

Practical Exam Case Study

Case Study No. 20: 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.

Introduction:

Case Study Overview-

In this case study, we explore the practical implementation of automated notifications using AWS services. The focus is on a system that uses AWS Lambda, S3, and SNS to notify users when a new file is uploaded to an S3 bucket. This scenario is common in various industries where real-time updates are crucial, such as data processing, monitoring systems, and collaborative environments.

- AWS Lambda: This serverless compute service runs code in response to events, such as an S3 file upload, without the need to provision or manage servers. It reduces operational overhead and enhances scalability.
- Amazon S3: Known for its durability, scalability, and security, S3 serves as the storage solution. The system triggers an event every time a new file is uploaded to a specified bucket.
- Amazon SNS: SNS (Simple Notification Service) is used for sending messages to users. It supports multiple protocols, but in this case, we use email to keep users updated on the file uploads.

Key Feature and Application-

Key Feature:

The standout feature of this system is its ability to automate notifications via email, leveraging the integration of AWS Lambda and SNS. The automation is triggered by events in an S3 bucket, which is a highly scalable object storage service used by developers and enterprises worldwide.

Application:

Practical Use: This system can be employed in multiple scenarios where timely notifications are essential. For instance, in a data processing pipeline, when a new data file is uploaded, stakeholders can be immediately informed. This ensures that the data processing begins promptly without manual intervention, thereby increasing efficiency.

Advantages:

- **Real-time Notifications:** Provides immediate awareness of new uploads, reducing delays.
- **Scalability:** The solution scales seamlessly with AWS infrastructure, handling varying loads without manual scaling.

Third-Year Project Integration:

In our Project 'E-yojana' we have the feature of sending the emails

- whenever the user applies for any scheme.
- When admin changes the status of the schemes applied.
- Upon successful signup

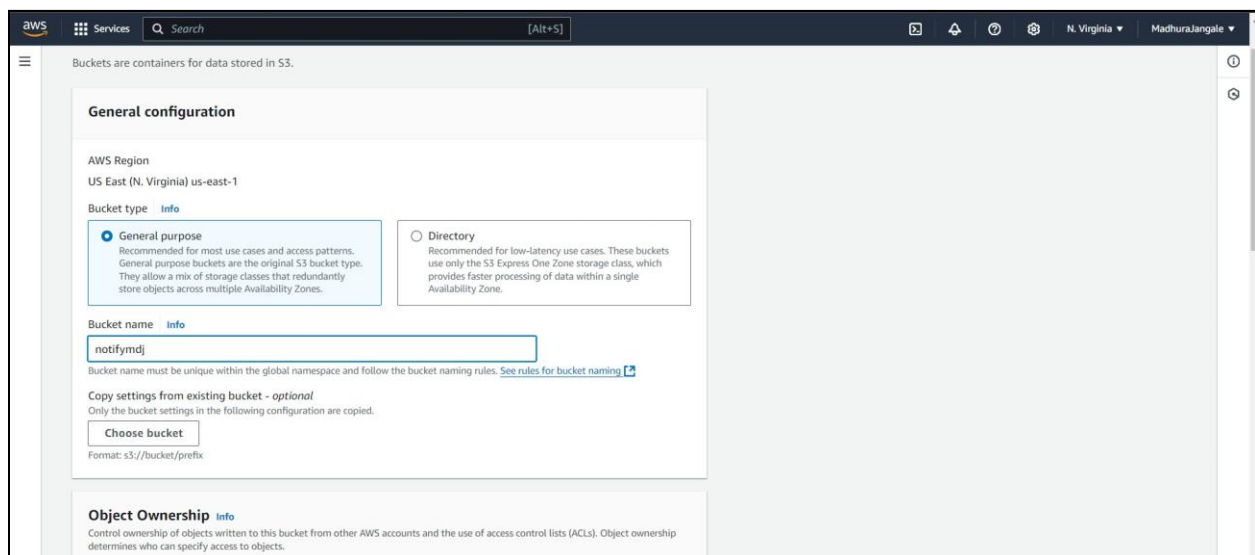
So, this case study is very relatable to our project wherein we can integrate this feature with this case study.

Implementation:

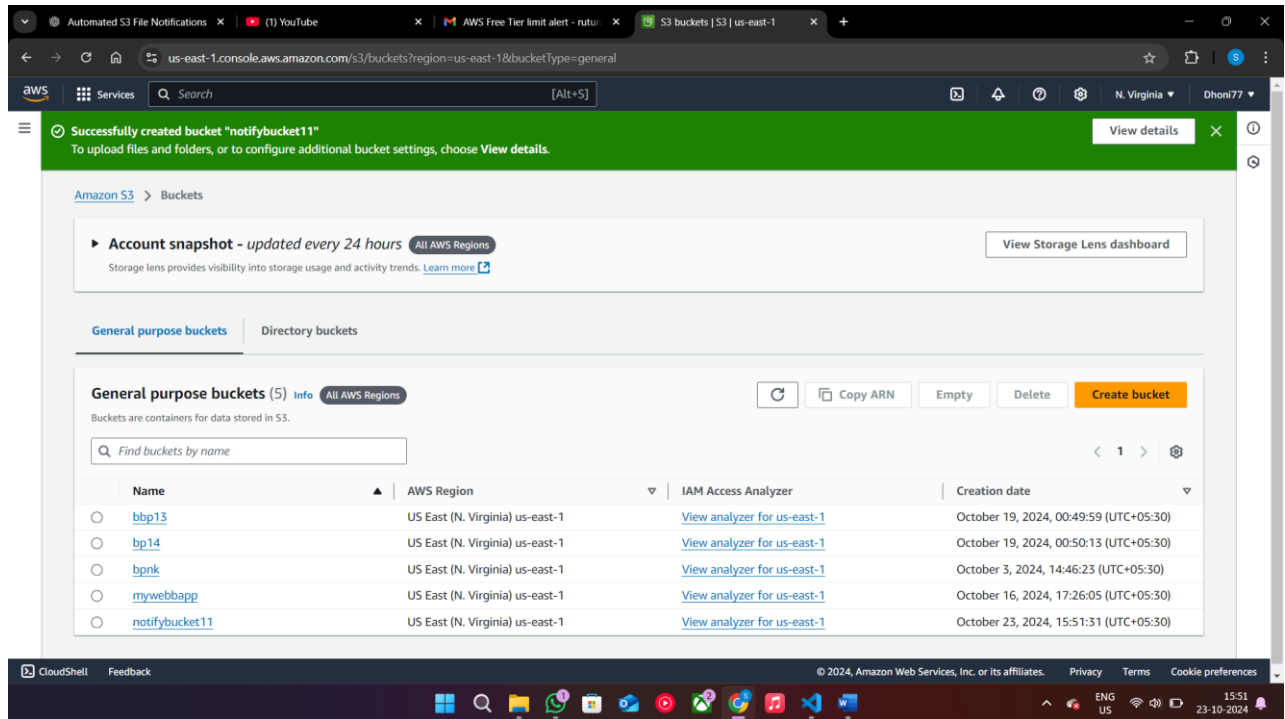
Step 1:

Creation of **S3 Bucket** -

Specify the unique name to the bucket and keep all other default settings as it is.



The screenshot displays the AWS Management Console interface for creating a new S3 bucket. The top navigation bar shows the AWS logo, 'Services' menu, a search bar, and the user's profile 'Madhura.Jangale'. The main content area is titled 'Buckets are containers for data stored in S3.' and features a 'General configuration' section. In this section, the 'AWS Region' is set to 'US East (N. Virginia) us-east-1'. Under 'Bucket type', the 'General purpose' option is selected, with a description: 'Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.' The 'Bucket name' field is populated with 'notifymdj', and a note states: 'Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming'. Below this, there is an optional section for 'Copy settings from existing bucket' with a 'Choose bucket' button. The 'Object Ownership' section is partially visible at the bottom, indicating control over ownership and ACLs.

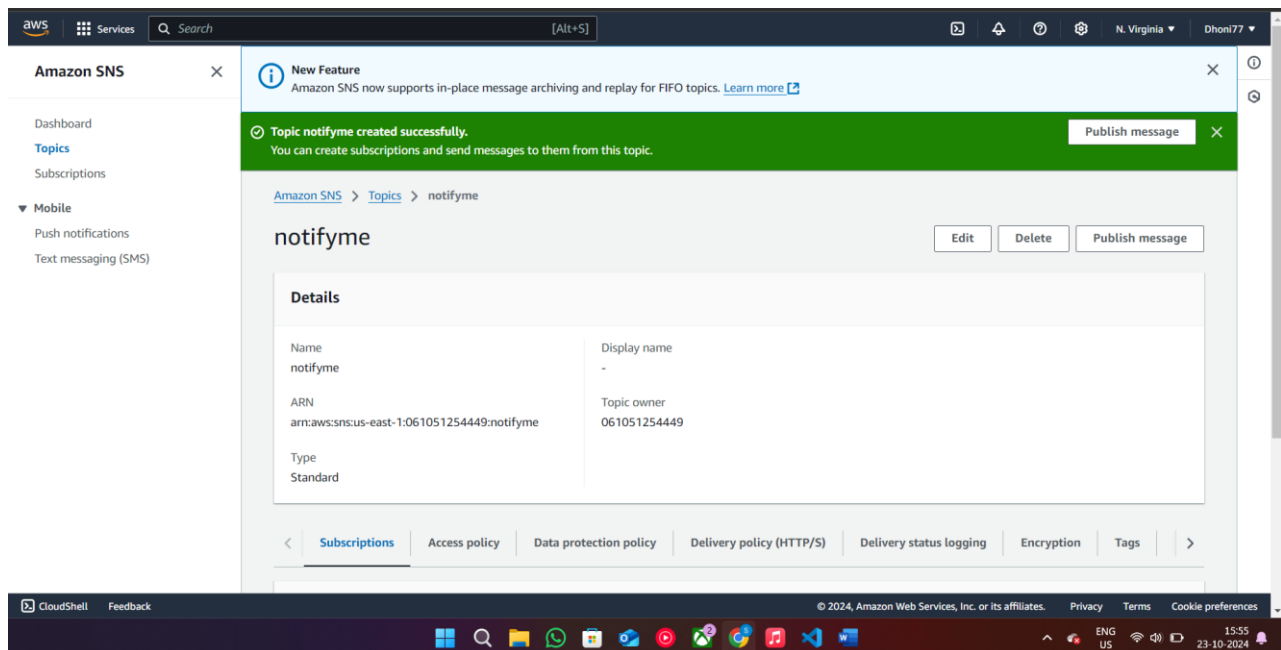


Successfully created S3 Bucket.

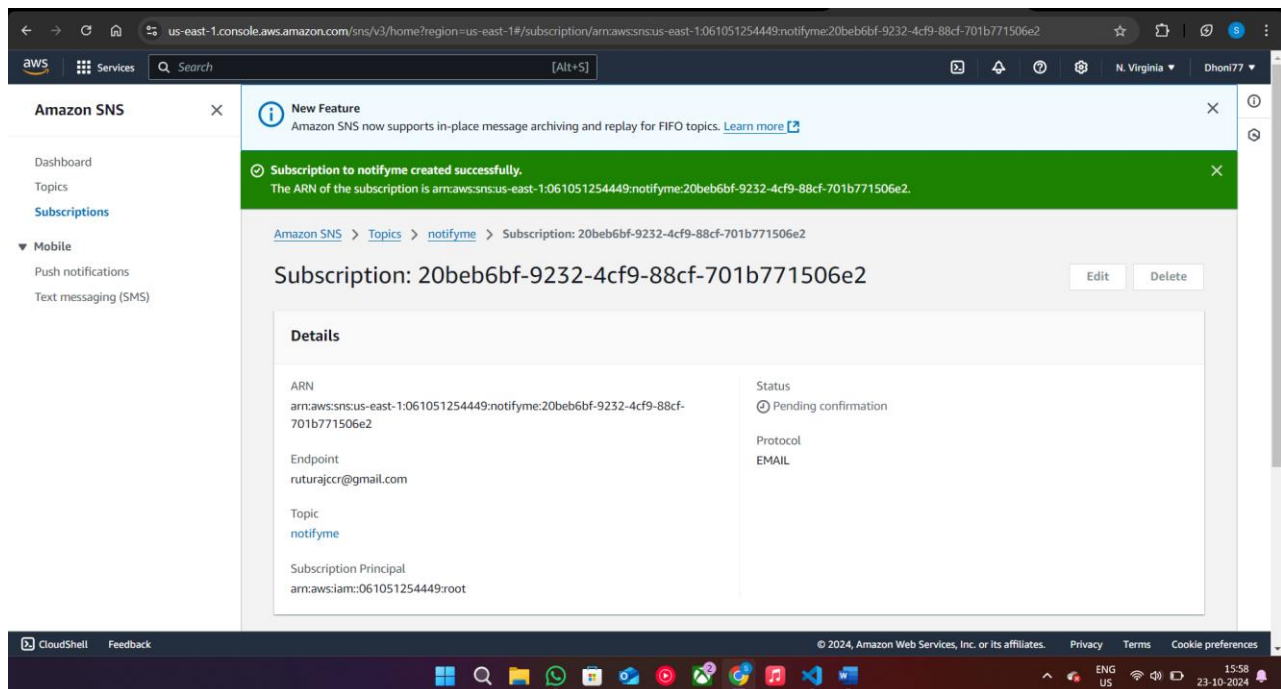
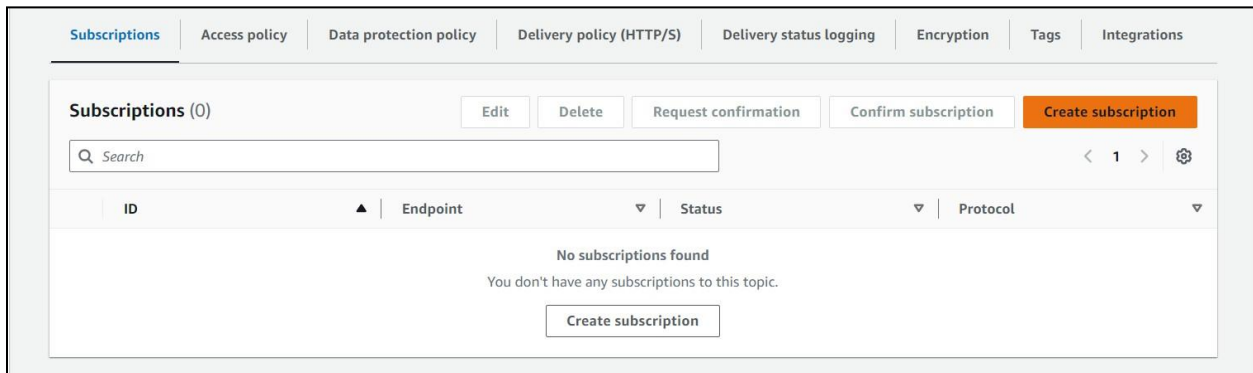
Step 2:

Set Up **SNS** Service-

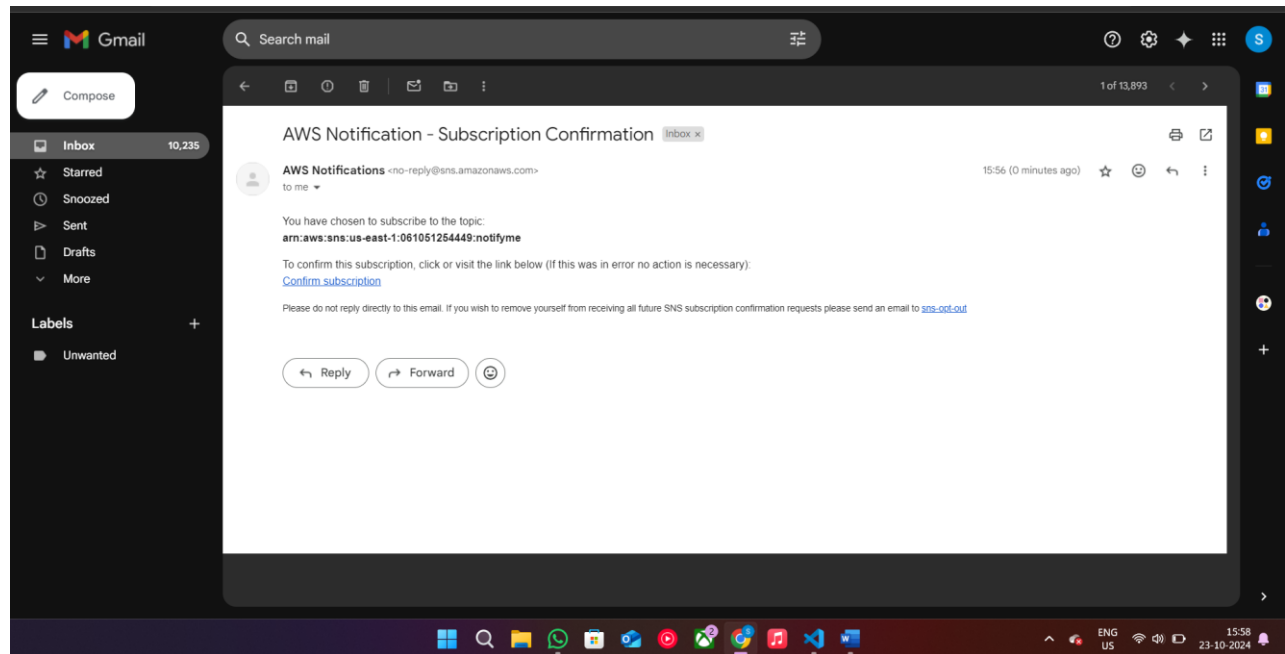
- Create a **new topic**:
 - Click "Create topic."
 - Select "Standard" as the type.
 - Enter a name for the topic and click "Create topic."

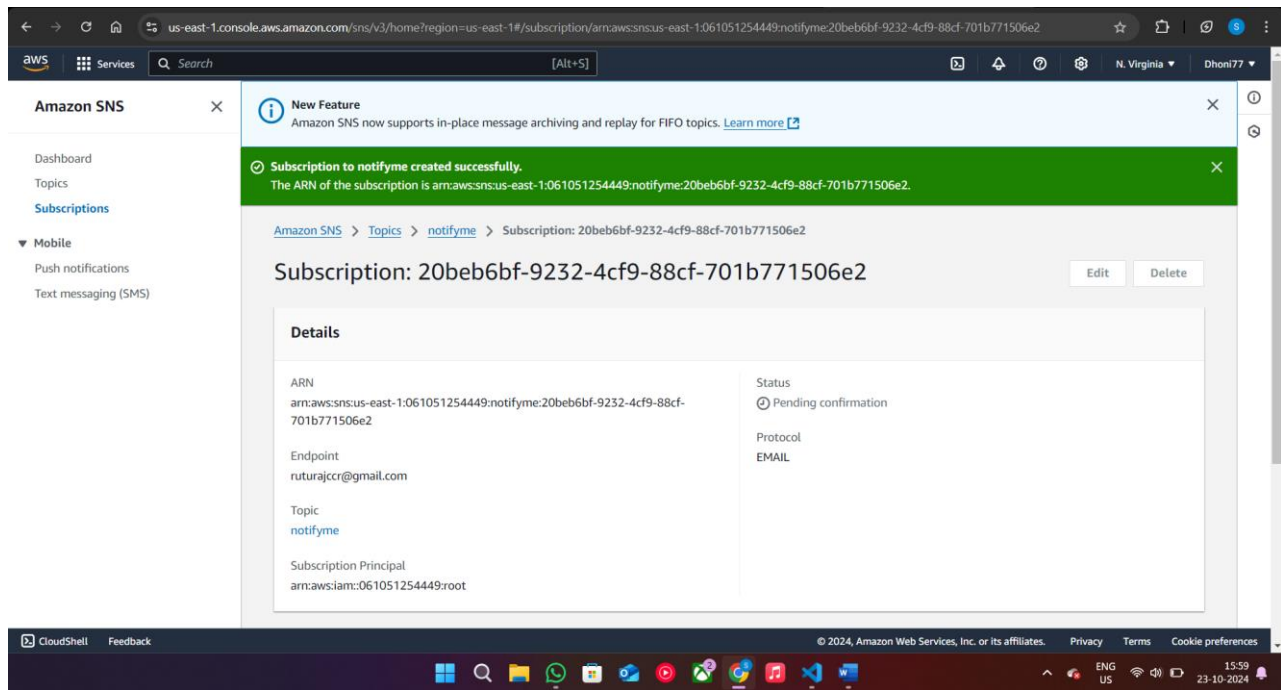


- Create a **subscription**:
 - Go to the topic's details page.
 - Click "Create subscription."
 - Choose "**Email**" as the **protocol**.
 - Enter the email address to receive notifications and click "Create subscription."



- **Confirm** subscription: Check your email for a confirmation message and click on the link to confirm the subscription.

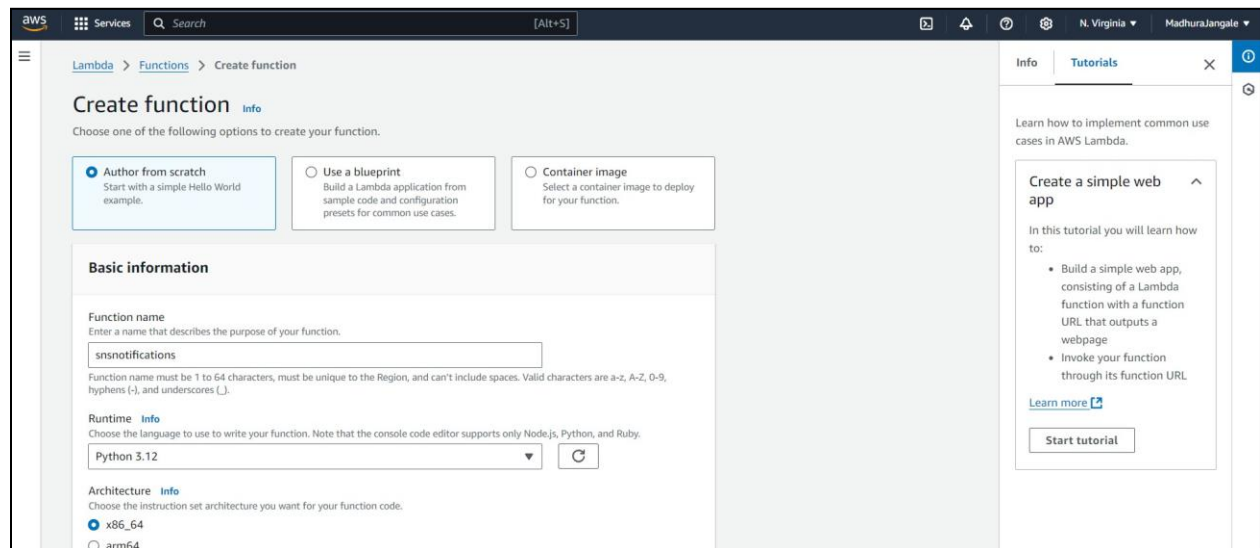




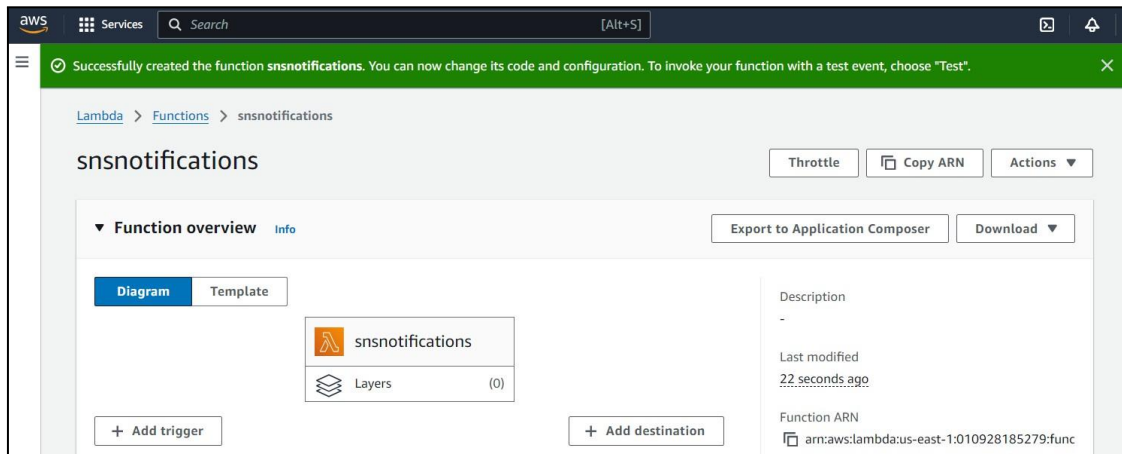
Email Confirmed.

Step 3:

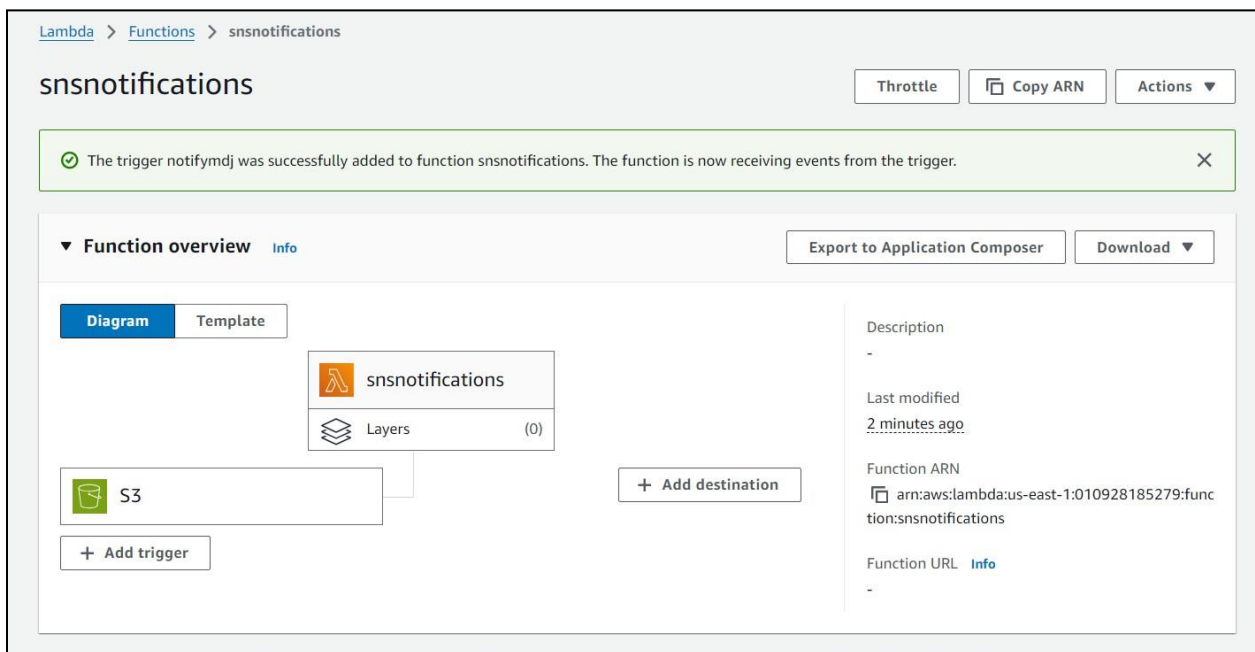
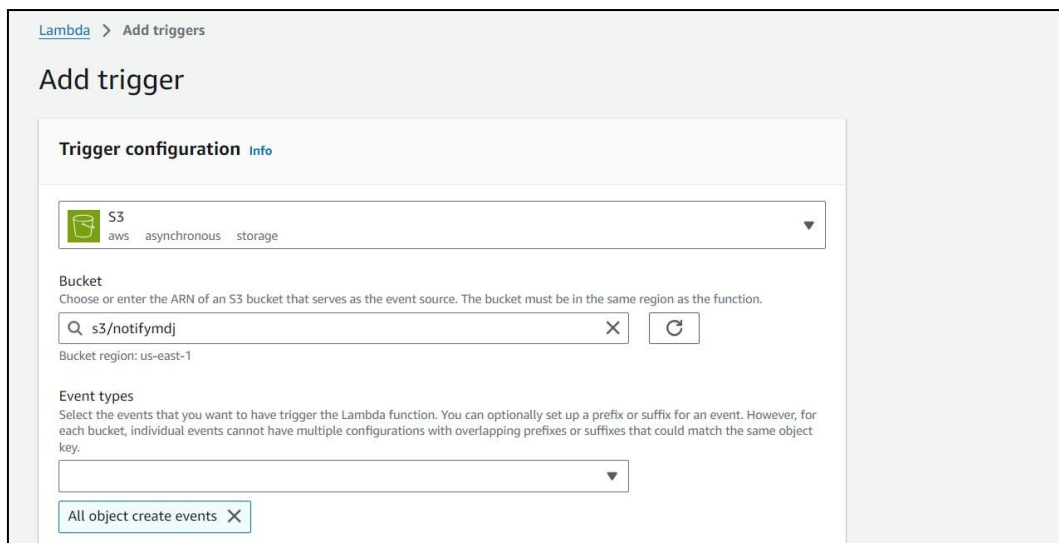
Create a Lambda Function -



Function created.

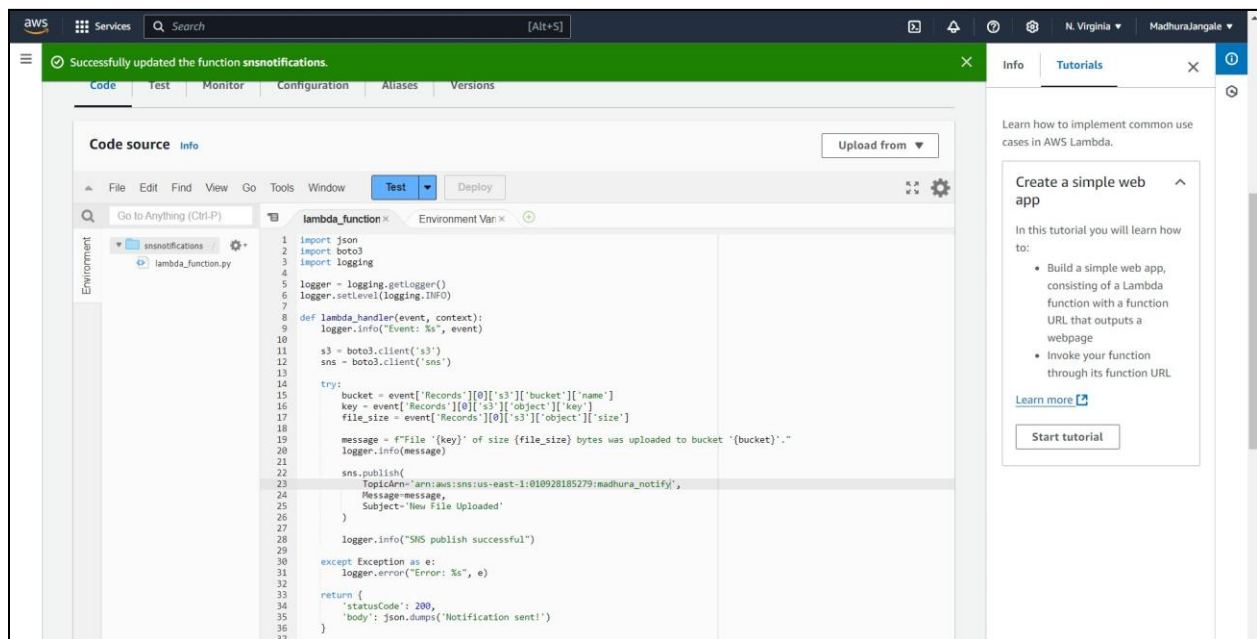


Now add **triggers** to it-



Now write the following code in the lambda function created-

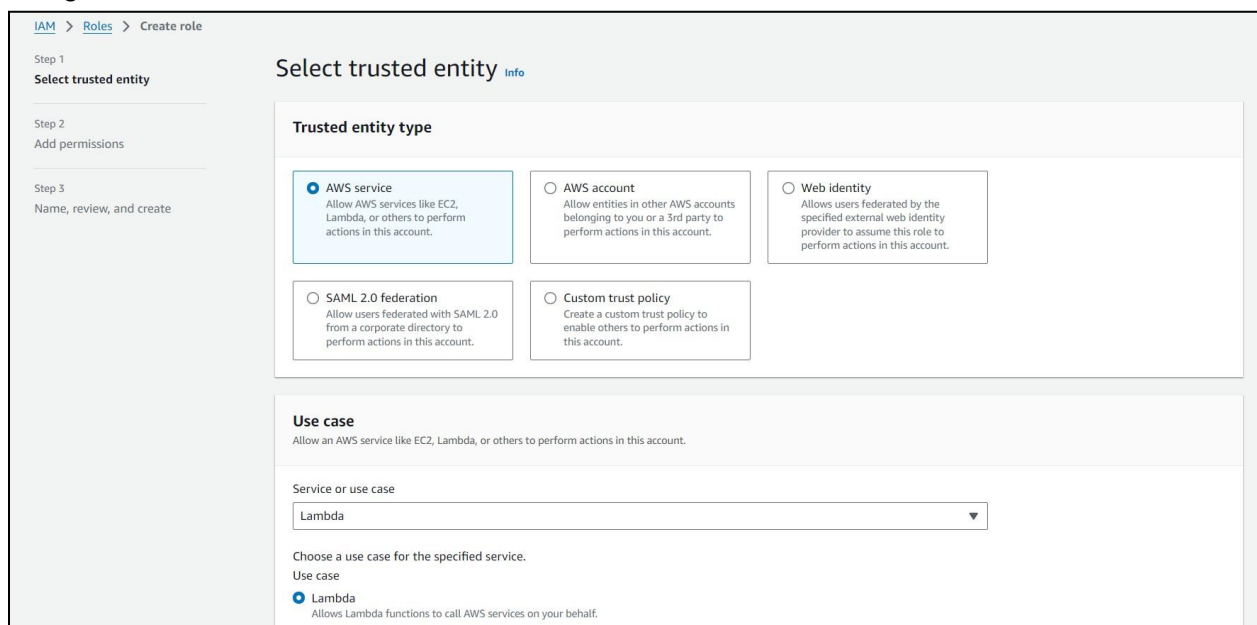
- Replace the **TopicArn** with the ARN of your SNS Topic.
- Then Click on Deploy.



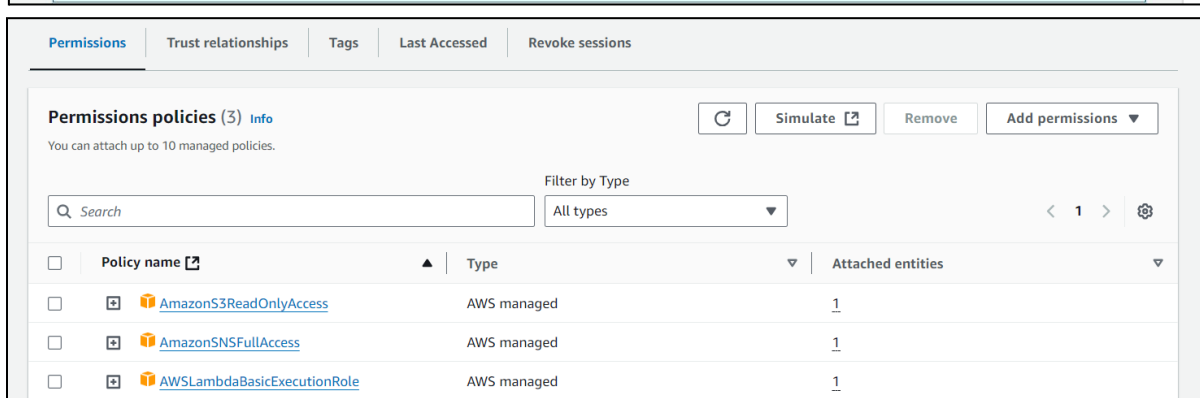
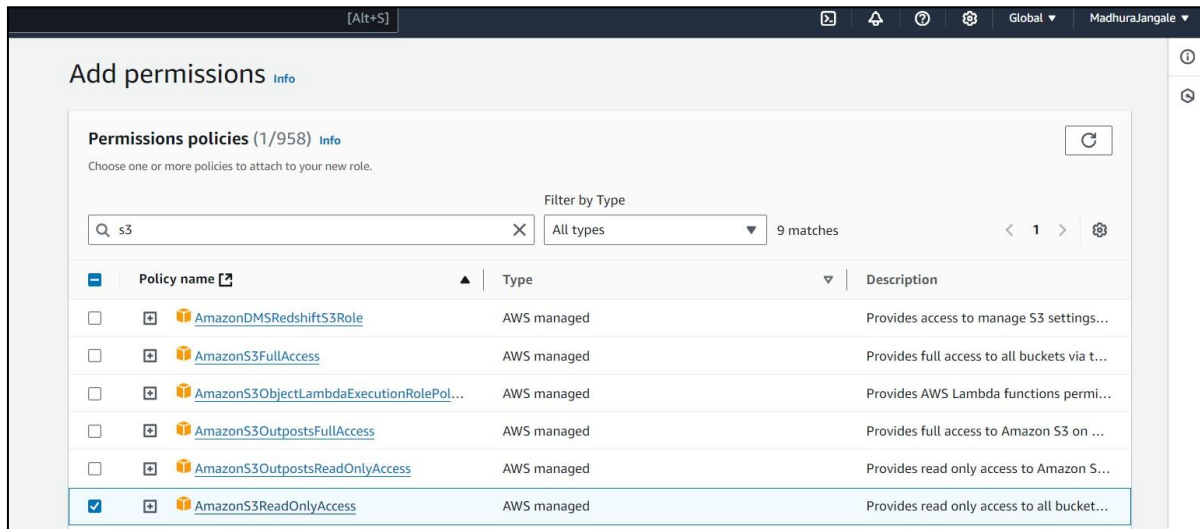
Step 4:

Configure **IAM role** for Lambda-

Navigate to IAM>Roles>Create role



Attach the "**AmazonS3ReadOnlyAccess**", "**AmazonSNSFullAccess**" and "**AWSLambdaBasicExecutionRole**" policies.



Give name to the role.

[Alt+S]

Global

MadhuraJangale

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

notifications

Maximum 64 characters. Use alphanumeric and '+=,@_-' characters.

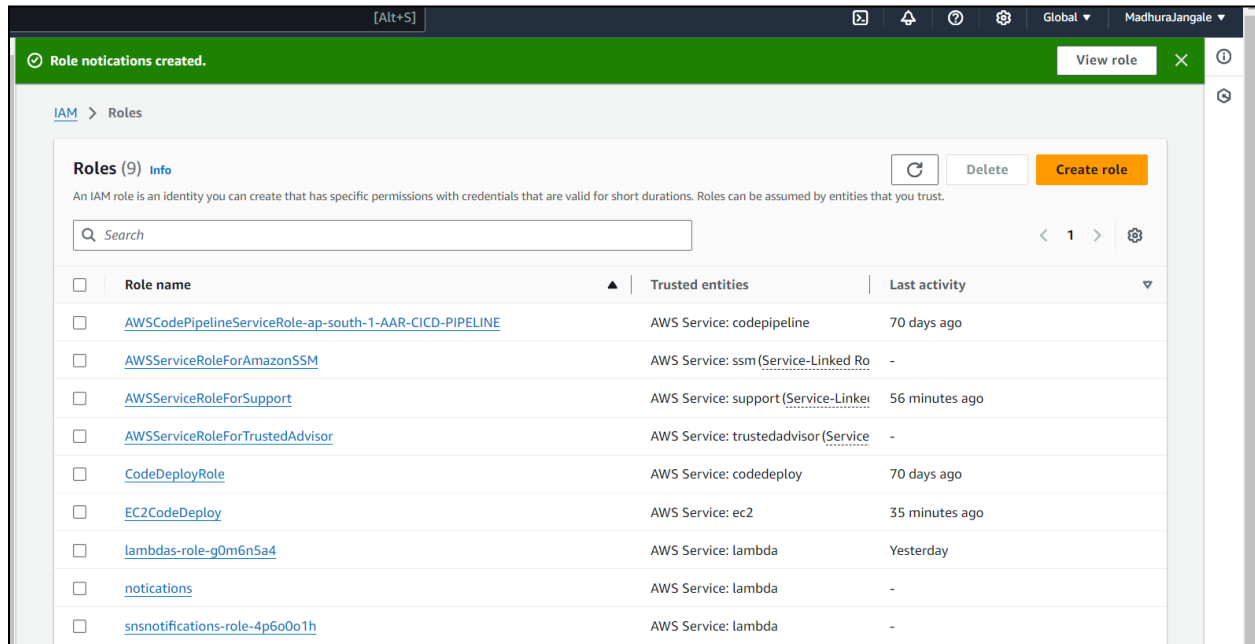
Description

Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=, @-/[]{}\$%'^";~"

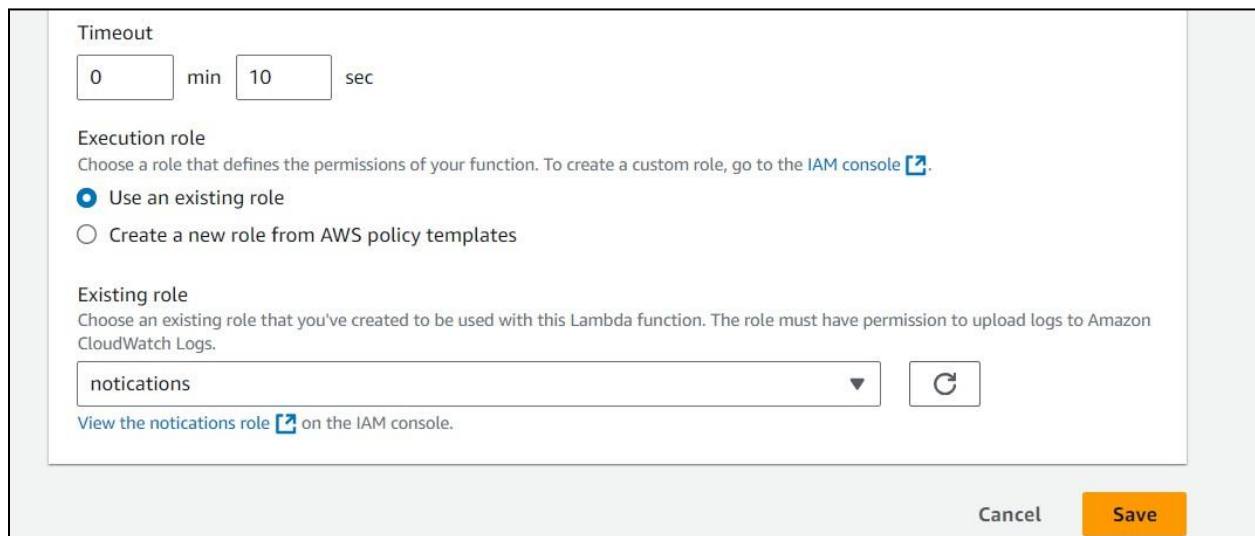
Role created.



Step 5:

Attach the role to your lambda function-

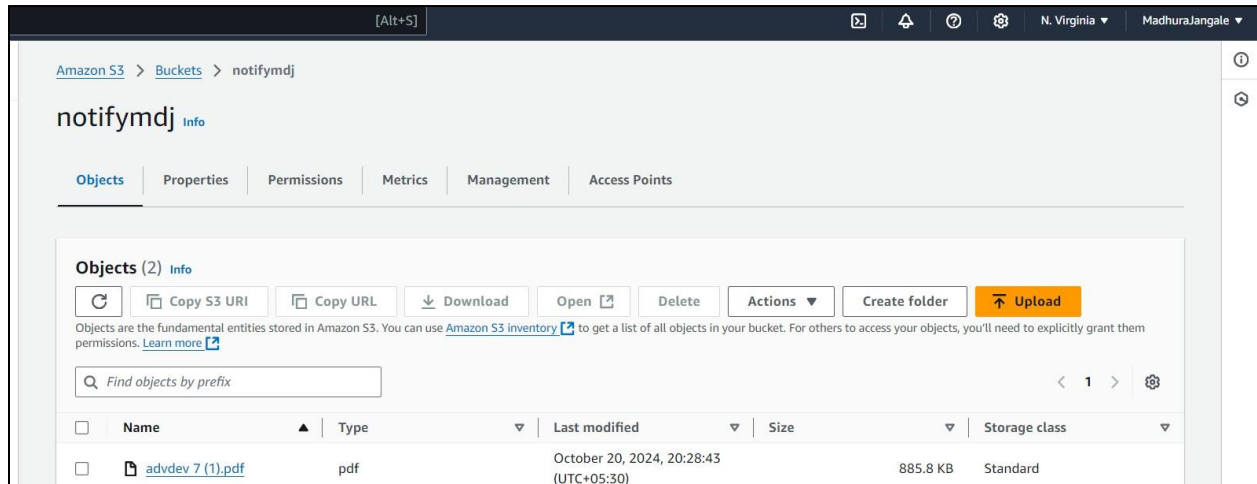
- Navigate to the lambda functions details page > under “execution role” > click “edit”
> Select “**use an existing role**” > choose the role created.
- Keep the **timeout of 10 sec**.



Step 6:

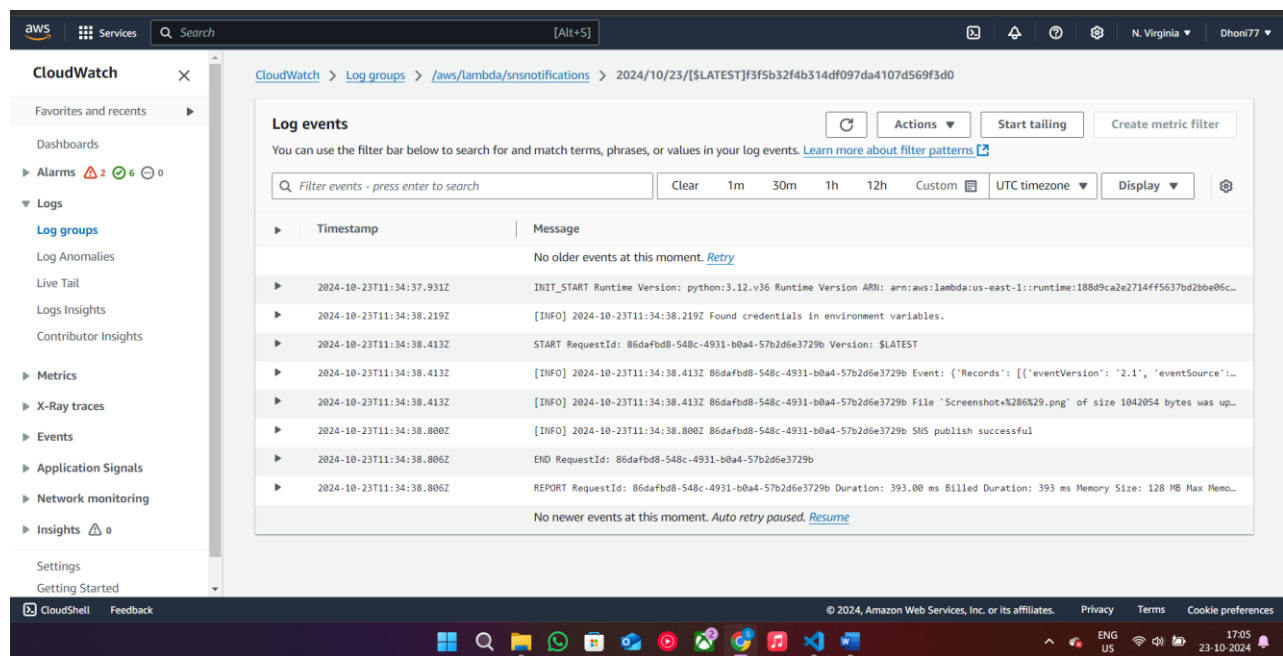
Check the setup-

Go to the S3 bucket you created > Click on upload and **upload the file.**

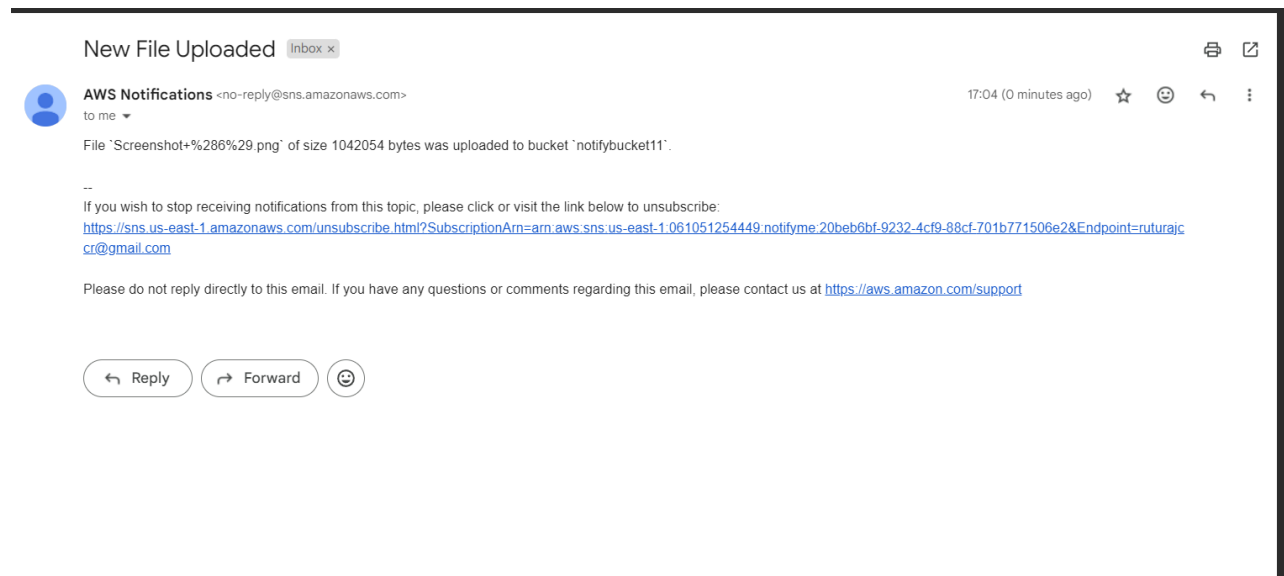


Check the **cloudwatch logs** of the lambda function used-

As below we can see the message **"SNS publish successful"** in logs.



Check your inbox of the email used in sns you will see the mail of file upload as below.



AWS Notifications <no-reply@sns.amazonaws.com>

to me ▾

File "S

--

If you

<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:061051254449:notifyme:20beb6bf-9232-4cf9-88cf-701b771506e2&Endpoint=ruturajccr@gmail.com>

Please

from: **AWS Notifications** <no-reply@sns.amazonaws.com>
to: ruturajccr@gmail.com
date: 23 Oct 2024, 17:04
subject: New File Uploaded
mailed-by: amazonses.com
Signed by: sns.amazonaws.com
security: Standard encryption (TLS) [Learn more](#)
👉 : Important according to Google magic.

← Reply

→ Forward



Execution completed successfully.

