**AWS EBS Snapshot Cleanup for Cost Optimization**

In the dynamic world of cloud infrastructure, managing resources efficiently is paramount. Our project aims to enhance cost optimization by automating the cleanup of stale **Amazon Elastic Block Store** (EBS) snapshots in AWS environments.

**Problem Statement:**

Over time, AWS environments accumulate EBS snapshots that are no longer associated with active Amazon Elastic Compute Cloud (EC2) instances or volumes. These stale snapshots consume valuable storage resources and contribute to unnecessary costs. Identifying and removing these snapshots manually can be a time-consuming and error-prone task.

**Project Goals:**

Our project has two primary objectives:

**Identify Stale EBS Snapshots**: Implement an automated mechanism to identify EBS snapshots that are no longer associated with any active EC2 instances or volumes. This process will involve analyzing metadata and status information.

**Automated Cleanup**: Develop a robust cleanup mechanism that securely deletes identified stale EBS snapshots. We will implement this as a scheduled process to ensure continuous cost optimization without manual intervention.

**Key Features:**

**Snapshot Analysis**: Implement algorithms to identify snapshots that no longer have corresponding volumes or instances.

**Secure Deletion**: Implement a deletion process with safety checks to prevent accidental deletion of important snapshots.

**Logging and Reporting**: Maintain detailed logs of deleted snapshots and generate regular reports for transparency and auditing purposes.

**Benefits**:

**Cost Savings:** By removing unnecessary EBS snapshots, we will significantly reduce storage costs, contributing to cost optimization.

**Efficiency**: Automation eliminates the need for manual snapshot management, saving time and reducing the risk of human errors.

**Resource Optimization**: This project aligns with AWS best practices for resource optimization and cost control.

**Implementation**

**Tech Stack to be utilized –** AWS, Lambda Functions, CloudWatch, Python, EBS Snapshots, Volumes, IAM Roles, Boto3

**Flowchart:**

**We** will create a lambda function with runtime environment of python. In the lambda function we will implement the logic which will first get the details of all the **Snapshots**, **Volumes**, **Active Instances** from AWS. Later it will iterate through all the snapshots from the list and delete them in the following order:

* Delete all the snapshots with no volume id attached with them. (Volume id not found for snapshots)
* Delete all the snapshots with volume Ids attached, volumes with those volume ids available but those volumes not being attached to any active EC2 instances
* Delete all the snapshots with Volume Ids attached but there is no volume available with those volume ids.

The lambda function can be scheduled to run in a fixed interval of time by creating a rule in AWS **CloudWatch**.

**Note** – We need to create and use a **role** having permissions to describe **snapshots**, **volumes** and **Ec2** instances and delete snapshot permission as well then role need to be attached to the lambda function.

**Lambda Function Python Code:**

***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') # corresponding volume id for the snapshot id***

***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 with that particular volume id then will check if there is any ec2 attached with it***

***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:***

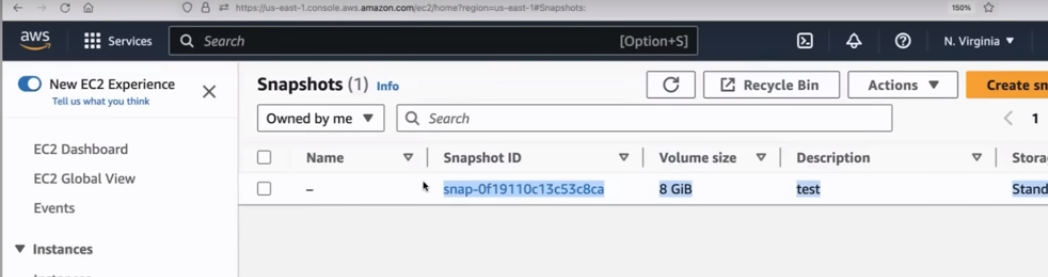
***if e.response['Error']['Code'] == 'InvalidVolume.NotFound':***

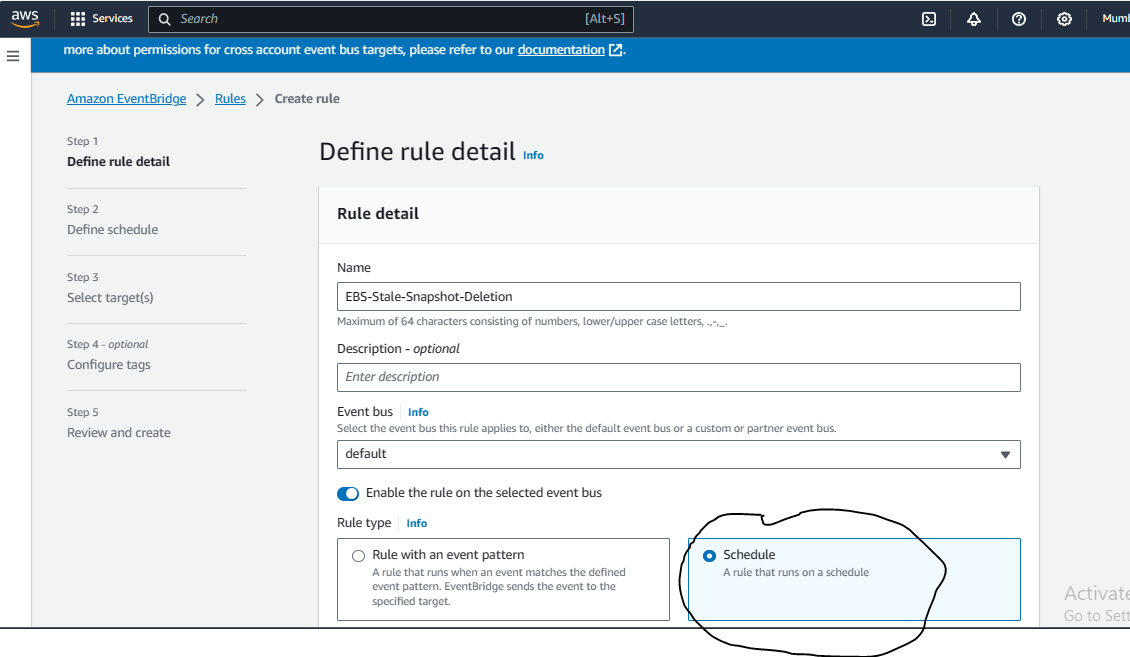
***# The volume associated with the snapshot is not found (it might have been deleted)***

***ec2.delete\_snapshot(SnapshotId=snapshot\_id)***

***print(f"Deleted EBS snapshot {snapshot\_id} as its associated volume was not found.")***

**Reference Snippets:**

****

**Snapshot in AWS **

**Rule creation in CloudWatch for schedule trigger of lambda function**

**Thank You!!**