

Message queue service using AWS SNS and SQS

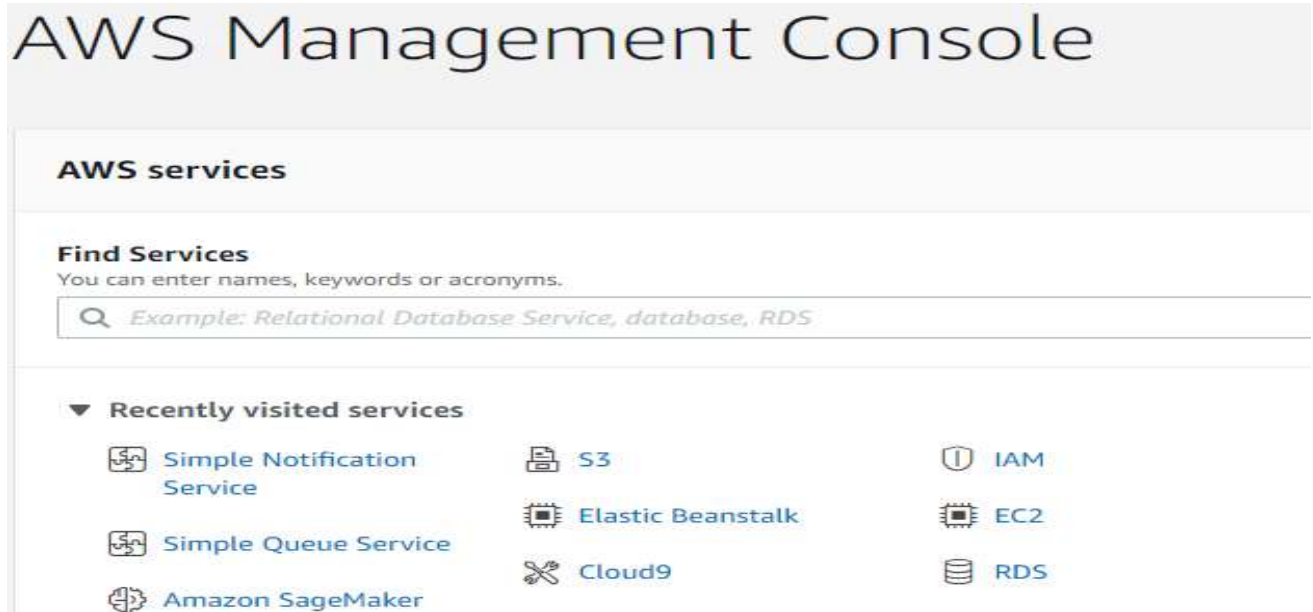
Theory of SNS:

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 and subscribers—also referred to as producers and consumers. 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.

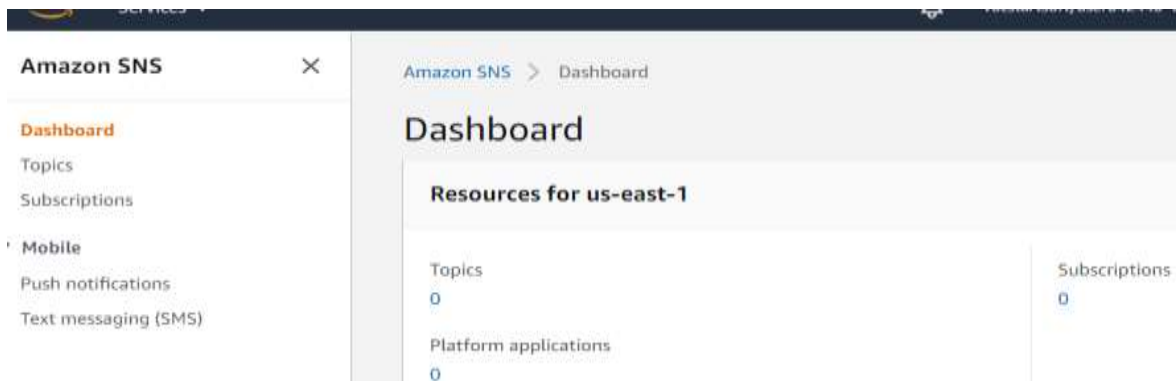
When using Amazon SNS, you (as the owner) create a topic and control access to it by defining policies that determine which publishers and subscribers can communicate with the topic. A publisher sends messages to topics that they have created or to topics they have permission to publish to. Instead of 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.

Procedure:

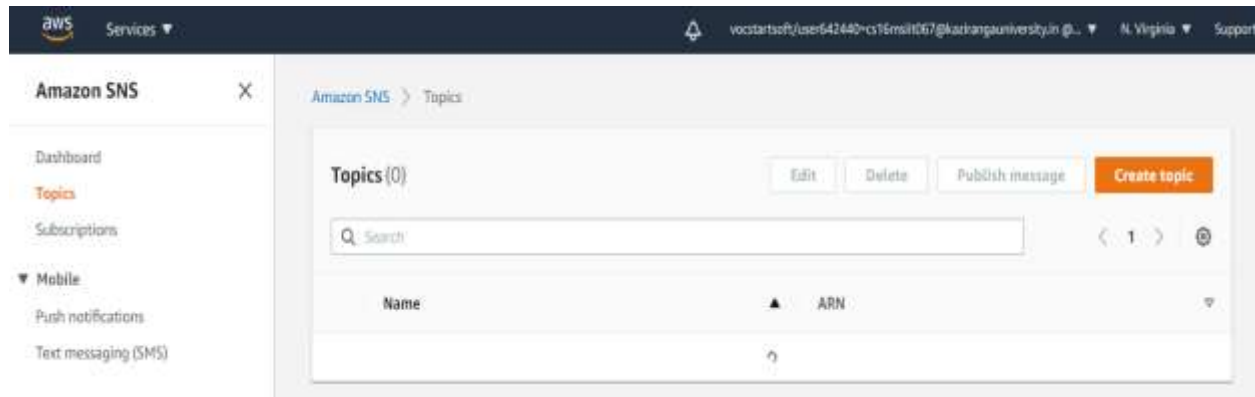
1: From the service catalogue select “**SNS and SQS**” (Simple Notification Service and Simple Queue Service).



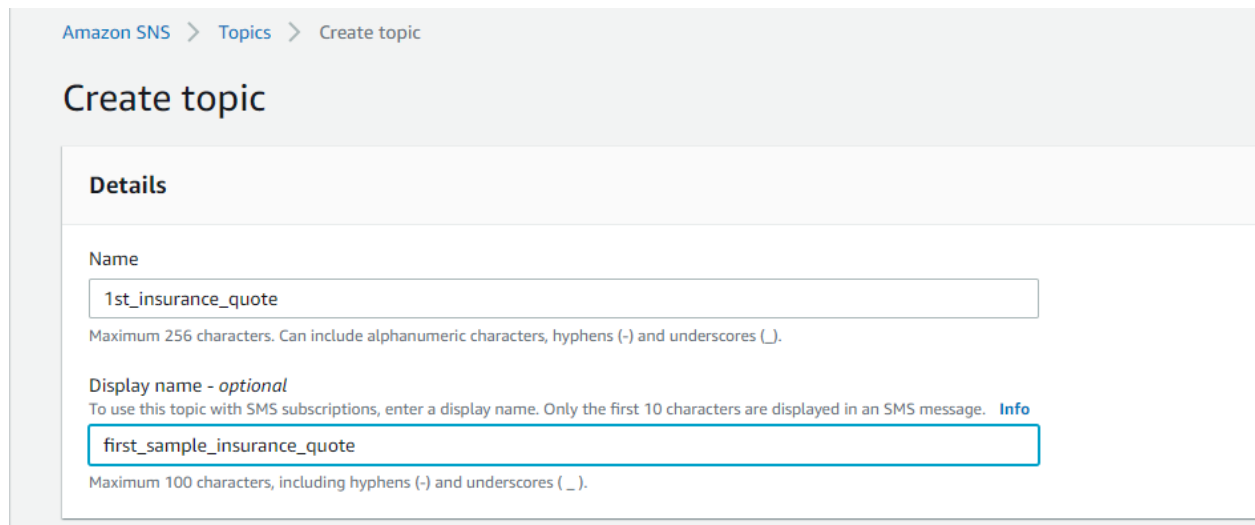
2: First we will go to the SNS dashboard.



3: Click on the “Create topic”



4: On the details we will give our “name” and “Display name”.



5: Click on the **Create topic**.

Maximum receive rate
-

Retry-backoff function
Linear

Override subscription policy
False

► **Delivery status logging - optional**
These settings configure the logging of message delivery status to CloudWatch Logs. [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**

6: Our Topic is successfully created.

aws Services

Topic 1st_insurance_quote created successfully.
You can create subscriptions and send messages to them from this topic. [Publish message](#)

Amazon SNS > Topics > 1st_insurance_quote

1st_insurance_quote [Edit](#) [Delete](#) [Publish message](#)

Details

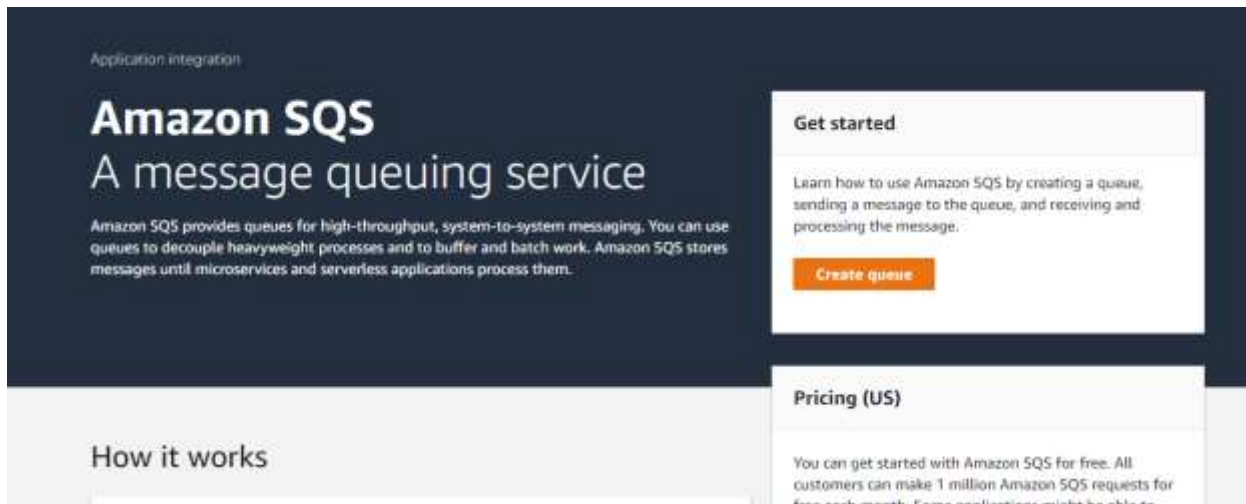
Name	Display name
1st_insurance_quote	first_sample_insurance_quote
ARN	Topic owner
arn:aws:sns:us-east-1:300648026287:1st_insurance_quote	300648026287

Theory of SQS:

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables us to decouple and scale micro services, distributed systems, and serverless applications. SQS eliminates the complexity and overhead associated with managing and operating message oriented middleware, and empowers developers to focus on differentiating work. Using SQS, we can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available. Get started with SQS in minutes using the AWS console, Command Line Interface or SDK of our choice, and three simple commands.

SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent

7: Select on the SQS and click on the “Create queue”.




8: We will give the name for the queue and click on the create queue.

Name

first_vehicle_insurance_quote

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).


-rest encryption to your queue, enable server-side encryption. [Info](#)

and filter your resources or track your AWS costs. [Learn more](#) 

[Cancel](#) [Create queue](#)

9: Like this we will Create queue (here we have created 3 queues)

Amazon SQS > Queues

Queues (3)  [Edit](#) [Delete](#) [Purge](#) [Actions](#) [Create queue](#)

	Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
<input type="radio"/>	first_generic_quote	Standard	9/28/2020, 18:27:36	0	0	-	-
<input type="radio"/>	first_life_insurance_quote	Standard	9/28/2020, 18:26:47	0	0	-	-
<input type="radio"/>	first_vehicle_insurance_quote	Standard	9/28/2020, 18:24:36	0	0	-	-

10: Now we will select one queue and from actions select **“Subscribe to Amazon SNS topic”**.

Amazon SQS > Queues

Queues (3) Refresh Edit Delete Purge Actions Create queue

Send and receive messages
Subscribe to Amazon SNS topic
Configure Lambda function trigger

Name	Type	Created	Messages available	Message flight	Message encryption	Message deduplication
<input type="radio"/> first_generic_quote	Standard	9/28/2020, 18:27:36	0	0	-	-
<input type="radio"/> first_life_insurance_quote	Standard	9/28/2020, 18:26:47	0	0	-	-
<input checked="" type="radio"/> first_vehicle_insurance_quote	Standard	9/28/2020, 18:24:36	0	0	-	-

11: We will **Specify an Amazon SNS topic available for this queue** and click **Save**

Amazon SQS > Queues > first_vehicle_insurance_quote > Subscribe to Amazon SNS topic

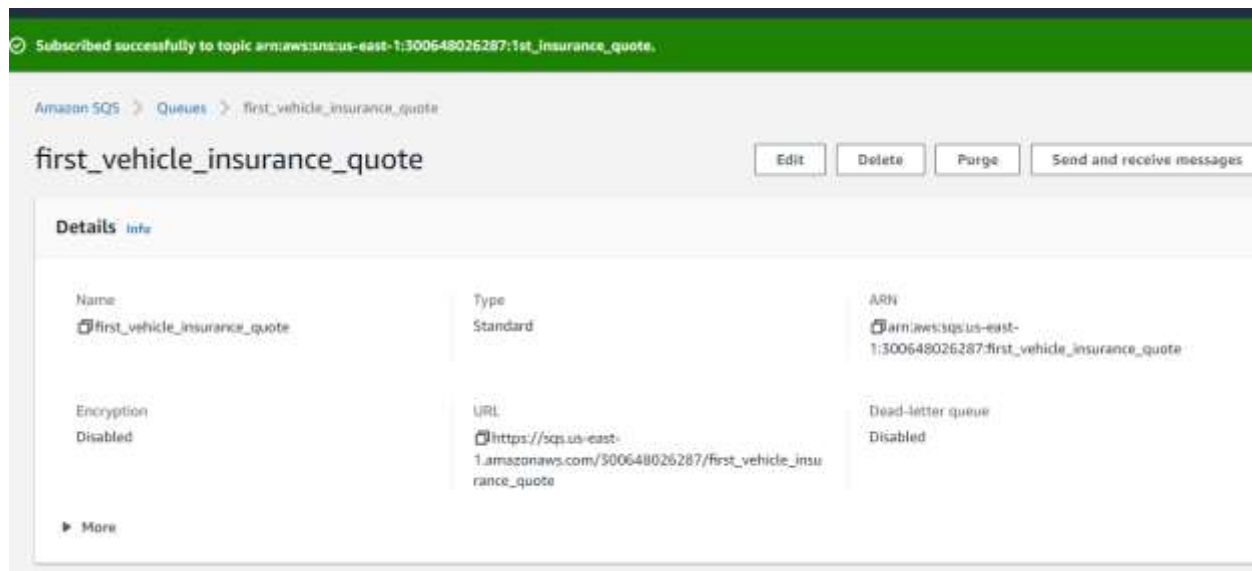
Subscribe to Amazon SNS topic [Info](#)

Amazon SNS topic
To allow your queue to receive messages from an Amazon SNS topic, subscribe it to an Amazon SNS topic.

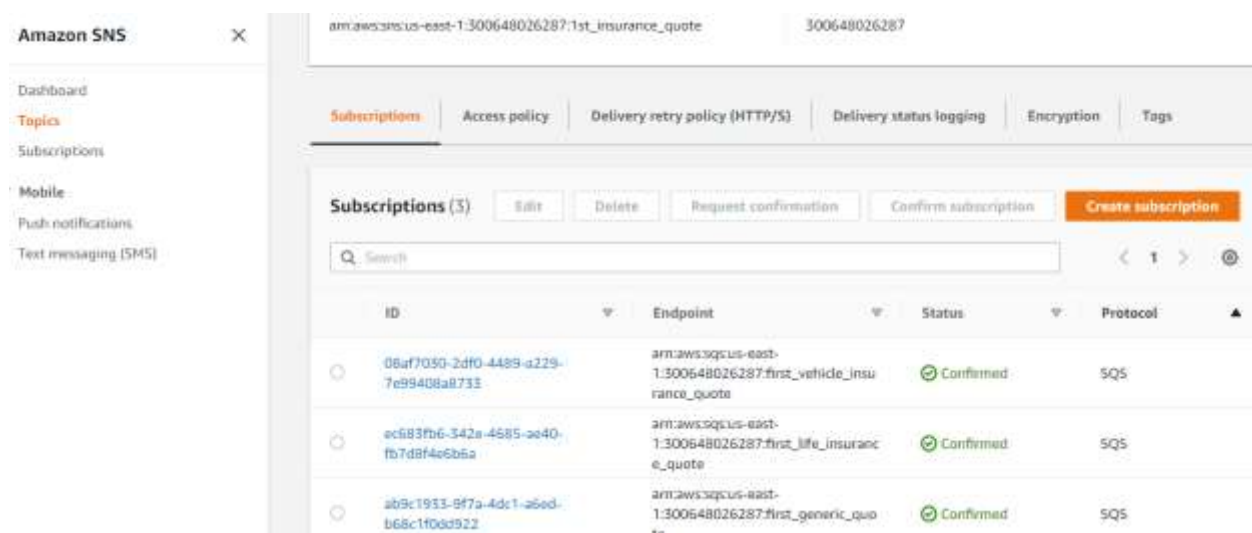
Specify an Amazon SNS topic available for this queue.

Cancel Save

12: We have **subscribed** successfully to the topic.



13: After connecting all the SNS topics we can see the subscription as below fig.



14: Now we will create **subscription filter policy** for each of the subscription . Click on edit and set the subscription policy with JASON code. (Below fig is for vehicle_insurance).

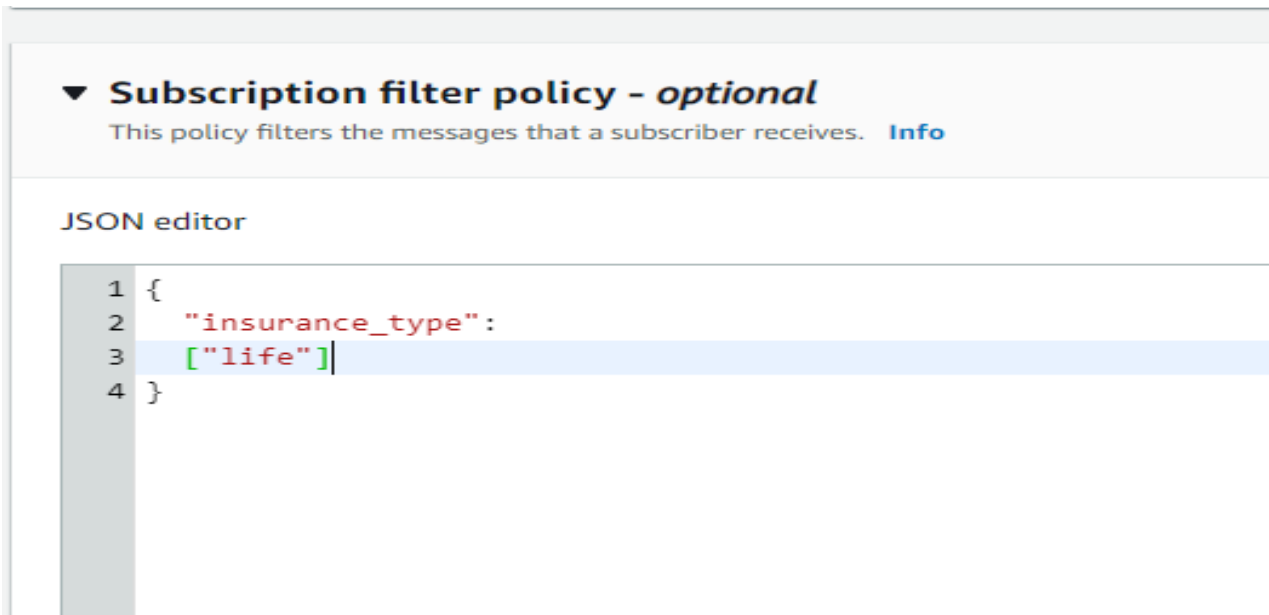
▼ **Subscription filter policy - optional**

This policy filters the messages that a subscriber receives. [Info](#)

JSON editor

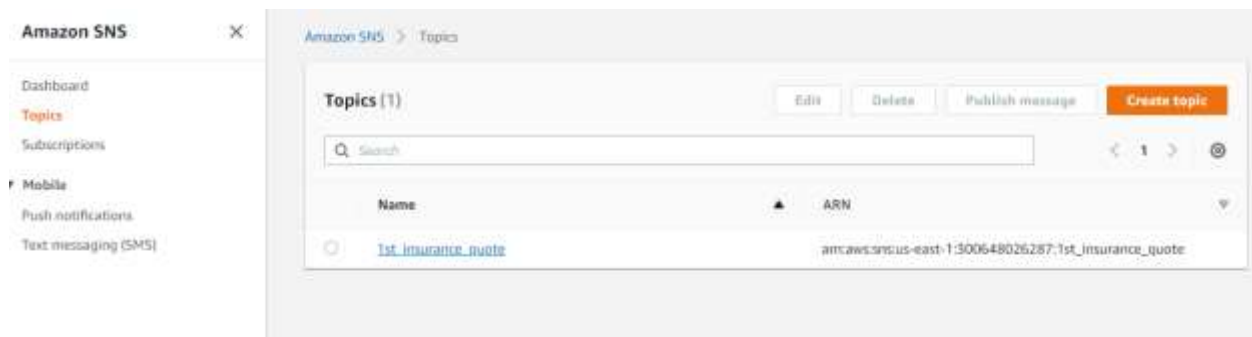
```
1 {  
2   "insurance_type":  
3   ["car","boat"]  
4 }
```

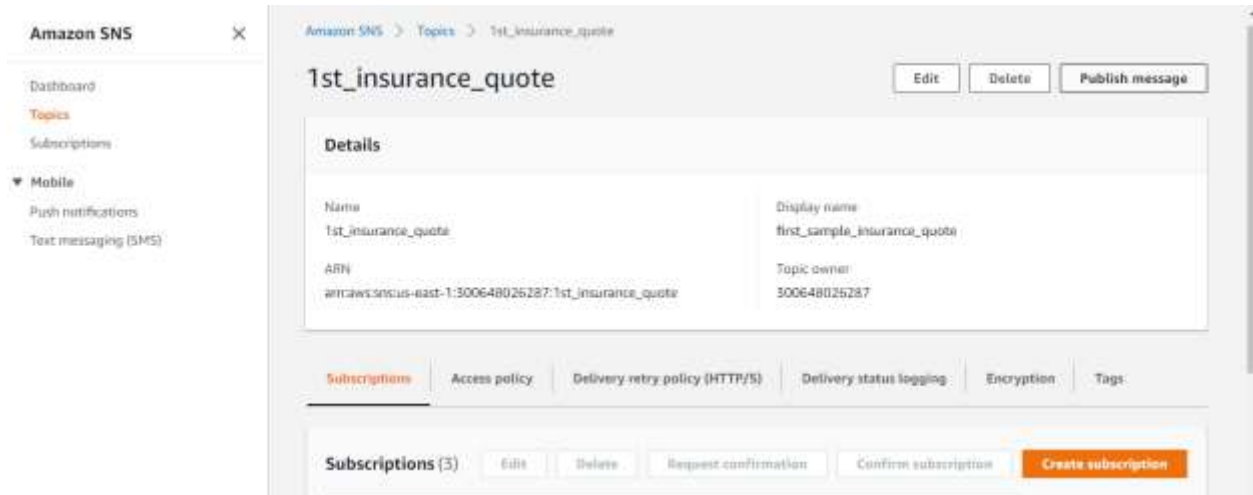
15: Similarly we created subscription policy for life_insurance.



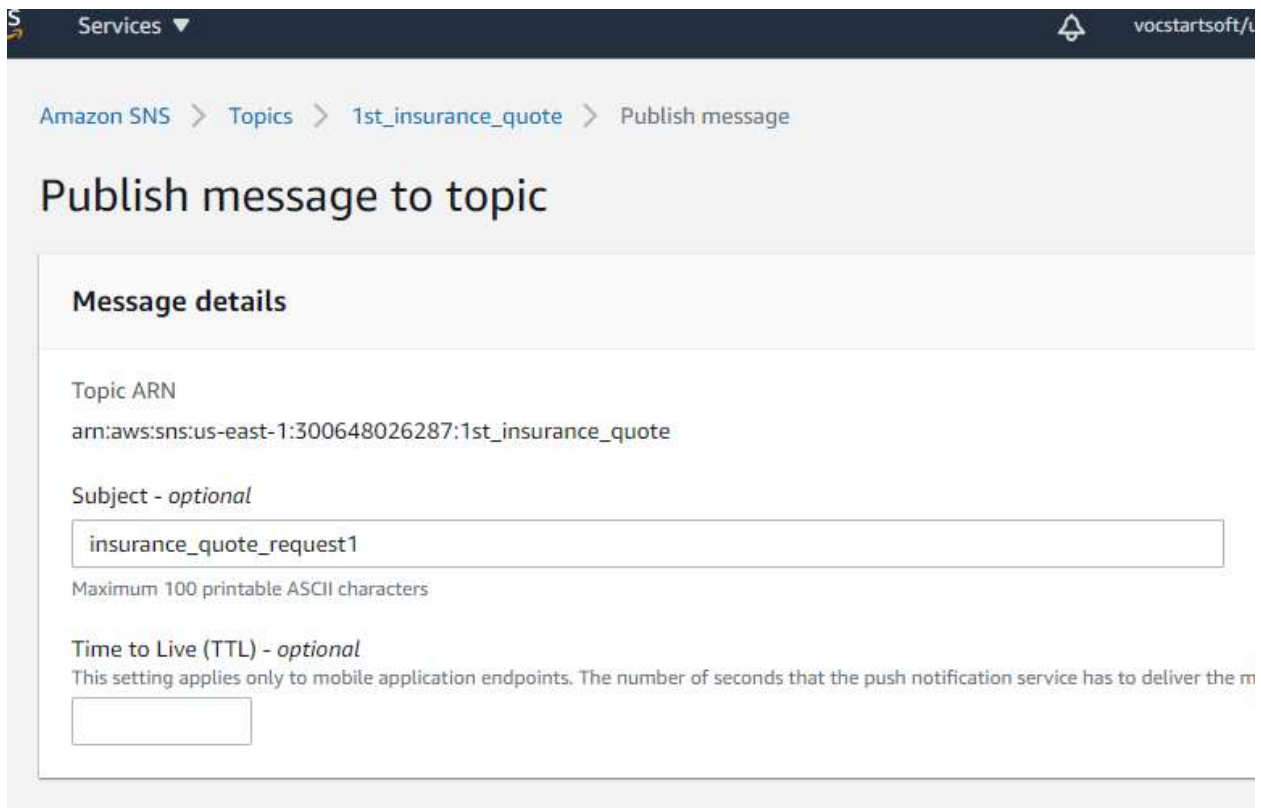
Publishing Message:

16: For publishing message on the SNS page click on the **topic** and click on the **publish the message**.





17: Then we have to provide the subject for the published message.



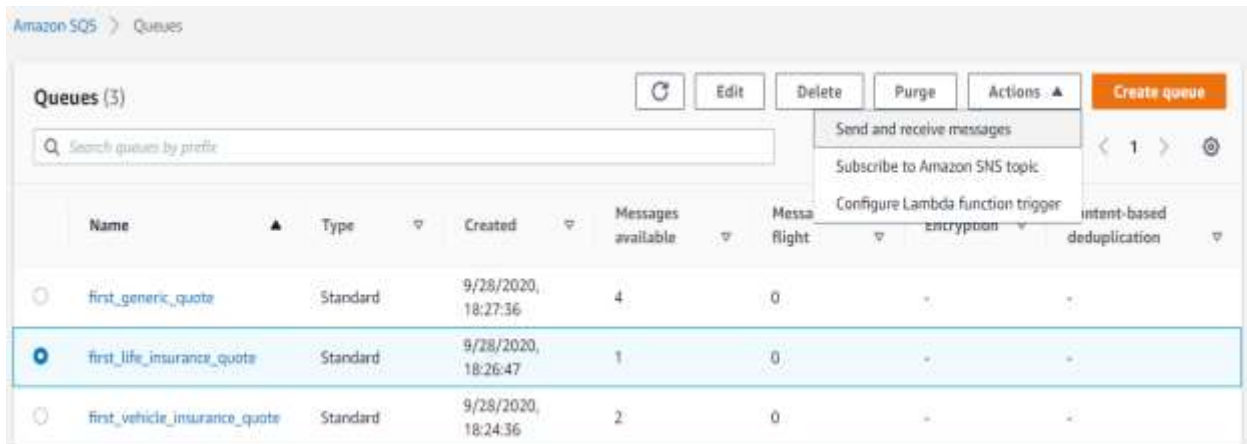
18: And we have to type the message and set the message attribute and click **publish message**.

The screenshot shows a configuration window for a message. At the top, there are two radio button options for the delivery protocol. The first option, 'Identical payload for all delivery protocols', is selected and highlighted with a blue border. It includes a subtext: 'The same payload is sent to endpoints subscribed to the topic, regardless of their delivery protocol.' The second option, 'Custom p protocol', is unselected. Below these options is a text area labeled 'Message body to send to the endpoint'. Inside this text area, the message body is pre-filled with the text '1 2018,volvo 560,Montreal'.

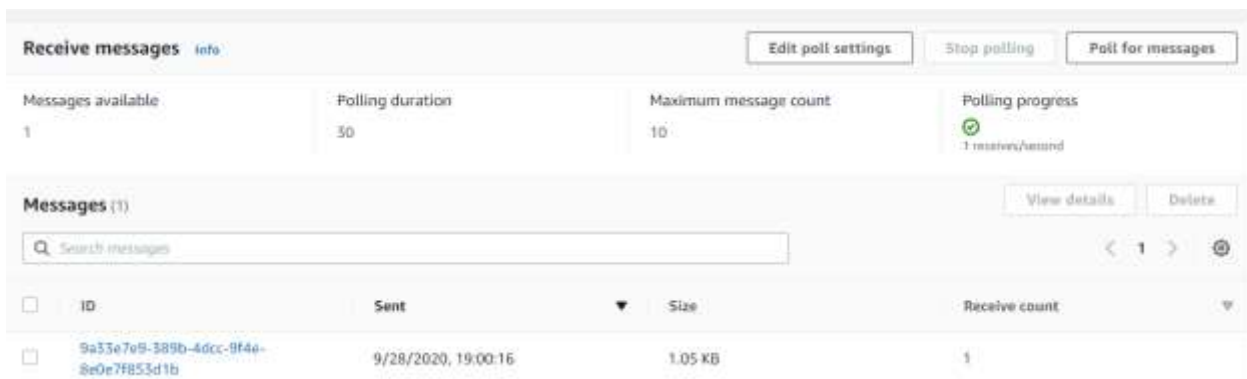
This screenshot shows the 'Message attributes' section of the configuration window. It features a header with the title 'Message attributes' and a descriptive subtitle: 'Message attributes let you provide structured metadata items (such as timestamps, geospatial data, signatures, and identifiers) for the message.' followed by a blue 'Info' link. Below the header is a table-like structure with three columns: 'Type', 'Name', and 'Value'. The 'Type' column has a dropdown menu currently set to 'String'. The 'Name' column contains the text 'insurance_type'. The 'Value' column contains the text 'car'. To the right of the 'Value' input is a 'Remove' button. Below the table is a button labeled 'Add another attribute'. At the bottom right of the window, there are two buttons: 'Cancel' and 'Publish message'.

Similarly we have created for vehicle and generic insurance message with attribute and published the message.

19: Now we will go to the SQS console and verify if the messages are routed to the corresponding message. We can see on the below fig that there are values on the “**messages available**” , then we click on the action and select “**Send and receive messages**” .



20: Click the **poll for messages**, we can see the receive count. Here we have shown that message come from the SNS to SQS.



Publishing message as a Notifications Services to E-mail:

21: To send notifications to the e-mail, first we have to create new subscription. Choose the Topic ARN(Amazon Resource Names) and select the protocol as **E-mail**, provide the e-mail we want to send the notification on **the Endpoint**. And click on the **Create Subscription**.

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

arn:aws:sns:us-east-1:300648026287:1st_in X

Protocol

The type of endpoint to subscribe

Email

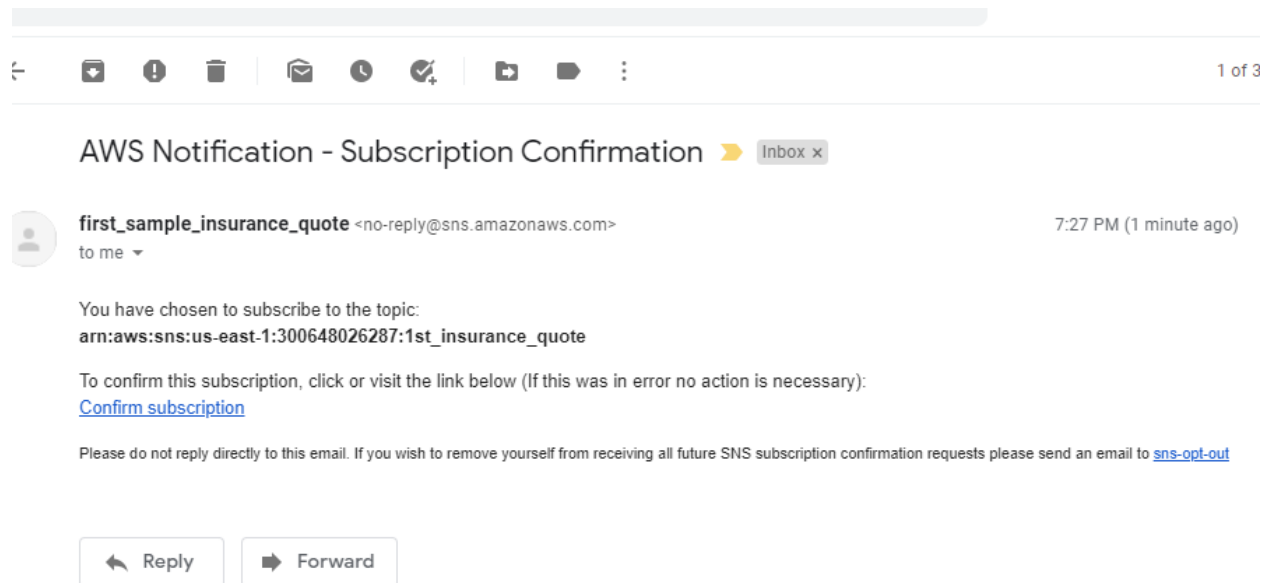
Endpoint

An email address that can receive notifications from Amazon SNS.

ankitmohari186@gmailcom

Create subscription

22: Now we will get a subscription request message.



22: After we click on the Confirm subscription we will get the window displaying that “Subscription confirmed”.



Simple Notification Service

Subscription confirmed!

You have subscribed [ankitmohari186@gmail.com](#) to the topic:
1st_insurance_quote.

Your subscription's id is:

arn:aws:sns:us-east-1:300648026287:1st_insurance_quote:bcd076a3-210a-42f6-bd2a-b23d1c48b99e

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

23: Now we can publish a message from the confirm subscription. Click on the **ID** of the confirm subscription .

Amazon SNS > Subscriptions

Subscriptions (4) Edit Delete Request confirmation Confirm subscription Create subscription

Search

ID	Endpoint	Status	Protocol	Topic
bcd076a3-210a-42f6-bd2a-b23d1c48b99e	ankitmohari186@gmail.com	Confirmed	EMAIL	1st_insurance_quote
ab9c1933-9f7a-4dc1-a6ed-b68c1f0dd922	arn:aws:sqs:us-east-1:300648026287:first_generi_c_quote	Confirmed	SQS	1st_insurance_quote
ec683fb6-342e-4685-ae40-fb7d8f4e6b6a	arn:aws:sqs:us-east-1:300648026287:first_life_insurance_quote	Confirmed	SQS	1st_insurance_quote
08af7030-2df0-4469-a229-7e99408a8733	arn:aws:sqs:us-east-1:300648026287:first_vehicle_insurance_quote	Confirmed	SQS	1st_insurance_quote

24: From particular subscription account click on the **Topic**.

aws Services

Amazon SNS

Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)

Amazon SNS > Topics > 1st_insurance_quote > Subscription: bcd076a3-210a-42f6-bd2a-b23d1c48b99e

Subscription: bcd076a3-210a-42f6-bd2a-b23d1c48b99e Edit Delete

Details

ARN arn:aws:sns:us-east-1:300648026287:1st_insurance_quote:bcd076a3-210a-42f6-bd2a-b23d1c48b99e	Status Confirmed
Endpoint ankitmohari186@gmail.com	Protocol EMAIL
Topic 1st_insurance_quote	

25: Select the ID and click on the **Publish message**.

Subscriptions (4)					Edit	Delete	Request confirmation	Confirm subscription	Create subscription
<input type="text" value="Search"/>					< 1 > ⚙				
	ID	Endpoint	Status	Protocol					
<input checked="" type="radio"/>	bcd076a3-210a-42f6-bd2a-b23d1c48b99e	ankitmohari186@gmail.com	✔ Confirmed	EMAIL					
<input type="radio"/>	08af7030-2df0-4489-a229-7e99408a8733	arn:aws:sqs:us-east-1:300648026287:first_vehicle_insu	✔ Confirmed	SQS					

26: We have provide our subject.

Amazon SNS > Topics > 1st_insurance_quote > Publish message

Publish message to topic

Message details

Topic ARN
arn:aws:sns:us-east-1:300648026287:1st_insurance_quote

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

27: We will type the **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 Hello , the message is for the experiment SNS and SQS
```

28: We will select the message attributes and click on the **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

String ▼

Name

insurance_type

Value

car

Remove

Add another attribute

Cancel

Publish message

29: We will get the published message on our confirmed e-mail



30: We have successfully send the published message on our confirmed e-mail.