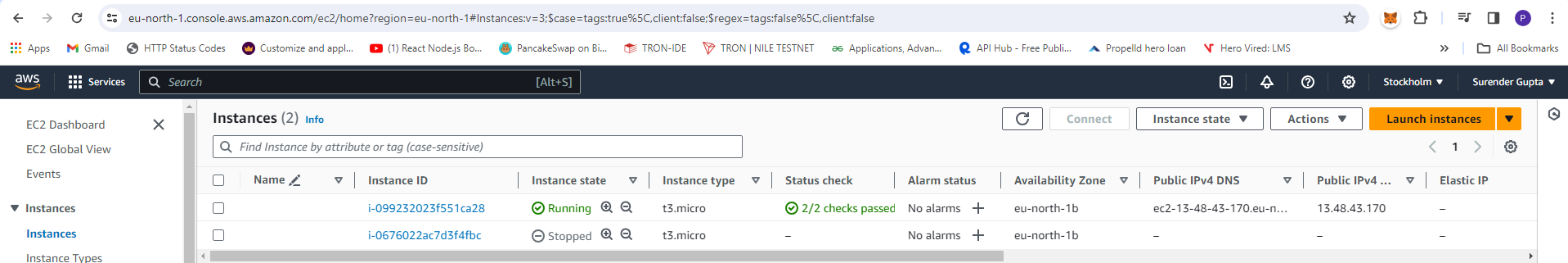
**Assignment On Serverless Architecture**

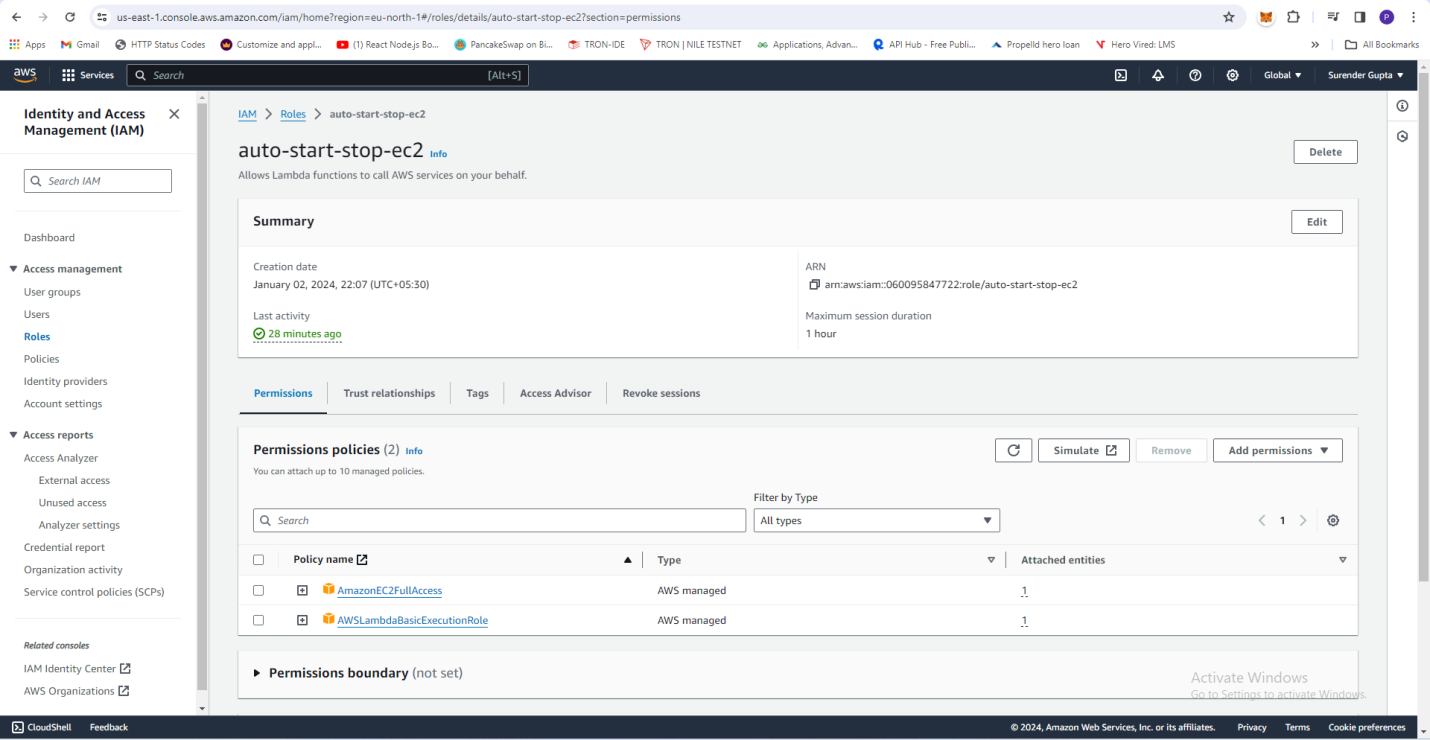
**Assignment 1: Automated Instance Management Using AWS Lambda and Boto3**

**Solution1.**

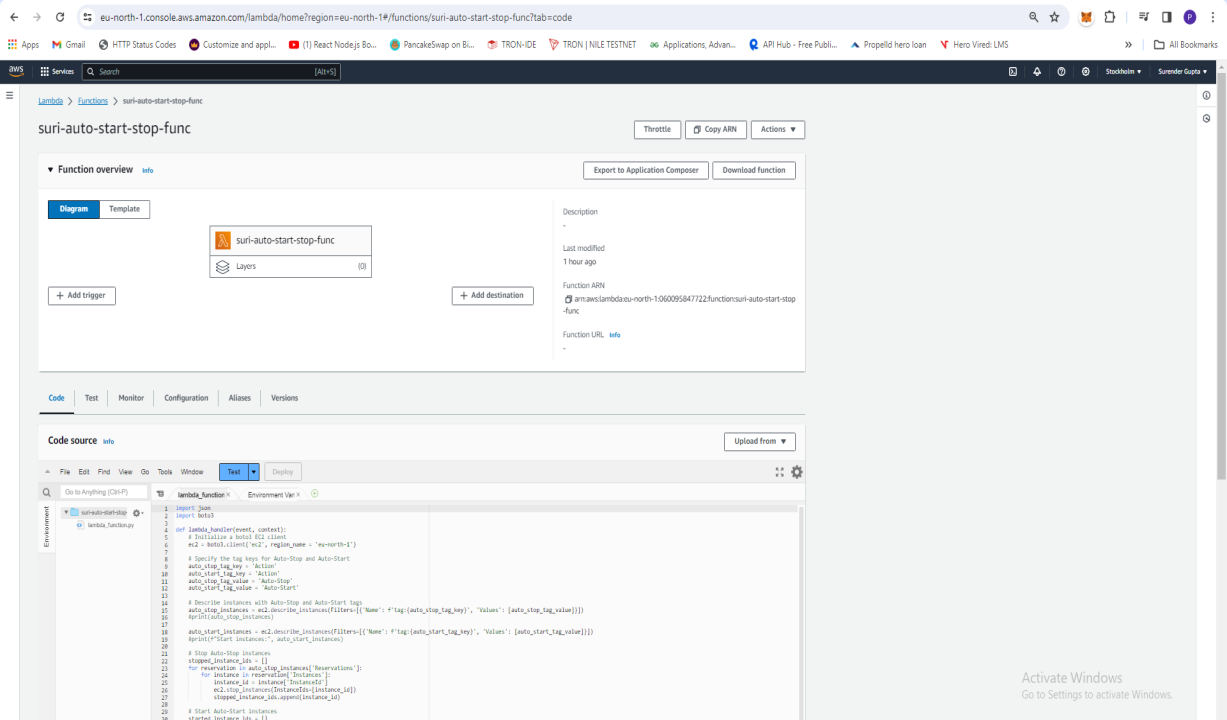
1. First Create Two EC2 Instance
2. Steps are below for creating EC2 Instance.
3. In Section of **Name and tags** set **Key** is **Action** and **Value** is **Auto-Start**
4. In Section of **Application and OS Images (Amazon Machine Image)** select **Quick Start tab** and **Ubuntu** option after select **Amazon Machine Image (AMI)** select any option as required but we are select **Ubuntu Server 22.04 LTS (HVM), SSD Volume Type** now **Architecture** option select **64-Bit (x86)**
5. In Section of **Instance type** select option as you need but we are choose **t3.micro**
6. In Section of **Key pair (login)** if you not have any key pair create new key pair but we have already created key pair just selecting option of my **auto-start-stop**
7. In Section of **Network settings** you are **just create new security group** or **existing security group** select as you need but we are **create security group** also check **Allow SSH traffic from** option of IP Address but we are selecting **Anywhere 0.0.0.0/0** also checkbox tick on **Allow HTTPS traffic from the internet** and **Allow HTTP traffic from the internet**
8. In Section of **Configure storage** and **Advanced details** do not need to set anything required in this project.
9. In Section of **Summary** if required set **Number of instances** as much you required type but we set **2** for now
10. Now click on **Launch instance** button.
11. After Success message , click on instance-id in message
12. Back to Instance list page edit tag name **Auto-Start** change to **Auto-Stop**.
13. Two Instances are Screenshot below:



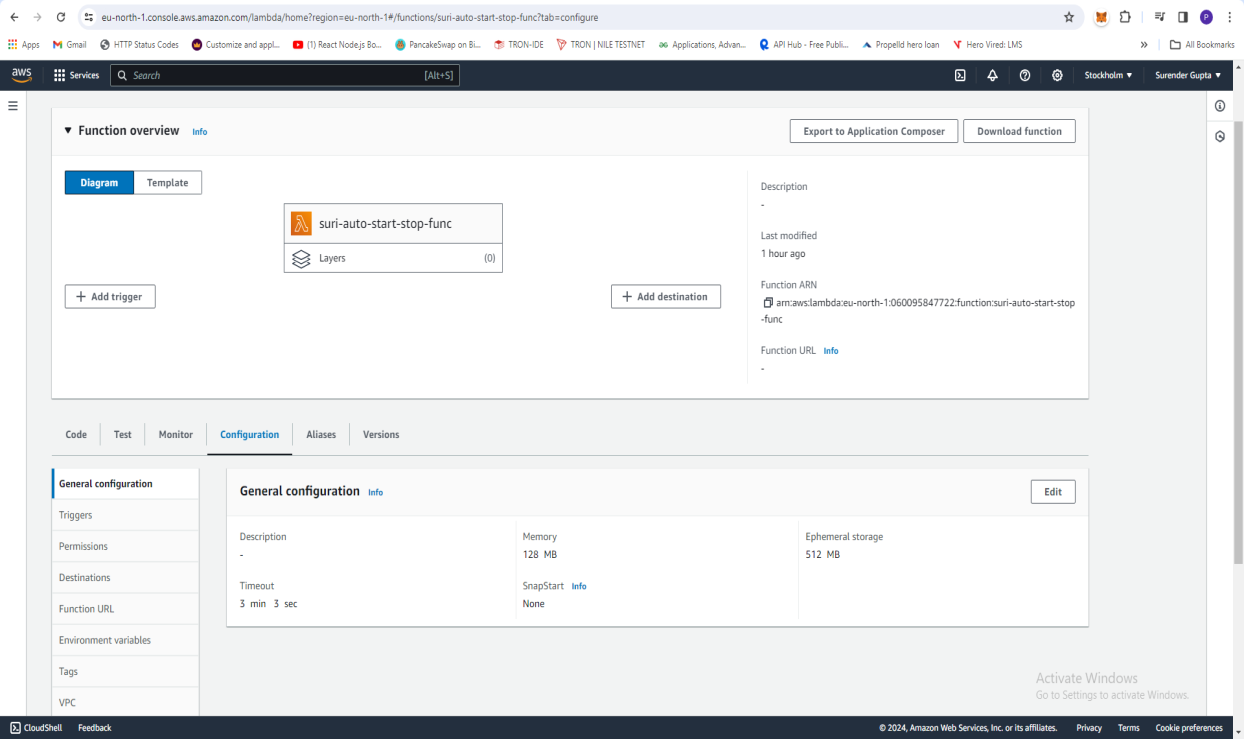
1. Create New Role for Lambda Function
2. First go to **IAM Dashboard** left sidebar under **Access Management -> Role** click it.
3. Create new Role button then **Trusted entity type** AWS Service and Use Case Lambda and click on Next Button
4. Search permission policy **AmazonEC2FullAccess** Checked it and **AWSLambdaBasicExecutionRole** also checked it click on Next
5. Enter Role Name **auto-start-stop-ec2** and click on create role button at bottom.



1. Create Lambda Function
2. Lambda Dashboard left sidebar **Function** click it
3. Click on **Create function** after choose **Author from Scratch**
4. Function Name **suri-auto-start-stop-func** and Runtime select **Python 3.8** and Architecture **x86\_64** selected.
5. **Change default execution role** Opted **Use an existing role** and select **Existing role** is previously created **auto-start-stop-ec2** and click on **create function** at bottom.



1. Going to Configuration Tab Edit Timeout to 3 second to add Minute.

****

1. Lambda function below:

import json

import boto3

def lambda\_handler(event, context):

# Initialize a boto3 EC2 client

ec2 = boto3.client('ec2', region\_name = 'eu-north-1')

# Specify the tag keys for Auto-Stop and Auto-Start

auto\_stop\_tag\_key = 'Action'

auto\_start\_tag\_key = 'Action'

auto\_stop\_tag\_value = 'Auto-Stop'

auto\_start\_tag\_value = 'Auto-Start'

# Describe instances with Auto-Stop and Auto-Start tags

auto\_stop\_instances = ec2.describe\_instances(Filters=[{'Name': f'tag:{auto\_stop\_tag\_key}', 'Values': [auto\_stop\_tag\_value]}])

auto\_start\_instances = ec2.describe\_instances(Filters=[{'Name': f'tag:{auto\_start\_tag\_key}', 'Values': [auto\_start\_tag\_value]}])

# Stop Auto-Stop instances

stopped\_instance\_ids = []

for reservation in auto\_stop\_instances['Reservations']:

for instance in reservation['Instances']:

instance\_id = instance['InstanceId']

ec2.stop\_instances(InstanceIds=[instance\_id])

stopped\_instance\_ids.append(instance\_id)

# Start Auto-Start instances

started\_instance\_ids = []

for reservation in auto\_start\_instances['Reservations']:

for instance in reservation['Instances']:

instance\_id = instance['InstanceId']

ec2.start\_instances(InstanceIds=[instance\_id])

started\_instance\_ids.append(instance\_id)

# Print affected instance IDs for logging purposes

print("Stopped instances:", stopped\_instance\_ids)

print("Started instances:", started\_instance\_ids)

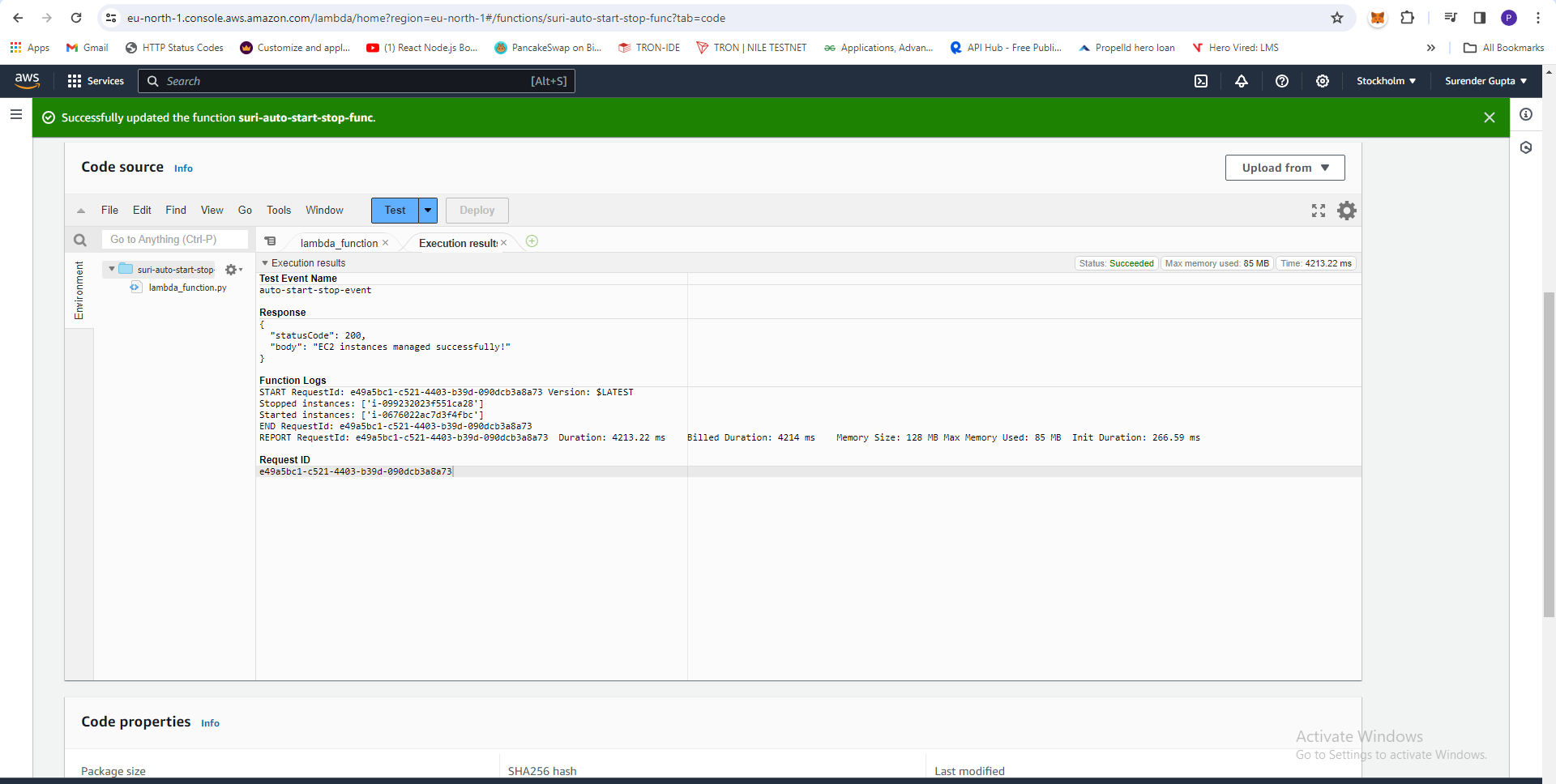
return {

'statusCode': 200,

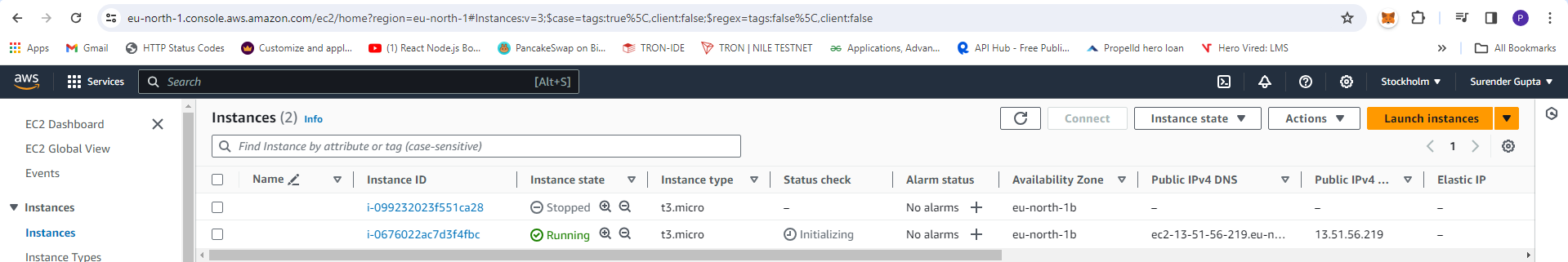
'body': 'EC2 instances managed successfully!'

}

1. Click on Deploy to save the python script and then click on Test script
2. After Test result show its Output below:



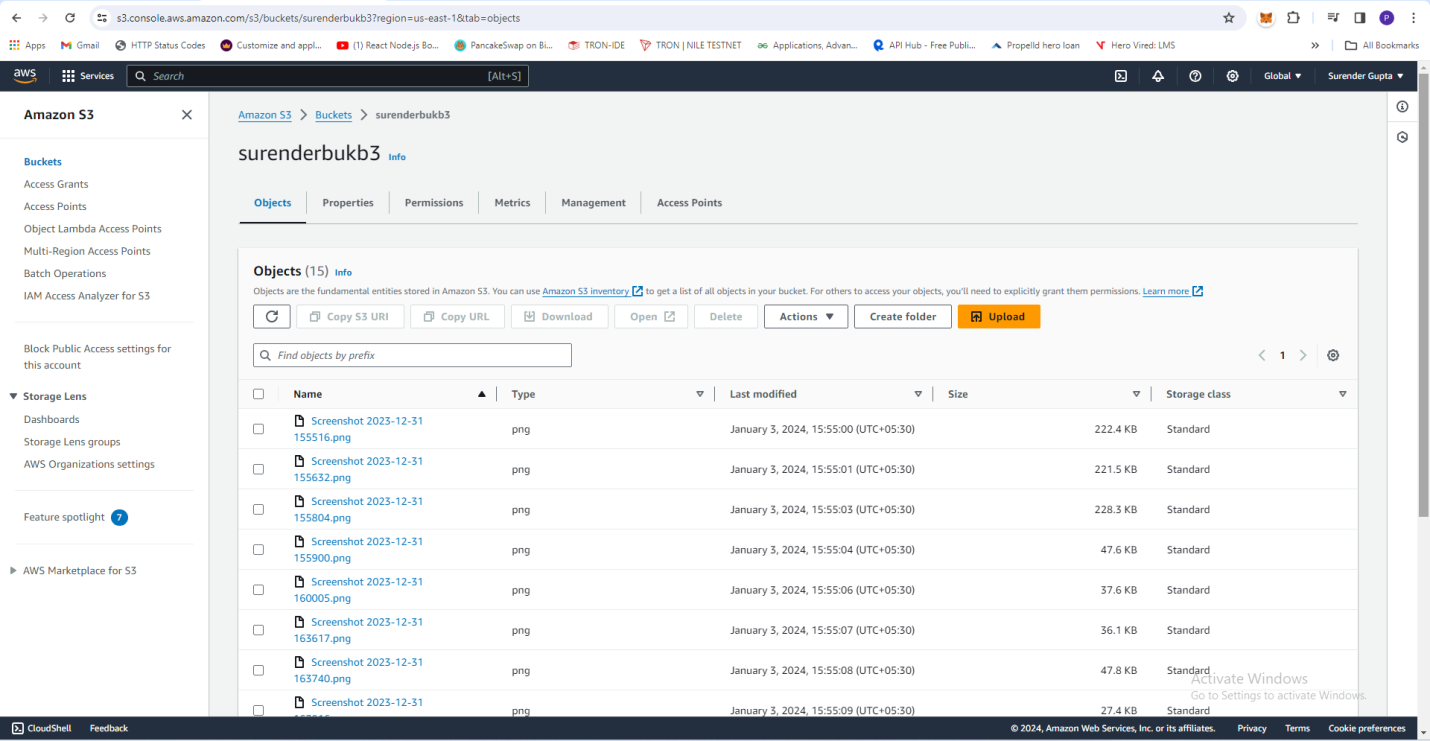
1. EC2 Instance update its state see below:



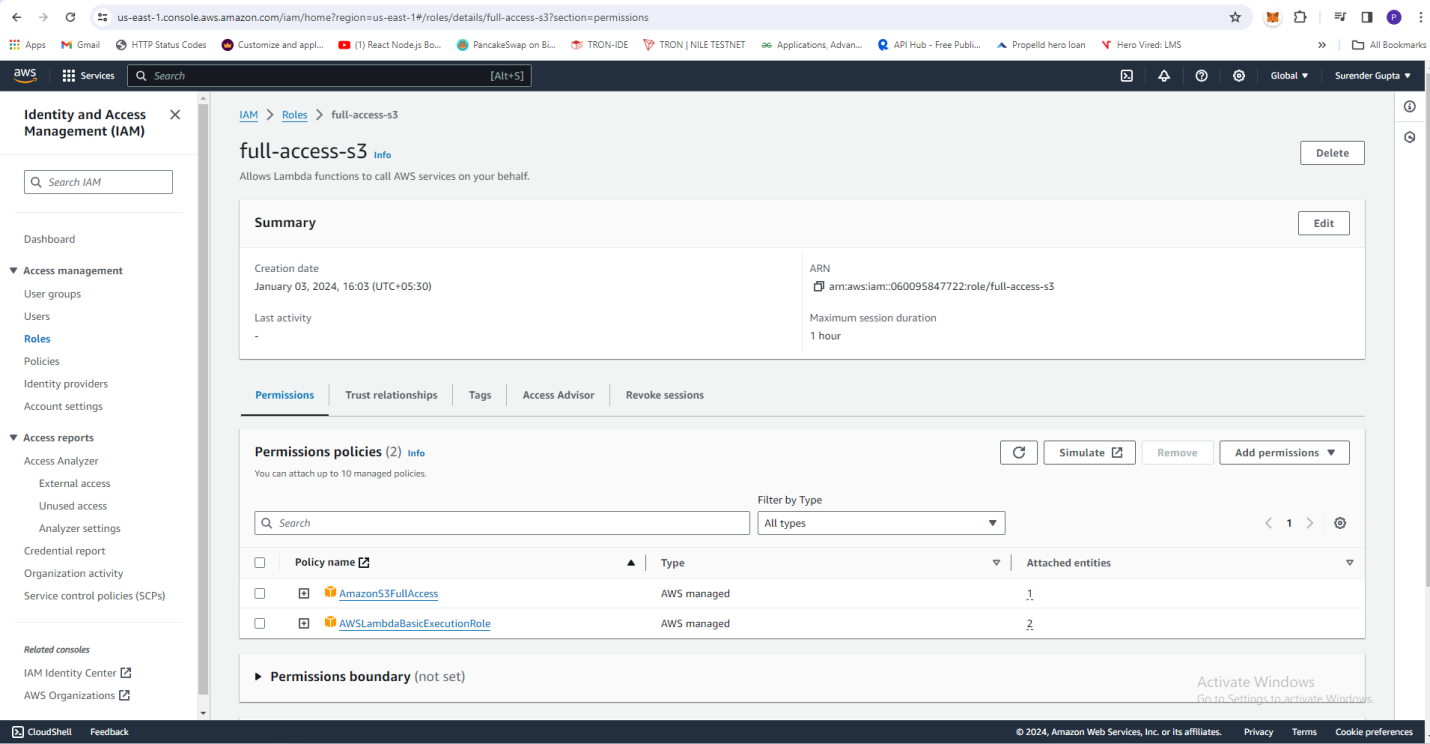
**Assignment 2: Automated S3 Bucket Cleanup Using AWS Lambda and Boto3**

**Solution2.**

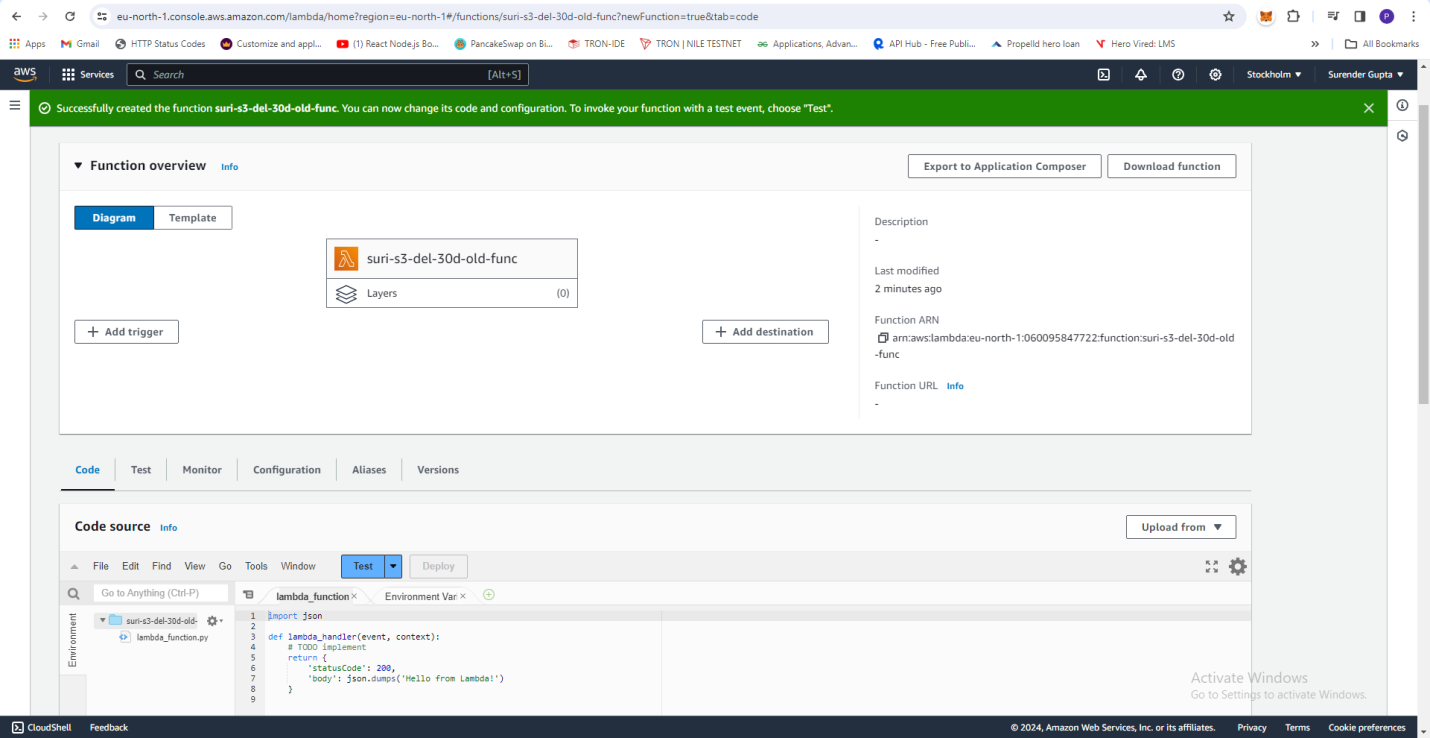
1. Create S3 bucket steps are Below:
2. Click on create bucket Button and you see Create Bucket Page.
3. AWS Region **US East (N. Virginia) us-east-1** and Bucket Type **General Purpose** opted
4. Bucket Name **surenderbukb3** and uncheck **Block all public access** and check **I acknowledge that the current settings might result in this bucket and the objects within becoming public**.
5. All other option rest default and click **Create bucket** button at bottom.
6. After click its show Successfully created bucket "surenderbukb3"
7. Click on list of bucket search surenderbukb3 click it
8. Upload files or images some are back date 30 days older



1. Create New Role for Lambda Function
2. First go to **IAM Dashboard** left sidebar under **Access Management -> Role** click it.
3. Create new Role button then **Trusted entity type** AWS Service and Use Case Lambda and click on Next Button
4. Search permission policy **AmazonS3FullAccess** Checked it and **AWSLambdaBasicExecutionRole** also checked it click on Next
5. Enter Role Name **full-access-s3**and click on create role button at bottom.



1. Create Lambda Function
2. Lambda Dashboard left sidebar **Function** click it
3. Click on **Create function** after choose **Author from Scratch**
4. Function Name **suri-s3-del-30d-old-func** and Runtime select **Python 3.10** and Architecture **x86\_64** selected.
5. **Change default execution role** Opted **Use an existing role** and select **Existing role** is previously created **full-access-s3** and click on **create function** at bottom.



1. Python Script below:

import boto3

from datetime import datetime, timedelta, timezone

def lambda\_handler(event, context):

# Replace 'your-bucket-name' with the actual name of your S3 bucket

bucket\_name = 'surenderbukb3'

# Initialize a boto3 S3 client

s3\_client = boto3.client('s3')

# List objects in the specified bucket

objects = s3\_client.list\_objects(Bucket=bucket\_name)['Contents']

# Calculate the date threshold for objects older than 30 days

threshold\_date = datetime.now(timezone.utc) - timedelta(days=30)

# Delete objects older than 30 days

deleted\_objects = []

for obj in objects:

last\_modified = obj['LastModified'].replace(tzinfo=timezone.utc)

if last\_modified < threshold\_date:

s3\_client.delete\_object(Bucket=bucket\_name, Key=obj['Key'])

deleted\_objects.append(obj['Key'])

# Print the names of deleted objects for logging purposes

if deleted\_objects:

print(f"Deleted objects: {deleted\_objects}")

else:

print("No objects deleted.")

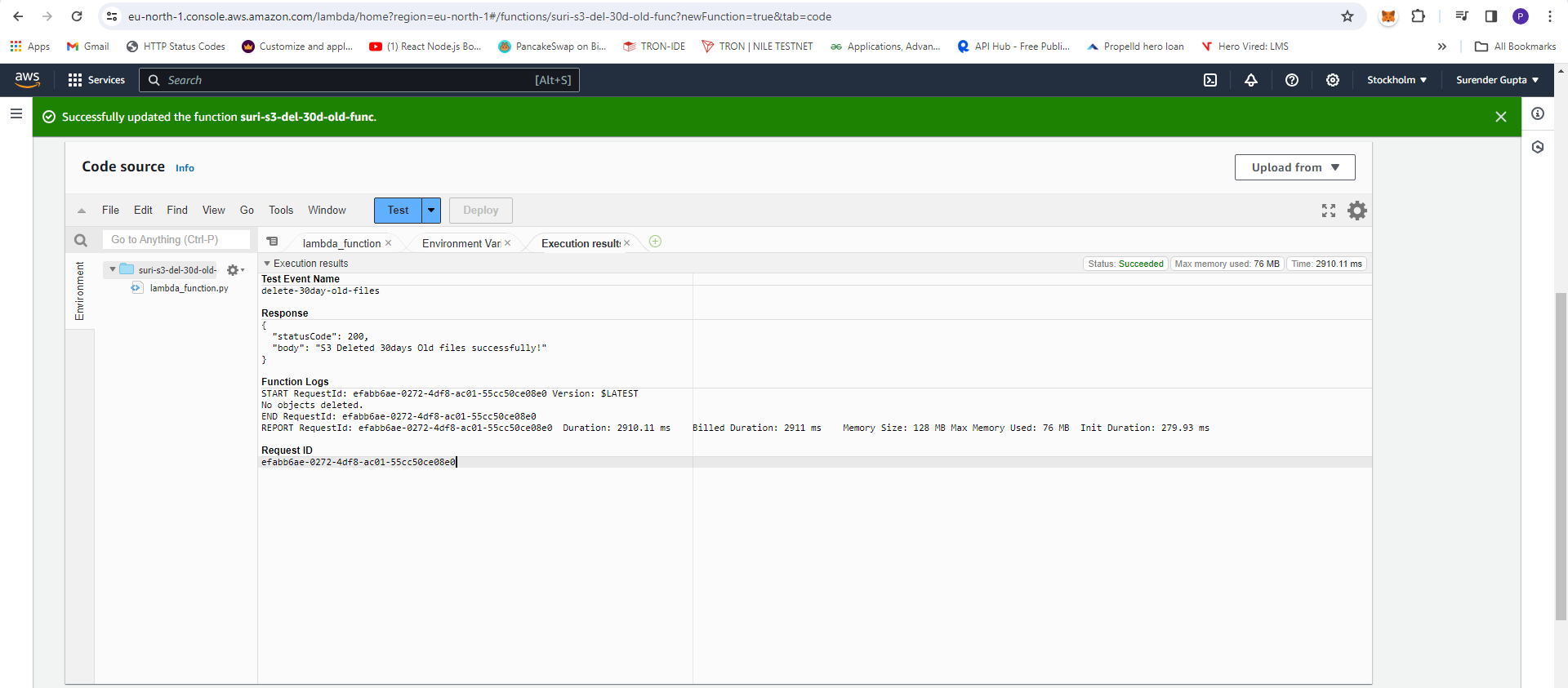
return {

'statusCode': 200,

'body': 'S3 Deleted 30days Old files successfully!'

}

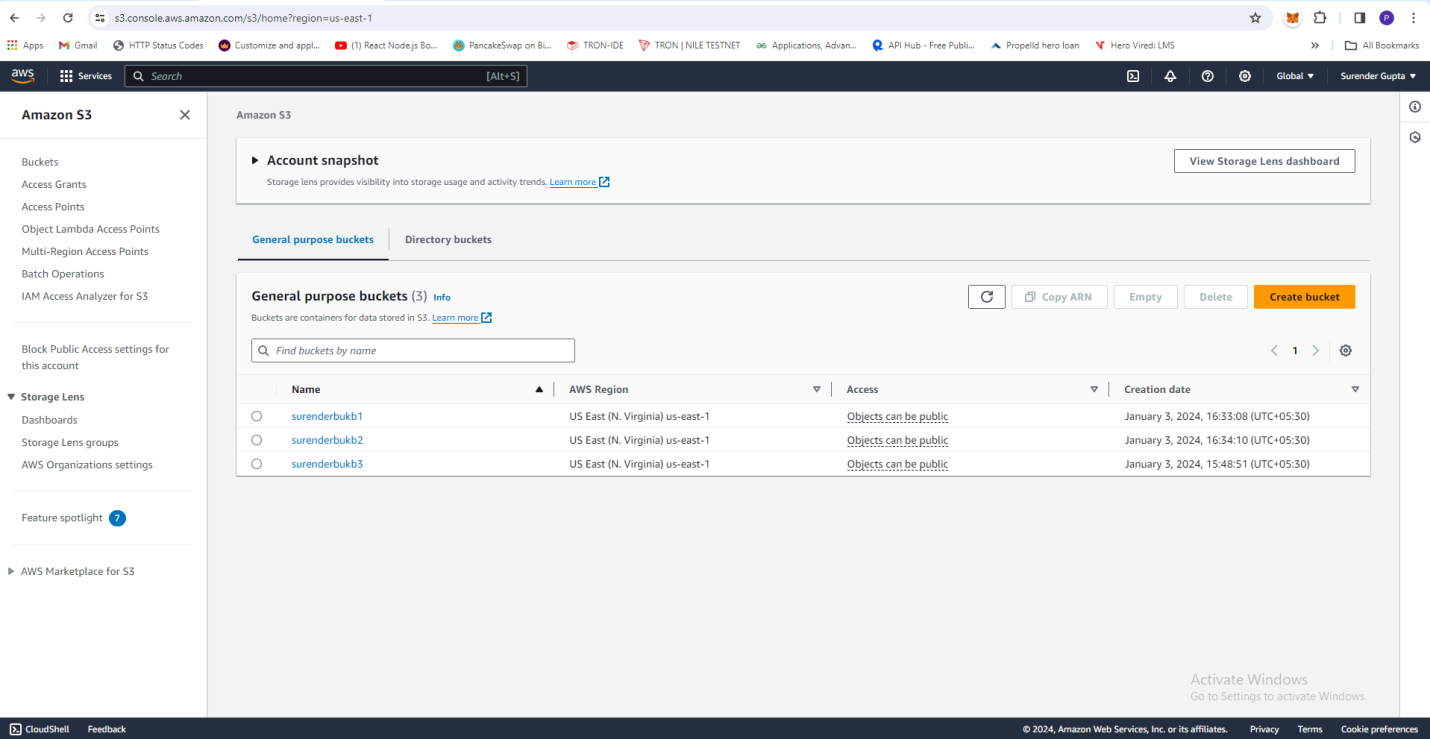
1. Above code paste in Lambda function and deploy and then Test script manually.



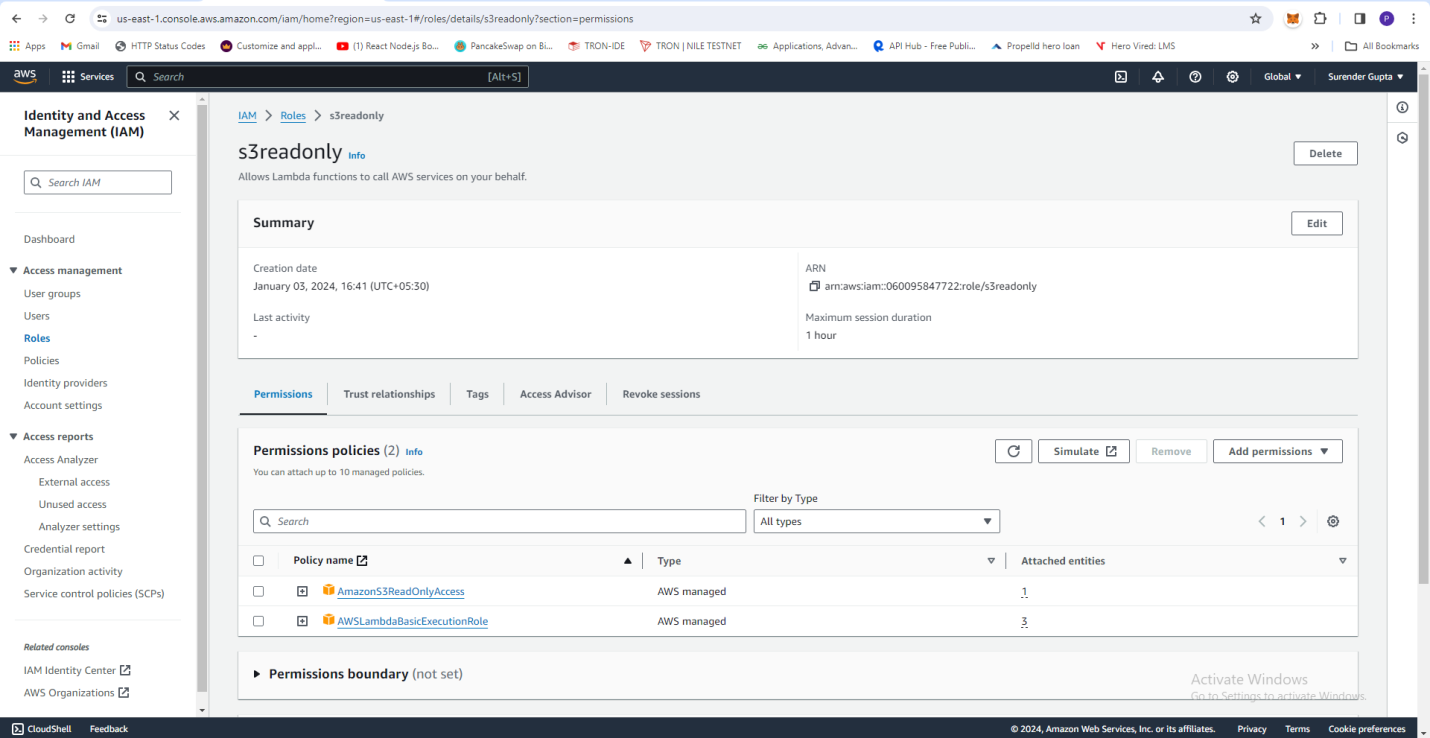
**Assignment 4: Monitor Unencrypted S3 Buckets Using AWS Lambda and Boto3**

**Solution2.**

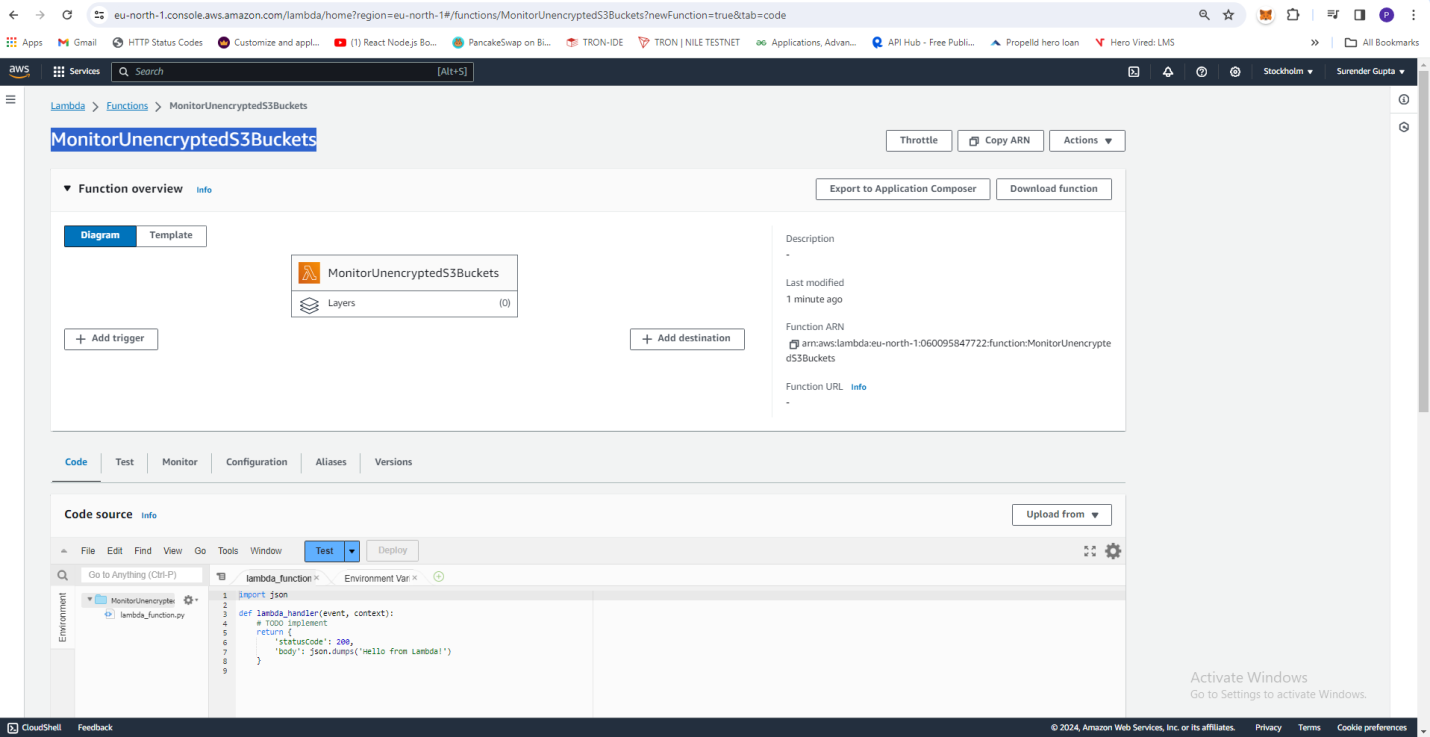
1. Create two or three S3 bucket steps are Below:
2. Click on create bucket Button and you see Create Bucket Page.
3. AWS Region **US East (N. Virginia) us-east-1** and Bucket Type **General Purpose** opted
4. Bucket Name **surenderbukb1, surenderbukb2** and uncheck **Block all public access** and check **I acknowledge that the current settings might result in this bucket and the objects within becoming public**.
5. All other option rest default and click **Create bucket** button at bottom.
6. After click its show Successfully created bucket "surenderbukb1", "surenderbukb2"
7. While create bucket **surenderbukb2** disable **Server-side encryption with Amazon S3** bucket key **Disable** opted



1. Create New Role for Lambda Function
2. First go to **IAM Dashboard** left sidebar under **Access Management -> Role** click it.
3. Create new Role button then **Trusted entity type** AWS Service and Use Case Lambda and click on Next Button
4. Search permission policy **AmazonS3ReadOnlyAccess** Checked it and **AWSLambdaBasicExecutionRole** also checked it click on Next
5. Enter Role Name **s3readonly** and click on create role button at bottom.



1. Create Lambda Function
2. Lambda Dashboard left sidebar **Function** click it
3. Click on **Create function** after choose **Author from Scratch**
4. Function Name **MonitorUnencryptedS3Buckets** and Runtime select **Python 3.10** and Architecture **x86\_64** selected.
5. **Change default execution role** Opted **Use an existing role** and select **Existing role** is previously created **s3readonly** and click on **create function** at bottom.



1. Python Script below:

import boto3

def lambda\_handler(event, context):

# Initialize a boto3 S3 client

s3\_client = boto3.client('s3', region\_name = 'us-east-1')

# List all S3 buckets

s3\_buckets = s3\_client.list\_buckets()['Buckets']

# Detect buckets without server-side encryption

unencrypted\_buckets = []

for bucket in s3\_buckets:

bucket\_name = bucket['Name']

#print(bucket\_name)

try:

bucket\_encryption = s3\_client.get\_bucket\_encryption(Bucket=bucket\_name)

server\_side\_encryption = bucket\_encryption.get('ServerSideEncryptionConfiguration', [])

for rules in server\_side\_encryption['Rules']:

#print(rules['BucketKeyEnabled'])

if rules['BucketKeyEnabled'] == False:

unencrypted\_buckets.append(bucket\_name)

#rule = rules['Rules']

#print(rule)

#if not server\_side\_encryption:

#unencrypted\_buckets.append(bucket\_name)

except s3\_client.exceptions.ClientError as e:

# Handle the exception if the bucket does not have server-side encryption configured

if e.response['Error']['Code'] == 'ServerSideEncryptionConfigurationNotFoundError':

unencrypted\_buckets.append(bucket\_name)

# Print the names of unencrypted buckets for logging purposes

if unencrypted\_buckets:

print(f"Unencrypted buckets: {unencrypted\_buckets}")

else:

print("All buckets are encrypted.")

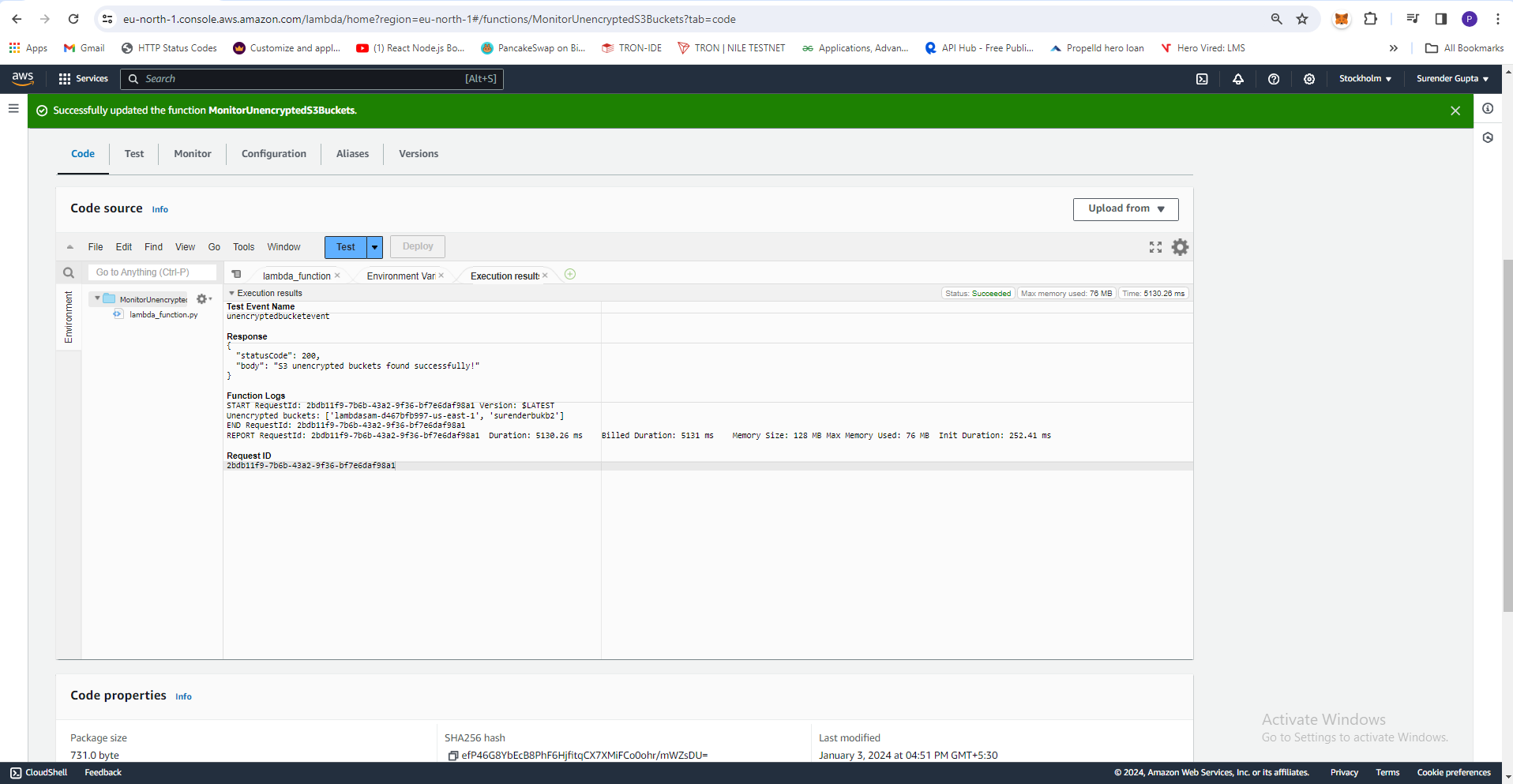
return {

'statusCode': 200,

'body': 'S3 unencrypted buckets found successfully!'

}

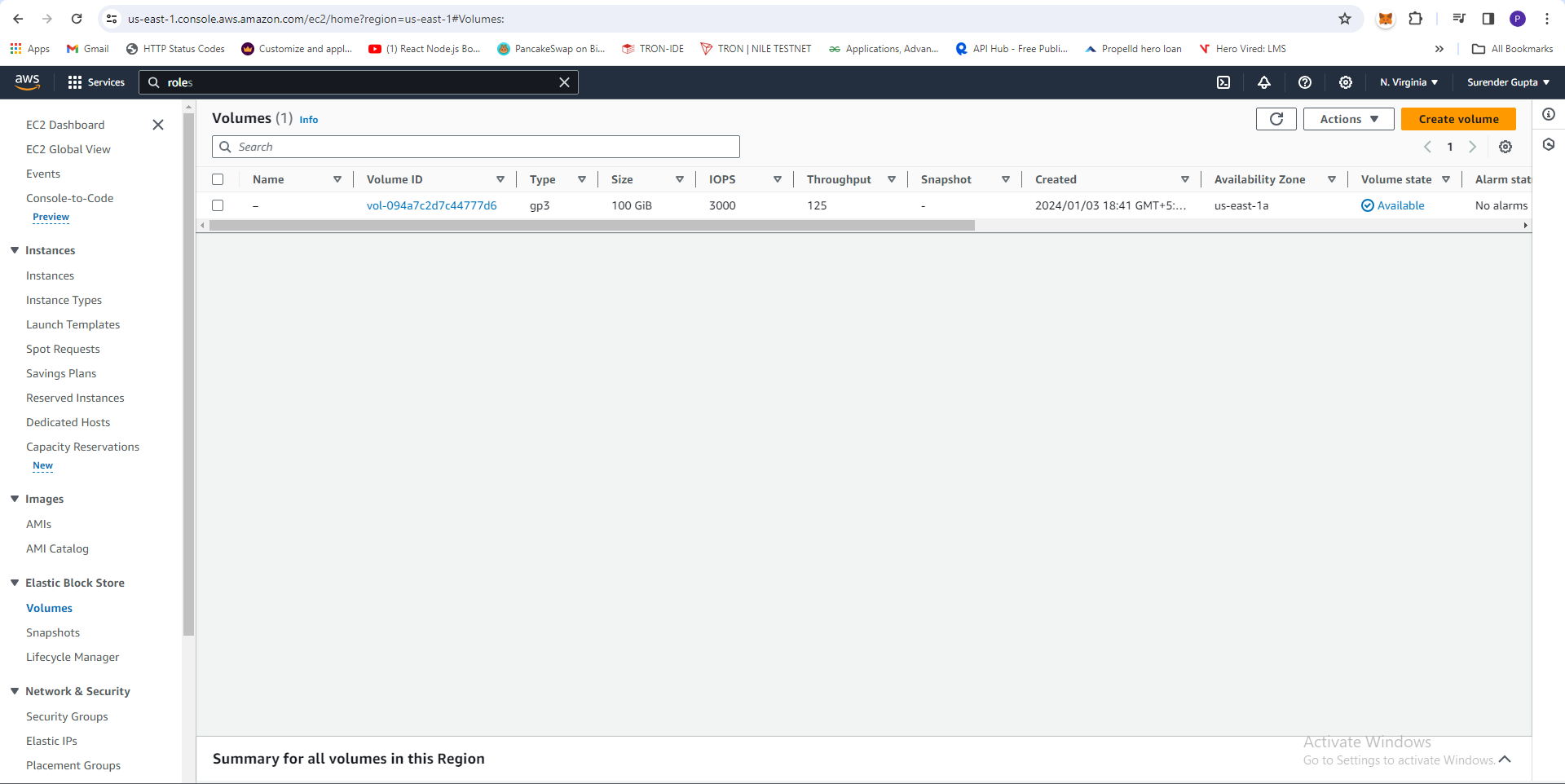
1. Above code paste in Lambda function and deploy and then Test script manually.



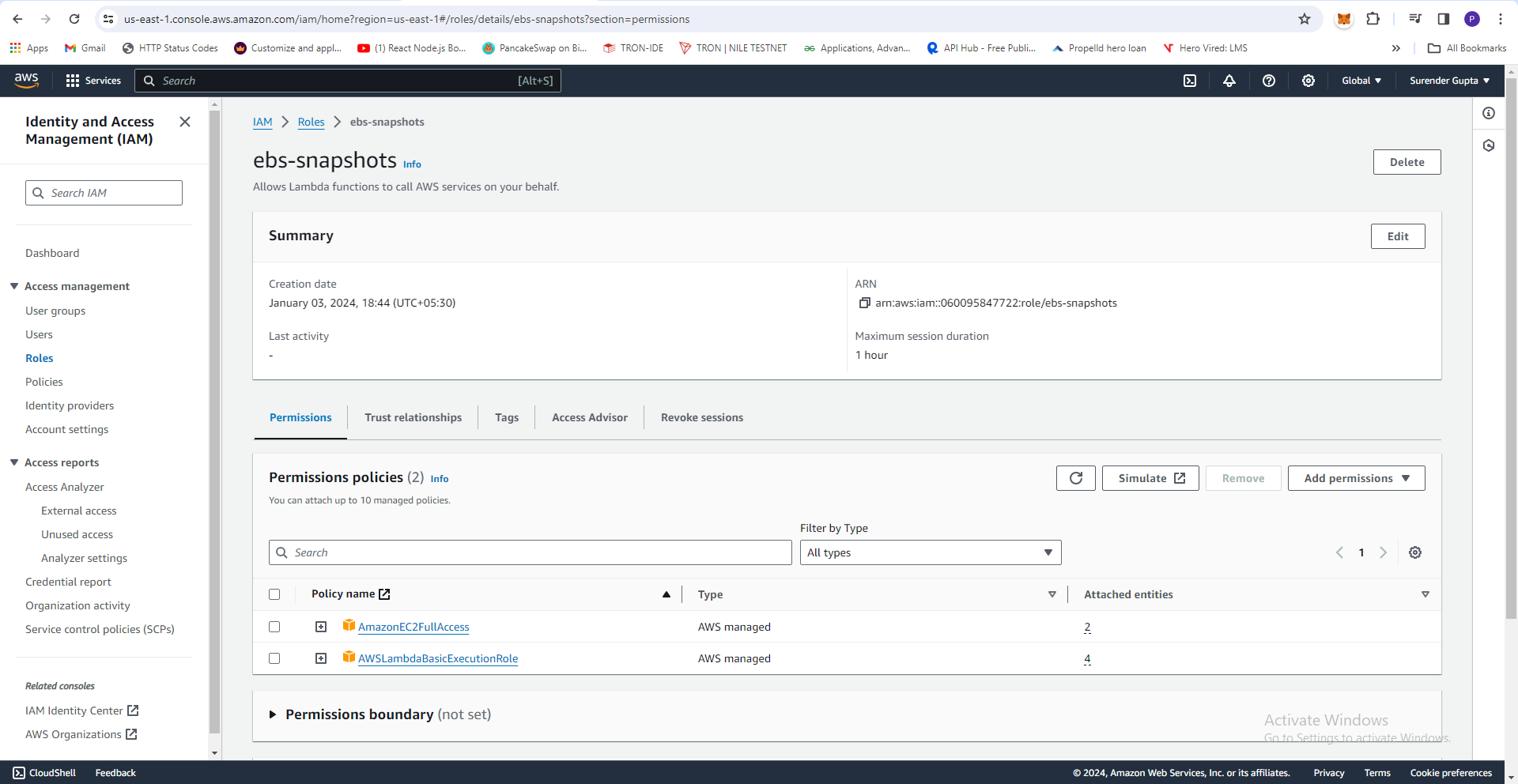
**Assignment 5: Automatic EBS Snapshot and Cleanup Using AWS Lambda and Boto3**

**Solution2.**

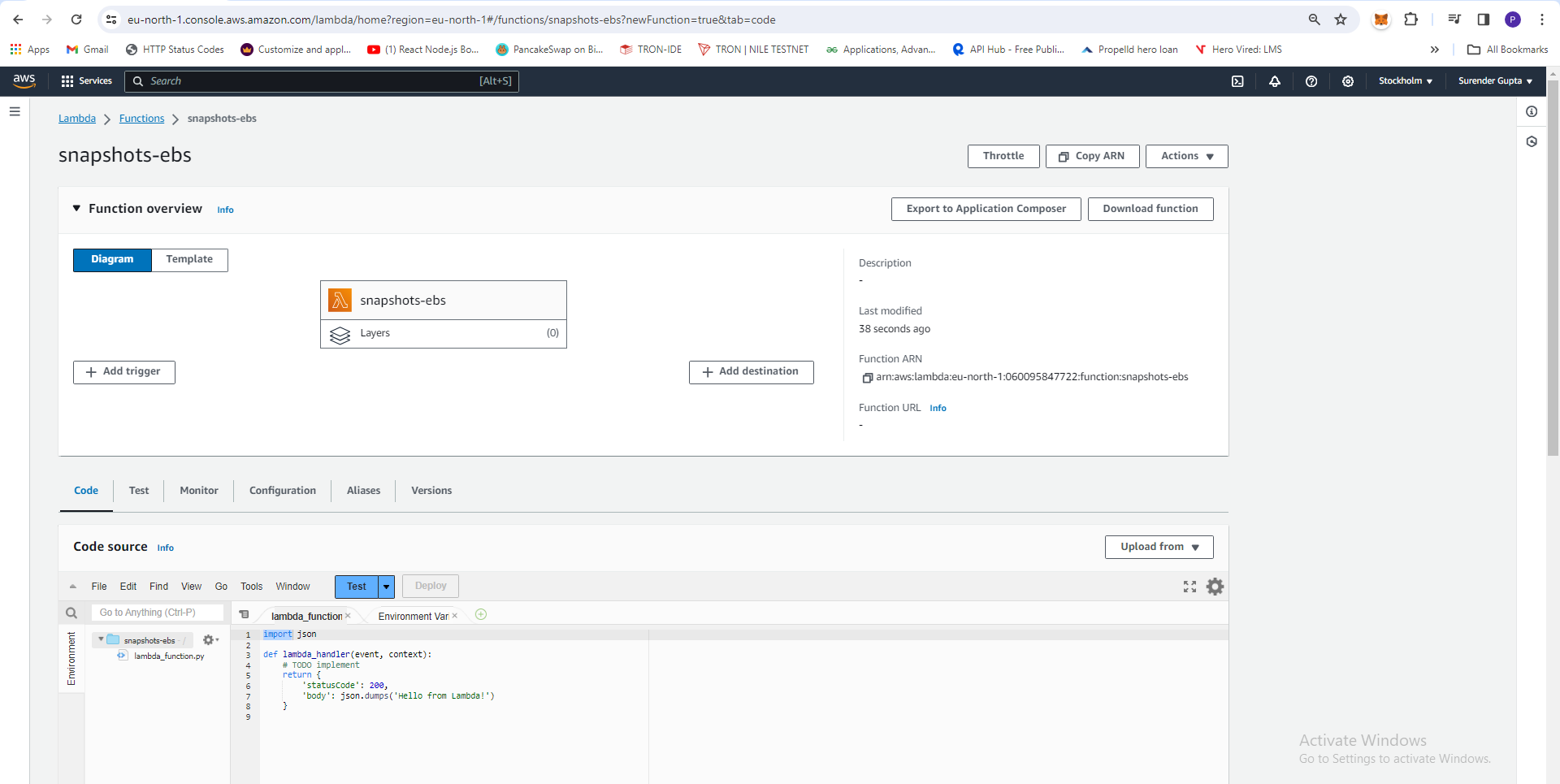
1. **Create Volume go to EC2 Dashboard left sidebar pane Elastic Block Store under the volumes click on it.**
2. **Now click on Create Volume button and every option set default as you see just click on Create volume at bottom.**
3. **After Create successfully volume it volume ID show copy volume ID.**



1. Create New Role for Lambda Function
2. First go to **IAM Dashboard** left sidebar under **Access Management -> Role** click it.
3. Create new Role button then **Trusted entity type** AWS Service and Use Case Lambda and click on Next Button
4. Search permission policy **AmazonEC2FullAccess** Checked it and **AWSLambdaBasicExecutionRole** also checked it click on Next
5. Enter Role Name **ebs-snapshots** and click on create role button at bottom.



1. Create Lambda Function
2. Lambda Dashboard left sidebar **Function** click it
3. Click on **Create function** after choose **Author from Scratch**
4. Function Name **snapshots-ebs** and Runtime select **Python 3.10** and Architecture **x86\_64** selected.
5. **Change default execution role** Opted **Use an existing role** and select **Existing role** is previously created **ebs-snapshots** and click on **create function** at bottom.



1. Python Script below:

import boto3

from datetime import datetime, timedelta, timezone

def lambda\_handler(event, context):

# Replace 'your-volume-id' with the actual ID of your EBS volume

volume\_id = 'vol-094a7c2d7c44777d6'

# Initialize a boto3 EC2 client

ec2\_client = boto3.client('ec2', region\_name='us-east-1')

# Create a snapshot for the specified EBS volume

snapshot\_response = ec2\_client.create\_snapshot(

VolumeId=volume\_id,

Description=f"EBS Snapshot for volume {volume\_id}"

)

snapshot\_id = snapshot\_response['SnapshotId']

# List snapshots for the specified EBS volume

snapshots = ec2\_client.describe\_snapshots(Filters=[{'Name': 'volume-id', 'Values': [volume\_id]}])['Snapshots']

# Calculate the date threshold for snapshots older than 30 days

threshold\_date = datetime.now(timezone.utc) - timedelta(days=30)

# Delete snapshots older than 30 days

deleted\_snapshots = []

for snapshot in snapshots:

# Convert the snapshot start time to an aware datetime object

snapshot\_start\_time = snapshot['StartTime'].replace(tzinfo=timezone.utc)

if snapshot\_start\_time < threshold\_date:

ec2\_client.delete\_snapshot(SnapshotId=snapshot['SnapshotId'])

deleted\_snapshots.append(snapshot['SnapshotId'])

# Print the IDs of the created and deleted snapshots for logging purposes

print(f"Created snapshot: {snapshot\_id}")

if deleted\_snapshots:

print(f"Deleted snapshots: {deleted\_snapshots}")

else:

print("No snapshots deleted.")

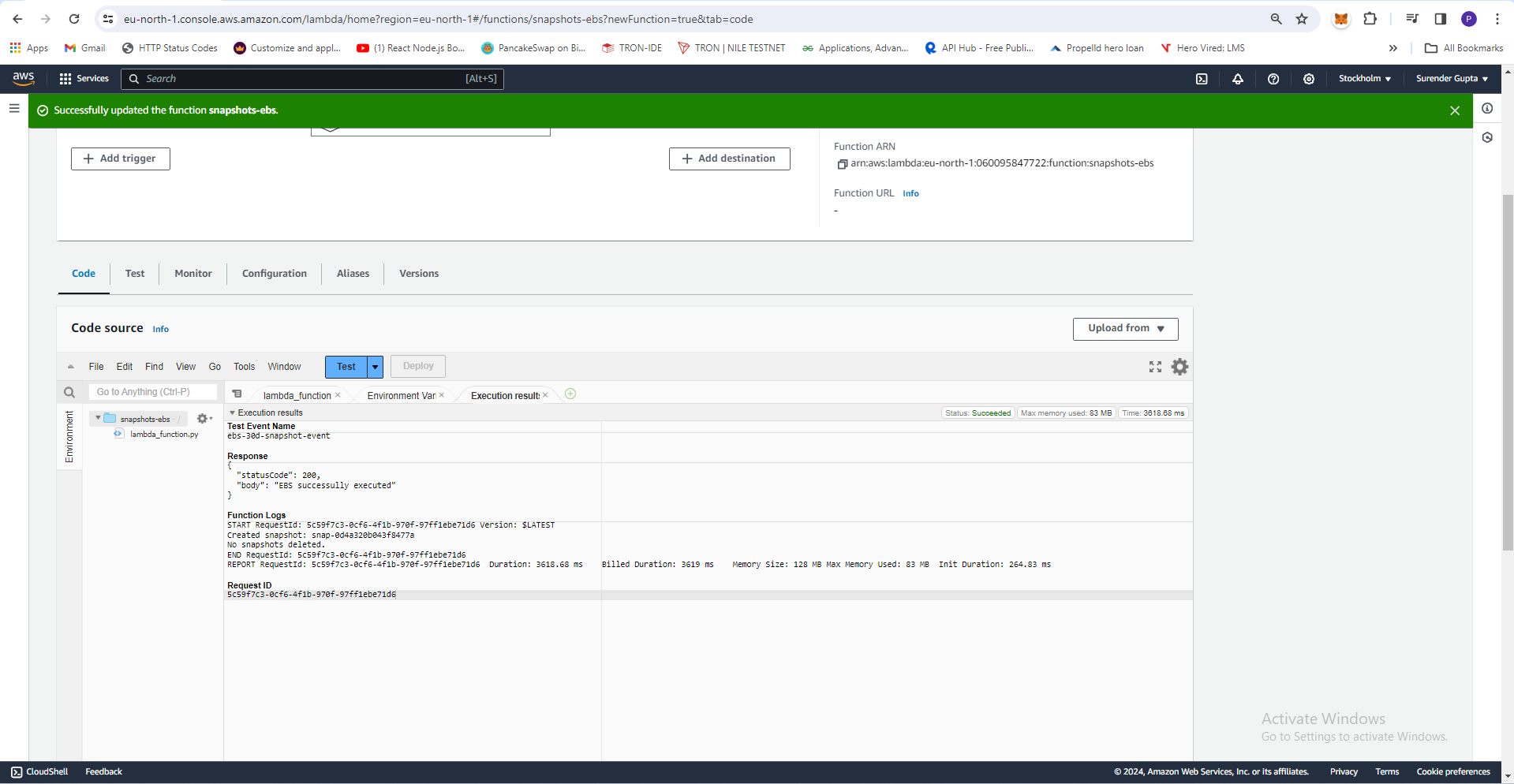
return {

'statusCode': 200,

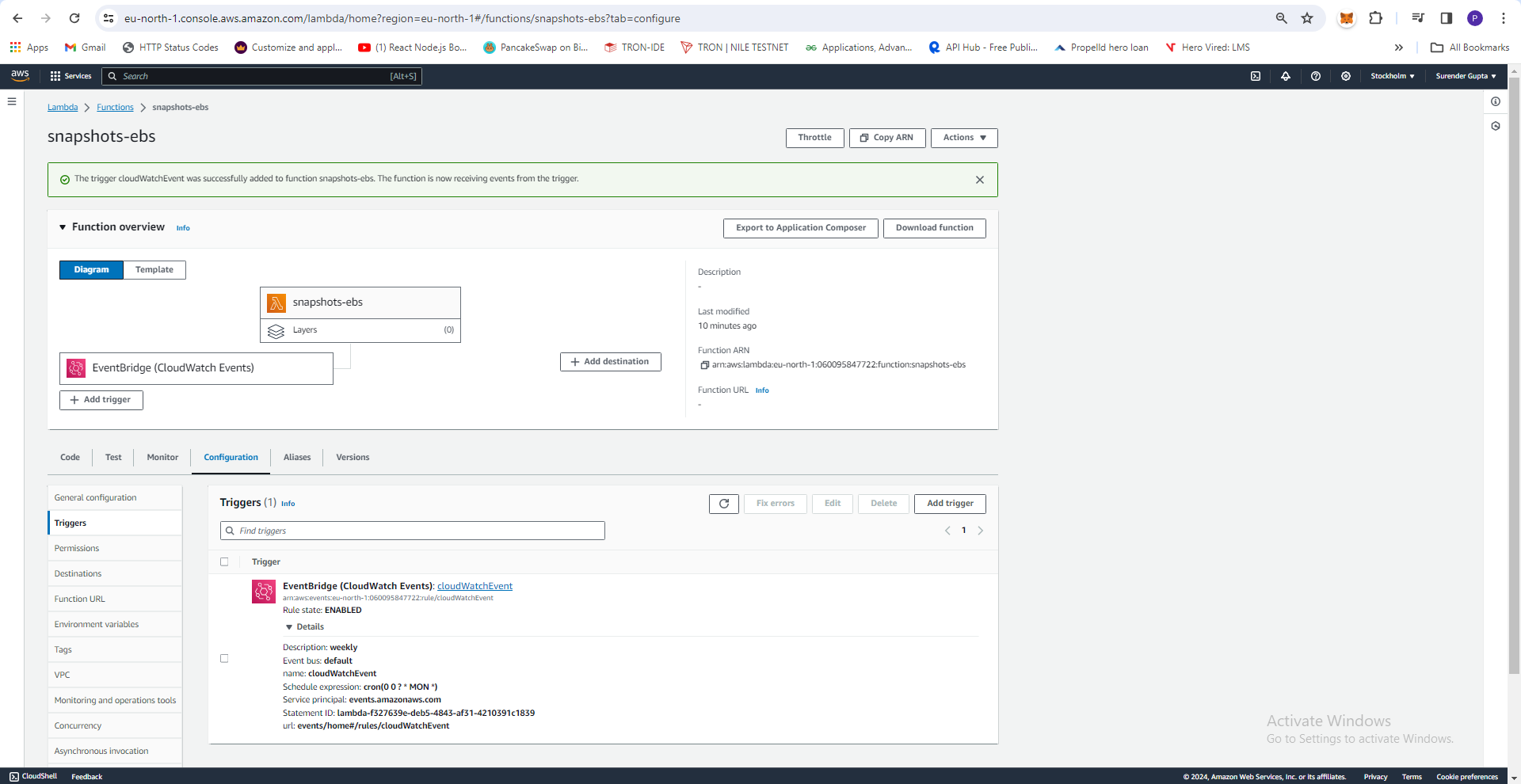
'body': 'EBS successully executed'

}

1. Above code paste in Lambda function and deploy and then Test script manually.



1. **Bonus Task: Click on Lambda function then click under the section of Function Overview Click on Add trigger Search Cloud Watch Event and select it.**
2. **After Select you see two option of Rule Create a new rule or Existing rules we choose Create a new rule**
3. **Choose any name Rule name related to task and Rule description as you want or leave empty.**
4. **Rule type also two option Event Pattern or Schedule expression but we choose Schedule expression**
5. **Now Schedule expression we set cron(0 0 ? \* MON \*)**
6. **Now click on Add button at bottom.**



GitHub Link: <https://github.com/surendergupta/Assignment-On-Serverless-Architecture>