https://github.com/suvaatnbu/serverlessaaws

Assignment 4: Automatic EBS Snapshot and Cleanup Using AWS Lambda and Boto3

Objective: To automate the backup process for your EBS volumes and ensure that backups older than a specified retention period are cleaned up to save costs.

Task: Automate the creation of snapshots for specified EBS volumes and clean up snapshots older than 30 days.

Instructions:

1. EBS Setup:

   - Navigate to the EC2 dashboard and identify or create an EBS volume you wish to back up.

   - Note down the volume ID.

2. Lambda IAM Role:

   - In the IAM dashboard, create a new role for Lambda.

   - Attach policies that allow Lambda to create EBS snapshots and delete them (`AmazonEC2FullAccess` for simplicity, but be more restrictive in real-world scenarios).

3. Lambda Function:

   - Navigate to the Lambda dashboard and create a new function.

   - Choose Python 3.x as the runtime.

   - Assign the IAM role created in the previous step.

   - Write the Boto3 Python script to:

     1. Initialize a boto3 EC2 client.

     2. Create a snapshot for the specified EBS volume.

     3. List snapshots and delete those older than 30 days.

     4. Print the IDs of the created and deleted snapshots for logging purposes.

4. Event Source (Bonus):

   - Attach an event source, like Amazon CloudWatch Events, to trigger the Lambda function at your desired backup frequency (e.g., every week).

5. Manual Invocation:

   - After saving your function, either manually trigger it or wait for the scheduled event.

   - Go to the EC2 dashboard and confirm that the snapshot is created and old snapshots are deleted.

ANSWER:

Volume ID-vol-063f076d89d6db8a0

Use case: AWS Lambda

Attach policy:  
 AmazonEC2FullAccess

ROLL NAME: LambdaEbsSnapshotRole1

FUNCTION NAME: LambdaEbsSnapshotRole1

import boto3

from datetime import datetime, timezone, timedelta

def lambda\_handler(event, context):

    ec2 = boto3.client('ec2')

    # Volume to back up

    volume\_id = 'vol-063f076d89d6db8a0'  # Replace with your volume ID

    retention\_days = 30

    # Create snapshot

    timestamp = datetime.now().strftime('%Y-%m-%d-%H-%M')

    description = f"Automated backup {timestamp} for {volume\_id}"

    snapshot = ec2.create\_snapshot(

        VolumeId=volume\_id,

        Description=description,

        TagSpecifications=[

            {

                'ResourceType': 'snapshot',

                'Tags': [

                    {'Key': 'CreatedBy', 'Value': 'LambdaBackup'},

                    {'Key': 'VolumeId', 'Value': volume\_id}

                ]

            }

        ]

    )

    print(f"✅ Created snapshot: {snapshot['SnapshotId']}")

    # Cleanup old snapshots

    delete\_before = datetime.now(timezone.utc) - timedelta(days=retention\_days)

    snapshots = ec2.describe\_snapshots(

        Filters=[

            {'Name': 'volume-id', 'Values': [volume\_id]},

            {'Name': 'tag:CreatedBy', 'Values': ['LambdaBackup']}

        ],

        OwnerIds=['self']

    )['Snapshots']

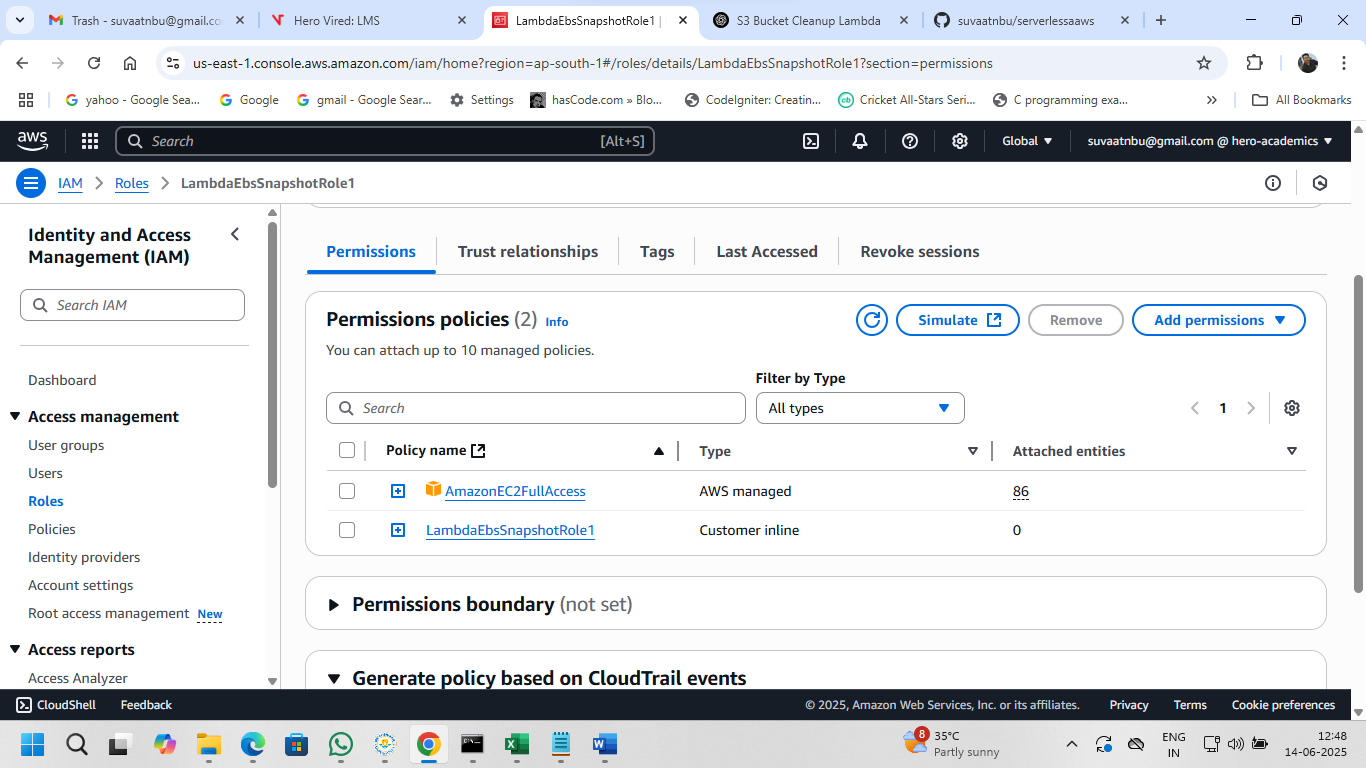
    for snap in snapshots:

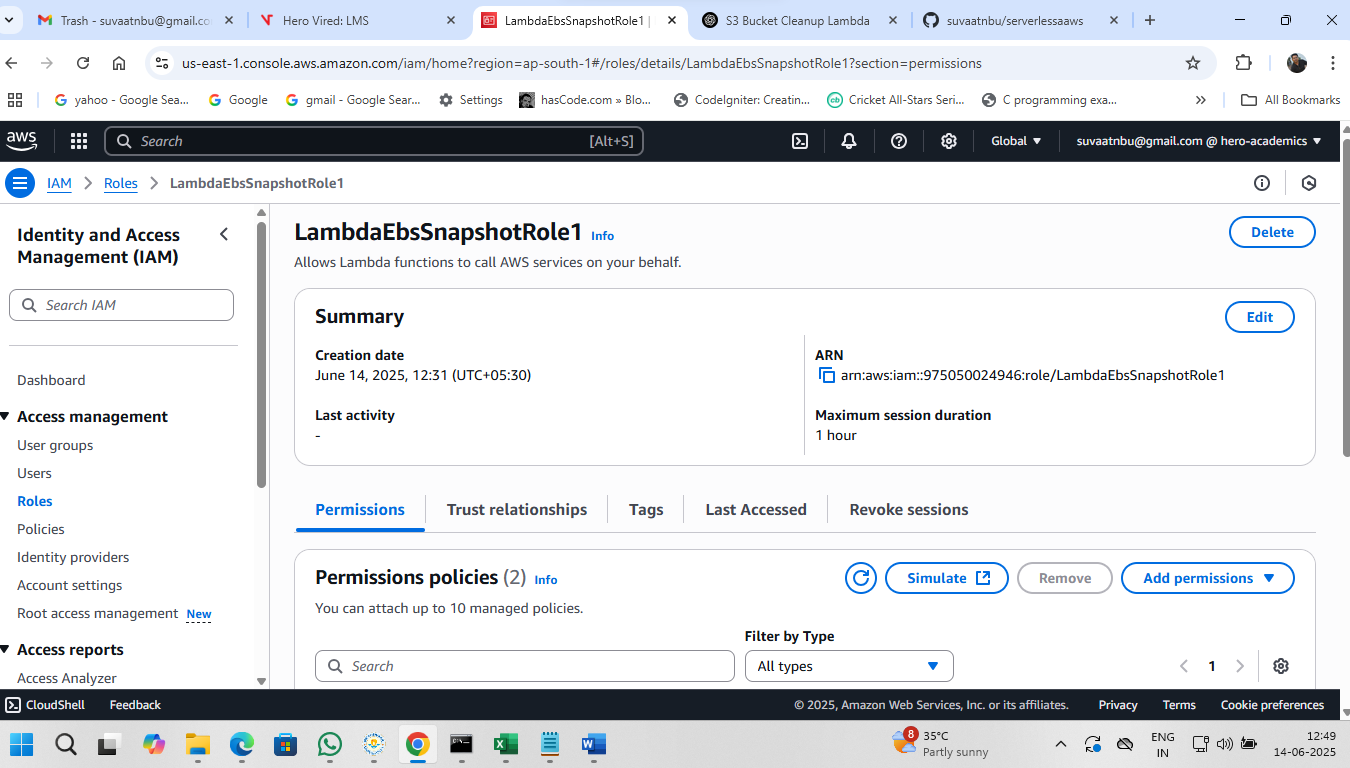
        start\_time = snap['StartTime']

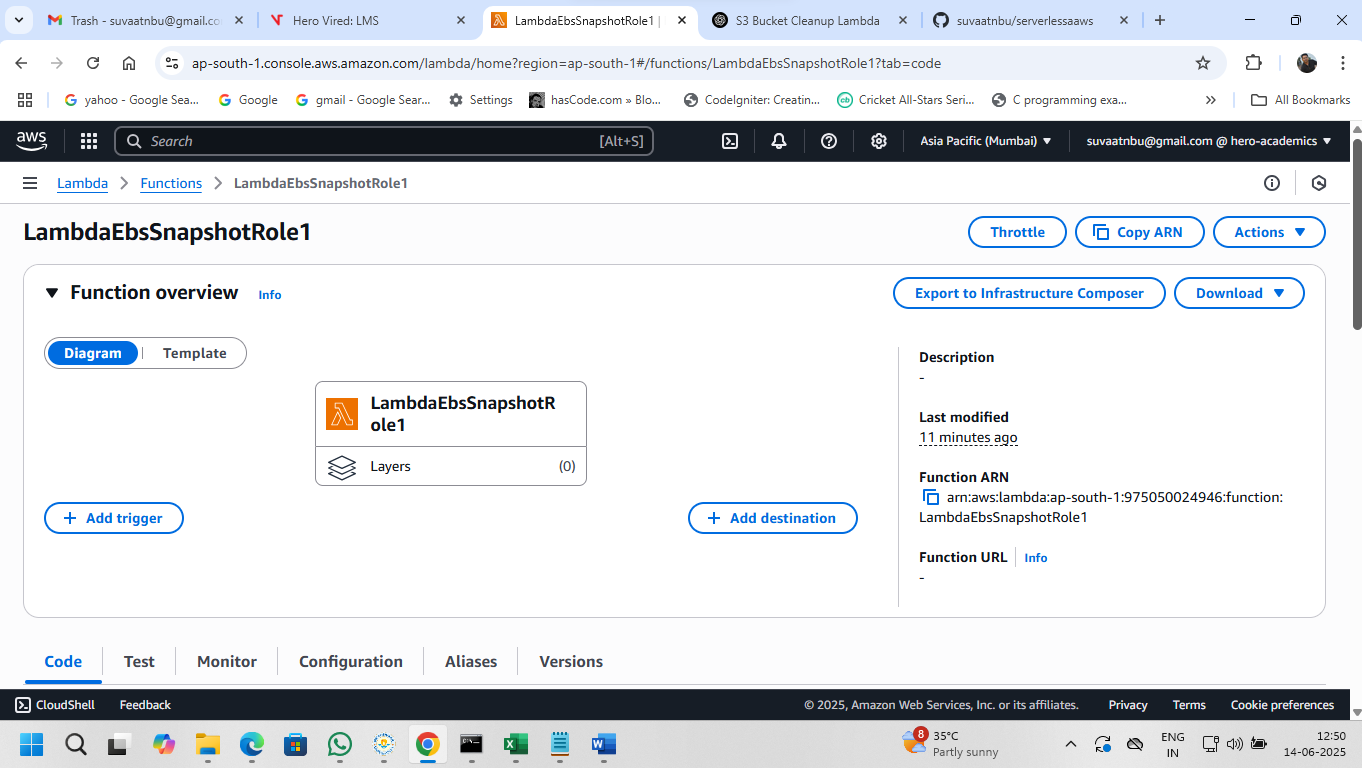
        if start\_time < delete\_before:

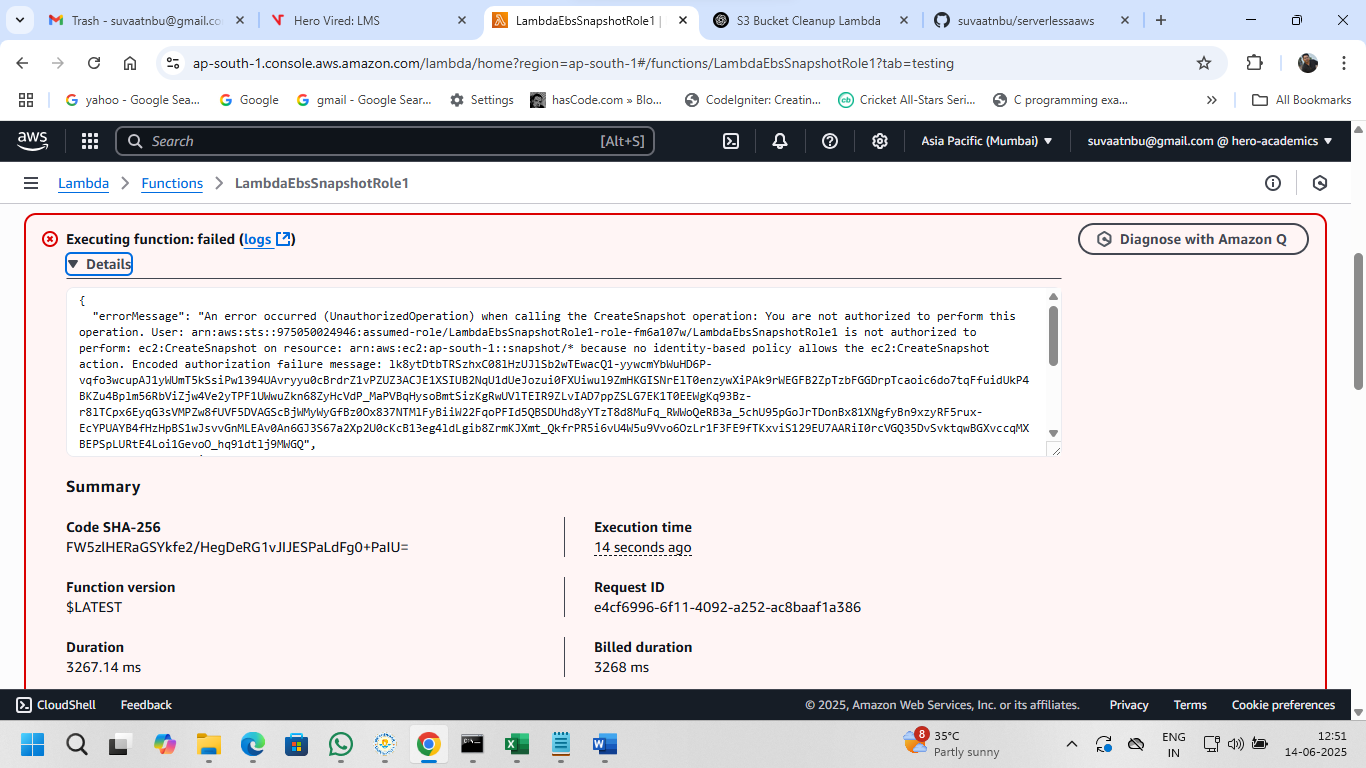
            ec2.delete\_snapshot(SnapshotId=snap['SnapshotId'])

            print(f"🗑️ Deleted old snapshot: {snap['SnapshotId']} from {start\_time}")









5. (Bonus) Add Scheduled Trigger (CloudWatch Events)

1. Go to Lambda > Your Function > Add Trigger
2. Select EventBridge (CloudWatch Events)
3. Choose "Create a new rule"
   * Name: ebs-snapshot-schedule
   * Schedule Expression: rate(7 days) *(or cron(...))*
4. Save — now your backup runs weekly!

**Add a Custom Inline Policy (More restrictive)**

If you prefer a safer policy, use this:

**🔐 Inline Policy JSON:**

json

CopyEdit

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:CreateSnapshot",

"ec2:DeleteSnapshot",

"ec2:DescribeSnapshots"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "ec2:DescribeVolumes",

"Resource": "\*"

}

]

}

To add this:

* IAM > Roles > Your Role > Permissions > **Add inline policy**
* Switch to **JSON** tab, paste above code, and **Save**