**Graded Assignment On Serverless Architecture**

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

import boto3

ec2 = boto3.client('ec2')

# List of instance IDs to stop and start

auto\_stop\_instance\_ids = ['i-00ad63bbf35948a9c']

auto\_start\_instance\_ids = ['i-0288ee490c4119041']

def lambda\_handler(event, context):

    # Stop 'Auto-Stop' instances

    if auto\_stop\_instance\_ids:

        ec2.stop\_instances(InstanceIds=auto\_stop\_instance\_ids)

        for instance\_id in auto\_stop\_instance\_ids:

            print(f"Stopped instance: {instance\_id}")

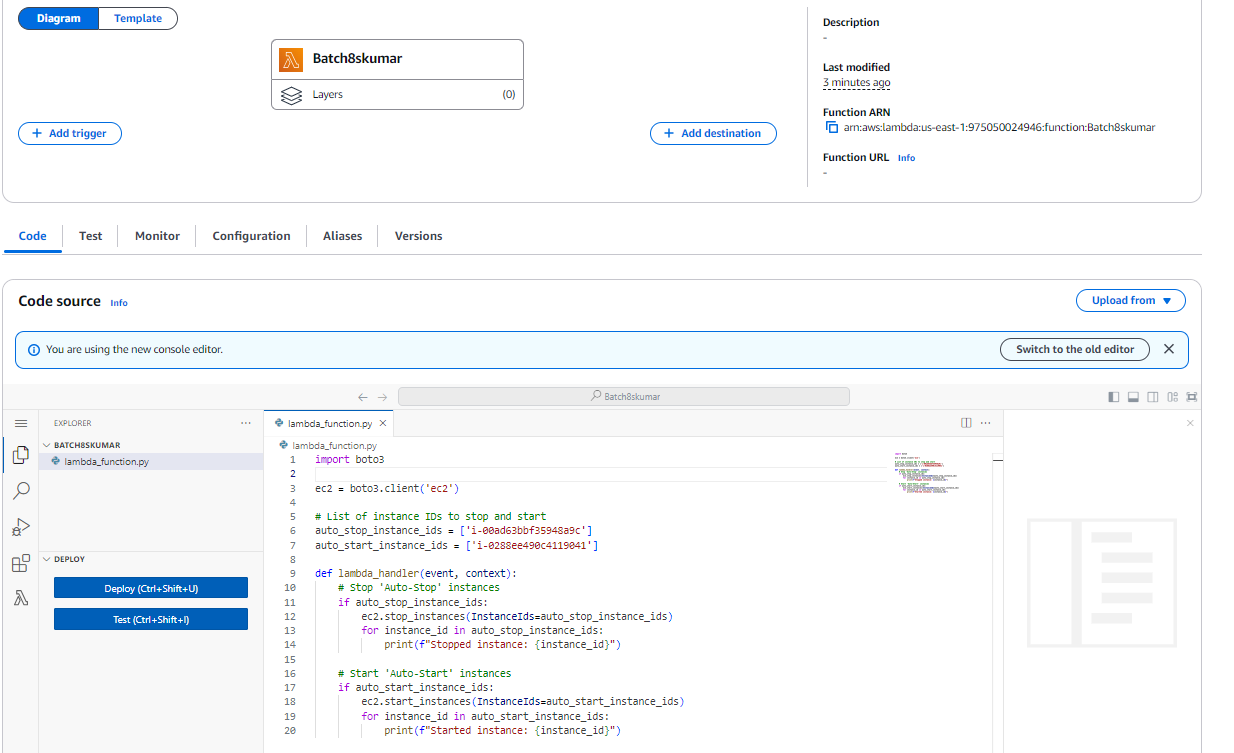
    # Start 'Auto-Start' instances

    if auto\_start\_instance\_ids:

        ec2.start\_instances(InstanceIds=auto\_start\_instance\_ids)

        for instance\_id in auto\_start\_instance\_ids:

            print(f"Started instance: {instance\_id}")





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

import boto3

from datetime import datetime, timezone, timedelta

# Initialize the S3 client outside the handler

s3 = boto3.client('s3')

def lambda\_handler(event, context):

    bucket\_name = 'batch8shiv'

    days\_old = 30

    cutoff\_date = datetime.now(timezone.utc) - timedelta(days=days\_old)

    # List objects in the specified bucket

    response = s3.list\_objects\_v2(Bucket=bucket\_name)

    if 'Contents' in response:

        for obj in response['Contents']:

            obj\_date = obj['LastModified']

            if obj\_date < cutoff\_date:

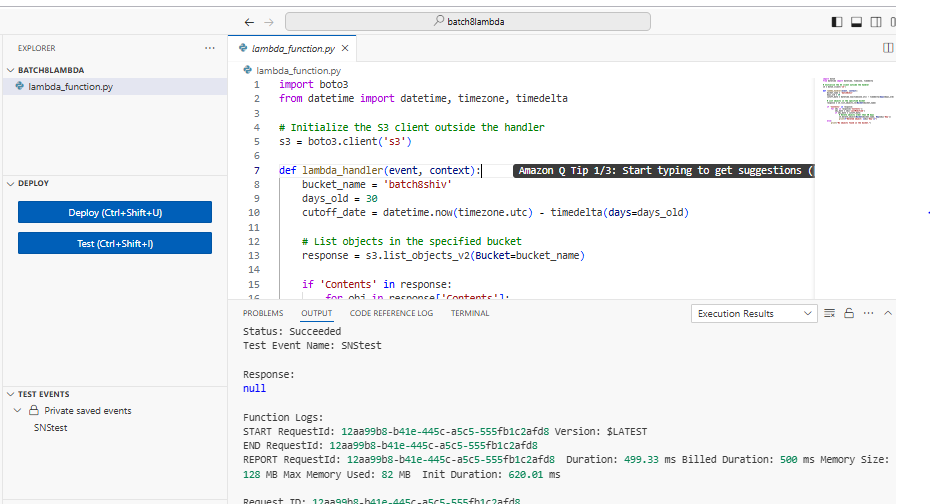
                # Delete objects older than 30 days

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

                print(f"Deleted object: {obj['Key']}")

    else:

        print("No objects found in the bucket.")



**Assignment 11: EC2 Backup and File Cleanup Using Lambda, Boto3, and S3**

import boto3

import os

import zipfile

from datetime import datetime, timedelta

ec2 = boto3.client('ec2')

s3 = boto3.client('s3')

# Replace with your S3 bucket name

S3\_BUCKET = 'batch8shiv'

# Replace with your EC2 instance ID

INSTANCE\_ID = 'i-00ad63bbf35948a9c'

# Directory to backup

DIRECTORY\_TO\_BACKUP = '/home/ubuntu/backup/backupfiles/'

def lambda\_handler(event, context):

    # Connect to EC2 instance and create a backup

    ec2\_instance = ec2.describe\_instances(InstanceIds=[INSTANCE\_ID])

    instance = ec2\_instance['Reservations'][0]['Instances'][0]

    instance\_ip = instance['PublicIpAddress']

    # Create a zip file of the directory

    zip\_filename = f'/tmp/backup\_{datetime.now().strftime("%Y%m%d%H%M%S")}.zip'

    with zipfile.ZipFile(zip\_filename, 'w') as zipf:

        for root, dirs, files in os.walk(DIRECTORY\_TO\_BACKUP):

            for file in files:

                zipf.write(os.path.join(root, file))

    # Upload the zip file to S3

    s3.upload\_file(zip\_filename, S3\_BUCKET, os.path.basename(zip\_filename))

    # Delete backups older than 30 days

    delete\_old\_backups(S3\_BUCKET)

    return {

        'statusCode': 200,

        'body': 'Backup and cleanup completed successfully.'

    }

def delete\_old\_backups(bucket\_name):

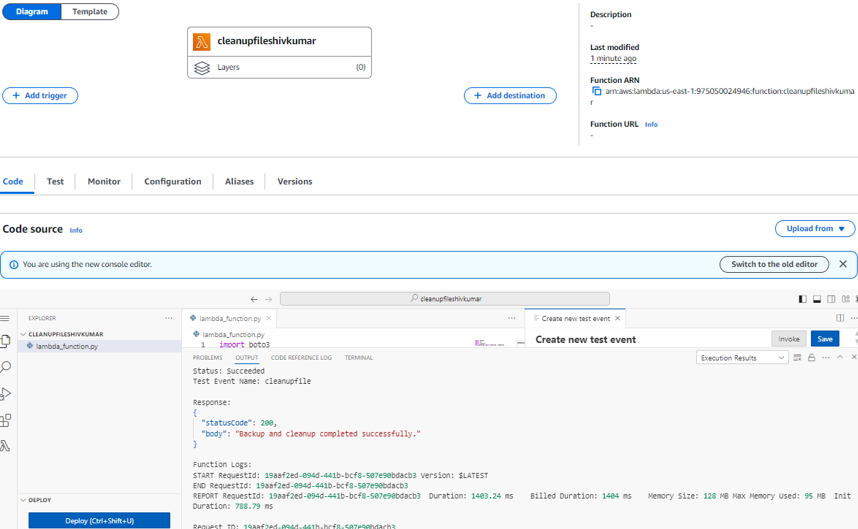
    response = s3.list\_objects\_v2(Bucket=bucket\_name)

    if 'Contents' in response:

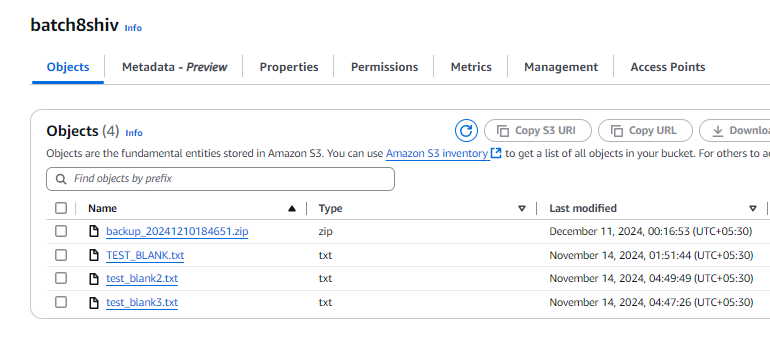
        for obj in response['Contents']:

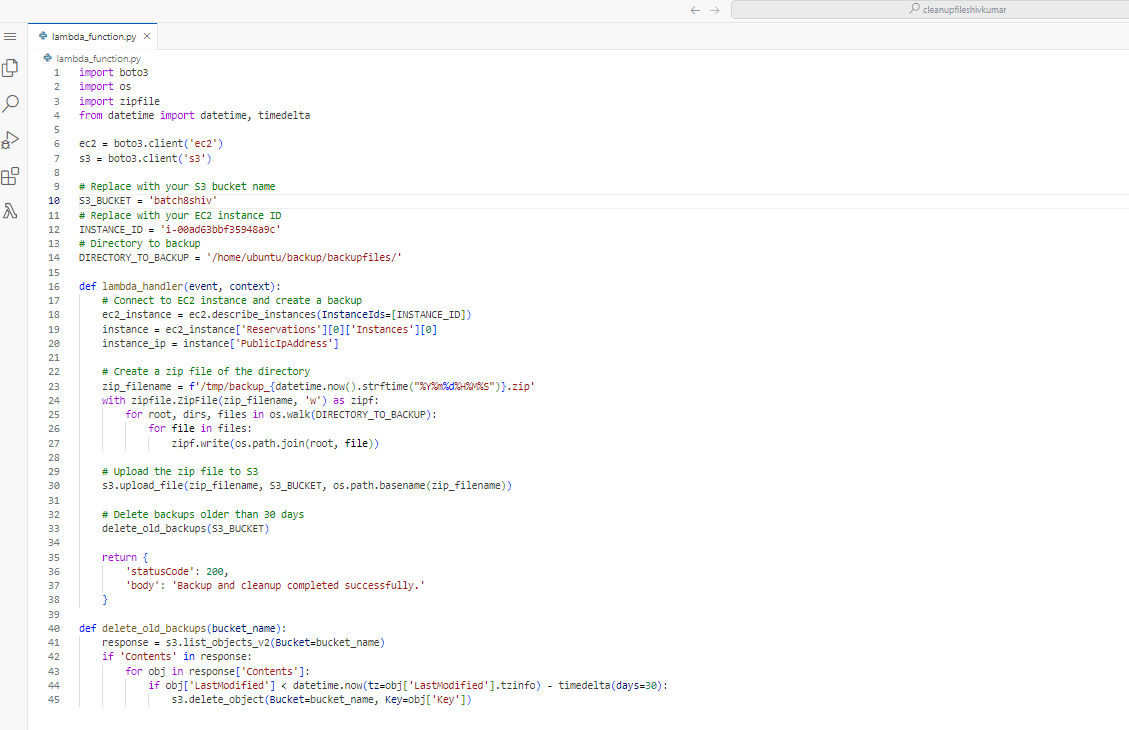
            if obj['LastModified'] < datetime.now(tz=obj['LastModified'].tzinfo) - timedelta(days=30):

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



-------------------------------------------





**Assignment 14: Monitor EC2 Instance State Changes Using AWS Lambda, Boto3, and SNS**

import boto3

import os

sns = boto3.client('sns')

# Retrieve the SNS topic ARN from environment variables

SNS\_TOPIC\_ARN = os.environ['SNS\_TOPIC\_ARN']

def lambda\_handler(event, context):

    # Extract details from the event

    detail = event['detail']

    instance\_id = detail['instance-id']

    state = detail['state']

    # Create the message

    message = f'EC2 Instance {instance\_id} is now {state}.'

    # Send the SNS notification

    sns.publish(

        TopicArn=SNS\_TOPIC\_ARN,

        Subject='EC2 Instance State Change',

        Message=message

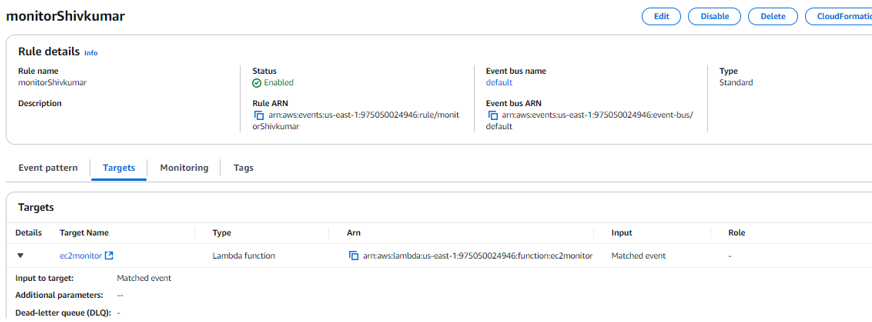
    )

    return {

        'statusCode': 200,

        'body': 'Notification sent successfully.'

    }



-------------------------------------

