Task-18

Objective: Automate the process of creating a new EC2 instance from the latest snapshot using a Lambda function.

Instructions:

1. Create a Lambda function.

2. Using Boto3, the function should:

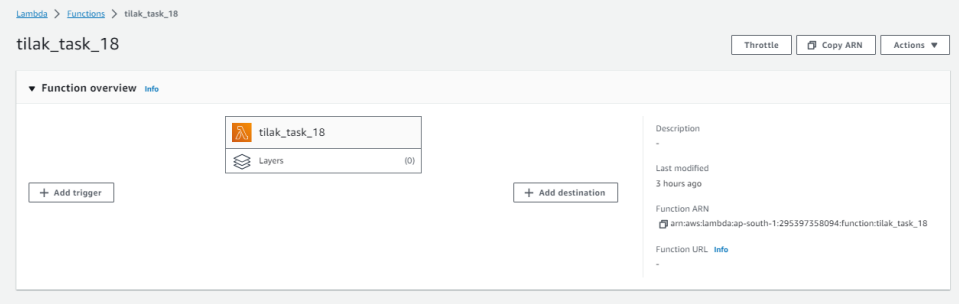
    1. Fetch the most recent snapshot of a given EC2 instance.
    2. Create a new EC2 instance using the fetched snapshot.

3. Trigger this Lambda function manually or on a schedule, depending on your recovery requirements.

Content Index:

Step1:

Create a lambda function as shown in the below screenshot



Please find the code below for reference

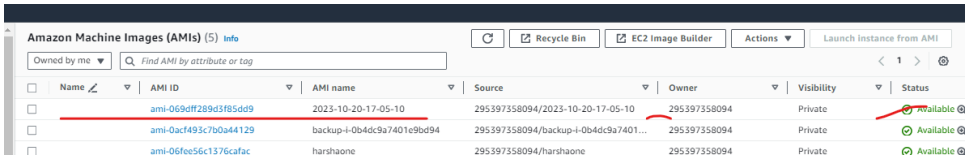
```
import boto3

import time

import datetime


def lambda_handler(event, context):
```

```
# Define your EC2 client
ec2_client = boto3.client('ec2')

# Specify your snapshot and instance properties
#snapshot_id = 'your-snapshot-id'  # Replace with your snapshot ID
instance_type = 't3.micro'
key_name = 'tilak-keypair'
security_group_ids = ['sg-03a0a98fe163a64bb']
subnet_id = 'subnet-0ea185273ead71a27'
instance_name = 'MyEC2Instance'

# Get the latest snapshot for the root volume in your region and account
snapshots = ec2_client.describe_snapshots(
    Filters=[
        {'Name': 'volume-id', 'Values': ['vol-0968b088a5ed88dd2']},
        {'Name': 'status', 'Values': ['completed']}
    ],
    OwnerIds=['295397358094'],
)['Snapshots']

# Sort the snapshots by their start time in descending order to get the latest
snapshots.sort(key=lambda x: x['StartTime'], reverse=True)

if not snapshots:
    raise Exception("No valid snapshots found for the root volume.")

latest_snapshot_id = snapshots[0]['SnapshotId']

# Create a custom AMI from the desired snapshot
AMI_Name = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
response = ec2_client.create_image(
```

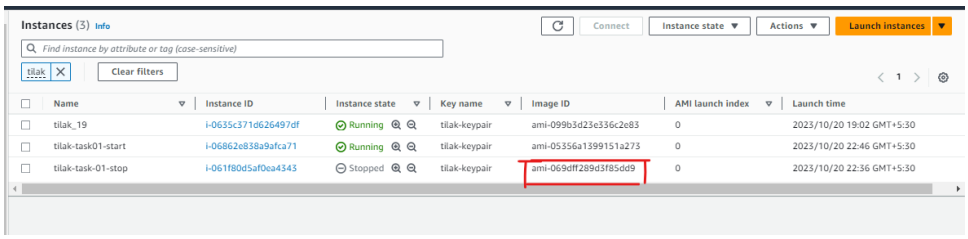
```
    InstanceId='i-0635c371d626497df',  # Replace with your instance ID
    Name=AMI_Name,
    Description='Custom AMI based on a specific snapshot',
    NoReboot=True,  # You can specify whether to reboot the instance or not
    BlockDeviceMappings=[
        {
            'DeviceName': '/dev/sda1',
            'Ebs': {
                'SnapshotId': latest_snapshot_id,
                'VolumeSize': 8,  # Specify the size of the root volume
                'DeleteOnTermination': True
            }
        }
    ]
)

custom_ami_id = response['ImageId']
# Wait for the custom AMI to be in the "available" state
while True:
    ami = ec2_client.describe_images(ImageIds=[custom_ami_id])['Images'][0]
    if ami['State'] == 'available':
        break
    time.sleep(5)  # Sleep for 5 seconds before checking again

# Launch an EC2 instance using the custom AMI
response = ec2_client.run_instances(
    ImageId=custom_ami_id,
    InstanceType=instance_type,
    KeyName=key_name,
    MaxCount=1,
    MinCount=1,
```

```
    SecurityGroupIds=security_group_ids,
    SubnetId=subnet_id,
    TagSpecifications=[
        {
            'ResourceType': 'instance',
            'Tags': [
                {'Key': 'Name', 'Value': instance_name},
            ]
        }
    ]
)

instance_id = response['Instances'][0]['InstanceId']

return {
    'statusCode': 200,
    'body': f'EC2 instance {instance_id} created using custom AMI {custom_ami_id}.'
}
```

Run the code to create a new AMI



Also, verify the Instance as well if it is launched using the AMI Image.



Completed.