Project Title: Serverless Image Processing System using AWS Lambda, S3, DynamoDB, and SNS

Project Overview: This project demonstrates a serverless image processing system using AWS services. The system automatically processes images uploaded to an S3 bucket, moves them to a processed bucket, stores metadata in DynamoDB, and sends an SNS notification upon successful processing.

Architecture Overview: The system follows a fully serverless architecture using AWS managed services:

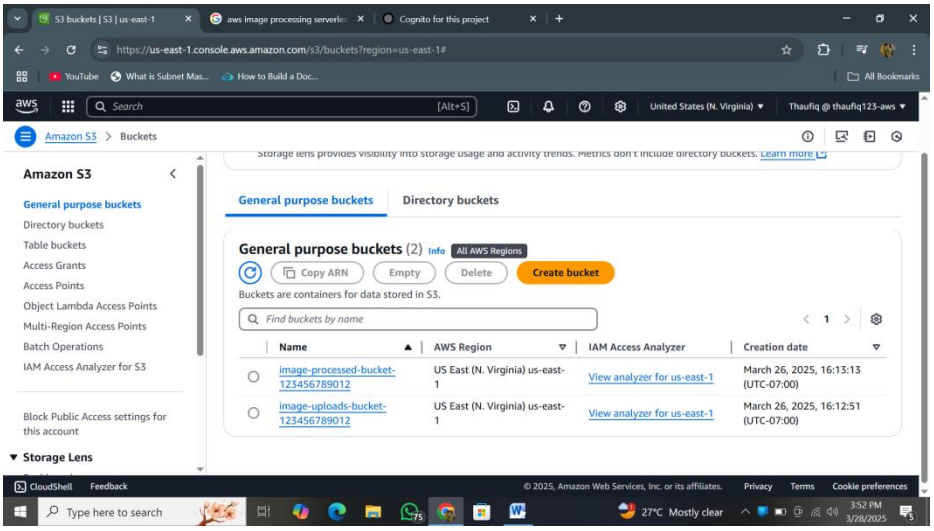
- **S3:** Stores uploaded and processed images.
- **S3 Event Notifications:** Triggers AWS Lambda when a new image is uploaded.
- **AWS Lambda:** Handles image transfer, metadata storage, and notification.
- **DynamoDB:** Stores metadata of processed images.
- **SNS:** Sends notifications upon image processing completion.

AWS Services Used:

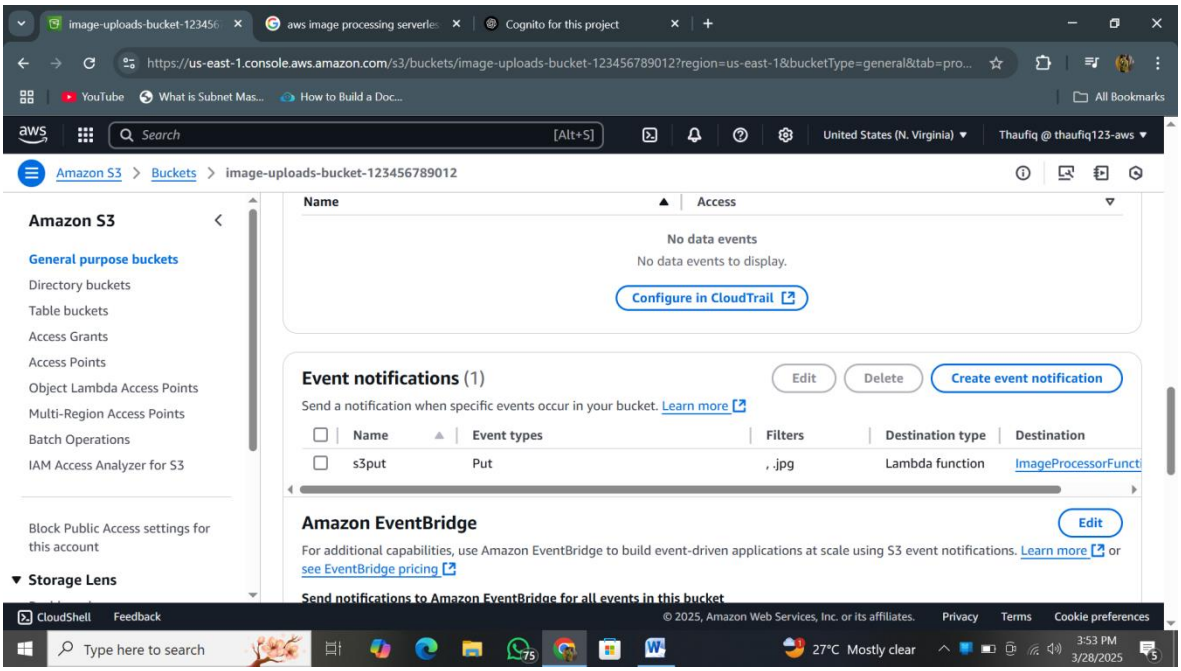
- Amazon S3 (image storage)
- AWS Lambda (image processing and metadata storage)
- Amazon DynamoDB (metadata storage)
- Amazon SNS (notifications)

Implementation Steps:

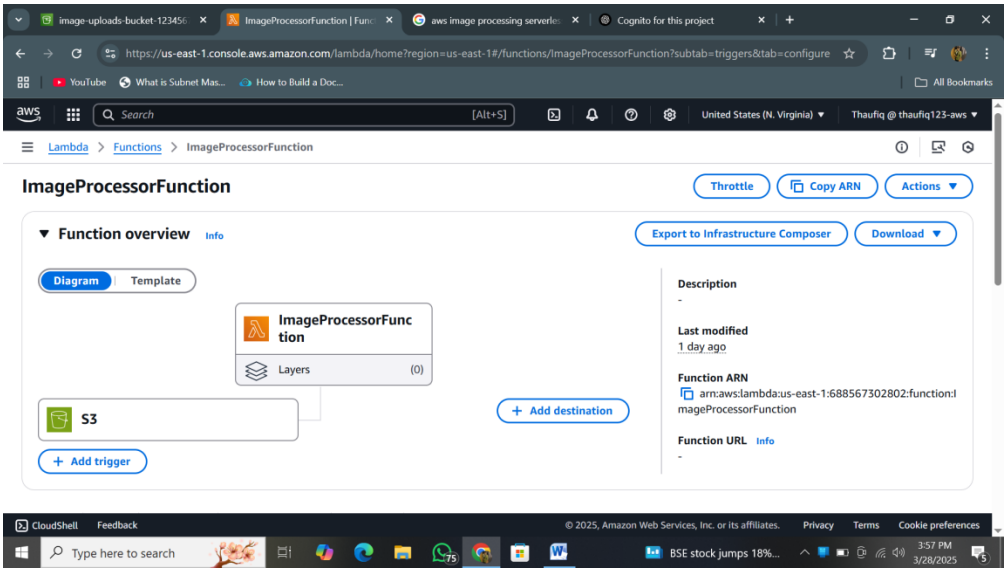
1. **Create Two S3 Buckets:**
    - `image-uploads-bucket-123456789012` **(for uploads)**
    - `image-processed-bucket-123456789012` **(for processed images)**



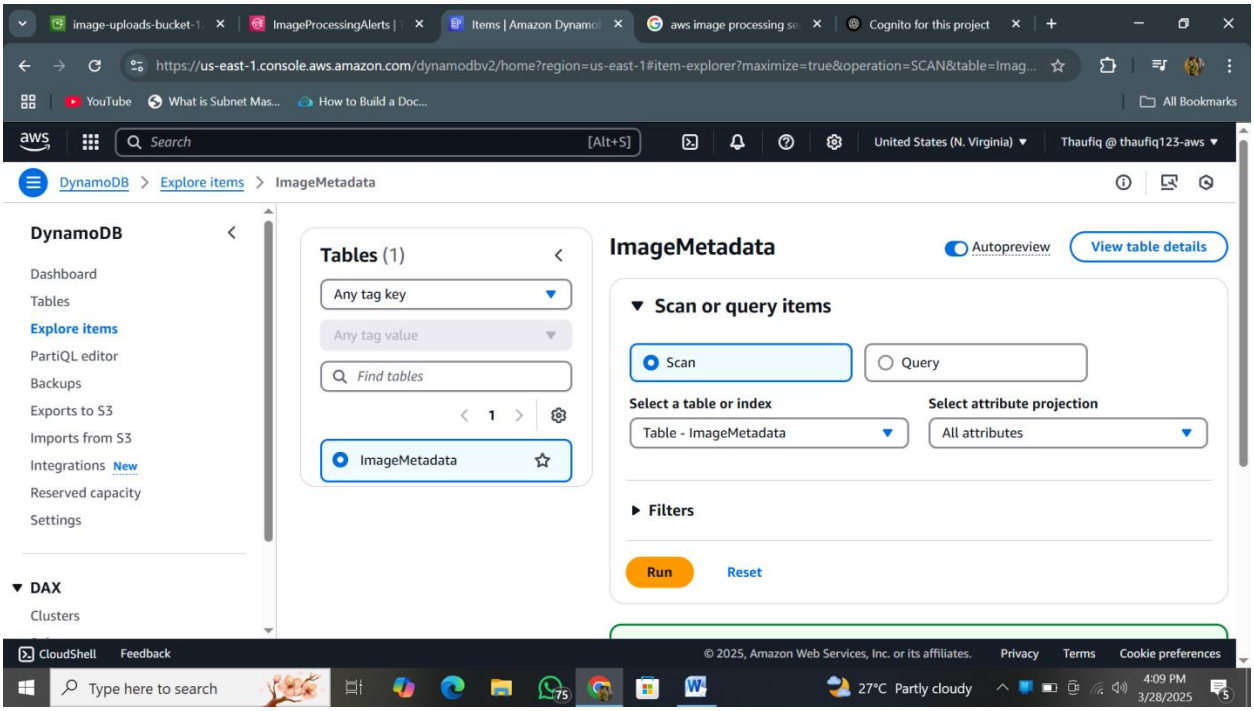
- **Configure S3 Event Notifications:**
    - Set up an event notification on `image-uploads-bucket-123456789012` to trigger a Lambda function on new object creation.



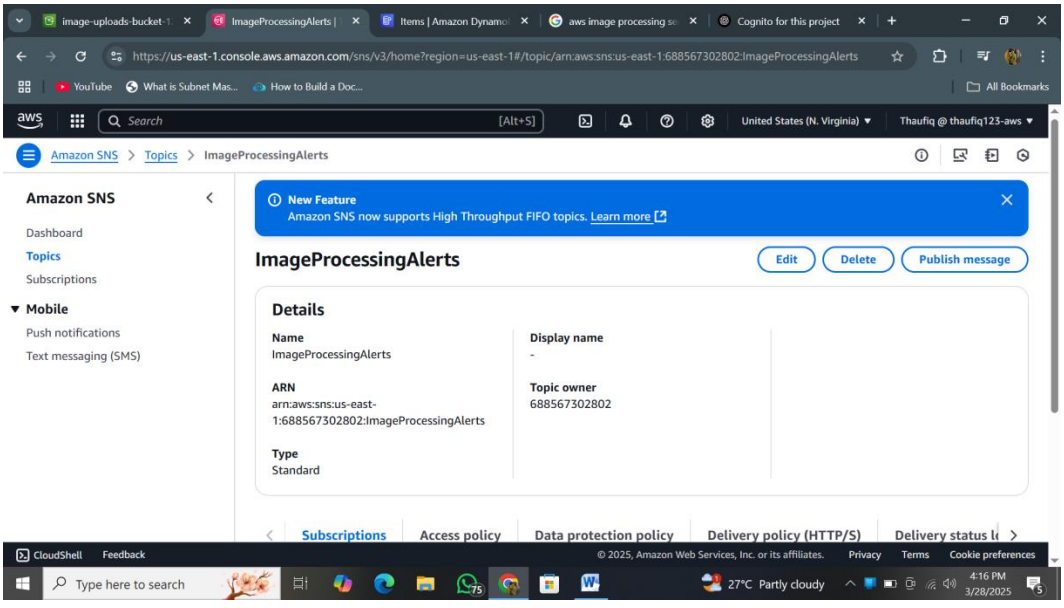
- **Create an AWS Lambda Function:**
    - The function copies the uploaded image to **`imageprocessedfunction`**
    - Stores metadata in a DynamoDB table **(`ImageMetadata`).**
    - Sends an SNS notification.

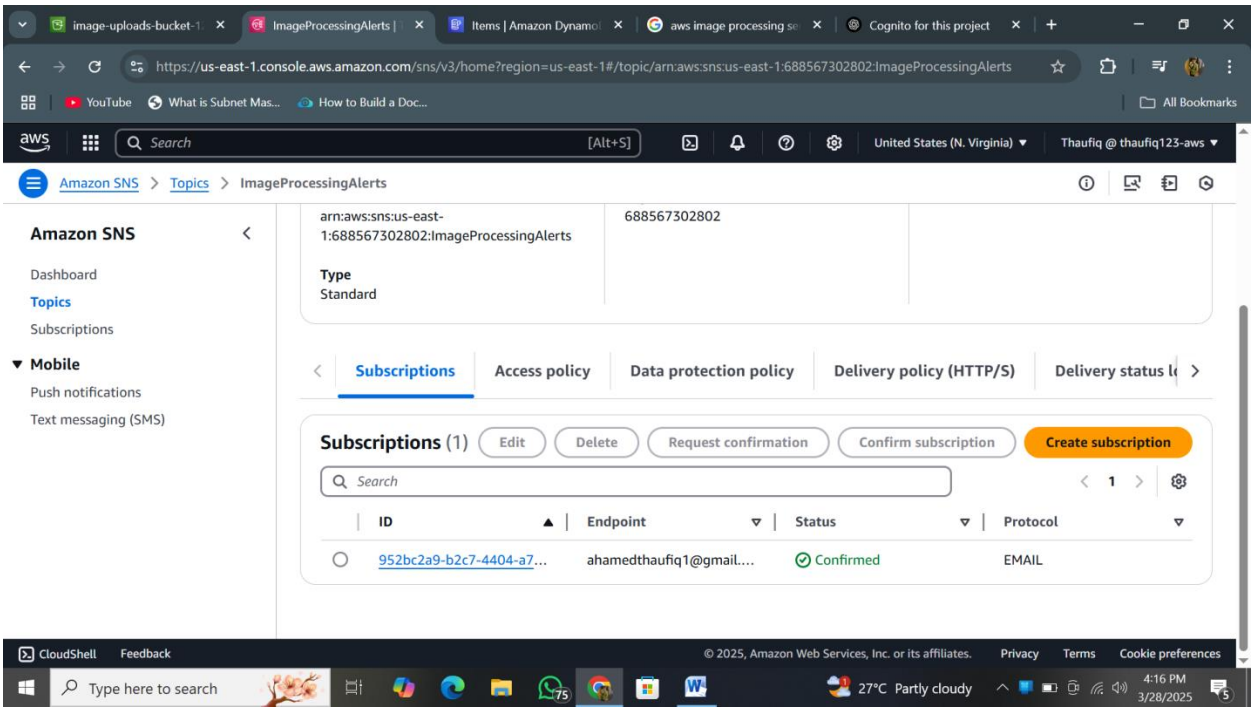


- **Set Up DynamoDB Table:**
    - Table Name: `ImageMetadata`
    - Primary Key: `ImageID`



- **Create an SNS Topic:**
    - Topic Name: `ImageProcessingAlerts`
    - Subscribe an email address to receive notifications.





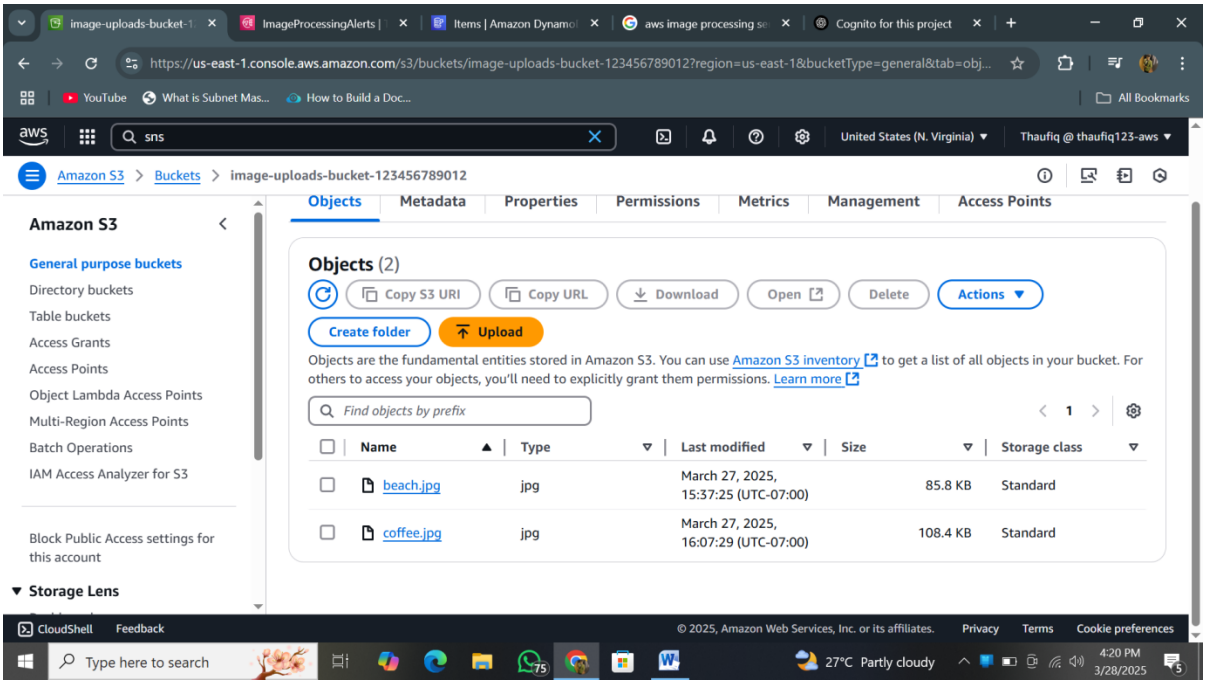
Code Snippets: *Lambda Function (Python - Boto3)*

```
import boto3
import json
import time

# AWS Clients
s3 = boto3.client("s3")
dynamodb = boto3.resource("dynamodb")
sns = boto3.client("sns")

# Environment Variables
UPLOAD_BUCKET = "image-uploads-bucket-123456789012"
PROCESSED_BUCKET = "image-processed-bucket-123456789012"
DYNAMODB_TABLE = "ImageMetadata"
SNS_ARN = "arn:aws:sns:us-east-1:688567302802:ImageProcessingAlerts"

def lambda_handler(event, context):
    try:
        table = dynamodb.Table(DYNAMODB_TABLE)

        for record in event['Records']:
            bucket = record['s3']['bucket']['name']
            key = record['s3']['object']['key']
            timestamp = int(time.time())  # Get current timestamp
```

```
            print(f"Received file: {key}")  # Debugging log

            if not key:
                print("No key found in event.")
                return {"statusCode": 400, "body": "No file key provided"}

            print(f"Copying {key} from {bucket} to {PROCESSED_BUCKET}")

            # Copy the uploaded file to the processed bucket
            s3.copy_object(
                Bucket=PROCESSED_BUCKET,
                Key=key,
                CopySource={'Bucket': bucket, 'Key': key}
            )

            print(f"Storing metadata in {DYNAMODB_TABLE}")

            # Store metadata in DynamoDB (Fixing key name to match the table's
primary key)
            table.put_item(
                Item={
                    "imageId": key,  # ✅ Changed from "image_key" to "imageId"
                    "uploaded_at": timestamp,
                    "processed": "yes"
                }
            )

            print(f"Sending SNS Notification for {key}")

            # Send SNS Notification
            message = {
                "message": f"Image {key} processed successfully!",
                "imageId": key,
                "timestamp": timestamp
            }

            sns.publish(
                TopicArn=SNS_ARN,
                Message=json.dumps(message),
                Subject="Image Processing Complete"
            )

            return {"statusCode": 200, "body": f"File {key} processed
successfully"}
```

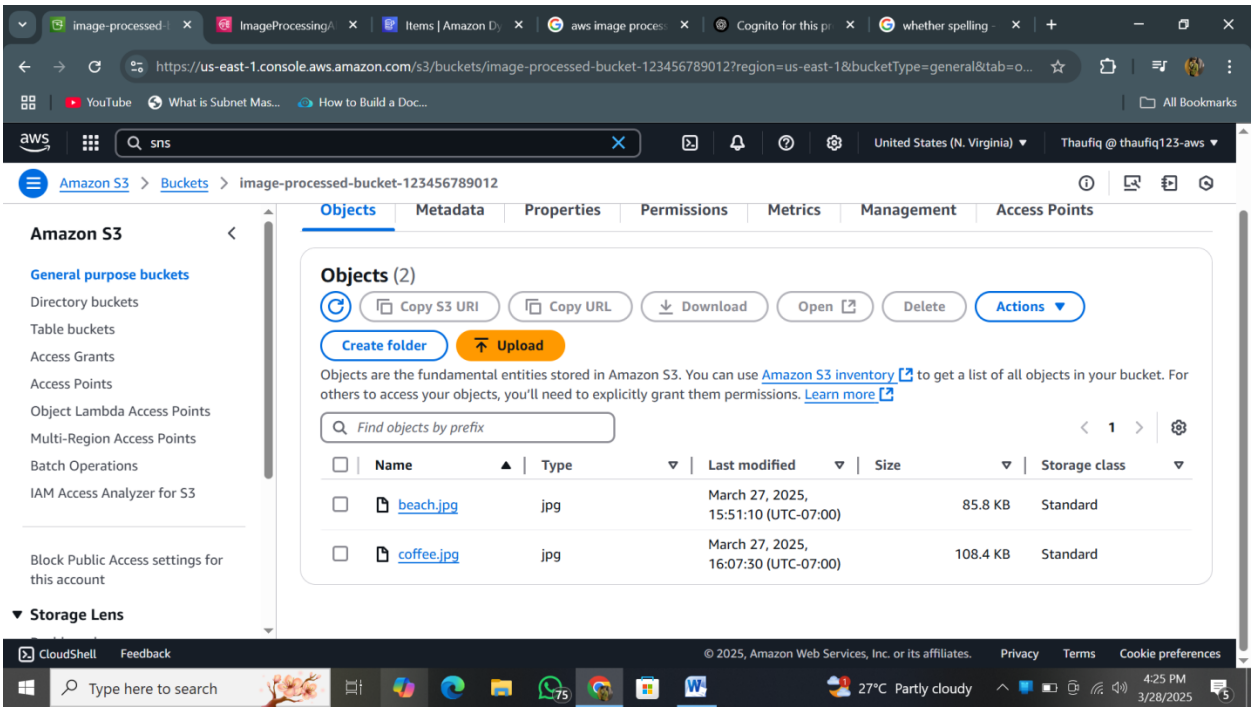
```
    except Exception as e:
        print(f"Error: {str(e)}")
        return {"statusCode": 500, "body": f"Error: {str(e)}"}
```

LET us check uploading images to s3 bucket



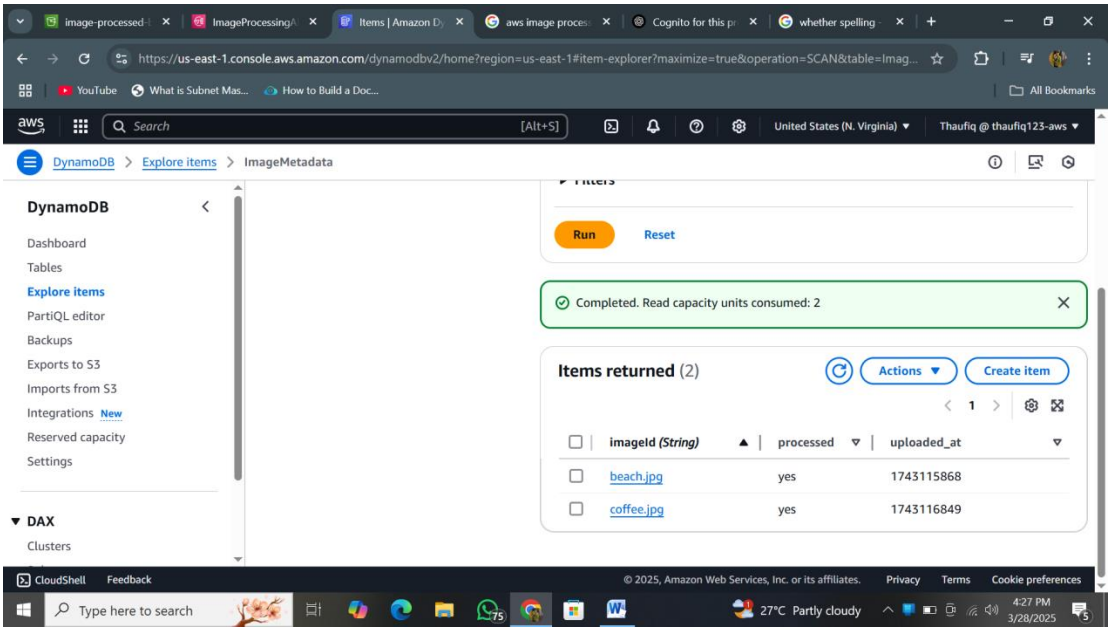
Uploaded two imges to bucket **`image-uploads-bucket-123456789012`**

Know we go check the **`image-processed-bucket-123456789012`** whether the uploaded images in the **`image-uploads-bucket-123456789012 are process and stored in image-processed-bucket-123456789012`**

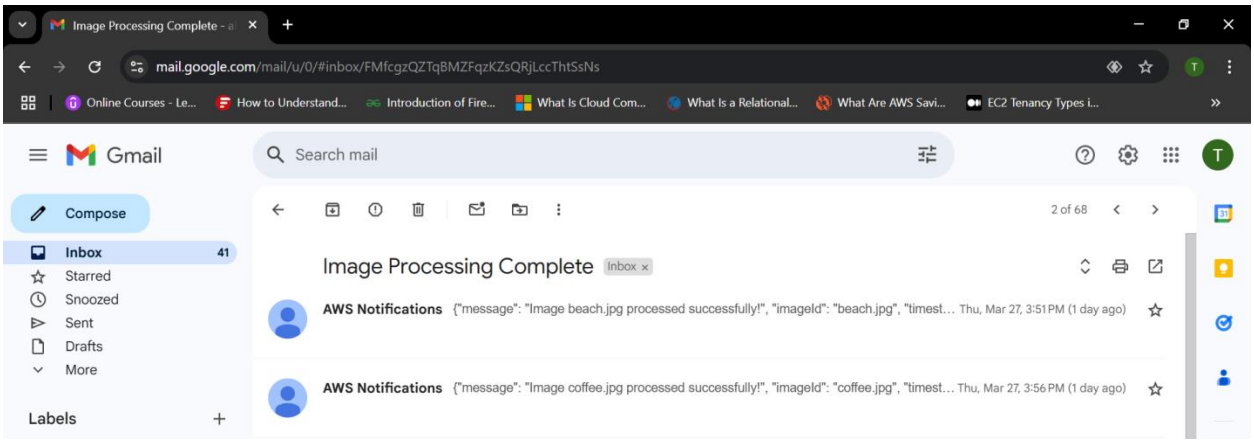


Perfect

Let us check dynamodb table



SNS



Challenges Faced & Solutions:

- **Issue:** Permissions error while writing to S3 and DynamoDB.
  - **Solution:** Updated IAM policies for Lambda to allow access to S3, DynamoDB, and SNS.
- **Issue:** SNS notification not received.
  - **Solution:** Verified email subscription confirmation for the SNS topic.

Conclusion: This project successfully automates the movement of images between S3 buckets using AWS Lambda, while also storing metadata in DynamoDB and sending notifications via SNS. This approach ensures scalability and real-time updates in a fully serverless manner.