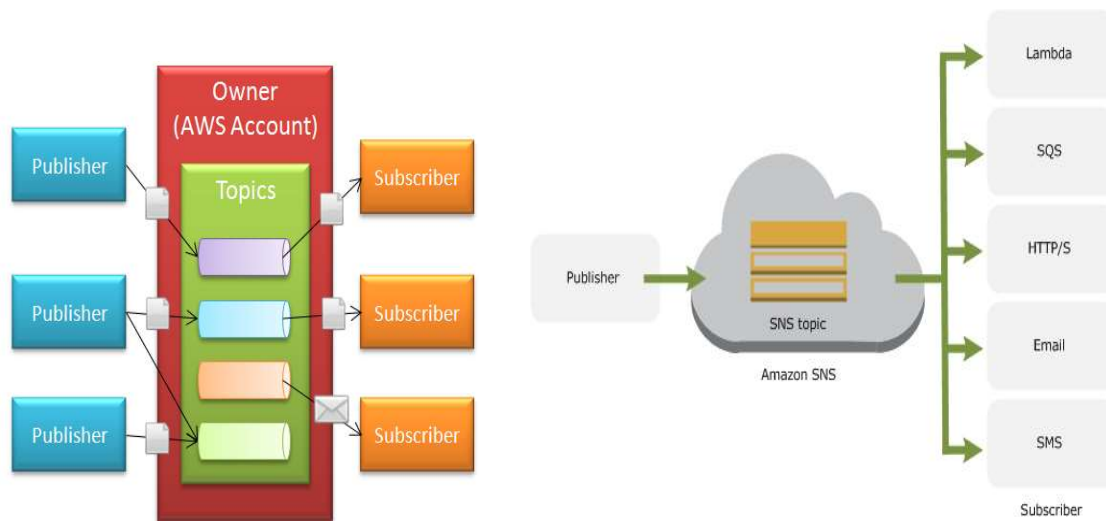


# AMAZON SIMPLE NOTIFICATION SERVICE (SNS)

## 1. What is Amazon Simple Notification Service?

- Amazon Simple Notification Service (SNS) is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications.
- Amazon SNS provides topics for high-throughput, **push-based**, many-to-many messaging.
- Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.
- In Amazon SNS, there are two types of clients:
  - Publishers (or) producers
  - Subscribers (or) consumers.
- Using Amazon SNS topics, your publisher systems can fan out messages to a large number of subscriber endpoints for parallel processing, including Amazon SQS queues, AWS Lambda functions, and HTTP/S webhooks.
- Additionally, SNS can be used to fan out notifications to end users using mobile push, SMS, and email.
- The Amazon Simple Notification Service (SNS) makes it easy for you to build an application in this way. You'll need to know the following terms in order to understand how SNS works:
  - **Topics** are named groups of events or access points, each identifying a specific subject, content, or event type. Each topic has a unique identifier (URI) that identifies the SNS endpoint for publishing and subscribing.
  - **Owners** create topics and control all access to the topic. The owner can define the permissions for all of the topics that they own.
  - **Subscribers** are clients (applications, end-users, servers, or other devices) that want to receive notifications on specific topics of interest to them.
  - **Publishers** send messages to topics. SNS matches the topic with the list of subscribers interested in the topic, and delivers the message to each and every one of them. Here's how it all fits together:
- Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel.
- Subscribers (that is, web servers, email addresses, Amazon SQS queues, AWS Lambda functions) consume or receive the message or notification over one of the supported

protocols (that is, Amazon SQS, HTTP/S, email, SMS, Lambda) when they are subscribed to the topic.



➤ The SNS API is, as the name should imply, clean and simple. Here's what it takes to get started:

1. Call the **CreateTopic** function to create a new topic. Topic names can be made up of upper and lower case letters, numbers, and hyphens and can be up to 256 characters long.
2. Call the **AddPermission** function to establish the set of publishers and subscribers with access to the topic.
3. Subscribers call the **Subscribe** function to express their interest in receiving messages on a particular topic. As part of their request each subscriber must specify a topic, a protocol (HTTP, HTTPS, Email, or Email-JSON), and an endpoint (a URL for HTTP or HTTPS, an email address for either flavor of Email). SNS is also integrated with other AWS services for example, you can have the notifications delivered to an SQS queue. A single subscriber can subscribe to the same topic more than once if desired.
4. As part of the subscription process, SNS will deliver a confirmation message with an embedded token to the endpoint. The subscriber must confirm the subscription by clicking a link in an email or using the **ConfirmSubscription** function in order to initiate message delivery.
5. Publishers call the **Publish** function to post messages to a topic which will immediately trigger delivery of the message to each of the topic's subscribers.

➤ Here are some ideas to get you started:

1. **Monitoring Alert Notification System** – Watch for failure events in a complex system and route information about them to appropriate people based on the type and complexity of the failure. The events could be at the system level (e.g. a process on an

EC2 instance is consuming an excessive amount of memory or CPU time) or at the application level (e.g. an application was not able to communicate with an outside data provider).

2. **News Distribution** – Watch a web site, a Twitter topic, search results, or an RSS feed for changes. Publish the changes to a series of topics (one per news topic) and allow subscribers to choose the topics that they find to be of interest to them.
  3. **Control EC2 Instances** – Subscribe a large array of EC2 instances to a single topic and publish messages to the topic to control the array. This could be system level, with messages used to tell each instance to update its installed software, report on free disk space. Or, it could be application level, with messages to start and stop application processes, change policies, or update reference data tables.
- When using Amazon SNS, you (as the owner) **create a topic** and control access to it by **defining policies** that determine *which publishers and subscribers can communicate with the topic*.
  - A publisher sends messages to topics that they have created or to topics they have permission to publish to. Instead of including a specific destination address in each message, a publisher sends a message to the topic.
  - Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers.
  - Each topic has a unique name that identifies the **Amazon SNS endpoint for publishers** to post messages and subscribers to register for notifications.
  - Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

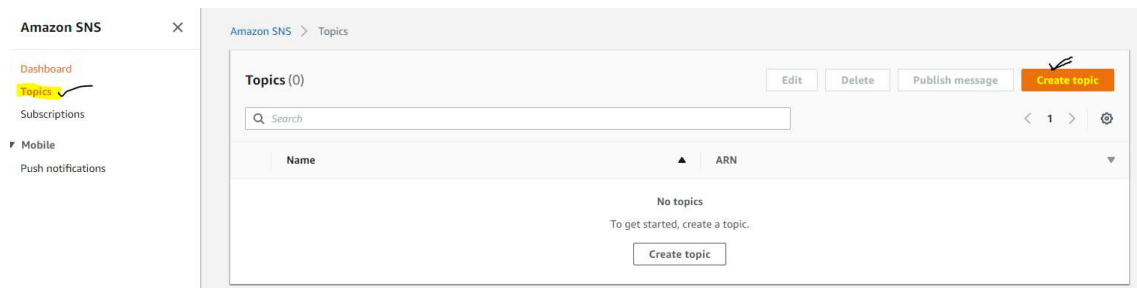
## 2. Setting Up Access for Amazon SNS

- To use Amazon SNS, you (user) must have permission. Need to attach the policy as **AmazonSNSFullAccess** to the user.

## 3. How to manage topics, subscriptions, and messages

### Step 1: Create a Topic

1. Sign in to the Amazon SNS console.
2. In the **Create topic** section, enter a **Topic name**, for example *SNS-Demo*.



## Create topic

### Details

Name

SNS-Demo

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_).

Display name - *optional*

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. [Info](#)

SNS-Demo

Maximum 100 characters, including hyphens (-) and underscores (\_).

► Encryption - *optional*

Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

► Access policy - *optional*

This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. [Info](#)

► Delivery retry policy (HTTP/S) - *optional*

The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. [Info](#)

► Tags - *optional*

A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

Cancel

Create topic

### 3. Choose **Create topic**.

The topic is created and the *SNS-Demo* page is displayed.

Topic SNS-Demo created successfully.

You can create subscriptions and send messages to them from this topic.

Publish message

Amazon SNS > Topics > SNS-Demo

SNS-Demo

Edit Delete Publish message

#### Details

Name

SNS-Demo

ARN

arn:aws:sns:ap-south-1:399808215315:SNS-Demo

Display name

SNS-Demo

Topic owner

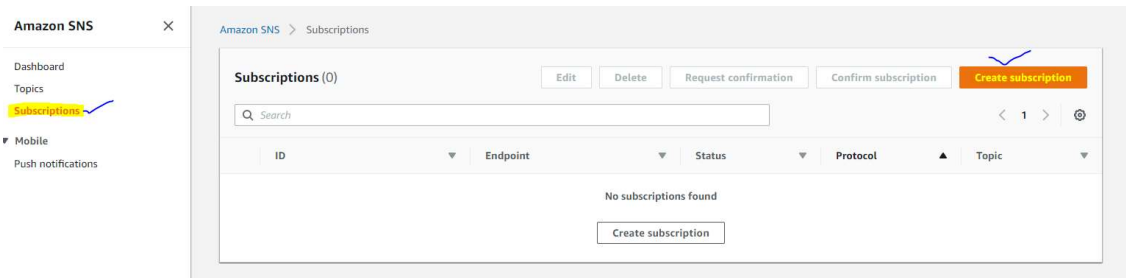
399808215315

### 4. Copy the topic ARN to the clipboard, for example:

```
arn:aws:sns:ap-south-1:399808215315:SNS-Demo
```

## Step 2: Create a Subscription for an Endpoint to the Topic

- On the navigation panel, choose **Subscriptions**.
- On the **Subscriptions** page, choose **Create subscription**.



3. On the **Create subscription** page, do the following:

a. Enter the **Topic ARN** of the topic you created earlier, for example:

arn:aws:sns:ap-south-1:399808215315:SNS-Demo

b. For **Protocol**, choose an endpoint type, for example **Email**.

c. For **Endpoint**, enter an email address that can receive notifications, for example:

name@example.com

**Note:** After your subscription is created, you must confirm it. Only HTTP/S endpoints, email addresses, and AWS resources in other AWS accounts require confirmation. (Amazon SQS queues and Lambda functions in the same AWS account—as well as mobile endpoints — don't require confirmation.)

d. Choose **Create subscription**.

Subscription to SNS-Demo created successfully.  
The ARN of the subscription is arn:aws:sns:ap-south-1:399808215315:SNS-Demo:c80f4311-f982-456d-95d5-ba4348dc0ff8.

Amazon SNS > Topics > SNS-Demo > Subscription: c80f4311-f982-456d-95d5-ba4348dc0ff8

Subscription: c80f4311-f982-456d-95d5-ba4348dc0ff8

Edit Delete

**Details**

ARN arn:aws:sns:ap-south-1:399808215315:SNS-Demo:c80f4311-f982-456d-95d5-ba4348dc0ff8	Status <b>Pending confirmation</b>
Endpoint [redacted]@gmail.com	Protocol EMAIL
Topic SNS-Demo	

2. In your email client, check the email address that you specified and choose **Confirm subscription** in the email from Amazon SNS.

1-100 of 514

Primary Social Promotions 4 new SBI Card, Udeny Forums

SNS-Demo **AWS Notification - Subscription Confirmation** You have chosen to subscribe to the topic: arn:aws:sns:ap-south-1:399808215315:SNS-D... 4:28 PM

AWS Notification - Subscription Confirmation

SNS-Demo no-reply@sns.amazonaws.com via amazonses.com to me

You have chosen to subscribe to the topic:  
arn:aws:sns:ap-south-1:399808215315:SNS-Demo

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):  
**Confirm subscription**

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

<https://sns.ap-south-1.amazonaws.com/confirmation.html?TopicArn=arn:aws:sns:ap-south-1:399808215315:SNS-Demo:c80f4311-f982-456d-95d5-ba4348dc0ff8>

amazon web services Simple Notification Service

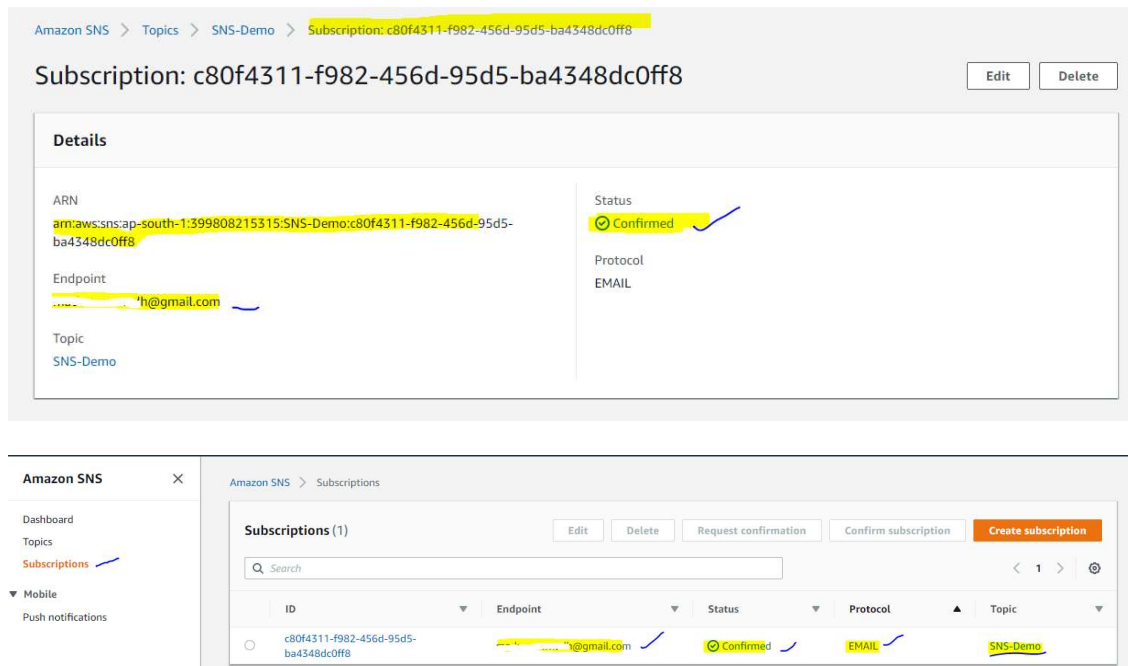
**Subscription confirmed!**

You have subscribed [redacted] to the topic:  
**SNS-Demo.**

Your subscription's id is:  
arn:aws:sns:ap-south-1:399808215315:SNS-Demo:c80f4311-f982-456d-95d5-ba4348dc0ff8

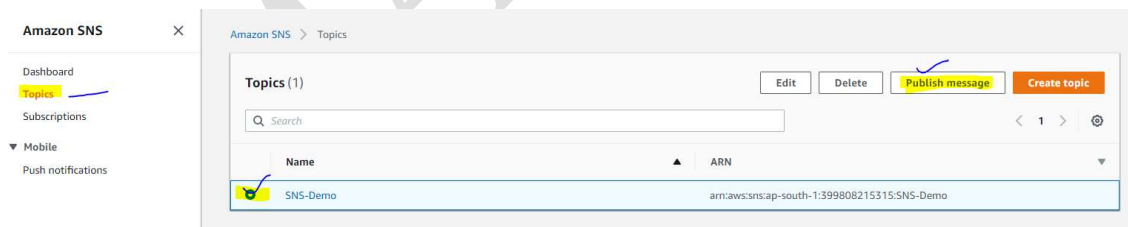
If it was not your intention to subscribe, [click here to unsubscribe](#).

3. In your web browser, a subscription confirmation with your subscription ID is displayed.



### Step 3: Publish a Message to the Topic

1. On the navigation panel, choose **Topics**.
2. On the **Topics** page, choose the topic you created earlier and then choose **Publish message**.



3. On the **Publish message to topic** page, do the following:
  - a. (Optional) In the **Message details** section, enter the **Subject**, for example:  
Hello from Amazon SNS!

Amazon SNS > Topics > SNS-Demo > Publish message

## Publish message to topic

### Message details

Topic ARN  
arn:aws:sns:ap-south-1:399808215315:SNS-Demo

Subject - *optional*  
  
Maximum 100 printable ASCII characters

Time to Live (TTL) - *optional*  
This setting applies only to mobile application endpoints. The number of seconds that the push notification service has to deliver the message to the endpoint. [Info](#)

b. In the **Message body** section, do one of the following:

- Choose **Identical payload for all delivery protocols** and then enter the message, for example:

If you receive this message, publishing a message to an Amazon SNS topic works

### Message body

Message structure

☒ **Identical payload for all delivery protocols.**  
The same payload is sent to endpoints subscribed to the topic, regardless of their delivery protocol.

☐ **Custom payload for each delivery protocol.**  
Different payloads are sent to endpoints subscribed to the topic, based on their delivery protocol.

Message body to send to the endpoint

```
1 If you receive this message, publishing a message to an Amazon SNS topic works.
```

- Choose **Custom payload for each delivery protocol** and then use a JSON object to define the message to send to each protocol, for example:

```
{
  "default": "Sample fallback message",
  "email": "Sample message for email endpoints",
  "sqs": "Sample message for Amazon SQS endpoints",
  "lambda": "Sample message for AWS Lambda endpoints",
}
```



```

"http": "Sample message for HTTP endpoints",
"https": "Sample message for HTTPS endpoints",
"sms": "Sample message for SMS endpoints",
"APNS": "{\"aps\":{\"alert\": \"Sample message for iOS
endpoints\"} }",
"APNS_SANDBOX": "{\"aps\":{\"alert\": \"Sample message for iOS
development endpoints\"} }",
"APNS_VOIP": "{\"aps\":{\"alert\": \"Sample message for Apple VoIP
endpoints\"} }",
"APNS_VOIP_SANDBOX": "{\"aps\":{\"alert\": \"Sample message for
Apple VoIP development endpoints\"} }",
"MACOS": "{\"aps\":{\"alert\": \"Sample message for MacOS
endpoints\"} }",
"MACOS_SANDBOX": "{\"aps\":{\"alert\": \"Sample message for MacOS
development endpoints\"} }",
"GCM": "{ \"data\": { \"message\": \"Sample message for Android
endpoints\" } }",
"ADM": "{ \"data\": { \"message\": \"Sample message for FireOS
endpoints\" } }",
"BAIDU": "{\"title\": \"Sample message
title\", \"description\": \"Sample message for Baidu endpoints\"}",
"MPNS": "<?xml version=\"1.0\" encoding=\"utf-
8\"?><wp:Notification
xmlns:wp=\"WPNotification\"><wp:Tile><wp:Count>ENTER
COUNT</wp:Count><wp:Title>Sample message for Windows Phone 7+
endpoints</wp:Title></wp:Tile></wp:Notification>",
"WNS": "<badge version=\"1\" value=\"42\"/>"
}

```

## Message body

### Message structure

- ☐ Identical payload for all delivery protocols.  
The same payload is sent to endpoints subscribed to the topic, regardless of their delivery protocol.

- ☒ Custom payload for each delivery protocol.  
Different payloads are sent to endpoints subscribed to the topic, based on their delivery protocol.

### Message body to send to the endpoint

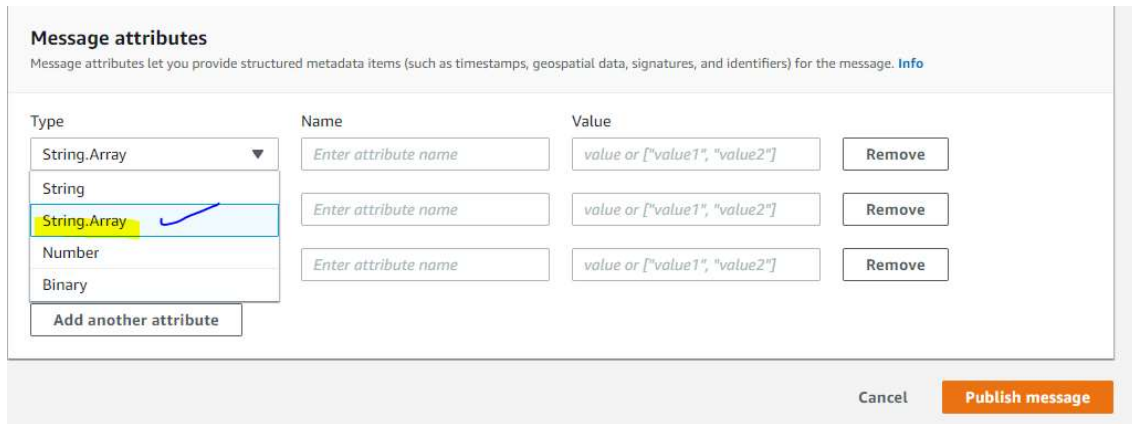
```

1 {
2   "default": "Sample fallback message",
3   "email": "Sample message for email endpoints",
4   "sqs": "Sample message for Amazon SQS endpoints",
5   "lambda": "Sample message for AWS Lambda endpoints",
6   "http": "Sample message for HTTP endpoints",
7   "https": "Sample message for HTTPS endpoints",
8   "sms": "Sample message for SMS endpoints",
9   "APNS": "{\"aps\":{\"alert\": \"Sample message for iOS endpoints\"} }",
10  "APNS_SANDBOX": "{\"aps\":{\"alert\": \"Sample message for iOS development
endpoints\"} }",
11  "APNS_VOIP": "{\"aps\":{\"alert\": \"Sample message for Apple VoIP endpoints\"} }",
12  "APNS_VOIP_SANDBOX": "{\"aps\":{\"alert\": \"Sample message for Apple VoIP
development endpoints\"} }",
13  "MACOS": "{\"aps\":{\"alert\": \"Sample message for MacOS endpoints\"} }",
14  "MACOS_SANDBOX": "{\"aps\":{\"alert\": \"Sample message for MacOS development
endpoints\"} }"

```

The message body must be a JSON object with an attribute for each delivery protocol. [Info](#)

- c. In the **Message attributes** section, add any attributes that you want Amazon SNS to match with the subscription attribute FilterPolicy to decide whether the subscribed endpoint is interested in the published message.



**Message attributes**  
Message attributes let you provide structured metadata items (such as timestamps, geospatial data, signatures, and identifiers) for the message. [Info](#)

Type	Name	Value	
String.Array	Enter attribute name	value or ["value1", "value2"]	Remove
String	Enter attribute name	value or ["value1", "value2"]	Remove
String.Array	Enter attribute name	value or ["value1", "value2"]	Remove
Number	Enter attribute name	value or ["value1", "value2"]	Remove
Binary	Enter attribute name	value or ["value1", "value2"]	Remove

[Add another attribute](#)

Cancel [Publish message](#)

- i. Select an attribute **Type**, for example **String.Array**.

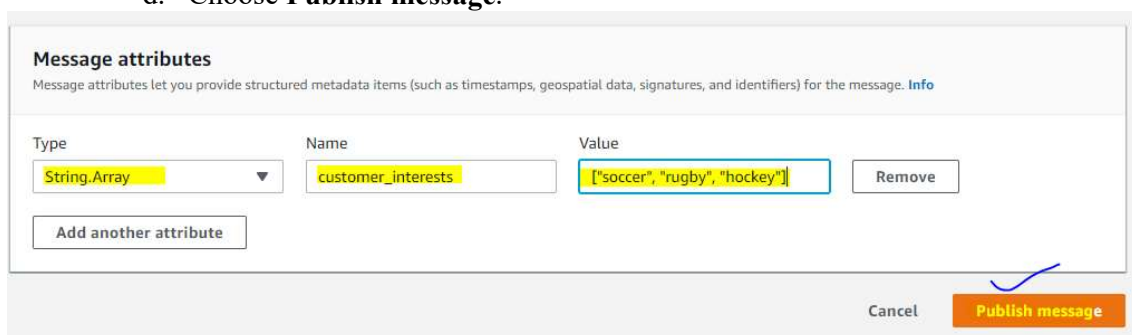
#### Note

If the attribute type is **String.Array**, enclose the array in square brackets ([]). Within the array, enclose string values in double quotation marks. You don't need quotation marks for numbers or for the keywords true, false, and null.

- ii. Enter a **Name** for the attribute, for example customer\_interests.  
iii. Enter a **Value** for the attribute, for example ["soccer", "rugby", "hockey"].

If the attribute type is **String**, **String.Array**, or **Number**, Amazon SNS evaluates the message attribute against a subscription's filter policy (if present) before sending the message to the subscription.

- d. Choose **Publish message**.



**Message attributes**  
Message attributes let you provide structured metadata items (such as timestamps, geospatial data, signatures, and identifiers) for the message. [Info](#)

Type	Name	Value	
String.Array	customer_interests	["soccer", "rugby", "hockey"]	Remove

[Add another attribute](#)

Cancel [Publish message](#)

The message is published to the topic and the **Demo-SNS** page is displayed.

4. In your email client, check the email address that you specified earlier and read the email from Amazon SNS.

The top screenshot shows an email client interface. The email is from "SNS-Demo" with the subject "Hello from Amazon SNS!". The body of the email contains a message about publishing to an Amazon SNS topic and a link to unsubscribe. The bottom screenshot shows the AWS Management Console page for the "SNS-Demo" topic. The "Subscriptions" tab is selected, showing a table with one subscription. The subscription ID is "c80f4311-f982-456d-95d5-ba4348dc0ff8", the endpoint is "mailto:amamadh@gmail.com", the status is "Confirmed", and the protocol is "EMAIL".

Amazon SNS > Topics > SNS-Demo

**SNS-Demo** Edit Delete Publish message

**Details**

Name	SNS-Demo	Display name	SNS-Demo
ARN	arn:aws:sns:ap-south-1:399808215315:SNS-Demo	Topic owner	amamadh@gmail.com

**Subscriptions** Access policy Delivery retry policy (HTTP/S) Delivery status logging Encryption Tags

**Subscriptions (1)** Edit Delete Request confirmation Confirm subscription Create subscription

Search

ID	Endpoint	Status	Protocol
c80f4311-f982-456d-95d5-ba4348dc0ff8	mailto:amamadh@gmail.com	Confirmed	EMAIL

## Step 4: Delete the Subscription and Topic

1. On the navigation panel, choose **Subscriptions**.
2. On the **Subscriptions** page, choose a *confirmed* subscription and then choose **Delete**.

The screenshot shows the AWS Management Console page for "Subscriptions". The left navigation panel has "Subscriptions" highlighted. The main content area shows the "Subscriptions (1)" page. The "Delete" button is highlighted with a blue checkmark. The table below shows one subscription with the status "Confirmed".

Amazon SNS Subscriptions

**Subscriptions (1)** Edit Delete Request confirmation Confirm subscription Create subscription

Search

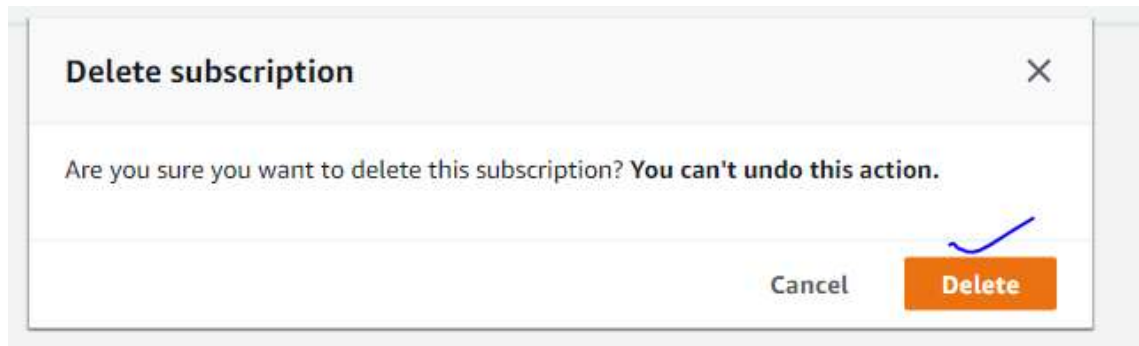
ID	Endpoint	Status	Protocol	Topic
c80f4311-f982-456d-95d5-ba4348dc0ff8	mailto:amamadh@gmail.com	Confirmed	EMAIL	SNS-Demo

## Note

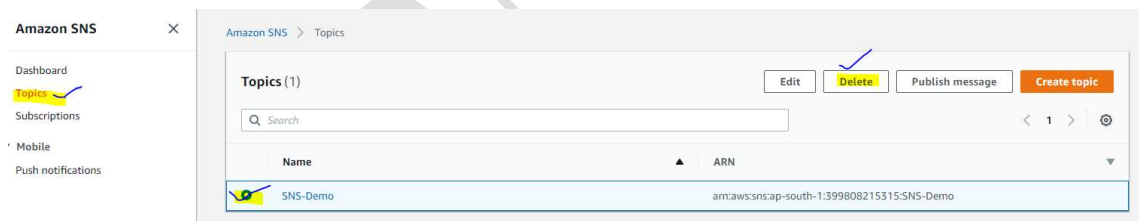
You can't delete a pending confirmation. **After 3 days**, Amazon SNS deletes it automatically.

3. In the **Delete subscription** dialog box, choose **Delete**.

The subscription is deleted.



4. On the navigation panel, choose **Topics**.
5. On the **Topics** page, choose a topic and then choose **Delete**.

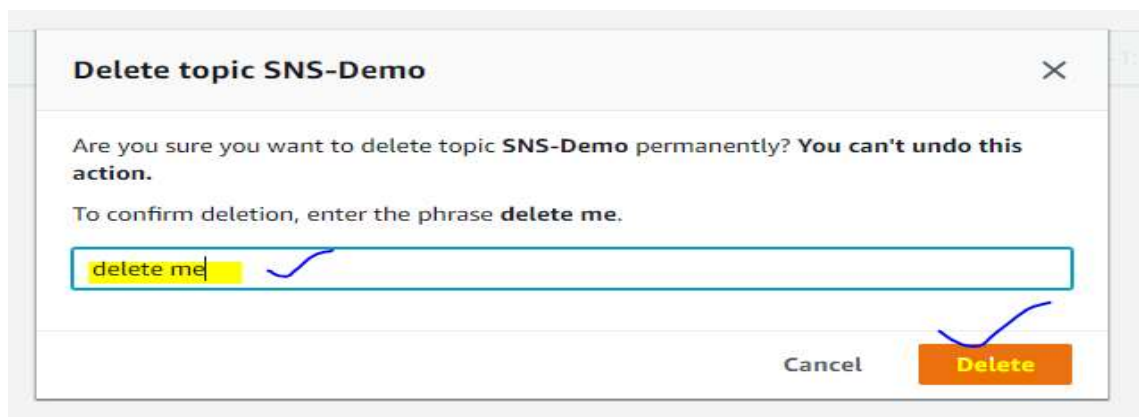


## Important

When you delete a topic, you also delete all subscriptions to the topic.

6. On the **Delete topic SNS-Demo** dialog box, enter delete me and then choose **Delete**.

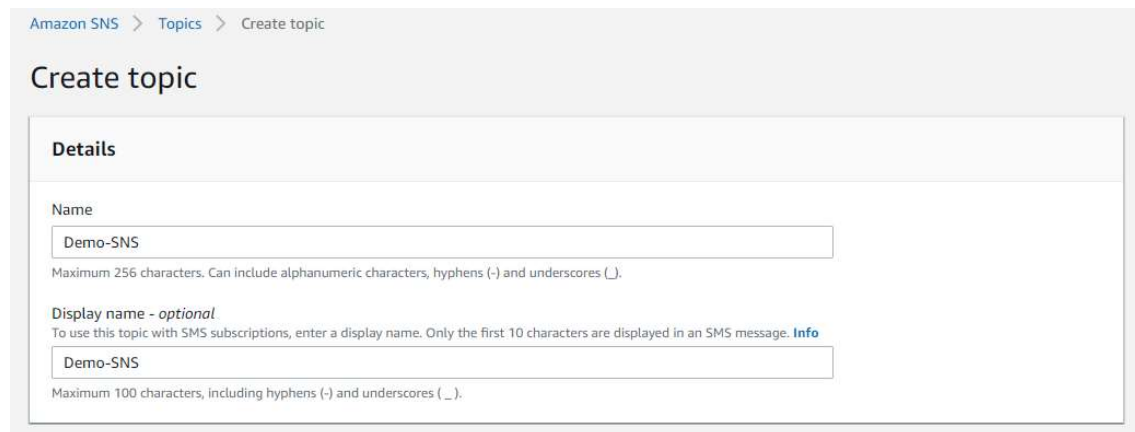
The topic is deleted.



## 4. Creating an Amazon SNS Topic

➤ An Amazon SNS topic is a logical access point which acts as a *communication channel*. A topic lets you group multiple *endpoints*(such as AWS Lambda, Amazon SQS, HTTP/S, or an email address).

1. Sign in to the [Amazon SNS console](#).
2. Do one of the following:
  - If no topics have ever been created under your AWS account before, read the description of Amazon SNS on the home page.
  - If topics have been created under your AWS account before, on the navigation panel, choose **Topics**.
3. In the **Create topic** section, enter a **Topic name**, for example *Demo-SNS*.



4. (Optional) Expand the **Encryption** section and do the following.
  - a. Choose **Enable encryption**.
  - b. Specify the customer master key (CMK).

For each CMK type, the **Description**, **Account**, and **CMK ARN** are displayed.

### Important

If you aren't the owner of the CMK, or if you log in with an account that doesn't have the `kms:ListAliases` and `kms:DescribeKey` permissions, you won't be able to view information about the CMK on the Amazon SNS console.

- The AWS managed CMK for Amazon SNS (**Default**) `alias/aws/sns` is selected by default.

## Note

Keep the following in mind:

- The first time you use the AWS Management Console to specify the AWS managed CMK for Amazon SNS for a topic, AWS KMS creates the AWS managed CMK for Amazon SNS.
- Alternatively, the first time you use the Publish action on a topic with SSE enabled, AWS KMS creates the AWS managed CMK for Amazon SNS.
- To use a custom CMK from your AWS account, choose the **Customer master key (CMK)** field and then choose the custom CMK from the list.

**Note:** For instructions on creating custom CMKs, see [Creating Keys](#) in the *AWS Key Management Service Developer Guide*

- To use a custom CMK ARN from your AWS account or from another AWS account, enter it into the **Customer master key (CMK)** field.

The screenshot shows the 'Encryption - optional' section in the AWS Management Console. It includes a sub-section 'Encryption' with two radio buttons: 'Enable encryption' (selected) and 'Disable encryption'. Below this is the 'Customer master key (CMK)' section, which has a text input field containing '(Default) alias/aws/sns'. A description below the field states: 'Default master key that protects my SNS data when no other key is defined'. At the bottom, the 'CMK ARN' is displayed as 'arn:aws:kms:ap-south-7:345678901234:key/ddbe80f4-5bd0-4b3d-8493-010e0fe59a70'.

5. (Optional) To configure access permissions for your topic, expand the **Access policy** section.

The screenshot shows the 'Access policy - optional' section in the AWS Management Console. It has two tabs: 'Basic' (selected) and 'Advanced'. Under the 'Basic' tab, there are two sections: 'Define who can publish messages to the topic' and 'Define who can subscribe to this topic'. Each section has three radio button options: 'Only the topic owner', 'Everyone', and 'Only the specified AWS accounts'. The 'Only the specified AWS accounts' option is selected in both sections. To the right, there is a 'JSON preview' section showing a JSON policy document. The document includes a 'Version' of '2008-10-17', an 'Id' of '\_\_\_default\_policy\_ID', and a 'Statement' with a single entry that allows 'Publish', 'RemovePermission', 'SetTopicAttributes', 'DeleteTopic', and 'ListSubscriptionsByTopic' actions for the principal 'AWS:\*'.

6. (Optional) To configure how Amazon SNS retries failed message delivery attempts, expand the **Delivery retry policy (HTTP/S)** section.

**▼ Delivery retry policy (HTTP/S) - optional**  
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. [Info](#)

☒ Use the default delivery retry policy

Number of retries  
  
Must be an integer from 0 to 100

Retries without delay  
  
Must be lower than the total number of retries

Minimum delay  
 seconds  
Must be lower than maximum delay

Maximum delay  
 seconds  
Must be higher than minimum delay and lower than 3,600

Minimum delay retries  
  
Must be lower than number of retries

Maximum delay retries  
  
Must be lower than number of retries

Maximum receive rate  
 per second  
Must be 1 or greater

Retry-backoff function  
The rate at which the delay increases from minimum to maximum.

☐ Override subscription policy  
Apply this policy to all subscriptions, even if they have their own policies.

JSON preview

```
{
  "http": {
    "defaultHealthyRetryPolicy": {
      "numRetries": 2,
      "numNoDelayRetries": 0,
      "minDelayTarget": 20,
      "maxDelayTarget": 20,
      "numMinDelayRetries": 0,
      "numMaxDelayRetries": 0,
      "backoffFunction": "linear"
    },
    "disableSubscriptionOverrides": false
  }
}
```

7. (Optional) To configure how Amazon SNS logs the delivery of messages to CloudWatch, expand the **Delivery status logging** section.

**▼ Delivery status logging - optional**  
These settings configure the logging of message delivery status to CloudWatch Logs. [Info](#)

Log delivery status for these protocols

☐ AWS Lambda  
☐ Amazon SQS  
☐ HTTP/S  
☐ Platform application endpoint

Success sample rate  
The percentage of successful message deliveries to log.  
 %

IAM roles  
Amazon SNS requires permission to write logs to CloudWatch Logs. You can use separate roles for successful and failed message deliveries.

Service role [Info](#)

☒ Use existing service role  
Choose an existing service role from your account

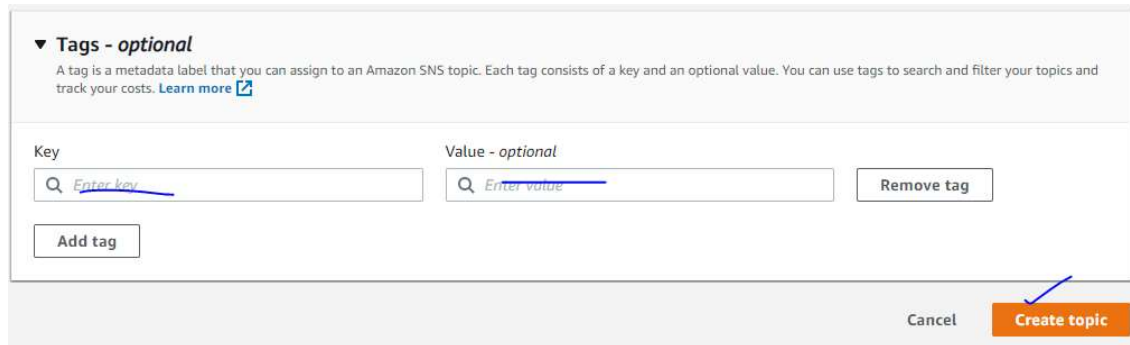
☐ Create new service role  
Create a new service role in your account

IAM role for successful deliveries

IAM role for failed deliveries



8. (Optional) To add metadata tags to the topic, expand the **Tags** section, enter a **Key** and a **Value** (optional) and choose **Add tag**.



9. Choose **Create topic**.

The topic is created and the **Demo-SNS** page is displayed.

10. Copy the topic ARN to the clipboard, for example:

arn:aws:sns:ap-south-1:399808215315:SNS-Demo

## 5. Protecting Amazon SNS Data Using Server-Side Encryption (SSE) and AWS KMS

- Server Side Encryption (SSE) protects the contents of messages in Amazon SNS topics using keys managed in AWS Key Management Service (AWS KMS).
- The messages are stored in encrypted form and Amazon SNS decrypts messages only when they are sent.
- All requests to topics with SSE enabled must use HTTPS and [Signature Version 4](#).
- **Signature Version 4:**
  - It is the process to add authentication information to AWS requests sent by HTTP.
  - How Signature Version 4 works by creating a **canonical request**.
- **AWS KMS** combines secure, highly available hardware and software to provide a key management system scaled for the cloud.
- When you use Amazon SNS with AWS KMS, the data keys that encrypt your message data are also encrypted and stored with the data they protect.
- The following are benefits of using AWS KMS:
  - You can create and manage **customer master keys (CMKs)** yourself.



- You can also use the AWS managed CMK for Amazon SNS, which is unique for each account and region.
  - The AWS KMS security standards can help you meet encryption-related compliance requirements.
- SSE encrypts the body of a message in an Amazon SNS topic.
- SSE doesn't encrypt the following:
- Topic metadata (topic name and attributes)
  - Message metadata (subject, message ID, timestamp, and attributes)
  - Per-topic metrics
- A message is encrypted only if it is sent after the encryption of a topic is enabled. Amazon SNS doesn't encrypt backlogged messages.
- Any encrypted message remains encrypted even if the encryption of its topic is disabled.

## 5.1 Configuring AWS KMS Permissions

- Before you can use SSE, you must configure AWS KMS key policies to allow encryption of topics and encryption and decryption of messages.

### 5.1.1 Allow a User to Send Messages to a Topic with SSE

The publisher must have the `kms:GenerateDataKey` and `kms:Decrypt` permissions for the customer master key (CMK).

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:default-region-us-east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

### 5.1.2 Enable Compatibility between Event Sources from AWS Services and Encrypted Topics

Several AWS services publish events to Amazon SNS topics. To allow these event sources to work with encrypted topics, you must perform the following steps.

1. Use the AWS managed CMK for Amazon SNS.
2. To allow the AWS service to have the `kms:GenerateDataKey*` and `kms:Decrypt` permissions, add the following statement to the CMK policy.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

NOTE: Here Service will be very based on the event source (ex: Cloudwatch, S3 Glacier, DynamoDB & etc)

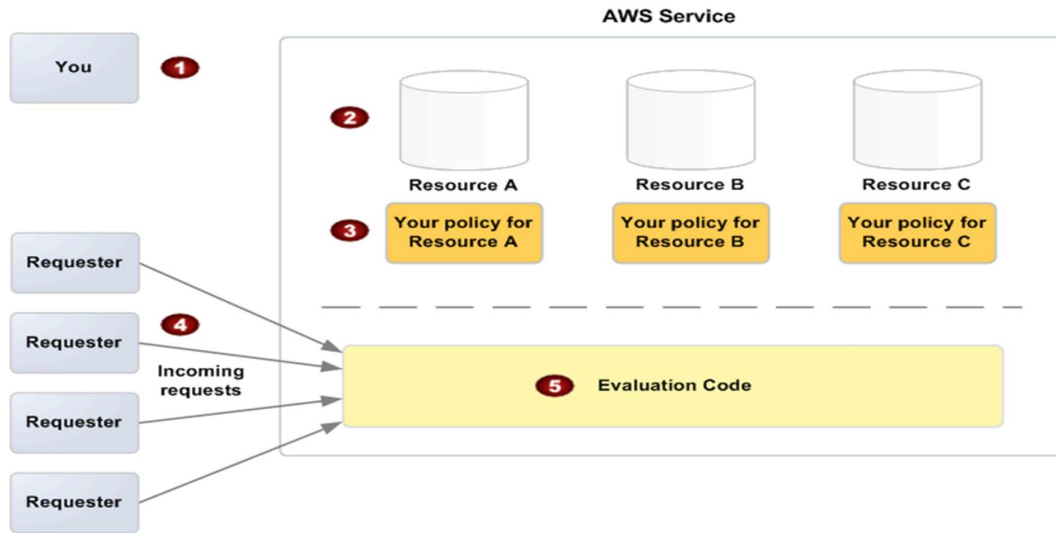
3. **Enable SSE for your topic** using your CMK.
4. Provide the ARN of the encrypted topic to the event source.

## 6. Access Control

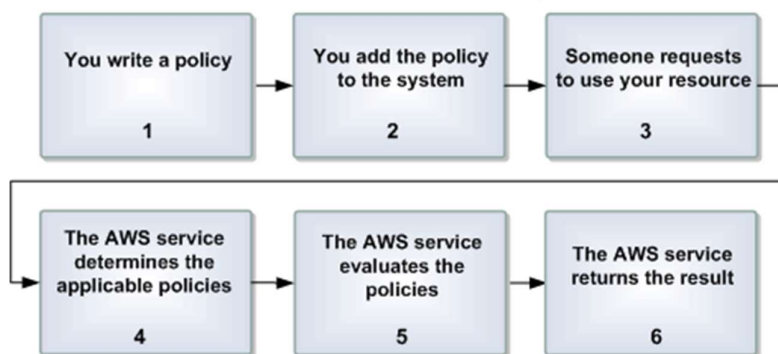
- Amazon SNS supports other protocols beside email. You can use HTTP, HTTPS, and Amazon SQS queues.
- You have detailed control over which endpoints a topic allows, who is able to publish to a topic, and under what conditions.
- Use case of Access Control:
  - You want to grant another AWS account a particular type of topic action (for example, Publish)
  - You want to limit subscriptions to your topic to only the HTTPS protocol.
  - You want to allow Amazon SNS to publish messages to your Amazon SQS queue.
- Key Concepts of Access Control:

- **Permission:** A *permission* is the concept of allowing or disallowing some kind of access to a particular resource.
- **Statement:** A *statement* is the formal description of a single permission, written in the access policy language.
- **Policy:** A *policy* is a document that acts as a container for one or more statements.
- **Issuer:** The *issuer* is the person who writes a policy to grant permissions for a resource. The issuer (by definition) is always the resource owner. AWS does not permit AWS service users to create policies for resources they don't own.
- **Principal:** The *principal* is the person or persons who receive the permission in the policy.
- **Action:** The *action* is the activity the principal has permission to perform. (Ex: Action=Subscribe)
- **Resource:** The *resource* is the object the principal is requesting access to.
- **Conditions and Keys:** The *conditions* are any restrictions or details about the permission. A *key* is the specific characteristic that is the basis for access restriction.
- **Requester:** The *requester* is the person who sends a request to an AWS service and asks for access to a particular resource.
- **Evaluation:** *Evaluation* is the process the AWS service uses to determine if an incoming request should be denied or allowed based on the applicable policies
- **Effect:** The *effect* is the result that you want a policy statement to return at evaluation time.
- **Default Deny:** A *default deny* is the default result from a policy in the absence of an allow or explicit deny
- **Allow:** An *allow* results from a statement that has effect=allow, assuming any stated conditions are met.
- **Explicit Deny:** An *explicit deny* results from a statement that has effect=deny, assuming any stated conditions are met.

➤ Architectural Overview:



➤ Using the Access Policy Language:



1. You write a policy to specify permissions for your Amazon SNS topics.
2. You use the Amazon `SNSSetTopicAttributes` action to upload a policy for a particular Amazon SNS topic.
3. A user sends a request to Amazon SNS to use one of your topics.
4. Amazon SNS looks at all the available Amazon SNS policies and determines which ones are applicable
5. Amazon SNS evaluates the policies and determines if the requester is allowed to use your topic or not.
6. Based on the policy evaluation result, the service either returns an "Access denied" error to the requester or continues to process the request.

## Amazon SNS Keys

- **sns:Endpoint**—The URL, email address, or ARN from a Subscribe request or a previously confirmed subscription. (for example, `*@example.com`).
- **sns:Protocol**—The protocol value from a Subscribe request or a previously confirmed subscription. Use with string conditions (for example, `https`).

## 7. Example Cases for Amazon SNS Access Control

### 7.1. Grant AWS Account Access to a Topic

- You want to allow one or more AWS accounts access to a specific topic action (for example, Publish).
- Ex: if you called `AddPermission` on the topic `arn:aws:sns:us-east-2:444455556666:MyTopic`, with AWS account ID `1111-2222-3333`, the Publish action, and the label `give-1234-publish`,

```
{
  "Version": "2012-10-17",
  "Id": "AWSAccountTopicAccess",
  "Statement": [{
    "Sid": "give-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

- Once this statement is added, the user with AWS account `1111-2222-3333` can publish messages to the topic.

### 7.2. Limit Subscriptions to HTTPS

- You limit the notification delivery protocol to HTTPS.
- Example of a full policy gives the AWS account ID `1111-2222-3333` the ability to subscribe to notifications from a topic.

```
{
  "Version": "2012-10-17",
  "Id": "SomePolicyId",
  "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"
      }
    }
  }
}

```

### 7.3. Publish Messages to an Amazon SQS Queue

- You want to publish messages from your topic to your Amazon SQS queue.

```

{
  "Version": "2012-10-17",
  "Id": "MyQueuePolicy",
  "Statement": [{
    "Sid": "Allow-SNS-SendMessage",
    "Effect": "Allow",
    "Principal": "*",
    "Action": ["sqs:SendMessage"],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:MyQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:444455556666:MyTopic"
      }
    }
  }]
}

```

- This policy uses the `aws:SourceArn` condition to restrict access to the queue based on the source of the message being sent to the queue.
- You can use this type of policy to allow Amazon SNS to send messages to your queue only if the messages are coming from one of your own topics.

### 7.4. Allow Any AWS Resource to Publish to a Topic

- You want to configure a topic's policy so that another AWS account's resource (for example, Amazon S3 bucket, Amazon EC2 instance, or Amazon SQS queue) can publish to your topic.
- Example statement, the topic owner in these policies is 1111-2222-3333 and the AWS resource owner is 4444-5555-6666. The example gives the AWS account ID 4444-5555-6666 the ability to publish to My-Topic from any AWS resource owned by the account.

```
{
  "Version": "2012-10-17",
  "Id": "MyAWSPolicy",
  "Statement": [{
    "Sid": "My-statement-id",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      }
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

- If you publish messages directly (rather than having an AWS resource publish messages on your behalf), a policy in which you specify an empty Principal *and* use AWS:SourceAccount as a condition will not work.

## 7.5. Allow an Amazon S3 Bucket to Publish to a Topic

- You want to configure a topic's policy so that another AWS account's Amazon S3 bucket can publish to your topic.

```
{
  "Version": "2012-10-17",
  "Id": "MyAWSPolicy",
  "Statement": [{
    "Sid": "My-statement-id",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      },
      "ArnLike": {
        "AWS:SourceArn": "arn:aws:s3:*:*:*"
      }
    }
  ]
}
```

- Example statement uses the ArnLike condition to make sure the ARN of the resource making the request (the AWS:SourceARN) is an Amazon S3 ARN.

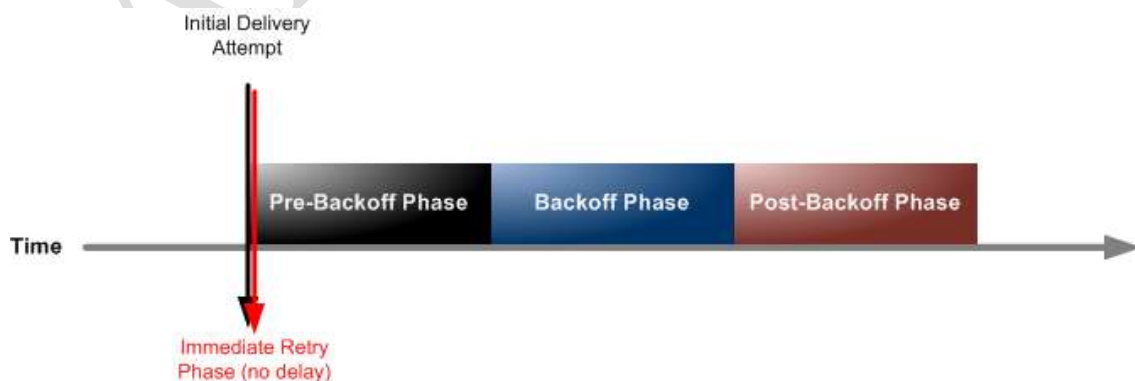
## 7.6. Allow a CloudWatch Alarm in an AWS Account to Publish to an Amazon SNS Topic in a Different AWS Account

- Ex: the CloudWatch alarm in account 111122223333 is allowed to publish to an Amazon SNS topic in account 444455556666.\

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:cloudwatch:us-east-2:111122223333:alarm:MyAlarm"
      }
    }
  }]
}
```

## 8. Setting Amazon SNS Delivery Retry Policies for HTTP/HTTPS Endpoints

- A successful Amazon SNS delivery to an HTTP/HTTPS endpoint sometimes requires more than one attempt.
- For example, if the web server that hosts the subscribed endpoint is down for maintenance or is experiencing heavy traffic. If an initial delivery attempt doesn't result in a successful response from the subscriber, Amazon SNS attempts to deliver the message again. We call such an attempt a *retry*.
- You can use delivery policies to control not only the total number of retries, but also the time delay between each retry. You can specify up to 100 total retries distributed among **four discrete phases**. The maximum lifetime of a message in the system is one hour. This one hour limit cannot be extended by a delivery policy.





1. **Immediate Retry Phase**—Also called the *no delay phase*, this phase occurs immediately after the initial delivery attempt. The value you set for **Retries with no delay** determines the number of retries immediately after the initial delivery attempt. There is no delay between retries in this phase.
2. **Pre-Backoff Phase**—The pre-backoff phase follows the immediate retry phase. Use this phase to create a set of retries that occur before a backoff function applies to the retries. Use the **Minimum delay retries** setting to specify the number of retries in the Pre-Backoff Phase. You can control the time delay between retries in this phase using the **Minimum delay** setting.
3. **Backoff Phase**—This phase is called the backoff phase because you can control the delay between retries in this phase using the retry backoff function. Set the **Minimum delay** and the **Maximum delay**, and then choose a **Retry backoff function** to define how quickly the delay increases from the minimum delay to the maximum delay.
4. **Post-Backoff Phase**—The post-backoff phase follows the backoff phase. Use the **Maximum delay retries** setting to specify the number of retries in the post-backoff phase. You can control the time delay between retries in this phase using the **Maximum delay** setting.

### 8.1. Setting the Maximum Receive Rate

1. Sign in to the [Amazon SNS console](#).
2. On the navigation panel, choose **Topics** and then chose the name of a topic.
3. On the *MyTopic* page, choose the **Edit**.
4. On the **Edit MyTopic** page, expand the **Delivery retry policy (HTTP/S)** section and specify the **Maximum retry rate**.
5. Choose **Save changes**.

### 8.2. Immediate Retry Phase

1. Sign in to the [Amazon SNS console](#).
2. On the navigation panel, choose **Topics** and then choose a topic ARN.
3. In the **Topic Details** section, choose **Edit topic delivery policy** from the **Other topic actions** drop-down list.
4. In the **Retries with no delay** box, type an integer value.
5. Choose **Update policy** to save your changes.

### 8.3. Pre-Backoff Phase

1. Sign in to the [Amazon SNS console](#).
2. On the navigation panel, choose **Topics** and then choose a topic ARN.
3. In the **Topic Details** section, choose **Edit topic delivery policy** from the **Other topic actions** drop-down list.
4. In the **Minimum delay retries** box, type an integer value.
5. In the **Minimum delay** box, type an integer value to set the delay between messages in this phase.

The value you set must be less than or equal to the value you set for **Maximum delay**.

6. Choose **Update policy** to save your changes.

### 8.4. Backoff Phase

The backoff phase is the only phase that applies by default. You can control the number of retries in the backoff phase using **Number of retries**.

You can choose from four retry backoff functions.

- Linear
- Arithmetic
- Geometric
- Exponential

### 8.4. Post-Backoff Phase

1. Sign in to the [Amazon SNS console](#).
2. On the navigation panel, choose **Topics** and then choose a topic ARN.
3. In the **Topic Details** section, choose **Edit topic delivery policy** from the **Other topic actions** drop-down list.
4. In the **Maximum delay retries** box, type an integer value.
5. In the **Maximum delay** box, type an integer value to set the delay between messages in this phase.

The value you set must be greater than or equal to the value you set for **Minimum delay**.

6. Choose **Update policy** to save your changes.

## 9. Enabling Server-Side Encryption (SSE) for an Amazon SNS Topic with an Encrypted Amazon SQS Queue Subscribed

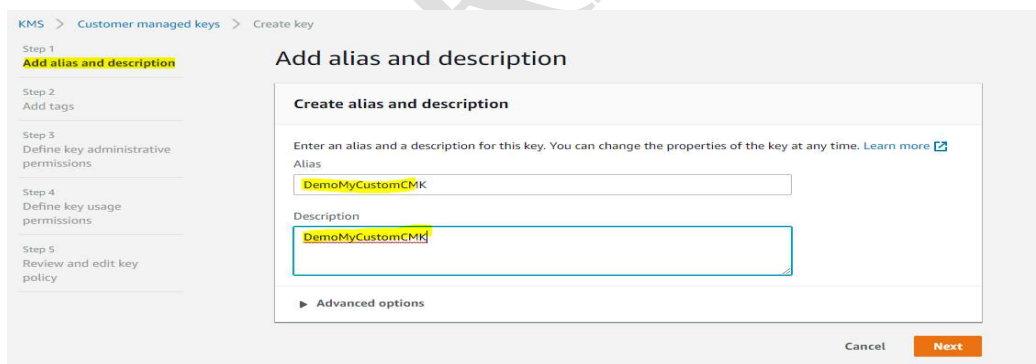
- To allow Amazon SNS to send messages to encrypted Amazon SQS queues, the customer master key (CMK) associated with the Amazon SQS queue must have a policy statement that grants Amazon SNS service-principal access to the AWS KMS API actions `GenerateDataKey` and `Decrypt`.

### Step 1: To Create a Custom CMK

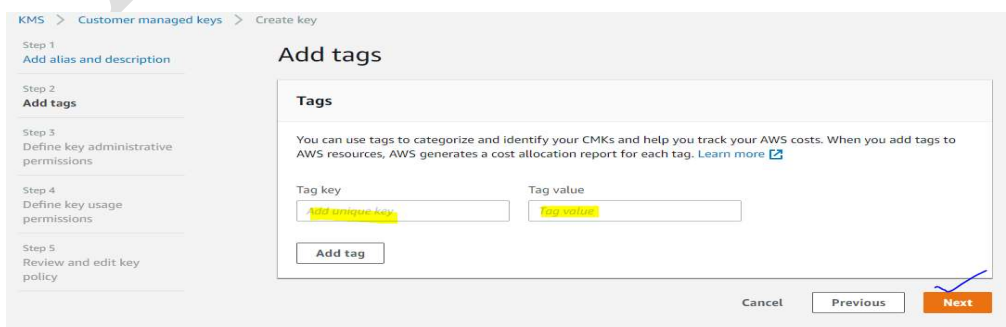
1. Sign in to the [AWS KMS console](#) with a user that has at least the `AWSKeyManagementServicePowerUser` policy.
2. Choose **Create a key**.



3. On the **Add alias and description** page, enter an **Alias** for your key (for example, `DemoMyCustomCMK`) and then choose **Next**.



4. On the **Add tags** page, choose **Next**.



5. On the **Define key administrative permissions** page, in the **Key administrators** section, choose an IAM role or an IAM user and then choose **Next**.

KMS > Customer managed keys > Create key

Step 1  
Add alias and description

Step 2  
Add tags

Step 3  
**Define key administrative permissions**

Step 4  
Define key usage permissions

Step 5  
Review and edit key policy

### Define key administrative permissions

**Key administrators**  
Choose the IAM users and roles who can administer this key through the KMS API. You may need to add additional permissions for the users or roles to administer this key from this console. [Learn more](#)

Q

<input type="checkbox"/>	Name	Path	Type
<input checked="" type="checkbox"/>	amar	/	User
<input type="checkbox"/>	Ganapathi	/	User
<input type="checkbox"/>	Allow_EC2_Instan..._role	/	Role
<input type="checkbox"/>	AWSServiceRoleForAutoScaling	/aws-service-role/autoscaling.amazonaws.com/	Role
<input type="checkbox"/>	AWSServiceRoleForElasticLoadBalancing	/aws-service-role/elasticloadbalancing.amazonaws.com/	Role
<input type="checkbox"/>	AWSServiceRoleForSupport	/aws-service-role/support.amazonaws.com/	Role
<input type="checkbox"/>	AWSServiceRoleForTrustedAdvisor	/aws-service-role/trustedadvisor.amazonaws.com/	Role
<input type="checkbox"/>	EC2_Instance_Stop_Start	/	Role

**Key deletion**

☒ Allow key administrators to delete this key.

Cancel Previous **Next**

6. On the **Define key usage permissions** page, in the **This account** section, choose an IAM role or an IAM user and then choose **Next**.

Step 1  
Add alias and description

Step 2  
Add tags

Step 3  
Define key administrative permissions

Step 4  
**Define key usage permissions**

Step 5  
Review and edit key policy

### Define key usage permissions

**This account**  
Select the IAM users and roles that can use the CMK to encrypt and decrypt data with the AWS KMS API. [Learn more](#)

Q

<input type="checkbox"/>	Name	Path	Type
<input checked="" type="checkbox"/>	amar	/	User
<input type="checkbox"/>	Ganapathi	/	User
<input type="checkbox"/>	Allow_EC2_Instan..._role	/	Role
<input type="checkbox"/>	AWSServiceRoleForAutoScaling	/aws-service-role/autoscaling.amazonaws.com/	Role
<input type="checkbox"/>	AWSServiceRoleForElasticLoadBalancing	/aws-service-role/elasticloadbalancing.amazonaws.com/	Role
<input type="checkbox"/>	AWSServiceRoleForSupport	/aws-service-role/support.amazonaws.com/	Role
<input type="checkbox"/>	AWSServiceRoleForTrustedAdvisor	/aws-service-role/trustedadvisor.amazonaws.com/	Role
<input type="checkbox"/>	EC2_Instance_Stop_Start	/	Role

**Other AWS accounts**  
Specify the AWS accounts that can use this key. Administrators of the accounts you specify are responsible for managing the permissions that allow their IAM users and roles to use this key. [Learn more](#)

arn:aws:iam:: Enter the ID of another AWS account root Remove

Add another AWS account

Cancel Previous **Next**

7. On the **Review and edit key policy** page, add the following statement to the key policy, and then choose **Finish**.

```
{  
  "Sid": "Allow Amazon SNS to use this key",  
  "Effect": "Allow",  
  "Principal": {  

```

```

        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}

```

```

{
    "Id": "key-consolepolicy-3",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::399808215315:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::399808215315:user/amar"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::399808215315:user/amar"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",

```

```

        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::399808215315:user/amar"
    },
    "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": "true"
        }
    }
},
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
]
}

```

KMS > Customer managed keys > Create key

Step 1  
[Add alias and description](#)

Step 2  
[Add tags](#)

Step 3  
[Define key administrative permissions](#)

Step 4  
[Define key usage permissions](#)

Step 5  
**[Review and edit key policy](#)**

### Review and edit key policy

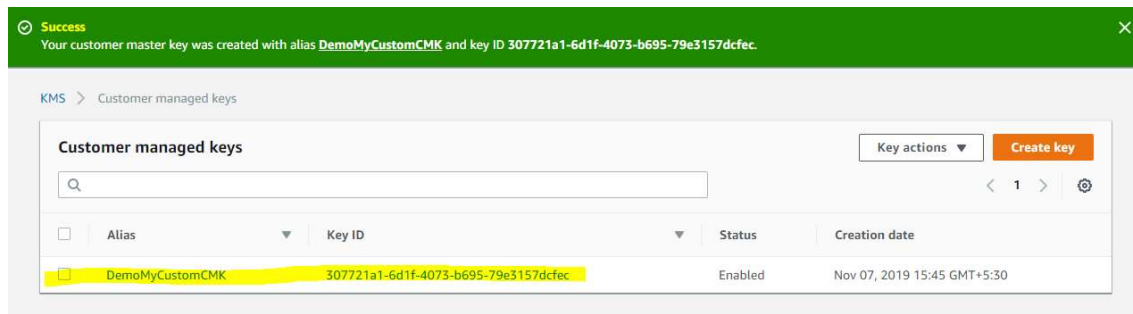
```

1 {
2   "Id": "key-consolepolicy-3",
3   "Version": "2012-10-17",
4   "Statement": [
5     {
6       "Sid": "Enable IAM User Permissions",
7       "Effect": "Allow",
8       "Principal": {
9         "AWS": "arn:aws:iam::399808215315:root"
10      },
11      "Action": "kms:*",
12      "Resource": "*"
13    },
14    {
15      "Sid": "Allow access for Key Administrators",

```

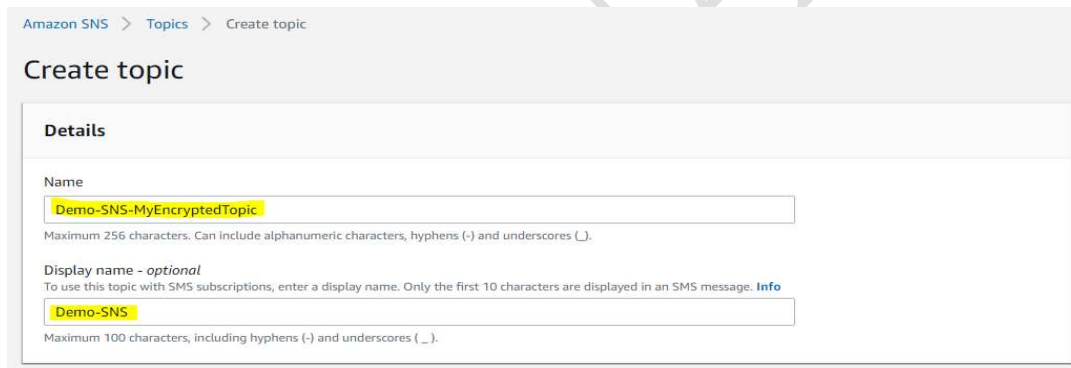
Cancel Previous **Finish**

- Your new custom CMK appears in the list of keys.



## Step 2: To Create an Encrypted Amazon SNS Topic

1. Sign in to the [Amazon SNS console](#).
2. On the navigation panel, choose **Topics**.
3. Choose **Create topic**.
4. On the **Create new topic** page, for **Name**, enter a topic name (for example, Demo-SNS-MyEncryptedTopic) and then choose **Create topic**.



5. Expand the **Encryption** section and do the following:
  - a. Choose **Enable server-side encryption**.
  - b. Specify the customer master key (CMK).  
For each CMK type, the **Description**, **Account**, and **CMK ARN** are displayed.

**Important:** If you aren't the owner of the CMK, or if you log in with an account that doesn't have the `kms:ListAliases` and `kms:DescribeKey` permissions, you won't be able to view information about the CMK on the Amazon SNS console.

- c. For **Customer master key (CMK)**, choose **DemoMyCustomCMK** [which you created earlier](#) and then choose **Enable server-side encryption**.

▼ **Encryption - optional**  
 Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Encryption

☒ **Enable encryption** [Learn more](#)

Enabling server side encryption adds at-rest encryption to your topic. Amazon SNS encrypts your message as soon as it is received. The message is decrypted immediately prior to delivery.

☐ Disable encryption

Customer master key (CMK)  
 Select a custom CMK or enter an existing CMK ARN.

Description  
 DemoMyCustomCMK

Account  
 399808215315

CMK ARN  
 arn:aws:kms:ap-south-1:399808215315:key/307721a1-6d1f-4073-b695-79e3157dcfec

6. Choose **Save changes**.  
 SSE is enabled for your topic and the **MyTopic** page is displayed.

➤ Your new encrypted topic appears in the list of topics.

Amazon SNS > Topics

Topics (1) Edit Delete Publish message Create topic

Name	ARN
<input checked="" type="radio"/> Demo-SNS-MyEncryptedTopic	arn:aws:sns:ap-south-1:399808215315:Demo-SNS-MyEncryptedTopic

### Step 3: To Create and Subscribe Encrypted Amazon SQS Queues

1. Sign in to the [Amazon SQS console](#).
2. Choose **Create New Queue**.



3. On the **Create New Queue** page, do the following:
  - a. Enter a **Queue Name** (for example, Demo-MyEncryptedQueue1).
  - b. Choose **Standard Queue**, and then choose **Configure Queue**.



Create New Queue

What do you want to name your queue?

Queue Name **1** Demo-MyEncryptedQueue1

Region **1** Asia Pacific (Mumbai)

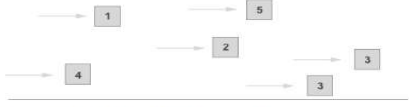
What type of queue do you need?

**Standard Queue**

Unlimited Throughput: Standard queues support a nearly unlimited number of transactions per second (TPS) per API action.

At-Least-Once Delivery: A message is delivered at least once, but occasionally more than one copy of a message is delivered.

Best-Effort Ordering: Occasionally, messages might be delivered in an order different from which they were sent.



Send data between applications when the throughput is important, for example:


- Decouple live user requests from intensive background work: let users upload media while realizing or encoding it.
- Allocate tasks to multiple worker nodes: process a high number of credit card validation requests.
- Batch messages for future processing: schedule multiple entries to be added to a database.

**FIFO Queue**

High Throughput: FIFO queues support up to 300 messages per second (300 send, receive, or delete operations per second). When you batch 10 messages per operation (maximum), FIFO queues can support up to 3,000 messages per second. To request a limit increase, file a support request.

First-In-First-out Delivery: The order in which messages are sent and received is strictly preserved.

Exactly-Once Processing: A message is delivered once and remains available until a consumer processes and deletes it. Duplicates are not introduced into the queue.



Send data between applications when the order of events is important, for example:

- Ensure that user-entered commands are executed in the right order.
- Display the correct product price by sending price modifications in the right order.
- Prevent a student from enrolling in a course before registering for an account.

For more information, see the [Amazon SQS FAQs](#) and the [Amazon SQS Developer Guide](#).

To create a new queue, choose **Quick-Create Queue**. To configure your queue's parameters, choose **Configure Queue**.

**Configure Queue** **Quick-Create Queue**

- c. Choose **Use SSE**.
- d. For **AWS KMS Customer Master Key (CMK)**, choose **DemoMyCustomCMK** which you created earlier, and then choose **Create Queue**.

Queue Attributes

Default Visibility Timeout **1** 30 seconds Value must be between 9 seconds and 12 hours.

Message Retention Period **1** 4 days Value must be between 1 minute and 14 days.

Maximum Message Size **1** 256 KB Value must be between 1 and 256 KB.

Delivery Delay **1** 0 seconds Value must be between 9 seconds and 15 minutes.

Receive Message Wait Time **1** 0 seconds Value must be between 0 and 20 seconds.

Dead Letter Queue Settings

Use Redrive Policy **1** ☐

Dead Letter Queue **1** Value must be an existing queue name.

Maximum Receives **1** Value must be between 1 and 1000.

**Server-Side Encryption (SSE) Settings**

Use SSE **1** ☒

AWS KMS Customer Master Key (CMK) **1** DemoMyCustomCMK

Description DemoMyCustomCMK

Account 399808215315

Key ARN arn:aws:kms:ap-south-1:399808215315:key/207721a1-6d1f-4073-b695-7963157dcfc

Data Key Reuse Period **1** 5 minutes This value must be between 1 minute and 24 hours.

**Create Queue**

4. Repeat the process to create a second queue (for example, named Demo-MyEncryptedQueue2).
- Your new encrypted queues appear in the list of queues.

Create New Queue

Queue Actions

Filter by Prefix:

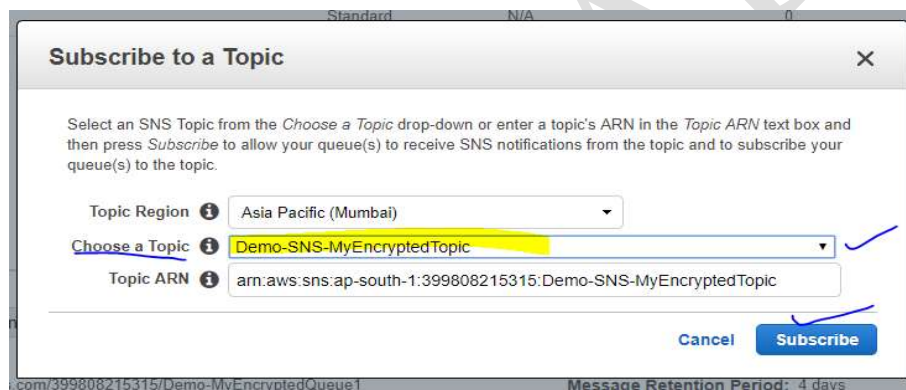
<< 1 to 2 of 2 items >>

<input type="checkbox"/>	Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
<input type="checkbox"/>	Demo-MyEncryptedQueue1	Standard	N/A	0	0	2019-11-07 16:00:55 GMT+05:30
<input checked="" type="checkbox"/>	Demo-MyEncryptedQueue2	Standard	N/A	0	0	2019-11-07 16:02:09 GMT+05:30

- On the Amazon SQS console, choose Demo-MyEncryptedQueue1 and Demo-MyEncryptedQueue2 and then choose **Queue Actions, Subscribe Queues to SNS Topic**.



- In the **Subscribe to a Topic** dialog box, for **Choose a Topic** select **Demo-SNS-MyEncryptedTopic**, and then choose **Subscribe**.



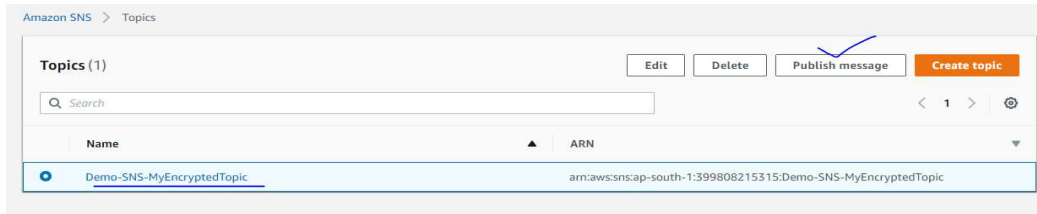
Your encrypted queues' subscriptions to your encrypted topic are displayed in the **Topic Subscription Result** dialog box.

- Choose **OK**.



## Step 4: To Publish a Message to Your Encrypted Topic

1. Sign in to the [Amazon SNS console](#).
2. On the navigation panel, choose **Topics**.
3. From the list of topics, choose **Demo-SNS-MyEncryptedTopic** and then choose **Publish message**.



4. On the **Publish a message** page, do the following:
  - a. (Optional) In the **Message details** section, enter the **Subject** (for example, Testing message publishing).
  - b. In the **Message body** section, enter the message body (for example, My message body is encrypted at rest.).
  - c. Choose **Publish message**.

A screenshot of the 'Publish a message' page in the Amazon SNS console. The page is divided into three main sections: 'Message details', 'Message body', and 'Message attributes'. In the 'Message details' section, the 'Subject' field is highlighted with a yellow box and contains the text 'Testing message publishing'. In the 'Message body' section, the 'Message structure' is set to 'Identical payload for all delivery protocols', and the 'Message body to send to the endpoint' field is highlighted with a yellow box and contains the text 'This is my message body. It is encrypted at rest.'. In the 'Message attributes' section, there are fields for 'Type', 'Name', and 'Value', but they are empty. At the bottom right, the 'Publish message' button is highlighted with a blue checkmark.

- Your message is published to your subscribed encrypted queues.

The screenshot shows the Amazon SNS console interface for a topic named "Demo-SNS-MyEncryptedTopic". The "Details" tab is selected, displaying the topic's name, ARN, display name, and owner. Below the details, there are tabs for "Subscriptions", "Access policy", "Delivery retry policy (HTTP/S)", "Delivery status logging", "Encryption", and "Tags". The "Subscriptions" tab is active, showing a list of three subscriptions, all with a status of "Confirmed" and using the "SQS" protocol. The endpoints for the subscriptions are ARNs pointing to specific queues.

ID	Endpoint	Status	Protocol
003979ff-86a0-46f7-b4ca-590ad8b64cb2	arn:aws:sqs:ap-south-1:399808215315:Demo-MyEncryptedQueue2	Confirmed	SQS
c00a90b9-df64-453c-a793-2dd69673eb2a	arn:aws:sqs:ap-south-1:399808215315:Demo-MyEncryptedQueue3	Confirmed	SQS
ba401a4e-af5d-48c7-89e4-46bc41fc69eb	arn:aws:sqs:ap-south-1:399808215315:Demo-MyEncryptedQueue1	Confirmed	SQS

### Step 5: To Verify Message Delivery

1. Sign in to the [Amazon SQS console](#).
2. From the list of queues, choose **Demo-MyEncryptedQueue1** and then choose **Queue Actions, View/Delete Messages**.

The screenshot shows the Amazon SQS console interface. A dropdown menu is open for the "Queue Actions" of a queue named "Demo-MyEncryptedQueue1". The menu options include "Send a Message", "View/Delete Messages" (which is highlighted with a blue checkmark), "Configure Queue", "Add a Permission", "Purge Queue", "Delete Queue", "Subscribe Queue to SNS Topic", and "Configure Trigger for Lambda Function".

3. On the **View/Delete Messages in Demo-MyEncryptedQueue1** page, choose **Start polling for messages**.

The message [that you sent earlier](#) is displayed.

The screenshot shows the "View/Delete Messages in Demo-MyEncryptedQueue1" page. At the top, there are controls for "View up to: 10 messages" and "Poll queue for: 30 seconds". Below these, there is a "Start Polling for Messages" button and a "Stop Now" button. The main area displays a table with columns for "Delete", "Body", "Size", "Sent", and "Receive Count". A modal dialog is open, providing instructions on how to use the "Start Polling for Messages" button and how to delete messages. The dialog includes a "Start Polling for Messages" button and a "Don't show this again" checkbox.

View/Delete Messages in Demo-MyEncryptedQueue1

View up to: 10 messages Poll queue for: 30 seconds

Start Polling for Messages Stop Now

Polling for new messages once every 2 seconds.

Delete	Body	Size	Sent	Receive Count	
<input checked="" type="checkbox"/>	{ "Type" : "Notification", "MessageId" : "2a884aea-3725-51fd-bf9f-1cd7cff1ec2f", "TopicArn" : "arn:aws:sns:ap-sou-1:399808215315:Demo-SNS-MyEncryptedTopic", "Subject" : "Testing message publishing", "Message" : "This is My message body is encrypted at rest", "Timestamp" : "2019-11-07T10:48:52.341Z", "SignatureVersion" : "1",	1 KB	2019-11-07 16:18:52 GMT+05:30	1	<a href="#">More Details</a>

Stopped after polling the queue at 0.2 receives/second for 30.0 seconds. Messages shown above are now available to other consumers.

Close Delete Messages

4. Choose **More Details** to view your message.

Message Details

Message Body Message Attributes

```

{
  "Type" : "Notification",
  "MessageId" : "2a884aea-3725-51fd-bf9f-1cd7cff1ec2f",
  "TopicArn" : "arn:aws:sns:ap-south-1:399808215315:Demo-SNS-MyEncryptedTopic",
  "Subject" : "Testing message publishing",
  "Message" : "This is My message body is encrypted at rest",
  "Timestamp" : "2019-11-07T10:48:52.341Z",
  "SignatureVersion" : "1",
}

```

Message ID: aae23b94-d0ac-4ac7-b3c7-3a10dc05266f

Size: 1 KB

MD5 of Body: 787f7f94881c4508a563114ac218c7d8

Sender Account ID: AIDAJKTPXYAO6CI6DPKS

Sent: 2019-11-07 16:18:52.454 GMT+05:30

First Received: 2019-11-07 16:24:45.420 GMT+05:30

Receive Count: 1

Message Attribute Count: 0

Close

- When you're finished, choose **Close**.
- Repeat the process for **Demo-MyEncryptedQueue2**.

## 10. Set CloudWatch Alarms for Amazon SNS Metrics

CloudWatch also allows you to set alarms when a threshold is met for a metric. For example, you could set an alarm for the metric, **NumberOfNotificationsFailed**, so that when your specified threshold number is met within the sampling period, then an email notification would be sent to inform you of the event.

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Alarms**, and then choose the **Create Alarm** button. This launches the **Create Alarm** wizard.
3. Scroll through the Amazon SNS metrics to locate the metric you want to place an alarm on. Select the metric to create an alarm on and choose **Continue**.
4. Fill in the **Name**, **Description**, **Threshold**, and **Time** values for the metric, and then choose **Continue**.
5. Choose **Alarm** as the alarm state. If you want CloudWatch to send you an email when the alarm state is reached, choose either an existing Amazon SNS topic or choose **Create New Email Topic**. If you choose **Create New Email Topic**, you can set the name and email addresses for a new topic. This list will be saved and appear in the drop-down box for future alarms. Choose **Continue**.

### Note

If you use **Create New Email Topic** to create a new Amazon SNS topic, the email addresses must be verified before they will receive notifications. Emails are sent only when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, they will not receive a notification.

6. At this point, the **Create Alarm** wizard gives you a chance to review the alarm you're about to create. If you need to make any changes, you can use the **Edit** links on the right. Once you are satisfied, choose **Create Alarm**.

## 11. AWS Event Fork Pipelines to an Amazon SNS Topic

- AWS Event Fork Pipelines is a suite of open-source [nested applications](#), based on the [AWS Serverless Application Model](#) (AWS SAM), which you can deploy directly from the [AWS Event Fork Pipelines suite](#) (choose **Show apps that create custom IAM roles or resource policies**) into your AWS account.

## 11.1. To Deploy and Subscribe the Event Storage and Backup Pipeline

- How to deploy the **Event Storage and Backup Pipeline** and subscribe it to an Amazon SNS topic.
- This process automatically turns the **SAM template** associated with the pipeline **into** an **AWS CloudFormation stack**, and then **deploys** the stack into **your AWS account**.
- This process also creates and configures the set of resources which comprise the Event Storage and Backup Pipeline, including the following:
  - Amazon SQS queue
  - Lambda function
  - Kinesis Data Firehose delivery stream
  - Amazon S3 backup bucket

1. Sign in to the **AWS Lambda console**.
2. On the navigation panel, choose **Functions** and then choose **Create function**.
3. On the **Create function** page, do the following:
  - a. Choose **Browse serverless app repository**, **Public applications**, **Show apps that create custom IAM roles or resource policies**.
  - b. Search for **fork-event-storage-backup-pipeline** and then choose the application.
4. On the **fork-event-storage-backup-pipeline** page, do the following:
  - a. In the **Application settings** section, enter an **Application name** (for example, **my-app-backup**).

**Note:** For each deployment, the application name must be unique. If you reuse an application name, the deployment will update only the previously deployed AWS CloudFormation stack (rather than create a new one).

- b. (Optional) For **BucketArn**, enter the ARN of the S3 bucket into which incoming events are loaded. If you don't enter a value, a new S3 bucket is created in your AWS account.
- c. (Optional) For **DataTransformationFunctionArn**, enter the ARN of the Lambda function through which the incoming events are transformed. If you don't enter a value, data transformation is disabled.
- d. (Optional) Enter one of the following **LogLevel** settings for the execution of your application's Lambda function:
  - **DEBUG**
  - **ERROR**



- INFO (default)
  - WARNING
- e. For **TopicArn**, enter the ARN of the Amazon SNS topic to which this instance of the fork pipeline is to be subscribed.
  - f. (Optional) For **StreamBufferingIntervalInSeconds** and **StreamBufferingSizeInMBs**, enter the values for configuring the buffering of incoming events. If you don't enter any values, 300 seconds and 5 MB are used.
  - g. (Optional) Enter one of the following **StreamCompressionFormat** settings for compressing incoming events:
    - GZIP
    - SNAPPY
    - UNCOMPRESSED (default)
    - ZIP
  - h. (Optional) For **StreamPrefix**, enter the string prefix to name files stored in the S3 backup bucket. If you don't enter a value, no prefix is used.
  - i. (Optional) For **SubscriptionFilterPolicy**, enter the Amazon SNS subscription filter policy, in JSON format, to be used for filtering incoming events. The filter policy decides which events are stored in the S3 backup bucket. If you don't enter a value, no filtering is used (all events are stored).
  - j. Choose **I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.** and then choose **Deploy**.
5. On the **Deployment status for *my-app*** page, Lambda displays the **Your application is being deployed** status.
  6. In the **Resources** section, AWS CloudFormation begins to create the stack and displays the **CREATE\_IN\_PROGRESS** status for each resource. When the process is complete, AWS CloudFormation displays the **CREATE\_COMPLETE** status.
  7. When the deployment is complete, Lambda displays the **Your application has been deployed** status.
  8. Messages published to your Amazon SNS topic are stored in the S3 backup bucket provisioned by the Event Storage and Backup pipeline automatically.

## 11.2. To Deploy and Subscribe the Event Search and Analytics Pipeline

- How to deploy the [Event Search and Analytics Pipeline](#) and subscribe it to an Amazon SNS topic.
- This process also creates and configures the set of resources which comprise the Event Search and Analytics Pipeline, including the following:
  - Amazon SQS queue
  - Lambda function



- Kinesis Data Firehose delivery stream
  - Amazon Elasticsearch Service domain
  - Amazon S3 dead-letter bucket
1. Sign in to the [AWS Lambda console](#).
  2. On the navigation panel, choose **Functions** and then choose **Create function**.
  3. On the **Create function** page, do the following:
    - a. Choose **Browse serverless app repository**, **Public applications**, **Show apps that create custom IAM roles or resource policies**.
    - b. Search for `fork-event-search-analytics-pipeline` and then choose the application.
  4. On the `fork-event-search-analytics-pipeline` page, do the following:
    - a. In the **Application settings** section, enter an **Application name** (for example, my-app-search).

**Note:** For each deployment, the application name must be unique. If you reuse an application name, the deployment will update only the previously deployed AWS CloudFormation stack (rather than create a new one).

- b. (Optional) For **DataTransformationFunctionArn**, enter the ARN of the Lambda function used for transforming incoming events. If you don't enter a value, data transformation is disabled.
- c. (Optional) Enter one of the following **LogLevel** settings for the execution of your application's Lambda function:
  - DEBUG
  - ERROR
  - INFO (default)
  - WARNING
- d. (Optional) For **SearchDomainArn**, enter the ARN of the Amazon ES domain, a cluster which configures the needed compute and storage functionality. If you don't enter a value, a new domain is created with the default configuration.
- e. For **TopicArn**, enter the ARN of the Amazon SNS topic to which this instance of the fork pipeline is to be subscribed.
- f. For **SearchIndexName**, enter the name of the Amazon ES index for event search and analytics.

**Note:** The following limits apply to index names:

- Can't include uppercase letters

- Can't include the following characters: \ / \* ? " < > | ` , #
  - Can't begin with the following characters: - + \_
  - Can't be the following: ...
  - Can't be longer than 80 characters
  - Can't be longer than 255 bytes
  - Can't contain a colon (from Amazon ES 7.0)
- g. (Optional) Enter one of the following **SearchIndexRotationPeriod** settings for the rotation period of the Amazon ES index:
- NoRotation (default)
  - OneDay
  - OneHour
  - OneMonth
  - OneWeek

Index rotation appends a timestamp to the index name, facilitating the expiration of old data.

- h. For **SearchTypeName**, enter the name of the Amazon ES type for organizing the events in an index.

#### Note

- Amazon ES type names can contain any character (except null bytes) but can't begin with \_.
  - For Amazon ES 6.x, there can be only one type per index. If you specify a new type for an existing index that already has another type, Kinesis Data Firehose returns a runtime error.
- i. (Optional) For **StreamBufferingIntervalInSeconds** and **StreamBufferingSizeInMBs**, enter the values for configuring the buffering of incoming events. If you don't enter any values, 300 seconds and 5 MB are used.
- j. (Optional) Enter one of the following **StreamCompressionFormat** settings for compressing incoming events:
- GZIP
  - SNAPPY
  - UNCOMPRESSED (default)
  - ZIP
- k. (Optional) For **StreamPrefix**, enter the string prefix to name files stored in the S3 dead-letter bucket. If you don't enter a value, no prefix is used.
- l. (Optional) For **StreamRetryDurationInSeconds**, enter the retry duration for cases when Kinesis Data Firehose can't index events in the Amazon ES index. If you don't enter a value, then 300 seconds is used.
- m. (Optional) For **SubscriptionFilterPolicy**, enter the Amazon SNS subscription filter policy, in JSON format, to be used for filtering incoming events. The filter policy decides which events are indexed in the Amazon ES index. If you don't enter a value, no filtering is used (all events are indexed).

- n. Choose **I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.** and then choose **Deploy**.
5. On the **Deployment status for *my-app-search*** page, Lambda displays the **Your application is being deployed** status.
6. In the **Resources** section, AWS CloudFormation begins to create the stack and displays the **CREATE\_IN\_PROGRESS** status for each resource. When the process is complete, AWS CloudFormation displays the **CREATE\_COMPLETE** status.
7. When the deployment is complete, Lambda displays the **Your application has been deployed** status.
8. Messages published to your Amazon SNS topic are indexed in the Amazon ES index provisioned by the Event Search and Analytics pipeline automatically. If the pipeline can't index an event, it stores it in a S3 dead-letter bucket.

## 12. Deploying and Testing the AWS Event Fork Pipelines Sample Application

<https://docs.aws.amazon.com/sns/latest/dg/sns-tutorial-deploy-test-fork-pipelines-sample-application.html?shortFooter=true>

## 13. Reference Links:

- <https://docs.aws.amazon.com/sns/latest/dg/welcome.html?shortFooter=true>
- <https://aws.amazon.com/getting-started/tutorials/filter-messages-published-to-topics/>
- <https://aws.amazon.com/sns/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc>
- AWS Simple Notification Service (SNS) | AWS Tutorial For Beginners:  
<https://www.youtube.com/watch?v=z0JADXyH8Kg>
- Using Mobile Push Notifications with Amazon SNS - Simply Notification Service on AWS  
<https://www.youtube.com/watch?v=qR4aCEEBKAI>
- AWS Monitoring - SNS and CloudWatch Introduction  
<https://www.youtube.com/watch?v=GPIQikruIIk>