

Cloud Cost Optimization — AWS

Used the combination of AWS cloud watch and Lambda functions To decrease the cloud cost by 25 percent.

Implementation:

Created lambda function in python, used the boto3 module to interact With AWS service APIs and this lambda function is triggered by Cloud Watch events. This lambda function would watch for any unused EBS snapshots and Either delete them or send out notification to the snapshot owner.

AWS Cloud Cost Optimization - Identifying Stale Resources

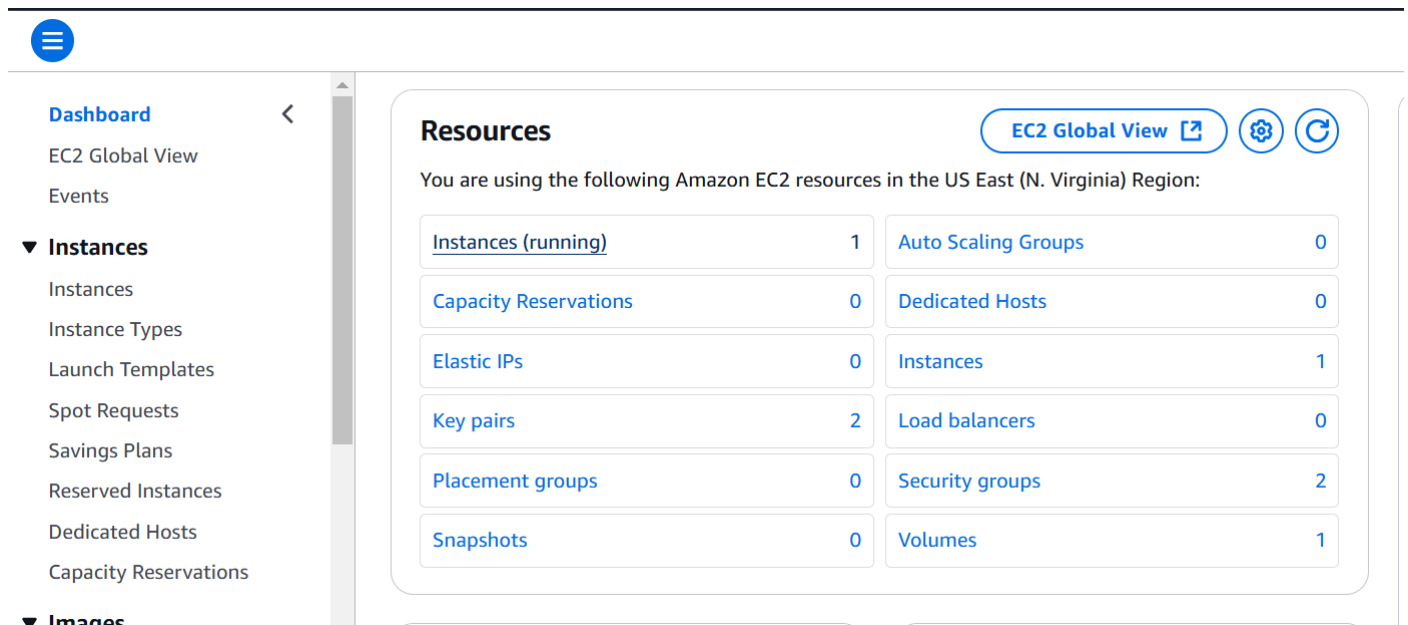
Identifying Stale EBS Snapshots

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. Go to Aws console, check the resources (instances, volumes, snapshots...etc) in EC2 instances



The screenshot shows the AWS Management Console interface. On the left is a navigation sidebar with a hamburger menu icon at the top. Below it, the 'Dashboard' is selected, followed by 'EC2 Global View' and 'Events'. Under the 'Instances' section, various options are listed: Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations. Below that is the 'Images' section. The main content area is titled 'Resources' and shows a summary of EC2 resources in the 'US East (N. Virginia) Region'. It includes a link to 'EC2 Global View' and icons for settings and refresh. Below this, a table lists the following resources and their counts:

Resource Type	Count
Instances (running)	1
Capacity Reservations	0
Elastic IPs	0
Key pairs	2
Placement groups	0
Snapshots	0
Auto Scaling Groups	0
Dedicated Hosts	0
Instances	1
Load balancers	0
Security groups	2
Volumes	1

{Because I don't have any Stale Resources in my AWS account for the purpose of this project demonstration, so I have created a Stale Resources i.e, from 2 - }

2. Created a EC2 instance by default it will add a volume and launch instance.

The screenshot shows the AWS Management Console's 'Instances' page. The instance 'test-ec2' (ID: i-05924f8654774b334) is in a 'Running' state. The left sidebar lists navigation options: Dashboard, EC2 Global View, Events, and Instances. The top bar includes the AWS logo, search bar, and user information. The instance details table shows columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability zone.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
test-ec2	i-05924f8654774b334	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1

3. Click on instance id go to storage you find volume id of the created instance.

The screenshot shows the 'Storage' tab for the instance 'test-ec2'. The 'Root device details' section shows the root device name as '/dev/sda1' and the root device type as 'EBS'. The 'Block devices' section shows a table with columns for Volume ID, Device name, Volume size (GiB), Attachment status, and Attachment time. The volume 'vol-0632cc6d38272e680' is attached to the device '/dev/sda1'.

Volume ID	Device name	Volume size (GiB)	Attachment status	Attachment time
vol-0632cc6d38272e680	/dev/sda1	8	Attached	2024/12/26 18:07 GMT+5:30

4. Go back to the EC2 dashboard and click on snapshots and create snapshot.

The screenshot shows the AWS Management Console's 'Snapshots' page. The snapshot 'snap-0442688e7a3157041' is shown with a volume size of 8 GiB and a description of 'test'. The left sidebar lists navigation options: Dashboard, EC2 Global View, Events, and Instances. The top bar includes the AWS logo, search bar, and user information. The snapshot details table shows columns for Name, Snapshot ID, Volume size, Description, and Storage tier.

Name	Snapshot ID	Volume size	Description	Storage tier
-	snap-0442688e7a3157041	8 GiB	test	Standard

5. Go to lambda function and create a lambda function.

The screenshot shows the 'Create function' page in the AWS Lambda console. The 'Author from scratch' option is selected. The 'Basic information' section shows the function name 'cost-optimization-efs-snapshot' and the runtime 'Python 3.13'. The 'Architecture' section is partially visible. On the right, there is a 'Tutorials' sidebar with a link to 'Create a simple web app'.

Create function Info

Choose one of the following options to create your function.

- ☒ **Author from scratch**
Start with a simple Hello World example.
- ☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.
- ☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture Info

Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#)

6. You can write your own lambda function code if you know, else go to this [gitHub](#).

7. Remove the existing code and past the code, save this, give a Test name and click on Deploy.

The screenshot shows the AWS Lambda console with the 'cost-optimization-efs-snapshot' function selected. A green notification bar at the top states 'Successfully updated the function cost-optimization-efs-snapshot.' Below it, a yellow warning bar indicates that Lambda will stop supporting this version of the console code editor on January 31, 2025. The code editor shows the following Python code:

```
1 import boto3
2
3 def lambda_handler(event, context):
4     ec2 = boto3.client('ec2')
5
6     # Get all EBS snapshots
7     response = ec2.describe_snapshots(OwnerIds=['self'])
8
9     # Get all active EC2 instance IDs
10    instances_response = ec2.describe_instances(Filters=[{'Name': 'instance-state-name', 'Values': ['running']}])
11    active_instance_ids = set()
12
13    for reservation in instances_response['Reservations']:
14        for instance in reservation['Instances']:
15            active_instance_ids.add(instance['InstanceId'])
16
17    # Iterate through each snapshot and delete if it's not attached to any volume or the volume is not attached to a running instance
18    for snapshot in response['Snapshots']:
19        snapshot_id = snapshot['SnapshotId']
20        volume_id = snapshot.get('VolumeId')
21
22        if not volume_id:
```

8. Now configure the timeout to 10sec

The screenshot shows the AWS Lambda console for the function 'cost-optimization-ebs-snapshot'. A green notification bar at the top states 'Successfully updated the function cost-optimization-ebs-snapshot.' The 'Configuration' tab is selected, showing the 'General configuration' section. The 'Timeout' is set to '0 min 10 sec'. Other settings include 'Memory' at 128 MB, 'Ephemeral storage' at 512 MB, and 'SnapStart' set to 'None'. The left sidebar shows navigation options like 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'.

9. Go to permissions and click on Role name and add the permissions if you want you can create policies or add existing policies. Here I am creating my policies suitable for my project(snapshot permissions, EC2, Volume) and select resources all and name to the policy, create it.

10. Now attached the created policies and add the permission.

The screenshot shows the AWS IAM console for the role 'cost-optimization-ebs-snapshot-role-5rfh4dv5'. The 'Permissions' tab is selected, showing 'Permissions policies (2)'. A table lists the attached policies:

Policy name	Type	Attached entities
AWSLambdaBasicExecutionRole-...	Customer managed	1
cost-optimization-ebs	Customer managed	1

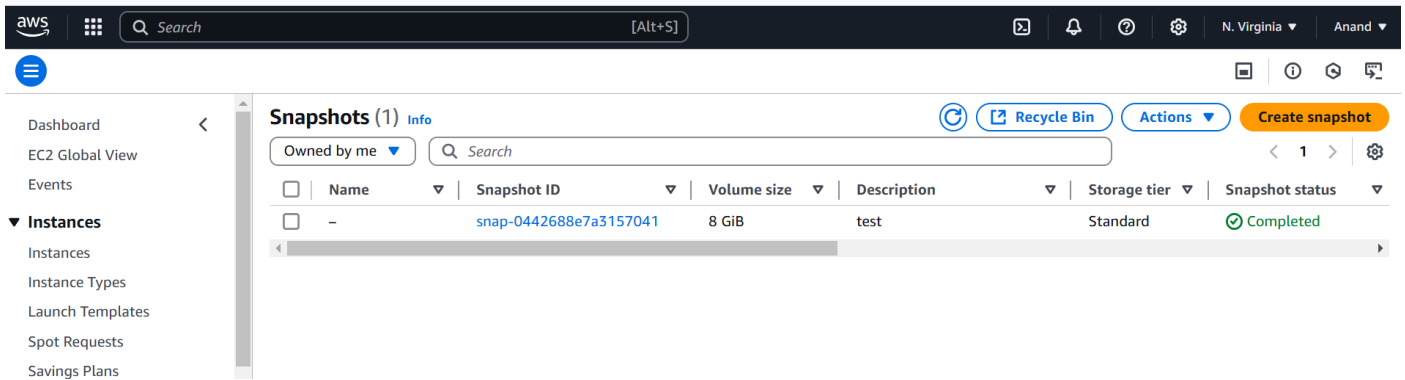
Below the table, it says 'Permissions boundary (not set)'. The left sidebar shows navigation options like 'Dashboard', 'Access management', 'Roles', 'Policies', 'Identity providers', 'Account settings', 'Root access management', 'Access reports', and 'Access Analyzer'.

11. Click on the Test you will see like this below if you don't have stale resources.

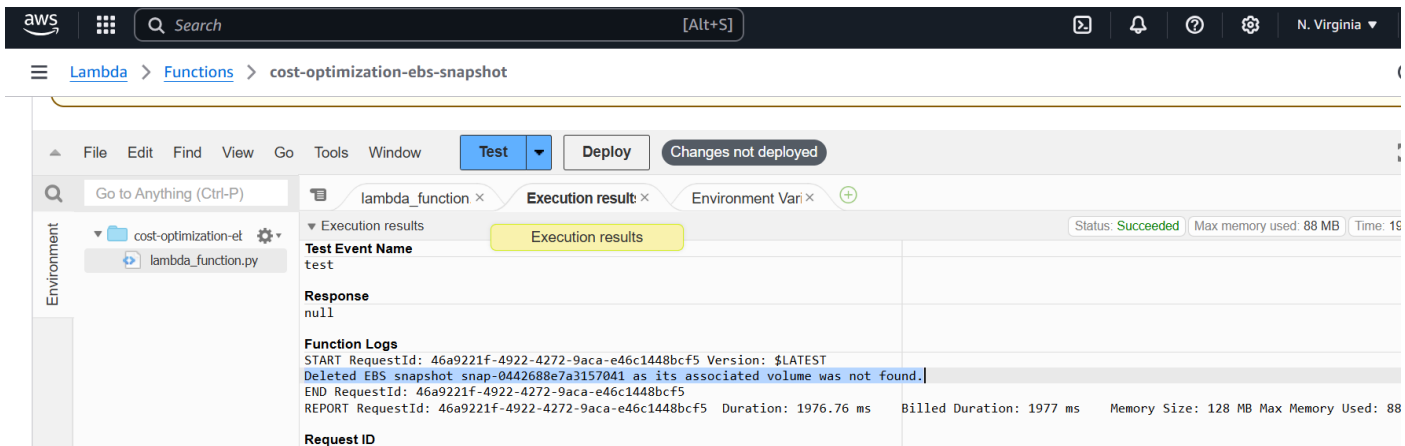
The screenshot shows the AWS Lambda console interface. The top navigation bar includes the AWS logo, a search bar, and the user's name 'Anand' in the 'N. Virginia' region. The breadcrumb trail indicates the path: Lambda > Functions > cost-optimization-ebs-snapshot. The main content area displays the 'Test' tab for the 'lambda_function' resource. The 'Execution results' section shows a 'Status: Succeeded' with 'Max memory used: 88 MB' and 'Time: 4337.69 ms'. The 'Test Event Name' is 'test', and the 'Response' is 'null'. The 'Function Logs' section shows the execution details, including the 'Request ID' and 'Duration'. The 'Request ID' is '24f610ef-85f2-46dc-8fdf-cdaff4f1aeff'.

12. For project purpose I will delete the instance, but the snapshot is still present. Now I have stale resource.

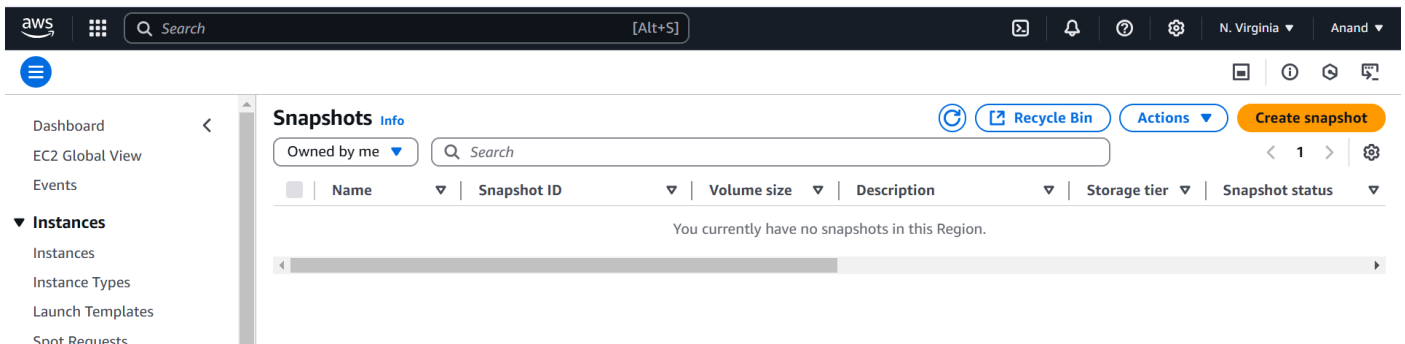
The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo, a search bar, and the user's name 'Anand' in the 'N. Virginia' region. The breadcrumb trail indicates the path: Dashboard > EC2 Global View > Instances. The main content area displays the 'Instances' page. A green banner at the top indicates 'Successfully initiated termination (deletion) of i-05924f8654774b334'. The 'Instances (1/1)' section shows a table with one instance, 'test-ec2', with ID 'i-05924f8654774b334'. The instance is in the 'Shutting-down' state. The 'Instance summary' section shows the instance details, including the 'Instance ID', 'Public IPv4 address', and 'Private IPv4 addresses'. The 'Resources' section shows a table of resources used by the instance, including 'Instances (running)', 'Capacity Reservations', 'Elastic IPs', 'Key pairs', 'Placement groups', 'Snapshots', 'Auto Scaling Groups', 'Dedicated Hosts', 'Instances', 'Load balancers', 'Security groups', and 'Volumes'. The 'EC2 Free Tier' section shows the usage of free tier resources, including the 'End of month forecast' and 'Exceeds free tier' warning.



13. Now again Test the code, the stale resource is deleted.



14. The stale snapshot is deleted here and project is completed.



NOTE: For Hand on experience I used my AWS free tier account to do this while learning, if you also using Free tier account, make sure delete all the things you performed.