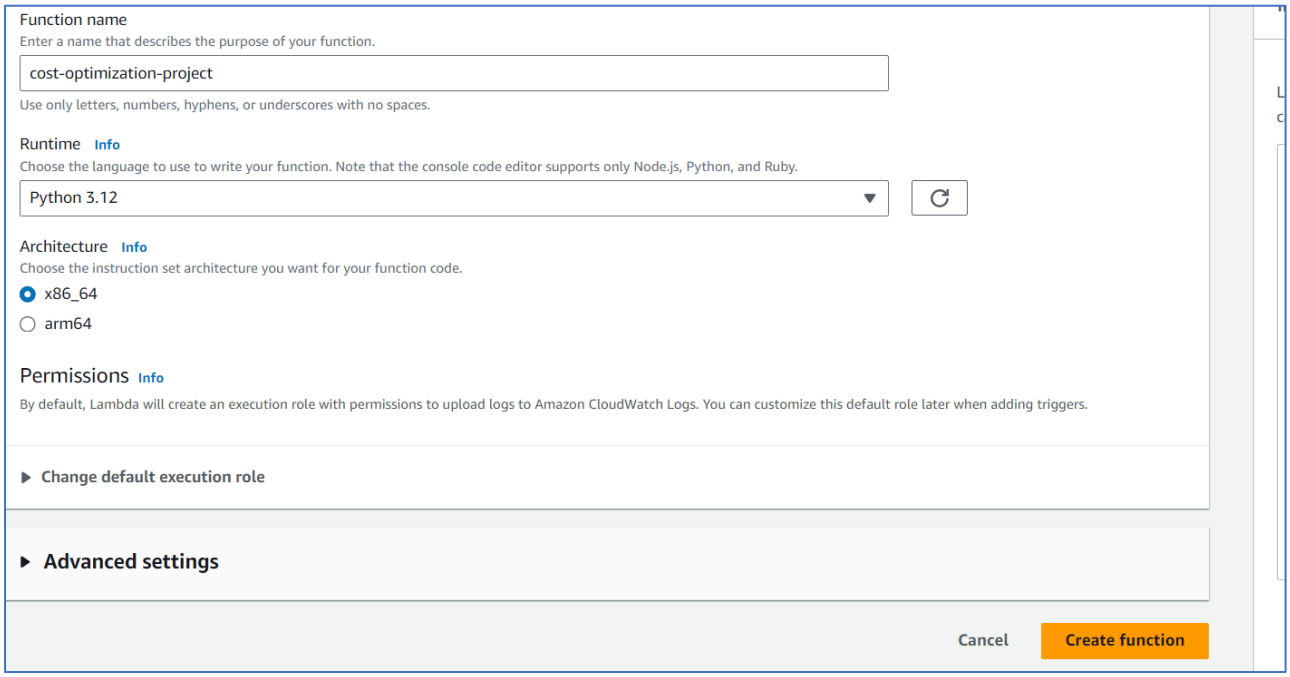
Purpose:

Our project aims to develop a Lambda function tailored to optimize storage costs within the AWS environment. By leveraging this function, we will systematically identify EBS snapshots that have become disassociated from active EC2 instances and subsequently execute their deletion. This initiative is driven by the goal of reducing unnecessary storage overheads and fostering a more efficient resource management strategy within the AWS Cloud infrastructure. Through the implementation of this Lambda function, we anticipate significant cost savings while maintaining operational agility and scalability.

Python script for the lambda function (base script, can be modified as per usecase):

```
import boto3

def lambda_handler(event, context):

    ec2 = boto3.client('ec2')


    # Get all EBS snapshots

    response = ec2.describe_snapshots(OwnerIds=['self'])


    # Get all active EC2 instance IDs

    instances_response = ec2.describe_instances(Filters=[{'Name': 'instance-state-name', 'Values':
['running']}])

    active_instance_ids = set()


    for reservation in instances_response['Reservations']:

        for instance in reservation['Instances']:

            active_instance_ids.add(instance['InstanceId'])


    # Iterate through each snapshot and delete if it's not attached to any volume or the volume is not
attached to a running instance

    for snapshot in response['Snapshots']:

        snapshot_id = snapshot['SnapshotId']

        volume_id = snapshot.get('VolumeId')


        if not volume_id:
```

```
        # Delete the snapshot if it's not attached to any volume
        ec2.delete_snapshot(SnapshotId=snapshot_id)
        print(f"Deleted EBS snapshot {snapshot_id} as it was not attached to any volume.")
    else:
        # Check if the volume still exists
        try:
            volume_response = ec2.describe_volumes(VolumeIds=[volume_id])
            if not volume_response['Volumes'][0]['Attachments']:
                ec2.delete_snapshot(SnapshotId=snapshot_id)
                print(f"Deleted EBS snapshot {snapshot_id} as it was taken from a volume not attached to
any running instance.")
        except ec2.exceptions.ClientError as e:
            if e.response['Error']['Code'] == 'InvalidVolume.NotFound':
                # The volume associated with the snapshot is not found (it might have been deleted)
                ec2.delete_snapshot(SnapshotId=snapshot_id)
                print(f"Deleted EBS snapshot {snapshot_id} as its associated volume was not found.")
```



lambda_function × Environment Var ×

```python
import boto3

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')

    # Get all EBS snapshots
    response = ec2.describe_snapshots(OwnerIds=['self'])

    # Get all active EC2 instance IDs
    instances_response = ec2.describe_instances(Filters=[{'Name': 'instance-state-name', 'Values': ['running']}])
    active_instance_ids = set()

    for reservation in instances_response['Reservations']:
        for instance in reservation['Instances']:
            active_instance_ids.add(instance['InstanceId'])

    # Iterate through each snapshot and delete if it's not attached to any volume or the volume is not attached to a
    for snapshot in response['Snapshots']:
        snapshot_id = snapshot['SnapshotId']
        volume_id = snapshot.get('VolumeId')

        if not volume_id:
            # Delete the snapshot if it's not attached to any volume
            ec2.delete_snapshot(SnapshotId=snapshot_id)
            print(f"Deleted EBS snapshot {snapshot_id} as it was not attached to any volume.")
        else:
            # Check if the volume still exists
            try:
                volume_response = ec2.describe_volumes(VolumeIds=[volume_id])
                if not volume_response['Volumes'][0]['Attachments']:
                    ec2.delete_snapshot(SnapshotId=snapshot_id)
                    print(f"Deleted EBS snapshot {snapshot_id} as it was taken from a volume not attached to any runn
            except ec2.exceptions.ClientError as e:
                if e.response['Error']['Code'] == 'InvalidVolume.NotFound':
                    # The volume associated with the snapshot is not found (it might have been deleted)
                    ec2.delete_snapshot(SnapshotId=snapshot_id)
                    print(f"Deleted EBS snapshot {snapshot_id} as its associated volume was not found.")
```

12:1 Python Spaces: 4

Learn how
cases in A

Creat
app

In this t
to:

- E
- I

Learn m

Star

Explaination of the code:

Importing boto3: boto3 is the AWS SDK for Python, which allows Python developers to write software that makes use of services like Amazon S3 and Amazon EC2.

Defining the lambda_handler function: This is the entry point for the Lambda function. It takes two parameters: event and context. In this code, only the context parameter is being used.

Creating an EC2 client: The code initializes an EC2 client using boto3.client('ec2'). This client will be used to interact with EC2 resources in AWS.

Getting all EBS snapshots: The code calls ec2.describe_snapshots(OwnerIds=['self']) to retrieve information about all EBS snapshots that are owned by the current AWS account ('self').

Getting all active EC2 instance IDs: The code calls ec2.describe_instances() with a filter to get information about all running EC2 instances. It then extracts the instance IDs and stores them in a set called active_instance_ids.

Iterating through each snapshot: The code loops through each snapshot obtained in step 4. For each snapshot, it checks if it's attached to any volume.

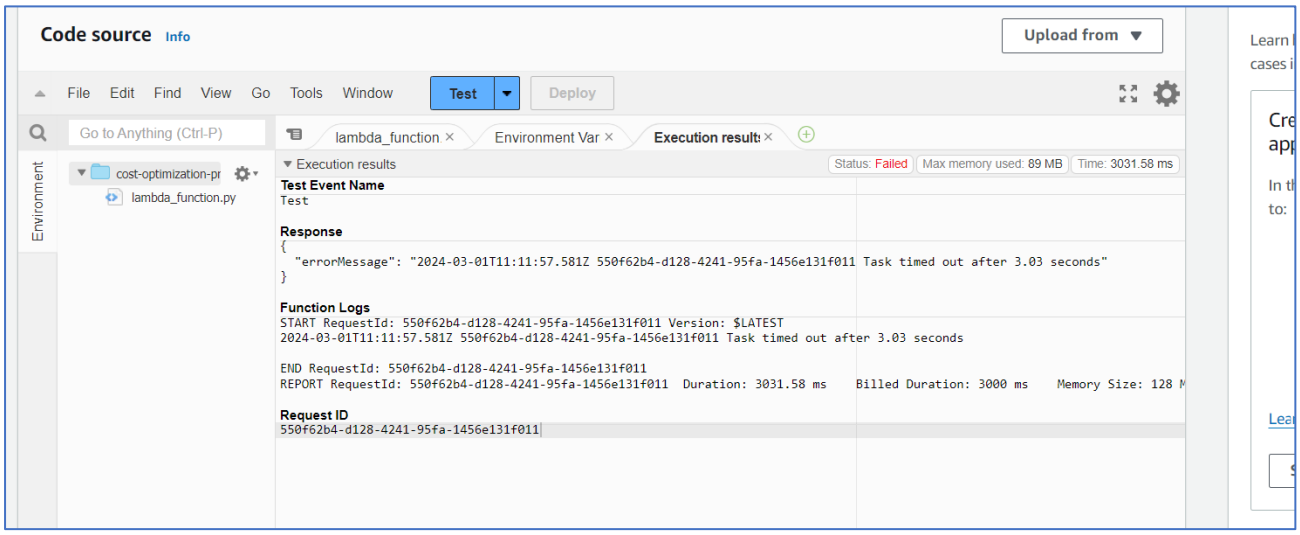
a. If the snapshot is not attached to any volume (volume_id is None), it deletes the snapshot using ec2.delete_snapshot(SnapshotId=snapshot_id) and prints a message indicating that the snapshot was deleted because it was not attached to any volume.

b. If the snapshot is attached to a volume, it checks if the volume still exists by calling ec2.describe_volumes(VolumeIds=[volume_id]). If the volume exists but is not attached to any running instance, it deletes the snapshot and prints a message indicating that the snapshot was deleted because it was taken from a volume not attached to any running instance.

c. If the volume associated with the snapshot is not found (possibly deleted), it catches the InvalidVolume.NotFound exception, deletes the snapshot, and prints a message indicating that the snapshot was deleted because its associated volume was not found.

Challenges faced during the project :

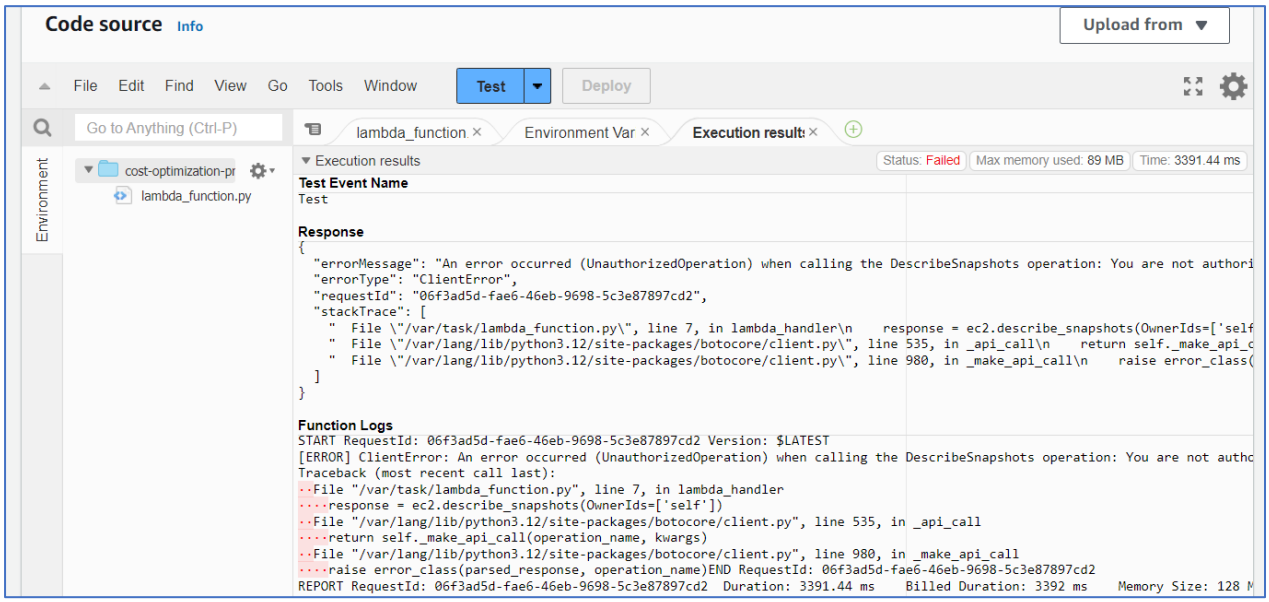
1) Lambda function execution time out error :



Solution :

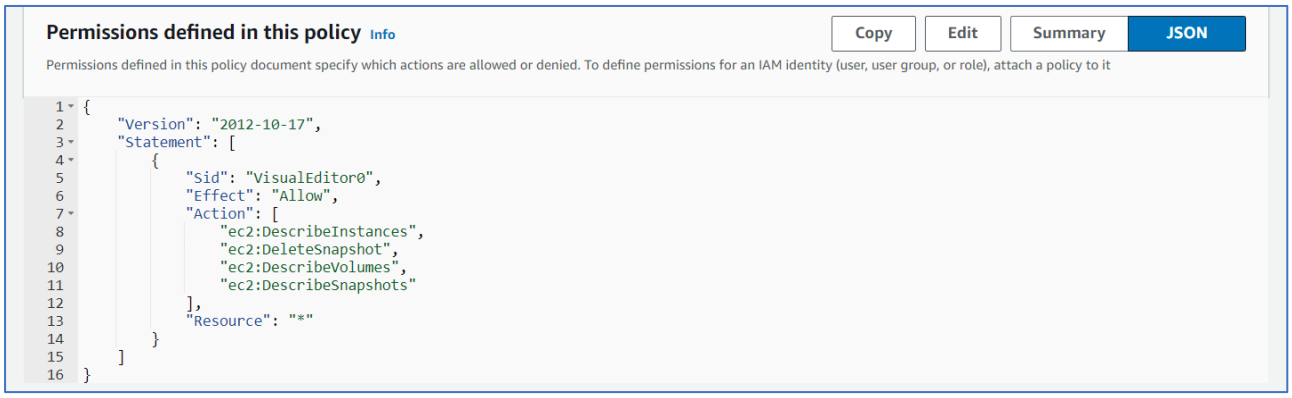
Increased the lambda execution time from 3 seconds to 10 seconds (for demo purposes, can vary according to use case)

2) Lambda UNAUTHORIZED EXCEPTION :

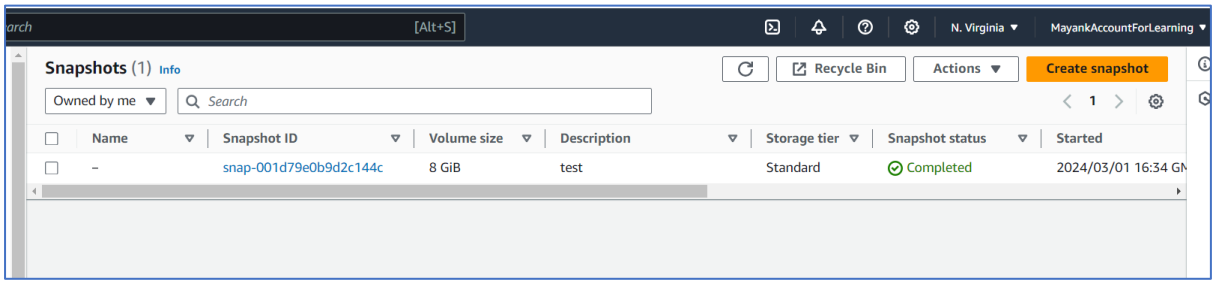


Solution:

To assign the necessary permissions to the role associated with the lambda function, Here, I included DescribeSnapshots, DescribeInstances,DeleteSnapshots, DescribeVolumes to the policy attached to the role.



Final Result:



Above snapshot got deleted as it was not associated with any running EC2 instance