Case Study :-

Automated Notifications Using SNS

- Concepts Used: AWS Lambda, S3, SNS.
- Problem Statement: "Create a Lambda function that triggers when a new file is uploaded to an S3 bucket and sends an email notification using SNS with details of the
uploaded file."
- Tasks:
  - Write a Python Lambda function that triggers on S3 upload events.
  - Extract the file name and size from the event and format a notification message.
  - Use SNS to send the notification to a configured email address.
  - Test by uploading a file to the S3 bucket and verifying that an email is received.

1. Introduction :-

Case Study Overview:
This case study focuses on automating notifications for file uploads using AWS services. The system is designed to notify users when a new file is uploaded to an S3 bucket by leveraging AWS Lambda and Simple Notification Service (SNS). This automation ensures timely communication of important file uploads without manual intervention.

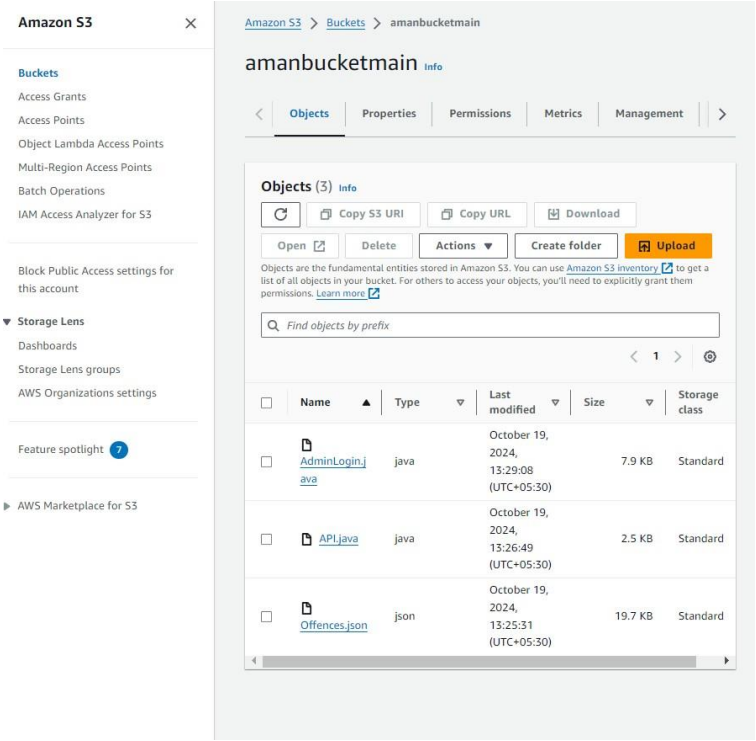
Key Feature and Application:
The key feature of this system is the use of an AWS Lambda function that triggers automatically whenever a file is uploaded to the S3 bucket. It extracts details such as the file name and size, formats them into a message, and sends an email notification using AWS SNS. This application is useful for scenarios requiring real-time alerts, such as file monitoring or data processing pipelines.

Third-Year Project Integration (Optional):
In our third-year project, we implemented a similar feature where email notifications were sent to vaccination centers when a child was registered for a vaccine. This aligns with the current case study, where notifications are automated using AWS services. In both cases, automated notifications play a key role in ensuring timely updates to relevant parties, whether it’s notifying the vaccination center about a new registration or alerting users about file uploads in the S3 bucket. This demonstrates the practical application of event-driven notifications in different domains, enhancing communication and operational efficiency.

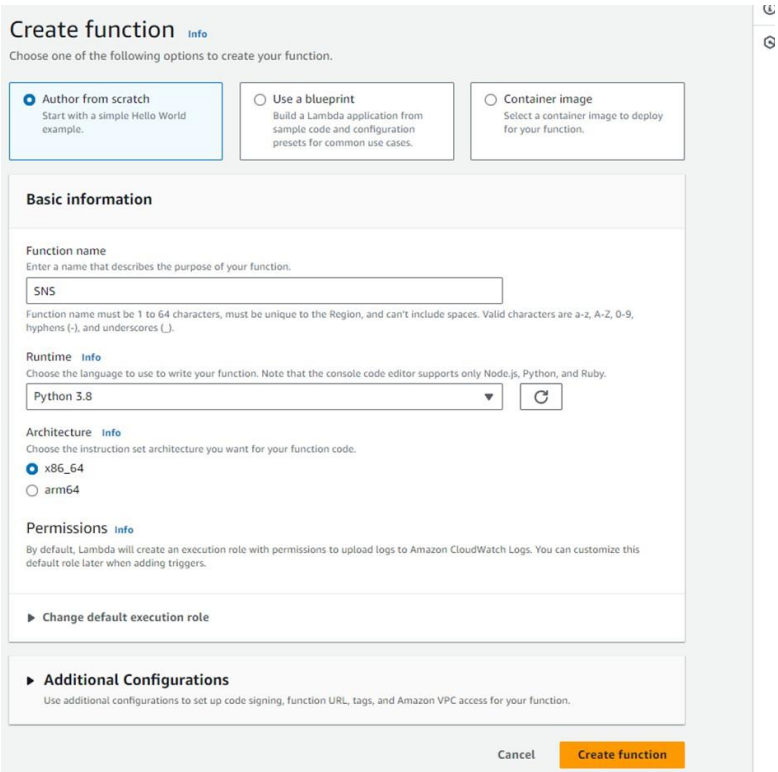
Name : Aman Yadav Class :- D15C Roll No :- 60

2. Step-by-Step Explanation :-

Step 1 :- Create a s3 Bucket

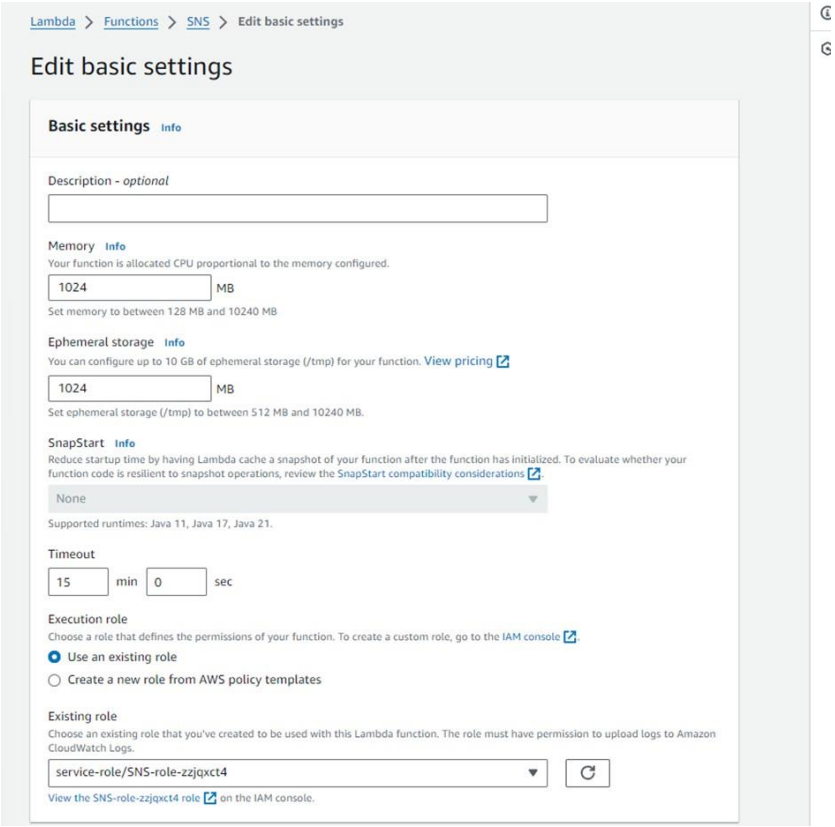


Step 2:- Create a lambda function :-

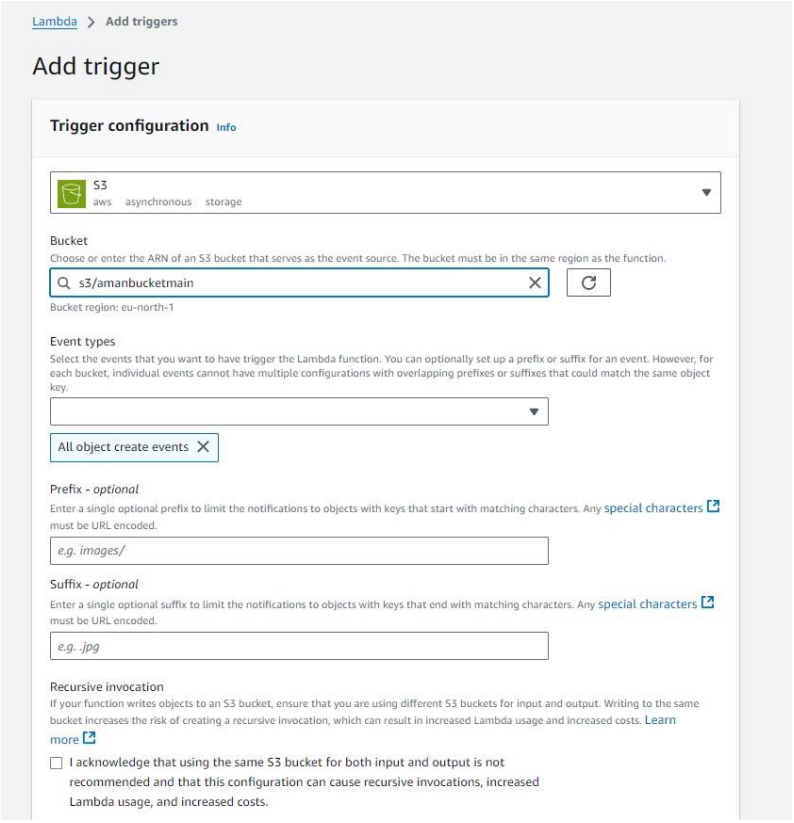


Name : Aman Yadav Class :- D15C Roll No :- 60

Step 3:- Go to the sns lambda function -> configuration



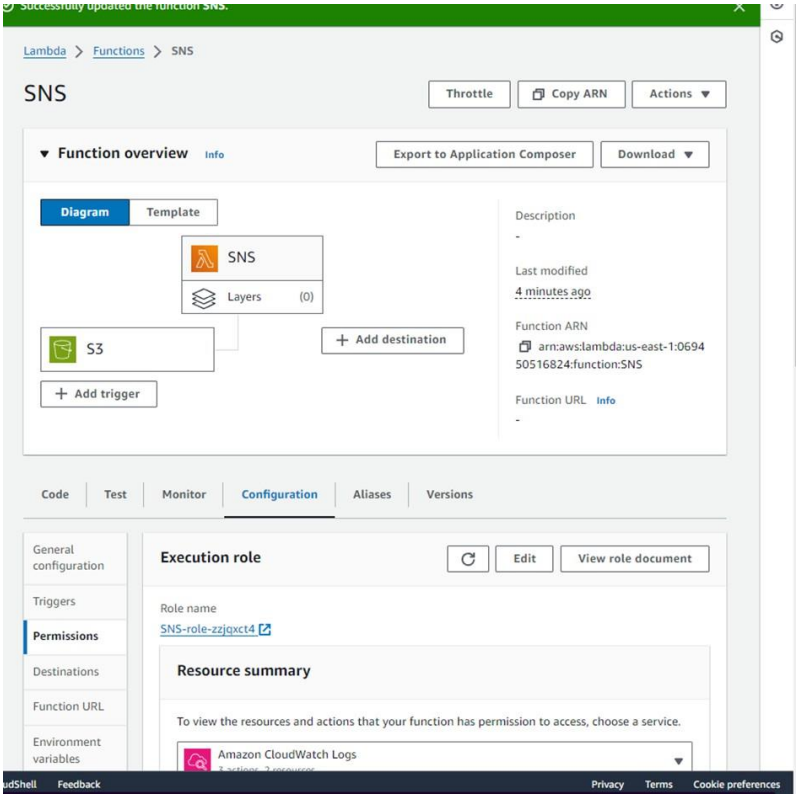
Step 4: Add the trigger :-



Name : Aman Yadav Class :- D15C Roll No :- 60

Step 5 :- After Configuration your Code look

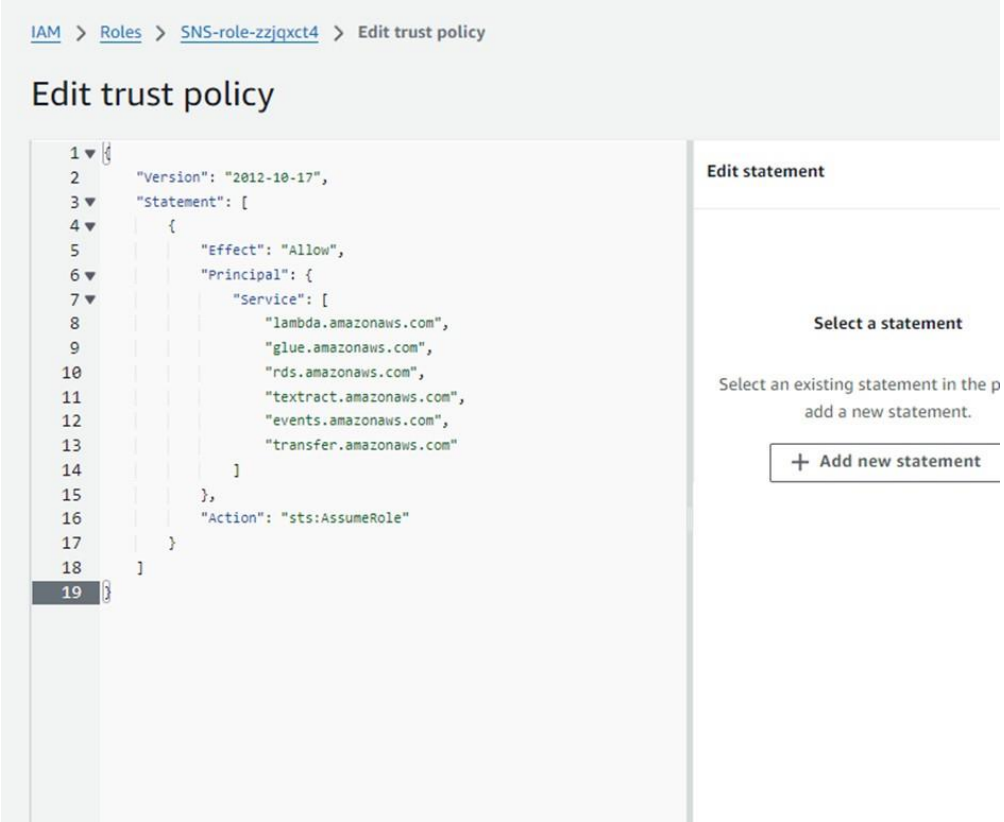
Step 6 :- Go the permission



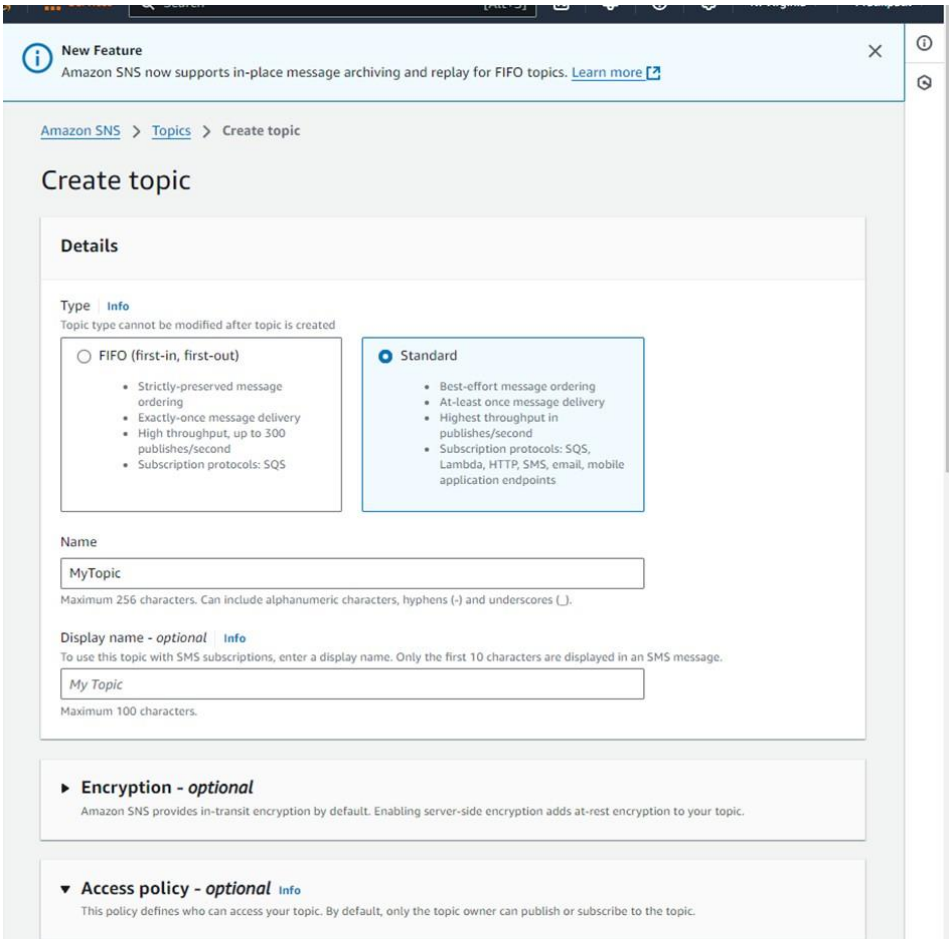
Step 7:- when you click on this link role name you will see the trust relationships do the changes in services

```
"Service": [
                "lambda.amazonaws.com",
                "glue.amazonaws.com",
                "rds.amazonaws.com",
                "textract.amazonaws.com",
                "events.amazonaws.com",
                "transfer.amazonaws.com"
            ]
```

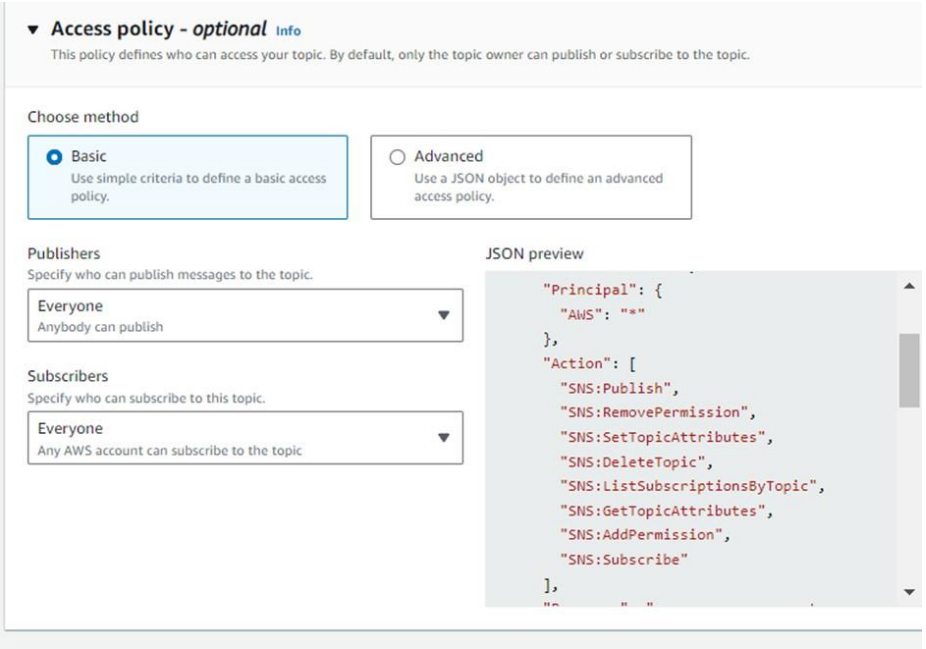
Step 8:- Save those changes



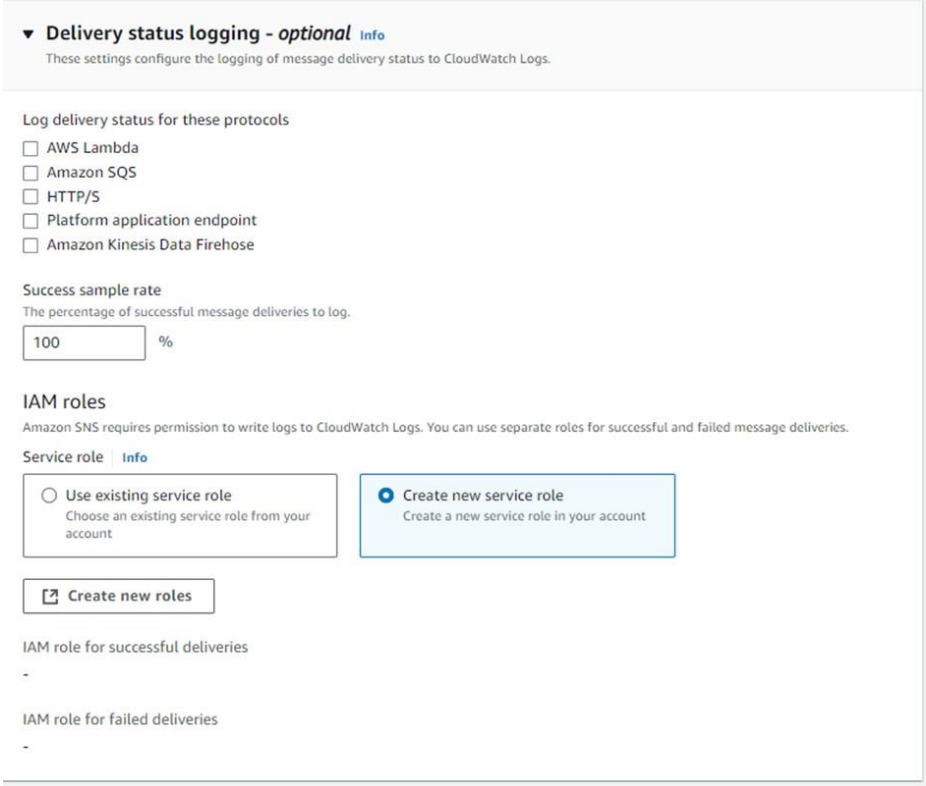
Step 9:- Now go the Amazon SNS create topic

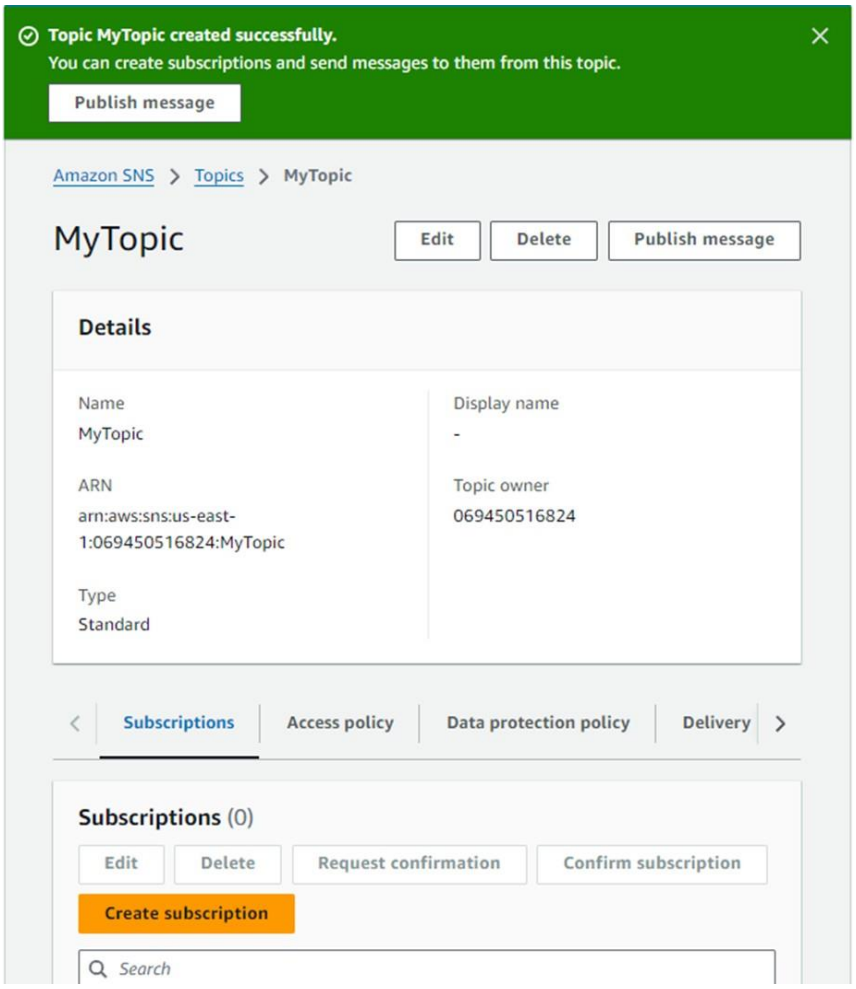


Step 10 :- Now edit access policy and delivery status logging



Name : Aman Yadav Class :- D15C Roll No :- 60





Step 11 :- Go to subscription in sns choose the topic then copy that topic arn paste in code in lambda function

```
import boto3

topic_arn = "arn:aws:sns:us-east-1:069450516824:PratikTopic"

def send_sns(message, subject):
    try:
        client = boto3.client("sns")
        result = client.publish(TopicArn=topic_arn, Message=message, Subject=subject)
        if result['ResponseMetadata']['HTTPStatusCode'] == 200:
            print(result)
            print("Notification sent successfully..!!!")
            return True
    except Exception as e:
        print("Error occurred while publishing notification: ", e)
        return False

def lambda_handler(event, context):
    print("Event collected: {}".format(event))

    for record in event['Records']:
        # Extract bucket name and key
        s3_bucket = record['s3']['bucket']['name']
        s3_key = record['s3']['object']['key']
        s3_size = record['s3']['object']['size']

        print("Bucket name: {}".format(s3_bucket))
        print("File key: {}".format(s3_key))
        print("File size: {} bytes".format(s3_size))

        from_path = "s3://{}/{}".format(s3_bucket, s3_key)

        # Construct message with file name and size
        message = "A file has been uploaded at {}\nFilename: {}\nFile size: {}
bytes".format(from_path, s3_key, s3_size)
        subject = "S3 File Upload Notification"

        # Send SNS notification
        SNSResult = send_sns(message, subject)
        if SNSResult:
            print("Notification Sent..")
            return SNSResult
        else:
            return False
```

Code | Test | Monitor | Configuration | Aliases | Versions

Code source Info

Upload from

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Go to Anything (Ctrl-P)

Environment

SNS - /

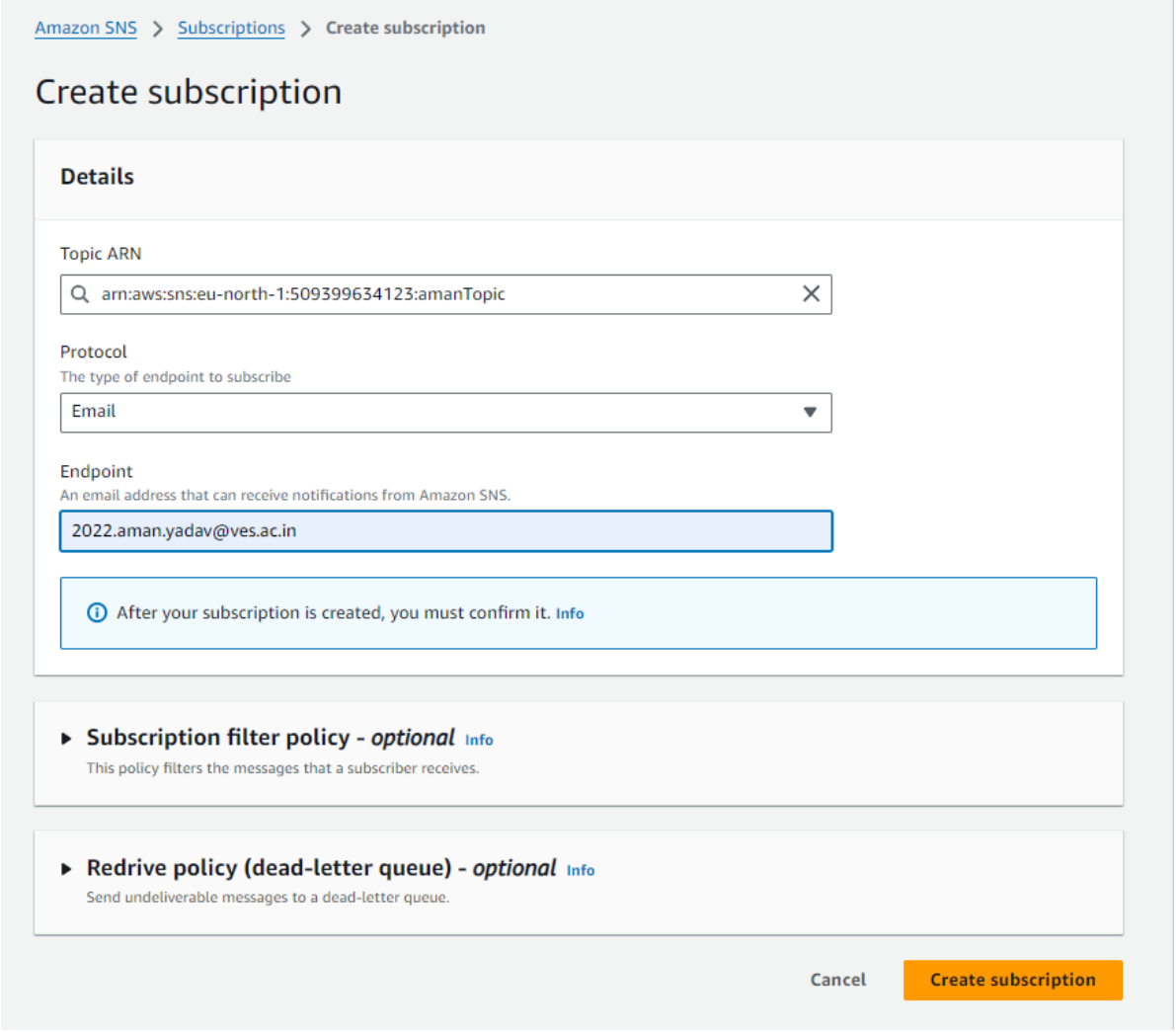
lambda_function.py

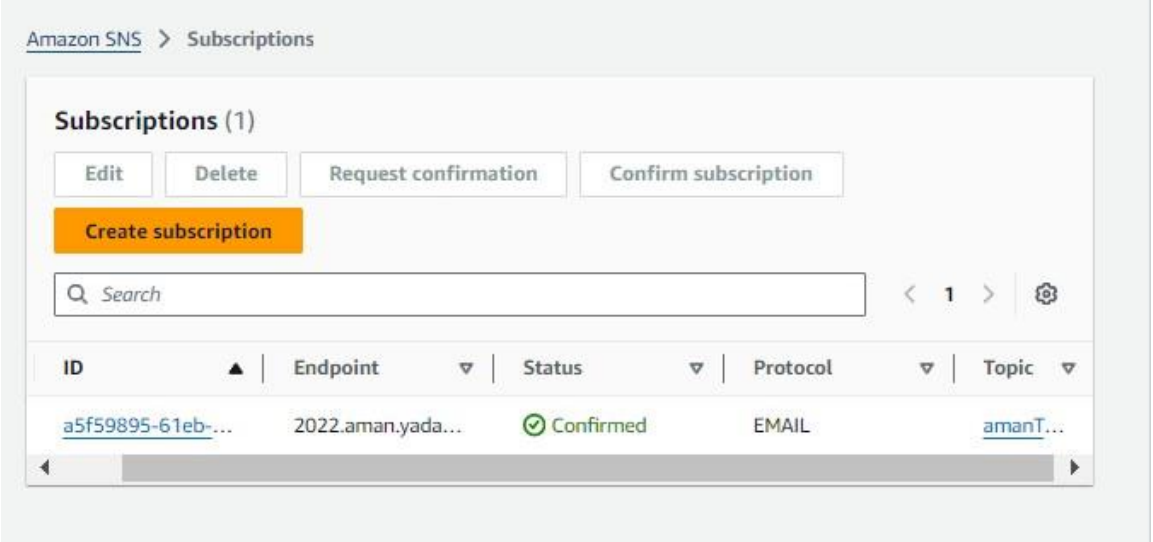
lambda_function × Environment Vari ×

```python
import boto3

topic_arn = "arn:aws:sns:us-east-1:069450516824:MyTopic2"
def send_sns(message, subject):
    try:
        client = boto3.client("sns")
        result = client.publish(TopicArn=topic_arn, Message=message, Subject=subject
        if result['ResponseMetadata']['HTTPStatusCode'] == 200:
            print(result)
            print("Notification send successfully..!!!")
            return True
    except Exception as e:
        print("Error occured while publish notifications and error is : ", e)
        return True

def lambda_handler(event, context):
    print("event collected is {}".format(event))
    for record in event['Records'] :
        s3_bucket = record['s3']['bucket']['name']
        print("Bucket name is {}".format(s3_bucket))
        s3_key = record['s3']['object']['key']
        print("Bucket key name is {}".format(s3_key))
        from_path = "s3://{}/{}".format(s3_bucket, s3_key)
        print("from path {}".format(from_path))
        message = "The file is uploaded at S3 bucket path {}".format(from_path)
        subject = "Processes completion Notification"
        SNSResult = send_sns(message, subject)
        if SNSResult :
            print("Notification Sent..")
            return SNSResult
        else:
            return False
```

Step 12 :- Now Create a Subscription :

Name : Aman Yadav Class :- D15C Roll No :- 60

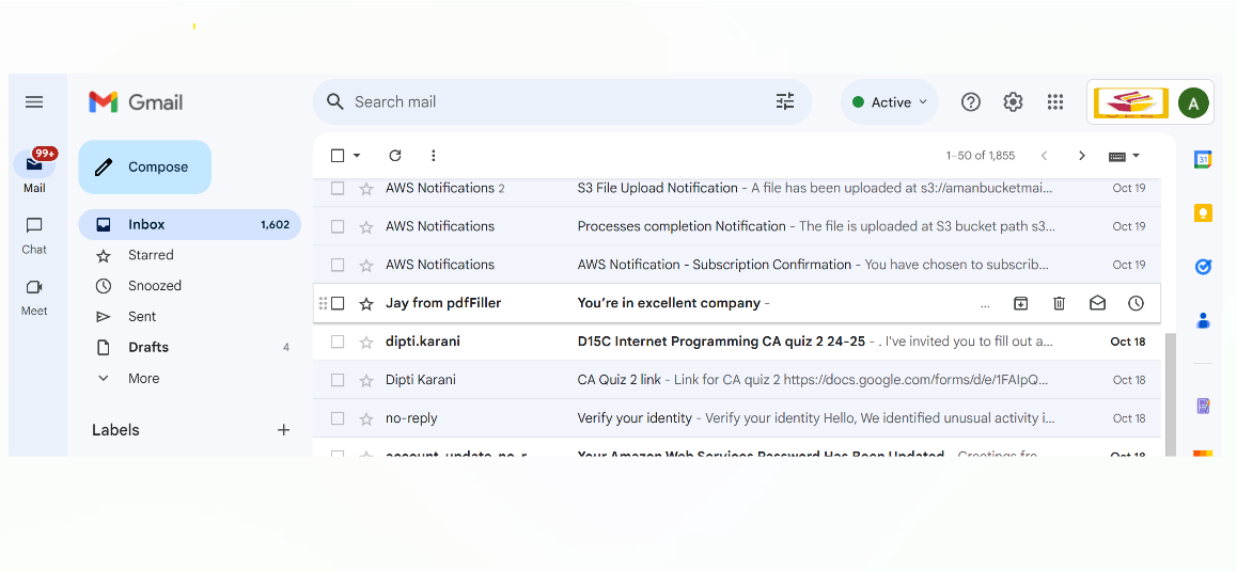




Name : Aman Yadav Class :- D15C Roll No :- 60

Step 13 :- Go to the email and verify it

Step 14 : - Upload files in S3 bucket You will get like this email



Conclusion

In conclusion, this case study showcases the power of AWS services, such as Lambda, S3, and SNS, to automate notifications in a highly efficient manner. By triggering events based on file uploads and delivering real-time alerts via email, the system ensures timely communication without manual intervention. This approach can be adapted to various use cases, including those involving data pipelines, file monitoring, or real-time system notifications. Furthermore, the integration with the third-year project highlights how similar event-driven mechanisms, such as sending email notifications to vaccination centers, can enhance practical applications across different fields, making operations smoother and more efficient.