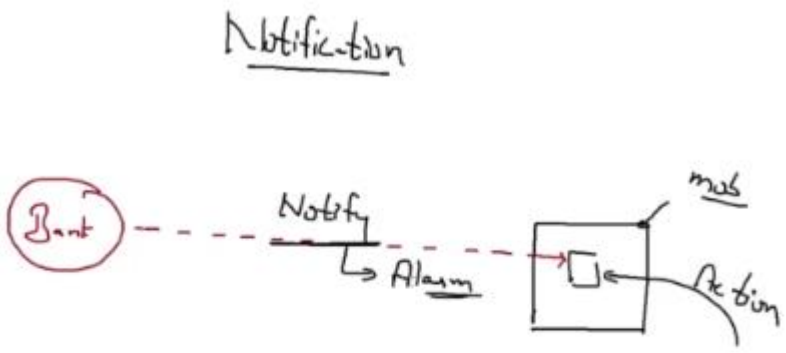
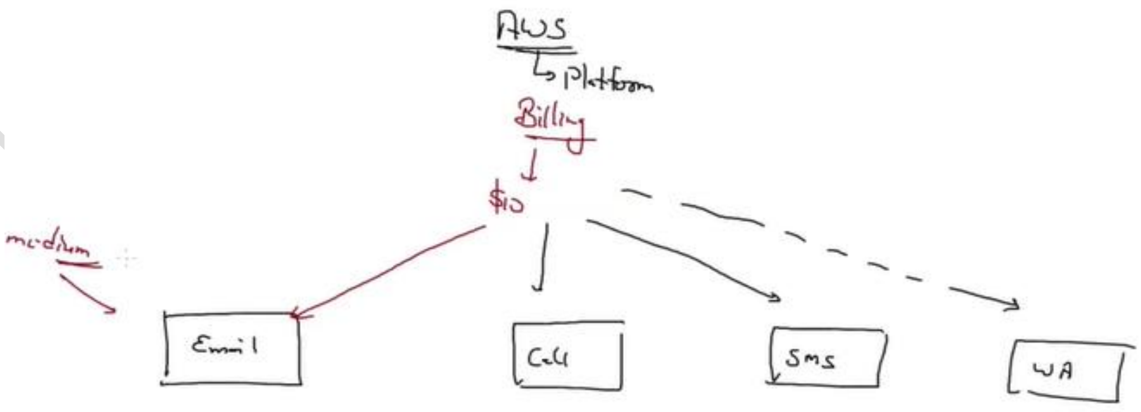
AWS Training Session No.6

Summary 13-03-2024

- In real if we create an app after we need to set the notification service we need to implement
- Every app needs notification in the AWS cloud we use **SNS service** as a notification
- To update your notification service give an alarm to take action
- Notification gives information about what happened in the app

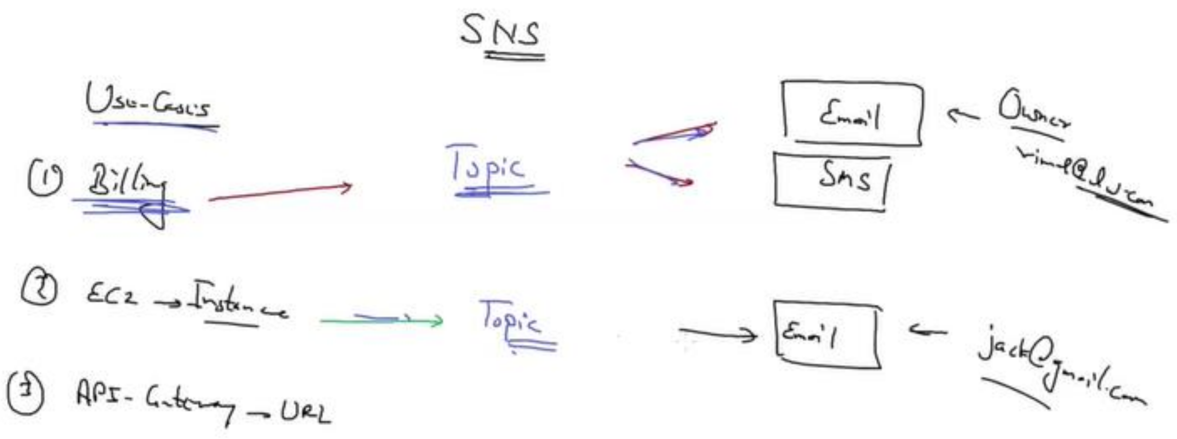


- For example we create one app
- In that app we have different services eg. email, call, SMS, WA, etc
- Every service needs notification, which means if any SMS comes to the app then how to know the SMS comes we use a notification



- SNS Service is a serverless service this service is used for notification
- SNS service is fully managed by AWS cloud

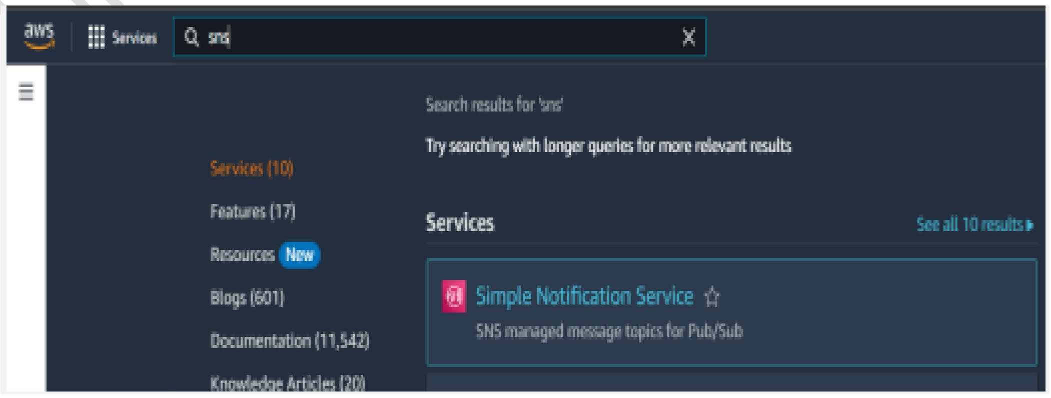
- Amazon **Simple Notification Service** can be used to send notifications to users about any event or change taking place inside their AWS account.
- Amazon **SNS** is a managed messaging service for communication, allowing messaging between decoupled microservices applications or directly to users with SMS
- For example we want your AWS billing service to give all the information the email or you want if your EC2 service stopped and launched
- And we want the notification that we need SNS service but we need to create a topic for that
- A **topic** is like a channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.



- An AWS SNS topic is a logical access point that acts as a communication channel. A topic lets you group multiple endpoints (such as AWS Lambda, Amazon SQS, HTTP/S, or an email address).
- For Every use case we need one topic

Getting started with SNS:

- To create a Topic we follow the steps
- First go and search for SNS service



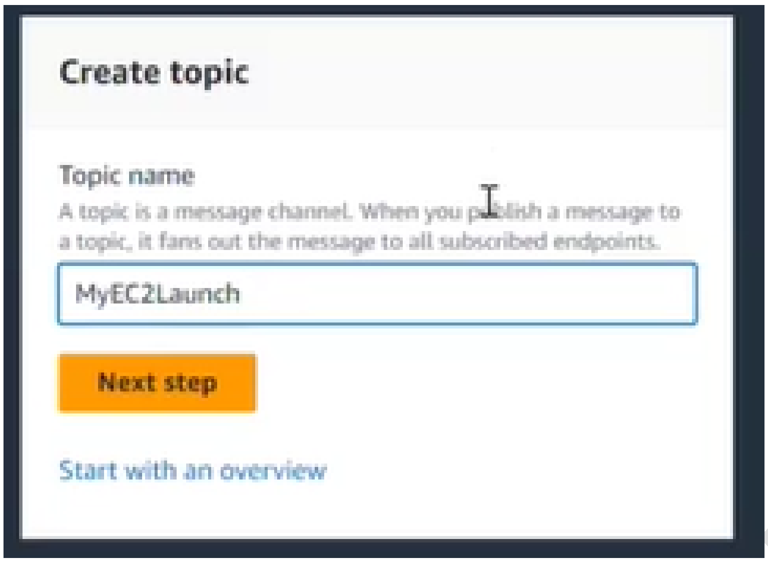
Search results for 'sns'

Try searching with longer queries for more relevant results

Simple Notification Service

SNS managed message topics for Pub/Sub

- In the Create topic box, enter a topic name, and click Next Step



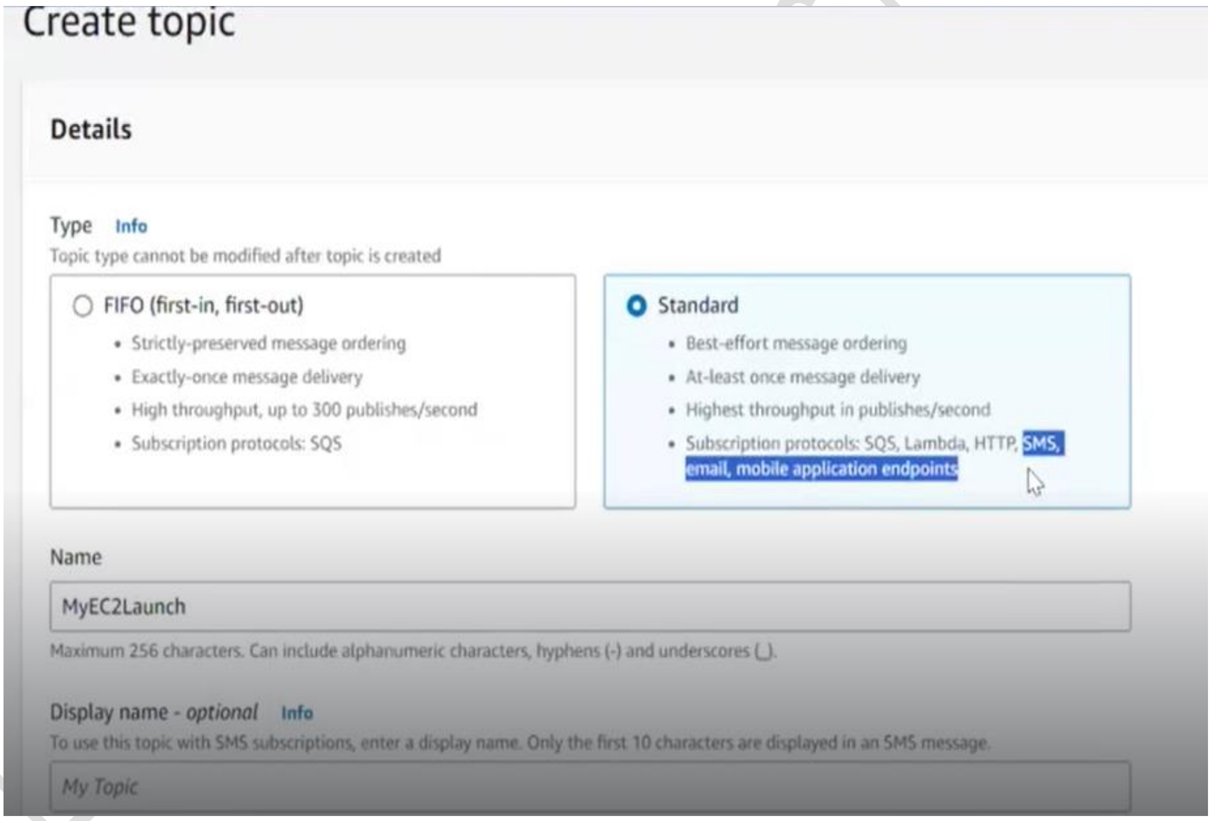
Create topic

Topic name

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

MyEC2Launch

- Click on the standard



Create topic

Details

Type Info

Topic type cannot be modified after topic is created

FIFO (first-in, first-out)
- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard
- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

MyEC2Launch

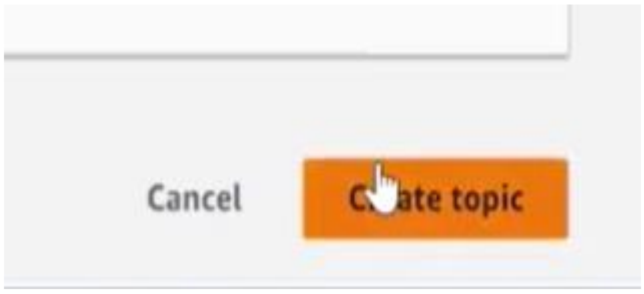
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* Info

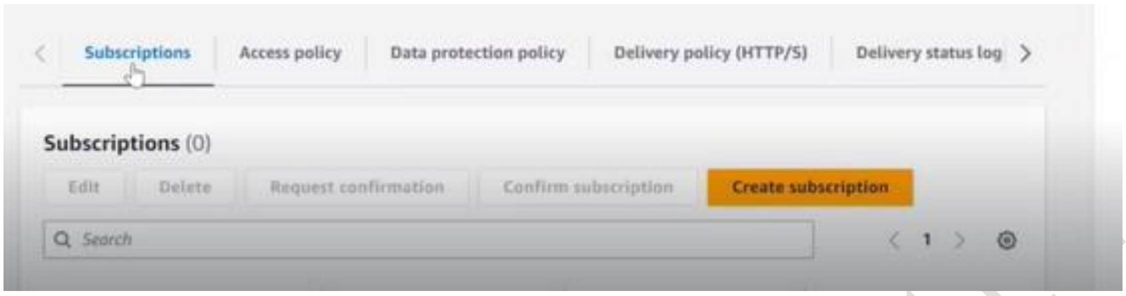
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

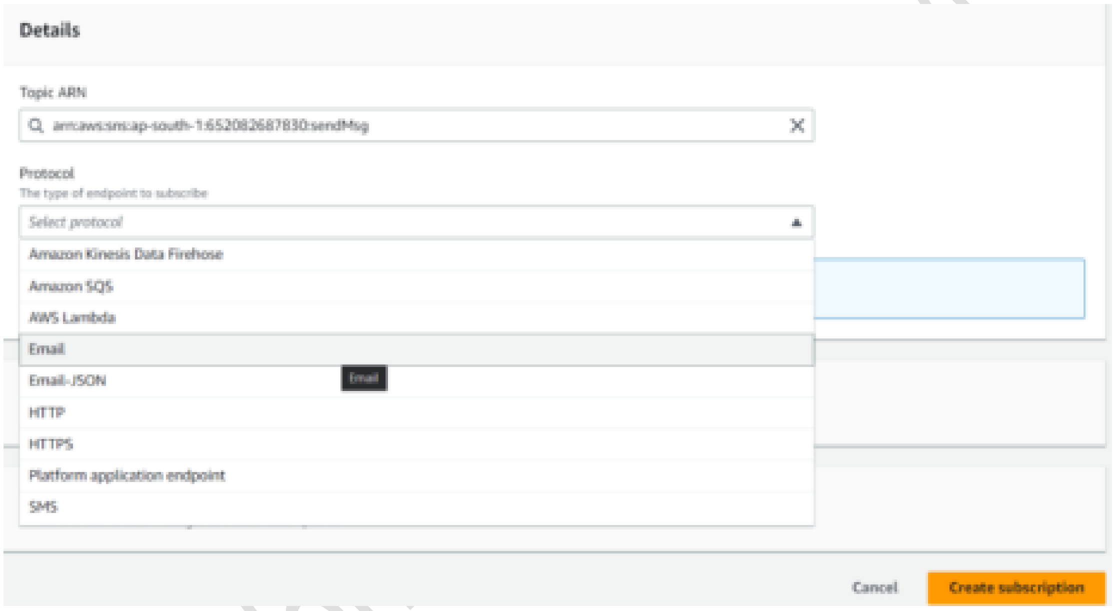
- Click Create a topic



- After that we need to create **subscriptions**
- A subscription in SNS means adding an endpoint where this notification will be sent. The endpoint could be some other AWS service, Email, SMS, etc.
- Scroll down to find the Subscriptions box, click Create Subscription



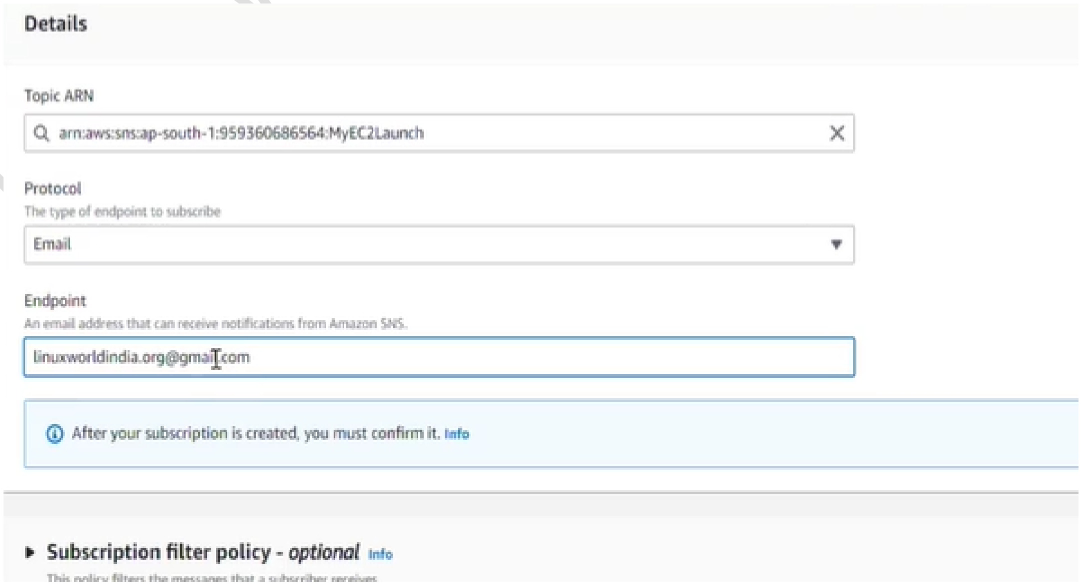
- Choose Email in the Protocol bar


Details

Topic ARN
arn:aws:sns:ap-south-1:652082687830:sendMsg

Protocol
The type of endpoint to subscribe
Select protocol

- Enter an email, where you would like to receive the email in the Endpoint bar


Details

Topic ARN
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch

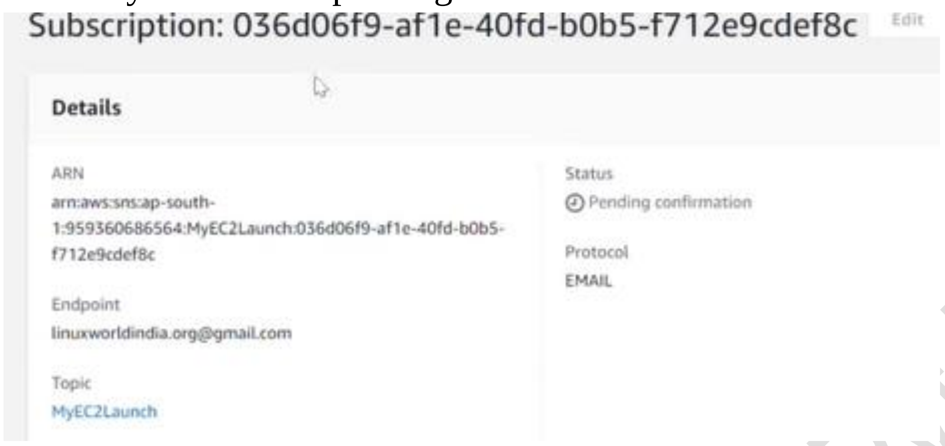
Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
linuxworldindia.org@gmail.com

After your subscription is created, you must confirm it. Info

Subscription filter policy - *optional* Info
This policy filters the messages that a subscriber receives.

- Click Create Subscription
- After that we need to confirm the email ID if you do not confirm the email ID then your status is pending



Subscription: 036d06f9-af1e-40fd-b0b5-f712e9cdef8c

Details

ARN
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch:036d06f9-af1e-40fd-b0b5-f712e9cdef8c

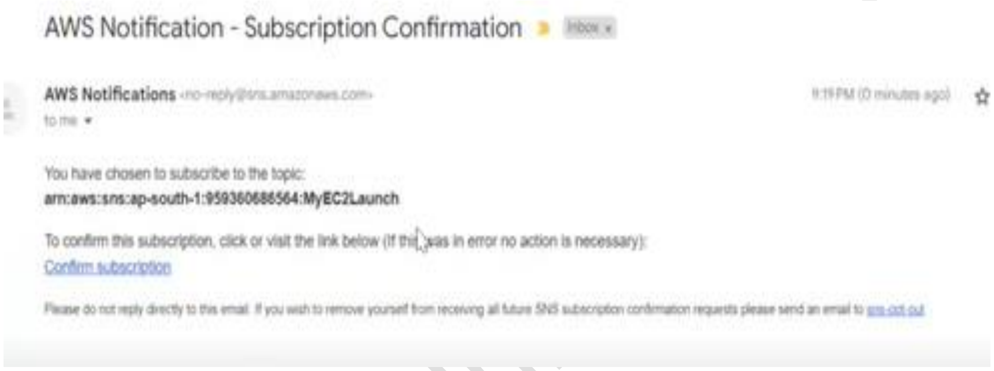
Endpoint
linuxworldindia.org@gmail.com

Topic
MyEC2Launch

Status
Pending confirmation

Protocol
EMAIL

- For that we need to go to the email ID and confirm



AWS Notification - Subscription Confirmation

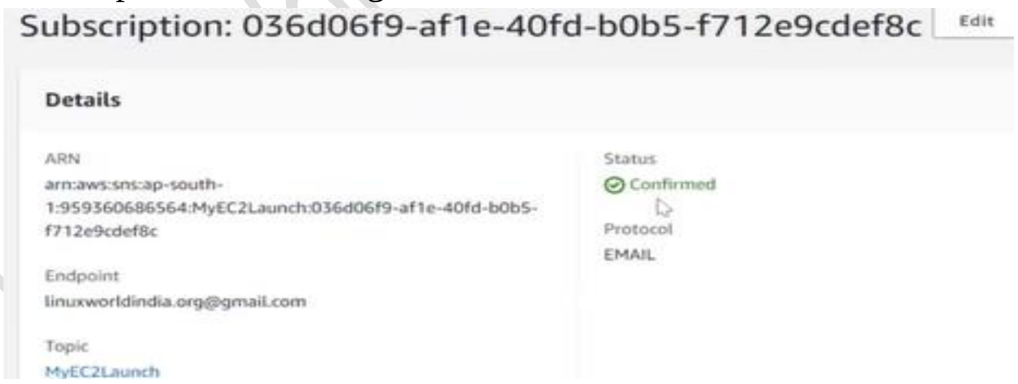
AWS Notifications <no-reply@sns.amazonaws.com>
to me
9:19 PM (0 minutes ago)

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch

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

- Go to your Gmail account to find the Subscription Confirmation email. Click Confirm subscription
- Subscription status changes to Confirmed



Subscription: 036d06f9-af1e-40fd-b0b5-f712e9cdef8c

Details

ARN
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch:036d06f9-af1e-40fd-b0b5-f712e9cdef8c

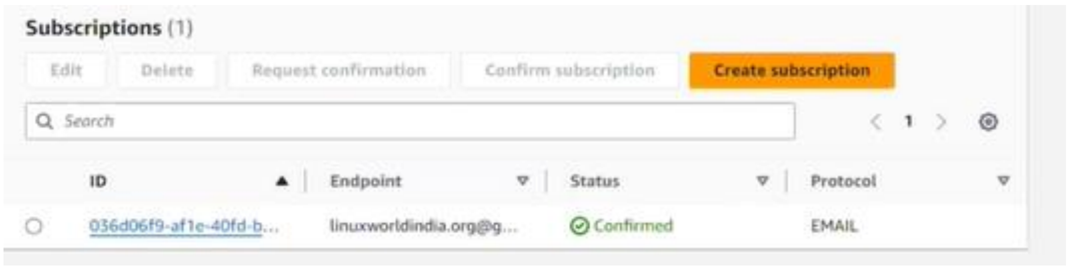
Endpoint
linuxworldindia.org@gmail.com

Topic
MyEC2Launch

Status
Confirmed

Protocol
EMAIL

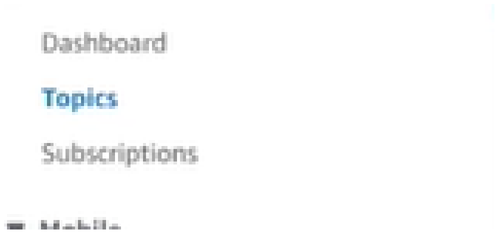
- As we can see we added a mail ID in the Topic



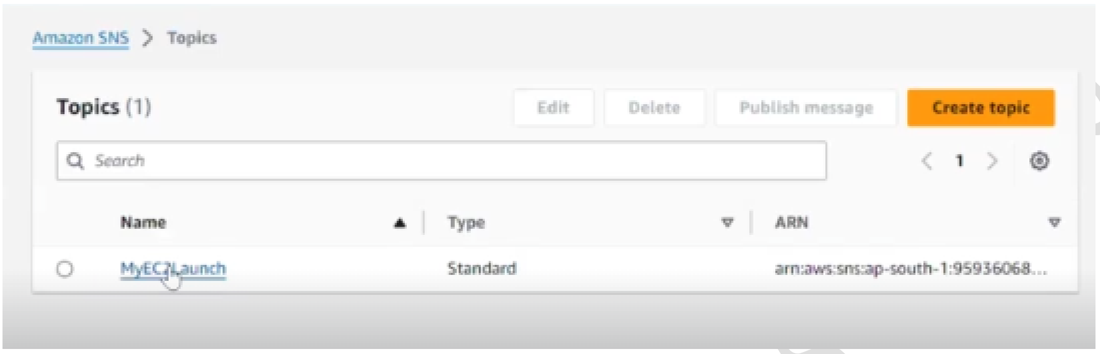
Subscriptions (1)

	ID	Endpoint	Status	Protocol
○	036d06f9-af1e-40fd-b...	linuxworldindia.org@g...	Confirmed	EMAIL

- Your topic is working or not for that click on the topic

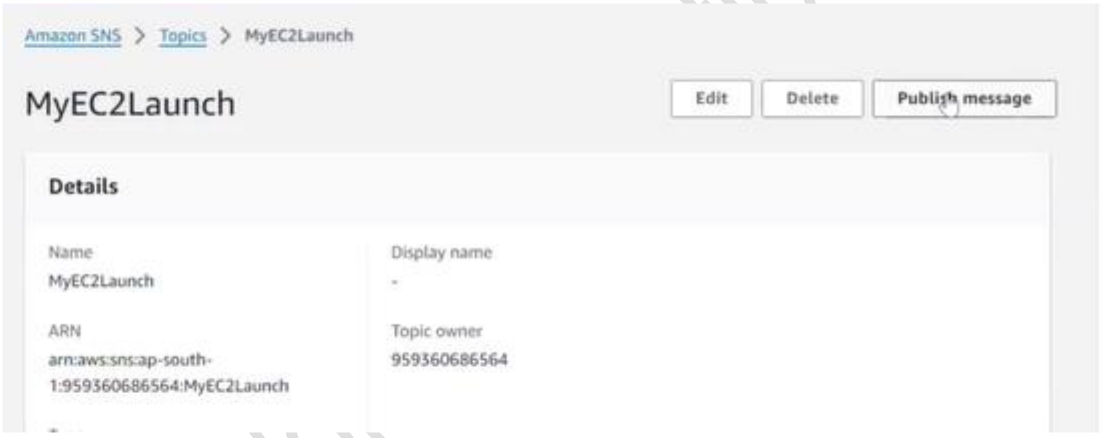


- Then click on the topic name



Name	Type	ARN
MyEC2Launch	Standard	arn:aws:sns:ap-south-1:95936068...

- To check the working, click **Publish message**



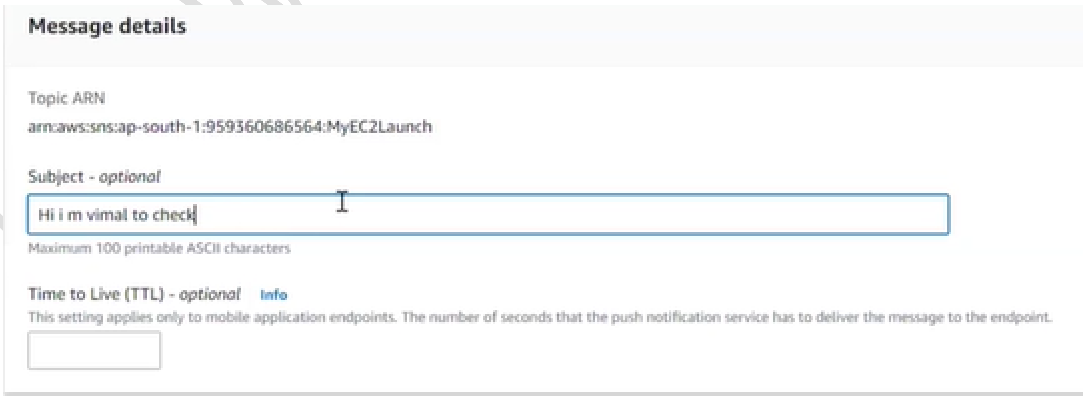
Name
MyEC2Launch

Display name
-

ARN
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch

Topic owner
959360686564

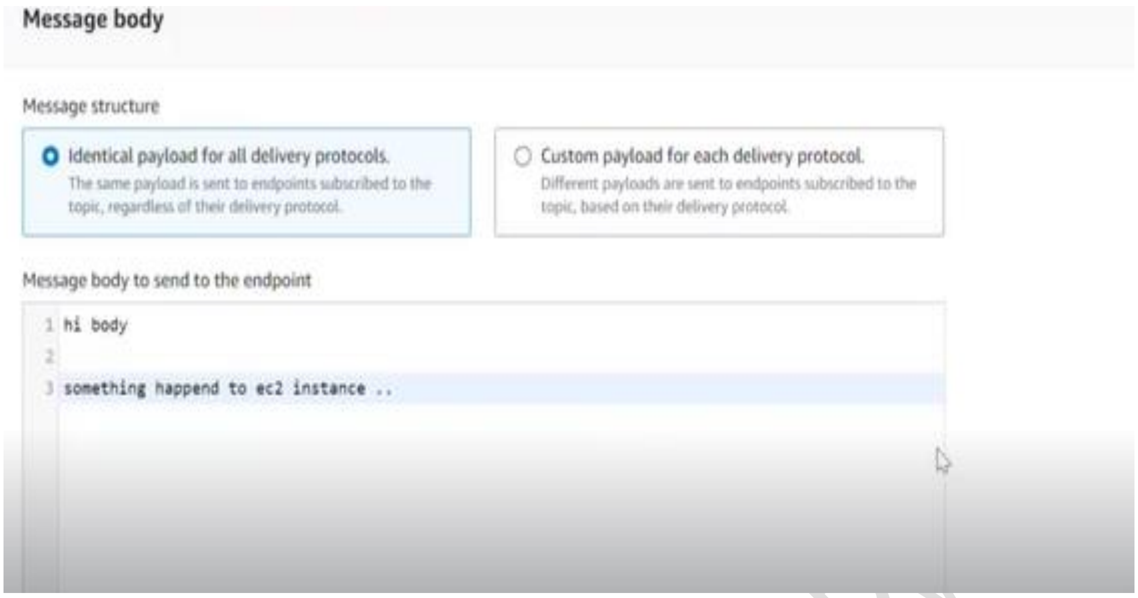
- Add a Subject and **Message body** > **Publish message**



Topic ARN
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch

Subject - *optional*
Hi i m vimal to check
Maximum 100 printable ASCII characters

Time to Live (TTL) - *optional* Info
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.



Message body

Message structure

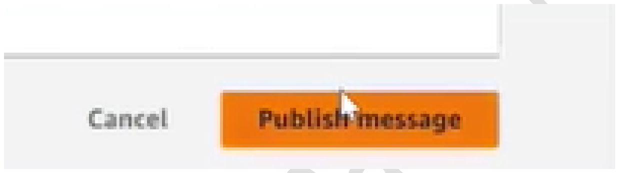
Identical payload for all delivery protocols.
The same payload is sent to endpoints subscribed to the topic, regardless of their delivery protocol.

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

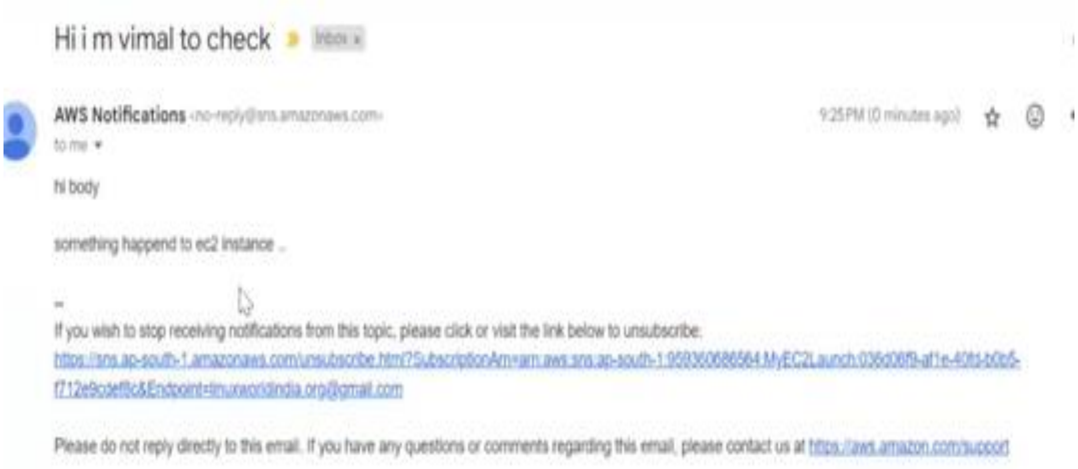
Message body to send to the endpoint

```
hi body

something happend to ec2 instance ..
```

- After that click on the **Publish message**



- You will receive an email from SNS



Hi i m vimal to check

AWS Notifications <no-reply@sns.amazonaws.com>
to me
9:25 PM (0 minutes ago)

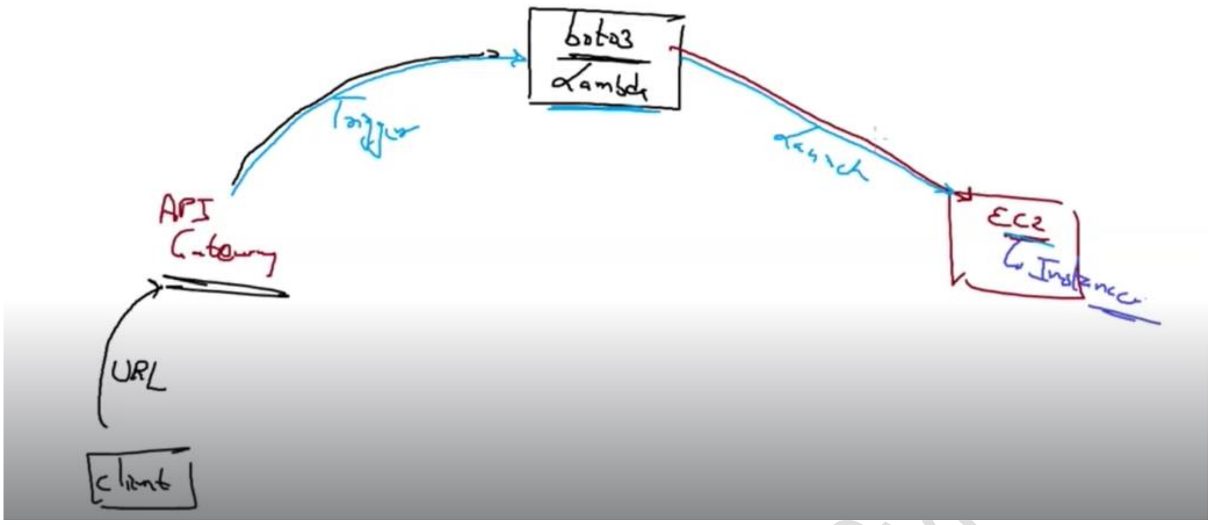
hi body

something happend to ec2 instance ..

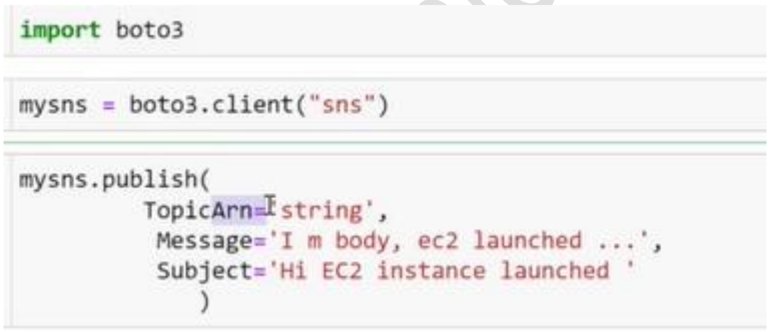
--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:959360686564:MyEC2Launch:036d06f9-af1e-40fd-b0b5-f712e9cdeffc&Endpoint=lmuxworldindia.org@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

- But in the above example we need to click always publish the message
- In the real world many notifications we have automated the SNS service
- In the last class we create one code in the lambda with the help of this code we launch the Ec2 instance for that we use the Boto3 python library
- Whenever you run the lambda then automatically instance launch
- Also we integrate with the API gateway



- This project we create in the last class
- Now we need to notify the project, which means whenever anyone hits the ULR one notification comes into my email
- For that we use the SNS service
- We create a code from lambda to publish the topic
- As we can see we create below code for publishing the topic
- First we import the boto3 then we create a one-variable after that we need the information of the topic for that we need to give the below information

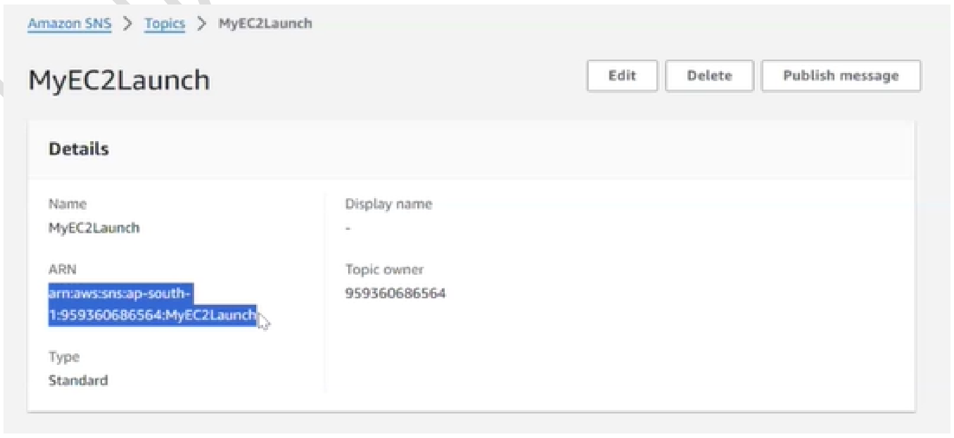


```
import boto3
```

```
mysns = boto3.client("sns")
```

```
mysns.publish(
          TopicArn='string',
           Message='I m body, ec2 launched ...',
           Subject='Hi EC2 instance launched '
              )
```

- But the important part is TopicArn

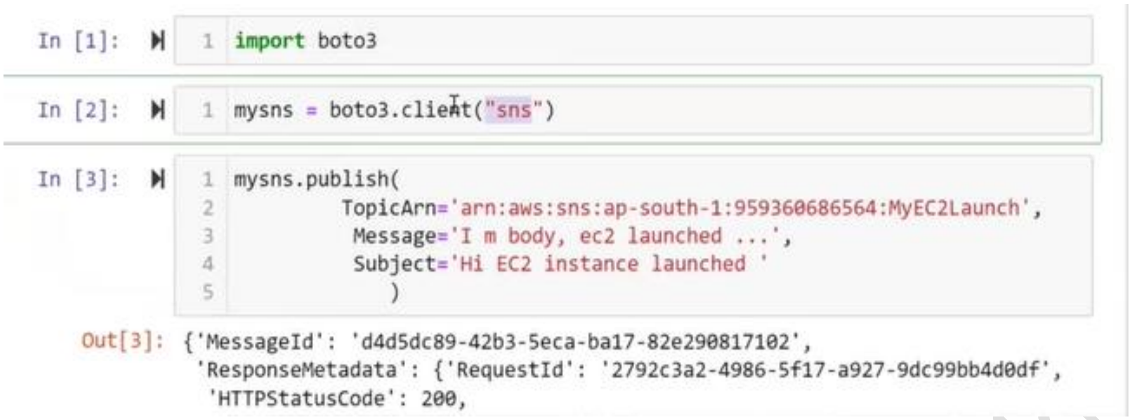


MyEC2Launch

Details

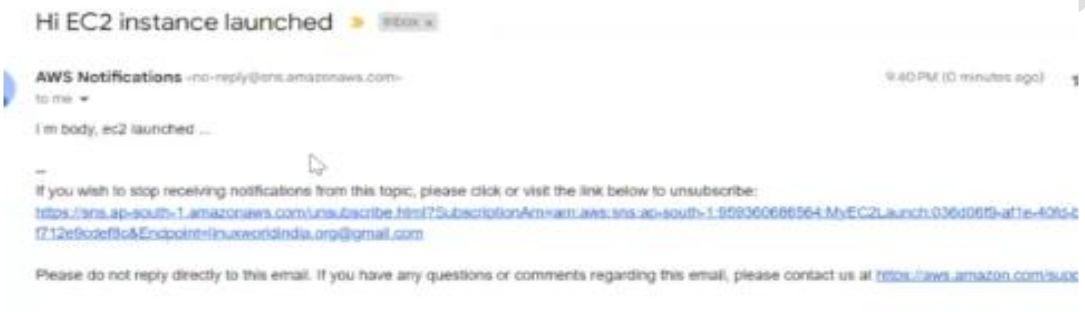
Name	Display name
MyEC2Launch	-
ARN	Topic owner
arn:aws:sns:ap-south-1:959360686564:MyEC2Launch	959360686564
Type	
Standard	

- Arn gives the information on which topic belongs to the account



```
In [1]:  import boto3

In [2]:  mysns = boto3.client("sns")

In [3]:  mysns.publish(
                    TopicArn='arn:aws:sns:ap-south-1:959360686564:MyEC2Launch',
                     Message='I m body, ec2 launched ...',
                     Subject='Hi EC2 instance launched '
                        )

Out[3]: {'MessageId': 'd4d5dc89-42b3-5eca-ba17-82e290817102',
          'ResponseMetadata': {'RequestId': '2792c3a2-4986-5f17-a927-9dc99bb4d0df',
           'HTTPStatusCode': 200,
```

- After running this code we get the email



Hi EC2 instance launched

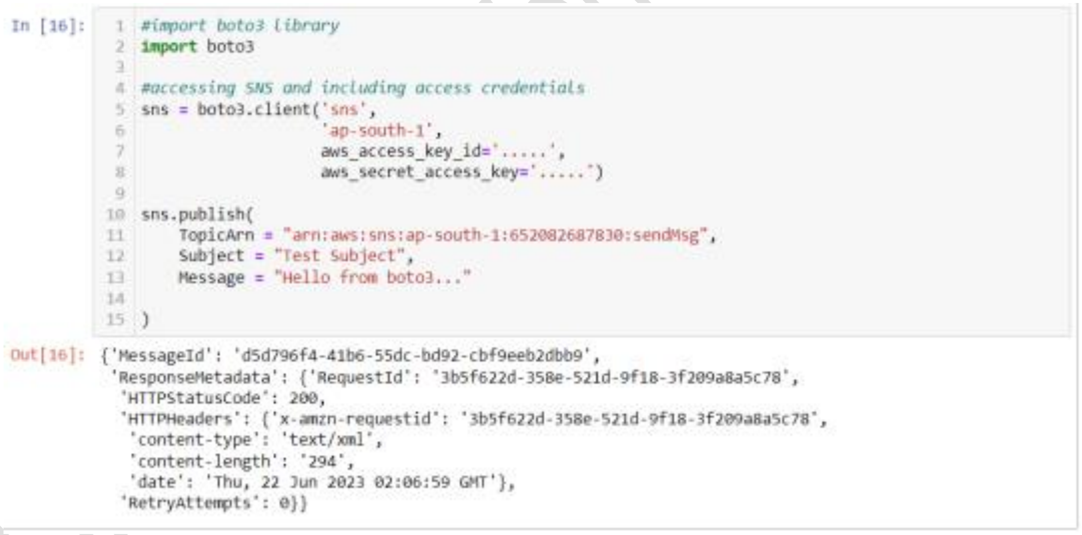
AWS Notifications <no-reply@sns.amazonaws.com>
to me
9:40 PM (0 minutes ago)

I m body, ec2 launched ...

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:959360686564:MyEC2Launch:036d06f9-af1e-40fd-b
f712e9cdef8c&Endpoint=linuxworldindia.org@gmail.com

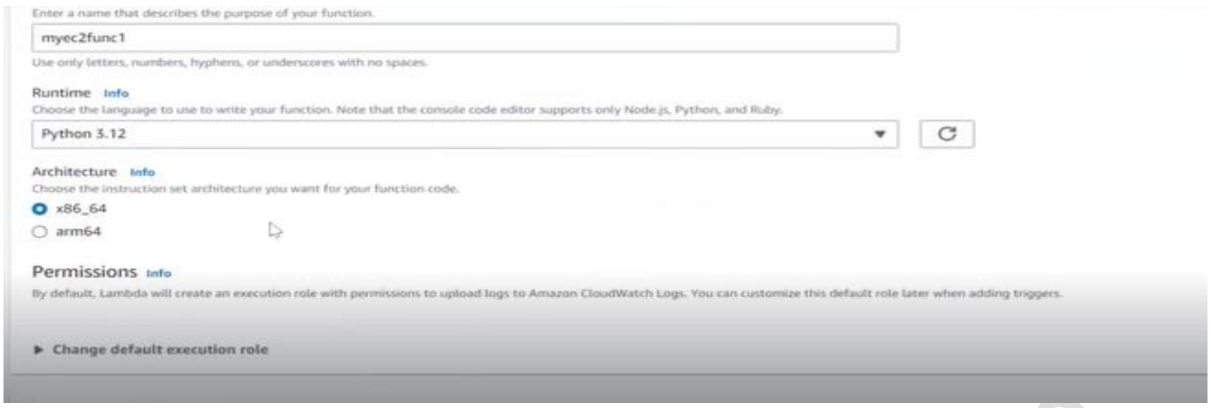
Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/supp

- Note:- A secrete key is important to store in the local system
- or you add the secrete in the code also



```
In [16]: #import boto3 library
         import boto3

         #accessing SNS and including access credentials
         sns = boto3.client('sns',
                            'ap-south-1',
                            aws_access_key_id='.....',
                            aws_secret_access_key='.....')

         sns.publish(
             TopicArn = "arn:aws:sns:ap-south-1:652082687830:sendMsg",
             Subject = "Test Subject",
             Message = "Hello from boto3..."

         )

Out[16]: {'MessageId': 'd5d796f4-41b6-55dc-bd92-cbf9eeb2dbb9',
          'ResponseMetadata': {'RequestId': '3b5f622d-358e-521d-9f18-3f209a8a5c78',
           'HTTPStatusCode': 200,
           'HTTPHeaders': {'x-amzn-requestid': '3b5f622d-358e-521d-9f18-3f209a8a5c78',
            'content-type': 'text/xml',
            'content-length': '294',
            'date': 'Thu, 22 Jun 2023 02:06:59 GMT'},
           'RetryAttempts': 0}}
```

- after that create one lambda function



Enter a name that describes the purpose of your function.

myec2func1

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.12

Architecture Info

Choose the instruction set architecture you want for your function code.

[✓ x86_64]

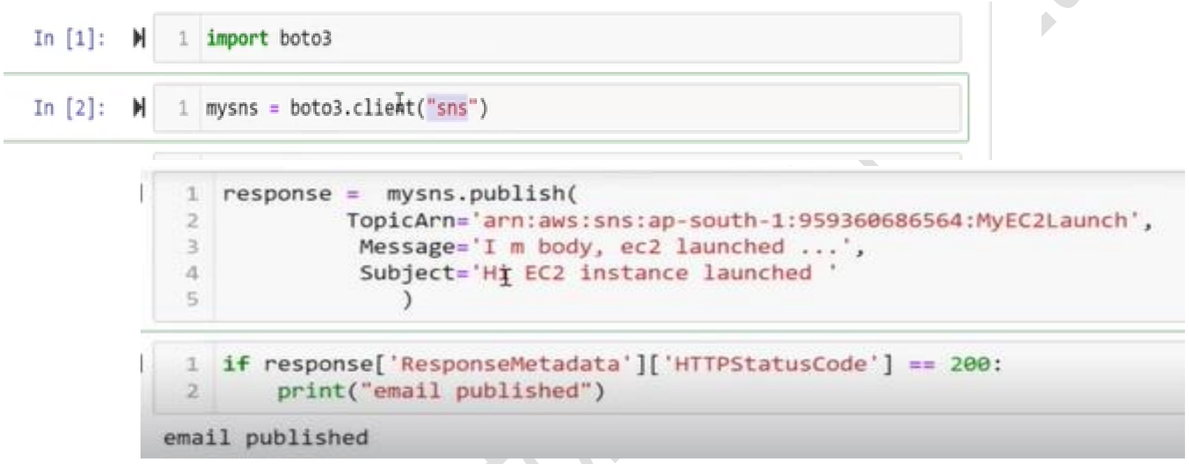
arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

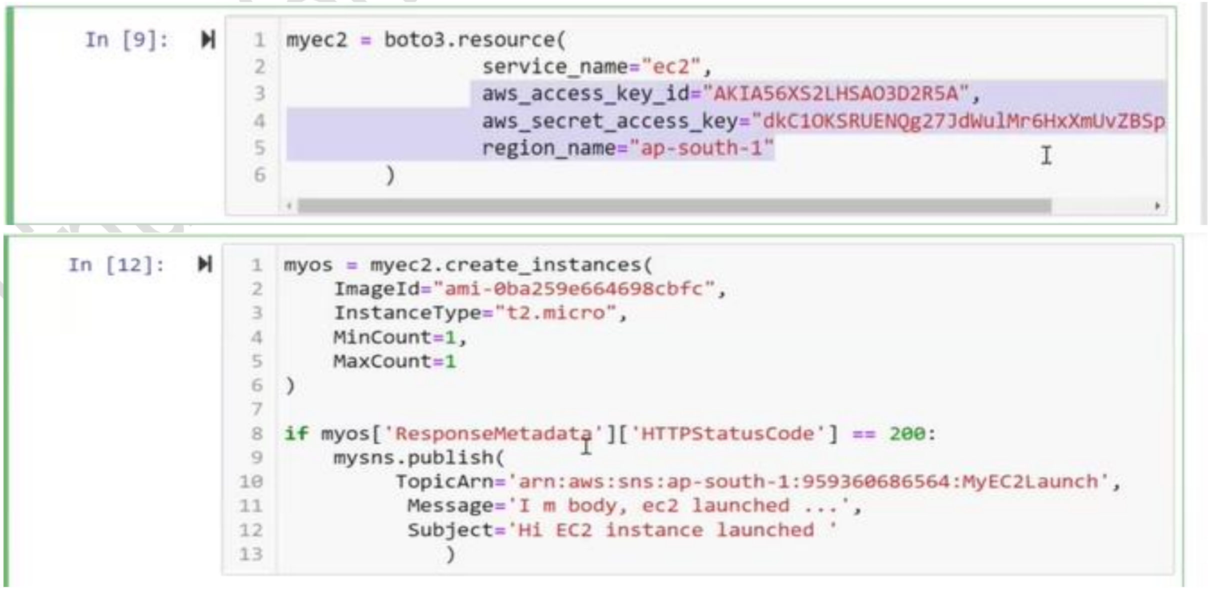
▶ Change default execution role

- after that we create one condition for the publish the message



```
In [1]:  import boto3

In [2]:  mysns = boto3.client("sns")

response =  mysns.publish(
         TopicArn='arn:aws:sns:ap-south-1:959360686564:MyEC2Launch',
          Message='I m body, ec2 launched ...',
          Subject='Hi EC2 instance launched '
            )

if response['ResponseMetadata']['HTTPStatusCode'] == 200:
    print("email published")

email published
```

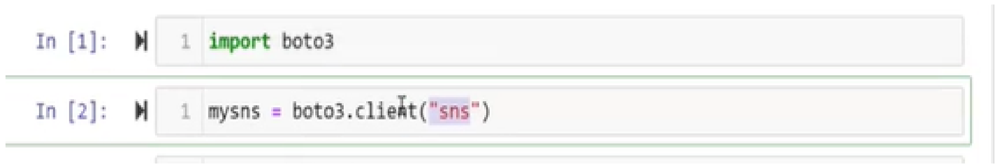
- Note:- 200 means your code run successfully
- Now we write the condition when 200 comes then publish the message
- After that we write the code for the instance when running this code launch the instance automatically



```
In [9]:  myec2 = boto3.resource(
                         service_name="ec2",
                         aws_access_key_id="AKIA56XS2LHSAO3D2R5A",
                         aws_secret_access_key="dkC1OKSRUENQg27JdWulMr6HxXmUvZBSp
                         region_name="ap-south-1"
             )

In [12]:  myos = myec2.create_instances(
              ImageId="ami-0ba259e664698cbfc",
              InstanceType="t2.micro",
              MinCount=1,
              MaxCount=1
          )

          if myos['ResponseMetadata']['HTTPStatusCode'] == 200:
              mysns.publish(
                   TopicArn='arn:aws:sns:ap-south-1:959360686564:MyEC2Launch',
                    Message='I m body, ec2 launched ...',
                    Subject='Hi EC2 instance launched '
                       )
```

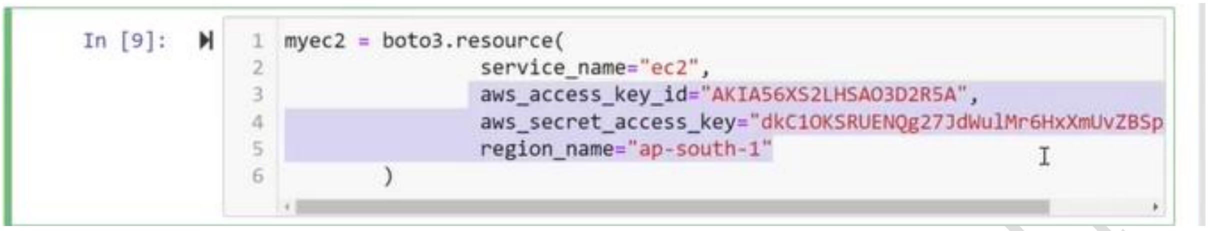
- When your instance launches successfully then this above code sends the message to the email that gives the notification

- Final code is below for the SNS and Ec2 instance

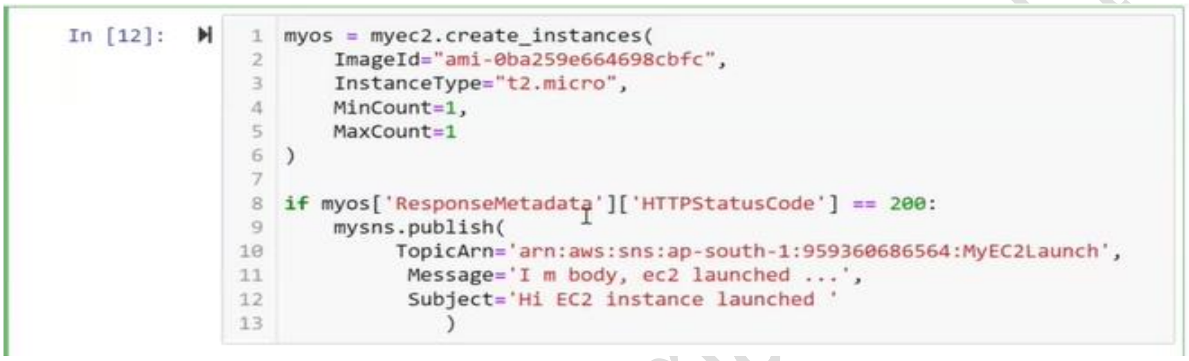


```
In [1]:  import boto3
```

```
In [2]:  mysns = boto3.client("sns")
```

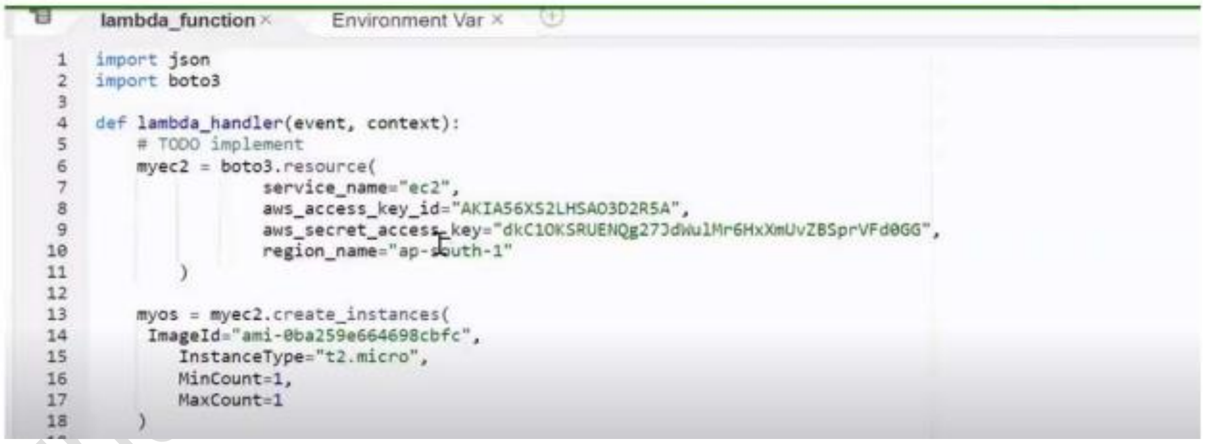


```
In [9]:  myec2 = boto3.resource(
                     service_name="ec2",
                     aws_access_key_id="AKIA56XS2LHSAO3D2R5A",
                     aws_secret_access_key="dkC1OKSRUENQg27JdWulMr6HxXmUvZBSp
                     region_name="ap-south-1"
             )
```

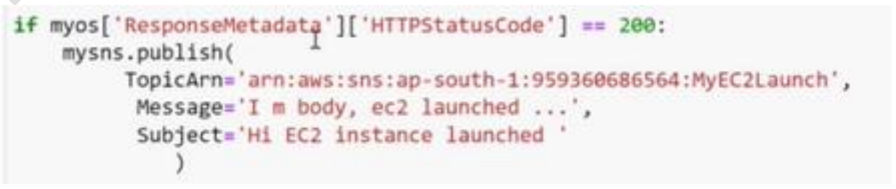


```
In [12]:  myos = myec2.create_instances(
              ImageId="ami-0ba259e664698cbfc",
              InstanceType="t2.micro",
              MinCount=1,
              MaxCount=1
          )

          if myos['ResponseMetadata']['HTTPStatusCode'] == 200:
              mysns.publish(
                   TopicArn='arn:aws:sns:ap-south-1:959360686564:MyEC2Launch',
                    Message='I m body, ec2 launched ...',
                    Subject='Hi EC2 instance launched '
                       )
```

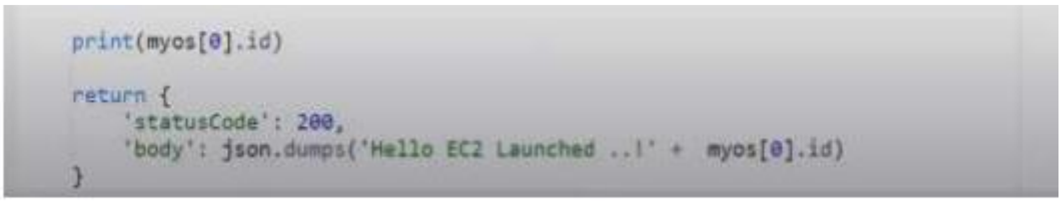
- After that put all the code in the Lambda service
- For example is given below



```
import json
import boto3

def lambda_handler(event, context):
    # TODO implement
    myec2 = boto3.resource(
                    service_name="ec2",
                    aws_access_key_id="AKIA56XS2LHSAO3D2R5A",
                    aws_secret_access_key="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",
                    region_name="ap-south-1"
        )

    myos = myec2.create_instances(
     ImageId="ami-0ba259e664698cbfc",
        InstanceType="t2.micro",
        MinCount=1,
        MaxCount=1
    )
```



```
if myos['ResponseMetadata']['HTTPStatusCode'] == 200:
    mysns.publish(
         TopicArn='arn:aws:sns:ap-south-1:959360686564:MyEC2Launch',
          Message='I m body, ec2 launched ...',
          Subject='Hi EC2 instance launched '
             )
```



```
print(myos[0].id)

return {
    'statusCode': 200,
    'body': json.dumps('Hello EC2 Launched ..!' +  myos[0].id)
}
```