AWS Cost Optimization | Most Popular Cloud and DevOps project | Event Driven Serverless

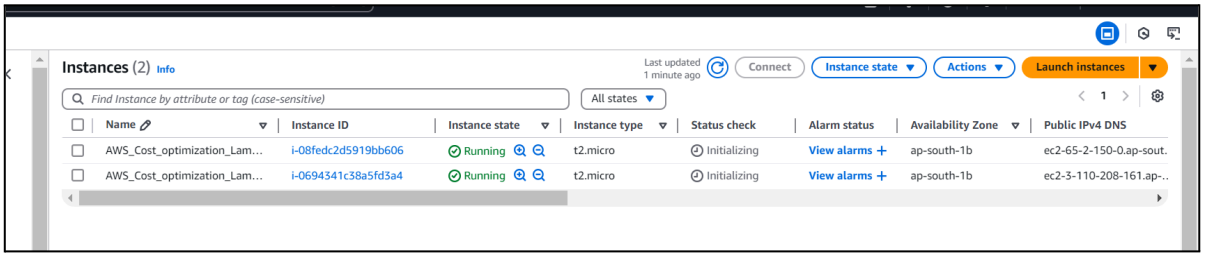
---

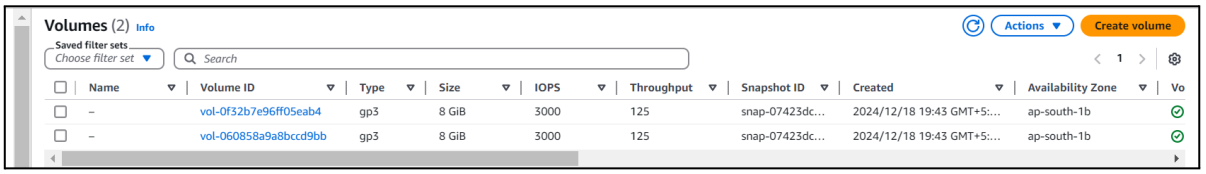
Identifying Stale EBS Snapshots :
In this example, we'll create a Lambda function that identifies EBS snapshots that are no longer associated with any active EC2 instance and deletes them to save on storage costs.

Description:
The Lambda function fetches all EBS snapshots owned by the same account ('self') and also retrieves a list of active EC2 instances (running and stopped). For each snapshot, it checks if the associated volume (if exists) is not associated with any active instance. If it finds a stale snapshot, it deletes it, effectively optimizing storage costs.

1. Launch the EC2 Instance

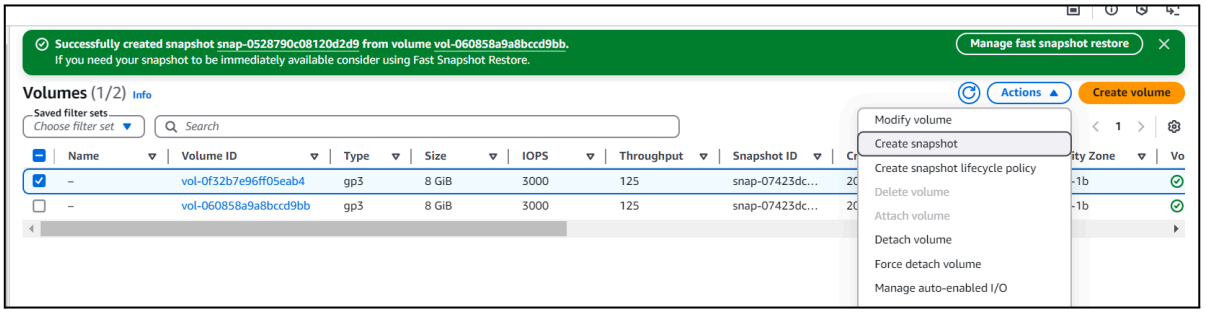
- Successfully launched two EC2 instances.
- Verified that the volumes for both instances have been created.
  (Screenshots of the created volumes)

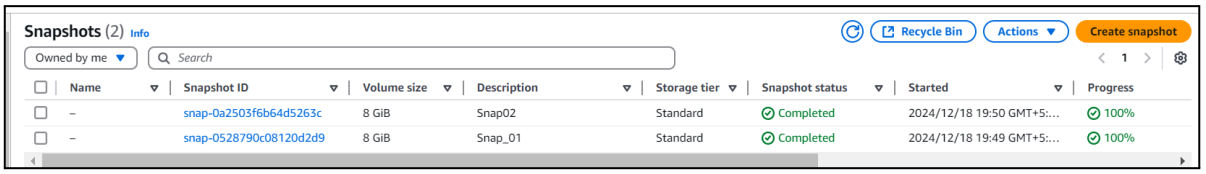




2. Take Snapshots of the Volumes

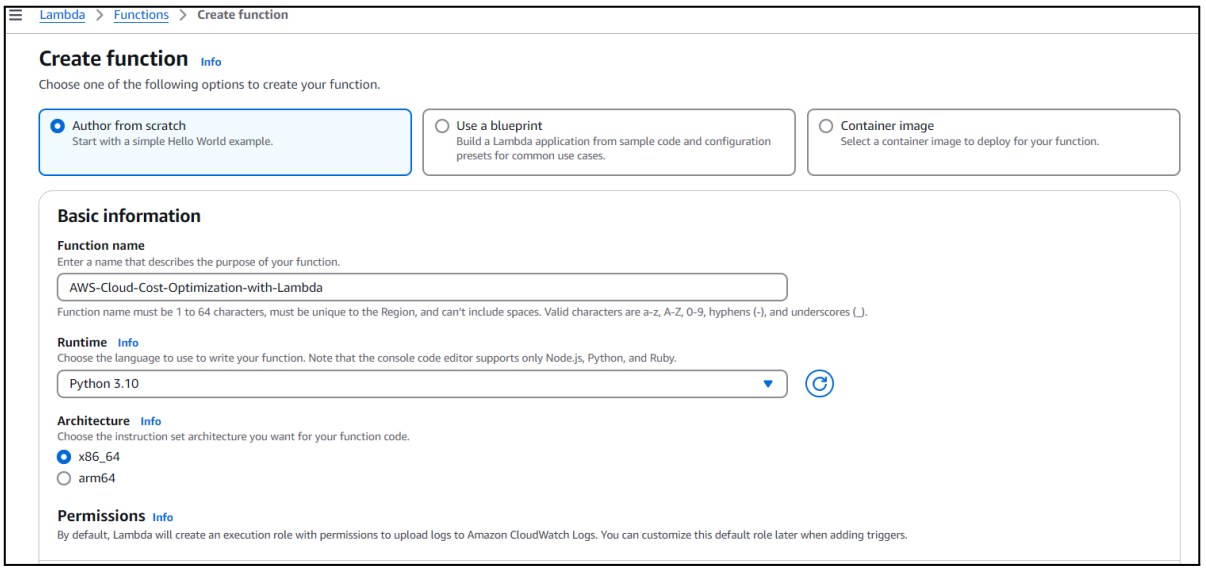
- Navigate to the EC2 Dashboard.
- For each volume, go to **Actions** → **Create Snapshot**.
  (Snapshots of both volumes taken and displayed in screenshots)





3. Create the Lambda Function

- Open the **AWS Management Console** and search for **Lambda**.
- Start creating a new function. Initially, no permissions are modified; adjustments will be made after testing.



4. Add the Code from GitHub

- Clone or copy the code from the GitHub repository:
  [GitHub Repo: AWS Cloud Cost Optimization with Lambda](#).

```
==================================================================
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.")
```
==========

Paste the code into the Lambda code editor, deploy, and test with the event name **Testing Phase.**

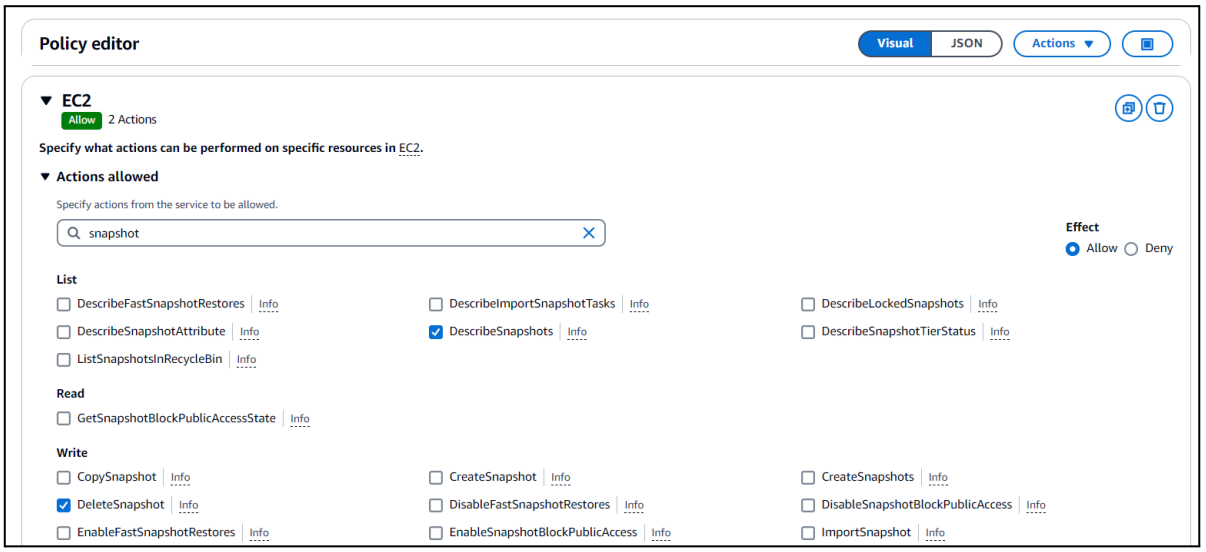
5. Handle Initial Errors

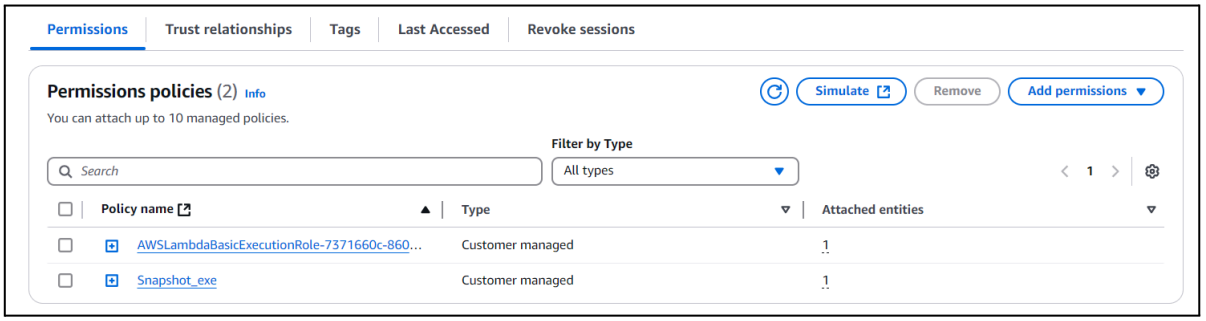
- **Error 1**: Execution timeout.
  Fix: Go to **Configuration** → **General Settings** and increase the timeout from 3 seconds to 10 seconds.
- **Error 2**: Insufficient permissions.
  Fix: Adjust the Lambda execution role by creating a new policy.

```
Response:
{
  "errorMessage": "2024-12-18T14:50:10.104Z 0748f397-67c0-42a5-b0a7-ac09df2882ee Task timed out after 3.01 seconds"
}

Function Logs:
START RequestId: 0748f397-67c0-42a5-b0a7-ac09df2882ee Version: $LATEST
2024-12-18T14:50:10.104Z 0748f397-67c0-42a5-b0a7-ac09df2882ee Task timed out after 3.01 seconds

END RequestId: 0748f397-67c0-42a5-b0a7-ac09df2882ee
REPORT RequestId: 0748f397-67c0-42a5-b0a7-ac09df2882ee  Duration: 3009.98 ms    Billed Duration: 3000 ms    Memory Size: 128 MB Max Memory Used: 89 MB  Init Duration: 279.24 ms
```

6. Create a Custom IAM Policy

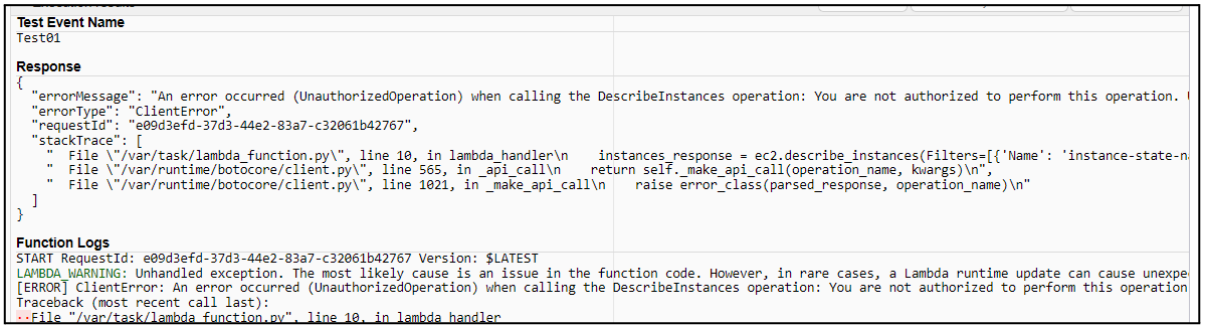
- Navigate to **IAM** → **Policies** and create a new policy.
- Grant permissions for:
  - **EC2**: `DescribeInstances`, `DescribeSnapshots`, `DeleteSnapshot`.
  - **EBS**: Relevant permissions for snapshots.
- Save and name the policy.
- Attach this new policy to the Lambda function's execution role.



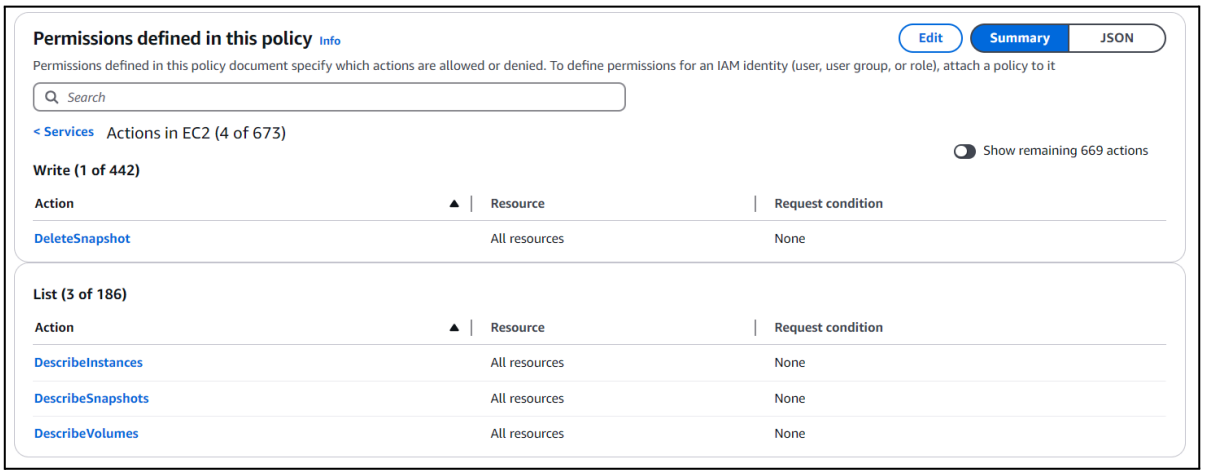


7. Re-Test the Lambda Function

- Run the Lambda function again.
- **Error 3**: Missing permissions for `Describe` and `Delete` on EC2.
  Fix: Update the policy to include these permissions.

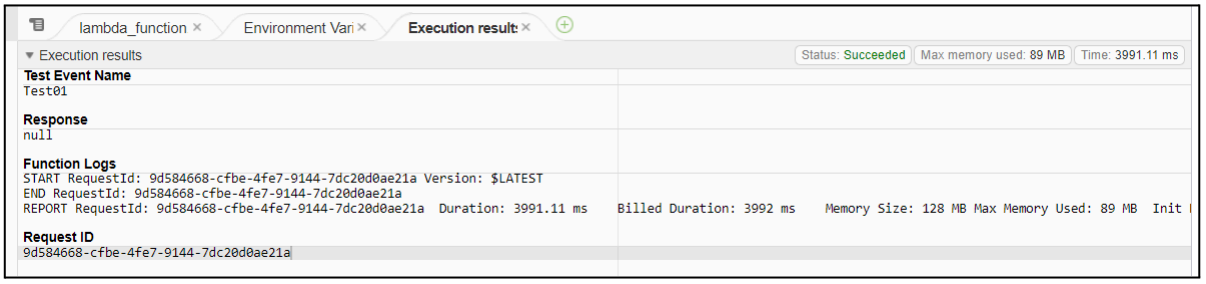


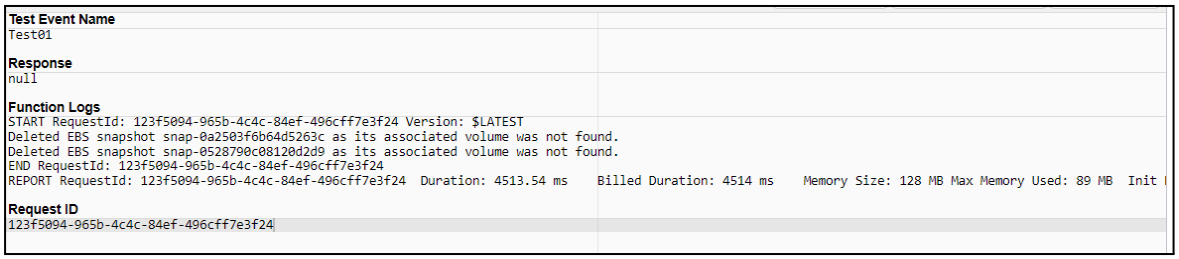
As we don't have the description and delete permission for the EC2 instance, we have edited the above permission and saved



8. Final Testing

- After adjusting permissions, test the Lambda function again.
- **Result**: Snapshots from running instances are not deleted (as expected).
- To verify functionality, terminate the instances and delete associated volumes.
- Re-run the Lambda function.
- **Result**: Snapshots were successfully deleted as no active instances were associated.





Summary

This process automates snapshot management by ensuring only necessary snapshots are retained. The Lambda function, combined with proper IAM roles and policies, effectively optimizes EBS storage costs by deleting unused snapshots.

Source of this Tutorial:

Day-18 | AWS Cost Optimization | Most Popular Cloud and DevOps project| Event Driv…