

Aditi Taksale/D15A/60

Subject: Advanced DevOps

## Practical Exam Case Study

### Case Study No. 20: Automated Notifications using SNS

Concepts Used: AWS Lambda, S3, SNS.

**Problem Statement :** Develop 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:

- **Python Lambda Function:** Write a Python-based Lambda function that is triggered by S3 upload events.
- **Extract File Information:** Capture the file name and file size from the S3 event and format this information into a notification.
- **Send Notifications via SNS:** Use SNS to deliver the notification to a configured email address.
- **Test:** Upload a file to the S3 bucket and verify the receipt of the email notification.

#### Introduction:

##### Overview:

In this case study, we focus on implementing a system that leverages AWS services to automate email notifications whenever a file is uploaded to an Amazon S3 bucket. This system uses AWS Lambda, Amazon S3, and Amazon SNS to provide real-time updates, which are essential in scenarios like data processing, monitoring, and other environments that require instant feedback.

##### Key AWS Components:

- **AWS Lambda:** A serverless compute service that runs code in response to specific events, such as file uploads to S3. It eliminates the need to manage infrastructure, providing a scalable and cost-effective solution.
- **Amazon S3:** A highly reliable and secure object storage service. In this system, S3 triggers an event each time a new file is uploaded to a specified bucket.
- **Amazon SNS:** SNS (Simple Notification Service) is used to send notifications. In this use case, we employ the email protocol to keep users informed about file uploads in real-time.

#### Key Feature and Application-

##### Key Feature:

The primary feature of this system is the automated email notifications powered by the integration of AWS Lambda and SNS. As soon as a file is uploaded to the S3 bucket, a notification is triggered automatically, reducing manual intervention and ensuring timely alerts..

### Application:

This system is highly useful in scenarios where real-time notifications are critical. For example, in a data processing workflow, when a new file is uploaded, relevant team members are instantly notified, allowing processes to start promptly without human intervention. This improves operational efficiency and reduces delays.

### Advantages:

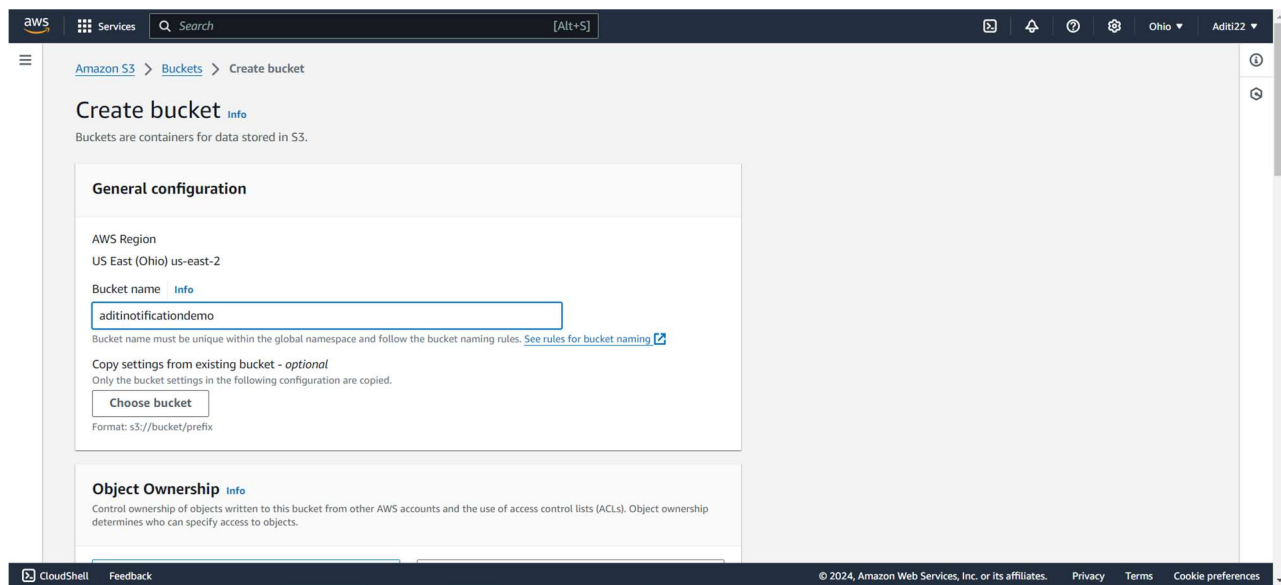
**Instant Notifications:** Users receive immediate alerts for new uploads, ensuring that actions can be taken in a timely manner.

**Seamless Scalability:** The system scales automatically with the demand, thanks to the elasticity and scalability of AWS infrastructure.

### Implementation:

#### Step 1:

##### Creation of **S3 Bucket** -



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

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."

The screenshot shows the AWS Management Console interface for creating a new Amazon SNS topic. At the top, there's a navigation bar with the AWS logo, 'Services' link, a search bar, and user information. Below this is a 'New Feature' banner. The main content area is titled 'Create topic' and includes a breadcrumb trail: 'Amazon SNS > Topics > Create topic'. Under the 'Details' section, the 'Type' is set to 'Standard' (selected with a radio button). The 'FIFO (first-in, first-out)' option is also visible but not selected. Below the type selection, there's a 'Name' field containing the text 'practical'. At the bottom, there's a 'Display name - optional' field which is currently empty. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for 2024.

aws Services [Alt+S] Ohio Aditi22

**New Feature**  
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Amazon SNS > Topics > Create topic

## 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

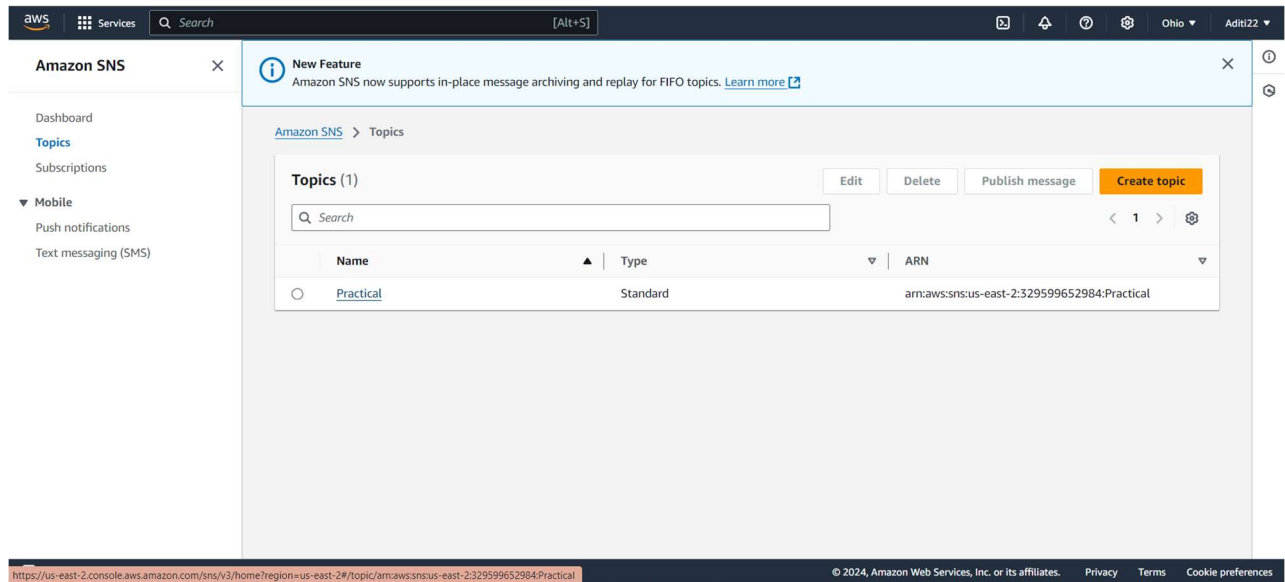
**Name**

practical

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

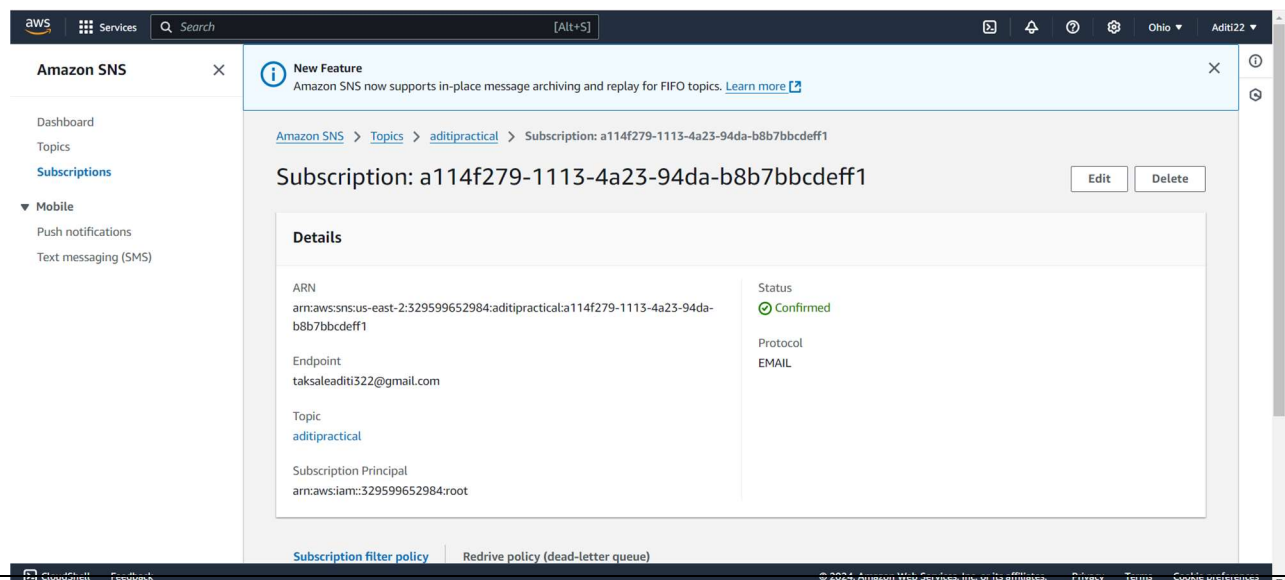
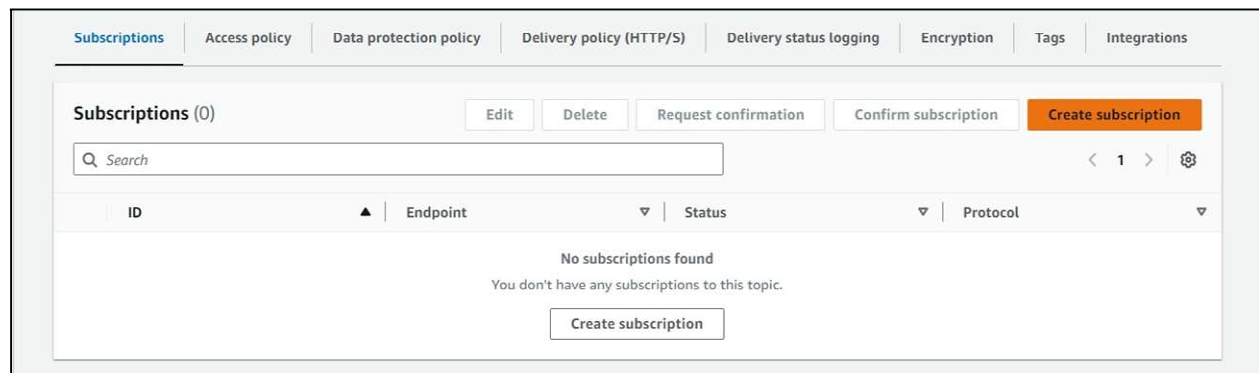
**Display name - optional** [Info](#)  
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

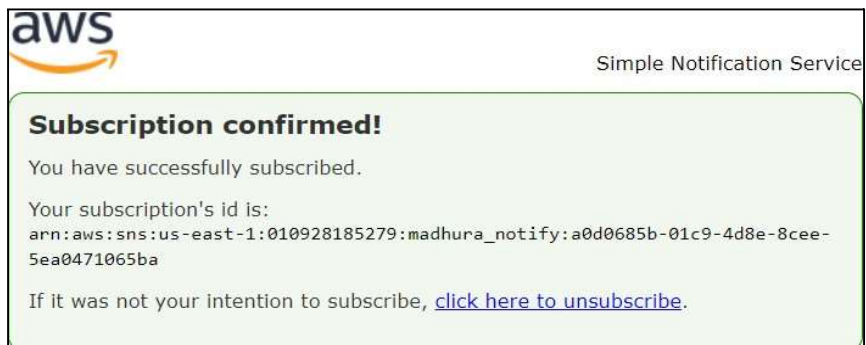
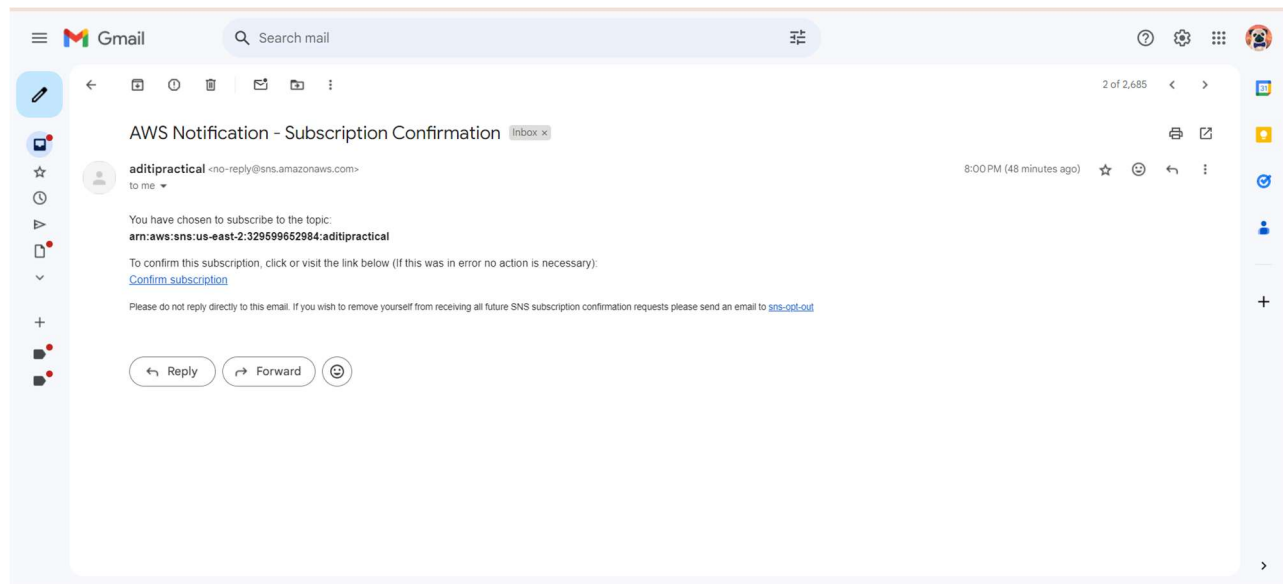


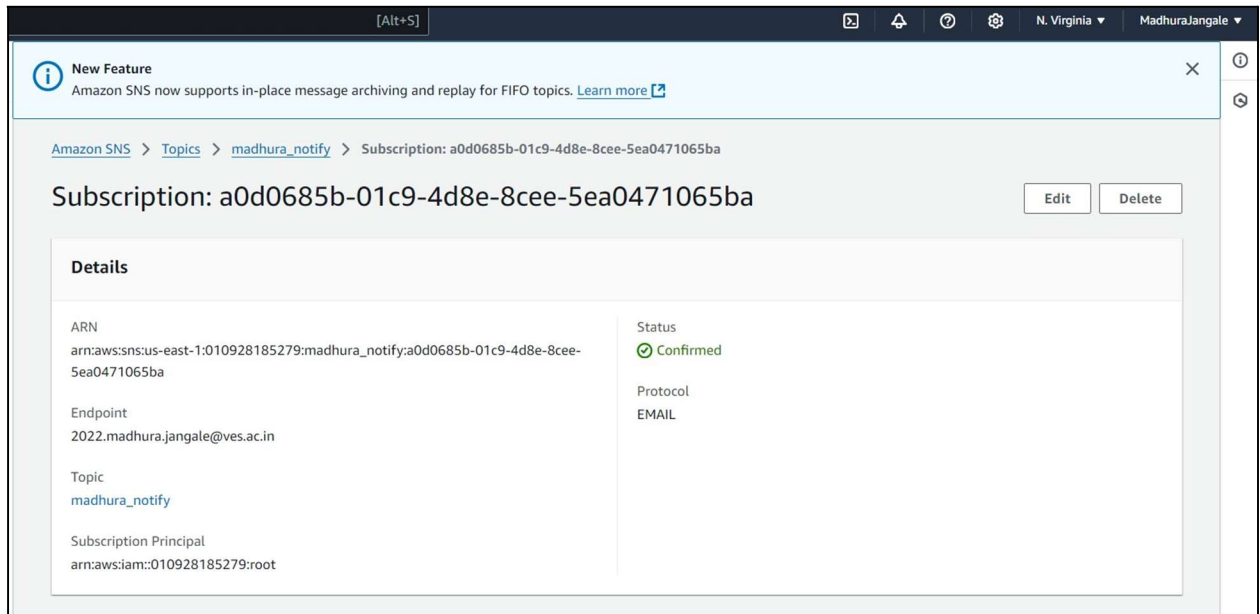
- Create a **subscription**:

- 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.

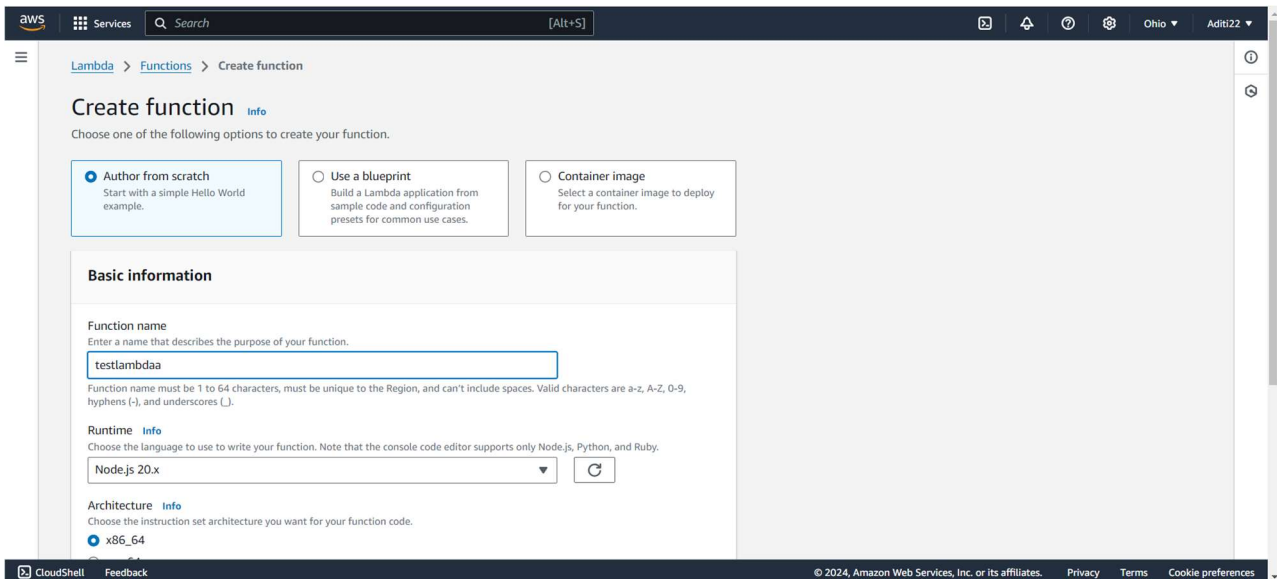




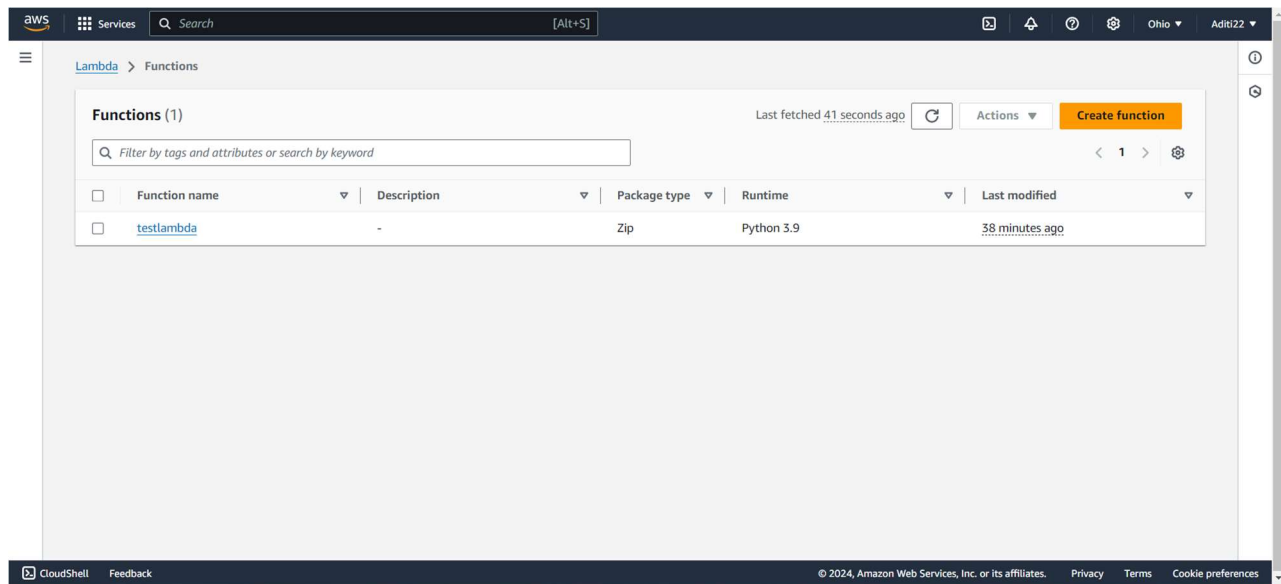
Email Confirmed.

### Step 3:

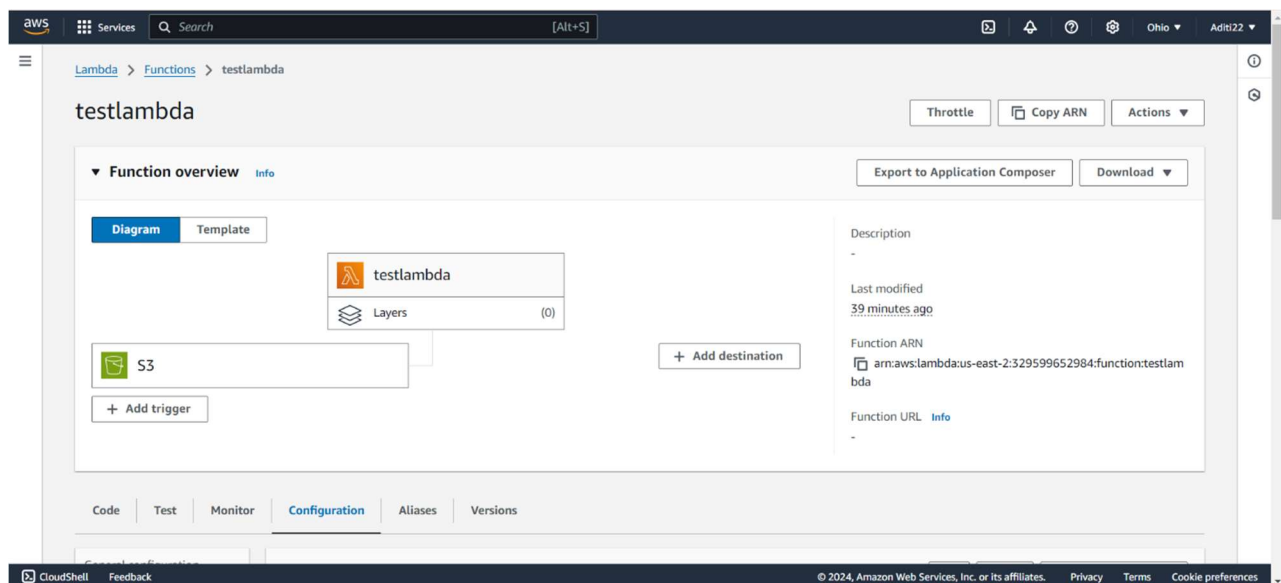
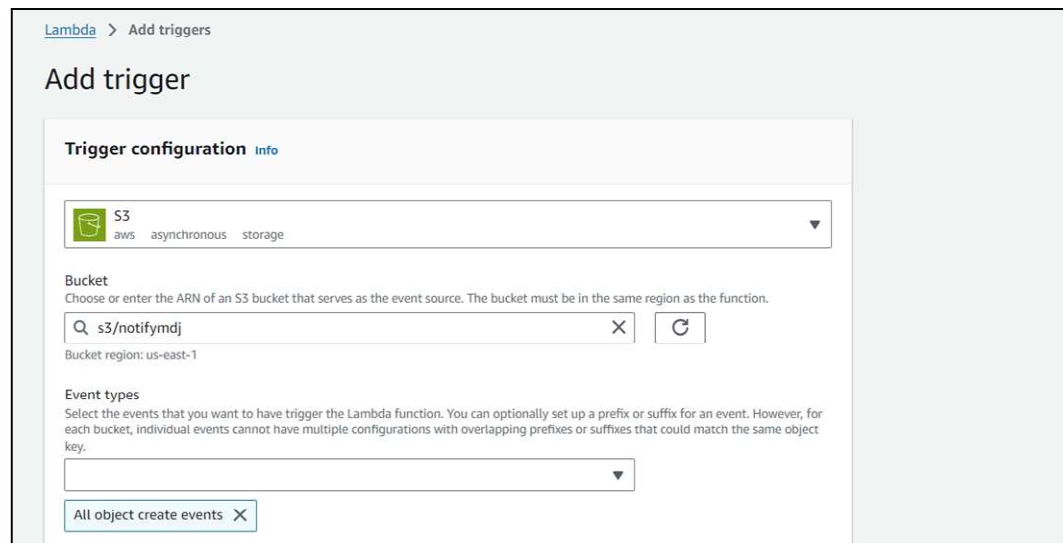
#### Create a Lambda Function -



Function created.

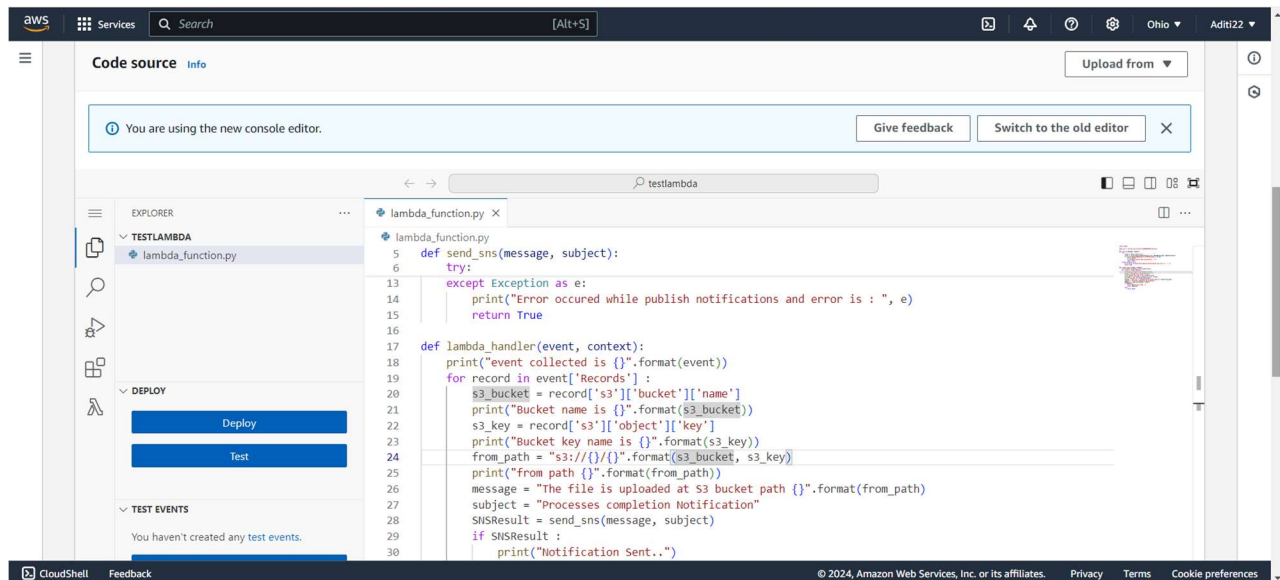


Now add **triggers** to it-



Now write the following code in the lambda function created-

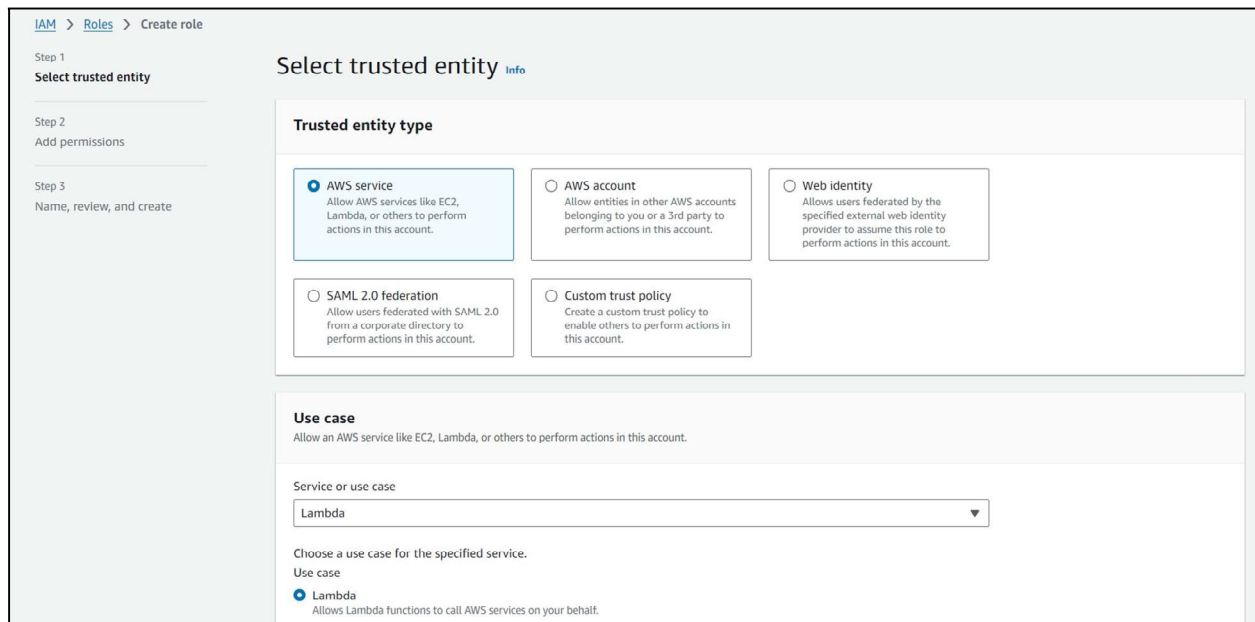
- Replace the **TopicArn** with the ARN of your SNS Topic.
- Then Click on Deploy.



#### Step 4:

Configure **IAM role** for Lambda-

Navigate to **IAM>Roles>Create role**

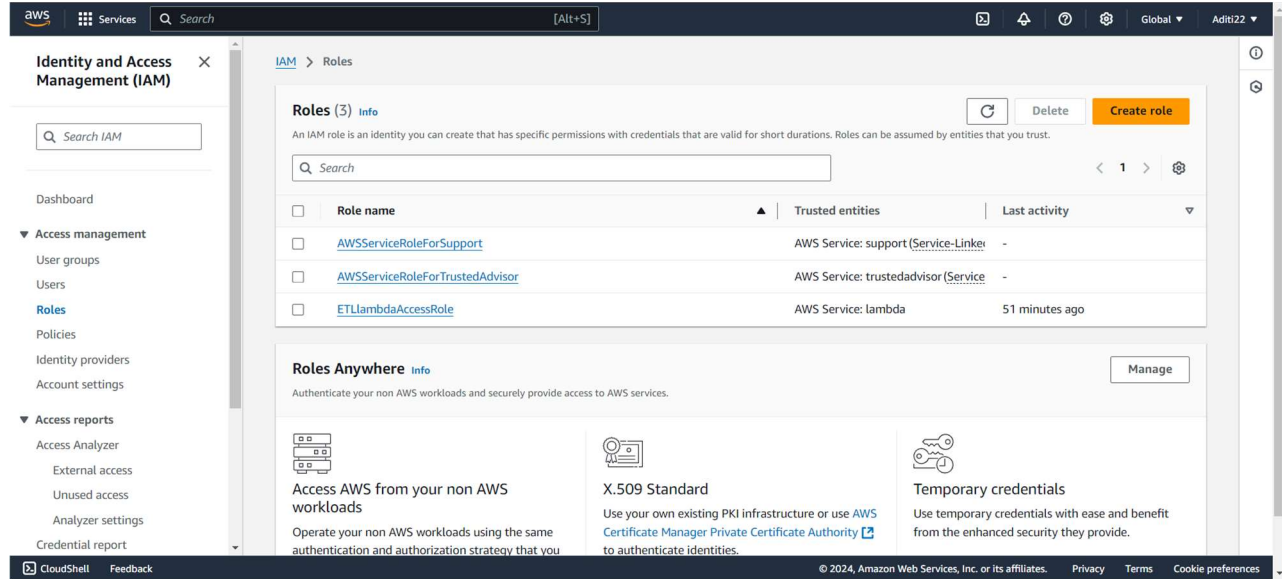


Attach the "**AmazonS3ReadOnlyAccess**", "**AmazonSNSFullAccess**" and "**AWSLambdaBasicExecutionRole**" policies.





## Role created.



The screenshot shows the AWS IAM console. The left sidebar is titled 'Identity and Access Management (IAM)' and includes a search bar and a navigation menu with sections like 'Access management' (Users, Roles, Policies, etc.) and 'Access reports'. The main content area is titled 'IAM > Roles' and shows a list of roles. The roles listed are:

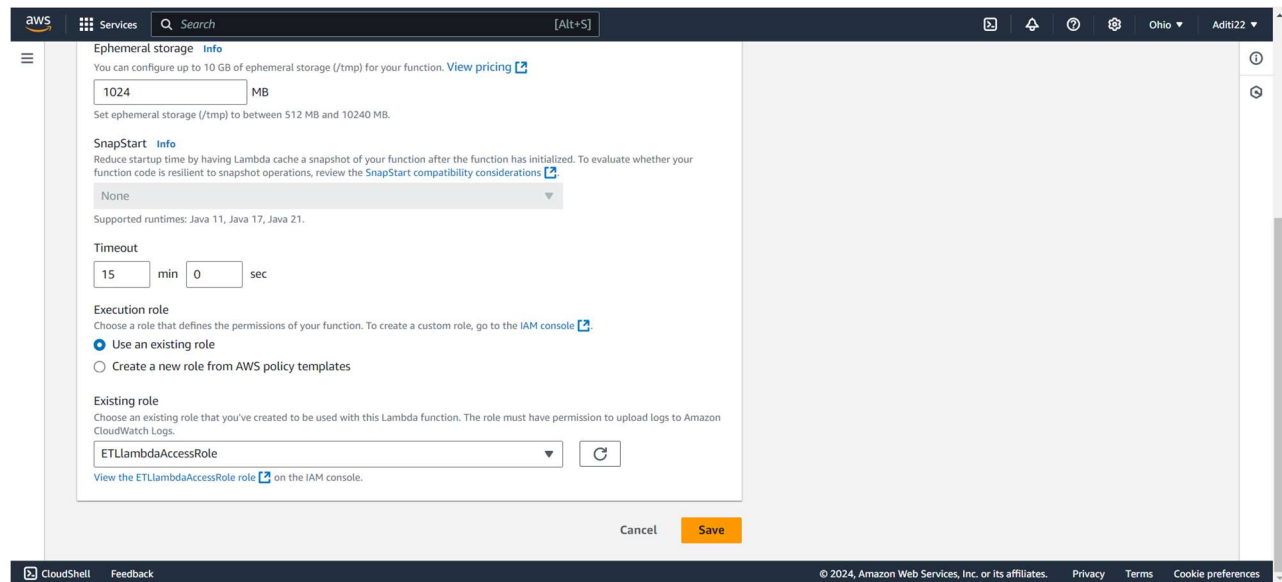
Role name	Trusted entities	Last activity
<a href="#">AWSServiceRoleForSupport</a>	AWS Service: support (Service-Linker)	-
<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS Service: trustedadvisor (Service-Linker)	-
<a href="#">ETLambdaAccessRole</a>	AWS Service: lambda	51 minutes ago

Below the list is a 'Roles Anywhere' section with a 'Manage' button. It contains three cards: 'Access AWS from your non AWS workloads', 'X.509 Standard', and 'Temporary credentials'.

## 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 15 min**.

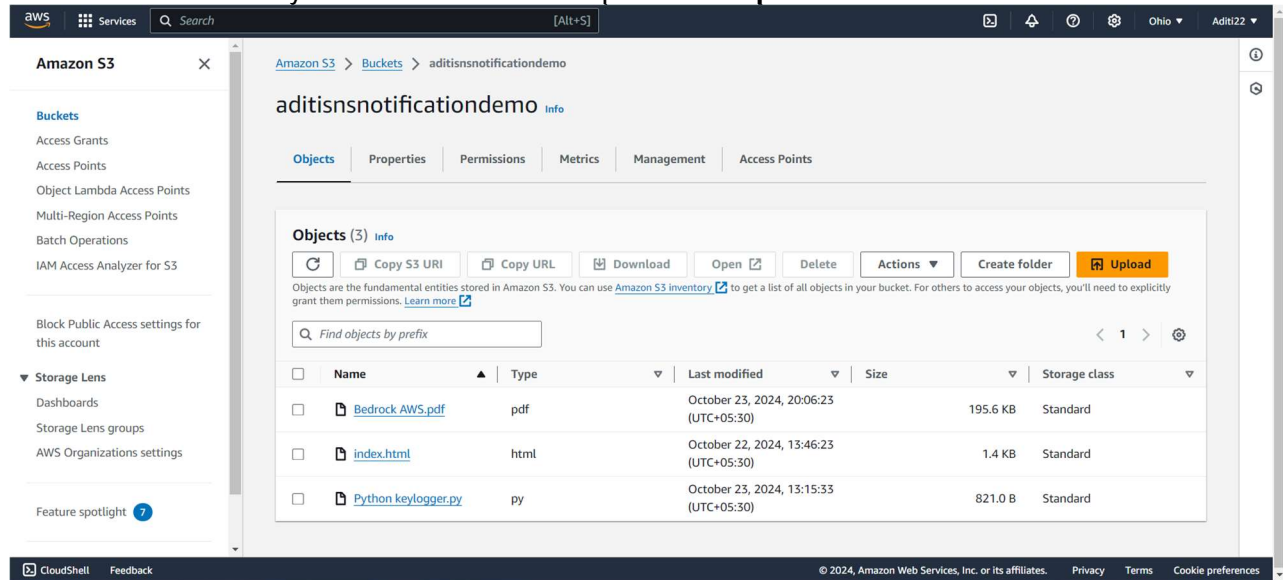


The screenshot shows the AWS Lambda console. The left sidebar is titled 'Services' and includes a search bar. The main content area is titled 'Ephemeral storage' and shows configuration options for a function. The 'Execution role' section is expanded, showing options to 'Use an existing role' or 'Create a new role from AWS policy templates'. The 'Existing role' dropdown is set to 'ETLambdaAccessRole'.

## Step 6:

Check the setup-

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



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

