

Project Document

AWS Cloud Cost Optimization – Identifying Snapshots and Deleting Stale EBS Snapshots

[LinkedIn: - Deepak Patra](#)

Overview

This project aims to reduce unnecessary AWS storage costs by automatically identifying and deleting stale EBS snapshots — snapshots that are:

- Not associated with any volume
- Associated with a volume that no longer exists
- Associated with a volume not attached to any active EC2 instance

This is achieved by running an AWS Lambda function on a scheduled basis (via Event Bridge) to clean up unused snapshots.

Project Objective

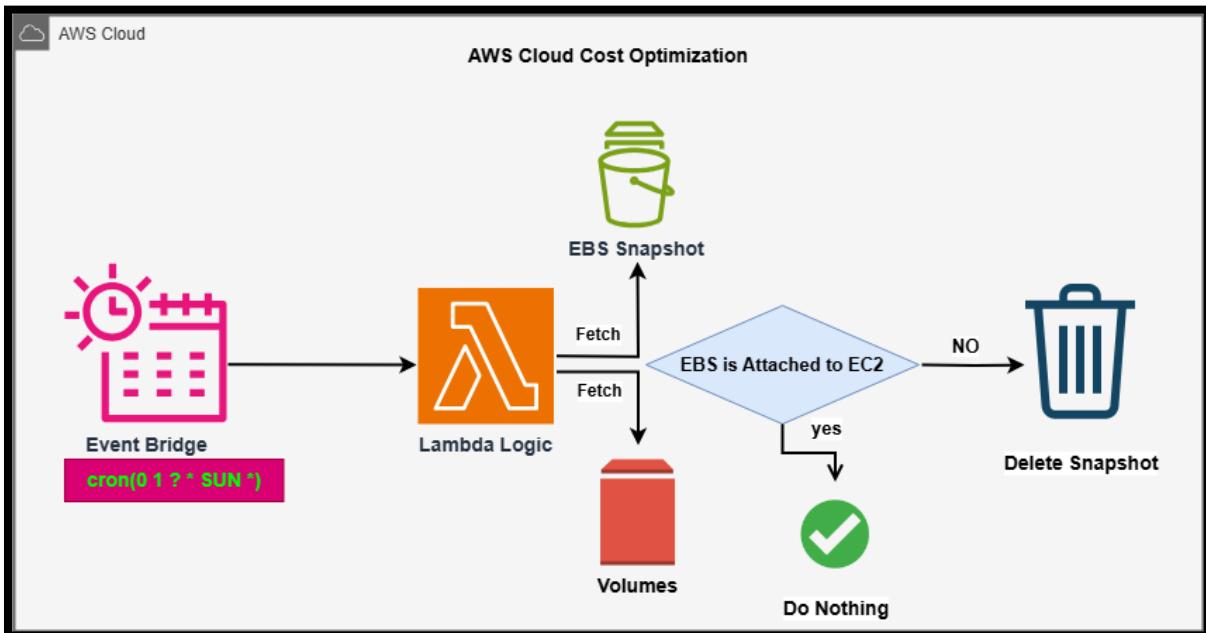
The objective of this project is to:

To optimize EBS storage costs by identifying and deleting stale EBS snapshots using serverless automation.

Architecture Overview

The architecture includes:

1. **Amazon Event Bridge:** Triggers the Lambda function on a schedule.
2. **AWS Lambda:** Executes the logic to find and delete stale snapshots.
3. **Amazon EC2:** Source of snapshots and associated volumes.
4. **Amazon CloudWatch Logs:** Stores logs for Lambda executions.



Architecture

Components Used

Service	Purpose
AWS Lambda	Executes snapshot cleanup logic
Amazon EC2	Source of EBS volumes and snapshots
Amazon EventBridge	Schedules the Lambda to run periodically
IAM Role	Provides necessary permissions to Lambda
CloudWatch Logs	Logs execution details for monitoring and debugging

Lambda Function Logic

The Lambda function:

- Fetches all EBS snapshots owned by the account.
- Fetches all active EC2 instances (running or stopped).
- For each snapshot:
 - Deletes it if it is not linked to any volume.
 - Deletes it if the linked volume no longer exists.
 - Deletes it if the volume exists but is not attached to any EC2 instance.
 - Skips snapshots with the tag `Keep=true`.

Screen Shot:

The screenshot shows the AWS Lambda console interface. In the top navigation bar, there are several tabs: AWS Lambda, EC2, IAM, VPC, CloudWatch, S3, IAM Identity Center, AWS Organizations, WAF & Shield, Simple Notification Service, and Route 53. The 'Lambda' tab is selected. Below the navigation bar, the 'Functions' section is shown, with 'demo-func' selected. The main area displays the 'Function overview' with a diagram showing the function name 'demo-func' and its layers. It also shows triggers like 'EventBridge (CloudWatch Events)' and options to 'Add destination' or 'Add trigger'. On the right side, there are sections for 'Description', 'Last modified' (19 minutes ago), 'Function ARN' (arn:aws:lambda:ap-south-1:767398014530:function:demo-func), and 'Function URL' (Info). At the bottom, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration' (which is currently selected), 'Aliases', and 'Versions'. The status bar at the bottom right indicates '7:51 PM 5/15/2025'.

```

# This sample demonstrates the usage of the AWS Lambda Python runtime.
# The function will be triggered by an event from CloudWatch Events.
# The event will contain a list of EBS snapshot IDs.
# The function will then fetch all EBS snapshots owned by the Lambda function
# and log the details.

logger = logging.getLogger()
logger.setLevel(logging.INFO)

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

    # Fetch all EBS snapshots owned by self
    logger.info("Fetching all EBS snapshots...")
    response = ec2.describe_snapshots(OwnerIds=['self'])
    snapshots = response.get('Snapshots', [])
    logger.info(f"Found {len(snapshots)} snapshots.")

    # Fetch all running and stopped EC2 instance IDs
    logger.info("Fetching all active EC2 instances (running + stopped)...")

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

    active_instance_ids = set()
    for reservation in instances_response['Reservations']:
        for instance in reservation['Instances']:
            active_instance_ids.add(instance['InstanceId'])
    logger.info(f"Active EC2 instances: {active_instance_ids}")

```

The screenshot shows the AWS Lambda code editor for the 'lambda_function.py' file. The code uses the AWS SDK for Python (Boto3) to interact with the AWS Lambda and EC2 services. It logs information about EBS snapshots owned by the function and lists active EC2 instances. The browser status bar indicates '7:52 PM 5/15/2025'.

IAM Role Permissions

Attach the following permissions to your Lambda's IAM role:

- **EC2:DescribeSnapshots**
- **EC2:DescribeVolumes**
- **EC2:DescribeInstances**
- **EC2>DeleteSnapshot**

Screen Shot:

The screenshot shows the AWS IAM Policy Editor interface. A navigation bar at the top includes links for AWS Lambda, EC2, IAM, VPC, CloudWatch, S3, IAM Identity Center, AWS Organizations, WAF & Shield, Simple Notification Service, and Route 53. The main area is titled "Modify permissions in demo-policy" and contains a "Policy editor" section. The policy JSON is as follows:

```

1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Sid": "VisualEditor0",
6             "Effect": "Allow",
7             "Action": [
8                 "ec2:DescribeInstances",
9                 "ec2>DeleteSnapshot",
10                "ec2:DescribeVolumes",
11                "ec2:DescribeSnapshots"
12            ],
13            "Resource": "*"
14        }
15    ]
16 }

```

On the right side of the editor, there are tabs for "Visual", "JSON", and "Actions". Below the editor, there are sections for "Edit statement", "Add actions", "Choose a service" (with a dropdown menu for "Filter services"), and "Included" and "Available" services like EC2, AI Operations, AMP, and API Gateway.

Event Bridge (CloudWatch) Trigger

To run the function every Sunday at 1:00 AM UTC, use this cron expression:

Screenshot:

The screenshot shows the AWS EventBridge Scheduler console. The left sidebar includes links for AWS Lambda, EC2, IAM, VPC, CloudWatch, S3, IAM Identity Center, AWS Organizations, WAF & Shield, Simple Notification Service, and Route 53. The main area displays a schedule named "Your schedule delete-snapshot is being created." The "Schedule" tab is active, showing the cron expression:

```

0 1 ? * SUN *

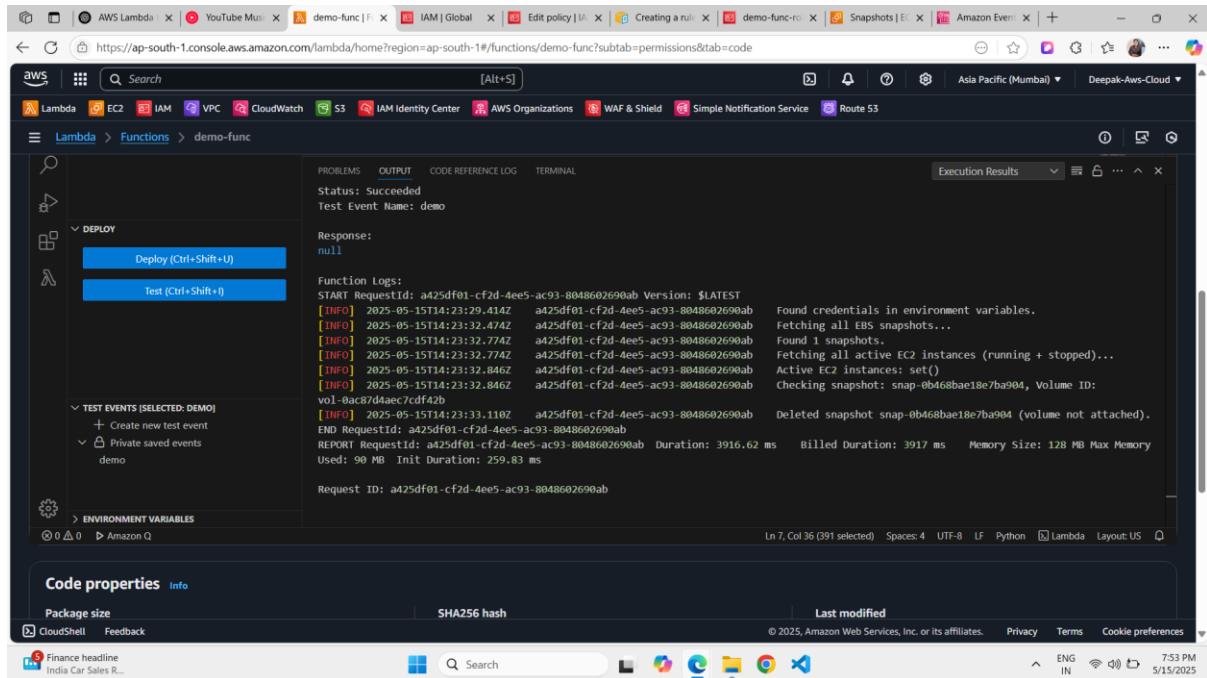
```

The "Cron expression" field has a "Copy cron expression" button. To the right, there is a "Cron expression" section with a detailed explanation of what a cron expression is and how it works. At the bottom, there are "Was this content helpful?" buttons for "Yes" and "No".

Testing the Lambda

1. In Lambda console, click Test.
2. Create a test event (name: `demo`) with any sample JSON (or `{}`).
3. Run the function.
4. Check CloudWatch Logs for output.
5. Confirm if unused snapshots were deleted.

Screenshot:



Benefits

- **Saves cost by deleting unneeded EBS snapshots.**
- **Runs automatically — no manual intervention.**
- **Easily customizable (tags, frequency, filters).**
- **Serverless and low-maintenance.**

💰 Cost Estimation (as of 2025 AWS Pricing)

AWS Service	Usage Estimate	Monthly Cost (USD)
AWS Lambda	~1 execution/week (~4/month) × <1s/runtime	Free (under free tier)
Event Bridge	1 rule, 4 invocations/month	Free (First 100K invocations are free)
CloudWatch Logs	~2–5 MB/month (log storage)	~\$0.01–\$0.03
EBS Snapshot Deletion	N/A (deletion is free)	Free
IAM Role	For permissions	Free

◆ **Total Monthly Cost:**

\$0.01 to \$0.03 USD/month