la première clé possède trois opérateurs de correspondance, la seconde un opérateur de correspondance et la troisième deux opérateurs de correspondance.

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

La combinaison totale est calculée comme le produit du nombre d'opérateurs de correspondance pour chaque clé de la politique de filtrage :

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

Contraintes de politique pour le 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.

- Amazon SNS accepte une politique de filtre imbriquée pour le filtrage basé sur la charge utile.
- Pour une politique imbriquée, seules les clés en forme de feuille sont prises en compte dans le calcul de la limite des cinq clés.

Exemple de politique en matière de limite de clés :

Dans l'exemple de stratégie suivant :

- Il existe deux clés en forme de feuille : `key_c` et`key_e`.
- `key_c`possède quatre opérateurs de correspondance avec un niveau imbriqué de trois, et **`key_e`** possède trois opérateurs de correspondance avec un niveau imbriqué de deux.

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

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

La combinaison totale est calculée comme le produit du nombre d'opérateurs de correspondance et du niveau imbriqué pour chaque clé de la politique de filtrage :

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

Consignes d'utilisation des modèles Wildcard

Amazon SQS met en œuvre des protections lorsque vous enregistrez une politique de filtre contenant des caractères génériques afin de garantir que des politiques de filtrage trop complexes ne soient pas créées, car cela aurait un impact sur les performances de votre application.

Structure du motif

Les champs contiennent un ou plusieurs modèles. L'exemple suivant montre un champ utilisant deux modèles :

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

Règles de complexité

- La complexité totale des caractères génériques dans tous les champs ne doit pas dépasser 100 points
- Maximum de 3 jokers par motif

Calcul de la complexité

- Complexité du champ = `(Sum of pattern points)` × `(Number of patterns)`
- Points du motif :

  Un joker unique : 1 point

  Plusieurs jokers : 3 points chacun

  Tout sauf : 1 point

  Voici un exemple de calcul de la complexité :

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

Logique AND/OR

Utilisez AND/OR la logique des politiques de filtrage pour faire correspondre les attributs des messages ou les propriétés du corps du message dans Amazon SNS. Cela permet un filtrage des messages plus précis et plus flexible.

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**

OR 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

Utilisez l'`exists`opérateur dans une politique de filtrage pour faire correspondre les messages entrants en fonction de la présence ou de l'absence d'une propriété spécifique.

- `exists`fonctionne uniquement sur les nœuds foliaires (attributs finaux de la structure).
- Elle ne s'applique pas aux nœuds intermédiaires au sein d'une structure JSON imbriquée.

- 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

Filtrez les messages en faisant correspondre les valeurs numériques aux valeurs des attributs du message ou aux valeurs des propriétés du corps du 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.

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

Filtrez les messages en faisant correspondre les valeurs de chaîne aux valeurs des attributs du message ou aux valeurs des propriétés du corps du message. Les valeurs de chaîne sont placées entre guillemets doubles dans la politique JSON. Vous pouvez utiliser les opérations de chaîne suivantes pour faire correspondre les attributs ou les propriétés du corps du message :

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. Pour les attributs `String.Array` de type, chaque élément du tableau est traité comme une chaîne distincte à des fins de correspondance.

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"}
```

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

Elle correspond également aux corps de message suivants :

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

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

Toutefois, il ne correspond pas aux attributs de message suivants :

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

```
"customer_interests": {"Type": "String.Array", "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`. Pour les attributs `String.Array` de type, cela correspond si aucun des éléments du tableau n'est répertorié dans la propriété policy.

Examinons la propriété de politique suivante :

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

Il correspond à l'un des attributs de message suivants :

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

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

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

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"]
}
```

Toutefois, il ne correspond pas aux attributs de message suivants :

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

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

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

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

Utilisation d'un préfixe avec **`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"
}
```

tout sauf un joker

La propriété de politique suivante refuse le `*ball` caractère générique :

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

Il correspond aux attributs suivants :

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

Toutefois, il ne correspond pas à l'attribut de message suivant :

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

tout sauf un suffixe

La propriété de politique suivante refuse `-ball`

suffixe :

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

Il correspond aux attributs suivants :

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

Toutefois, il ne correspond pas à l'attribut de message suivant :

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

Equals-ignore-case correspondance

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"
}
```

Caractère générique

Vous pouvez utiliser le caractère générique (*) pour mettre en correspondance des valeurs de chaîne dans les modèles d'événements.

La politique suivante utilise le caractère générique (*) :

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

Il correspond aux attributs suivants :

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

Appliquer une politique de filtrage des abonnements dans Amazon SNS

Le filtrage des messages dans Amazon SNS vous permet de distribuer des messages de manière sélective aux abonnés en fonction des politiques de filtrage. Ces politiques définissent les conditions que les messages doivent remplir pour être livrés dans le cadre d'un abonnement. Bien que la livraison de messages bruts soit une option susceptible d'affecter le traitement des messages, elle n'est pas obligatoire pour que les filtres d'abonnement fonctionnent.

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.

Activation de la livraison de messages bruts

La livraison de messages bruts garantit que les charges utiles des messages sont livrées telles quelles aux abonnés, sans aucun codage ni transformation supplémentaire. Cela peut être utile

lorsque les abonnés ont besoin du format de message d'origine pour le traitement. Cependant, la livraison de messages bruts n'est pas directement liée à la fonctionnalité des filtres d'abonnement.

Appliquer des filtres d'abonnement

Pour appliquer des filtres de messages à un abonnement, vous définissez une politique de filtrage à l'aide de la syntaxe JSON. Cette politique précise les conditions qu'un message doit remplir pour être délivré dans le cadre de l'abonnement. Les filtres peuvent être basés sur les attributs du message, tels que les attributs du message, la structure du message ou même le contenu du message.

Relation entre la livraison de messages bruts et les filtres d'abonnement

Bien que l'activation de la livraison de messages bruts puisse affecter la manière dont les messages sont distribués et traités par les abonnés, elle n'est pas une condition préalable à l'utilisation de filtres d'abonnement. Toutefois, dans les scénarios où les abonnés ont besoin du format de message d'origine sans aucune modification, l'activation de la livraison des messages bruts peut être avantageuse parallèlement aux filtres d'abonnement.

Considérations relatives à un filtrage efficace

Lors de la mise en œuvre du filtrage des messages, tenez compte des exigences spécifiques de votre application et de vos abonnés. Définissez des politiques de filtrage qui correspondent exactement aux critères de diffusion des messages afin de garantir une distribution efficace et ciblée des messages.

> **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 SDKs

Les exemples de code suivants illustrent 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

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).

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).

Supprimer une politique de filtrage des abonnements dans Amazon SNS

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.

À l'aide du 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 *EXAMPLE1-23bc-4567-d890-ef12g3hij456* page Modifier, développez la section Politique de filtrage des abonnements.
5. Dans le champ Éditeur JSON, indiquez un corps JSON vide pour votre politique de filtre : `{}`.
6. Sélectionnez Enregistrer les modifications.

   Amazon SNS applique votre politique de filtrage à l'abonnement.

À l'aide du AWS CLI

Pour supprimer une politique de filtrage avec le AWS CLI, utilisez la `set-subscription-attributes`commande et fournissez un corps JSON vide pour l'`--attribute-value`argument :

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

Utilisation de l'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`.

Protection des données des messages dans Amazon SNS

Qu'est-ce que Message Data Protection ?

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, 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 ouhttps://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html.
- Anonymisation : masque ou supprime les données sensibles sans interrompre la publication ou la distribution des messages.
- Refuser — Bloquez la transmission de données entre les applications et AWS les ressources 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 d'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) par rapport à la rubrique Amazon SNS.
- DataIdentifier— Les données sensibles que la rubrique Amazon SNS doit scanner. 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 relatives à la politique de protection des données dans Amazon SNS

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.

Opération d'audit

L'opération d'audit échantillonne les messages entrants du sujet et enregistre les résultats relatifs aux données sensibles dans une AWS destination. 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 éléments suivants Services AWS dans chacun des types de destination de journal suivants :

- Amazon CloudWatch Logs (facultatif) — Le nom `LogGroup` doit figurer dans la région du sujet et commencer par /aws/vendedlogs/.
- (Facultatif) — `DeliveryStream` Ils doivent se trouver dans la région du sujet et avoir 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 logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy

Destination d'audit	Autorisation IAM
	Des actions supplémentaires sont requises pour utiliser le compartiment Amazon S3 avec le chiffrement SSE-KMS activé.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-1:123456789012: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 de journal, vous pouvez protéger les données qu'il contient en activant le chiffrement côté serveur avec des clés gérées par Amazon S3 (SSE-S3) ou le chiffrement côté serveur avec (SSE-KMS). AWS KMS keys 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 la section 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
	Espace
!	Point d'exclamation
$	Symbole dollar
%	Signe pourcentage
&	Esperluette
()	Parenthèse
+	Signe plus
,	Virgule
-	Trait d'union
.	Période
/\	Barre oblique, barre oblique inverse
#	Signe numérique

Caractère de masque pris en charge	Nom
:	Deux-points
;	Point-virgule
=, <>	Égal, inférieur ou supérieur à
@	Signe arobase
[]	Crochets
^	Symbole caret
_	Trait de soulignement
`	Accent grave
|	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 Amazon SNS

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.

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 et Amazon S3. https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

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 AWS compte 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 dans Amazon SNS

Les politiques de protection des données vous aident à protéger les données publiées dans vos rubriques Amazon SNS en auditant, en anonymisant (masquage ou suppression) et en refusant des informations sensibles qui se déplacent entre des applications ou des Services AWS. Vous pouvez utiliser l' AWS API, AWS CLI AWS CloudFormation, ou 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 d'anonymisation ou de refus, mais seulement une instruction d'audit.

Rubriques

- Création de politiques de protection des données dans Amazon SNS à l'aide de l'API
- Création de politiques de protection des données dans Amazon SNS à l'aide de la CLI
- Création de politiques de protection des données dans Amazon SNS à l'aide de CloudFormation
- Création de politiques de protection des données dans Amazon SNS à l'aide de la console
- Création de politiques de protection des données Amazon SNS pour sécuriser les données des messages à l'aide du SDK

Création de politiques de protection des données dans Amazon SNS à l'aide de l'API

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.

Création d'une politique de protection des données à l'aide de l'API

Créez une politique de protection des données Amazon SNS à l'aide de l' AWS API.

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 dans Amazon SNS à l'aide de la 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.

Création de politiques de protection des données à l'aide du AWS CLI

Créez une politique de protection des données Amazon SNS à l'aide du. 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 dans Amazon SNS à l'aide de CloudFormation

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.

Création de politiques de protection des données (CloudFormation)

Créez 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 dans Amazon SNS à l'aide de la console

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 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 Destination de l'audit, sélectionnez celle Services AWS à laquelle vous souhaitez envoyer les résultats de l'audit et entrez un nom de destination pour chacune des destinations 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.
         - — Firehose répond aux demandes de diffusion en temps réel vers Splunk, et OpenSearch Amazon Redshift pour une analyse plus approfondie des journaux. Pour plus d'informations à ce sujet, consultez le guide de l'utilisateur.

- 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 Définir cette déclaration d'anonymisation pour, sélectionnez les AWS comptes ou les principaux IAM auxquels cette déclaration d'anonymisation s'applique. Vous pouvez l'appliquer à tous les AWS comptes, ou à des AWS comptes ou entités IAM spécifiques (racines de compte, rôles ou utilisateurs) qui utilisent un compte IDs ou une entité IAM. ARNs Séparez les multiples IDs ou ARNs utilisez 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 AWS comptes, à des entités spécifiques Comptes AWS ou IAM (par exemple, racines de comptes, rôles ou utilisateurs) qui utilisent un compte IDs ou une entité IAM. ARNs Séparez les multiples IDs ou ARNs utilisez 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 Amazon SNS pour sécuriser les données des messages à l'aide du SDK

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.

Création de politiques de protection des données à l'aide du AWS SDK

Créez une politique de protection des données Amazon SNS à l'aide du AWS SDK.

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 les politiques de protection des données Amazon SNS à l'aide de l' AWS API, AWS CLI AWS CloudFormation, ou. 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 d'Amazon SNS.

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.

Suppression des politiques de protection des données à l'aide de la console

Pour supprimer une politique de protection des données gérée à l'aide de la 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 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 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 du 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 de données Amazon SNS

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.

Utilisation des identifiants de données gérés dans Amazon SNS

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, telles que les numéros de carte de crédit, les clés d'accès AWS secrètes ou les numéros de passeport d'un pays ou d'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 :

- Informations d'identification, telles que les clés privées ou les clés d'accès AWS secrètes
- 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

Identifiant	Catégorie	Pays/Langues
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, 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

Identifiant	Catégorie	Pays/Langues
NhsNumber	Santé	Go
NieNumber	Personnel	ES
NifNumber	Personnel	ES
PassportNumber	Personnel	CA, DE, ES, FR, GB, IT, US
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

Identifiant	Catégorie
CreditCardSecurityCode	Services financiers
EmailAddress	Personnel
IpAddress	Personnel
LatLong	Personnel
Nom	Personnel
OpenSshPrivateKey	Informations d’identification
PgpPrivateKey	Informations d’identification
PkcsPrivateKey	Informations d’identification
PuttyPrivateKey	Informations d’identification
VehicleIdentificationNumber	Personnel

Types de données sensibles Amazon SNS : 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
AWS clé d'accès secrète	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

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Pays et régions
Clé privée PGP	PgpPrivateKey	Non	N'importe quel compte
Clé privée PKCS (Public Key Cryptography Standard)	PkcsPrivateKey	Non	N'importe quel compte
Clé privée PuTTY	PuttyPrivateKey	Non	N'importe quel compte

Identifiant de données ARNs pour les types de données d'identification

Vous trouverez ci-dessous la liste des Amazon Resource Names (ARNs) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

Identifiant des données d'identification ARNs
arn:aws:dataprotection : :aws:data-identifier/ AwsSecretKey
arn:aws:dataprotection : :aws:data-identifier/ OpenSshPrivateKey
arn:aws:dataprotection : :aws:data-identifier/ PgpPrivateKey
arn:aws:dataprotection : :aws:data-identifier/ PkcsPrivateKey
arn:aws:dataprotection : :aws:data-identifier/ PuttyPrivateKey

Types de données sensibles Amazon SNS : 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

Identifiant de données ARNs pour les types de données de l'appareil

Vous trouverez ci-dessous la liste des Amazon Resource Names (ARNs) 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 Amazon SNS : 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.	Cela inclut : les numéros de compte bancaire internationaux (IBANs) composés d'un maximum de 34 caractères alphanumériques, y compris des éléments tels que le code du 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 (IBANs) composés de 34 caractères alphanumériques au maximum, y compris des éléments tels que le code du 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			

Pays ou région	Mots clés			
	account id, 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 #, accountnumber#, 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 #, accountnumber#, 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			

Identifiant de données ARNs pour les types de données financières

Vous trouverez ci-dessous la liste des Amazon Resource Names (ARNs) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

Identifiant des données financières ARNs
arn:aws:dataprotection : :aws:data-identifier/ -DE BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -ES BankAccountNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR BankAccountNumber
arn:aws: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 Amazon SNS : informations de santé protégées (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

Identifiant de données ARNs pour les types de données d'informations de santé protégées (PHI)

Vous trouverez ci-dessous la liste de l'identifiant de données Amazon Resource Names (ARNs) qui peut être utilisé dans les politiques de protection des données PHI.

Identifiant de données PHI ARNs
arn:aws:dataprotection : :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:dataprotection : :aws:data-identifier/ -US HealthcareProcedureCode
arn:aws:dataprotection : :aws:data-identifier/ -EU HealthInsuranceCardNumber
arn:aws:dataprotection : :aws:data-identifier/ -US HealthInsuranceClaimNumber
arn:aws:dataprotection : :aws:data-identifier/ -FR HealthInsuranceNumber
arn:aws:dataprotection : :aws:data-identifier/ -US MedicareBeneficiaryNumber
arn:aws:dataprotection : :aws:data-identifier/ -US NationalDrugCode
arn:aws: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 Amazon SNS : informations personnelles identifiables (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	–	Brésil

Type de détection	ID d'identifiant de données géré	Mot-clé requis	Informations supplémentaires	Pays et régions
		nacional da pessoa juridica, cnpj		
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	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	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	Royaume-Uni
Identification individuelle de contribuable	IndividualTaxIdentificationNumber	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 coordonné es 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 VINs une séquence de 17 caractères et respecter 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’identit é, 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.

Identifiant de données ARNs pour les informations personnelles identifiables (PII)

Le tableau suivant répertorie les Amazon Resource Names (ARNs) pour les identifiants de données que vous pouvez ajouter à vos politiques de protection des données.

Identifiant de données PII ARNs
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection : :aws:data-identifier/ -BR CepCode

Identifiant de données PII ARNs
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

Identifiant de données PII ARNs
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

Identifiant de données PII ARNs
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

Identifiant de données PII ARNs
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

Les identificateurs de données personnalisés (CDIs) vous permettent de définir vos propres expressions régulières personnalisées qui peuvent être utilisées 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 un employé spécifique à l'entreprise. IDs Les identifiants de données personnalisés peuvent être utilisés conjointement avec des identifiants de données gérés.

En quoi consistent les identifiants de données personnalisés ?

Les identificateurs de données personnalisés (CDIs) vous permettent de définir vos propres expressions régulières personnalisées qui peuvent être utilisées 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 un employé spécifique à l'entreprise. IDs 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 indique à la rubrique Amazon SNS de détecter les charges utiles qui transportent des IDs employés spécifiques à l'entreprise, puis de IDs les masquer 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 rubrique décrit comment Amazon SNS gère la livraison des messages dans différents scénarios. Vous découvrirez la livraison de messages bruts, où Amazon SNS envoie les messages dans leur format d'origine non modifié au point de terminaison. Vous découvrirez également comment envoyer des messages depuis une rubrique Amazon SNS vers une file d'attente Amazon SQS située dans une Compte AWS autre rubrique, afin d'obtenir des informations sur la messagerie entre comptes.

Cette rubrique fournit des informations sur la distribution de messages Amazon SNS à une file d'attente Amazon SQS ou à une fonction Lambda de Régions AWS différentes manières, sur le fonctionnement de la diffusion entre régions et sur les considérations à prendre en compte.

En outre, vous apprendrez à surveiller et à interpréter l'état de livraison des messages, qui fournit des informations essentielles pour savoir si les messages ont été correctement livrés ou ont rencontré des problèmes. En cas d'échec de la livraison des messages, vous comprendrez le processus de nouvelle tentative de remise des messages, notamment la façon dont Amazon SNS tente automatiquement de les redistribuer pour s'assurer qu'ils atteignent leur destination prévue. Cette rubrique décrit également l'utilisation de files d'attente de lettres mortes pour capturer les messages qui n'ont pas pu être remis après plusieurs tentatives, ce qui vous permet d'analyser et de résoudre efficacement ces défaillances.

Remise des messages bruts Amazon SNS

Pour éviter que les points de terminaison 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 ou pour les points de terminaison Amazon SQS, toutes les métadonnées Amazon SNS sont supprimées du message publié et le message est envoyé tel quel.
- Lorsque vous activez la livraison de messages bruts pour les HTTP/S points de terminaison, l'en-tête HTTP `x-amz-sns-rawdelivery` dont la valeur `true` est définie sur est ajouté au message, indiquant que le message a été publié sans formatage JSON.
- Lorsque vous activez la livraison de messages bruts pour les HTTP/S points de terminaison, le corps du message, l'adresse IP du client et les en-têtes requis sont transmis. Lorsque vous spécifiez des attributs de message, ils ne seront pas envoyés.

- Lorsque vous activez la livraison de messages bruts pour les terminaux 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 Firehose, à Amazon SQS ou à un point de terminaison. HTTP/S
4. Sur la ***MyTopic*** page, dans la section Abonnement, choisissez un abonnement, puis sélectionnez Modifier.
5. Sur la *EXAMPLE1-23bc-4567-d890-ef12g3hij456* page Modifier, dans la section Détails, choisissez Activer la livraison de 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 Notifications Amazon SNS envoyées aux files d'attente Amazon SQS pour un traitement

asynchrone). 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.

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. Sur la *`MyTopic`* page Modifier, développez la section 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. Sélectionnez Enregistrer les modifications.

   Un utilisateur possédant les informations d'identification du compte `111122223333` peut s'abonner à MyTopic.

Étape 2 : Pour ajouter un abonnement à la file d'attente Amazon SQS à une rubrique d'une autre Compte AWS à l'aide du AWS Management Console

Avant de commencer, assurez-vous que vous disposez des informations correspondant ARNs à votre sujet et à votre file d'attente, et que vous avez autorisé le sujet 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 la file d'attente Amazon SQS à une rubrique d'une autre Compte AWS à l'aide du AWS Management Console

Avant de commencer, assurez-vous que vous disposez des informations correspondant ARNs à votre sujet et à votre file d'attente, et que vous avez autorisé le sujet 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 Abonnements, choisissez 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 du 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 la région où Amazon SNS est hébergé, dans la région correspondante. 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'activation

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
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.`*`queue-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.`*`topic-region`*`.amazonaws.com` dans le principal de service de la file d'attente
Région d'inscription vers la région d'inscription	Prise en charge avec `sns.`*`queue-region`*`.amazonaws.com` dans le principal de service de la file d'attente

Voici un exemple de déclaration de politique d'accès qui autorise une rubrique Amazon SNS d'une région optionnelle (af-south-1) à 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`.

JSON

```
{
    "Version": "2008-10-17",
    "Id": "__default_policy_ID",
```

```
    "Statement": [
        {
            "Sid": "allow_sns_arn:aws:sns:af-
south-1:111111111111:source_topic_name",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.af-south-1.amazonaws.com"
            },
            "Action": "SQS:SendMessage",
            "Resource": "arn:aws:sqs:us-
west-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 des AWS Lambda fonctions 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	Prise en charge avec `sns.`*`lambda-function-region`*`.amazonaws.com` dans le principal de service de la fonction Lambda	
Région d'inscription vers région activée par défaut	Prise en charge avec `sns.`*`topic-region`*`.amazonaws.com` dans le principal de service de la fonction Lambda	

Type de livraison interrégionale	Prise en charge/Non prise en charge	
Région d'inscription vers région d'inscription	Prise en charge avec `sns.`*`lambda-function-region`*`.amazonaws.com` dans le principal de service de la fonction Lambda	

Statut de distribution de message Amazon SNS

Amazon SNS prend en charge la journalisation de l'état de livraison des messages de notification envoyés à des rubriques comportant les points de terminaison Amazon SNS suivants :

- Amazon Data Firehose
- Amazon Simple Queue Service
- AWS Lambda
- HTTPS
- Point de terminaison de l'application de plateforme

Les journaux d'état de livraison sont envoyés à Amazon CloudWatch Logs, fournissant des informations sur les opérations de livraison des messages. Ces journaux vous aident à :

- Déterminez si un message a été correctement envoyé à un point de terminaison.
- Identifiez la réponse du point de terminaison à Amazon SNS.
- Mesurez le temps d'attente des messages (temps entre l'horodatage de publication et le transfert au terminal).

Vous pouvez configurer la journalisation de l'état de livraison à l' AWS Management Console aide de l'API,, Query ou AWS CloudFormation. AWS SDKs

Conditions préalables à l'enregistrement de l'état de livraison

Cette rubrique décrit les autorisations IAM nécessaires pour permettre à Amazon SNS d'écrire des journaux de livraison et explique la convention CloudWatch de dénomination des groupes de journaux par défaut. Cela garantit que vous disposez de la configuration et de l'accès appropriés pour surveiller et analyser les journaux de distribution des messages dans CloudWatch les journaux.

Autorisations IAM requises

Le rôle IAM associé à la journalisation de l'état de livraison doit inclure les autorisations suivantes pour permettre à Amazon SNS d'écrire CloudWatch dans Logs. Vous pouvez utiliser un rôle existant avec ces autorisations ou créer un nouveau rôle lors de la configuration.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

Convention de dénomination des groupes de journaux

Par défaut, Amazon SNS crée des groupes de journaux pour les CloudWatch journaux d'état de livraison en utilisant la convention de dénomination suivante. Les flux de journaux au sein de ce groupe correspondent aux protocoles des points de terminaison (par exemple, Lambda, Amazon SQS). Vérifiez que vous êtes autorisé à consulter ces journaux dans la console CloudWatch Logs.

```
sns/<region>/<account-id>/<topic-name>
```

Configuration de le statut de distribution à l'aide de AWS Management Console

Cette rubrique explique comment activer la journalisation de l'état de livraison des messages pour les rubriques Amazon SNS, notamment la configuration des paramètres de journalisation, l'attribution de rôles IAM et la vérification que les CloudWatch journaux capturent les journaux de livraison à des fins de surveillance et de résolution des problèmes.

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques.
3. Sélectionnez le sujet souhaité, puis choisissez Modifier.
4. Développez la section Enregistrement de l'état de livraison.
5. Choisissez le protocole pour lequel vous souhaitez activer la journalisation (par exemple, HTTP, Lambda, Amazon SQS).
6. Entrez le taux d'échantillonnage de réussite, qui est le pourcentage de messages réussis pour lesquels vous souhaitez recevoir des CloudWatch journaux.
7. Dans la section Rôles IAM, vous devez configurer les rôles pour la journalisation des réussites et des échecs :
   - Utiliser un rôle de service existant : sélectionnez un rôle IAM existant disposant des autorisations requises pour qu'Amazon SNS puisse y écrire des journaux. CloudWatch
   - Créer un nouveau rôle de service : choisissez Créer de nouveaux rôles pour définir les rôles IAM pour les livraisons réussies et échouées dans la console IAM. Pour plus de détails sur les autorisations, voirConditions préalables à l'enregistrement de l'état de livraison.
8. Sélectionnez Enregistrer les modifications.

   Après avoir activé la journalisation, vous pouvez afficher et analyser les CloudWatch journaux contenant l'état de livraison des messages. Pour plus d'informations sur l'utilisation CloudWatch, consultez la CloudWatchdocumentation.

Vérification de la configuration du journal

1. Connectez-vous à la console CloudWatch Logs.
2. Localisez le groupe de journaux nommés`sns/<region>/<account-id>/<topic-name>`.
3. Assurez-vous que les flux de journaux existent pour le protocole de point de terminaison configuré.
4. Envoyez un message test à votre sujet et vérifiez que les entrées du journal apparaissent, indiquant les livraisons réussies ou échouées.

Configuration de l'enregistrement de l'état de livraison à l'aide du AWS SDKs

Ils AWS SDKs fournissent, APIs en plusieurs langues, des attributs de rubrique pour l'enregistrement de l'état de livraison des messages. Par exemple, utilisez l'SetTopicAttributesAPI pour configurer :

- `LambdaSuccessFeedbackRoleArn`— Rôle IAM pour une transmission réussie des messages aux points de terminaison Lambda.
- `LambdaSuccessFeedbackSampleRate`— Taux d'échantillonnage des messages envoyés avec succès aux points de terminaison Lambda.
- `LambdaFailureFeedbackRoleArn`— Rôle IAM en cas d'échec de transmission de messages aux points de terminaison Lambda.

Exemple de AWS CLI commande

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name LambdaSuccessFeedbackRoleArn \
    --attribute-value arn:aws:iam::123456789012:role/MyFeedbackRole
```

Attributs de rubrique

Utilisez les valeurs de nom d'attribut de rubrique suivantes pour l'état de livraison des messages :

HTTP

- `HTTPSuccessFeedbackRoleArn`— État de livraison du message réussi pour une rubrique Amazon SNS abonnée à un point de terminaison HTTP.
- `HTTPSuccessFeedbackSampleRate`— Pourcentage de messages réussis à échantillonner pour une rubrique Amazon SNS abonnée à un point de terminaison HTTP.
- `HTTPFailureFeedbackRoleArn`— État d'échec de livraison du message pour une rubrique Amazon SNS abonnée à un point de terminaison HTTP.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— État de livraison du message réussi pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon Data Firehose.

- `FirehoseSuccessFeedbackSampleRate`— Pourcentage de messages réussis à échantillonner pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon Data Firehose.
- `FirehoseFailureFeedbackRoleArn`— État d'échec de livraison du message pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— État de livraison du message réussi pour une rubrique Amazon SNS abonnée à un point de terminaison Lambda.
- `LambdaSuccessFeedbackSampleRate`— Pourcentage de messages réussis à échantillonner pour une rubrique Amazon SNS abonnée à un point de terminaison Lambda.
- `LambdaFailureFeedbackRoleArn`— État d'échec de livraison du message pour une rubrique Amazon SNS abonnée à un point de terminaison Lambda.

Points de terminaison des applications de la plateforme

- `ApplicationSuccessFeedbackRoleArn`— État de livraison du message réussi pour une rubrique Amazon SNS abonnée à un point de terminaison d' AWS application.
- `ApplicationSuccessFeedbackSampleRate`— Pourcentage de messages réussis à échantillonner pour une rubrique Amazon SNS abonnée à un point de terminaison d' AWS application.
- `ApplicationFailureFeedbackRoleArn`— État d'échec de livraison du message pour une rubrique Amazon SNS abonnée à un point de terminaison d' AWS application.

> **Note**
>
> En outre, vous pouvez configurer les attributs de l'application pour enregistrer l'état de livraison directement dans les 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`— État de livraison du message réussi pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon SQS.

- `SQSSuccessFeedbackSampleRate`— Pourcentage de messages réussis à échantillonner pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon SQS.
- `SQSFailureFeedbackRoleArn`— État d'échec de livraison du message pour une rubrique Amazon SNS abonnée à un point de terminaison Amazon SQS.

Les journaux des points de terminaison des applications de plate-forme sont écrits dans le même groupe de CloudWatch journaux que les autres points de terminaison.

> **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, tous les messages ayant échoué génèrent des CloudWatch journaux.

AWS Exemples de SDK pour configurer les attributs des rubriques

Les exemples de code suivants illustrent 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour 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 pour Ruby.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK pour 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.

Assurez-vous que les rôles IAM sont référencés `SuccessFeedbackRoleArn` et `FailureFeedbackRoleArn` disposent des autorisations de CloudWatch journalisation requises.

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 de plus amples informations, veuillez consulter Files d'attente pour les lettres mortes Amazon SNS.

Protocoles et politiques de distribution

> **Note**
>
> - À l'exception des politiques personnalisées prises HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S en charge. Consultez Création d'une politique HTTP/S de livraison.
> - 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 HTTP/S point de terminaison 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	[1] AWS Lambda Amazon SQS	3 fois, sans délai	2 fois, 1 seconde d'intervalle	10 fois, avec retour exponentiel, de 1 seconde à 20 secondes	100 000 fois, 20 secondes d'intervalle	100 015 fois, plus de 23 jours
Points de terminaison gérés par le client	SMTP SMS Push mobile	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

[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 a lieu immédiatement après la tentative de livraison initiale. Il n'y a aucun délai entre les relances dans cette phase.

2. Phase préalable à l'arrêt : cette phase fait suite à la phase de réessai 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 de retrait — 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 post-arrêt — Cette phase fait suite à la phase d'arrêt. 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 HTTP/S de livraison

Vous pouvez définir la manière dont Amazon SNS tente à nouveau de transmettre des messages aux HTTP/S points de terminaison à l'aide d'une politique de livraison comportant quatre phases : sans délai, avant le retrait, puis après le retrait. Cette politique vous permet de remplacer les paramètres de nouvelle tentative par défaut et de les personnaliser en fonction de la capacité de votre serveur HTTP.

Vous pouvez définir votre politique HTTP/S de livraison sous la forme d'un objet JSON au niveau du sujet ou de l'abonnement :

- Politique au niveau du sujet : s'applique à tous les HTTP/S abonnements liés au sujet. Utilisez l'action `CreateTopic`ou `SetTopicAttributes`API pour définir cette politique.
- Politique au niveau de l'abonnement — S'applique uniquement à un abonnement spécifique. Utilisez l'action `Subscribe`ou `SetSubscriptionAttributes`API pour définir cette politique.

Vous pouvez également utiliser la AWS::SNS::Subscriptionressource dans vos AWS CloudFormation modèles.

Vous devez personnaliser votre politique de livraison en fonction de la capacité de votre HTTP/S serveur :

- Serveur unique pour tous les abonnements : si tous les HTTP/S abonnements d'une rubrique utilisent le même serveur, définissez la politique de diffusion en tant qu'attribut de rubrique pour garantir la cohérence entre tous les abonnements.

- Différents serveurs pour les abonnements — Si les abonnements ciblent différents serveurs, créez une politique de livraison unique pour chaque abonnement, adaptée à la capacité du serveur spécifique.

Vous pouvez également définir l'`Content-Type`en-tête dans la politique de demande pour spécifier le type de support de la notification. Par défaut, Amazon SNS envoie toutes les notifications aux HTTP/S points de terminaison dont le type de contenu est défini sur. `text/plain; charset=UTF-8` Toutefois, vous pouvez remplacer cette valeur par défaut en utilisant le `headerContentType`champ de la politique de demande.

L'objet JSON suivant définit une politique de livraison avec des tentatives structurées en quatre phases :

1. Phase sans délai : réessayez 3 fois immédiatement.
2. Phase préalable à la pause : réessayez 2 fois avec un intervalle de 1 seconde.
3. Phase de pause : réessayez 10 fois avec des délais exponentiels allant de 1 à 60 secondes.
4. Phase post-pause : réessayez 35 fois avec un intervalle fixe de 60 secondes.

Amazon SNS effectue un total de 50 tentatives pour envoyer un message avant de le rejeter. Pour conserver les messages qui ne peuvent pas être remis après chaque nouvelle tentative, configurez votre abonnement pour déplacer les messages non remis vers une file d'attente de lettres mortes (DLQ). Pour de plus amples informations, veuillez consulter Files d'attente pour les lettres mortes Amazon SNS.

Amazon SNS considère toutes les erreurs 5XX et 429 (trop de demandes envoyées) comme réessayables. Ces erreurs sont soumises à la politique de livraison. Toutes les autres erreurs sont considérées comme des échecs permanents et aucune nouvelle tentative ne sera tentée.

> **Note**
>
> Cette politique de livraison utilise cette `maxReceivesPerSecond` propriété pour limiter le trafic de livraison à une moyenne de 10 messages par seconde et par abonnement. Bien que ce mécanisme contribue à éviter que votre HTTP/S terminal ne soit submergé par un trafic élevé, il est conçu pour maintenir un taux de livraison moyen et n'impose pas de plafond strict. Des pics occasionnels de trafic de livraison au-dessus de la limite spécifiée peuvent se produire, en particulier si votre taux de publication est nettement supérieur à la limite de limitation.

Lorsque le trafic de publication (entrant) dépasse le taux de livraison (sortant), cela peut entraîner un arriéré de messages et une latence de livraison plus élevée. Pour éviter de tels problèmes, assurez-vous que la `maxReceivesPerSecond` valeur correspond aux exigences de capacité et de charge de travail de votre HTTP/S serveur.

L'exemple de politique de livraison suivant remplace le type de contenu par défaut pour les notifications HTTP/S destinées 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 livraison est composée d'une politique de nouvelle tentative, d'une politique d'accélération et d'une politique de demande. Au total, une politique de livraison comporte 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.	Délai minimal jusqu'à 3 600 Par défaut : 20

Politique	Description	Contrainte
	Unité : secondes	
`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 moyen maximal de livraisons de messages par seconde, par abonnement.	1 ou plus Par défaut : aucune limitation (aucune limite sur le débit de livraison)

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 contrôler la fréquence des tentatives pendant la phase d'attente à l'aide de trois paramètres :

- **`minDelayTarget`**— Définit le délai de la première tentative pendant la phase d'attente.
- **`maxDelayTarget`**— Définit le délai pour la dernière tentative pendant la phase d'attente.
- **`backoffFunction`**— Détermine l'algorithme utilisé par Amazon SNS pour calculer les délais de toutes les tentatives entre la première et la dernière tentative. Vous pouvez choisir l'une des quatre fonctions de réessayage et de désactivation disponibles.

Le schéma suivant montre comment les différentes fonctions d'annulation des nouvelles tentatives affectent les délais entre les nouvelles tentatives pendant la phase d'annulation. La politique de livraison utilisée dans cet exemple inclut les paramètres suivants : 10 tentatives au total, un délai minimum de 5 secondes et un délai maximum de 260 secondes.

- L'axe vertical indique le délai (en secondes) pour chaque nouvelle tentative.
- L'axe horizontal représente la séquence de nouvelles tentatives, allant de la première à la dixième tentative.

Files d'attente pour les lettres mortes 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

plus d'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 associés au même compte et à la AWS même région.
> - 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 comme file d'attente en lettres mortes, vous devez utiliser un KMS personnalisé doté d'une politique clé qui accorde au service Amazon SNS un accès principal aux actions d'API. AWS KMS Pour plus d'informations, consultez Sécurisation des données Amazon SNS grâce au chiffrement côté serveur 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.

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 AWS gérés soutenus par Amazon SQS ou Amazon AWS Lambda SNS réessaie de livrer jusqu'à 100 015 fois, sur une période de 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 située dans la même Compte AWS 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.
- Rédigez la logique client Amazon SQS : utilisez l'API Amazon SQS AWS , le SDK AWS CLI ou écrivez une logique client personnalisée pour interroger, traiter et supprimer les messages de la file d'attente de lettres mortes.

Comment puis-je contrôler et consigner les files d'attente de lettres mortes ?

Vous pouvez utiliser CloudWatch les métriques Amazon pour surveiller les files d'attente de lettres mortes associées à vos abonnements Amazon SNS. Toutes les files d'attente Amazon SQS émettent des CloudWatch métriques à intervalles d'une minute. Pour plus d'informations, consultez la section CloudWatch Mesures disponibles pour Amazon SQS dans le manuel Amazon Simple Queue Service Developer Guide. Tous les abonnements Amazon SNS comportant des files d'attente contenant des 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 informé de l'activité dans vos files d'attente réservées aux lettres mortes, vous pouvez utiliser des CloudWatch métriques et des alarmes. La configuration d'une alarme pour la `NumberOfMessagesSent` métrique n'est pas appropriée car cette métrique ne capture pas les messages envoyés à un DLQ suite à des tentatives de traitement infructueuses. Utilisez plutôt la `ApproximateNumberOfMessagesVisible` métrique, qui capture tous les messages actuellement disponibles dans le DLQ, y compris ceux déplacés en raison d'échecs de traitement.

Exemple de configuration CloudWatch d'alarme

1. Créez une CloudWatchalarme pour la **`ApproximateNumberOfMessagesVisible`** métrique.
2. Définissez le seuil d'alarme sur 1 (ou sur une autre valeur appropriée en fonction de vos attentes et du trafic DLQ).
3. Spécifiez une rubrique Amazon SNS pour être averti 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 les journaux pour étudier les exceptions à l'origine de l'échec de toute livraison Amazon SNS et de l'envoi de messages dans des files d'attente réservées aux lettres mortes. Amazon SNS peut enregistrer à la fois les livraisons réussies et les livraisons échouées. CloudWatch Pour de plus amples informations, veuillez consulter Attributs de l'application mobile 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 plus d’informations, consultez Files d'attente pour les lettres mortes 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`.

   JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
      }
    }
  ]
}
```

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 Edit (Modifier) ***1234a567-bc89-012d-3e45-6fg7h890123i***, développez la section Redrive policy (dead-letter queue (Stratégie 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 la section Les fichiers de configuration et d'identification partagés dans le guide de référence des outils AWS SDKs 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 sur 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 plus d'informations, consultez le Guide de l'utilisateur AWS Command Line Interface .
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. 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 et rediffusion des messages Amazon SNS pour les rubriques FIFO.

Fonctionnalités	Rubriques standard	Rubriques FIFO
Archivage des messages	Streams de diffusion de Fanout to Firehose	Archivage des messages Amazon SNS pour les propriétaires de rubriques 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.	Rediffusion des messages Amazon SNS pour les abonnés à la rubrique FIFO

Gestion et optimisation des ressources dans Amazon SNS

Cette rubrique fournit des conseils sur la manière d'exploiter tout le potentiel d'Amazon SNS en garantissant des performances optimales, en réduisant les coûts inutiles et en maintenant des ressources bien organisées.

Rubriques

- Identification des rubriques Amazon SNS

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 sujets, à renforcer la sécurité de vos politiques AWS d'Identity and Access Management (IAM) et à rechercher ou filtrer facilement des milliers de sujets. Le balisage vous permet de gérer vos rubriques Amazon SNS à l'aide de Resource AWS Groups. Pour plus d'informations sur Resource Groups, consultez le Guide de l'utilisateur AWS Resource Groups.

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 balises de répartition des coûts pour organiser votre AWS facture afin de refléter votre propre structure de coûts. Pour ce faire, inscrivez-vous pour obtenir la facture de votre AWS compte incluant les clés et les valeurs des balises. 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

Ressource	Clé	Valeur
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 en fonction 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`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": [
 "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
}
```

Identification pour la recherche et le filtrage des ressources

Un AWS compte peut contenir des dizaines de milliers de rubriques Amazon SNS (voir Amazon SNS Quotas 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 rubrique explique comment configurer les balises pour une rubrique Amazon SNS à l'aide du AWS Management Console, d'un AWS SDK ou du. AWS CLI

> **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.

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 la section Les fichiers de configuration et d'identification partagés dans le guide de référence des outils AWS SDKs et des outils.

Les exemples de code suivants illustrent 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.fromEnvironment { 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

Sources et destinations des évènements Amazon SNS

Amazon SNS connecte Services AWS les systèmes externes en acheminant les notifications basées sur les événements. Amazon SNS reçoit des événements provenant de diverses sources Services AWS, tels que des mises à jour du pipeline de données, des actions de EC2 dimensionnement Amazon ou des alertes de sécurité, et publie ces événements sur les rubriques Amazon SNS. Ces rubriques envoient ensuite des notifications aux destinations désignées.

Amazon SNS prend en charge deux principaux types de destinations : Application-to-Application (A2A) et Application-to-Person (A2P). Dans la messagerie A2A, Amazon SNS peut envoyer des événements à Lambda pour déclencher une logique métier personnalisée, à Amazon SQS pour mettre les messages en file d'attente et à Amazon Data Firehose pour diffuser des données vers des services de stockage et d'analyse. Pour la messagerie A2P, Amazon SNS peut envoyer des notifications par SMS, e-mail et notifications push aux appareils mobiles, afin de garantir que les utilisateurs ou les équipes reçoivent des alertes en temps opportun.

En agissant comme un hub central, Amazon SNS achemine les notifications vers les bons endroits, vous aidant ainsi à automatiser et à gérer votre AWS infrastructure de manière plus efficace. Cette configuration permet une intégration fluide entre les services et une communication fiable avec les utilisateurs et les systèmes.

Sources des évènements Amazon SNS

Amazon SNS s'intègre à un large éventail de Services AWS catégories, ce qui permet à ces services de publier des événements sur des sujets Amazon SNS. Cette intégration fournit des notifications en temps réel des événements clés, tels que les modifications de l'infrastructure, les performances des applications et la gestion des coûts.

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

Services analytiques

Le tableau suivant décrit comment Amazon SNS s'intègre à des services d' AWS analyse tels qu'Athena et AWS Data Pipeline Amazon Redshift pour fournir des notifications en temps réel pour

les événements clés, notamment les violations des limites de contrôle, les mises à jour de l'état du pipeline et les activités d'entrepôt de données.

Vous pouvez tirer parti de ces intégrations pour automatiser les réponses et maintenir une surveillance efficace de vos opérations de données.

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'informations, 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 évènements Amazon Redshift. Pour plus d'informations, consultez Notifications d'évènements Amazon Redshift dans le Guide de gestion Amazon Redshift.

Services d'intégration d'applications

Le tableau suivant décrit comment Amazon SNS s'intègre aux services d'intégration d'applications tels que EventBridge et en permettant le routage des données et AWS Step Functions les notifications en temps réel pour les applications critiques pour l'entreprise.

Vous pouvez tirer parti de ces intégrations pour recevoir des alertes en cas d' EventBridge événements et orchestrer des flux de travail à l'aide de Step Functions, améliorant ainsi l'automatisation et la réactivité de vos 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-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 Step Functions— Vous permet de combiner des AWS Lambda fonctions et d'autres AWS services pour créer des applications critiques pour l'entreprise.	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

Le tableau suivant décrit comment AWS Billing and Cost Management s'intègre à Amazon SNS pour fournir des notifications concernant les budgets, les modifications de prix et les anomalies de coûts.

Vous pouvez tirer parti de cette intégration pour configurer les rubriques Amazon SNS afin de recevoir des alertes en temps réel concernant vos AWS dépenses, ce qui vous aide à surveiller les coûts et à répondre efficacement aux frais imprévus.

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'informations, consultez les pages suivantes dans le Guide de l'utilisateur AWS Billing : • Création d'une rubrique Amazon SNS pour les notifications de budget •

Service AWS	Avantages offerts par Amazon SNS
	• Configuration des notifications • Détecter les dépenses inhabituelles grâce à AWS Cost Anomaly Detection

Services d'applications métier

Le tableau suivant décrit comment Amazon Chime s'intègre à Amazon SNS pour envoyer des notifications lors d'événements de réunion importants, vous permettant ainsi de rester informé de vos communications et de votre planification.

Vous pouvez tirer parti de cette intégration pour utiliser les notifications d'événements du SDK Amazon Chime afin d'améliorer vos outils de collaboration au sein et en dehors de votre organisation.

Service AWS	Avantages offerts par Amazon SNS
Amazon Chime – vous permet de rencontrer, de discuter et de passer des appels professio nnels 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

Le tableau suivant décrit comment Amazon SNS s'intègre à différents services AWS informatiques, vous permettant de recevoir des notifications pour des événements clés tels que les actions Auto Scaling, les complétions d' EC2 Image Builder, les modifications de l'environnement d'Elastic Beanstalk, les sorties de la fonction Lambda et les seuils métriques de Lightsail.

Vous pouvez tirer parti de ces intégrations pour gérer efficacement vos applications et vos ressources en restant informé des mises à jour et des actions critiques. Services AWS

Service AWS	Avantages offerts par Amazon SNS
Amazon EC2 Auto Scaling : vous permet de disposer du nombre correct d'instances Amazon Elastic Compute Cloud (Amazon EC2) disponibles pour gérer la charge de votre application.	Recevez des notifications lorsqu'Auto Scaling lance ou met fin à des EC2 instances Amazon dans votre groupe Auto Scaling. Pour plus d'informations, consultez la section Recevoir des notifications Amazon SNS lorsque votre groupe Auto Scaling évolue dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
EC2 Image Builder : permet d'automatiser la création, la gestion et le déploiement d'images personnalisées, sécurisées et de up-to-date serveur préinstallées 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 la section 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 créer des sites Web AWS 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

Service AWS	Avantages offerts par Amazon SNS
	dans le Guide du développeur Amazon Lightsail.

Services de conteneurs

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS conteneur tels qu'Amazon EKS Distro et Amazon ECS, ce qui vous permet de suivre les mises à jour et les correctifs de sécurité pour les clusters Amazon EKS et de recevoir des notifications pour les nouvelles versions d'AMI optimisées pour ECS.

Vous pouvez tirer parti de ces intégrations pour garantir la sécurité et l'efficacité de vos déploiements de conteneurs en vous tenant au courant des mises à jour et des modifications importantes.

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

Le tableau suivant décrit comment Amazon SNS améliore les services d'engagement client en s'intégrant à Amazon Connect et Amazon Simple Email Service (SES) AWS End User Messaging

SMS, ce qui vous permet de recevoir des alertes et des validations, de configurer des SMS bidirectionnels et de surveiller les notifications par e-mail pour détecter les rebonds, les plaintes et les livraisons.

Ces intégrations vous aident à gérer les communications avec les clients sur plusieurs canaux.

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 The power of AWS with Amazon Connect dans le manuel Amazon Connect Administrator Guide.
AWS End User Messaging SMS— Vous aide à engager vos clients en leur envoyant des e-mails, des SMS et des messages vocaux, ainsi que des notifications push.	Configurez des SMS bidirectionnels, ce qui vous permet de recevoir des messages de vos clients. Pour plus d'informations, consultez la section Messagerie SMS bidirectionnelle dans le guide de l'AWS End User Messaging SMS utilisateur.
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'expédit eur, 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 développe ur Amazon Simple Email Service.

Services de base de données

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS base de données tels que AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon, Amazon Neptune ElastiCache, Amazon Redshift et Amazon Relational Database Service (RDS) pour envoyer des notifications concernant des événements importants tels que les migrations de données, les activités de maintenance, les mises à jour du cache et les modifications de base de données.

Ces intégrations vous aident à surveiller et à gérer plus efficacement vos environnements de base de données en fournissant des alertes en temps opportun sur les principaux événements opérationnels.

Service AWS	Avantages offerts par Amazon SNS
AWS Database Migration Service— Migre les données des bases de données locales vers le. AWS Cloud	Recevez des notifications lorsque AWS DMS des événements 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 (Memcached).
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 le dimension	Recevez des notifications pour les évènements Amazon RDS. Pour plus d'informations,

Service AWS	Avantages offerts par Amazon SNS
nement d'une base de données relationnelle dans AWS le cloud.	consultez Utilisation de la notification d'évènement Amazon RDS dans le Guide de l'utilisateur Amazon RDS.

Services et outils pour développeurs

Le tableau suivant décrit comment Amazon SNS s'intègre aux services d'outils de AWS développement, tels que,,,, AWS CodeDeploy Amazon AWS CodeBuild AWS CodeCommit CodeGuru, et permet de fournir des notifications en cas AWS CodePipeline d'événements critiques tels que les modifications du statut de construction, les mises à jour du référentiel, la progression du déploiement, les anomalies de performances et les actions du pipeline.

Ces intégrations vous aident à surveiller et à gérer efficacement vos flux de travail de développement logiciel en recevant des alertes en temps opportun sur des événements importants.

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 la rubrique Exemple : création d'un AWS CodeCommit déclencheur pour un Amazon SNS dans le guide de l'AWS CodeCommit utilisateur.
AWS CodeDeploy— Automatise les déploiements d'applications sur les EC2 instances Amazon, les instances sur site, les fonctions	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

Service AWS	Avantages offerts par Amazon SNS
Lambda sans serveur ou les services Amazon ECS.	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'informa tions, 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 logiciell es 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'utilisa teur.

Services web et mobiles frontaux

Le tableau suivant décrit comment Amazon SNS s'intègre AWS End User Messaging SMS pour améliorer l'engagement des clients en envoyant des e-mails, des SMS, des messages vocaux et des notifications push, y compris la possibilité de configurer des SMS bidirectionnels pour recevoir des messages clients.

Cette intégration vous permet d'interagir plus efficacement avec vos clients par le biais de différents canaux de communication.

Service AWS	Avantages offerts par Amazon SNS
AWS End User Messaging SMS— Vous aide à engager vos clients en leur envoyant des e-mails, des SMS et des messages vocaux, ainsi que des notifications push.	Configurez des SMS bidirectionnels, ce qui vous permet de recevoir des messages de vos clients. Pour plus d'informations, consultez la section Messagerie SMS bidirectionnelle dans le guide de l'AWS End User Messaging SMS utilisateur.

Services de développement de jeux

Le tableau suivant décrit comment Amazon SNS s'intègre aux GameLift serveurs Amazon pour fournir des notifications concernant les événements de matchmaking et de file d'attente sur les serveurs de jeux multijoueurs basés sur des sessions.

Cette intégration permet aux développeurs de jeux d'automatiser et de surveiller le déploiement, le fonctionnement et la mise à l'échelle de leurs serveurs de jeux, garantissant ainsi une expérience de jeu fluide.

Service AWS	Avantages offerts par Amazon SNS
Amazon GameLift Servers — 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 Servers 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 Servers Developer Guide.

Services d'Internet des objets (IoT)

Le tableau suivant décrit comment Amazon SNS s'intègre AWS IoT à des services AWS IoT Core tels que AWS IoT Device Defender,, et, pour fournir des AWS IoT Events notifications en cas d'événements AWS IoT Greengrass et d'alertes liés à l'IoT.

Ces intégrations vous permettent de surveiller efficacement le comportement des appareils, de recevoir des alertes en cas d'activités anormales et de gérer les appareils IoT avec des mises à jour et des actions en temps réel.

Service AWS	Avantages offerts par Amazon SNS
AWS IoT Core— Fournit les services cloud qui connectent vos appareils IoT à d'autres appareils et AWS Cloud services.	Recevez des notifications d' AWS IoT Core événements. Pour plus d'informations, consultez Création d'une règle Amazon SNS dans le Guide du développeur AWS IoT .
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'informa tions, consultez la section Comment utiliser AWS IoT Device Defender Detect dans le Guide du AWS IoT développeur.
AWS IoT Events – Vous permet de pour surveiller vos flottes d'équipements et d'apparei ls pour détecter les défaillances ou les modifications de fonctionnement, et pour déclencher des actions lorsque ces évènement s se produisent.	Recevez des notifications d' AWS IoT Events événements. Pour plus d'informations, consultez Amazon Simple Notification Service dans le Guide du développeur AWS IoT Events .
AWS IoT Greengrass— S'étend AWS aux appareils physiques afin qu'ils puissent agir localement sur les données qu'ils génèrent, tout en utilisant le cloud pour la gestion, l'analyse et le stockage durable.	Recevez des notifications d' AWS IoT Greengrass événements. Pour plus d'informa tions, consultez Connecteur SNS dans le Guide du développeur AWS IoT Greengrass Version 1 .

Services Machine Learning

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS machine learning, tels qu'Amazon, Amazon DevOps Guru, CodeGuru Amazon Lookout for Metrics, Amazon Rekognition et SageMaker Amazon AI, afin de fournir des notifications en cas d'anomalies, des informations opérationnelles et des activités d'étiquetage des données.

Ces intégrations vous permettent de surveiller les performances des applications, de recevoir des alertes en cas d'irrégularités dans les données et de rationaliser le déploiement de modèles d'apprentissage automatique grâce à des mises à jour en temps réel.

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 performances de vos applications opérationnelles.	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 grâce à PagerDuty Amazon DevOps Guru sur le blog de AWS gestion 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 AI — Permet aux data scientists et aux développeurs de créer et d'entraîner des modèles d'apprentissage automatique, 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'informations, consultez la section Création d'une tâche d'étiquetage en streaming dans le manuel Amazon SageMaker AI Developer Guide.

Services de gestion et de gouvernance

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS gestion et de gouvernance tels qu'Amazon Q Developer dans les applications de chat,,, AWS CloudFormation, CloudTrail, CloudWatch,, AWS Config AWS Control Tower AWS License Manager AWS Service Catalog AWS Systems Manager, et fournit des notifications pour des événements clés tels que les modifications de l'infrastructure, les alertes de conformité et les informations opérationnelles.

Ces intégrations vous aident à surveiller et à gérer efficacement vos AWS environnements en fournissant des alertes et des mises à jour en temps opportun aux équipes et aux systèmes concernés.

Service AWS	Avantages offerts par Amazon SNS
Amazon Q Developer dans les applications de chat — Permet aux équipes de développement logiciel DevOps et aux équipes de développe ment logiciel d'utiliser les forums de discussio n Amazon Chime et Slack pour surveiller et répondre aux événements opérationnels dans le. AWS Cloud	Envoyez des notifications aux salles de chat. Pour plus d'informations, consultez la section Configuration d'Amazon Q Developer dans les applications de chat du Guide de l'adminis trateur d'Amazon Q Developer dans les applications de chat.
AWS CloudFormation— Vous permet de créer et de provisionner des déploiements AWS d'infrastructure 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 la pile AWS 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 CloudTrail utilisateur.

Service AWS	Avantages offerts par Amazon SNS
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 AWS ressources de votre Compte AWS.	Recevez des notifications lorsque les ressource s sont mises à jour ou lorsque AWS Config év alue les règles personnalisées ou gérées par rapport à vos ressources. Pour plus d'informa tions, consultez les rubriques Notifications AWS Config envoyées à un réseau SNS et Exemples de notifications de modification d'élément de configuration dans le guide du AWS Config développeur.
AWS Control Tower— Vous permet de configurer et de gérer un AWS environnement multi-comptes 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 .
AWS License Manager— Vous aide à gérer vos licences logicielles auprès des fournisse urs de logiciels de manière centralisée dans vos environnements sur site AWS et dans 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.

Service AWS	Avantages offerts par Amazon SNS
AWS Service Catalog – Permet aux administrateurs informatiques de créer, gérer et distribuer 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 évènements 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 de visualiser 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

Le tableau suivant décrit comment Amazon SNS s'intègre à Amazon Elastic Transcoder pour envoyer des notifications lorsque le statut des tâches de transcodage multimédia change, ce qui vous permet de surveiller et de gérer efficacement la conversion des fichiers multimédias stockés dans Amazon S3 dans des formats adaptés aux appareils de lecture grand public.

Cette intégration vous aide à rationaliser les flux de traitement multimédia en fournissant des alertes en temps réel sur l'état des tâches.

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

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS migration et de transfert, tels que AWS Database Migration Service (DMS) AWS Application Discovery Service, et comment fournir des notifications pour des événements tels que la collecte de données sur le serveur AWS Snowball Edge, les activités de migration de bases de données et les tâches de transfert de données.

Ces intégrations vous aident à gérer et à surveiller efficacement vos processus de migration vers le cloud en proposant des alertes en temps réel et des mises à jour sur les tâches de migration critiques.

Service AWS	Avantages offerts par Amazon SNS
AWS Application Discovery Service— Vous aide à planifier votre migration vers le AWS Cloud en collectant des données d'utilisation et de configuration concernant vos serveurs sur site.	Recevez des notifications d'événements 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 locales vers le. AWS Cloud	Recevez des notifications lorsque AWS DMS des événements 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 Edge— 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.	Recevez des notifications pour les offres d'emploi de Snowball Edge. Pour plus d'informations, consultez la section Notifications pour les appareils Snow Family dans le guide de AWS Snowball Edge l'utilisateur.

Services de réseaux et de diffusion de contenu

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS mise en réseau et de diffusion de contenu, tels qu'Amazon API Gateway, Amazon, Elastic Load Balancing CloudFront AWS Direct Connect, Amazon Route 53 et Amazon VPC, pour envoyer des notifications pour des événements tels que les messages d'API, les alarmes CloudFront métriques, les changements d'état de connexion, les événements d'équilibrage de charge, les états de santé et les activités des points de terminaison VPC.

Ces intégrations vous aident à surveiller et à gérer votre réseau et vos opérations de diffusion de contenu en fournissant des alertes et des mises à jour en temps opportun.

Service AWS	Avantages offerts par Amazon SNS
Amazon API Gateway : vous permet de créer et de déployer votre propre REST, WebSocket APIs à n'importe quelle échelle.	Recevez des messages postés sur un point de terminaison API Gateway. Pour plus d'informa tions, consultez Tutoriel : Création d'une API REST API Gateway avec AWS intégration dans le Guide du développeur d'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 spécifiée s 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 AWS Direct Connect emplacement via un câble à fibre optique 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 — Répartit automatiq uement votre trafic entrant sur plusieurs cibles, telles que les EC2 instances Amazon, les	Recevez des notifications des alarmes que vous avez créées pour les évènements d'équilibrage de charge. Pour plus d'informa

Service AWS	Avantages offerts par Amazon SNS
conteneurs et les adresses IP, dans plusieurs zones de disponibilité.	tions, voir Créer des CloudWatch alarmes pour votre équilibreur de charge dans le Guide de l'utilisateur des équilibreurs de charge classique s.
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 sain (console) dans le Guide du développeur Amazon Route 53.
Amazon Virtual Private Cloud (Amazon VPC) : vous permet de lancer AWS des ressources dans un réseau virtuel que vous avez défini.	Recevez des notifications pour les évènements spécifiques qui se produisent sur les points de terminaison de l'interface. Pour plus d'informa tions, 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

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS sécurité, d'identité et de conformité, tels qu' AWS Directory Service Amazon, Amazon Inspector GuardDuty, et comment fournir des notifications concernant les modifications du statut des annuaires AWS Security Hub, les résultats de sécurité, les événements liés à l'inspecteur et les annonces relatives au hub de sécurité.

Ces intégrations vous aident à maintenir des pratiques de sécurité robustes en proposant des alertes et des mises à jour en temps opportun sur les événements de sécurité et de conformité.

Service AWS	Avantages offerts par Amazon SNS
AWS Directory Service— Fournit plusieurs manières 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

Service AWS	Avantages offerts par Amazon SNS
	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 EC2 instances Amazon et l'état de sécurité de vos applications qui s'exécutent sur ces instances.	Recevez des notifications pour les évènements s 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.
AWS Security Hub – Automatise les contrôles de sécurité AWS et centralise les alertes de sécurité.	Recevez des notifications concernant AWS Security Hub des annonces, notamment des notifications concernant des AWS Security Hub contrôles ou des normes ajoutés, modifiés ou retirés. Pour plus d'informations, consultez la section Abonnement aux AWS Security Hub annonces via Amazon SNS.

Services sans serveur

Le tableau suivant décrit comment Amazon SNS s'intègre à des services tels qu'Amazon DynamoDB EventBridge, Amazon et Lambda pour envoyer des notifications concernant des événements clés tels que les mises à jour de maintenance, les flux de données en temps réel et les sorties des fonctions Lambda.

Ces intégrations vous aident à surveiller et à gérer efficacement vos applications en recevant des alertes en temps opportun sur les opérations critiques et en automatisant les réponses à ces événements.

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), Services AWS 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

Le tableau suivant décrit comment Amazon SNS s'intègre aux services de AWS stockage tels qu'Amazon Elastic File System (EFS) AWS Backup, Amazon S3 Glacier, Amazon S3, et AWS Snowball Edge comment il fournit des notifications pour divers événements tels que les activités de sauvegarde, les alarmes du système de fichiers, les tâches de récupération de données, les modifications de compartiment et les opérations de transfert de données.

Ces intégrations vous aident à surveiller et à gérer efficacement vos solutions de stockage en recevant des alertes en temps opportun sur les événements critiques liés au stockage.

Service AWS	Avantages offerts par Amazon SNS
AWS Backup— Vous aide à centraliser et à automatiser la sauvegarde des données S ervices AWS dans le cloud et sur site	Recevez des notifications d' AWS Backup événements. Pour plus d'informations, consultez la section Utilisation d'Amazon SNS pour suivre les AWS Backup événements dans le Guide du AWS Backup développeur.
Amazon Elastic File System — Fournit un espace de stockage de fichiers pour vos EC2 instances Amazon.	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 Edge— Utilise des périphéri ques 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.	Recevez des notifications pour les offres d'emploi de Snowball Edge. Pour plus d'informa tions, consultez la section Notifications pour les appareils Snow Family dans le guide de AWS Snowball Edge l'utilisateur.

Sources supplémentaires des évènements

Le tableau suivant décrit comment Amazon SNS peut être utilisé pour recevoir des notifications en temps opportun concernant les mises à jour AWS quotidiennes des fonctionnalités et les modifications apportées aux plages d'adresses AWS IP, afin de vous tenir au courant des dernières versions de AWS service, des types d'instances, des points de terminaison VPC et des modifications d'adresses IP publiques.

Ces intégrations vous aident à suivre les changements up-to-date d' AWS infrastructure et à gérer efficacement vos ressources.

Source	Avantages offerts par Amazon SNS
AWS Mises à jour quotidiennes des fonctionn alités	Recevez rapidement des informations détaillée s sur les versions et les mises à jour AWS via une rubrique Amazon SNS. Ces versions incluent les Régions AWS points de terminais on Amazon VPC Services AWS intégrés à AWS Service Quotas, les types d'instances Amazon, les types d' EC2 instances Amazon SageMaker AI, les types d'instances Amazon Nimble Studio, les versions du moteur de base de données Amazon RDS et les versions Amazon MSK Apache Kafka. Services AWS Pour plus d'informations, consultez la section S'abonner aux mises à jour AWS quotidiennes des fonctionnalités via Amazon SNS sur le blog d'AWS actualités.

Source	Avantages offerts par Amazon SNS
AWS Plages d'adresses IP	Recevez des notifications concernant les modifications apportées aux plages d' AWS adresses IP via une rubrique Amazon SNS. Pour plus d'informations, consultez les notifications relatives aux plages d'adresses AWS IP dans le Référence générale d'Amazon Web Servicesblog d'AWS actualités et Abonnez-vous aux modifications d'adresses IP AWS publiques via Amazon SNS.

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 ?
- 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 sur le blog Compute AWS
- Enrichir les architectures axées sur les événements avec les pipelines AWS Event Fork sur le Compute Blog AWS

Destinations des évènements Amazon SNS

Cette rubrique répertorie toutes les destinations des événements, organisées 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 AWS services prennent uniquement en charge la réception d'événements provenant de rubriques standard du SNS. Amazon SQS prend en charge la réception d'évènements à partir de rubriques SNS standard et FIFO.

Destinations A2A

Le tableau suivant décrit comment Amazon SNS peut diffuser des événements vers diverses destinations application-to-application (A2A) telles que Lambda, Amazon SQS, les pipelines Event Fork et les points de AWS terminaison. HTTP/S

Ces intégrations vous permettent d'archiver et d'analyser des données, de déclencher une logique métier personnalisée, de faciliter l'intégration des applications et d'acheminer les événements vers des webhooks externes, améliorant ainsi l'efficacité et la flexibilité des architectures axées sur les événements.

Destination de l'événement	Avantages offerts par Amazon SNS
https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html	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 de plus amples informations, veuillez consulter Streams de diffusion de Fanout to Firehose.
AWS Lambda	Fournir des évènements aux fonctions pour déclencher l'exécution de la logique métier personnalisée. Pour de plus amples informations, veuillez consulter Notifications Amazon SNS envoyées par Fanout aux fonctions Lambda pour un traitement automatisé.
Amazon SQS	Diffuser des évènements vers des files d'attente à des fins d'intégration d'application. Pour de plus amples informations, veuillez consulter Notifications Amazon SNS envoyées

Destination de l'événement	Avantages offerts par Amazon SNS
	aux files d'attente Amazon SQS pour un traitement asynchrone.
AWS Event Fork Pipelines	Fournissez des évènements pour la sauvegarde et le stockage d'évènements, la recherche et l'analyse d'évènements ou des pipelines de rediffusion d'évènements. Pour de plus amples informations, veuillez consulter Transférez les événements Amazon SNS à Event Fork AWS Pipelines.
HTTP/S	Diffuser des évènements à des webhooks externes. Pour de plus amples informations, veuillez consulter Notifications Amazon SNS envoyées aux points de terminaison HTTPS.

Destinations A2P

Le tableau suivant décrit comment Amazon SNS envoie des notifications application-to-person (A2P) vers différentes destinations, notamment les téléphones mobiles par SMS et notifications push natives, les boîtes de réception e-mail, les forums de discussion Amazon Chime, les canaux Slack, ainsi que des informations opérationnelles destinées aux équipes de garde via. PagerDuty

Ces intégrations améliorent la communication et l'efficacité opérationnelle en permettant des alertes et des mises à jour en temps réel sur plusieurs plateformes et canaux de communication.

Destination de l'événement	Avantages offerts par Amazon SNS
SMS	Diffuser des évènements sur les téléphones mobiles sous forme de messages texte. Pour de plus amples informations, veuillez consulter Messagerie texte mobile avec Amazon SNS.
E-mails	Diffuser les évènements dans les boîtes de réception sous forme d'e-mails. Pour de plus amples informations, veuillez consulter

Destination de l'événement	Avantages offerts par Amazon SNS
	Configuration et gestion de l'abonnement aux e-mails Amazon SNS.
Point de terminaison de plateforme	Diffusez des évènements sur les téléphone s mobiles sous forme de notifications push natives. Pour de plus amples informations, veuillez consulter Envoi de notifications push mobiles avec Amazon SNS.
Amazon Q Developer dans les applications de chat	Diffusez des évènements sur les salons de chat Amazon Chime ou les chaînes Slack. Pour plus d'informations, consultez les pages suivantes du guide d'administration d'Amazon Q Developer in chat applications : • Configuration d'Amazon Q Developer dans les applications de chat avec Amazon Chime • Configuration d'Amazon Q Developer dans les applications de chat avec Slack • Utilisation d'Amazon Q Developer dans des applications de chat avec d'autres AWS services
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 de garde via PagerDuty Amazon DevOps Guru sur le blog de AWS gestion et de gouvernance.

> **Note**
>
> Vous pouvez proposer à la fois AWS des événements natifs et des événements personnalisés aux applications de chat :
>
> - AWS Événements natifs : vous pouvez utiliser Amazon Q Developer dans des applications de chat pour envoyer AWS des événements natifs, via des rubriques Amazon SNS, à Amazon Chime et Slack. L'ensemble d' AWS événements natifs pris en charge inclut des événements provenant de AWS Billing and Cost Management AWS Health AWS CloudFormation, CloudWatch,, Amazon, etc. Pour plus d'informations, consultez la section Utilisation d'Amazon Q Developer dans des applications de chat avec d'autres services dans le Guide de l'administrateur d'Amazon Q Developer dans des applications de chat.
> - 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 ?

Utilisation d'Amazon SNS pour la messagerie application-to-application

Amazon SNS simplifie la messagerie application-to-application (A2A) en séparant les éditeurs des abonnés, ce qui prend en charge les microservices, les systèmes distribués et les applications sans serveur. Les messages sont envoyés vers des rubriques Amazon SNS, où ils peuvent être filtrés et envoyés aux abonnés tels que Lambda, Amazon SQS ou les points de terminaison HTTP. Si la livraison échoue, les messages sont stockés dans une file d'attente de lettres mortes pour une analyse plus approfondie ou un retraitement.

Streams de diffusion de Fanout to Firehose

Vous pouvez abonner des flux de diffusion 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 Amazon 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 de plus amples informations, veuillez consulter Statut de distribution de message Amazon SNS.

Conditions préalables à l'abonnement aux flux de diffusion Firehose aux rubriques Amazon SNS

Pour abonner un flux de diffusion à une rubrique SNS, vous Compte AWS devez disposer des éléments suivants :

- Une rubrique SNS standard. Pour de plus amples informations, veuillez consulter Création d'une rubrique Amazon SNS.
- Un flux de diffusion Firehose. Pour plus d'informations, consultez Création d'un flux de diffusion et accordez à votre application l'accès à vos ressources Firehose dans le manuel Amazon Data Firehose Developer Guide.
- Rôle AWS Identity and Access Management (IAM) qui fait confiance au principal du service Amazon SNS et qui est autorisé à 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 :

  JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:DescribeDeliveryStream",
        "firehose:ListDeliveryStreams",
        "firehose:ListTagsForDeliveryStream",
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:us-east-1:111111111111:deliverystream/firehose-sns-
delivery-stream"
      ],
```

```
      "Effect": "Allow"
    }
  ]
}
```

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 :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Pour plus d'informations sur la création de rôles, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le 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, 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 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 de plus amples informations, veuillez consulter 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 pour les lettres mortes Amazon SNS.
7. Choisissez Créer un abonnement.

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

Gestion des messages Amazon SNS sur plusieurs destinations de flux de diffusion

les flux de diffusion vous permettent de gérer les messages Amazon SNS sur plusieurs destinations, ce qui permet l'intégration avec Amazon S3, Amazon OpenSearch Service, Amazon Redshift et les points de terminaison HTTP pour le stockage, l'indexation et l'analyse. En configurant correctement le formatage et la distribution des messages, vous pouvez stocker les notifications Amazon SNS

dans Amazon S3 pour un traitement ultérieur, analyser les données des messages structurés à l'aide d'Amazon Athena, indexer les messages pour une recherche et une visualisation en temps réel, et structurer les archives OpenSearch dans Amazon Redshift pour les requêtes avancées.

Stockage et analyse des messages Amazon SNS dans les destinations Amazon S3

Cette rubrique explique comment les flux de livraison publient des données sur Amazon Simple Storage Service (Amazon S3).

Rubriques

- Formatage des notifications Amazon SNS pour le stockage dans les destinations Amazon S3
- Analyse des messages Amazon SNS stockés dans Amazon S3 à l'aide d'Athena

Formatage des notifications Amazon SNS pour le stockage dans les destinations Amazon S3

L'exemple suivant montre une notification Amazon SNS envoyée à un compartiment Amazon Simple Storage Service (Amazon S3), avec une indentation 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 au même compartiment Amazon S3. La mise en mémoire tampon est appliquée et des sauts de ligne séparent chaque message.

```
{"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"}
```

Analyse des messages Amazon SNS stockés dans Amazon S3 à l'aide d'Athena

Cette page explique comment analyser les messages Amazon SNS envoyés via des flux de livraison 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';
```

Intégration des messages Amazon SNS aux destinations Amazon Service OpenSearch

Cette section explique comment les flux de livraison publient des données sur Amazon OpenSearch Service (OpenSearch Service).

Rubriques

- Stockage et mise en forme des notifications Amazon SNS dans OpenSearch les index des services
- Analyse des messages Amazon SNS pour OpenSearch les destinations du service

Stockage et mise en forme des notifications Amazon SNS dans OpenSearch les index des services

L'exemple suivant illustre une notification Amazon SNS envoyée à un index Amazon OpenSearch Service (OpenSearch Service) appelé. `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": "49613100963111323203250405402193283794773886550985932802.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

Analyse des messages Amazon SNS pour OpenSearch les destinations du service

Cette rubrique explique comment analyser les messages Amazon SNS envoyés via des flux de diffusion vers des destinations Amazon OpenSearch Service (OpenSearch Service).

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"
            }
          }
        }
      ]
    }
  }
}
```

Configuration de la livraison et de l'analyse des messages Amazon SNS dans les destinations Amazon Redshift

Cette rubrique explique comment répartir les notifications Amazon SNS vers un flux de diffusion, qui publie ensuite les données sur Amazon Redshift. Avec cette configuration, vous pouvez vous connecter à la base de données Amazon Redshift et utiliser un outil de requête SQL pour récupérer les messages Amazon SNS qui répondent à des critères spécifiques.

Rubriques

- Structuration des archives de messages Amazon SNS dans les tables Amazon Redshift
- Analyse des messages Amazon SNS stockés dans les destinations Amazon Redshift

Structuration des archives de messages Amazon SNS dans les tables Amazon Redshift

Pour les points de terminaison Amazon Redshift, les messages Amazon SNS sont archivés sous forme de lignes dans un tableau. Voici un exemple de la façon dont les données sont stockées :

> **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/?Action=Désinscribe&=arn:aws:sns:us-east-1:111111111111:my-topicSubscriptionArn : 326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}
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=Désinscribe&=arn:aws:	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						sns:us-east-1:1111111111:my-topic SubscriptionArn : 326deeeb-cbf4-45da-b92b-ca77a247813b	

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=Désinscribe&=arn:aws:sns:us-east-1:111111111111:my-topicSubscriptionArn : 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 Configuration de la livraison et de l'analyse des messages Amazon SNS dans les destinations Amazon Redshift.

Analyse des messages Amazon SNS stockés dans les destinations Amazon Redshift

Cette rubrique explique comment analyser les messages Amazon SNS envoyés via des flux de diffusion 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';
```

Configuration de la livraison des messages Amazon SNS vers des destinations HTTP à l'aide de

Cette rubrique explique comment les flux de diffusion publient des données sur les points de terminaison HTTP.

Rubriques

- Format de notification Amazon SNS pour la livraison vers des destinations HTTP

Format de notification Amazon SNS pour la livraison vers des destinations HTTP

Voici un exemple de corps de requête HTTP POST provenant d'Amazon SNS, envoyé via un flux de diffusion à un point de terminaison HTTP. La notification Amazon SNS est codée sous forme de 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
      }
    ]
  }
```

Archivage et analyse des messages Amazon SNS : exemple de cas d'utilisation pour les plateformes de billetterie aérienne

Cette rubrique fournit un didacticiel pour un cas d'utilisation courant de l'archivage et de 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.

1. 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.
2. Pour atteindre l'objectif de conformité en matière de conservation des données, l'entreprise abonne un flux de diffusion à une rubrique Amazon SNS existante.
3. 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 AWS ressources correspondant à ce cas d'utilisation, vous pouvez utiliser le AWS Management Console ou un AWS CloudFormation modèle.

Rubriques

- Configuration des AWS ressources initiales pour l'archivage et l'analyse des messages Amazon SNS
- Configuration d'un flux de diffusion Amazon Data Firehose pour l'archivage des messages Amazon SNS
- Abonnement du flux de diffusion Firehose à la rubrique Amazon SNS
- Tester et interroger une configuration Amazon SNS pour une gestion efficace des données
- Automatiser l'archivage des messages Amazon SNS à l'aide d'un modèle AWS CloudFormation

Configuration des AWS ressources initiales pour l'archivage et l'analyse des messages Amazon SNS

Cette rubrique décrit comment créer les ressources nécessaires à l'archivage et à l'analyse des messages, exemple 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 :

JSON

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-
east-1:123456789012:ticketTopic"
        }
      }
    }
  ]
}
```

Dans cette politique d'accès, remplacez le Compte AWS numéro (*123456789012*) par le vôtre et modifiez la AWS région (*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 Endpoint, choisissez le Amazon Resource Name (ARN) de la ticketPaymentQueuefile d'attente.

   d. Choisissez Créer un abonnement.

   e. Répétez ces étapes pour créer un deuxième abonnement à l'aide de l'ARN de la ticketFraudQueuefile d'attente.

   Pour en savoir plus sur l'abonnement aux rubriques SNS, consultez la section Création d'un 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 Configuration d'un flux de diffusion Amazon Data Firehose pour l'archivage des messages Amazon SNS.

Configuration d'un flux de diffusion Amazon Data Firehose pour l'archivage des messages Amazon SNS

Cette rubrique 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 livraison Amazon Data Firehose

1. Ouvrez la console des services Amazon Data Firehose.
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. Sur la page Configurer les paramètres, pour Autorisations, choisissez de créer un rôle AWS Identity and Access Management (IAM) avec les autorisations requises attribuées automatiquement. Sélectionnez ensuite Suivant.
8. Sur la page Vérification, choisissez Créer un flux de diffusion.
9. Sur la page des flux de diffusion d'Amazon Data Firehose, choisissez le flux de diffusion que vous venez de créer () ticketUploadStream. 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 plus d'informations sur la création de rôles IAM, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le 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 rubrique explique comment créer les ressources suivantes pour l'exemple d'utilisation de l'archivage et de l'analyse des messages :

- Rôle AWS Identity and Access Management (IAM) qui permet à l'abonnement Amazon SNS d'enregistrer des enregistrements dans le flux de diffusion.
- L'abonnement au flux de diffusion Firehose à la rubrique Amazon 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 :

    JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams",
                "firehose:ListTagsForDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
```

```
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ticketUploadStream"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Dans cette politique, remplacez le Compte AWS numéro (*123456789012*) par le vôtre et modifiez la AWS région (*us-east-1*) en conséquence.

11. Choisissez Examiner 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 plus d'informations sur la création de rôles IAM, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le 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. Sous Détails, dans Protocole, sélectionnez.
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 Tester et interroger une configuration Amazon SNS pour une gestion efficace des données.

Tester et interroger une configuration Amazon SNS pour une gestion efficace des données

Cette rubrique explique comment tester l'exemple d'utilisation de l'archivage et de l'analyse des messages 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 d'un message 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 diffusion
- AWS Identity and Access Management Rôles et politiques (IAM)

Automatiser l'archivage des messages Amazon SNS à l'aide 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
```

Notifications Amazon SNS envoyées par Fanout aux fonctions Lambda pour un traitement automatisé

Amazon SNS s'intègre à Amazon SNS AWS Lambda, ce qui vous permet de déclencher des fonctions Lambda en réponse aux 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 en tant que paramètre d'entrée et peut manipuler les informations contenues dans le message, publier le message sur 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, veuillez consulter Statut de distribution de message Amazon SNS.

Rubriques

- Conditions préalables à l'intégration d'Amazon SNS aux fonctions Lambda dans toutes les régions
- Abonnement d'une fonction Lambda à une rubrique Amazon SNS

Conditions préalables à l'intégration d'Amazon SNS aux fonctions Lambda dans toutes les régions

Pour appeler des fonctions Lambda à l'aide de notifications Amazon SNS, vous devez disposer des éléments suivants :

- Une fonction Lambda
- Une 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 livraison entre régions vers Lambda depuis une région activée par défaut vers une région optionnelle. 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 Lambda à une rubrique Amazon SNS

Cette rubrique explique comment abonner une fonction Lambda à une rubrique Amazon SNS, en permettant à la fonction d'être déclenchée par des messages publiés.

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 Create subscription (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 AWS Lambda d'Amazon SNS, y compris un didacticiel, consultez la section Utilisation AWS Lambda avec Amazon SNS.

Notifications Amazon SNS envoyées aux files d'attente Amazon SQS pour un traitement asynchrone

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, choisissez l'une des options 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 voir un AWS CloudFormation modèle qui crée une rubrique qui envoie des messages à deux files d'attente, voirAutomatisez la messagerie Amazon SNS vers Amazon SQS avec AWS CloudFormation.

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 GetQueueAttributesAPI.

Pour obtenir l'ARN de la file d'attente à partir de la console Amazon SQS

1. Connectez-vous à la console Amazon SQS AWS Management Console et ouvrez-la 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 SetQueueAttributesAPI. 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 SendMessage politique sur une file d'attente à l'aide de la console Amazon SQS

1. Connectez-vous à la console Amazon SQS AWS Management Console et ouvrez-la 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 d' MyTopic 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 ***MyTopic*** page, dans la page Abonnements, 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 publier sur la rubrique Amazon SNS et à lire/supprimer des messages 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 autoriser un autre AWS compte à accéder à un sujet ou à une file d'attente, la seule façon de le faire est d'ajouter une politique dont le principal est la personne à laquelle Compte AWS vous souhaitez 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, vous devez utiliser la seconde méthode.

Ajout d'une politique à un utilisateur ou un groupe IAM

Si vous ajoutiez la politique suivante à un utilisateur ou à un groupe IAM, vous autoriseriez cet utilisateur ou les membres de ce groupe à effectuer l'`sns:Publish`action sur le sujet MyTopic.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Si vous ajoutiez la politique suivante à un utilisateur ou à un groupe IAM, vous autoriseriez cet utilisateur ou les membres de ce groupe à effectuer les `sqs:DeleteMessage` actions `sqs:ReceiveMessage` et sur les files d'attente MyQueue 1 et MyQueue 2.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

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 une autre personne l' Compte AWS accès à une ressource de votre compte, vous accordez également aux utilisateurs IAM disposant d'un accès de niveau administrateur (accès générique) des autorisations d'accès 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 un sujet MyTopic du compte 123456789012, vous autorisez le compte 111122223333 à effectuer l'action sur ce sujet. `sns:Publish`

```
{
  "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 du compte 123456789012, vous autoriseriez le compte 111122223333 à effectuer les actions et sur cette file d'attente. `sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "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. À l'aide des informations d'identification de l'utilisateur Compte AWS ou de l'utilisateur IAM autorisé à publier sur le sujet, connectez-vous à la console Amazon SNS AWS Management Console et ouvrez-la 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. À l'aide des informations d'identification de l'utilisateur Compte AWS ou de l'utilisateur IAM autorisé à consulter les messages de la file d'attente, connectez-vous à la console Amazon SQS AWS Management Console et ouvrez-la 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.

Automatisez la messagerie Amazon SNS vers Amazon SQS avec AWS CloudFormation

AWS CloudFormation vous permet d'utiliser un fichier modèle pour créer et configurer un ensemble de AWS ressources en une seule unité. 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 AWS ressources à l'aide d'un AWS CloudFormation modèle, consultez la section Get Started du guide de AWS CloudFormation l'utilisateur.

Utilisation d'un AWS CloudFormation modèle pour configurer des sujets et des files d'attente dans un 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 depuis la page AWS CloudFormation Modèles. Sur la page des modèles, choisissez Parcourir les exemples de modèles par AWS service, puis choisissez Amazon Simple Queue Service.

My SNSTopic est configuré pour publier sur deux points de terminaison abonnés, à savoir deux files d'attente Amazon SQS MyQueue (1 MyQueue et 2). MyPublishTopicGroup est un groupe IAM dont les membres sont autorisés à publier sur My à l'SNSTopic aide de l'action API Publish ou de la commande sns-publish. Le modèle crée les utilisateurs IAM MyPublishUser MyQueueUser et leur fournit 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 au MyPublishTopicGroup afin que l'utilisateur dispose des autorisations attribuées au groupe.

Il RDMessage QueueGroup s'agit d'un groupe IAM dont les membres sont autorisés à lire et à supprimer les messages des deux files d'attente Amazon SQS à l'aide des actions et de ReceiveMessageI'DeleteMessageAPI. AddUserToMyQueueGroup MyQueueUser s'ajoute au My RDMessage QueueGroup afin que l'utilisateur dispose des autorisations attribuées au groupe. MyQueuePolicy autorise My à publier ses notifications SNSTopic aux deux files d'attente.

La liste suivante indique le contenu du AWS CloudFormation modèle.

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

Notifications Amazon SNS envoyées aux points de terminaison HTTPS

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.
> URLs Les HTTPS ne sont accessibles qu'à partir de l'action d'API Amazon `GetSubscriptionAttributes` SNS, 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.

> **Note**
>
> Amazon SNS considère toutes les erreurs 5XX et 429 (trop de demandes envoyées) comme réessayables. Ces erreurs sont soumises à la politique de livraison. Toutes les autres erreurs sont considérées comme des échecs permanents et aucune nouvelle tentative ne sera tentée.

```
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"
}
```

Abonnement d'un point de terminaison HTTPS à une rubrique Amazon SNS

Cette rubrique explique 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 : confirmez votre abonnement Amazon SNS
- Étape 4 : Facultatif - Définissez la politique de livraison pour l'abonnement Amazon SNS
- Étape 5 : Facultatif : autorisez les utilisateurs à publier sur la rubrique Amazon SNS

- Étape 6 : envoyer des messages Amazon SNS 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 de visiter le`SubscribeURL`, vous pouvez confirmer l'abonnement à l'aide de l'ConfirmSubscription`Token`action en définissant la valeur correspondante dans le `SubscriptionConfirmation` message. 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 dans 15 secondes environ. Si votre terminal ne répond pas avant l'expiration de la connexion, ou s'il renvoie un code d'état compris entre 200 et 4 xx, Amazon SNS considérera l'envoi 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 au bout de 15 secondes environ. 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 IDs messages que vous avez traités avec les messages entrants, vous pouvez déterminer s'il s'agit d'une nouvelle tentative.
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 Abonnements.
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 Fermer.

   Le numéro d'abonnement de votre nouvel abonnement s'affiche PendingConfirmation. Lorsque vous confirmez l'abonnement, le champ ID de l'abonnement affiche l'ID d'abonnement.

Étape 3 : confirmez votre abonnement Amazon SNS

Pour confirmer votre abonnement Amazon SNS, suivez ces étapes pour vous assurer que votre terminal peut bien recevoir des messages. Ce processus implique de configurer votre terminal pour gérer les messages de confirmation entrants, de récupérer l'URL de confirmation et de confirmer l'abonnement. Vous pouvez confirmer l'abonnement automatiquement ou manuellement, en fonction de votre configuration.

1. Après vous être abonné à une rubrique Amazon SNS, Amazon SNS envoie un message de confirmation à votre terminal. Ce message contient un `SubscribeURL` que vous devez utiliser pour confirmer l'abonnement.
2. Votre point de terminaison doit être configuré pour écouter les messages entrants provenant d'Amazon SNS. Lorsque le message de confirmation arrive, **`SubscribeURL`**extrayez-le du message.
3. Une fois que vous avez obtenu le`SubscribeURL`, vous pouvez confirmer l'abonnement de deux manières :

   - Confirmation automatique — Votre terminal peut confirmer automatiquement l'abonnement en envoyant une requête HTTP GET au`SubscribeURL`.

     Cette méthode ne nécessite aucune intervention manuelle.

- Confirmation manuelle — Si la confirmation automatique n'est pas configurée, copiez le message **`SubscribeURL`**de confirmation et collez-le dans la barre d'adresse de votre navigateur.

  Cela confirmera l'abonnement manuellement.

4. Vous pouvez également vérifier le statut de l'abonnement à l'aide de la console Amazon SNS :

   a. Connectez-vous à la console Amazon SNS.
   b. Dans le panneau de navigation, choisissez Abonnements.
   c. Trouvez votre abonnement dans la liste.

      - En cas de confirmation, le `SubscriptionArn` sera affiché.
      - S'il n'est toujours pas confirmé, il s'affichera sous la forme`PendingConfirmation`.

Étape 4 : Facultatif - Définissez la politique de livraison pour l'abonnement Amazon SNS

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 HTTP/S de livraison.

Étape 5 : Facultatif : autorisez les utilisateurs à publier sur la rubrique Amazon SNS

Par défaut, le propriétaire de la rubrique dispose d''autorisations de publication dans la rubrique. Pour permettre à d'autres utilisateurs ou applications de publier sur le sujet, vous devez utiliser AWS Identity and Access Management (IAM) pour autoriser la publication du sujet. 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 ajoutiez la politique suivante à un utilisateur ou à un groupe IAM, vous autoriseriez cet utilisateur ou les membres de ce groupe à effectuer l'`sns:Publish`action sur le sujet MyTopic.

```
{
  "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 une autre personne l' Compte AWS accès à une ressource de votre compte, vous accordez également aux utilisateurs IAM disposant d'un accès de niveau administrateur (accès générique) des autorisations d'accès à cette ressource. Tous les autres utilisateurs IAM de l'autre compte se voient automatiquement refuser l'accès à votre ressource. Si vous souhaitez accorder à des utilisateurs IAM spécifiques de cette ressource un Compte AWS accès à votre ressource, le compte ou un utilisateur IAM disposant d'un accès de niveau administrateur doit déléguer les 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 un sujet MyTopic du compte 123456789012, vous autorisez le compte 111122223333 à effectuer l'action sur ce sujet. `sns:Publish`

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Étape 6 : envoyer des messages Amazon SNS 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. À l'aide des informations d'identification de l'utilisateur Compte AWS ou de l'utilisateur IAM autorisé à publier sur le sujet, connectez-vous à la console Amazon SNS AWS Management Console et ouvrez-la 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

Amazon SNS utilise les signatures des messages pour confirmer l'authenticité des messages envoyés à votre point de terminaison HTTP. Pour garantir l'intégrité des messages et empêcher toute usurpation d'identité, vous devez vérifier la signature avant de traiter les messages Amazon SNS.

Quand devez-vous vérifier les signatures Amazon SNS ?

Vous devez vérifier les signatures des messages Amazon SNS dans les scénarios suivants :

- Lorsqu'Amazon SNS envoie un message de notification à votre point de terminaison HTTP (S).
- Lorsqu'Amazon SNS envoie un message de confirmation à votre point de terminaison après un appel `Subscribe`d'`Unsubscribe`API.

Amazon SNS prend en charge deux versions de signature :

- SignatureVersion1 — Utilise un SHA1 hachage du message.
- SignatureVersion2 — Utilise un SHA256 hachage du message. Cela renforce la sécurité et constitue l'option recommandée.

Pour vérifier correctement les signatures des messages SNS, suivez les meilleures pratiques suivantes :

- Récupérez toujours le certificat de signature à l'aide du protocole HTTPS pour empêcher les attaques d'interception non autorisées.
- Vérifiez que le certificat est émis par Amazon SNS.
- Vérifiez que la chaîne de confiance du certificat est valide.
- Le certificat doit provenir d'une URL signée par le SNS.
- Ne faites confiance à aucun certificat fourni dans le message sans validation.
- Rejetez tout message contenant un message inattendu `TopicArn` pour éviter toute usurpation d'identité.
- AWS SDKs Pour Amazon SNS, ils fournissent une logique de validation intégrée, ce qui réduit le risque de mauvaise implémentation.

Configuration de la version de signature des messages sur les rubriques Amazon SNS

La configuration de la version de signature des messages sur les rubriques Amazon SNS vous permet d'améliorer la sécurité et la compatibilité de votre processus de vérification des messages.

Sélectionnez entre `SignatureVersion` 1 (SHA1) et `SignatureVersion` 2 (SHA256) pour contrôler l'algorithme de hachage utilisé pour signer les messages. La valeur par défaut des rubriques Amazon SNS est 1. **`SignatureVersion`** Vous pouvez configurer ce paramètre à l'aide de l'action `SetTopicAttributes`API.

Utilisez l'exemple suivant pour définir l'attribut topic à l'`SignatureVersion`aide de AWS CLI :

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

Vérification de la signature d'un message Amazon SNS lors de l'utilisation de requêtes HTTP

La vérification de la signature d'un message Amazon SNS lors de l'utilisation de requêtes HTTP garantit l'authenticité et l'intégrité du message. Ce processus confirme que le message provient d'Amazon SNS et qu'il n'a pas été falsifié pendant le transport. En analysant le message, en créant la chaîne appropriée à signer et en validant la signature par rapport à une clé publique fiable, vous protégez votre système contre l'usurpation d'identité et les modifications non autorisées des messages.

1. Extrayez les paires clé-valeur du document JSON dans le corps de la requête HTTP POST envoyée par Amazon SNS. Ces champs sont obligatoires pour créer la chaîne à signer.

   - `Message`
   - `Subject`(si présent)
   - `MessageId`
   - `Timestamp`
   - `TopicArn`
   - `Type`

   Par exemple :

```
MESSAGE_FILE="message.json"
FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
```

> **Note**
>
> Si un champ contient des caractères échappés (par exemple,`\n`), convertissez-les dans leur forme d'origine pour garantir une correspondance exacte.

2. Localisez le `SigningCertURL` champ dans le message Amazon SNS. Ce certificat contient la clé publique nécessaire pour vérifier la signature du message. Par exemple :

```
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
```

3. Assurez-vous qu'`SigningCertURL`il provient d'un AWS domaine fiable (par exemple, https://sns.us-east-1.amazonaws.com). Rejetez tous AWS les domaines URLs extérieurs pour des raisons de sécurité.
4. Téléchargez le certificat X.509 à partir de l'URL fournie. Par exemple :

```
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
```

5. Extrayez la clé publique du certificat X.509 téléchargé. La clé publique permet de déchiffrer la signature du message et de vérifier son intégrité. Par exemple :

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. Les différents types de messages nécessitent différentes paires clé-valeur dans la chaîne à signer. Identifiez le type de message (`Type`champ dans le message Amazon SNS) pour déterminer les paires clé-valeur à inclure :
   - Message de notification — Comprend `Message`e`MessageId`, `Subject` (le cas échéant)`Timestamp`,`TopicArn`, et`Type`.
   - SubscriptionConfirmationou UnsubscribeConfirmation message — Comprend`Message`, `MessageId`e`SubscribeURL`,`Timestamp`,`Token`,`TopicArn`, et`Type`.
7. Amazon SNS exige que la chaîne à signer respecte un ordre de champs strict et fixe à des fins de vérification. Seuls les champs explicitement obligatoires doivent être inclus. Aucun champ supplémentaire ne peut être ajouté. Les champs facultatifs, tels que`Subject`, ne doivent être

inclus que s'ils sont présents dans le message et doivent apparaître dans la position exacte définie par l'ordre des champs requis. Par exemple :

```
KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
```

> **Important**
>
> N'ajoutez pas de caractère de nouvelle ligne à la fin de la chaîne.

8. Disposez les paires clé-valeur par ordre de tri des octets (ordre alphabétique du nom de la clé).
9. Construisez la chaîne à signer en utilisant l'exemple de format suivant :

```
STRING_TO_SIGN=""
for FIELD in "${FIELDS[@]}"; do
    VALUE=$(jq -r --arg field "$FIELD" '.[$field]' "$MESSAGE_FILE")
    STRING_TO_SIGN+="$FIELD\n$VALUE"
    # Append a newline after each field except the last one
    if [[ "$FIELD" != "Type" ]]; then
        STRING_TO_SIGN+="\n"
    fi
done
```

Exemple de message de 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 exemple :

```
Message
```

```
Please confirm your subscription
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/...
Timestamp
2024-01-01T00:00:00.000Z
Token
abc123...
TopicArn
arn:aws:sns:us-east-2:123456789012:MyTopic
Type
SubscriptionConfirmation
```

10. Le `Signature` champ du message est codé en Base64. Vous devez le décoder pour comparer sa forme binaire brute avec le hachage dérivé. Par exemple :

```
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
echo "$SIGNATURE" | base64 -d > signature.bin
```

11. Utilisez le `SignatureVersion` champ pour sélectionner l'algorithme de hachage :
    - Pour `SignatureVersion` 1, utilisez SHA1(par exemple,`-sha1`).
    - Pour `SignatureVersion` 2, utilisez SHA256(par exemple,`-sha256`).
12. Pour confirmer l'authenticité du message Amazon SNS, générez un hachage de la chaîne construite et vérifiez la signature à l'aide de la clé publique.

```
openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
 "$STRING_TO_SIGN"
```

    Si la signature est valide, le résultat est`Verified OK`. Dans le cas contraire, le résultat est`Verification Failure`.

Exemple de script avec gestion des erreurs

L'exemple de script suivant automatise le processus de vérification :

```
#!/bin/bash

# Path to the local message file
MESSAGE_FILE="message.json"
```

```
# Extract the SigningCertURL and Signature from the message
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")

# Fetch the X.509 certificate
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem

# Extract the public key from the certificate
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem

# Define the fields to include in the string to sign
FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")

# Initialize the string to sign
STRING_TO_SIGN=""

# Iterate over the fields to construct the string to sign
for FIELD in "${FIELDS[@]}"; do
    VALUE=$(jq -r --arg field "$FIELD" '.[$field]' "$MESSAGE_FILE")
    STRING_TO_SIGN+="$FIELD\n$VALUE"
    # Append a newline after each field except the last one
    if [[ "$FIELD" != "Type" ]]; then
        STRING_TO_SIGN+="\n"
    fi
done

# Verify the signature
echo -e "$STRING_TO_SIGN" | openssl dgst -sha256 -verify public_key.pem -signature
 <(echo "$SIGNATURE" | base64 -d)
```

Analyse des formats de message Amazon SNS

Lorsqu'Amazon SNS envoie des messages à des points de terminaison HTTP/HTTPS, ils contiennent à la fois des en-têtes HTTP et un corps de message JSON. Ces messages suivent un format structuré qui inclut des métadonnées telles que le type de message, l'ARN du sujet, les horodatages et les signatures numériques. En analysant correctement les messages Amazon SNS, vous pouvez déterminer s'il s'agit d'une confirmation d'abonnement, d'une notification ou d'une confirmation de désinscription, extraire les données pertinentes et vérifier l'authenticité à l'aide de la validation de signature.

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 HTTP/S de livraison.

`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

Après avoir inscrit un HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS point de terminaison. 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 chesen 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"
}
```

SetSubscriptionAttributespolitique de livraison au format JSON

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 cette `SetSubscriptionAttribute` action, consultez le SetSubscriptionAttributesmanuel Amazon Simple Notification Service API Reference. Pour plus d'informations sur les en-têtes content-type HTTP pris en charge, consultez Création d'une politique HTTP/S de livraison.

SetTopicAttributes politique de livraison au format JSON

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 plus d'informations sur cette `SetTopicAttribute` action, consultez le SetTopicAttributesmanuel Amazon Simple Notification Service API Reference. Pour plus d'informations sur les en-têtes content-type HTTP pris en charge, consultez Création d'une politique HTTP/S de livraison.

Transférez les événements Amazon SNS à Event Fork AWS Pipelines

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 de plus amples informations, veuillez consulter 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 des pipelines de gestion d'événements, alimentés par Event Fork AWS Pipelines, à des rubriques Amazon SNS. AWS Event Fork Pipelines est une suite d'applications imbriquées open source, basée sur le modèle d'application AWS sans serveur (AWS SAM), que vous pouvez déployer directement depuis la suite AWS Event Fork Pipelines (choisissez Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés) sur votre compte. AWS

Pour un AWS cas d'utilisation d'Event Fork Pipelines, voirDéploiement et test de l'exemple d'application Amazon SNS Event Fork Pipelines.

Rubriques

- Comment fonctionne AWS Event Fork Pipelines
- Déploiement de pipelines AWS Event Fork
- Déploiement et test de l'exemple d'application Amazon SNS Event Fork Pipelines
- Abonnement d' AWS Event Fork Pipelines à une rubrique Amazon SNS

Comment fonctionne AWS Event Fork Pipelines

AWS Event Fork Pipelines est un modèle de conception sans serveur. Cependant, il s'agit également d'une suite d'applications sans serveur imbriquées basées sur le AWS SAM (que vous pouvez déployer directement depuis le AWS Serverless Application Repository (AWS SAR) sur votre site Compte AWS afin d'enrichir vos plateformes axées sur les événements). Vous pouvez déployer ces applications imbriquées individuellement, comme requis par votre architecture.

Rubriques

- Pipeline de stockage et de sauvegarde d'événements

- Le pipeline de recherche et d'analyse d'événements
- Le pipeline de relecture d'événements

Le schéma suivant montre une application AWS Event Fork Pipelines complétée par trois applications imbriquées. Vous pouvez déployer n'importe lequel des pipelines de la suite AWS Event Fork Pipelines sur le AWS SAR indépendamment, selon les exigences de 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**
>
> Comme vous placez les trois pipelines AWS Event Fork à côté de vos pipelines de traitement d'événements habituels (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 pipelines AWS Event Fork dans vos charges de travail 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 transmet dans un flux, 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 transmis 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, d'un domaine Amazon OpenSearch 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 mémoire tampon les événements transmis par la rubrique Amazon SNS, et d'une AWS Lambda fonction qui interroge les événements de la file d'attente et les réachemine vers votre pipeline de traitement des événements normal, qui est également abonné à votre sujet.

> **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 de pipelines AWS Event Fork

La suite AWS Event Fork Pipelines (choisissez Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés) est disponible sous la forme d'un groupe d'applications publiques dans le AWS Serverless Application Repository, à partir duquel vous pouvez les déployer et les tester manuellement à l'aide de la AWS Lambda console. Pour plus d'informations sur le déploiement de pipelines à l'aide de la AWS Lambda console, consultezAbonnement d' AWS Event Fork Pipelines à une rubrique Amazon SNS.

Dans un scénario de production, nous recommandons d'intégrer AWS Event Fork Pipelines dans le modèle AWS SAM global de votre application. La fonctionnalité d'application imbriquée vous permet de le faire en ajoutant la ressource `AWS::Serverless::Application` à votre modèle AWS SAM, en référençant le AWS SAR `ApplicationId` et le `SemanticVersion` de l'application imbriquée.

Par exemple, vous pouvez utiliser Event Storage and Backup Pipeline comme application imbriquée en ajoutant l'extrait YAML suivant à la `Resources` section de votre modèle SAM. AWS

```
Backup:
    Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-2:123456789012:applications/fork-
event-storage-backup-pipeline
      SemanticVersion: 1.0.0
    Parameters:
      #The ARN of the Amazon SNS topic whose messages should be backed up to the Amazon
 S3 bucket.
      TopicArn: !Ref MySNSTopic
```

Lorsque vous spécifiez des valeurs de paramètres, vous pouvez utiliser des fonctions AWS CloudFormation intrinsèques pour référencer d'autres ressources dans votre modèle. Par exemple, dans l'extrait YAML ci-dessus, le `TopicArn` paramètre fait référence à la `AWS::SNS::Topic`

ressource`MySNSTopic`, définie ailleurs dans le modèle. AWS SAM Pour plus d'informations, consultez la section de Référence des fonctions intrinsèques dans le AWS CloudFormation Guide de l'utilisateur.

> **Note**
>
> La page de AWS Lambda console de votre application AWS SAR inclut le bouton Copier en tant que ressource SAM, qui copie le code YAML requis pour imbriquer une application AWS SAR dans le presse-papiers.

Déploiement et test de l'exemple d'application Amazon SNS Event Fork Pipelines

Pour accélérer le développement de vos applications basées sur les événements, vous pouvez abonner des pipelines de gestion d'événements, alimentés par Event Fork AWS Pipelines, à des rubriques Amazon SNS. AWS Event Fork Pipelines est une suite d'applications imbriquées open source, basée sur le modèle d'application AWS sans serveur (AWS SAM), que vous pouvez déployer directement depuis la suite AWS Event Fork Pipelines (choisissez Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés) sur votre compte. AWS Pour de plus amples informations, veuillez consulter Comment fonctionne AWS Event Fork Pipelines.

Cette page explique comment vous pouvez utiliser le AWS Management Console pour déployer et tester l'exemple d'application AWS Event Fork Pipelines.

> **Important**
>
> Pour éviter d'encourir des coûts indésirables une fois le déploiement de l'exemple d'application AWS Event Fork Pipelines terminé, supprimez sa AWS CloudFormation pile. Pour plus d'informations, consultez Suppression d'une pile dans la console AWS CloudFormation dans le Guide de l'utilisateur AWS CloudFormation .

AWS Exemple de cas d'utilisation d'Event Fork Pipelines

Le scénario suivant décrit une application de commerce électronique sans serveur pilotée par des événements qui utilise AWS Event Fork Pipelines. 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 de commerce électronique prend les commandes des acheteurs via une RESTful API hébergée par API Gateway et soutenue par la AWS Lambda fonction`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 comprend la file d'attente Amazon SQS `CheckoutQueue` qui met en mémoire tampon toutes les commandes reçues, une AWS Lambda fonction nommée `CheckoutFunction` qui interroge la file d'attente pour traiter ces commandes, et la table `CheckoutTable` DynamoDB qui enregistre de manière sécurisée toutes les commandes passées.

Appliquer les pipelines AWS Event Fork

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 AWS Event Fork Pipelines à la `CheckoutEventsTopic` rubrique Amazon SNS

- 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 AWS et alimentée par le 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, veuillez consulter Filtrage des messages Amazon SNS.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

En utilisant le modèle AWS Event Fork Pipelines, le propriétaire de l'application de commerce électronique peut éviter les frais de développement qui accompagnent souvent le codage d'une logique indifférenciée pour la gestion des événements. Au lieu de cela, elle peut déployer AWS Event Fork Pipelines directement depuis elle Compte AWS. AWS Serverless Application Repository

Étape 1 : déploiement de l'exemple d'application Amazon SNS

1. Connectez-vous à la AWS Lambda console.
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 le nom d'une application, le déploiement ne mettra à jour que la AWS CloudFormation pile précédemment déployée (plutôt que d'en créer une nouvelle).

   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 *`my-app`* page État du déploiement pour fork-example-ecommerce -, Lambda affiche l'état Votre application est en cours de déploiement.

Dans la section Ressources, AWS CloudFormation commence à créer la pile et affiche le statut CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche le statut 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 : Exécution de l'exemple d'application lié au SNS

1. Dans la AWS Lambda console, sur le panneau de navigation, sélectionnez 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,
  "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 : vérification des performances de l'application et du pipeline Amazon SNS

Étape 1 : Vérification de 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 : Vérification de l'exécution du pipeline de stockage et de sauvegarde des 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 : Vérification de 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 car il fait référence à une commande dont la valeur est supérieure à 100 USD, sur la ***abcdefgh1ijk*** page serverl-analyt-, choisissez Indices, checkout_events.

Étape 4 : Vérification de l'exécution du pipeline de rediffusion 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 *123ABCD4E5F6* dialogue Envoyer et recevoir des messages dans fork-example-ecommerce -***my-app***... ReplayP- ReplayQueue -, 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 : Simulation d'un problème et rediffusion des événements pour le rétablissement

Étape 1 : activer le problème simulé et envoyer une deuxième demande d'API

1. Connectez-vous à la AWS Lambda console.
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,
"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 : vérifier la corruption de données simulée

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 : désactiver le problème simulé

1. Connectez-vous à la AWS Lambda console.
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 : activer la rediffusion pour résoudre le problème

1. Dans la AWS Lambda console, sur le panneau de navigation, choisissez Functions.
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 Save (Enregistrer).
5. Pour afficher les attributs récupérées, revenez à la console Amazon DynamoDB.
6. Pour désactiver le replay, retournez sur la AWS Lambda console et désactivez le déclencheur de la source d'événements Amazon SQS pour. `ReplayFunction`

Abonnement d' AWS Event Fork Pipelines à une rubrique Amazon SNS

Pour accélérer le développement de vos applications basées sur les événements, vous pouvez abonner des pipelines de gestion d'événements, alimentés par Event Fork AWS Pipelines, à des rubriques Amazon SNS. AWS Event Fork Pipelines est une suite d'applications imbriquées open source, basée sur le modèle d'application AWS sans serveur (AWS SAM), que vous pouvez déployer directement depuis la suite AWS Event Fork Pipelines (choisissez Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés) sur votre compte. AWS Pour de plus amples informations, veuillez consulter Comment fonctionne AWS Event Fork Pipelines.

Cette section explique comment vous pouvez utiliser le AWS Management Console pour déployer un pipeline, puis abonner AWS Event Fork Pipelines 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.

Déploiement et abonnement du pipeline de stockage et de sauvegarde des événements sur Amazon SNS

> 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 de plus amples informations, veuillez consulter 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 AWS SAM modèle associé au pipeline en une AWS CloudFormation pile, 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 Amazon 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 dans le manuel Amazon Data Firehose Developer Guide.

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 Lambda console.
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 le nom d'une application, le déploiement mettra à jour uniquement la AWS CloudFormation pile précédemment déployée (plutôt que d'en créer une nouvelle).

   b. (Facultatif) Pour BucketArn, entrez l'ARN du compartiment Amazon S3 dans lequel les événements entrants sont chargés. Si vous ne saisissez aucune valeur, un nouveau compartiment Amazon S3 est créé dans votre AWS compte.
   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 pour 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 Amazon 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 *`my-app`* 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 le statut CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche le statut 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 stockés dans le compartiment de sauvegarde Amazon S3 approvisionné automatiquement par le pipeline Event Storage and Backup.

Déploiement et abonnement du pipeline d'analyse et de recherche d'événements sur Amazon SNS

> 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 de plus amples informations, veuillez consulter 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 AWS SAM modèle associé au pipeline en une AWS CloudFormation pile, 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 dans le manuel Amazon Data Firehose Developer Guide.

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 Lambda console.
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 le nom d'une application, le déploiement mettra à jour uniquement la AWS CloudFormation pile précédemment déployée (plutôt que d'en créer une nouvelle).

   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 des types de service peuvent contenir n'importe quel caractère (sauf les 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 pour 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 Amazon 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 le statut CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche le statut 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 parvient pas à indexer un événement, il le stocke dans un compartiment contenant des lettres mortes Amazon S3.

Déploiement du pipeline de rediffusion d'événements avec l'intégration d'Amazon SNS

Cette page montre comment déployer le Pipeline de relecture des événements et vous abonner à une rubrique Amazon SNS. Ce processus transforme automatiquement le AWS SAM modèle associé au pipeline en une AWS CloudFormation pile, 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 Lambda console.
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 le nom d'une application, le déploiement mettra à jour uniquement la AWS CloudFormation pile précédemment déployée (plutôt que d'en créer une nouvelle).

   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 le statut CREATE_IN_PROGRESS pour chaque ressource. Lorsque le processus est terminé, AWS CloudFormation affiche le statut 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 d'Amazon EventBridge Scheduler avec Amazon SNS

Amazon EventBridge Scheduler est un planificateur sans serveur qui vous permet de créer, d'exécuter et de gérer des tâches à partir d'un service géré centralisé. Avec EventBridge Scheduler, vous pouvez créer des plannings à l'aide de Cron et d'expressions de taux pour les modèles récurrents, ou configurer des appels ponctuels. 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 EventBridge planificateur pour publier un message à partir d'une rubrique Amazon SNS selon un calendrier.

Configuration du rôle d'exécution

Lorsque vous créez un nouveau calendrier, le EventBridge planificateur doit être autorisé à appeler son opération d'API cible en votre nom. Vous accordez ces autorisations au EventBridge planificateur 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 le EventBridge Scheduler doit appeler.

Lorsque vous utilisez la console du EventBridge planificateur pour créer un calendrier, comme dans la procédure suivante, le EventBridge planificateur définit automatiquement un rôle d'exécution en fonction de la cible que vous avez sélectionnée. Si vous souhaitez créer un calendrier à l'aide de l'un des EventBridge planificateurs SDKs AWS CloudFormation, vous devez disposer d'un rôle d'exécution existant qui accorde les autorisations dont le EventBridge planificateur a besoin pour appeler une cible. AWS CLIPour plus d'informations sur la configuration manuelle d'un rôle d'exécution pour votre calendrier, consultez la section Configuration d'un rôle d'exécution dans le guide de l'utilisateur du EventBridge planificateur.

Créer une planification

Pour créer une planification à l’aide de la console

1. Ouvrez la console Amazon EventBridge Scheduler à https://console.aws.amazon.com/scheduler/la maison.
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

Occurrence	Faites ceci...	
	sur cron et entrez l'expression cron. • Pour utiliser une expression de rythme pour définir la planification, choisissez Planification basée sur le rythme. Pour plus d'informations sur les expressions cron et rate, consultez la section Types de planification sur EventBridge Scheduler dans le guide de l'utilisateur d'Amazon EventBridge Scheduler. 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	

Occurrence	Faites ceci...	
	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 Suivant.
7. Sur la page Sélectionner une cible, choisissez l'opération AWS d'API invoquée par le EventBridge planificateur :
   a. Sélectionnez Publication Amazon SNS.
   b. Dans la section Publier, sélectionnez un SNS sujet ou choisissez Créer un nouveau SNS sujet.
   c. (Facultatif) Entrez un JSON charge utile. Si vous n'entrez aucune charge utile, le EventBridge planificateur utilise un événement vide pour appeler la fonction.
8. Choisissez 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 maximal de l'événement, entrez le nombre maximum d'heures et de minutes pendant lequel le EventBridge planificateur doit conserver un événement non traité.
      - La durée maximale est 24 heures.
      - Pour Nombre maximum de tentatives, entrez le nombre maximal de fois que le EventBridge planificateur réessaie le calendrier si la cible renvoie une erreur.

La valeur maximale est 185 nouvelles tentatives.

Avec les politiques de nouvelle tentative, si un calendrier ne parvient pas à invoquer sa cible, le EventBridge planificateur le réexécute. Si elle est configurée, vous devez définir la durée de rétention maximale et les nouvelles tentatives pour la planification.

c. Choisissez l'endroit où EventBridge Scheduler stocke les événements non livrés.

Option File d'attente de lettres mortes (DLQ)	Faites ceci...
Ne stockez pas	Sélectionnez Aucun.
Enregistrez l'événeme nt dans le même Compte AWS endroit où vous créez le calendrier	a. Choisissez Sélection nez une file d'attente Amazon SQS dans my Compte AWS as a DLQ. b. Choisissez l'Amazon Resource Name (ARN) de la file d'attente Amazon SQS.
Stockez l'événement dans un endroit Compte AWS différent de celui dans lequel vous créez le calendrier	a. Choisissez Spécifier une file d'attente Amazon SQS dans un autre en Comptes AWS 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

manière dont EventBridge Scheduler chiffre vos données au repos, consultez la section Chiffrement au repos dans le guide de l'utilisateur d'Amazon EventBridge Scheduler.

e. Pour que le EventBridge planificateur crée un nouveau rôle d'exécution pour vous, choisissez Créer un nouveau rôle pour ce calendrier. Ensuite, saisissez un nom pour Nom du rôle. Si vous choisissez cette option, le EventBridge planificateur associe au rôle les autorisations requises pour votre cible modélisée.

10. Choisissez 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 plus d'informations sur le EventBridge planificateur, consultez les rubriques suivantes :

- EventBridge Guide de l'utilisateur du planificateur
- EventBridge Référence de l'API Scheduler
- EventBridge Tarification du planificateur

Utilisation d'Amazon SNS pour la messagerie application-to-person

La messagerie Amazon SNS application-to-person (A2P) vous permet d'envoyer des notifications et des alertes directement sur les appareils mobiles de vos clients par SMS (Short Message Service). Grâce à cette fonctionnalité, vous pouvez envoyer des notifications push aux applications mobiles, des SMS aux numéros de téléphone portable et des e-mails en texte brut aux adresses e-mail. En outre, vous avez la possibilité de distribuer des messages en utilisant des sujets pour atteindre plusieurs destinataires, ou de publier des messages directement sur des terminaux mobiles individuels pour une communication personnalisée.

Cette rubrique explique comment utiliser Amazon SNS pour les notifications utilisateur adressées aux abonnés, telles que les applications mobiles, les numéros de téléphone portable et les adresses e-mail.

Messagerie texte mobile avec Amazon SNS

> **Important**
>
> Le guide du développeur de SMS Amazon SNS a été mis à jour. Amazon SNS s'est intégré à AWS End User Messaging SMSla livraison de messages SMS. Ce guide contient les dernières informations sur la création, la configuration et la gestion de vos messages SMS Amazon SNS.

La messagerie texte (SMS) mobile Amazon SNS est conçue pour faciliter la transmission de messages vers diverses plateformes, telles que les applications Web, mobiles et professionnelles qui prennent en charge les SMS. Les utilisateurs peuvent envoyer des messages à un ou plusieurs numéros de téléphone en les abonnant à une rubrique, ce qui simplifie le processus de distribution.

Les messages Amazon SNS sont envoyés par AWS End User Messaging SMS, ce qui garantit une transmission fiable des messages. Dans Amazon SNS APIs, vous pouvez définir diverses propriétés telles que les types de messages (promotionnels ou transactionnels), les limites de dépenses mensuelles, les listes de désinscription et l'optimisation de la diffusion des messages.

AWS End User Messaging SMS gère la transmission des messages au numéro de téléphone de destination via son réseau mondial de fourniture de SMS. Il gère le routage, le statut de livraison et toute conformité requise aux réglementations régionales. Pour accéder à des fonctionnalités SMS supplémentaires telles que les autorisations granulaires, les pools téléphoniques, les ensembles de configurations, le simulateur de SMS et les règles nationales, consultez le guide de l'AWS End User Messaging SMS utilisateur.

Les fonctionnalités clés suivantes vous permettent d'envoyer des messages SMS Amazon SNS évolutifs et facilement extensibles :

Personnaliser les préférences relatives aux messages

Personnalisez les envois de SMS Compte AWS selon vos besoins en configurant les préférences SMS en fonction de votre budget et de votre cas d'utilisation. Par exemple, vous pouvez choisir si vos messages privilégient la rentabilité ou la fiabilité de leur diffusion.

Définissez des quotas de dépenses

Personnalisez vos envois de SMS en spécifiant des quotas de dépenses ou pour des envois de messages individuels et des quotas de dépenses mensuels pour votre Compte AWS. Lorsque les lois et réglementations locales (telles que les États-Unis et le Canada) l'exigent, les destinataires de SMS peuvent se désinscrire, ce qui signifie qu'ils choisissent de ne plus recevoir de SMS de votre part Compte AWS. Une fois qu'un destinataire a refusé de recevoir des messages, vous pouvez, avec certaines restrictions, réinscrire le numéro de téléphone afin de pouvoir recommencer à envoyer des messages.

Envoyez des SMS dans le monde entier

Amazon SNS prend en charge la messagerie SMS dans plusieurs régions, ce qui vous permet d'envoyer des messages dans plus de 240 pays et régions.

Comment Amazon SNS livre-t-il mes SMS ?

Lorsque vous demandez à Amazon SNS d'envoyer des SMS en votre nom, les messages sont envoyés à l'aide de. AWS End User Messaging SMS L'intégration entre Amazon SNS et Amazon AWS End User Messaging SMS offre les avantages suivants :

Politiques IAM

Vous pouvez tirer parti de l'IAM et des politiques de ressources pour contrôler et répartir l'accès à vos ressources SMS entre d'autres AWS services et régions.

Configurations AWS End User Messaging SMS

Toutes les configurations liées à l'identifiant d'origine (création, mise à jour de configuration, mise en service d'une nouvelle origine IDs, modification des modèles d'enregistrement) sont utilisées. AWS End User Messaging SMS

AWS End User Messaging SMS facturation

Cependant, toutes les facturations par SMS sont effectuées AWS End User Messaging SMS. Vous pouvez consolider vos AWS dépenses pour vos charges de travail liées aux SMS, tout en centralisant l'approvisionnement et la gestion de vos ressources SMS.

Commencer à utiliser Amazon SNS SMS

> **Important**
>
> Le guide du développeur de SMS Amazon SNS a été mis à jour. Amazon SNS s'est intégré à AWS End User Messaging SMSla livraison de messages SMS. Ce guide contient les dernières informations sur la création, la configuration et la gestion de vos messages SMS Amazon SNS.

Cette rubrique vous explique comment gérer votre sandbox SMS et configurer les politiques IAM et basées sur les ressources afin d'accorder à Amazon SNS les autorisations nécessaires pour accéder et utiliser le. AWS End User Messaging SMS APIs

Prérequis

Amazon SNS recommande de mettre à jour votre politique IAM afin d'inclure les actions suivantes afin de garantir un contrôle et une visibilité complets sur vos ressources Amazon SNS :

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

Utilisation du sandbox SMS Amazon SNS

Les comptes SMS Amazon SNS nouvellement créés sont automatiquement placés dans le bac à sable des SMS afin de garantir la sécurité des AWS clients et des destinataires en atténuant les risques de fraude et d'abus. Cet environnement sert d'espace sécurisé à des fins de test et de développement. Lorsque vous travaillez dans le sandbox SMS, vous avez accès à toutes les fonctionnalités d'Amazon SNS, mais vous êtes soumis à certaines restrictions :

- 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'au moins 24 heures après la vérification ou la dernière tentative de vérification.

Une fois que votre compte sort du sandbox, ces restrictions sont supprimées et vous pouvez envoyer des SMS à n'importe quel destinataire.

Premiers pas

Les nouveaux comptes SMS Amazon SNS sont placés dans un sandbox SMS. Suivez les étapes ci-dessous pour créer et gérer des numéros de téléphone dans votre sandbox, créer des numéros d'origine et d'expéditeur IDs, et enregistrer votre entreprise.

1. Ajoutez un numéro de téléphone de destination au sandbox SMS. Pour en savoir plus sur l'ajout, la gestion et le transfert de numéros de téléphone hors du sandbox Amazon SNS SMS, consultez. Ajouter et vérifier des numéros de téléphone dans le sandbox Amazon SNS SMS
2. Créez une identité d'origine que vos destinataires verront sur leurs appareils lorsque vous leur envoyez un message SMS. Pour en savoir plus sur les identités d'origine, y compris les différents types que vous pouvez utiliser, consultez la Identités d'origine pour les SMS Amazon SNS documentation.
3. Enregistrez votre entreprise. Certains pays exigent que vous enregistriez l'identité de votre entreprise pour pouvoir acheter des numéros de téléphone ou des expéditeurs IDs et consulter les messages que vous envoyez aux destinataires dans leur pays. Pour plus d'informations sur les pays nécessitant un enregistrement, consultez la section Pays et régions pris en charge pour la messagerie SMS AWS End User Messaging SMS dans le guide de AWS End User Messaging SMS l'utilisateur.

4. Envoyez vos messages à un sujet ou à un téléphone portable. Pour de plus amples informations, veuillez consulter Envoi de SMS à l'aide d'Amazon SNS.

Ajouter et vérifier des numéros de téléphone dans le sandbox Amazon SNS SMS

Avant de pouvoir commencer à envoyer des SMS depuis votre Compte AWS navigateur dans le sandbox SMS, vous devez suivre les étapes de configuration suivantes. Cela garantit que votre compte est prêt pour la messagerie SMS et que vos numéros de téléphone de destination sont correctement vérifiés.

1. Créez un identifiant d'origine. Comme pour les comptes extérieurs au sandbox SMS, un identifiant d'origine est nécessaire pour pouvoir envoyer des SMS à des destinataires dans certains pays ou régions.
2. Ajoutez les numéros de téléphone de destination auxquels vous souhaitez envoyer des messages dans le sandbox SMS.
3. Vérifiez les numéros de téléphone pour vous assurer que les numéros de téléphone de destination sont valides pour être utilisés dans vos messages SMS.

Ajouter et vérifier les numéros de téléphone de destination

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, choisissez une région qui prend en charge la messagerie SMS.
3. Dans le panneau de navigation, sélectionnez Text messaging (SMS).
4. Dans la section Numéros de téléphone de destination du Sandbox, sélectionnez Ajouter un numéro de téléphone.
5. Sous Détails de la destination, fournissez les informations suivantes, puis sélectionnez Ajouter un numéro de téléphone :
   - Code du pays et numéro de téléphone de la destination.
   - Langue dans laquelle vous souhaitez que le message de vérification soit envoyé.
6. Après avoir ajouté le numéro de téléphone, Amazon SNS enverra un OTP au numéro de téléphone de destination fourni. Cet OTP est requis à des fins de vérification.
7. Vous recevrez l'OTP sous forme de SMS standard sur le numéro de téléphone de destination que vous avez fourni.

- Si vous ne recevez pas l'OTP dans les 15 minutes, sélectionnez Renvoyer le code de vérification dans la console Amazon SNS.
- Vous pouvez renvoyer l'OTP jusqu'à cinq fois par période de 24 heures.

8. Une fois que vous avez reçu l'OTP, saisissez-le dans le champ Code de vérification et sélectionnez Vérifier le numéro de téléphone.
9. Vérifiez l'état de la vérification.

   - Une fois le numéro de téléphone vérifié, le numéro de téléphone et son statut de vérification apparaîtront dans la section des numéros de téléphone de destination du Sandbox.
   - Si le statut est En attente, la vérification a échoué. Cela peut se produire si, par exemple, vous n'avez pas saisi correctement le code du pays.
   - Vous ne pouvez supprimer les numéros de téléphone en attente ou vérifiés qu'au bout de 24 heures ou plus depuis la dernière tentative de vérification.
10. Si vous souhaitez utiliser le même numéro de téléphone de destination dans d'autres régions, répétez les étapes précédentes pour chaque région dans laquelle vous souhaitez l'utiliser.

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.

Supprimer des numéros de téléphone du sandbox Amazon SNS SMS

Vous pouvez supprimer les numéros de téléphone de destination en attente et vérifiés du sandbox SMS.

> **Important**
>
> Vous ne pouvez supprimer un numéro de téléphone que 24 heures après l'avoir vérifié ou 24 heures après votre dernière tentative de vérification.

Pour supprimer les numéros de téléphone de destination de l'environnement de test (sandbox) pour SMS

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, choisissez une région qui prend en charge la messagerie SMS dans laquelle vous avez ajouté un numéro de téléphone de destination.
3. Dans le volet de navigation, sélectionnez Messagerie texte (SMS).
4. Sur la page Messagerie texte (SMS) mobile, accédez à la section Numéros de téléphone de destination du Sandbox.
5. Choisissez le numéro de téléphone que vous souhaitez supprimer, puis sélectionnez Supprimer le numéro de téléphone.
6. Pour confirmer que vous souhaitez supprimer le numéro de téléphone, saisissez **`delete me`**, puis choisissez Supprimer.

   Assurez-vous que 24 heures ou plus se sont écoulées depuis que vous avez vérifié ou tenté de vérifier le numéro de téléphone de destination avant de procéder à la suppression.

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.

Sortir du sandbox SMS d'Amazon SNS

Pour Compte AWS sortir du sandbox SMS, vous devez d'abord ajouter, vérifier et tester les numéros de téléphone de destination. Après cela, créez un étui 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, consultezPublier des SMS sur un téléphone mobile à l'aide d'Amazon SNS.
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 Support.

L' 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 pour les SMS Amazon SNS

> **Important**
>
> Le guide du développeur de SMS Amazon SNS a été mis à jour. Amazon SNS s'est intégré à AWS End User Messaging SMSla livraison de messages SMS. Ce guide contient les dernières informations sur la création, la configuration et la gestion de vos messages SMS Amazon SNS.

Les identités d'origine des messages SMS sont des identifiants utilisés pour représenter l'expéditeur d'un message SMS. Vous pouvez vous identifier auprès de vos destinataires en utilisant les types d'identités d'origine suivants :

Numéros d'origine

Chaîne numérique identifiant le numéro de téléphone de l'expéditeur d'un SMS. Il existe plusieurs types de numéros d'origine, notamment les codes longs (numéros de téléphone standard comportant généralement 10 chiffres ou plus), les codes longs à 10 chiffres (10DLC), les numéros gratuits (TFN) et les codes courts (numéros de téléphone contenant entre quatre et sept chiffres).

Support pour les numéros d'origine n'est pas disponible dans les pays où les lois locales exigent l'utilisation de l'expéditeur IDs plutôt que du numéro d'origine. 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.

Pour plus d'informations, consultez la section Numéros de téléphone dans le guide de AWS End User Messaging SMS l'utilisateur.

> **Tip**
>
> Pour afficher la liste de tous les numéros d'origine existants dans votre AWS compte, dans le volet de navigation de la console Amazon SNS, sélectionnez Numéros d'origine.

Expéditeur IDs

Nom alphabétique identifiant l'expéditeur d'un message SMS. Lorsque vous envoyez un SMS à l'aide d'un identifiant d'expéditeur et que le destinataire se trouve dans une zone où

l'authentification par identifiant d'expéditeur est prise en charge, votre identifiant d'expéditeur apparaît sur l'appareil du destinataire au lieu de votre 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.

IDs Les expéditeurs sont pris en charge dans plusieurs pays et régions du 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 exigent l'envoi d'un expéditeur IDs, consultez la section Pays et régions pris en charge pour la messagerie SMS AWS End User Messaging SMS dans le guide de AWS End User Messaging SMS l'utilisateur.

L'utilisation de l'expéditeur est gratuite IDs. Cependant, la prise en charge et les exigences relatives à l'authentification de l'identifiant de l'expéditeur varient selon les pays. Plusieurs grands marchés (dont le Canada, la Chine et les États-Unis d'Amérique) ne prennent pas en charge l'utilisation de l'expéditeur. IDs 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.

Pour plus d'informations, voir Expéditeur IDs dans le guide de AWS End User Messaging SMS l'utilisateur.

Configuration de la messagerie SMS dans Amazon SNS

> **Important**
>
> Le guide du développeur de SMS Amazon SNS a été mis à jour. Amazon SNS s'est intégré à AWS End User Messaging SMSla livraison de messages SMS. Ce guide contient les dernières informations sur la création, la configuration et la gestion de vos messages SMS Amazon SNS.

Vous pouvez utiliser les configurations d'Amazon SNS SMS pour définir les préférences SMS en fonction de vos besoins, telles que l'ajustement des quotas de dépenses et la configuration de l'enregistrement du statut de livraison. Cette rubrique explique également comment publier des messages SMS sur des sujets à l'aide de la console Amazon SNS et du AWS SDK, gérer efficacement les quotas et récupérer des statistiques détaillées sur l'activité des SMS.

Envoi de SMS à l'aide d'Amazon SNS

Cette section explique comment envoyer des SMS à l'aide d'Amazon SNS, notamment en publiant sur un sujet, en abonnant des numéros de téléphone à des sujets, en définissant des attributs sur les messages et en publiant directement sur les téléphones mobiles.

Publication de messages SMS sur une rubrique Amazon SNS

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 le sujet jusqu'à ce que vous annuliez son abonnement ou jusqu'à ce qu'il refuse de recevoir des SMS de votre AWS compte.

Envoi d'un message à un sujet à l'aide de la AWS 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 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 SMS, votre AWS compte se trouve dans le sandbox 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 Utilisation du sandbox SMS Amazon SNS.

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 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 Publier des SMS sur un téléphone mobile à l'aide d'Amazon SNS.

5. (Facultatif) Pour les attributs de message, ajoutez des métadonnées de message telles que les horodatages, les signatures et. IDs
6. Choisissez Publier le message. Amazon SNS envoie le message SMS et affiche un message de réussite.

Envoi d'un message à un sujet à l'aide du AWS SDKs

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

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);
        }
    }
}
```

Publier des SMS sur un téléphone mobile à l'aide d'Amazon SNS

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 de messages SMS sur une rubrique Amazon SNS.

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 SDKs du, 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 apparaît sous la forme +1 XXX555 0100.

Envoi d'un message (console)

1. Connectez-vous à la console Amazon SNS.
2. Dans le menu de la console, choisissez une région 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.

     Support pour les expéditeurs IDs varie en fonction du pays et de and/or la région. 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 expéditeurs IDs, consultez la section Pays et régions pris en charge pour la messagerie SMS AWS End User Messaging SMS dans le guide de AWS End User Messaging SMS l'utilisateur.

     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 expéditeurs IDs sont pris en charge, un AVIS est affiché.

     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 répertoire 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 de plus amples informations, veuillez consulter Identités d'origine pour les SMS Amazon SNS.

     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 la configuration de votre 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 SMS aux destinataires en Inde, consultez le guide de l'utilisateur sur le processus d'enregistrement de l'identifiant de AWS End User Messaging SMS l'expéditeur en 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 SDKs)

Pour envoyer un message SMS à l'aide de l'un des AWS SDKs, 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.

Support pour les expéditeurs IDs varie en fonction du pays ou de la région. Par exemple, les messages diffusés à des numéros de téléphone américains n'affichent pas l'ID d'expéditeur. Pour obtenir la liste des pays ou régions qui prennent en charge les expéditeurs IDs, consultez la section Pays et régions pris en charge pour la messagerie SMS AWS End User Messaging SMS dans le guide de AWS End User Messaging SMS l'utilisateur.

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 la section Numéros de téléphone dans le guide de AWS End User Messaging SMS l'utilisateur.

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

SDK pour .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 pour .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 pour 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.fromEnvironment { 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 pour PHP.
- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK pour 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).

Configuration des préférences de messagerie SMS dans Amazon SNS

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 de plus amples informations, veuillez consulter Publier des SMS sur un téléphone mobile à l'aide d'Amazon SNS.

Définition des préférences SMS à l'aide de AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Choisissez une région qui prend en charge les SMS.
3. Sur le panneau de navigation, choisissez Mobile, puis Messagerie texte (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**
>
> Support pour les expéditeurs IDs varie selon les pays.

7. (Facultatif) Saisissez le nom du compartiment Amazon S3 pour les rapports d'utilisation.

> **Note**
>
> La politique relative aux compartiments Amazon S3 doit accorder un accès en écriture à Amazon SNS.

8. Sélectionnez Enregistrer les modifications.

Configuration des préférences (AWS SDKs)

Pour définir vos préférences en matière de SMS à l'aide de l'un des AWS SDKs, 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

des SMS, consultez la section Set SMSAttributes dans le manuel Amazon Simple Notification Service API Reference.

Les exemples de code suivants illustrent 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 par défautSMSType .

```
//! 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 plus de détails sur l'API, voir Set SMSAttributes in AWS SDK pour C++ API Reference.

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 de détails sur l'API, voir Set SMSAttributes in AWS CLI Command Reference.

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 plus de détails sur l'API, voir Set SMSAttributes in AWS SDK for Java 2.x API Reference.

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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, voir Set SMSAttributes in AWS SDK pour JavaScript API Reference.

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 pour PHP.
- Pour plus de détails sur l'API, voir Set SMSAttributes in AWS SDK pour PHP API Reference.

Configuration des préférences de messagerie SMS pour une livraison spécifique au pays

Vous pouvez gérer et contrôler votre trafic SMS en n'envoyant des messages qu'à des pays de destination spécifiques. Cela garantit que vos messages sont envoyés uniquement aux pays approuvés, évitant ainsi des frais de SMS indésirables. Les instructions suivantes utilisent la configuration Protect d'Amazon Pinpoint pour spécifier les pays que vous souhaitez autoriser ou bloquer.

1. Ouvrez la AWS SMS console à l'adresse https://console.aws.amazon.com/sms-voice/.
2. Dans le volet de navigation, sous Vue d'ensemble, dans la section Démarrage rapide, choisissez Créer une configuration de protection.
3. Sous Détails de configuration de Protect, entrez un nom convivial pour votre configuration de protection (par exemple, Allow-Only-AU).
4. Dans les règles nationales relatives aux SMS, cochez la case Région/Pays pour bloquer l'envoi de messages vers tous les pays pris en charge.
5. Décochez les cases correspondant aux pays dans lesquels vous souhaitez envoyer des messages. Par exemple, pour autoriser les messages uniquement destinés à l'Australie, décochez la case Australie.
6. Dans la section Protéger les associations de configuration, sous Type d'association, sélectionnez Compte par défaut. Cela garantira que la configuration AWS End User Messaging SMS Protect affecte tous les messages envoyés via Amazon SNS, Amazon Cognito et l'appel d'API Amazon Pinpoint. `SendMessages`
7. Choisissez Créer une configuration de protection pour enregistrer vos paramètres.

Le message de confirmation suivant s'affiche :

```
Success Protect configuration protect-abc0123456789 has been created.
```

8. Connectez-vous à la console Amazon SNS.
9. Publiez un message dans l'un des pays bloqués, comme l'Inde.

   Le message ne sera pas délivré. Vous pouvez vérifier cela dans les journaux des échecs de livraison à l'aide de CloudWatch. Recherchez un groupe de journaux sns/region/AccountID/DirectPublishToPhoneNumber/Failurepour obtenir une réponse similaire à l'exemple suivant :

```
{
"notification": {
"messageId": "bd59a509-XXXX-XXXX-82f8-fbdb8cb68217",
"timestamp": "YYYY-MM-DD XX:XX:XX.XXXX“
},
"delivery": {
"destination": "+91XXXXXXXXXX",
"smsType": "Transactional",
"providerResponse": "Cannot deliver message to the specified destination country",
"dwellTimeMs": 85
},
"status": "FAILURE"
}
```

Gestion des numéros de téléphone et des abonnements Amazon SNS

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.

Désactivation de la réception des SMS

Lorsque les lois et réglementations locales (telles que les États-Unis et le Canada) l'exigent, les destinataires de SMS peuvent utiliser leur appareil pour se désinscrire en répondant au message par l'un des moyens suivants :

- ARRÊTER (français)
- ANNULER
- FIN
- REFUSER
- REFUSER
- SORTIR
- SUPPRIMER
- ARRÊTER
- TD
- SE DÉSABONNER

Pour se désinscrire, le destinataire doit répondre au même numéro d'origine qu'Amazon SNS a utilisé pour envoyer le message. Après votre désinscription, 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 désinscription ne supprime pas l'abonnement, mais les SMS ne seront pas envoyés à cet abonnement, sauf si vous avez activé le numéro de téléphone.

Gestion des numéros de téléphone et des abonnements à l'aide de la console Amazon SNS

Vous pouvez utiliser la console Amazon SNS pour contrôler quels numéros de téléphone recevront des SMS de votre compte.

Inscription d'un numéro de téléphone qui a été désactivé sur la console Amazon SNS

Vous pouvez voir quels numéros de téléphone n'ont pas été autorisés à recevoir des SMS de votre compte, et vous pouvez activer ces numéros de téléphone pour recommencer à leur envoyer des messages.

Vous ne pouvez saisir 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 Messagerie texte (SMS) mobile, dans la section Numéros de téléphone désactivés, les numéros de téléphone désactivés sont affichés.
5. Cochez la case correspondant au numéro de téléphone que vous souhaitez inscrire, puis choisissez Se connecter. Le numéro de téléphone n'est plus désactivé et recevra les SMS que vous lui enverrez.

Supprimer un abonnement SMS sur la console Amazon SNS

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.

Supprimer un sujet dans la console Amazon SNS

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 à l'aide du AWS SDK

Vous pouvez l'utiliser AWS SDKs 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 SDKs et Tools.

Afficher tous les numéros de téléphone désactivés à l'aide du SDK AWS

Pour consulter tous les numéros de téléphone désactivés, soumettez une `ListPhoneNumbersOptedOut` demande à l'aide de l'API Amazon SNS.

Les exemples de code suivants illustrent 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 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.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 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;


/**
 * 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 pour PHP.
- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK pour PHP API.

Vérifier si un numéro de téléphone est désactivé à l'aide du SDK AWS

Pour vérifier si un numéro de téléphone est désactivé, soumettez une `CheckIfPhoneNumberIsOptedOut` demande à l'aide de l'API Amazon SNS.

Les exemples de code suivants illustrent comment utiliser `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK pour .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.

```
    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 pour .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 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.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 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 { 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK pour JavaScript API.

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;


/**
 * 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 pour PHP.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK pour PHP API.

Inscription d'un numéro de téléphone qui a été désactivé à l'aide de l'API Amazon SNS

Pour activer un numéro de téléphone, soumettez une `OptInPhoneNumber` demande à l'aide de l'API Amazon SNS.

Vous ne pouvez saisir un numéro de téléphone qu'une fois tous les 30 jours.

Supprimer un abonnement SMS à l'aide du AWS SDK

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 illustrent comment utiliser `Unsubscribe`.

.NET

SDK pour .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.

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 pour .NET .

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 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 pour 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 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.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 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 { 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 pour JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK pour JavaScript .

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 unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- 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 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 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 pour PHP.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK pour PHP .

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_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 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->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.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

```
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Pour plus de détails sur l'API, voir Désabonnement dans le AWS SDK pour une référence à l'API Swift.

Supprimer un sujet à l'aide du AWS SDK

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 illustrent comment utiliser `DeleteTopic`.

.NET

SDK pour .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 pour .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 pour 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.

```
import (
```

```
 "context"
 "encoding/json"
 "log"

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

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



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

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour 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.fromEnvironment { 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 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 pour 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.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

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

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

Surveillance de l'activité des SMS Amazon SNS

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 résumant les statistiques dans la console, en envoyant des informations à Amazon CloudWatch et en envoyant des rapports quotidiens sur l'utilisation des SMS à un compartiment Amazon S3 que vous spécifiez.

Afficher les statistiques de livraison des SMS Amazon SNS

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 Abonnement aux rapports quotidiens d'utilisation des SMS d'Amazon SNS.

Surveillance de la livraison des SMS Amazon SNS avec les CloudWatch métriques et les journaux Amazon

Vous pouvez utiliser Amazon CloudWatch et Amazon CloudWatch Logs pour surveiller la livraison de vos SMS.

Afficher les CloudWatch statistiques Amazon

Amazon SNS collecte automatiquement les statistiques relatives à la livraison de vos SMS et les transmet à Amazon. CloudWatch Vous pouvez les utiliser CloudWatch pour surveiller ces mesures et créer des alarmes pour vous avertir lorsqu'une métrique franchit un seuil. Par exemple, vous pouvez surveiller CloudWatch les statistiques pour connaître votre taux de livraison de SMS et vos frais de month-to-date SMS.

Pour plus d'informations sur la surveillance CloudWatch des métriques, le paramétrage des CloudWatch alarmes et les types de métriques disponibles, consultezSurveillance des rubriques Amazon SNS à l'aide de CloudWatch.

Affichage des CloudWatch journaux

Vous pouvez collecter des informations sur les envois de SMS réussis et infructueux en autorisant Amazon SNS à écrire sur 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 CloudWatch les journaux de 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 le taux d'échantillonnage de réussite, spécifiez le pourcentage de livraisons de SMS réussies pour lesquelles Amazon SNS rédigera 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 plus d'informations sur la création de rôles IAM, consultez la section Création d'un rôle pour un AWS service dans le Guide de l'utilisateur IAM.
8. Sélectionnez 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

Abonnement aux rapports quotidiens d'utilisation des SMS d'Amazon SNS

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 24 heures pour que le rapport d'utilisation des SMS soit disponible dans le compartiment Amazon S3.

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 la console Compte AWS qui envoie des SMS, connectez-vous à la console 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 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 obtenir un exemple, consultez Exemple de politique de compartiment.

   Si vous utilisez l'exemple de politique, n'oubliez pas de le *`my-s3-bucket`* remplacer par le nom du bucket 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. Sélectionnez 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 dont les principaux responsables ont eu accès aux ressources de votre compte. Lorsque le principal d'une déclaration de politique relative aux compartiments Amazon S3 est un problème d'adjoint 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.

JSON

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:us-west-1:123456789012:*"
                }
            }
        },
        {
            "Sid": "AllowGetBucketLocation",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:GetBucketLocation",
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:us-west-1:123456789012:*"
                }
            }
        },
        {
            "Sid": "AllowListBucket",
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "account_id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:sns:us-west-1:123456789012:*"
                }
            }
        }
    ]
}
```

> **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 appartenant à Compte AWS un autre, consultez Comment puis-je copier des objets Amazon S3 d'un autre compartiment 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
```

Demande d'assistance pour la messagerie SMS Amazon SNS

> **Important**
>
> Le guide du développeur de SMS Amazon SNS a été mis à jour. Amazon SNS s'est intégré à AWS End User Messaging SMSla livraison de messages SMS. Ce guide contient les dernières informations sur la création, la configuration et la gestion de vos messages SMS Amazon SNS.

Certaines options de SMS avec Amazon SNS ne sont pas disponibles pour votre AWS compte tant que vous ne nous contactez pas. 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 Sortir du sandbox SMS d'Amazon SNS.
- Un numéro d'origine dédié
- Un identifiant d'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. Support pour les expéditeurs IDs varie en fonction du pays ou de la région. Pour plus d'informations, consultez la section Pays et régions pris en charge pour la messagerie SMS AWS End User Messaging SMS dans le Guide de AWS End User Messaging SMS l'utilisateur.

Demande d'augmentation de votre quota de dépenses mensuel pour les SMS 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. Pour demander une augmentation du quota de dépenses par SMS pour votre compte, ouvrez une demande d'augmentation du quota dans le AWS Support Center.

Rubriques

- Étape 1 : ouvrir une demande de SMS Amazon SNS
- Étape 2 : mettre à jour vos paramètres de SMS dans la console Amazon SNS

Étape 1 : ouvrir une demande de SMS Amazon SNS

Vous pouvez demander une augmentation de votre quota de dépenses mensuel en ouvrant un dossier d'augmentation de quota dans le AWS Support Center.

> **Note**
>
> Certains champs du formulaire de demande sont marqués comme « facultatifs ». Toutefois, 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 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 Type de limite, sélectionnez SNS Messagerie texte.
   - (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 AWS la région à partir de laquelle vous allez 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 maximum attendu transactions-per-second (TPS) pour votre messagerie.
- 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 d'assistance pour la messagerie SMS 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' 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.

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 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 le problème persiste, ouvrez un dossier dans le AWS Support Center.

Lorsque vous créez votre dossier dans le Support Centre, veillez à inclure toutes les informations requises pour le type de demande que vous soumettez. Dans le cas contraire, vous Support devez 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.

Pour plus d'informations sur l'expéditeur IDs, consultez la documentation suivante dans le guide de AWS End User Messaging SMS l'utilisateur :

AWS End User Messaging SMS Rubrique	Description
Demande d'augmentation du quota de dépenses	Votre quota de dépenses détermine le montant que vous pouvez dépenser pour envoyer des SMS AWS End User Messaging SMS chaque mois.
Ouvrez un dossier dans le centre d'assistance pour obtenir un identifiant d'expéditeur	Si vous prévoyez d'envoyer des messages à des destinataires dans un pays où IDs les expéditeurs sont obligatoires, vous pouvez demander un identifiant d'expéditeur en créant un nouveau dossier dans le Support Centre.

Bonnes pratiques pour la messagerie SMS Amazon SNS

> **Important**
>
> Le guide du développeur de SMS Amazon SNS a été mis à jour. Amazon SNS s'est intégré à AWS End User Messaging SMSla livraison de messages SMS. Ce guide contient les dernières informations sur la création, la configuration et la gestion de vos messages SMS Amazon SNS.

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.

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 court 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 travail 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 de l'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 travail opt-in, vous pouvez fournir une capture d'écran de votre script d'opt-in, de votre matériel marketing ou de votre base de données dans laquelle les numéros 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. Dans ces situations, il vous Support contacte pour vous fournir une liste des informations dont le transporteur ou l'agence a besoin. 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 character-per-message limite de 160 signifie que vos messages doivent être 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 person-to-person message

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 messages SMS, les messages très longs URLs peuvent être répartis sur plusieurs messages. Vous devez utiliser des domaines de redirection pour fournir des informations abrégées URLs. 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 abrégés, consultez la section Demande de codes abrégés pour la messagerie SMS AWS End User Messaging SMS dans le Guide de AWS End User Messaging SMS l'utilisateur.

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 des SMS, consultez la section Tarification des SMS AWS dans le monde entier.

Lorsqu'Amazon SNS accepte une demande d'envoi de SMS (suite à un appel à l'SendMessagesAPI, ou suite au 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 l'utiliser AWS End User Messaging SMS pour envoyer un message test afin de déterminer si un numéro de téléphone est valide et de quel

type de numéro il s'agit (mobile, fixe ou VoIP). Pour plus d'informations, voir Envoyer un message de test avec le simulateur de SMS dans le guide de AWS End User Messaging SMS l'utilisateur.

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 en plusieurs versions. 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 abrégé pour envoyer des SMS à des destinataires aux États-Unis d'Amérique, vous devez demander des codes abrégés distincts pour Région AWS chacun des messages que vous comptez 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, qu'ils soient identiques Région AWS ou répartis sur plusieurs Régions AWS, fournit une couche supplémentaire de redondance, ce qui peut contribuer à améliorer la résilience.

Limites et restrictions pour les SMS

Pour connaître les limites et restrictions relatives aux SMS, consultez la section Limites et restrictions relatives aux SMS et MMS dans le guide de AWS End User Messaging SMS l'utilisateur.

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 CreatePoolla section Référence des AWS End User Messaging SMS API.

PutKeyword

Utilisez l'action `PutKeyword` API pour créer ou mettre à jour une configuration de mots clés sur un numéro de téléphone ou un pool d'origine. Pour plus d'informations, consultez PutKeywordla section Référence des AWS End User Messaging SMS API.

Gestion des paramètres des numéros

Pour gérer les paramètres des codes courts et longs dédiés que vous avez demandés au AWS Support et attribués à votre compte, voir Modifier les fonctionnalités d'un numéro de téléphone avec AWS CLI l'entrée AWS End User Messaging SMS.

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	s, sem	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	&	*	@	:	,	¤

Caractères standard GSM 03.38												
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	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.

Envoi de notifications push mobiles avec Amazon SNS

Vous pouvez utiliser Amazon SNS pour envoyer des messages de notification push directement aux applications sur les appareils mobiles. Les messages de notification push envoyés à un terminal mobile peuvent apparaître dans l'application mobile sous forme d'alertes, de mises à jour de badges ou d'alertes sonores.

Rubriques

- Comment fonctionnent les notifications utilisateur Amazon SNS
- Configuration des notifications push avec Amazon SNS
- Configuration d'une application mobile sur Amazon SNS
- Utilisation d'Amazon SNS pour les notifications push mobiles
- Attributs de l'application mobile Amazon SNS
- Notifications d'événements liés aux applications Amazon SNS pour les applications mobiles
- Actions d'API push mobile
- Erreurs courantes de l'API push mobile Amazon SNS
- Utilisation de l'attribut Amazon SNS time to live message pour les notifications push mobiles
- Régions prises en charge par l'application mobile Amazon SNS
- Bonnes pratiques pour gérer les notifications push mobiles Amazon SNS

Comment fonctionnent les notifications utilisateur Amazon SNS

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)
- Service de notification push Apple (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 notifications push, tels que APNs FCM, maintiennent une connexion avec chaque application et appareil mobile associé enregistrés 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 de plus amples informations, veuillez consulter Configuration des notifications push avec Amazon SNS.

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.

Configuration des notifications push avec Amazon SNS

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 de plus amples informations, veuillez consulter Création d'une application de plateforme Amazon SNS.
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 de plus amples informations, veuillez consulter Configuration d'un point de terminaison sur la plateforme Amazon SNS pour les notifications mobiles.
5. Utilisez le `EndpointArn` pour publier un message dans une application sur un appareil mobile. Pour plus d'informations, consultez Messagerie directe sur appareil mobile Amazon SNS et l'API Publish dans la Référence d'API Amazon Simple Notification Service.

Configuration d'une application mobile sur Amazon SNS

Cette rubrique décrit comment configurer des applications mobiles à l' AWS Management Console aide des informations décrites dansConditions préalables pour les notifications utilisateur Amazon SNS.

Conditions préalables pour les notifications utilisateur Amazon SNS

Pour commencer à utiliser les notifications push mobiles Amazon SNS, vous aurez besoin des éléments suivants :

- Ensemble d'informations d'identification permettant de se connecter 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 l'utilisez APNs comme service de notification 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 de l'appareil (encore une fois avec quelques paramètres supplémentaires) sont représentés sous forme d'objets appelés points de terminaison de plate-forme. 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 requises, vous pouvez envoyer un message de notification push en utilisant le AWS Management Console ou le push mobile Amazon SNS. APIs Pour de plus amples informations, veuillez consulter Configuration des notifications push avec Amazon SNS.

Création d'une application de plateforme Amazon SNS

Pour envoyer des notifications depuis Amazon SNS aux terminaux mobiles, que ce soit directement ou par le biais d'abonnements à un sujet, vous devez d'abord créer une application de plateforme. Après avoir enregistré l'application AWS, vous devez créer un point de terminaison pour l'application et pour l'appareil mobile. Ce point de terminaison permet à Amazon SNS d'envoyer des messages à l'appareil.

Pour créer une application de plateforme

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation, sélectionnez Notifications push.
3. Dans la section Platform applications (Applications de plateforme), choisissez Create platform application (Créer une application de plateforme).
4. Choisissez votre Région AWS. Pour obtenir une liste des Régions AWS dans lesquelles vous pouvez créer des applications mobiles, consultez la section Régions prises en charge par l'application mobile Amazon SNS.
5. Entrez les informations de candidature suivantes :

- Nom de l'application : donnez un nom à l'application de votre plateforme. Le nom doit comporter entre 1 et 256 caractères et peut contenir des lettres majuscules et minuscules, des chiffres, des traits de soulignement, des traits d'union et des points.
- Plateforme de notifications push : sélectionnez le service de notification approprié auprès duquel l'application est enregistrée (par exemple, Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM)).

6. En fonction de la plateforme que vous avez sélectionnée, vous devrez fournir des informations d'identification spécifiques :

   - Pour APNs(Apple Push Notification Service) : choisissez entre une authentification basée sur des jetons ou une authentification basée sur des certificats.
     - Pour l'authentification basée sur des jetons, téléchargez un fichier .p8 (généré via Keychain Access).
     - Pour l'authentification par certificat, téléchargez un fichier .p12 (également exporté depuis Keychain Access).
   - Pour FCM (Firebase Cloud Messaging) : entrez la clé du serveur depuis la console Firebase.
   - Pour les autres plateformes (telles que ADM ou GCM), entrez les clés d'API ou les informations d'identification correspondantes.
7. Après avoir saisi les informations nécessaires, choisissez Créer une application de plateforme. Cette action enregistre l'application auprès d'Amazon SNS et crée l'objet d'application de plateforme correspondant.
8. Lors de la création, Amazon SNS génère et renvoie un (`PlatformApplicationArn`Amazon Resource Name). Cet ARN identifie de manière unique l'application de votre plateforme et est utilisé lors de la création de points de terminaison pour les appareils mobiles.

Configuration d'un point de terminaison sur la plateforme Amazon SNS pour les notifications mobiles

Lorsqu'une application et un appareil mobile s'enregistrent auprès d'un service de notification push (tel que APNs Firebase Cloud Messaging), le service de notification push renvoie un jeton d'appareil. Amazon SNS utilise ce jeton d'appareil pour créer un point de terminaison de plateforme, qui sert de cible pour envoyer des messages de notification push directs à l'application installée sur l'appareil. Le point de terminaison de la plateforme sert de pont et achemine les messages envoyés par Amazon SNS vers le service de notification push pour qu'ils soient transmis à l'appareil mobile correspondant.

Pour plus d’informations, consultez Conditions préalables pour les notifications utilisateur Amazon SNS et Configuration des notifications push avec Amazon SNS.

Comprendre les jetons des appareils et les points de terminaison de la plateforme

Un jeton d'appareil identifie de manière unique un appareil mobile enregistré auprès d'un service de notification push (par exemple APNs, Firebase Cloud Messaging). Lorsqu'une application s'enregistre auprès du service de notification push, elle génère un jeton spécifique à cette application et à cet appareil. Amazon SNS utilise ce jeton d'appareil pour créer un point de terminaison de plateforme dans l'application de plateforme correspondante.

Le point de terminaison de la plateforme permet à Amazon SNS d'envoyer des messages de notification push à l'appareil via le service de notification push, en maintenant la connexion entre votre application et l'appareil de l'utilisateur.

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 la plateforme existe déjà, ne le créez pas à nouveau. Renvoyez à l'appelant l'ARN du point de terminaison de plateforme existant.
- Si le point de terminaison de la plateforme avec le même jeton d'appareil mais des paramètres différents existe déjà, ne le créez pas à nouveau. Envoyez une exception à l'appelant.
- Si le point de terminaison de la 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
      (GetEndpointAttributes 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. Il est également appelé si l'appel `GetEndpointAttributes` d'action initial échoue avec une exception introuvable, comme cela se produirait si l'application connaît son ARN de point de terminaison mais qu'il a été supprimé.
- L'`GetEndpointAttributes`action est appelée pour vérifier l'état du point de terminaison de la plate-forme même si le point de terminaison de la plate-forme 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 le. AWS SDKs

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

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 device token or registration ID of the mobile device.
This is a unique
               identifier provided by the device platform (e.g., Apple Push
Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
push notifications.

               platformApplicationArn - The ARN value of platform application.
You can get this value from the AWS Management Console.\s

            """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
response.endpointArn());
```

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

Pour de plus amples informations, veuillez consulter 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 points de terminaison Firebase Cloud Messaging par Amazon SNS

Réactivation d'un point de terminaison de plateforme associé à un jeton d'appareil non valide

Lorsqu'une plateforme mobile (telle que APNs 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 associé 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.

Intégration de jetons d'appareils à Amazon SNS pour les notifications mobiles

Lorsque vous enregistrez pour la première fois une application et un appareil mobile auprès d'un service de notification, tel que Apple Push Notification Service (APNs) et Firebase Cloud Messaging (FCM), les jetons d'appareil ou l'enregistrement IDs sont renvoyés par le service. Ces jetons/ IDs sont ajoutés à Amazon SNS afin de créer un point de terminaison pour l'application et l'appareil, à l'aide de l'API. `PlatformApplicationArn` Une fois le point de terminaison créé, un `EndpointArn`est renvoyé, qu'Amazon SNS utilise pour diriger les notifications vers l'application/l'appareil approprié.

Vous pouvez ajouter des jetons d'appareil ou un enregistrement IDs sur Amazon SNS de la manière suivante :

- Ajoutez manuellement un seul jeton via le AWS Management Console

- Chargement de plusieurs jetons à l'aide de l'API `CreatePlatformEndpoint`
- Enregistrez des jetons pour les futurs appareils

Pour ajouter manuellement un jeton d'appareil ou un identifiant d'enregistrement

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation, sélectionnez 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, suivez le Création d'une application de plateforme Amazon SNS guide pour le faire dès maintenant.
4. Choisissez Créer un point de terminaison.
5. Dans le champ Endpoint Token, entrez le jeton ou l'ID d'enregistrement, selon le service de notification que vous utilisez (par exemple, l'ID d'enregistrement FCM).
6. (Facultatif) Entrez des données supplémentaires dans le champ Données utilisateur. Ces données doivent être codées en UTF-8 et être inférieures à 2 Ko.
7. Choisissez Créer un point de terminaison.

Une fois le point de terminaison créé, vous pouvez envoyer des messages directement à l'appareil mobile ou aux appareils mobiles abonnés à une rubrique Amazon SNS.

Pour télécharger plusieurs jetons à l'aide de l'**`CreatePlatformEndpoint`**API

Les étapes suivantes montrent comment utiliser l'exemple d'application Java (`bulkuploadpackage`) fourni par AWS pour télécharger plusieurs jetons (jetons d'appareil ou enregistrement IDs) sur 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. Les étapes supposent que vous avez installé le AWS SDK pour Java et que vous disposez des informations d'identification AWS de sécurité pour votre Compte AWS. Pour de plus amples informations, veuillez consulter AWS SDK pour Java. Pour plus d'informations sur les informations d'identification, consultez les informations d'identification AWS de sécurité dans le guide de l'utilisateur IAM.

1. Téléchargez et décompressez le fichier snsmobilepush.zip.
2. Créez un nouveau projet Java dans Eclipse et importez le `SNSSamples` dossier dans le projet.
3. Téléchargez la bibliothèque OpenCSV et ajoutez-la au chemin de compilation.
4. Dans le `BulkUpload.properties` fichier, spécifiez les éléments suivants :

   - Votre `ApplicationArn` (ARN de l'application de plateforme).
   - Le chemin absolu vers votre fichier CSV contenant les jetons.
   - Enregistrement des noms de fichiers pour les jetons réussis et échoués. Par exemple : `goodTokens.csv` et `badTokens.csv`.
   - (Facultatif) Configuration du délimiteur, du guillemet et du nombre de fils à utiliser.

   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
   ```

5. Exécutez l'application BatchCreatePlatformEndpointSample.java pour télécharger les jetons sur Amazon SNS. Les jetons téléchargés avec succès seront connectés`goodTokens.csv`, tandis que les jetons mal formés seront connectés. `badTokens.csv`

Pour enregistrer des jetons provenant d'appareils pour de futures installations d'applications

Deux options s'offrent à vous pour ce processus :

Utiliser le service Amazon Cognito

Votre application mobile peut utiliser des informations d'identification de sécurité temporaires pour créer des points de terminaison. Il est recommandé d'utiliser Amazon Cognito pour générer des informations d'identification temporaires. Pour plus d'informations, consultez le guide du développeur Amazon Cognito

Pour suivre les inscriptions aux applications, utilisez les événements Amazon SNS pour recevoir des notifications lors de la création d'un nouveau point de terminaison ARNs .

Vous pouvez également utiliser l'`ListEndpointByPlatformApplication`API pour récupérer la liste des points de terminaison enregistrés.

Utilisation d'un serveur proxy

Si l'infrastructure de votre application prend déjà en charge l'enregistrement des appareils lors de l'installation, vous pouvez utiliser votre serveur comme proxy. Il transmettra les jetons de l'appareil à Amazon SNS via l'`CreatePlatformEndpoint`API.

L'ARN du point de terminaison créé par Amazon SNS sera renvoyé et pourra être stocké par votre serveur pour la publication de futurs messages.

Amazon SNS Méthodes d'authentification par notification push 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

Clé de signature privée utilisée par Amazon SNS pour signer les jetons d'authentification Apple Push Notification Service (APNs).

Si vous fournissez une clé de signature, Amazon SNS utilise un jeton pour s'authentifier APNs pour chaque notification push que vous envoyez. Avec votre clé de signature, vous pouvez envoyer des notifications push aux environnements de APNs production et de sandbox.

Votre clé de signature n'expire pas, et vous pouvez l'utiliser pour plusieurs appli. Pour plus d'informations, consultez la section Communiquer avec APNs des jetons d'authentification dans la section d'aide relative au compte développeur du site Web d'Apple.

Certificat

Certificat TLS utilisé par Amazon SNS pour s'authentifier lorsque vous envoyez APNs 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 Etablissement d'une connexion basée sur des certificats APNs sur le site Web des développeurs Apple.

Pour gérer les APNs paramètres à l'aide de la console AWS de gestion

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation, sélectionnez Notifications push.
3. Dans la section Applications de plate-forme, sélectionnez l'application dont vous souhaitez modifier les APNs paramètres, puis choisissez Modifier. Si vous n'avez pas encore créé d'application de plateforme, suivez le Création d'une application de plateforme Amazon SNS guide pour le faire dès maintenant.
4. Choisissez Modifier pour modifier les paramètres de votre application de plateforme.
5. Dans la section Type d'authentification, choisissez l'une des options suivantes :
   - Authentification basée sur des jetons (recommandée pour les intégrations modernes APNs)
   - Authentification basée sur des certificats (ancienne méthode)
6. Configurez vos informations d'identification en fonction du type d'authentification :
   - Pour l'authentification basée sur des jetons :
     - Téléchargez le fichier .p8, qui est la clé de signature du jeton d'authentification que vous avez téléchargée depuis votre compte Apple Developer.
     - Entrez l'identifiant de la clé de signature que vous trouverez dans votre compte Apple Developer. Accédez à Certificats IDs, profils, clés, puis sélectionnez la clé que vous souhaitez utiliser.
     - Fournissez l'identifiant d'équipe de votre compte Apple Developer. Vous pouvez le trouver sur la page d'adhésion.
     - Entrez l'identifiant du bundle attribué à votre application. Vous pouvez le trouver sous Certificats IDs et profils, App IDs.
   - Pour l'authentification basée sur des certificats :
     - Téléchargez le fichier .p12 pour votre certificat TLS. Ce fichier peut être exporté depuis Keychain Access sur macOS après avoir téléchargé le certificat depuis votre compte Apple Developer.
     - Si vous avez attribué un mot de passe à votre certificat .p12, saisissez-le ici.
7. Après avoir saisi les informations d'identification nécessaires, choisissez Enregistrer les modifications pour mettre à jour les paramètres.

Configuration de l'authentification de l'intégration d'Amazon SNS avec Firebase Cloud Messaging

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.

> **Important**
>
> 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.
> 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.
> 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.

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. Le FCM Legacy APIs sera supprimé 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 la section Migrer de l'ancienne FCM APIs vers HTTP v1 dans la documentation de Google Firebase.

Gestion des paramètres FCM à l'aide de la CLI

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 à l'aide de la console

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 suivantes pour saisir et gérer vos informations d'identification Firebase Cloud Messaging (FCM) dans Amazon SNS.

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation, sélectionnez Notifications push.
3. Dans la section Applications de plate-forme, sélectionnez l'application de plate-forme FCM dont vous souhaitez modifier les informations d'identification, puis choisissez Modifier.
4. Dans la section Informations d'identification de Firebase Cloud Messaging, choisissez l'une des options suivantes :
   - Authentification basée sur des jetons (méthode recommandée) : téléchargez le fichier de clé privée (JSON) que vous avez téléchargé depuis la console Firebase. Ce fichier contient les informations d'identification nécessaires pour générer des jetons d'accès de courte durée pour les notifications FCM. Pour obtenir ce fichier :
     1. Accédez à la console de votre application Firebase.
     2. Dans les paramètres du projet, sélectionnez Cloud Messaging.
     3. Téléchargez le fichier JSON de clé privée (à utiliser dans la méthode d'authentification basée sur des jetons).
   - Authentification par clé d'API — Si vous préférez utiliser l'ancienne méthode d'authentification par clé d'API, entrez la clé d'API Google dans le champ prévu à cet effet. Pour obtenir ce fichier :
     1. Accédez à la console de votre application Firebase.
     2. Dans les paramètres du projet, sélectionnez Cloud Messaging.
     3. Copiez la clé du serveur (clé API) à utiliser pour envoyer des notifications.
5. Lorsque vous avez terminé, choisissez Enregistrer les modifications.

Voir aussi

- Utilisation des charges utiles de Google Firebase Cloud Messaging v1 dans Amazon SNS

Gestion des points de terminaison Firebase Cloud Messaging par Amazon SNS

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, le point de terminaison ARNs Amazon SNS correspondant et les horodatages sur votre serveur d'applications.
2. Supprimez tous les jetons périmés et supprimez le point de terminaison Amazon ARNs SNS correspondant.

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 d'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 `FailureType` de`InvalidNotification`, et le corps *`FailureMessage`* de notification 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.

Utilisation d'Amazon SNS pour les notifications push mobiles

Cette section décrit comment envoyer un message de notification push mobile.

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.

Messagerie directe sur appareil mobile Amazon SNS

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 de notifications Amazon SNS avec des charges utiles spécifiques à la plate-forme

Vous pouvez utiliser Amazon SNS AWS Management Console ou Amazon APIs 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 d'Amazon SNS APIs, consultez la section Actions d'API push mobile et le `SNSMobilePush.java` fichier dans. `snsmobilepush.zip`

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 clé 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 des appareils pour plusieurs plateformes, telles que FCM APNs, vous devez d'abord abonner les points de terminaison mobiles à une rubrique sur Amazon SNS, puis publier le message dans cette rubrique.

L'exemple suivant montre un message à envoyer aux terminaux 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 APNs sous forme d'alerte ou de notifications en arrière-plan

Amazon SNS peut envoyer des messages `alert` ou des `background` notifications (pour plus d'informations, consultez la section Envoyer des mises à jour en arrière-plan à votre application dans la APNs documentation). APNs

- Une `alert` APNs notification informe l'utilisateur en affichant un message d'alerte, en émettant un son ou en ajoutant un badge à l'icône de votre application.
- Une `background` APNs notification active ou demande à votre application d'agir en fonction du contenu de la notification, sans en informer l'utilisateur.

Spécification de valeurs APNs d'en-tête 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 SDKs, 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
```

> **Note**
>
> Assurez-vous que la structure JSON est valide. Ajoutez une virgule après chaque paire clé-valeur, sauf la dernière.

Déduire l'en-tête de type APNs push à partir de la charge utile

Si vous ne définissez pas l'`apns-push-type` APNs en-tête, Amazon SNS définit l'en-tête `background` selon `alert` ou selon la `content-available` clé du `aps` dictionnaire de votre configuration de charge utile au format JSON APNs .

> **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 APNs sous forme de notification de fond uniquement, vous devez inclure `content-available` set to `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'`apns-push-type` APNs en-tête du message sur. `background` 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 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.

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 de l'ancienne FCM APIs 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 de l'ancienne FCM APIs 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 APNs appareils. Si vous envoyez actuellement des notifications silencieuses à vos APNs appareils, 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 APNs en-têtes et leur utilité, consultez la section Communiquer avec APNs 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 terminal. 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 le point de terminais

Code d'erreur/d'état FCM	Type de panne Amazon SNS	Message d'échec	Cause et atténuation
			on 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 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 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 de l'application mobile Amazon SNS

Amazon Simple Notification Service (Amazon SNS) prend en charge la journalisation de l'état de diffusion des messages de notification push. Après avoir configuré les attributs de l'application, les entrées du journal seront envoyées à CloudWatch Logs pour les messages envoyés depuis Amazon SNS vers les terminaux mobiles. 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 de l'application en fonction de l'état de livraison des messages AWS Management Console, vous pouvez utiliser les kits de développement AWS logiciel (SDKs) ou l'API de requête.

Configuration des attributs d'état de livraison des messages à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Mobile, puis Notifications Push.
3. Dans la section Applications de plate-forme, choisissez l'application qui contient les points de terminaison pour lesquels vous souhaitez recevoir des CloudWatch journaux.
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. Choisissez Autoriser pour accorder à Amazon SNS l'accès en écriture afin qu'il puisse utiliser CloudWatch Logs en votre nom.
7. Maintenant, revenez dans la boîte de dialogue État de livraison, entrez un nombre dans le champ Pourcentage de réussite par rapport à l'échantillon (0-100) pour le pourcentage de messages envoyés avec succès pour lesquels vous souhaitez recevoir des CloudWatch journaux.

> **Note**
>
> Une fois que vous avez configuré les attributs de l'application pour l'état de livraison des messages, tous les messages ayant échoué génèrent des CloudWatch journaux.

8. Enfin, sélectionnez Enregistrer la configuration. 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.

Exemples de journaux d'état CloudWatch de livraison des messages Amazon SNS

Une fois que vous avez configuré les attributs d'état de livraison des messages pour un point de terminaison d'application, CloudWatch des journaux 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 d'état de livraison des messages à l'aide du AWS SDKs

Ils AWS SDKsfournissent APIs en plusieurs langues l'utilisation des attributs d'état de livraison des messages 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 `FailureFeedbackRoleArn` attributs `SuccessFeedbackRoleArn` et sont utilisés pour donner à Amazon SNS un accès en écriture lui permettant d'utiliser 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'`FailureFeedbackRoleArn`attribut, toutes les livraisons de messages ayant échoué génèrent des CloudWatch journaux.

```
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 pour 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.
Service de notification push Apple (APNs)	Voir la réponse HTTP/2 de APNs la section Communiquer avec APNs dans le guide de programmation des notifications locales et distantes.
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.
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.

Notifications d'événements liés aux applications Amazon SNS pour les applications mobiles

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 inclure la prise en charge d'un service de notification push tel que 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 du. AWS CLI AWS SDKs

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.
`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'informa

Nom d’attribut	Déclencheur de notification
	tions, 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 du. AWS SDKs

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. Entrez ARNs les sujets 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 SDKs

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 le service push mobile Amazon SNS APIs, vous devez d'abord remplir les conditions requises pour le service de notification push, telles que le service de notification push Apple (APNs) et 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 à un appareil à l'aide de APIs, vous devez d'abord utiliser l'`CreatePlatformApplication`action, qui renvoie un `PlatformApplicationArn` attribut. 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 Configuration des notifications push avec Amazon SNS.

Les notifications push d'Amazon SNS sur mobile APIs sont les suivantes :

`CreatePlatformApplication`

Crée un objet d'application de plate-forme pour l'un des services de notification push pris en charge, tel que APNs FCM, auquel les appareils et les applications mobiles peuvent s'enregistrer. 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 terminal doit lancer l' ConfirmSubscription action à l'aide du jeton figurant dans le message de confirmation.

`Unsubscribe`

Supprime un abonnement.

Erreurs courantes de l'API push mobile Amazon SNS

Les erreurs renvoyées par Amazon SNS APIs pour mobile push sont répertoriées dans le tableau suivant. Pour plus d'informations sur Amazon SNS APIs pour mobile push, consultez. 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 une out-of-range valeur a été fournie pour le nom de plate-forme.	400	`CreatePlatformApplication`
APNs — Le principal n'est pas un certificat valide	Un certificat non valide a été fourni pour le APNs principal, à savoir le certificat SSL. Pour plus d'informations, consultez CreatePlatformApplicationle manuel Amazon Simple Notification Service API Reference.	400	`CreatePlatformApplication`
APNs — Le principal est un certificat	Un certificat valide qui n'est pas au format .pem a été	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
valide mais pas au format .pem	fourni pour le APNs principal, à savoir le certificat SSL.		
APNs — Le principal est un certificat expiré	Un certificat expiré a été fourni pour le APNs principal, à savoir le certificat SSL.	400	`CreatePlatformApplication`
APNs — Le principal n'est pas un certificat émis par Apple	Un certificat non émis par Apple a été fourni pour le mandataire APNs qui est le certificat SSL.	400	`CreatePlatformApplication`
APNs — Le principal n'est pas fourni	Le APNs principal, à savoir le certificat SSL, n'a pas été fourni.	400	`CreatePlatformApplication`
APNs — Les informations d'identification ne sont pas fournies	L' APNs identifiant, qui est la clé privée, n'a pas été fourni. Pour plus d'informations, consultez CreatePlatformApplicationle manuel Amazon Simple Notification Service API Reference.	400	`CreatePlatformApplication`
APNs — Les informations d'identification ne sont pas au format .pem valide	L' APNs identifiant, qui est la clé privée, n'est pas dans un format .pem valide.	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
FCM — le serveur n'APIKey est pas fourni	Les informations d'identification FCM, qui sont la clé API, n'ont pas été fournies. Pour plus d'informations, consultez CreatePlatformApplicationle manuel Amazon Simple Notification Service API Reference.	400	`CreatePlatformApplication`
FCM — le serveur APIKey est vide	Les informations d'identification FCM, qui sont la clé API, sont vides.	400	`CreatePlatformApplication`
FCM — le serveur APIKey est une chaîne nulle	Les informations d'identification FCM, qui sont la clé API, sont null.	400	`CreatePlatformApplication`
FCM — le serveur APIKey n'est pas valide	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`

Erreur	Description	Code d'état HTTPS	Action d'API
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`
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 possède un format ARN non valide	EventEndpointCreated possède un format ARN non valide.	400	`CreatePlatformApplication`
EventEndpointDeleted possède un format ARN non valide	EventEndpointDeleted possède un format ARN non valide.	400	`CreatePlatformApplication`
EventEndpointUpdated possède un format ARN non valide	EventEndpointUpdated possède un format ARN non valide.	400	`CreatePlatformApplication`

Erreur	Description	Code d'état HTTPS	Action d'API
EventDeliveryAttemptFailure possède un format ARN non valide	EventDeliveryAttemptFailure possède un format ARN non valide.	400	`CreatePlatformApplication`
EventDeliveryFailure possède un format ARN non valide	EventDeliveryFailure possède un format ARN non valide.	400	`CreatePlatformApplication`
EventEndpointCreated n'est pas un sujet existant	EventEndpointCreated n'est pas un sujet existant.	400	`CreatePlatformApplication`
EventEndpointDeleted n'est pas un sujet existant	EventEndpointDeleted n'est pas un sujet existant.	400	`CreatePlatformApplication`
EventEndpointUpdated n'est pas un sujet existant	EventEndpointUpdated n'est pas un sujet existant.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure n'est pas un sujet existant	EventDeliveryAttemptFailure n'est pas un sujet existant.	400	`CreatePlatformApplication`
EventDeliveryFailure n'est pas un sujet existant	EventDeliveryFailure n'est pas un sujet existant.	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`

Erreur	Description	Code d'état HTTPS	Action d'API
APNs — Le principal n'est pas un certificat valide	Un certificat non valide a été fourni pour le APNs principal, à savoir le certificat SSL. Pour plus d'informations, consultez CreatePlatformApplicationle manuel Amazon Simple Notification Service API Reference.	400	`SetPlatformAttributes`
APNs — Le principal est un certificat valide mais pas au format .pem	Un certificat valide qui n'est pas au format .pem a été fourni pour le APNs principal, à savoir le certificat SSL.	400	`SetPlatformAttributes`
APNs — Le principal est un certificat expiré	Un certificat expiré a été fourni pour le APNs principal, à savoir le certificat SSL.	400	`SetPlatformAttributes`
APNs — Le principal n'est pas un certificat émis par Apple	Un certificat non émis par Apple a été fourni pour le mandataire APNs qui est le certificat SSL.	400	`SetPlatformAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
APNs — Le principal n'est pas fourni	Le APNs principal, à savoir le certificat SSL, n'a pas été fourni.	400	`SetPlatformAttributes`
APNs — Les informations d'identification ne sont pas fournies	L' APNs identifiant, qui est la clé privée, n'a pas été fourni. Pour plus d'informations, consultez CreatePlatformApplicationle manuel Amazon Simple Notification Service API Reference.	400	`SetPlatformAttributes`
APNs — Les informations d'identification ne sont pas au format .pem valide	L' APNs identifiant, qui est la clé privée, n'est pas dans un format .pem valide.	400	`SetPlatformAttributes`
FCM — le serveur n'APIKey est pas fourni	Les informations d'identification FCM, qui sont la clé API, n'ont pas été fournies. Pour plus d'informations, consultez CreatePlatformApplicationle manuel Amazon Simple Notification Service API Reference.	400	`SetPlatformAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
FCM — le serveur APIKey est une chaîne nulle	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`
ADM — clientsecret is a null string	La chaîne requise pour le code secret du client est null.	400	`SetPlatformAttributes`
EventEndpointUpdated possède un format ARN non valide	EventEndpointUpdated possède un format ARN non valide.	400	`SetPlatformAttributes`
EventEndpointDeleted possède un format ARN non valide	EventEndpointDeleted possède un format ARN non valide.	400	`SetPlatformAttributes`
EventEndpointUpdated possède un format ARN non valide	EventEndpointUpdated possède un format ARN non valide.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure possède un format ARN non valide	EventDeliveryAttemptFailure possède un format ARN non valide.	400	`SetPlatformAttributes`

Erreur	Description	Code d'état HTTPS	Action d'API
EventDeliveryFailure possède un format ARN non valide	EventDeliveryFailure possède un format ARN non valide.	400	`SetPlatformAttributes`
EventEndpointCreated n'est pas un sujet existant	EventEndpointCreated n'est pas un sujet existant.	400	`SetPlatformAttributes`
EventEndpointDeleted n'est pas un sujet existant	EventEndpointDeleted n'est pas un sujet existant.	400	`SetPlatformAttributes`
EventEndpointUpdated n'est pas un sujet existant	EventEndpointUpdated n'est pas un sujet existant.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure n'est pas un sujet existant	EventDeliveryAttemptFailure n'est pas un sujet existant.	400	`SetPlatformAttributes`
EventDeliveryFailure n'est pas un sujet existant	EventDeliveryFailure n'est pas un sujet existant.	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`

Erreur	Description	Code d'état HTTPS	Action d'API
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`
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`

Erreur	Description	Code d'état HTTPS	Action d'API
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`
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`

Erreur	Description	Code d'état HTTPS	Action d'API
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`
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 Amazon SNS time to live message pour les notifications push mobiles

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 de spécifier le délai dont dispose le service de notification push, tel que Apple Push Notification Service (APNs) ou FCM, pour transmettre le message au terminal. 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 (SDKs) ou l'API de requête.

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 de l'API AWS SDKs or query :

Service de notification push	Attribut de message TTL
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Service de notification push Apple (APNs)	`AWS.SNS.MOBILE.APNS.TTL`
Sandbox du service de notification push Apple (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, par exemple with APNs et 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 le paramètre TTL défini sur `0` lors de l'utilisation APNs, consultez le tableau A-3 Identifiants d'élément pour les notifications à distance dans la documentation de l'API Binary Provider.

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 par l'application mobile Amazon SNS

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)
- Afrique (Le Cap)
- Asie-Pacifique (Hong Kong)
- Asie-Pacifique (Jakarta)
- Asie-Pacifique (Mumbai)
- Asie-Pacifique (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-Ouest)

Bonnes pratiques pour gérer les notifications push mobiles Amazon SNS

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 lorsque les jetons de l'appareil changent en raison d'une action de l'utilisateur sur l'appareil (par exemple, une application est réinstallée sur l'appareil) ou lorsque des mises à jour de certificats affectent des appareils fonctionnant sur une version iOS particulière. Il s'agit d'une bonne pratique recommandée par Apple pour s'inscrire à APNs chaque lancement de votre application.

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 points de terminaison Firebase Cloud Messaging par Amazon SNS.

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.

Configuration et gestion de l'abonnement aux e-mails Amazon SNS

Vous pouvez abonner une adresse e-mail à une rubrique Amazon SNS en utilisant le AWS Management Console AWS SDK pour Java, ou. AWS SDK pour .NET

> **Remarques**
>
> - La personnalisation du corps du message électronique n'est pas prise en charge. 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. Pour plus d'informations, consultez la section 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 .

Abonnement d'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 de plus amples informations, veuillez consulter 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 de plus amples informations, veuillez consulter 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 pour les lettres mortes 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.

Abonnement d'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 la section Les fichiers de configuration et d'identification partagés dans le guide de référence AWS SDKs et Tools.

Les exemples de code suivants illustrent comment utiliser `Subscribe`.

.NET

SDK pour .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.

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 pour .NET .

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.

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 pour 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 sur 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.

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

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

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
```

```
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK pour Go .

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.

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 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 { 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 pour JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK pour JavaScript Référence de l'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.

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.fromEnvironment { 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.fromEnvironment { 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 sur 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 pour PHP Référence de l'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.

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 sur 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 pour Ruby Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK pour Ruby Référence de l'API.

Rust

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.

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 sur 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.

Swift

Kit SDK pour Swift

> **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.

Abonnez une adresse e-mail à un sujet.

```
import AWSSNS

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

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Abonnez un numéro de téléphone à un sujet pour recevoir des notifications par SMS.

```
import AWSSNS
```

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

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Pour plus de détails sur l'API, voir AWS Subscribe in SDK for Swift API reference.

Exemples de code pour Amazon SNS à l'aide de AWS SDKs

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres 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

SDK pour .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 pour .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 CMake fichier CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)
```

```
# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code 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 pour 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() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK pour 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 pour 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

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

Swift

Kit SDK pour Swift

> **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.

Le fichier Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Le programme principal de Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation
```

```
struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

```
        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

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

Exemples de code

- Exemples de base relatifs à l'utilisation d'Amazon SNS AWS SDKs
  - Hello Amazon SNS
  - Actions pour Amazon SNS utilisant AWS SDKs
    - 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
- 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

- Scénarios d'utilisation d'Amazon SNS AWS SDKs
  - 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'un point de terminaison de plateforme pour les notifications push Amazon SNS à l'aide d'un SDK AWS
  - 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
  - Création et publication sur une rubrique FIFO Amazon SNS à l'aide d'un SDK AWS
  - Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition 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
  - Utiliser API Gateway pour invoquer une fonction Lambda
  - Utilisent des événements planifiés pour invoquer une fonction Lambda
- Exemples de solutions sans serveur pour Amazon SNS
  - Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

Exemples de base relatifs à l'utilisation d'Amazon SNS AWS SDKs

Les exemples de code suivants montrent comment utiliser les bases d'Amazon Simple Notification Service avec AWS SDKs.

Exemples

- Hello Amazon SNS
- Actions pour Amazon SNS utilisant AWS SDKs
  - 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
  - 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

Hello Amazon SNS

Les exemples de code suivants montrent comment démarrer avec Amazon SNS.

.NET

SDK pour .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 pour .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 CMake fichier CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code 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 pour 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() {
```

```
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK pour 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 pour 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
```

```
}
```

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

Swift

Kit SDK pour Swift

> **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.

Le fichier Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
```

```
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Le programme principal de Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )
```

```
        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

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

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.

Actions pour Amazon SNS utilisant AWS SDKs

Les exemples de code suivants montrent comment effectuer des actions Amazon SNS individuelles avec. AWS SDKs Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Ces extraits appellent l'API Amazon SNS et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Vous pouvez voir les actions dans leur contexte dans Scénarios d'utilisation d'Amazon SNS AWS SDKs .

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour une description complète, veuillez consulter le document Référence de l'API Amazon Simple Notification Service.

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

- 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 illustrent comment utiliser `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK pour .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 pour .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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK pour 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 pour PHP.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK pour 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 illustrent 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 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous ConfirmSubscriptionà la section Référence des AWS SDK pour 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 pour 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 illustrent 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

SDK pour .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 pour .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 pour 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.

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

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

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
```

```
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour PHP.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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 pour Ruby.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour 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, reportez-vous CreateTopicà la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

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

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

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

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 illustrent 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

SDK pour .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 pour .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 pour 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.

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

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

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



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
```

```
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour 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.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

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

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

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 illustrent 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 plus de détails sur l'API, voir Get SMSAttributes in AWS SDK pour C++ API Reference.

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 de détails sur l'API, voir Get SMSAttributes in AWS CLI Command Reference.

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 plus de détails sur l'API, voir Get SMSAttributes in AWS SDK for Java 2.x API Reference.

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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, voir Get SMSAttributes in AWS SDK pour JavaScript API Reference.

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 pour PHP.
- Pour plus de détails sur l'API, voir Get SMSAttributes in AWS SDK pour PHP API Reference.

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 illustrent comment utiliser `GetTopicAttributes`.

.NET

SDK pour .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 pour .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 pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour 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, reportez-vous GetTopicAttributesà la 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 illustrent 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 pour PHP.
- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK pour 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 illustrent comment utiliser `ListSubscriptions`.

.NET

SDK pour .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 pour .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 pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK pour 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- 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 pour 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 pour Ruby.
- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK pour 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, reportez-vous ListSubscriptionsà la 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 illustrent comment utiliser `ListTopics`.

.NET

SDK pour .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 pour .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 pour 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() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour 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.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Pour de plus amples informations, consultez le Guide du développeur AWS SDK pour Ruby.
- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK pour 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, reportez-vous ListTopicsà la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
```

```
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

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

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 illustrent 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

SDK pour .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 pour .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 pour 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.

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

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

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



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

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

- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK pour 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;
```

```
    let deduplicationId;
    let choices;

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

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

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

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

```
            }
          : {}),
      }),
    );

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

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

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

PowerShell

Outils pour PowerShell V4

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, voir Publier dans la référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

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, voir Publier dans la référence des Outils AWS pour PowerShell applets de commande (V5).

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

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

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

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

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

- Pour plus de détails sur l'API, voir Publier dans le AWS SDK pour la référence de l'API Swift.

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 illustrent 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 par défautSMSType .

```
//! 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 plus de détails sur l'API, voir Set SMSAttributes in AWS SDK pour C++ API Reference.

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 de détails sur l'API, voir Set SMSAttributes in AWS CLI Command Reference.

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 plus de détails sur l'API, voir Set SMSAttributes in AWS SDK for Java 2.x API Reference.

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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, voir Set SMSAttributes in AWS SDK pour JavaScript API Reference.

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 pour PHP.
- Pour plus de détails sur l'API, voir Set SMSAttributes in AWS SDK pour PHP API Reference.

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 illustrent 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

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 illustrent 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 pour JavaScript Guide du développeur.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK pour 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.fromEnvironment { 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 pour 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 pour Ruby.
- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK pour 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.

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 illustrent 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

SDK pour .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 pour .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 pour 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.

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

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

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
```

```
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Pour de plus amples informations sur l'API, consultez S'abonner dans Référence de l'API AWS SDK pour 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 pour JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez S'abonner dans AWS SDK pour 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.fromEnvironment { 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.fromEnvironment { 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 pour 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 pour Ruby Guide du développeur.
- Pour de plus amples informations sur l’API, consultez S’abonner dans AWS SDK pour 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.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Abonnez un numéro de téléphone à un sujet pour recevoir des notifications par SMS.

```
import AWSSNS

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

        let output = try await snsClient.subscribe(
```

```
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Pour plus de détails sur l'API, voir AWS Subscribe in SDK for Swift API reference.

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 illustrent 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.fromEnvironment { 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 illustrent 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

SDK pour .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 pour .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 pour 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 pour JavaScript Guide du développeur.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK pour 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.fromEnvironment { 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 pour PHP.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK pour 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.

Swift

Kit SDK pour Swift

> **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 AWSSNS

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

```
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Pour plus de détails sur l'API, voir Désabonnement dans le AWS SDK pour une référence à l'API Swift.

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 d'utilisation d'Amazon SNS AWS SDKs

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans Amazon SNS avec. AWS SDKs Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'Amazon SNS ou en les combinant à d'autres. Services AWS Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code.

Les scénarios ciblent un niveau d'expérience intermédiaire pour vous aider à comprendre les actions de service dans leur contexte.

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'un point de terminaison de plateforme pour les notifications push Amazon SNS à l'aide d'un SDK AWS
- 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
- Création et publication sur une rubrique FIFO Amazon SNS à l'aide d'un SDK AWS

- Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition 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
- 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 pour JavaScript guide du développeur v3.

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

SDK pour .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'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 device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s

            """;

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

```
        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

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

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

SDK pour .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 pour 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 pour 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 le mode asynchrone APIs pour démarrer une tâche qui publie une notification dans une rubrique Amazon Simple Notification Service (Amazon SNS) une fois la tâche terminée.
- 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 Cognito Identity
- 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.

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

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

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

            return topicArn;

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

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

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

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

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
```

```
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

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

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

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

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

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

```
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


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

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error
```

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

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


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

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
 "wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
```

```
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


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

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

- 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.

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.

Java

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui détecte les visages et les objets dans des vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

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 S3
- Amazon SES
- Amazon SNS
- Amazon SQS

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 S3
- Amazon SES
- 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.

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

SDK pour .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 pour .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 pour 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.fromEnvironment { 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 pour PHP.
- Pour de plus amples informations sur l'API, consultez Publier dans AWS SDK pour 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

SDK pour .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 feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
```

```
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }
```

```
        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
_useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");
```

```
                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
```

```
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");
```

```
        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);
```

```
                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
```

```
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
```

```
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
```

```
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

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 pour .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 pour 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.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

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

```
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
```

```
    "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
    "but not delivered. For more information about deduplication, see:\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
   contentBasedDeduplication = runner.questioner.AskBool(
    "\nDo you want to use content-based deduplication instead of entering a
  deduplication ID? (y/n) ", "y")
  }
  log.Println(strings.Repeat("-", 88))

  topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
  if isFifoTopic {
   topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
   log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
    "the topic name.", FIFO_SUFFIX)
  }

  topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
  contentBasedDeduplication)
  if err != nil {
   panic(err)
  }
  log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
  \n"+
   "'%v' has been created.", topicName, topicArn)

  return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
  isFifoTopic bool) (string, string) {
  queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
  queue. ", ordinal))
  if isFifoTopic {
   queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
   if ordinal == "first" {
    log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
     "be appended to the queue name.\n", FIFO_SUFFIX)
   }
  }
  queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
  if err != nil {
   panic(err)
```

```
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")
```

```
    var toneSelections []string
    askAboutTones := true
    for askAboutTones {
     toneIndex := runner.questioner.AskChoice(
      "Enter the number of the tone you want to filter by:\n", ToneChoices)
     toneSelections = append(toneSelections, ToneChoices[toneIndex])
     askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
  the filter? (y/n) ", "y")
    }
    log.Printf("Your subscription will be filtered to only pass the following
  tones: %v\n", toneSelections)
    filterPolicy = map[string][]string{TONE_KEY: toneSelections}
   }
  }

  subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
  filterPolicy)
  if err != nil {
   panic(err)
  }
  log.Printf("The queue %v is now subscribed to the topic %v with the subscription
  ARN %v.\n",
   queueName, topicName, subscriptionArn)

  return subscriptionArn, filterPolicy != nil
 }

 func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
  string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
  var message string
  var groupId string
  var dedupId string
  var toneSelection string
  publishMore := true
  for publishMore {
   groupId = ""
   dedupId = ""
   toneSelection = ""
   message = runner.questioner.Ask("Enter a message to publish: ")
   if isFifoTopic {
    log.Println("Because you are using a FIFO topic, you must set a message group
  ID.\n" +
```

```
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
```

```
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
```

```
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
```

```
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Définissez une structure qui englobe les actions Amazon SNS utilisées dans cet exemple.

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

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

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



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

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
```

```
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Définissez une structure qui englobe les actions Amazon SQS utilisées dans cet exemple.

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

```
 "log"

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

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
```

```
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
```

```
    string(types.QueueAttributeNamePolicy): string(policyBytes),
   },
   QueueUrl: aws.String(queueUrl),
  })
  if err != nil {
   log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
  queueUrl, err)
  }
  return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
```

```
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Nettoyez les ressources.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management
 Console\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
```

```
  if err != nil {
   panic(err)
  }
 }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK pour Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

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.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

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

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
```

```
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
```

```
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,
\n"
                +
                "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
 must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
```

```
            ]
        }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
```

```
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
```

```
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
 duplication, groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
 continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
 Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
```

```
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

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

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

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

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

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

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
```

```
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

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

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

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

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
```

```
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

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

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

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

```
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
 HashMap<>();
                messageAttributes.put(msgAttValue,
 MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
```

```
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
```

```
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

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

    // Attach a policy to the queue.
```

```
   public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
       try {
           Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
           attrMap.put(QueueAttributeName.POLICY, policy);

           SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
               .queueUrl(queueUrl)
               .attributes(attrMap)
               .build();

           sqsClient.setQueueAttributes(attributesRequest);
           System.out.println("The policy has been successfully attached.");

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

   public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
       // Specify the attributes to retrieve.
       List<QueueAttributeName> atts = new ArrayList<>();
       atts.add(QueueAttributeName.QUEUE_ARN);

       GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
           .queueUrl(queueUrl)
           .attributeNames(atts)
           .build();

       GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
       Map<String, String> queueAtts = response.attributesAsStrings();
       for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
           return queueAtt.getValue();

       return "";
   }

   public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
```

```
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

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

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

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

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

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

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

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

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

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

- 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.

C'est le point d'entrée de ce scénario.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);
```

```
  wkflw.start();
};
```

Le code précédent fournit les dépendances nécessaires et lance le scénario. 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.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );
```

```
    this.queues.push({
      queueName,
      queueArn: Attributes.QueueArn,
      queueUrl: response.QueueUrl,
    });

    await this.logger.log(
      MESSAGES.queueCreatedNotice
        .replace("${QUEUE_NAME}", queueName)
        .replace("${QUEUE_URL}", response.QueueUrl)
        .replace("${QUEUE_ARN}", Attributes.QueueArn),
    );
  }
}

async attachQueueIamPolicies() {
  for (const [index, queue] of this.queues.entries()) {
    const policy = JSON.stringify(
      {
        Statement: [
          {
            Effect: "Allow",
            Principal: {
              Service: "sns.amazonaws.com",
            },
            Action: "sqs:SendMessage",
            Resource: queue.queueArn,
            Condition: {
              ArnEquals: {
                "aws:SourceArn": this.topicArn,
              },
            },
          },
        ],
      },
      null,
      2,
    );

    if (index !== 0) {
      this.logger.logSeparator();
    }
```

```
      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
```

```
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

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

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

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

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

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

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

```
    );

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

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

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
```

```
      }
    }

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

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
```

```
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK pour JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

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.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
```

```
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)
```

```
    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
```

```
        },
        "Action": "sqs:SendMessage",
                "Resource": "$sqsQueueArn",
                "Condition": {
            "ArnEquals": {
                "aws:SourceArn": "$topicArn"
            }
        }
    }
    ]
    }"""
   setQueueAttr(sqsQueueUrl, policy)
   println(DASHES)

   println(DASHES)
   println("6. Subscribe to the SQS queue.")
   if (selectFIFO) {
       println(
           """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
       )
       println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
       val filterAns: String = input.nextLine()
       if (filterAns.compareTo("y") == 0) {
           var moreAns = false
           println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
           println("1. cheerful")
           println("2. funny")
           println("3. serious")
           println("4. sincere")
           while (!moreAns) {
               println("Select a number or choose 0 to end.")
               val ans: String = input.nextLine()
               when (ans) {
                   "1" -> filterList.add("cheerful")
                   "2" -> filterList.add("funny")
                   "3" -> filterList.add("serious")
                   "4" -> filterList.add("sincere")
                   else -> moreAns = true
```

```
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
```

```
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

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

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

```
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
```

```
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

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

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

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
```

```
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
```

```
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }
```

```
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
```

```
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

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

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

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
```

```
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Swift

Kit SDK pour Swift

> **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 ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
            str = readLine()

            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
```

```
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
```

```
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
 attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
```

```
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the
 'tone' attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
 throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]
```

```
            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
String? {
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }
```

```
        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }
```

```
    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
                                ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
```

```
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
            }
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues
example.
              In this workflow, you'll create an SNS topic, then create two SQS
```

```
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well
 as
              the queue subscriptions. You can then post messages to the topic
 and
              receive the results on the queues.
              \(rowOfStars)\n
              """
       )

       // 0. Create SNS and SQS clients.

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

       let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
 region)
       let sqsClient = SQSClient(config: sqsConfig)

       // 1. Ask the user whether to create a FIFO topic. If so, ask whether
       //    to use content-based deduplication instead of requiring a
       //    deduplication ID.

       let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
       var isContentBasedDeduplication = false

       if isFIFO {
           print("""
                 \(rowOfStars)
                 Because you've chosen to create a FIFO topic, deduplication is
                 supported.

                 Deduplication IDs are either set in the message or are
 automatically
                 generated from the content using a hash function.

                 If a message is successfully published to an SNS FIFO topic,
 any
                 message published and found to have the same deduplication ID
                 (within a five-minute deduplication interval), is accepted but
                 not delivered.
```

```
                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
 deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
 create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
```

```
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")
```

```
        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
 messages will
                be received in the queue. For information about message
 filtering, see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's
 subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
```

```
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil

        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }
```

```
        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
 publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.
\n
                        """
                    )
                }
```

```
                publishInput.messageGroupId = stringRequest(prompt: "Enter a
message group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based
deduplication, you
                              must enter a deduplication ID. If other messages
with the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue]
= [:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType:
"String", stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }
```

```
            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )
```

```
        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Pour plus d’informations sur l’API, consultez les rubriques suivantes dans la référence de l’API du kit SDK AWS pour Swift.
  - 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.

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 pour JavaScript guide du développeur v3.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK pour Python (Boto3)

Cet exemple montre comment créer et utiliser une API REST Amazon API Gateway qui cible une fonction AWS Lambda . Le gestionnaire Lambda explique comment router en fonction des méthodes HTTP, comment obtenir des données à partir de la chaîne de requête, de l'en-tête et du corps, et comment renvoyer une réponse JSON.

- Déploier une fonction Lambda.
- Créer une API REST avec API Gateway.
- Créer une ressource REST qui cible la fonction Lambda.
- Accorder à API Gateway l'autorisation d'invoquer la fonction Lambda.
- Utiliser le package Requests (Requêtes) pour envoyer des requêtes à l'API REST.

- Nettoyer toutes les ressources créées lors de la démonstration.

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

- 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

- CloudWatch Journaux

- 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 pour JavaScript guide du développeur v3.

Les services utilisés dans cet exemple

- CloudWatch Journaux
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK pour Python (Boto3)

Cet exemple montre comment enregistrer une AWS Lambda fonction en tant que cible d'un EventBridge événement Amazon planifié. Le gestionnaire Lambda écrit un message

convivial et les données complètes de l'événement dans Amazon CloudWatch Logs pour une récupération ultérieure.

- Déploie une fonction Lambda.
- Crée un événement EventBridge planifié et fait de la fonction Lambda la cible.
- Accorde l'autorisation de laisser EventBridge invoquer la fonction Lambda.
- Imprime les dernières données des CloudWatch journaux pour afficher le résultat des appels planifiés.
- Nettoie toutes les ressources créées lors de la démonstration.

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

- CloudWatch Journaux
- 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.

Exemples de solutions sans serveur pour Amazon SNS

Les exemples de code suivants montrent comment utiliser Amazon SNS avec. AWS SDKs

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

SDK pour .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.

Consommation 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.

Sécurité Amazon SNS

Le modèle de responsabilité AWS partagée s'applique à la protection des données dans Amazon Simple Notification Service. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Ce contenu inclut la configuration de la sécurité et les tâches de gestion pour les AWS services que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (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 AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

Pour des raisons de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer des 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 multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous recommandons TLS 1.2 ou version ultérieure.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut au sein AWS des services.
- 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 par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).
- 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 de vos clients, dans des balises ou des champs de forme libre tels qu'un champ Nom. Cela inclut lorsque vous travaillez avec Amazon SNS ou d'autres Amazon Web Services à l'aide de la console, de l'API ou. AWS CLI AWS SDKs Toutes les données que vous saisissez dans des identifications ou des champs de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement des données Amazon SNS

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.

Sécurisation des données Amazon SNS grâce au chiffrement côté serveur

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 de SSE à l'aide du AWS Management Console ou du AWS SDK pour Java (en définissant l'`KmsMasterKeyId`attribut à l'aide des actions de `SetTopicAttributes` l'API `CreateTopic` et), consultezConfiguration du chiffrement des rubriques Amazon SNS avec le chiffrement côté serveur.
- Pour plus d'informations sur la création de sujets chiffrés à l'aide de AWS CloudFormation (en définissant la `KmsMasterKeyId` propriété à l'aide de la `AWS::SNS::Topic` ressource), consultez le guide de AWS CloudFormation l'utilisateur.

> **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 des clés adapté au 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 AWS gérées pour Amazon SNS, qui sont uniques pour chaque compte et chaque région.
- Les normes AWS KMS de sécurité peuvent vous aider à respecter les exigences de conformité liées au chiffrement.

Pour plus d'informations, voir Qu'est-ce que c'est AWS Key Management Service ? dans le Guide AWS Key Management Service du développeur.

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 .

AWS KMS key ID

L'alias, l'alias ARN, l'ID de clé ou l'ARN clé d'un AWS KMS key compte ou d'un ARN AWS KMS personnalisé dans votre compte ou dans un autre compte. Bien que l'alias du AWS managé AWS KMS pour Amazon SNS soit toujours le `casalias/aws/sns`, l'alias d'un utilisateur personnalisé

AWS KMS 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 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.

Vous pouvez créer des AWS KMS clés, définir les politiques qui contrôlent la manière dont AWS KMS les clés peuvent être utilisées et auditer l' AWS KMS utilisation à l'aide de la AWS KMS keyssection de la AWS KMS console ou de l'`CreateKey` AWS KMS action. 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 KMS identifiants, consultez la référence KeyIdde l'AWS Key Management Service API. Pour plus d'informations sur la recherche AWS KMS d'identifiants, voir Trouver l'ID de clé et l'ARN dans le guide du AWS Key Management Service développeur.

> **Important**
>
> L'utilisation entraîne des frais supplémentaires AWS KMS. Pour plus d'informations, veuillez consulter les sections Estimation AWS KMS des coûts et Tarification d'AWS Key Management Service.

Gestion des clés de chiffrement et des coûts Amazon SNS

Les sections suivantes fournissent des informations sur l'utilisation des clés gérées dans AWS Key Management Service (AWS KMS).

> **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.

Estimation AWS KMS des coûts

Pour prévoir les coûts et mieux comprendre votre AWS facture, vous souhaiterez peut-être savoir à quelle fréquence 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' AWS KMS API 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 AWS KMS autorisations

Avant de pouvoir utiliser SSE, vous devez configurer des AWS KMS key politiques pour autoriser le chiffrement des sujets ainsi que le chiffrement et le déchiffrement des 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 développeur AWS Key Management Service . Pour plus d'informations sur la configuration d'une rubrique Amazon SNS avec le chiffrement côté serveur, consultez Informations supplémentaires.

> **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 la section Utilisation des politiques IAM avec AWS KMS.
> Bien que vous puissiez configurer des autorisations globales pour envoyer et recevoir depuis Amazon SNS, AWS KMS il est nécessaire de nommer explicitement l'ARN complet de certaines régions KMSs dans la `Resource` section d'une politique IAM.

Vous devez également vous assurer que les principales politiques du pays AWS KMS key autorisent 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 AWS KMS actions requises et l'ARN KMS dans une politique IAM attribuée aux principaux responsables qui publient et s'abonnent pour recevoir des messages

chiffrés sur Amazon SNS. 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"
  }]
}
```

Activez la compatibilité entre les sources d'événements issues AWS des services et des sujets chiffrés

Plusieurs AWS services publient des événements sur des sujets 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 plus d’informations, consultez Création des clés dans le Guide du développeur AWS Key Management Service .

2. Pour autoriser le AWS service à disposer des `kms:Decrypt` autorisations `kms:GenerateDataKey*` et, ajoutez l'instruction suivante à la politique 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`
CloudWatch Événements Amazon	`events.amazonaws.com`
AWS CodeCommit	`codecommit.amazonaws.com`
AWS Database Migration Service	`dms.amazonaws.com`
AWS Directory Service	`ds.amazonaws.com`
Amazon DynamoDB	`dynamodb.amazonaws.com`
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`
Amazon Simple Email Service	`ses.amazonaws.com`

Source de l'événement	principal du service
Amazon Simple Storage Service	`s3.amazonaws.com`
AWS Snowball Edge	`importexport.amazonaws.com`
AWS Systems Manager Gestionnaire des incidents	AWS Systems Manager Incident Manager repose sur deux principes de service : `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Certaines sources d'événements Amazon SNS nécessitent que vous fournissiez un rôle IAM (plutôt que le rôle 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:SourceAccountaws:SourceArn`, et `aws:SourceOrgID` à une AWS KMS politique n'est pas pris en charge pour les EventBridge-to-encrypted rubriques.

```
{
```

```
  "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.

AWS KMS erreurs

Lorsque vous travaillez avec Amazon SNS AWS KMS, vous pouvez 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' AWS accès nécessite un abonnement au 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

Configuration du chiffrement des rubriques Amazon SNS avec le chiffrement côté serveur

Amazon SNS prend en charge le chiffrement côté serveur (SSE) pour protéger le contenu des messages à l'aide de (). AWS Key Management Service AWS KMS Suivez les instructions ci-dessous pour activer SSE à l'aide de la console Amazon SNS ou du CDK.

Option 1 : activer le chiffrement à l'aide du AWS Management Console

1. Connectez-vous à la console Amazon SNS.
2. Accédez à la page Sujets, sélectionnez votre sujet, puis choisissez Modifier.
3. Développez la section Encryption (Chiffrement) et effectuez les opérations suivantes :
   - Activez le chiffrement.

- Sélectionnez le AWS géré SNS Clé (alias/aws/sns) comme clé de chiffrement. Cette option est sélectionnée par défaut.

4. Sélectionnez Enregistrer les modifications.

> **Note**
>
> - Le Clé gérée par AWS est créé automatiquement s'il n'existe pas déjà.
> - Si vous ne voyez pas la clé ou si vous ne disposez pas d'autorisations suffisantes, demandez à votre administrateur de vous `kms:ListAliases` fournir et`kms:DescribeKey`.

Option 2 : activer le chiffrement à l'aide de AWS CDK

Pour utiliser le AWS managé SNS dans votre application CDK, ajoutez l'extrait suivant :

```
import software.amazon.awscdk.services.sns.*;
import software.amazon.awscdk.services.kms.*;
import software.amazon.awscdk.core.*;

public class SnsEncryptionExample extends Stack {
    public SnsEncryptionExample(final Construct scope, final String id) {
        super(scope, id);

        // Define the managed SNS key
        IKey snsKey = Alias.fromAliasName(this, "helloKey", "alias/aws/sns");

        // Create the SNS Topic with encryption enabled
        Topic.Builder.create(this, "MyEncryptedTopic")
            .masterKey(snsKey)
            .build();
    }
}
```

Informations supplémentaires

- Clé KMS personnalisée : vous pouvez spécifier une clé personnalisée si nécessaire. Dans la console Amazon SNS, sélectionnez votre clé KMS personnalisée dans la liste ou saisissez l'ARN.

- Autorisations pour les clés KMS personnalisées : si vous utilisez une clé KMS personnalisée, incluez les éléments suivants dans la politique relative aux clés pour autoriser Amazon SNS à chiffrer et déchiffrer les messages :

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

Incidence sur les consommateurs

L'activation de SSE ne modifie pas la façon dont les abonnés consomment les messages. AWS gère le chiffrement et le déchiffrement de manière transparente. Les messages restent chiffrés au repos et sont automatiquement déchiffrés avant d'être envoyés aux abonnés. Pour une sécurité optimale, il est AWS recommandé d'activer le protocole HTTPS pour tous les points de terminaison afin de garantir une transmission sécurisée des messages.

Configuration du chiffrement des rubriques Amazon SNS avec un abonnement chiffré à la file d'attente Amazon SQS

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 Sécurisation des données Amazon SNS grâce au chiffrement côté serveur.

Cette rubrique explique comment activer SSE pour une rubrique Amazon SNS avec un abonnement à une file d'attente Amazon SQS chiffrée à l'aide du. AWS Management Console

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 Suivant.
6. Saisissez un alias pour la clé KMS. Le nom d'alias ne peut pas commencer par **`aws/`**. Le **`aws/`** préfixe est réservé par Amazon Web Services pour être représenté 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 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'CreateKeyopération.

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 le client existante, modifiez la description dans l'opération AWS Management Console ou utilisez l'UpdateKeyDescriptionopération.

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 AWS ressources, AWS génère un rapport de répartition des coûts avec l'utilisation et les coûts agrégés 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 Suivant.

10. Sélectionnez les utilisateurs et les rôles IAM qui peuvent administrer la clé KMS.

    > **Note**
    >
    > Cette politique clé donne le contrôle Compte AWS 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 Suivant.
13. Sélectionnez les utilisateurs et les rôles IAM qui peuvent utiliser la clé dans les opérations de chiffrement. Choisissez 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 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 Autorisations d'API AWS KMS : référence des actions et ressources dans le Guide du développeur AWS Key Management Service .

   c. Pour la clé gérée par le client, choisissez MyCustomKeycelle que vous avez créée précédemment, puis sélectionnez Activer le chiffrement côté serveur.
6. Sélectionnez Enregistrer les modifications.

   SSE est activé pour votre sujet et la MyTopicpage 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 celui MyCustomKeyque vous avez créé 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 un sujet, pour Choisir un sujet MyEncryptedTopic, sélectionnez, 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. Choisissez 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 sujets, choisissez MyEncryptedTopicpuis cliquez sur Publier le 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 sélectionnez Envoyer et recevoir des messages.
3. Sur la page Envoyer et recevoir des messages en MyEncryptedQueue 1, choisissez Sondage pour les 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 le processus pour MyEncryptedQueue2.

Sécurisation du trafic Amazon SNS avec des points de terminaison VPC

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 AWS ressources, vous pouvez établir une connexion privée 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 AWS service que vous pouvez utiliser pour lancer AWS des ressources 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 AWS des services. 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 la section Accès et Service AWS utilisation d'un point de terminaison VPC d'interface 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 Planifier votre 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.

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 EC2 instance Amazon qui réside dans votre VPC. Pour plus d'informations sur la connexion, consultez Connect to your Linux instance ou Connecting to your Windows Instance dans la EC2 documentation Amazon.

   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 lorsque vous avez créé l'instance. EC2
   - instance-hostname est le nom d'hôte public de l'instance. Pour obtenir le nom d'hôte dans la EC2console Amazon : choisissez Instances, choisissez votre instance et recherchez la valeur du DNS public.
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 depuis une EC2 instance Amazon hébergée sur Amazon Virtual Private Cloud (Amazon VPC). Le message reste sur le AWS réseau 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 AWS CloudFormation modèle 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 EC2 instance Amazon et publiez un message en privé sur 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 décrit le réseau privé que vous créez dans votre AWS compte au fur et à mesure que vous effectuez ces étapes :

Ce réseau est constitué 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 alimentés par AWS PrivateLink. Une fois cette connexion établie, vous pouvez vous connecter à l' EC2 instance Amazon et publier des messages sur la rubrique Amazon SNS, même si le réseau est déconnecté de l'Internet public. Le sujet répartit les messages qu'il reçoit vers deux AWS Lambda fonctions d'abonnement. Ces fonctions enregistrent les messages qu'elles reçoivent dans Amazon CloudWatch Logs.

Il vous faudra environ 20 minutes pour réaliser ces étapes.

Rubriques

- Avant de commencer
- Étape 1 : créer une paire de EC2 clés Amazon
- Étape 2 : Création des AWS ressources
- Étape 3 : Vérifiez que votre EC2 instance Amazon n'a pas accès à Internet
- Étape 4 : Création d'un point de terminaison Amazon VPC pour Amazon SNS
- Étape 5 : Publication d'un message dans votre rubrique Amazon SNS
- Étape 6 : Vérifier la livraison de vos messages
- Étape 7 : nettoyer
- Ressources connexes

Avant de commencer

Avant de commencer, vous avez besoin d'un compte Amazon Web Services (AWS). Lorsque vous vous inscrivez, votre compte est automatiquement connecté à tous les services AWS, y compris 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 EC2 clés Amazon

Une paire de clés est utilisée pour se connecter à une EC2 instance Amazon. 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. Vous utiliserez ensuite la paire de clés pour vous connecter à une EC2 instance Amazon. 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 EC2 console Amazon AWS Management Console et ouvrez-la 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 (Créer une paire de clés).
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 attribue au fichier une extension de `.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 : Création des AWS ressources

Pour configurer l'infrastructure, vous utilisez un AWS CloudFormation modèle. Un modèle est un fichier qui sert de modèle pour créer des AWS ressources, telles que des EC2 instances Amazon et

des rubriques Amazon SNS. Le modèle de ce processus est fourni ci-dessous GitHub pour que vous puissiez le télécharger.

Vous fournissez le modèle et AWS CloudFormation vous AWS CloudFormation provisionnez les ressources dont vous avez besoin sous forme de pile dans votre Compte AWS. Une pile est un ensemble de ressources que vous gérez comme une seule unité. Lorsque vous avez terminé ces étapes, vous pouvez l'utiliser AWS CloudFormation pour supprimer toutes les ressources de la pile en une seule fois. Ces ressources ne restent pas dans votre mémoire 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 EC2 instance Amazon lancée dans le sous-réseau du VPC.
- Une rubrique Amazon SNS
- Deux AWS Lambda fonctions. Ces fonctions reçoivent des messages publiés sur la rubrique Amazon SNS et enregistrent les événements dans CloudWatch des journaux.
- Statistiques CloudWatch et journaux Amazon.
- Un rôle IAM qui permet à l' EC2 instance Amazon d'utiliser Amazon SNS, et un rôle IAM qui permet aux fonctions Lambda d'écrire dans des journaux. CloudWatch

Pour créer les AWS ressources

1. Téléchargez le fichier modèle depuis le GitHub site Web.
2. Connectez-vous à la console AWS CloudFormation.
3. Sélectionnez 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 de**`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 Fonctionnalités, reconnaissez que cela AWS CloudFormation peut créer des ressources IAM avec des noms personnalisés.
9. Choisissez Créer.

   La AWS CloudFormation console ouvre la page Stacks. Le statut VPCE-Tutorial-Stack est 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 : Vérifiez que votre EC2 instance Amazon n'a pas accès à Internet

L' EC2 instance Amazon 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 EC2 instance Amazon

1. Ouvrez la EC2 console Amazon 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 fourni dans la colonne DNS public.

5. Ouvrez un terminal . Dans le répertoire contenant la paire de clés, connectez-vous à l'instance à l'aide de la commande suivante, où se *instance-hostname* trouve le nom d'hôte que vous avez copié depuis la EC2 console Amazon :

```
$ 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. Après avoir ajouté le point de terminaison, vous pouvez vous connecter à l' EC2 instance Amazon dans votre VPC, puis 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 AWS réseau et ne naviguent pas sur l'Internet public.

> **Note**
>
> L'instance n'a toujours pas accès aux autres AWS services 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 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 avez choisi USA East (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 Groupe de sécurité, choisissez Sélectionner un groupe de sécurité, puis choisissez VPCE-Tutorial-. SecurityGroup
10. Choisissez 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 EC2 Amazon et publier des messages sur le sujet.

Pour publier un message

1. Si votre terminal n'est plus connecté à votre EC2 instance Amazon, 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 enregistrent l'événement dans les CloudWatch journaux. Pour vérifier que la livraison de votre message a réussi, vérifiez que les fonctions ont été invoquées et que les CloudWatch journaux ont été mis à jour.

Pour vérifier que les fonctions Lambda ont été appelées

1. Ouvrez la AWS Lambda console 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 CloudWatch journaux ont été mis à jour

1. Ouvrez la CloudWatch console 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). Répétez ensuite les étapes précédentes pour le groupe de logs the /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 AWS les ressources que vous n'utilisez plus, vous évitez des frais inutiles pour votre Compte AWS.

Tout d'abord, supprimez votre point de terminaison de VPC à l'aide de la console Amazon VPC. Supprimez ensuite les autres ressources que vous avez créées en supprimant la pile dans la AWS CloudFormation console. Lorsque vous supprimez une pile, AWS CloudFormation les ressources de la pile sont supprimées 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 AWS CloudFormation pile

1. Ouvrez la AWS CloudFormation console 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.

- AWS Blog sur la sécurité : Sécurisation des messages publiés sur Amazon SNS avec AWS PrivateLink
- Qu’est-ce qu’Amazon VPC ?
- Points de terminaison d’un VPC
- Qu'est-ce qu'Amazon EC2 ?
- Concepts AWS CloudFormation

Connectez-vous à Amazon SNS à l'aide de points de terminaison à double pile (IPv4 et) IPv6

Les terminaux à double pile prennent en charge à la fois le trafic IPv4 et IPv6 le trafic. Lorsque vous envoyez une demande à un point de terminaison à double pile, l'URL du point de terminaison est une adresse IPv4 ou une IPv6 adresse. Pour plus d'informations sur les points de terminaison à double pile et FIPS, consultez le guide de référence du SDK.

Amazon SNS prend en charge les points de terminaison régionaux à double pile, ce qui signifie que vous devez spécifier la AWS région dans le nom du point de terminaison. Les noms de point de terminaison à double pile utilisent la convention de dénomination suivante :`sns.Region.amazonaws.com`. Par exemple, le nom du point de terminaison à double pile de la région `eu-west-1` est `sns.eu-west-1.amazonaws.com`.

Pour obtenir la liste complète des points de terminaison Amazon SNS, consultez la AWS référence générale.

Renforcer la sécurité d'Amazon SNS grâce à la protection des données des messages

- 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 contenus 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 chaque abonné et auditer l'ensemble du processus de flux de contenus.

Gestion des identités et des accès dans Amazon SNS

L'accès à Amazon SNS nécessite des informations d'identification qui AWS peuvent être utilisées pour authentifier vos demandes. Ces informations d'identification doivent être autorisées à accéder aux AWS ressources, telles que les sujets 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 outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. 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 outil que vous pouvez utiliser sans frais supplémentaires.

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez sur 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 Comment Amazon SNS fonctionne 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 est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur Compte AWS root, en tant qu'utilisateur IAM ou en assumant un rôle IAM.

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs

(IAM Identity Center), l'authentification 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 l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez AWS Signature Version 4 pour les demandes d'API 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, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour plus d'informations, consultez Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Authentification multifactorielle AWS dans IAM dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez Tâches nécessitant des informations d'identification d'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent 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 pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. 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 au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule 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 telles que des mots de passe et des 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 d'effectuer une rotation des 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 nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les 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 plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Pour assumer temporairement un rôle IAM dans le AWS Management Console, vous pouvez passer d'un rôle d'utilisateur à un rôle IAM (console). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Méthodes pour endosser un rôle 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, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités 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, un rôle de service ou un rôle lié au service.

- Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. 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.
- Rôle de 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 un rôle 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 à un service — 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 apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal

(utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON 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 assumer 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 appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l'identité

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 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 Définition d'autorisations IAM personnalisées avec des politiques gérées par le client 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 associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées 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. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. 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. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste 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 courants. 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 fonctionnalité 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 en résultant 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 annule 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 (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités des comptes membres, y compris pour chaque utilisateur Compte AWS root. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de contrôle des ressources (RCPs) : RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les autorisations effectives pour les identités, y compris l'utilisateur Compte AWS root, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organizations RCPs, y compris une liste de ces Services AWS supports RCPs, consultez la section Resource control policies (RCPs) dans le guide de AWS Organizations l'utilisateur.
- 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 séance en résultant 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 annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Contrôle d’accès

Amazon SNS possède son propre système d'autorisations basé sur les ressources qui utilise des politiques rédigées dans le même langage que celui utilisé pour les 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 Comptes AWS peuvent déléguer leurs autorisations aux utilisateurs de leurs 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.

Cas d'utilisation du contrôle d'accès Amazon SNS

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 Compte AWS une autre personne un type particulier d'action sur le sujet (par exemple, Publier). Pour de plus amples informations, veuillez consulter Autoriser Compte AWS l'accès à un sujet.
- Vous souhaitez limiter les abonnements à votre rubrique au seul protocole HTTPS. Pour de plus amples informations, veuillez consulter Limiter les abonnements à HTTPS.
- Vous souhaitez autoriser Amazon SNS à publier des messages dans votre file d'attente Amazon SQS. Pour de plus amples informations, veuillez consulter Publier des messages dans une file d'attente Amazon SQS.

Concepts clés de la politique d'accès Amazon SNS

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.

Autorisations

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` contourner le scénario dans lequel vous devez connecter d'autres AWS services ARNs 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. L'émetteur (par définition) est toujours le propriétaire de la ressource. AWS n'autorise pas les utilisateurs AWS du service à créer des politiques pour les ressources dont ils ne sont pas propriétaires. Si John est le propriétaire de la ressource, AWS authentifie l'identité de John lorsqu'il soumet la politique qu'il a écrite pour accorder des autorisations pour 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 règle générale, l'action est simplement l'opération figurant dans la demande 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 AWS service lui-même (par exemple, Amazon SQS ou Amazon SNS) peut également définir des clés spécifiques au service. Pour de plus amples informations, veuillez consulter Autorisations d'API Amazon SNS : référence des actions et ressources.

Demandeur

Le demandeur est la personne qui envoie une demande à un AWS service et demande l'accès à une ressource particulière. Le demandeur envoie une demande AWS qui dit essentiellement : « M'autoriserez-vous à faire B to C là où D s'applique ? »

Evaluation

L'évaluation est le processus utilisé par le AWS service pour déterminer si une demande entrante doit être refusée ou autorisée conformément aux 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 de plus amples informations, veuillez consulter 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 de contrôle d'accès Amazon SNS

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 AWS service ; 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 AWS service lui-même fournit une API que vous utilisez pour télécharger et gérer vos politiques.
4	Les demandeurs et leurs demandes entrantes auprès du AWS service.
5	Code d'évaluation du langage de la politique d'accès. Il s'agit de l'ensemble de code du AWS service qui évalue les demandes entrantes par rapport aux politiques applicables et 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 politique d'accès dans Amazon SNS

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 téléchargez votre politique sur AWS. Le AWS service lui-même fournit une API que vous utilisez pour télécharger 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 AWS service détermine les politiques 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 AWS service 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 AWS service 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'informa tions, 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 en question ne fait pas du tout référence à celle de Compte AWS l'utilisateur, cette politique entraîne 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.

Autoriser Compte AWS l'accès à un sujet

Supposons que vous ayez un sujet sur Amazon SNS et que vous souhaitiez autoriser un ou plusieurs d'entre eux Comptes AWS à effectuer une action spécifique sur ce sujet, telle que la publication de messages. Vous pouvez y parvenir en utilisant l'action d'API Amazon SNS. `AddPermission`

L'`AddPermission`action vous permet de spécifier un sujet, une liste de Compte AWS IDs, une liste d'actions et une étiquette. Amazon SNS génère et ajoute ensuite automatiquement une nouvelle déclaration de politique à la politique de contrôle d'accès du sujet. Vous n'avez pas besoin de rédiger vous-même la déclaration de politique : Amazon SNS s'en charge pour vous. Si vous devez supprimer la politique ultérieurement, vous pouvez le faire en appelant `RemovePermission` et en fournissant l'étiquette que vous avez utilisée lors de l'ajout de l'autorisation.

Par exemple, si vous appelez le `AddPermission` sujet arn:aws:sns:us-east-2:444455556666 :MyTopic, que vous spécifiez l'ID 1111-2222-3333, l'action et le libellé, Amazon `grant-1234-publish` SNS générera `Publish` et Compte AWS insérera la déclaration de politique suivante dans la politique de contrôle d'accès du sujet :

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
```

```
}
```

Une fois cette déclaration ajoutée, le Compte AWS 1111-2222-3333 sera autorisé à publier des messages sur le sujet.

Informations supplémentaires :

- Gestion des politiques personnalisées : Bien que cela `AddPermission` soit pratique pour accorder des autorisations, il est souvent utile de gérer manuellement la politique de contrôle d'accès du sujet pour des scénarios plus complexes, tels que l'ajout de conditions ou l'octroi d'autorisations à des rôles ou services IAM spécifiques. Vous pouvez le faire en utilisant l'`SetTopicAttributes`API pour mettre à jour directement l'attribut policy.
- Bonnes pratiques en matière de sécurité : soyez prudent lorsque vous accordez des autorisations afin de vous assurer que seules les entités Comptes AWS ou entités de confiance ont accès à vos rubriques Amazon SNS. Passez régulièrement en revue et auditez les politiques associées à vos sujets afin de garantir la sécurité.
- Limites des politiques : gardez à l'esprit que la taille et la complexité des politiques Amazon SNS sont limitées. Si vous devez ajouter de nombreuses autorisations ou des conditions complexes, assurez-vous que votre politique respecte ces limites.

Limiter les abonnements à HTTPS

Pour restreindre le protocole d'envoi des notifications pour votre rubrique Amazon SNS au protocole HTTPS, vous devez créer une politique personnalisée. L'`AddPermission`action dans Amazon SNS ne vous permet pas de spécifier des restrictions de protocole lorsque vous accordez l'accès à votre rubrique. Par conséquent, vous devez écrire manuellement une politique qui applique cette restriction, puis utiliser l'`SetTopicAttributes`action pour appliquer la politique à votre sujet.

Voici comment créer une politique qui limite les abonnements au protocole HTTPS :

1. Rédigez la politique. La politique doit spécifier l' Compte AWS ID auquel vous souhaitez accorder l'accès et appliquer la condition selon laquelle seuls les abonnements HTTPS sont autorisés. Vous trouverez ci-dessous un exemple de politique qui accorde à l' Compte AWS ID 1111-2222-3333 l'autorisation de s'abonner à la rubrique, mais uniquement si le protocole utilisé est HTTPS.

   ```
   {
     "Statement": [{
   ```

```
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
      }
    }
  }]
}
```

2. Appliquez la politique. Utilisez l'`SetTopicAttributes`action de l'API Amazon SNS pour appliquer cette politique à votre sujet. Définissez l'`Policy`attribut de la rubrique selon la politique JSON que vous avez créée.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informations supplémentaires :

- Personnalisation du contrôle d'accès. Cette approche vous permet d'appliquer des contrôles d'accès plus précis, tels que la restriction des protocoles d'abonnement, ce qui n'est pas possible par le seul biais de cette `AddPermission` action. Les politiques personnalisées offrent de la flexibilité pour les scénarios nécessitant des conditions spécifiques, telles que l'application du protocole ou les restrictions d'adresse IP.
- Bonnes pratiques en matière de sécurité. La limitation des abonnements au protocole HTTPS améliore la sécurité de vos notifications en garantissant que les données en transit sont cryptées. Passez régulièrement en revue vos politiques thématiques pour vous assurer qu'elles répondent à vos exigences en matière de sécurité et de conformité.
- Tests de politiques. Avant d'appliquer la politique dans un environnement de production, testez-la dans un environnement de développement pour vous assurer qu'elle se comporte comme prévu. Cela permet d'éviter les problèmes d'accès accidentels ou les restrictions involontaires.

Publier des messages dans une file d'attente Amazon SQS

Pour publier des messages de votre rubrique Amazon SNS dans une file d'attente Amazon SQS, vous devez configurer les autorisations appropriées sur la file d'attente Amazon SQS. Bien qu'Amazon SNS et Amazon SQS AWS utilisent le langage de politique de contrôle d'accès, vous devez définir explicitement une politique sur la file d'attente Amazon SQS afin d'autoriser l'envoi de messages depuis la rubrique Amazon SNS.

Vous pouvez y parvenir en utilisant l'`SetQueueAttributes`action pour appliquer une politique personnalisée à la file d'attente Amazon SQS. Contrairement à Amazon SNS, Amazon SQS ne prend pas en charge `AddPermission` la création de déclarations de politique assorties de conditions. Par conséquent, vous devez écrire la politique manuellement.

Voici un exemple de politique Amazon SQS qui accorde à Amazon SNS l'autorisation d'envoyer des messages à votre file d'attente. Notez que cette politique est associée à la file d'attente Amazon SQS, et non à la rubrique Amazon SNS. Les actions spécifiées sont des actions Amazon SQS, et la ressource est l'Amazon Resource Name (ARN) de la file d'attente. Vous pouvez récupérer l'ARN de la file d'attente à l'aide de l'`GetQueueAttributes`action.

```
{
  "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 cette `aws:SourceArn` condition pour restreindre l'accès à la file d'attente SQS en fonction de la source des messages envoyés. Cela garantit que seuls les messages provenant de la rubrique SNS spécifiée (dans ce cas, arn:aws:sns:us-east- 2:444455556666 :) sont autorisés à être remis à la file d'attente. MyTopic

Informations supplémentaires :

- ARN de la file d'attente. Assurez-vous de récupérer le bon ARN de votre file d'attente Amazon SQS à l'aide de cette action. `GetQueueAttributes` Cet ARN est essentiel pour définir les autorisations correctes.
- Bonnes pratiques en matière de sécurité. Lorsque vous définissez des politiques, respectez toujours le principe du moindre privilège. Accordez uniquement les autorisations nécessaires à la rubrique Amazon SNS pour interagir avec la file d'attente Amazon SQS, et passez régulièrement en revue vos politiques pour vous assurer qu'elles sont sécurisées up-to-date
- Politiques par défaut dans Amazon SNS. Amazon SNS n'accorde pas automatiquement de politique par défaut permettant à d'autres utilisateurs Services AWS ou à des comptes d'accéder aux sujets nouvellement créés. Par défaut, les sujets Amazon SNS sont créés sans autorisation, ce qui signifie qu'ils sont privés et accessibles uniquement au compte qui les a créés. Pour permettre l'accès à d'autres personnes Services AWS, comptes ou principaux, vous devez définir et associer explicitement une politique d'accès à la rubrique. Cela est conforme au principe du moindre privilège, garantissant qu'aucun accès involontaire n'est accordé par défaut.
- Tests et validation. Après avoir défini la politique, testez l'intégration en publiant des messages dans la rubrique Amazon SNS et en vérifiant qu'ils sont correctement transmis à la file d'attente Amazon SQS. Cela permet de confirmer que la politique est correctement configurée.

Autoriser les notifications d'évènements Amazon S3 à publier dans une rubrique

Pour autoriser un compartiment Amazon S3 d'un autre Compte AWS à publier des notifications d'événements sur votre rubrique Amazon SNS, vous devez configurer la politique d'accès de la rubrique en conséquence. Cela implique de rédiger une politique personnalisée qui accorde l'autorisation au service Amazon S3 à partir du service spécifique, Compte AWS puis d'appliquer cette politique à votre rubrique Amazon SNS.

Voici comment le configurer :

1. Rédigez la politique. La politique doit accorder au service Amazon S3 (s3.amazonaws.com) les autorisations nécessaires pour publier sur votre rubrique Amazon SNS. Vous utiliserez cette `SourceAccount` condition pour vous assurer que seule la personne spécifiée Compte AWS, propriétaire du compartiment Amazon S3, peut publier des notifications sur votre sujet.

   Voici un exemple de politique :

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

- Propriétaire de la rubrique : 111122223333 est l' Compte AWS ID propriétaire de la rubrique Amazon SNS.
- Propriétaire du compartiment Amazon S3 — 444455556666 est l' Compte AWS ID propriétaire du compartiment Amazon S3 qui envoie les notifications.

2. Appliquez la politique. Utilisez cette `SetTopicAttributes` action pour définir cette politique sur votre rubrique Amazon SNS. Cela mettra à jour le contrôle d'accès du sujet afin d'inclure les autorisations spécifiées dans votre politique personnalisée.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informations supplémentaires :

- **`SourceAccount`**Condition d'utilisation. Cette `SourceAccount` condition garantit que seuls les événements provenant de la valeur spécifiée Compte AWS (444455556666 dans ce cas) peuvent déclencher la rubrique Amazon SNS. Il s'agit d'une mesure de sécurité visant à empêcher les comptes non autorisés d'envoyer des notifications à votre sujet.

- Autres services de soutien**`SourceAccount`**. La `SourceAccount` condition est prise en charge par les services suivants. Il est essentiel d'utiliser cette condition lorsque vous souhaitez restreindre l'accès à votre rubrique Amazon SNS en fonction du compte d'origine.
  - Amazon API Gateway
  - Amazon CloudWatch
  - Amazon DevOps Guru
  - Amazon EventBridge
  - GameLift Serveurs Amazon
  - 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
- Tests et validation. Après avoir appliqué la politique, testez la configuration en déclenchant un événement dans le compartiment Amazon S3 et en confirmant qu'il a bien été publié sur votre rubrique Amazon SNS. Cela permettra de garantir que votre politique est correctement configurée.
- Bonnes pratiques en matière de sécurité. Passez régulièrement en revue et auditez vos politiques thématiques Amazon SNS pour vous assurer qu'elles sont conformes à vos exigences de sécurité. Limiter l'accès aux seuls comptes et services fiables est essentiel pour garantir la sécurité des opérations.

Autoriser Amazon SES à publier sur une rubrique appartenant à un autre compte

Vous pouvez autoriser une autre Service AWS personne à publier sur une rubrique appartenant à une autre personne 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.

JSON

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
  ]
}
```

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
- GameLift Serveurs Amazon
- 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. Lorsque cette option est prise en charge, la valeur sera l'identifiant de AWS compte à 12 chiffres pour le compte duquel le service publie les données. Certains services prennent en charge l'un et d'autres prennent en charge l'autre.

- Découvrez comment Autoriser les notifications d'évènements Amazon S3 à publier dans une rubrique les notifications Amazon S3 sont utilisées `aws:SourceAccount` et consultez la liste des AWS services compatibles avec cette condition.
- Découvrez Autoriser Amazon SES à publier sur une rubrique appartenant à un autre compte comment Amazon SES l'utilise `aws:SourceOwner` et la liste des AWS services qui répondent à cette condition.

Autoriser les comptes d'une organisation AWS Organizations à publier sur un sujet dans un autre compte

Le AWS Organizations service vous aide à gérer de manière centralisée la facturation, à contrôler l'accès et la sécurité, et à partager les ressources entre vos 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 membre Compte AWS d'une organisation `myOrgId` peut publier sur Amazon SNS la rubrique associée `MyTopic` à son 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

Suivez les étapes ci-dessous pour appeler une rubrique Amazon SNS contenant une CloudWatch alarme différente. Comptes AWS Cet exemple utilise deux comptes :

- Le compte A est utilisé pour créer l' CloudWatch alarme.
- Le compte B est utilisé pour créer un sujet SNS.

Créez un sujet SNS dans le compte B

1. Connectez-vous à la console Amazon SNS.
2. Dans le panneau de navigation, choisissez Rubriques, puis Créer une rubrique.
3. Choisissez Standard pour le type de sujet, puis créez un nom pour le sujet.

4. Choisissez Créer un sujet, puis copiez l'ARN du sujet.
5. Dans le panneau de navigation, choisissez Abonnements, puis Créer un abonnement.
6. Ajoutez l'ARN du sujet dans la section ARN du sujet, choisissez E-mail comme protocole, puis entrez une adresse e-mail.
7. Choisissez Créer un abonnement, puis vérifiez vos e-mails pour confirmer l'abonnement.

Créez une CloudWatch alarme dans le compte A

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le volet de navigation, sélectionnez Alarmes, puis sélectionnez Créer des alarmes.
3. Si vous n'avez pas encore créé d'alarme, créez-en une maintenant. Sinon, sélectionnez votre métrique, puis fournissez les détails du seuil et des paramètres de comparaison.
4. Dans Configurer les actions, sous Notifications, choisissez Utiliser l'ARN du sujet pour notifier les autres comptes, puis entrez l'ARN du sujet à partir du compte B.
5. Donnez un nom à l'alarme, puis choisissez Créer une alarme.

Mettre à jour la politique d'accès de la rubrique SNS dans le compte B

1. Connectez-vous à la console Amazon SNS.
2. Dans le volet de navigation, choisissez Rubriques, puis sélectionnez la rubrique.
3. Choisissez Modifier, puis ajoutez ce qui suit à la politique :

> **Note**
>
> Remplacez les exemples de valeurs de la politique ci-dessous par les vôtres.

JSON

```
{
    "Version": "2008-10-17",
    "Id": "__default_policy_ID",
    "Statement": [
        {
            "Sid": "__default_statement_ID",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "SNS:GetTopicAttributes",
                "SNS:SetTopicAttributes",
                "SNS:AddPermission",
                "SNS:RemovePermission",
                "SNS:DeleteTopic",
                "SNS:Subscribe",
                "SNS:ListSubscriptionsByTopic",
                "SNS:Publish"
            ],
            "Resource": "arn:aws:cloudwatch:us-west-1:111122223333:alarm:",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
west-1:111122223333:alarm:"
                }
            }
        }
    ]
}
```

Testez l'alarme

Pour tester l'alarme, modifiez le seuil d'alarme en fonction des points de données métriques ou modifiez manuellement l'état de l'alarme. Lorsque vous modifiez le seuil d'alarme ou l'état de l'alarme, vous recevez une notification par e-mail.

Solution pour l'utilisation d'une rubrique Amazon SNS locale et le transfert de messages

Suivez les étapes suivantes pour activer les notifications CloudWatch Amazon SNS entre comptes pour les alarmes :

1. Créez une rubrique Amazon SNS dans le même compte que l'CloudWatchalarme (111122223333).
2. Abonnez une fonction Lambda ou une EventBridgerègle Amazon à cette rubrique Amazon SNS.
3. La fonction ou la EventBridge règle Lambda peut ensuite publier le message sur la rubrique Amazon SNS du compte cible (444455556666).

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"
      }
    }
  }]
}
```

Comment Amazon SNS fonctionne 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

Fonctionnalité IAM	Prise en charge d'Amazon SNS
ACLs	Non
ABAC (identifications dans les politiques)	Partielle
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Rôles de service	Oui
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont Amazon SNS et les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez les AWS services compatibles avec IAM dans le guide de l'utilisateur IAM.

AWS politiques gérées pour Amazon Simple Notification Service

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : Amazon SNSFull Access

`AmazonSNSFullAccess`fournit un accès complet à Amazon SNS à l'aide du. AWS Management Console Cette politique inclut également les actions de lecture et d'écriture suivantes en AWS End User Messaging SMS cas d'appel via Amazon SNS. Vous pouvez associer cette politique à vos utilisateurs, groupes ou rôles.

Détails de l'autorisation

Les autorisations suivantes s'appliquent uniquement lors de l'utilisation d'Amazon SNS APIs :

- `sns:*`— Permet d'obtenir des autorisations complètes pour effectuer toute action liée à Amazon SNS. Ce caractère générique (*) signifie que l'utilisateur peut exécuter toutes les actions Amazon SNS possibles.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Vous permet de récupérer une liste de numéros de téléphone dont l'envoi de SMS a été vérifié dans le Compte AWS.
- `sms-voice:CreateVerifiedDestinationNumber`— Vous permet de vérifier un nouveau numéro de téléphone à utiliser avec les services de messagerie SMS intégrés AWS.
- `sms-voice:SendDestinationNumberVerificationCode`— Vous permet d'envoyer un code de vérification à un numéro de téléphone en cours de vérification pour la messagerie SMS qu'il contient AWS.
- `sms-voice:SendTextMessage`— Vous permet de créer un nouveau message texte et de l'envoyer au numéro de téléphone du destinataire. `SendTextMessage`envoie uniquement un message SMS à un destinataire à chaque fois qu'il est invoqué.
- `sms-voice:DeleteVerifiedDestinationNumber`— Vous permet de supprimer un numéro de téléphone de la liste des numéros vérifiés dans le Compte AWS
- `sms-voice:VerifyDestinationNumber`— Vous permet de lancer et de terminer le processus de vérification d'un numéro de téléphone à utiliser pour les services de messagerie SMS qu'il contient AWS.
- `sms-voice:DescribeAccountAttributes`— Vous permet de récupérer des informations détaillées sur les attributs au niveau du compte liés aux services de messagerie SMS qu'il contient. AWS
- `sms-voice:DescribeSpendLimits`— Vous permet de récupérer des informations sur les limites de dépenses associées aux services de messagerie SMS au sein du Compte AWS

- `sms-voice:DescribePhoneNumbers`— Vous permet de récupérer des informations détaillées sur les numéros de téléphone associés aux services de messagerie SMS au sein du Compte AWS
- `sms-voice:SetTextMessageSpendLimitOverride`— Vous permet de définir ou de modifier la limite de dépenses pour la messagerie texte par SMS dans le Compte AWS
- `sms-voice:DescribeOptedOutNumbers`— Vous permet de récupérer une liste des numéros de téléphone qui ont choisi de ne pas recevoir de SMS depuis votre AWS compte.
- `sms-voice:DeleteOptedOutNumber`— Vous permet de supprimer un numéro de téléphone de la liste des numéros désactivés dans le Compte AWS

`AmazonSNSFullAccess`exemple de politique

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSFullAccess",
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:CreateVerifiedDestinationNumber",
                "sms-voice:SendDestinationNumberVerificationCode",
                "sms-voice:SendTextMessage",
                "sms-voice:DeleteVerifiedDestinationNumber",
                "sms-voice:VerifyDestinationNumber",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:SetTextMessageSpendLimitOverride",
                "sms-voice:DescribeOptedOutNumbers",
                "sms-voice:DeleteOptedOutNumber"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Pour consulter les autorisations associées à cette politique, consultez Amazon SNSFull Access dans le manuel AWS Managed Policy Reference.

AWS politique gérée : Amazon SNSRead OnlyAccess

`AmazonSNSReadOnlyAccess`fournit un accès en lecture seule à Amazon SNS à l'aide du. AWS Management Console Cette politique inclut également les actions en lecture seule suivantes en AWS End User Messaging SMS cas d'appel via Amazon SNS. Vous pouvez associer cette politique à vos utilisateurs, groupes et rôles.

Détails de l'autorisation

Les autorisations suivantes s'appliquent uniquement lors de l'utilisation d'Amazon SNS APIs :

- `sns:GetTopicAttributes`— Vous permet de récupérer les attributs d'une rubrique Amazon SNS. Cela inclut des informations telles que l'ARN (Amazon Resource Name) du sujet, la liste des abonnés, les politiques de livraison, les politiques de contrôle d'accès et toute autre métadonnée associée au sujet.
- `sns:List*`— Vous permet d'effectuer n'importe quelle opération commençant par `List` pour les ressources Amazon SNS. Cela inclut les autorisations permettant de répertorier divers éléments liés à Amazon SNS, tels que :
  - `sns:ListTopics`— Vous permet de récupérer une liste de toutes les rubriques Amazon SNS dans le. Compte AWS
  - `sns:ListSubscriptions`— Vous permet de récupérer la liste de tous les abonnements aux rubriques Amazon SNS.
  - `sns:ListSubscriptionsByTopic`— Vous permet de répertorier tous les abonnements relatifs à une rubrique Amazon SNS spécifique.
  - `sns:ListPlatformApplications`— Vous permet de répertorier toutes les applications de plateforme créées pour les notifications push mobiles.

  - `sns:ListEndpointsByPlatformApplication`— Permet de répertorier tous les points de terminaison associés à une application de plateforme.
- `sns:CheckIfPhoneNumberIsOptedOut`— Vous permet de vérifier si un numéro de téléphone spécifique a choisi de ne pas recevoir de SMS via Amazon SNS.
- `sns:GetEndpointAttributes`— Vous permet de récupérer les attributs d'un point de terminaison associé à une application de plateforme Amazon SNS. Cela peut inclure des attributs tels que le statut activé du point de terminaison, les données utilisateur personnalisées et toute autre métadonnée associée au point de terminaison.
- `sns:GetDataProtectionPolicy`— Vous permet de récupérer la politique de protection des données associée à une rubrique Amazon SNS.
- `sns:GetPlatformApplicationAttributes`— Vous permet de récupérer les attributs d'une application de plateforme Amazon SNS. Les applications de plateforme sont utilisées dans Amazon SNS pour envoyer des notifications push aux appareils mobiles via des services tels que Apple Push Notification Service (APNS) ou Firebase Cloud Messaging (FCM).
- `sns:GetSMSAttributes`— Vous permet de récupérer les paramètres SMS par défaut pour Compte AWS.
- `sns:GetSMSSandboxAccountStatus`— Vous permet de récupérer l'état actuel du sandbox SMS pour votre Compte AWS.
- `sns:GetSubscriptionAttributes`— Vous permet de récupérer les attributs d'un abonnement spécifique à une rubrique Amazon SNS.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Vous permet de consulter ou de récupérer une liste de numéros de téléphone dont l'envoi de SMS a été vérifié dans le Compte AWS
- `sms-voice:DescribeAccountAttributes`— Vous permet d'afficher ou de récupérer des informations sur les attributs au niveau du compte liés aux services de messagerie SMS qu'il contient. AWS
- `sms-voice:DescribeSpendLimits`— Vous permet de consulter ou de récupérer des informations sur les limites de dépenses associées aux services de messagerie SMS au sein de votre Compte AWS
- `sms-voice:DescribePhoneNumbers`— Vous permet de consulter ou de récupérer des informations sur les numéros de téléphone utilisés pour les services de messagerie SMS au sein du Compte AWS
- `sms-voice:DescribeOptedOutNumbers`— Vous permet de consulter ou de récupérer une liste des numéros de téléphone qui ont choisi de ne pas recevoir de SMS de votre Compte AWS

AmazonSNSReadOnlyAccessexemple de politique

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "sns:GetTopicAttributes",
                "sns:List*",
                "sns:CheckIfPhoneNumberIsOptedOut",
                "sns:GetEndpointAttributes",
                "sns:GetDataProtectionPolicy",
                "sns:GetPlatformApplicationAttributes",
                "sns:GetSMSAttributes",
                "sns:GetSMSSandboxAccountStatus",
                "sns:GetSubscriptionAttributes"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:DescribeOptedOutNumbers"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Pour consulter les autorisations associées à cette politique, consultez Amazon SNSFull Access dans le manuel AWS Managed Policy Reference.

Mises à jour des politiques gérées par Amazon SNS AWS

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour Amazon SNS depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique des documents Amazon SNS.

Modification	Description	Date
AmazonSNSFullAccess – Mise à jour d’une stratégie existante	Amazon SNS a ajouté de nouvelles autorisations pour permettre un accès complet à Amazon SNS à l'aide du. AWS Management Console	24/09/2024
Amazon SNSRead OnlyAccess — Mise à jour d'une politique existante	Amazon SNS a ajouté de nouvelles autorisations pour autoriser l'accès en lecture seule à Amazon SNS à l'aide du. AWS Management Console	24/09/2024
Amazon SNS a commencé à suivre les modifications	Amazon SNS a commencé à suivre les modifications apportées à ses politiques AWS gérées.	27/08/2024

Actions de politique pour Amazon SNS

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON 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 stratégie portent généralement le même nom que l'opération AWS d'API 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 politique 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 AWS JSON 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 auxquels 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 consulter la liste des types de ressources Amazon SNS et leurs caractéristiques ARNs, consultez la section Actions définies par Amazon Simple Notification Service dans le Service Authorization Reference. 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

Prend en charge les clés de condition de politique spécifiques au service : oui

Les administrateurs peuvent utiliser les politiques AWS JSON 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 `OR` opération 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 d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales 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).

ACLs sur Amazon SNS

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs 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) : partiellement

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises aux entités IAM (utilisateurs ou rôles) et à de nombreuses AWS ressources. 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 étiquettes, vous devez fournir les informations d'étiquette 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 ABAC, consultez Définition d'autorisations avec l'autorisation 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 lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation d'IAM dans le guide de l'utilisateur d'IAM.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, 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 Passage d'un rôle utilisateur à un rôle IAM (console) dans le Guide de l'utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide de l' AWS API AWS CLI or. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires 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 utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une

action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. 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 rôles de 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 un rôle 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.

> **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 apparaissent dans votre Compte AWS répertoire et appartiennent au 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 effectuer de tâches à l'aide de l'API AWS Management Console, AWS Command Line Interface (AWS CLI) ou de AWS l'API. 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 assumer 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 (console) 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 de chaque type de ressource, consultez la section Actions, ressources et clés de condition ARNs pour Amazon Simple Notification Service dans la référence d'autorisation du service.

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 :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez 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.

- Utilisez 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 par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM 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 politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec 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 de consulter les informations relatives aux ressources Amazon SNS de votre. Compte AWS Si vous créez une politique 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 politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la console Amazon SNS, associez également Amazon `ConsoleAccess` SNS `ReadOnly` AWS ou la politique gérée 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 courants. 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 fonctionnalité 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 en résultant 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 annule 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 (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités des comptes membres, y compris pour chaque utilisateur Compte AWS root. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de contrôle des ressources (RCPs) : RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les autorisations effectives pour les identités, y compris l'utilisateur Compte AWS root, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organizations RCPs, y compris une liste de ces Services AWS supports RCPs, consultez la section Resource control policies (RCPs) dans le guide de AWS Organizations l'utilisateur.

- 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 séance en résultant 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 annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des 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 permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or 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 l'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 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 Définition d'autorisations IAM personnalisées avec des politiques gérées par le client 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 les ressources	Oui

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. 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. 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 intercompte à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Lorsque le principal et la ressource sont différents Comptes AWS, un administrateur IAM du compte sécurisé doit également accorder à l'entité principale (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 Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de politiques basées sur l'identité avec 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 de Compte AWS vos utilisateurs 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 restreindre l'accès uniquement aux utilisateurs de votre AWS compte, pas aux autres 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 utilisateurs ou à des utilisateurs appartenant aux vôtres. 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 `Subscribe` SNS pour le sujet intitulé topic_xyz dans Compte AWS votre. La politique IAM est attachée aux utilisateurs Bob et Susan (ce qui signifie que Bob et Susan disposent des autorisations indiquées 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 une différence entre les politiques AWS IAM et Amazon SNS : le système de politique Amazon SNS vous permet d'accorder des autorisations à Comptes AWS d'autres personnes, 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 l'autorisation d'utiliser n'importe quel `Subscribe` sujet, tandis que la politique Amazon SNS lui accorde l'autorisation d'`Publish`utiliser un sujet spécifique (topic_xyz). Compte AWS Le diagramme suivant illustre le concept.

Si Bob envoyait une demande d'abonnement à n'importe quel sujet du AWS compte, la politique IAM autoriserait cette 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.

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 ARNs, consultez le Guide ARNsde 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 vous avez nommé une rubrique MyTopic dans chacune des différentes régions prises en charge par Amazon SNS, vous pouvez spécifier les rubriques 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 générales AWS suivantes, ainsi que certaines clés spécifiques aux services.

Pour obtenir la liste des clés de condition prises en charge par chacune d'entre elles Service AWS, voir Actions, ressources et clés de condition Services AWS dans le guide de l'utilisateur IAM. Pour obtenir une liste des clés de condition qui peuvent être utilisées à plusieurs reprises Services AWS, voir les clés contextuelles de condition AWS globales 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 : Donnez aux utilisateurs la Compte AWS possibilité de s'abonner à des sujets

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 dispose d'une politique Amazon SNS Compte AWS, il peut être plus facile d'utiliser une politique Amazon SNS. Cependant, toute personne de l'entreprise du partenaire possédant les informations AWS de sécurité peut publier des messages sur le sujet. 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"
    }
  ]
}
```

Politiques Amazon SNS pour les points de terminaison de messagerie

Amazon SNS normalise les adresses e-mail dans les politiques d'accès aux terminaux en supprimant les espaces de début et de fin, en convertissant tous les caractères en minuscules et en normalisant les caractères spéciaux et les séquences d'échappement. C'est pourquoi vous devez utiliser des adresses e-mail en minuscules dans les définitions de vos politiques afin de garantir la bonne correspondance des demandes d'abonnement.

Exemple de politique :

```
{
    "Condition": {
        "StringEquals": {
            "sns:Endpoint": "user.name@example.com"
        }
    }
}
```

Les adresses e-mail suivantes répondent à la condition ci-dessus :

- `user.name@example.com`
- `USER.NAME@EXAMPLE.COM`
- `User.N\ame@Example.com`

Gestion des politiques IAM Amazon SNS personnalisées

Les politiques IAM personnalisées vous permettent de spécifier des autorisations pour des utilisateurs, des groupes ou des rôles IAM individuels, en accordant ou en restreignant l'accès à des AWS ressources et à des actions spécifiques. Lorsque vous gérez les ressources Amazon SNS, les politiques IAM personnalisées vous permettent d'adapter les autorisations d'accès en fonction des exigences opérationnelles et de sécurité de votre organisation.

Suivez les étapes suivantes pour gérer les politiques IAM personnalisées pour Amazon SNS :

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le volet de navigation, sélectionnez Policies.

3. Pour créer une nouvelle stratégie IAM personnalisée, choisissez Create policy, puis choisissez SNS. Pour modifier une politique existante, sélectionnez-la dans la liste et choisissez Modifier la politique.
4. Dans l'éditeur de politiques, définissez les autorisations d'accès aux ressources Amazon SNS. Vous pouvez spécifier des actions, des ressources et des conditions en fonction de vos besoins spécifiques.
5. Pour accorder des autorisations pour les actions Amazon SNS, incluez les actions Amazon SNS pertinentes `sns:Publish` telles que`sns:Subscribe`, `sns:DeleteTopic` et dans votre politique IAM. Définissez l'ARN (Amazon Resource Name) des rubriques Amazon SNS auxquelles les autorisations s'appliquent.
6. Spécifiez les utilisateurs, groupes ou rôles IAM auxquels la politique doit être attachée. Vous pouvez associer la politique directement à des utilisateurs ou à des groupes IAM, ou l'associer à des rôles IAM utilisés par Services AWS ou par des applications.
7. Passez en revue la configuration de la politique IAM pour vous assurer qu'elle correspond à vos exigences en matière de contrôle d'accès. Une fois la vérification effectuée, enregistrez les modifications de politique.
8. Associez la politique IAM personnalisée aux utilisateurs, groupes ou rôles IAM concernés au sein de votre. Compte AWS Cela leur accorde les autorisations définies dans la politique de gestion des ressources Amazon SNS.

Utilisation d'informations d'identification de sécurité temporaires avec Amazon SNS

AWS Identity and Access Management (IAM) vous permet d'octroyer des informations d'identification de sécurité temporaires aux utilisateurs et aux applications qui ont besoin d'accéder à vos AWS ressources. Ces informations d'identification de sécurité temporaires sont principalement utilisées pour les rôles IAM et l'accès fédéré via des protocoles standard tels que SAML et OpenID Connect (OIDC).

Pour gérer efficacement l'accès aux AWS ressources, il est essentiel de comprendre les concepts clés suivants :

- Rôles IAM — Les rôles sont utilisés pour déléguer l'accès aux AWS ressources. Les rôles peuvent être assumés par des entités telles que des EC2 instances Amazon, des fonctions Lambda ou des utilisateurs d'autres entités. Comptes AWS

- Utilisateurs fédérés : il s'agit d'utilisateurs authentifiés via des fournisseurs d'identité externes (IdPs) à l'aide de SAML ou OIDC. L'accès fédéré est recommandé pour les utilisateurs humains, tandis que les rôles IAM doivent être utilisés pour les applications logicielles.
- Rôles n'importe où : pour les applications externes nécessitant un AWS accès, vous pouvez utiliser IAM Roles Anywhere pour gérer l'accès en toute sécurité sans créer d'informations d'identification à long terme.

Vous pouvez utiliser des informations d'identification de sécurité temporaires pour envoyer des demandes à Amazon SNS. Les bibliothèques SDKs et API calculent la signature nécessaire à l'aide de ces informations d'identification pour authentifier vos demandes. Les demandes dont les informations d'identification ont expiré seront refusées par Amazon SNS.

Pour plus d'informations sur les informations d'identification de sécurité temporaires, reportez-vous aux sections Utilisation des rôles IAM et Fourniture d'accès aux utilisateurs authentifiés de manière externe (fédération d'identité) dans le guide de l'utilisateur IAM.

Example Exemple de requête HTTPS

L'exemple suivant montre comment authentifier une demande Amazon SNS à l'aide des informations de sécurité temporaires obtenues AWS Security Token Service auprès de (STS).

```
https://sns.us-east-2.amazonaws.com/
?Action=CreateTopic
&Name=My-Topic
&SignatureVersion=4
&SignatureMethod=AWS4-HMAC-SHA256
&Timestamp=2023-07-05T12:00:00Z
&X-Amz-Security-Token=SecurityTokenValue
&X-Amz-Date=20230705T120000Z
&X-Amz-Credential=<your-access-key-id>/20230705/us-east-2/sns/aws4_request
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Étapes pour authentifier la demande

1. Obtenir des informations d'identification de sécurité temporaires : AWS STS à utiliser pour assumer un rôle ou obtenir des informations d'identification utilisateur fédérées. Cela vous fournira un identifiant de clé d'accès, une clé d'accès secrète et un jeton de sécurité.

2. Élaborez la demande : incluez les paramètres requis pour votre action Amazon SNS (par exemple, CreateTopic) et assurez-vous d'utiliser le protocole HTTPS pour sécuriser les communications.
3. Signez la demande — Utilisez le processus de AWS signature version 4 pour signer votre demande. Cela implique de créer une demande canonique string-to-sign, puis de calculer la signature. Pour en savoir plus sur AWS la version 4 de Signature, consultez la section Utiliser la signature de signature version 4 dans le guide de l'utilisateur d'Amazon EBS.
4. Envoyer la demande : incluez-le X-Amz-Security-Token dans l'en-tête de votre demande pour transmettre les informations de sécurité temporaires à Amazon SNS.

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.
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.

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 besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification 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 vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un 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 un rôle de 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 besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à moi 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 (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon SNS est compatible avec ces fonctionnalités, veuillez consulter Comment Amazon SNS fonctionne avec IAM (Fonctionnement d'Amazon Simple Notification Service avec IAM).
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par 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 en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Journalisation et surveillance dans Amazon SNS

Amazon SNS vous permet de suivre et de surveiller l'activité de messagerie en enregistrant les appels d'API CloudTrail et en surveillant les sujets avec. CloudWatch Ces outils vous aident à mieux comprendre la distribution des messages, à résoudre les problèmes et à garantir l'intégrité de vos flux de travail de messagerie. Cette rubrique aborde les points suivants :

- Journalisation des appels d'API AWS SNS à l'aide de AWS CloudTrail. Cet enregistrement vous permet de suivre les actions effectuées sur vos rubriques Amazon SNS, telles que la création de rubriques, la gestion des abonnements et la publication de messages. En analysant CloudTrail les journaux, vous pouvez identifier qui a fait des demandes d'API spécifiques et à quel moment ces demandes ont été faites, ce qui vous permet d'auditer et de résoudre les problèmes liés à votre utilisation d'Amazon SNS.
- Surveillance des rubriques Amazon SNS à l'aide de CloudWatch. CloudWatch fournit des indicateurs qui vous permettent d'observer les performances et l'état de vos rubriques Amazon SNS en temps réel. Configurez des alarmes en fonction de ces indicateurs, ce qui vous permettra de réagir rapidement à toute anomalie, telle qu'un échec de livraison ou une latence élevée des messages. Cette fonctionnalité de surveillance garantit que vous pouvez maintenir la fiabilité de

votre système de messagerie basé sur le SNS en résolvant les problèmes potentiels de manière proactive.

Journalisation des appels d'API AWS SNS à l'aide de AWS CloudTrail

AWS Le SNS est intégré à AWS CloudTrailun service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS. CloudTrail capture tous les appels d'API pour SNS sous forme d'événements. Les appels capturés incluent des appels provenant de la console SNS et des appels de code vers les opérations de l'API SNS. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite au SNS, l'adresse IP à partir de laquelle la demande a été faite, la date à laquelle elle a été faite et des informations supplémentaires.

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 :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou d'utilisateur root.
- Si la demande a été faite au nom d'un utilisateur du centre d'identité 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.

CloudTrail est actif dans votre compte Compte AWS lorsque vous créez le compte et vous avez automatiquement accès à l'historique des CloudTrail événements. L'historique des CloudTrail événements fournit un enregistrement consultable, consultable, téléchargeable et immuable des 90 derniers jours des événements de gestion enregistrés dans un. Région AWS Pour plus d'informations, consultez la section Utilisation de l'historique des CloudTrail événements dans le guide de AWS CloudTrail l'utilisateur. La consultation de CloudTrail l'historique des événements est gratuite.

Pour un enregistrement continu des événements de vos 90 Compte AWS derniers jours, créez un magasin de données sur les événements de Trail ou CloudTrailLake.

CloudTrail sentiers

Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Tous les sentiers créés à l'aide du AWS Management Console sont multirégionaux. Vous ne pouvez créer un journal de suivi en une ou plusieurs régions à l'aide de l' AWS CLI. Il est recommandé de créer un parcours multirégional, car vous capturez l'activité dans l'ensemble Régions AWS de votre compte. Si vous créez un journal de suivi pour une seule région, il convient de n'afficher que les événements enregistrés dans le journal de suivi pour une seule région Région AWS. Pour plus d'informations sur les journaux de suivi, consultez Créez un journal de suivi dans vos Compte AWS et Création d'un journal de suivi pour une organisation dans le AWS CloudTrail Guide de l'utilisateur.

Vous pouvez envoyer une copie de vos événements de gestion en cours dans votre compartiment Amazon S3 gratuitement CloudTrail en créant un journal. Toutefois, des frais de stockage Amazon S3 sont facturés. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification. Pour obtenir des informations sur la tarification Amazon S3, consultez Tarification Amazon S3.

CloudTrail Stockages de données sur les événements du lac

CloudTrail Lake vous permet d'exécuter des requêtes SQL sur vos événements. CloudTrail Lake convertit les événements existants au format JSON basé sur les lignes au format Apache ORC. ORC est un format de stockage en colonnes qui est optimisé pour une récupération rapide des données. Les événements sont agrégés dans des magasins de données d'événement. Ceux-ci constituent des collections immuables d'événements basées sur des critères que vous sélectionnez en appliquant des sélecteurs d'événements avancés. Les sélecteurs que vous appliquez à un magasin de données d'événement contrôlent les événements qui persistent et que vous pouvez interroger. Pour plus d'informations sur CloudTrail Lake, consultez la section Travailler avec AWS CloudTrail Lake dans le guide de AWS CloudTrail l'utilisateur.

CloudTrail Les stockages et requêtes de données sur les événements de Lake entraînent des coûts. Lorsque vous créez un magasin de données d'événement, vous choisissez l'option de tarification que vous voulez utiliser pour le magasin de données d'événement. L'option de tarification détermine le coût d'ingestion et de stockage des événements, ainsi que les périodes de conservation par défaut et maximale pour le magasin de données d'événement. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Événements liés aux données SNS dans CloudTrail

Les événements de données fournissent des informations sur les opérations de ressources effectuées sur ou dans une ressource (par exemple, lecture ou écriture de données dans un objet Amazon S3). Ils sont également connus sous le nom opérations de plans de données. Les événements de données sont souvent des activités dont le volume est élevé. Par défaut, CloudTrail n'enregistre pas les événements liés aux données. L'historique des CloudTrail événements n'enregistre pas les événements liés aux données.

Des frais supplémentaires s'appliquent pour les événements de données. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Vous pouvez enregistrer les événements de données pour les types de ressources SNS à l'aide de la CloudTrail console ou AWS CLI des opérations CloudTrail d'API. Pour plus d'informations sur la façon de journaliser les événements de données, consultez Journalisation des événements de données avec la AWS Management Console et Journalisation des événements de données avec l' AWS Command Line Interface dans le Guide de l'utilisateur AWS CloudTrail .

Le tableau suivant répertorie les types de ressources SNS pour lesquels vous pouvez enregistrer des événements de données. La colonne Type d'événement de données (console) indique la valeur à choisir dans la liste des types d'événements de données de la CloudTrail console. La colonne de valeur resources.type indique la **`resources.type`** valeur que vous devez spécifier lors de la configuration de sélecteurs d'événements avancés à l'aide du ou. AWS CLI CloudTrail APIs La CloudTrail colonne Données APIs enregistrées indique les appels d'API enregistrés CloudTrail pour le type de ressource.

Type d'événement de données (console)	valeur resources.type	Données APIs enregistrées sur CloudTrail
Rubrique SNS	`AWS::SNS::Topic`	• `Publish` • `PublishBatch`
Point de terminaison de la plateforme SNS	`AWS::SNS::PlatformEndpoint`	• Publish Pour plus de détails, consultez `AdvancedEventSelector` la

Type d'événement de données (console)	valeur resources.type	Données APIs enregistrées sur CloudTrail
		référence de AWS CloudTrail l'API.

> **Note**
>
> Le type de ressource SNS n'`AWS::SNS::PhoneNumber`est pas enregistré. CloudTrail

Vous pouvez configurer des sélecteurs d'événements avancés pour filtrer les champs `eventName`, `readOnly` et `resources.ARN` afin de ne journaliser que les événements importants pour vous. Pour plus d'informations sur ces champs, voir AdvancedFieldSelector dans la Référence d'API AWS CloudTrail

Pour plus d'informations sur la journalisation des événements liés aux données, consultez les sections Enregistrement des événements liés aux données avec le AWS Management Console et Enregistrement des événements liés aux données AWS CLI dans le guide de CloudTrail l'utilisateur.

Événements de gestion SNS dans CloudTrail

Les événements de gestion fournissent des informations sur les opérations de gestion effectuées sur les ressources de votre Compte AWS. Ils sont également connus sous le nom opérations de plan de contrôle. Par défaut, CloudTrail enregistre les événements de gestion.

AWS SNS enregistre les opérations du plan de contrôle SNS suivantes en CloudTrail tant qu'événements de gestion.

- `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 Unsubscribeactions ConfirmSubscriptionou sont invoquées, elles ne seront pas connectées. 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'`ConfirmSubscription`action ne serait pas enregistrée CloudTrail.

Exemples d'événements SNS

Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'opération d'API demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics. Les événements n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre un CloudTrail événement illustrant les **`DeleteTopic`** actions **`ListTopicsCreateTopic`**, et.

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

L'exemple suivant montre un CloudTrail événement illustrant l'**`Publish`**action.

```
{
  "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"
  }
}
```

L'exemple suivant montre un CloudTrail événement illustrant l'**`PublishBatch`**action.

```
{
  "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"
  }
}
```

Pour plus d'informations sur le contenu des CloudTrail enregistrements, voir le contenu des CloudTrail enregistrements dans le Guide de AWS CloudTrail l'utilisateur.

Surveillance des rubriques Amazon SNS à l'aide de CloudWatch

Amazon SNS et Amazon CloudWatch sont intégrés afin que vous puissiez collecter, consulter et analyser les métriques pour chaque notification Amazon SNS active. Une fois que vous avez configuré CloudWatch Amazon SNS, vous pouvez mieux comprendre les performances de vos rubriques Amazon SNS, de vos notifications push et de vos envois 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 auxquelles Amazon SNS envoie CloudWatch, consultez. Métriques Amazon SNS Pour plus d'informations sur les notifications push Amazon SNS, consultez Envoi de notifications push mobiles avec Amazon SNS.

> **Note**
>
> Les métriques que vous configurez CloudWatch pour vos rubriques Amazon SNS sont automatiquement collectées et transmises à CloudWatch intervalles d'une minute. Ces indicateurs sont collectés sur tous les sujets qui répondent aux CloudWatch directives relatives à l'activité physique. Un sujet est considéré comme actif jusqu' CloudWatch à six heures après la dernière activité (c'est-à-dire tout appel d'API) sur le sujet.
> Les métriques Amazon SNS indiquées dans le présent document sont gratuites CloudWatch ; elles sont fournies dans le cadre du service Amazon SNS.

Afficher CloudWatch les statistiques pour Amazon SNS

Vous pouvez surveiller les métriques pour Amazon SNS à l'aide de la CloudWatch console, CloudWatch de sa propre interface de ligne de commande (CLI) ou de manière programmatique 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 CloudWatch console

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 Topic Metrics, puis choisissez NumberOfMessagesPublished, le nombre moyen de messages Amazon SNS publiés sur une période d'une minute sur une période de 6 heures est affiché.
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`.

Définissez des CloudWatch alarmes 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, de sorte que lorsque le seuil que vous avez spécifié est atteint pendant la période d'échantillonnage, une notification par e-mail soit envoyée pour vous informer de l'événement.

Pour définir des alarmes à l'aide de la CloudWatch console

1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la 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 CloudWatch souhaitez vous envoyer un e-mail lorsque l'état d'alarme est atteint, choisissez une rubrique Amazon SNS existante ou choisissez Create New Email Topic. 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 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 CloudWatch et les alarmes, consultez la CloudWatchdocumentation.

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, plate-forme et TopicName Statistiques valides : Sum
AWS/SNS	`NumberOfNotificationsDelivered`	Nombre de messages transmis avec succès à des points de terminaison d'abonnement à partir de vos rubriques Amazon SNS.

Espace de noms	Métrique	Description
		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 Dimensions valides : application PhoneNumber, plate-forme et TopicName Statistiques valides : Sum

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 tentative s 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, plate-forme 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, plate-forme 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 : Nombre Dimensions valides : application PhoneNumber, plate-forme 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, plate-forme 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, plate-forme 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, plate-forme 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, plate-forme 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, plate-forme 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, plate-forme et TopicName Statistiques valides : somme, moyenne
AWS/SNS	`PublishSize`	Taille des messages publiés. Unités : octets Dimensions valides : application PhoneNumber, plate-forme 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 configurer une alarme pour cette métrique afin de savoir quand vos month-to-date frais sont proches du quota mensuel de dépenses par 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 Configuration des préférences de messagerie SMS dans Amazon SNS. Unités : USD Dimensions valides : Aucune Statistiques valides : Sum

Espace de noms	Métrique	Description
AWS/SNS	`SMSSuccessRate`	Taux de diffusions SMS réussies. Unités : Nombre Dimensions valides : PhoneNumber Statistiques valides : somme, moyenne, échantillons de données

Dimensions pour les métriques Amazon SNS

Amazon Simple Notification Service envoie les dimensions suivantes à CloudWatch.

Dimension	Description
`Application`	Filtres sur les objets d'application, qui représentent une applicati on et un appareil enregistrés auprès de l'un des services de notification push pris en charge, tels que APNs FCM.
`Application,Platform`	Filtres sur les objets d'application et de plate-forme, les objets de plate-forme étant destinés aux services de notification push pris en charge, tels que APNs 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`	Filtres sur les objets de la plateforme pour les services de notification push, tels que APNs 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 statistiques d'utilisation suivantes à CloudWatch.

Espace de noms	Service	Métrique	Ressource	Type	Description
AWS/Utilisation	SNS	`ResourceCount`	`NumberOfMessagesPublishedPerAccount`	Ressource	• Le nombre de messages publiés sur vos rubriques Amazon SNS sur votre AWS compte. • Unités : aucune • Statistiques valides : somme
AWS/Utilisation	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Ressource	• Le nombre approximatif de sujets sur votre AWS compte. • 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	• Le nombre approximatif d'abonnements en attente sur votre AWS compte. • 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`	« Hello, World! »	• Le nombre d'appels d'API pour l'API Amazon SNS sélectionnée sur votre AWS compte. • Le contenu n'est pas autorisé dans la section de suivi. 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

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Conformité et gouvernance de la sécurité : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- Référence des services éligibles HIPAA : liste les services éligibles HIPAA. Tous ne Services AWS sont pas éligibles à la loi HIPAA.
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

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 les politiques Amazon SNS pour contrôler l'accès à partir de points de terminaison Amazon VPC spécifiques ou spécifiques. VPCs Cela permet d'isoler efficacement l'accès réseau à une rubrique Amazon SNS donnée uniquement du VPC spécifique au sein du réseau. AWS Pour de plus amples informations,

veuillez consulter 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 fournit 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
- Utiliser les rôles IAM pour les applications et les AWS services qui nécessitent un accès 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 ne demandiez explicitement à quiconque sur Internet de lire ou d'écrire sur votre sujet Amazon SNS, vous devez vous assurer que votre sujet n'est pas accessible au public (accessible par tout le monde dans le monde ou par tout utilisateur authentifié AWS).

- 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

Utiliser les rôles IAM pour les applications et les AWS services qui nécessitent un accès Amazon SNS

Pour que les applications ou les AWS services, tels qu'Amazon EC2, puissent accéder aux rubriques Amazon SNS, ils doivent utiliser des AWS informations d'identification valides dans leurs demandes d' AWS API. Ces informations d'identification ne faisant pas l'objet d'une rotation automatique, vous ne devez pas les stocker AWS directement dans l'application ou l' EC2 instance.

Vous devez utiliser un rôle IAM pour gérer des informations d’identification temporaires pour les applications ou services devant accéder à Amazon SNS. Lorsque 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 clés d'accès) à une EC2 instance ou à un AWS service, tel que AWS Lambda. Le rôle fournit plutôt des autorisations temporaires que les applications peuvent utiliser lorsqu'elles appellent d'autres AWS ressources.

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 Sécurisation des données Amazon SNS grâce au chiffrement côté serveur et Gestion des clés de chiffrement et des coûts Amazon SNS.

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. Toutefois, lorsqu'un sujet est chiffré au repos à l'aide du protocole HTTPS AWS KMS, il est nécessaire d'utiliser le protocole HTTPS pour publier les messages afin de garantir le chiffrement à la fois au repos et en transit. Bien que le sujet ne rejette pas automatiquement les messages HTTP, l'utilisation du protocole HTTPS est nécessaire pour maintenir les normes de sécurité.

AWS recommande d'utiliser le protocole HTTPS au lieu du protocole 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 attaquant basé sur le réseau peut espionner le trafic réseau ou le manipuler à l'aide d'une attaque telle que. man-in-the-middle

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 :

JSON

```
{
    "Id": "ExamplePolicy",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPublishThroughSSLOnly",
            "Action": "SNS:Publish",
            "Effect": "Deny",
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:test-topic"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            },
            "Principal": "*"
        }
    ]
}
```

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 les politiques relatives aux sujets pour contrôler l'accès aux sujets depuis des points de terminaison Amazon VPC spécifiques ou depuis des points spécifiques. VPCs

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 points de terminaison VPCs ou quels points de terminaison VPC ont accès à votre sujet à l'aide d'une politique de sujet.

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 à l'aide de AWS X-Ray

AWS X-Ray collecte des données sur les demandes traitées par votre application et vous permet de visualiser et de filtrer les données afin d'identifier les problèmes potentiels et les opportunités d'optimisation. Pour toute demande retracée envoyée à votre application, vous pouvez consulter des informations détaillées sur la demande, la réponse et les appels que votre application fait aux AWS ressources en aval, aux microservices, aux bases de données et au Web APIs 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 le AWS Management Console pour afficher la carte des connexions entre Amazon SNS et les autres services utilisés par votre application. 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-Ray Guide du développeur.

Suivi actif dans Amazon SNS

Utilisez-le AWS X-Ray pour suivre et analyser les demandes des utilisateurs lorsqu'elles passent par vos rubriques Amazon SNS vers les abonnements Amazon Data Firehose, Amazon AWS LambdaSQS et HTTP/S Endpoint.

Avec X-Ray, vous avez un end-to-end aperçu de chaque demande, ce qui vous permet de :

- Identifiez le nom de votre rubrique Amazon SNS et les services proposés en aval de ses abonnements.
- Analysez les latences, telles que :
  - Temps passé dans la rubrique Amazon SNS avant le traitement.
  - Délais de livraison pour chaque point de terminaison abonné.

> **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 la section Quotas des services de radiographie dans la section Référence AWS générale.

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 .

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 à l'aide de la console AWS

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 à l'aide du SDK AWS

L'exemple de code suivant montre comment activer le suivi actif sur une rubrique Amazon SNS à l'aide du 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 à l'aide de la CLI AWS

L'exemple de code suivant montre comment activer le suivi actif sur une rubrique Amazon SNS à l'aide de la CLI 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 à l'aide de AWS CloudFormation

La AWS CloudFormation pile 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 Amazon SNS

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
Ajout de la prise en charge des terminaux à double pile (IPv4 et IPv6)	Amazon SNS prend désormais en charge IPv6 les demandes d'API, ce qui permet aux clients de se connecter à des points de terminaison publics à l'aide de la technologie IPv4 Dual IPv6 Stack.	3 avril 2025
Amazon SNS a ajouté l'`FifoThroughputScope` attribut pour les rubriques FIFO d'Amazon SNS	Amazon SNS prend en charge FifoThroughputScope cet attribut, qui spécifie le quota de débit et le comportement de déduplication à appliquer pour le sujet FIFO. Les valeurs valides sont `Topic` ou `MessageGroup` .	21 janvier 2025
`AmazonSNSFullAccess` et mises à jour des politiques `AmazonSNSReadOnlyAccess` gérées	Amazon SNS a ajouté de nouvelles autorisations AmazonSNSFullAccess et a AmazonSNSReadOnlyAccess géré des politiques, qui permettent un accès	24 septembre 2024

	supplémentaire à Amazon SNS via le. AWS Management Console	
Intégration à Amazon SNS AWS End User Messaging SMS pour la livraison de messages SMS	Amazon SNS prend en charge de nouvelles fonctionnalités telles que la gestion des ressources SMS, la messagerie bidirectionnelle, les autorisations granulaires des ressources, les règles par pays et la facturation centralisée pour tous les AWS SMS sans apporter de modifications aux configurations ou au réseau AWS SMS mondial utilisé par Amazon SNS.	24 septembre 2024
Soutien du Canada-Ouest (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 prise en charge de la configuration `DeliveryStatusLogging` pour les rubriques Amazon SNS	AWS CloudFormation une assistance est disponible pour la configuration `DeliveryStatusLogging` lors de la création ou de la mise à jour de rubriques Amazon SNS.	7 décembre 2023
Nouveaux opérateurs de filtrage des messages ajoutés	Vous pouvez désormais utiliser la correspondance de suffixes et les opérateurs 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 défaillance 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, il permet AWS X-Ray désormais de suivre et d'analyser les demandes des utilisateurs au fur et à mesure qu'elles passent par vos rubriques FIFO jusqu'à vos abonnemen ts Amazon Data Firehose, Amazon AWS Lambda SQS et aux terminaux. HTTP/S	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 notificat ion.	23 mars 2023
Ajout de la prise en charge du suivi actif	AWS X-Ray suit et analyse les demandes des utilisate urs au fur et à mesure qu'elles passent par les rubriques standard Amazon SNS pour accéder à vos abonnemen ts Amazon Data Firehose, Amazon AWS Lambda SQS et aux terminaux. HTTP/S	8 février 2023
Enregistrement de l'ID d'expéditeur pour Singapour	Des instructions ont été ajoutées pour enregistrer l'expéditeur IDs à 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
SHA256 algorithme de hachage ajouté pour la signature des messages Amazon SNS	Support ajouté pour l'algorithme de SHA256 hachage lors de l'utilisation de la signature de 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 ressource s IAM, telles que les utilisate urs 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'authent ification par jeton Apple pour les notifications push	Vous pouvez autoriser Amazon SNS à envoyer des notifications push à votre appli iOS ou macOS en communiqu ant 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 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 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 terminaux	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
Support pour l'utilisation de DLQs la capture de messages non distribuables	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
Support pour la spécification de valeurs APNs d'en-tête personnalisées	Vous pouvez spécifier une valeur d' APNs en-tête personnalisée.	18 octobre 2019
Support pour le champ d'en-tête `apns-push-type` « » pour APNs	Vous pouvez utiliser le champ `apns-push-type` d'en-tête pour les notifications mobiles envoyées 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

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.