**LAMBDA**

**COST-OPTIMIZATION -EBS-VOLUME:**

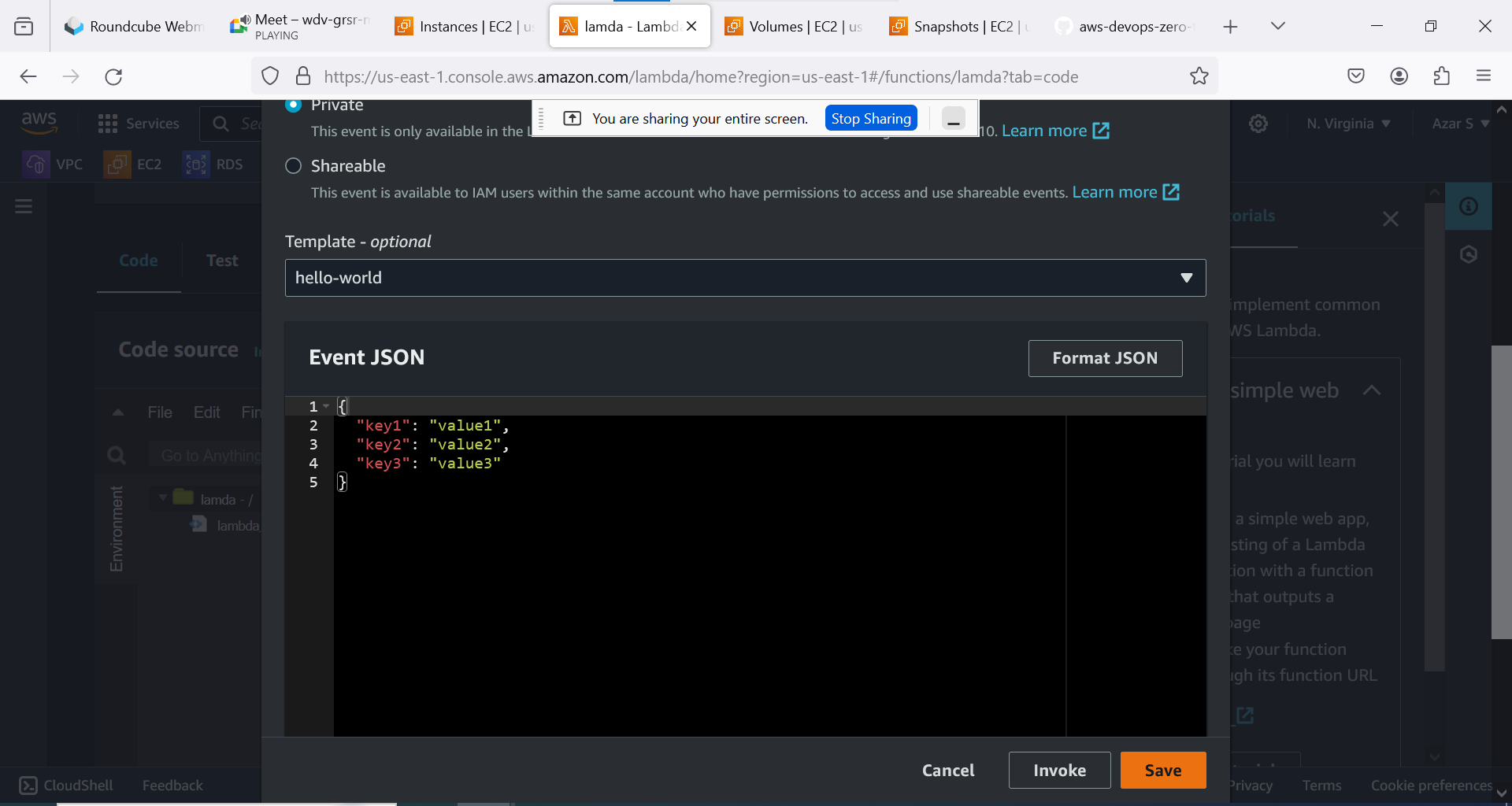
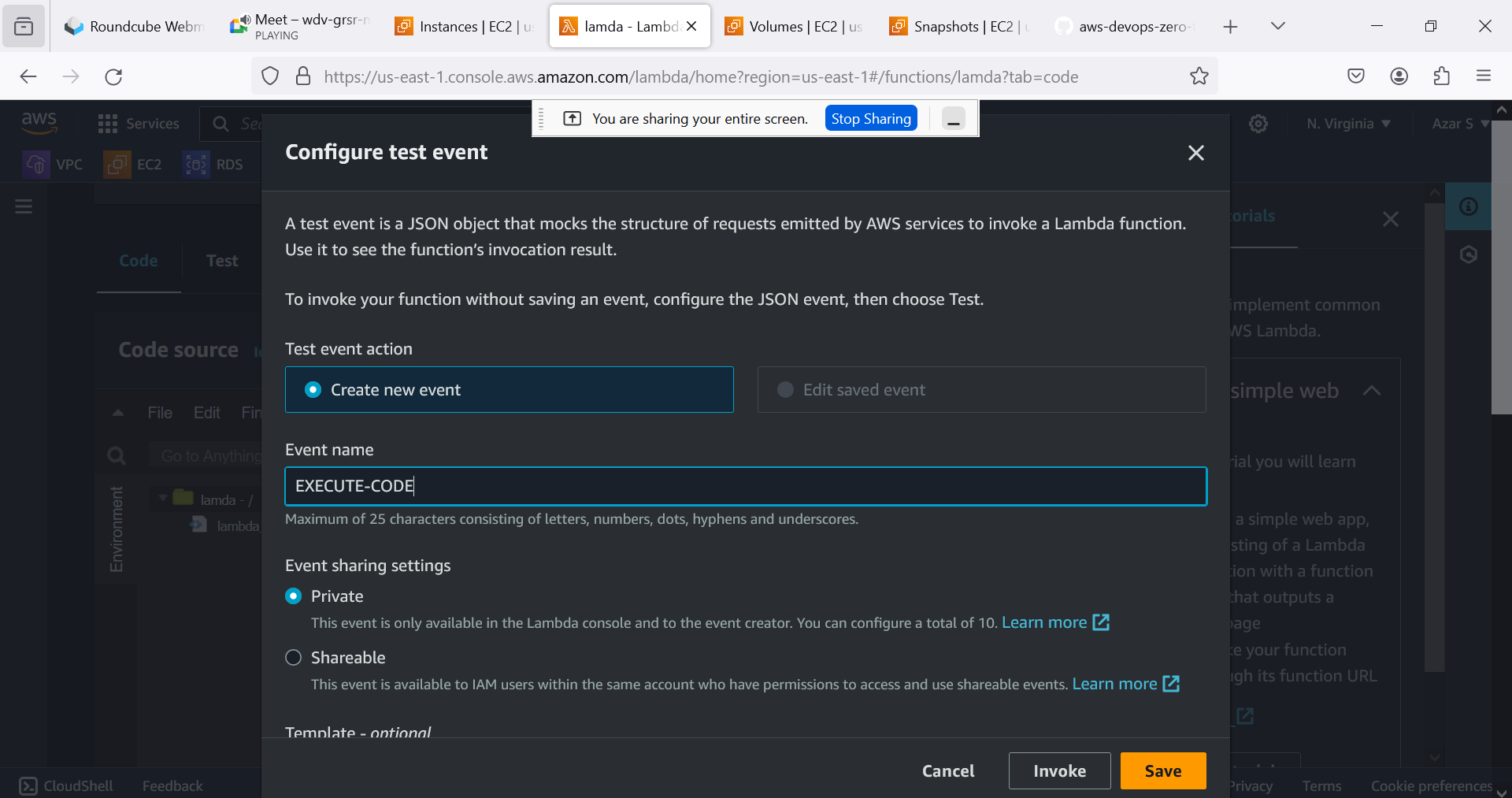
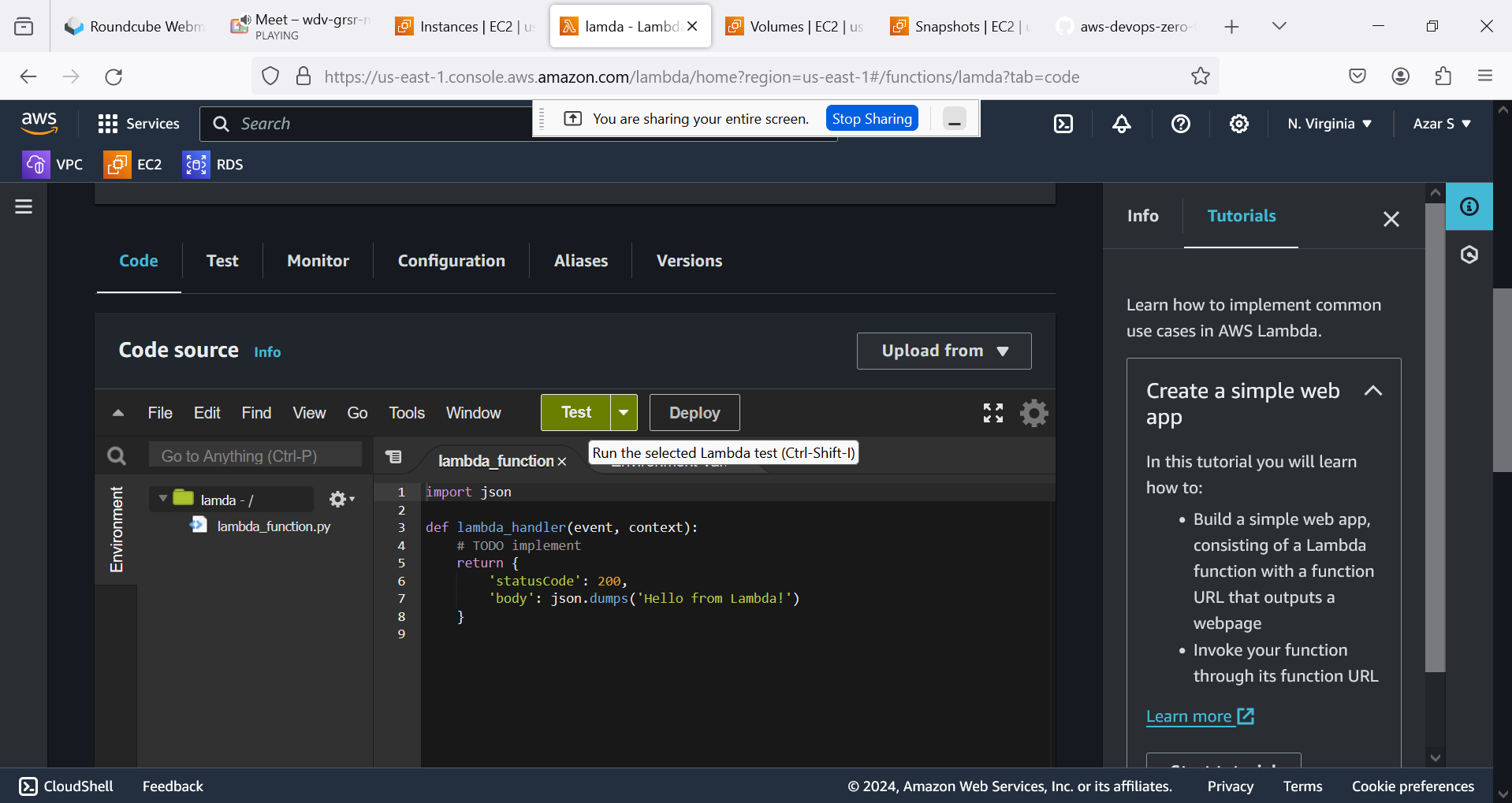
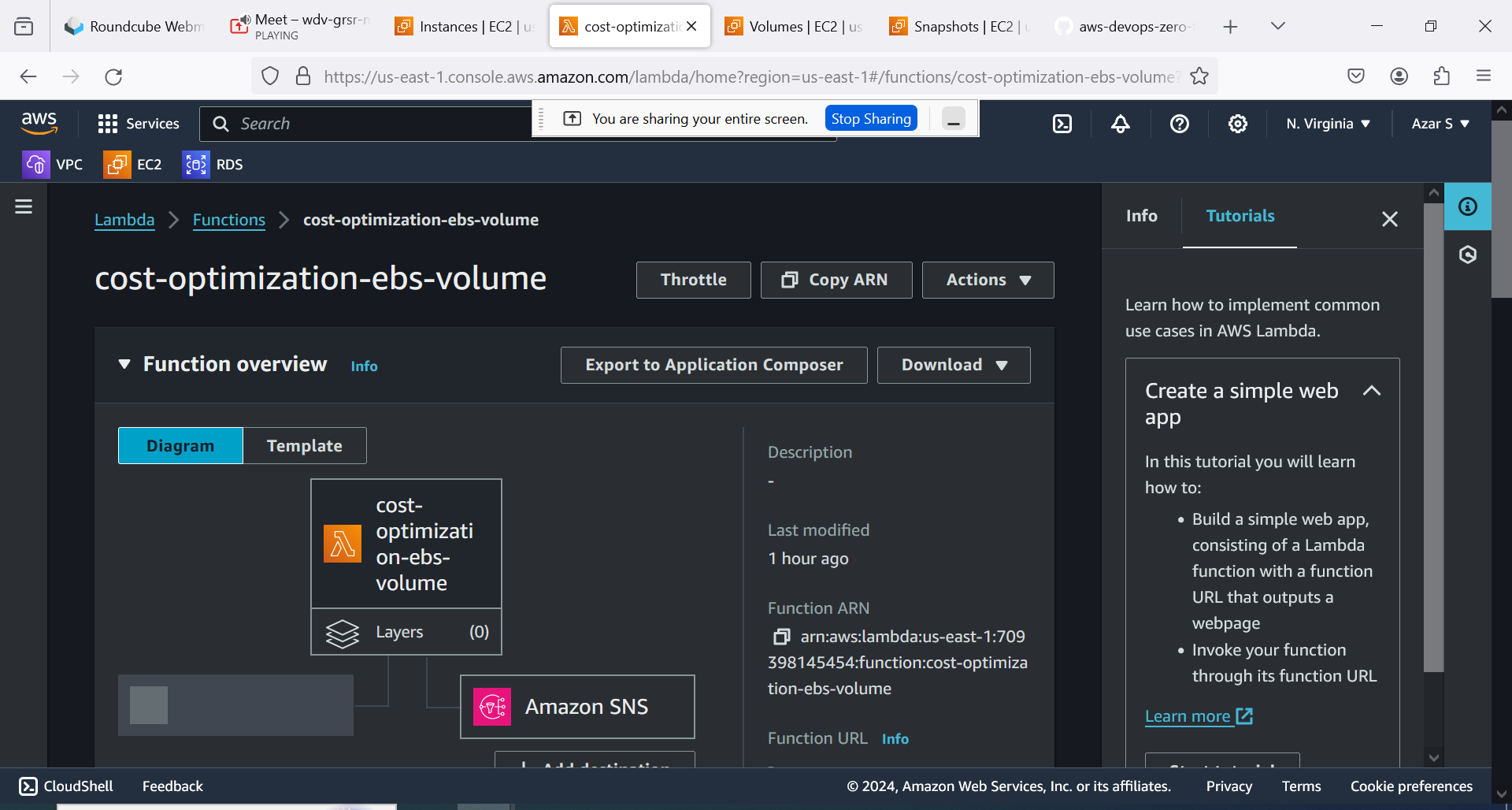
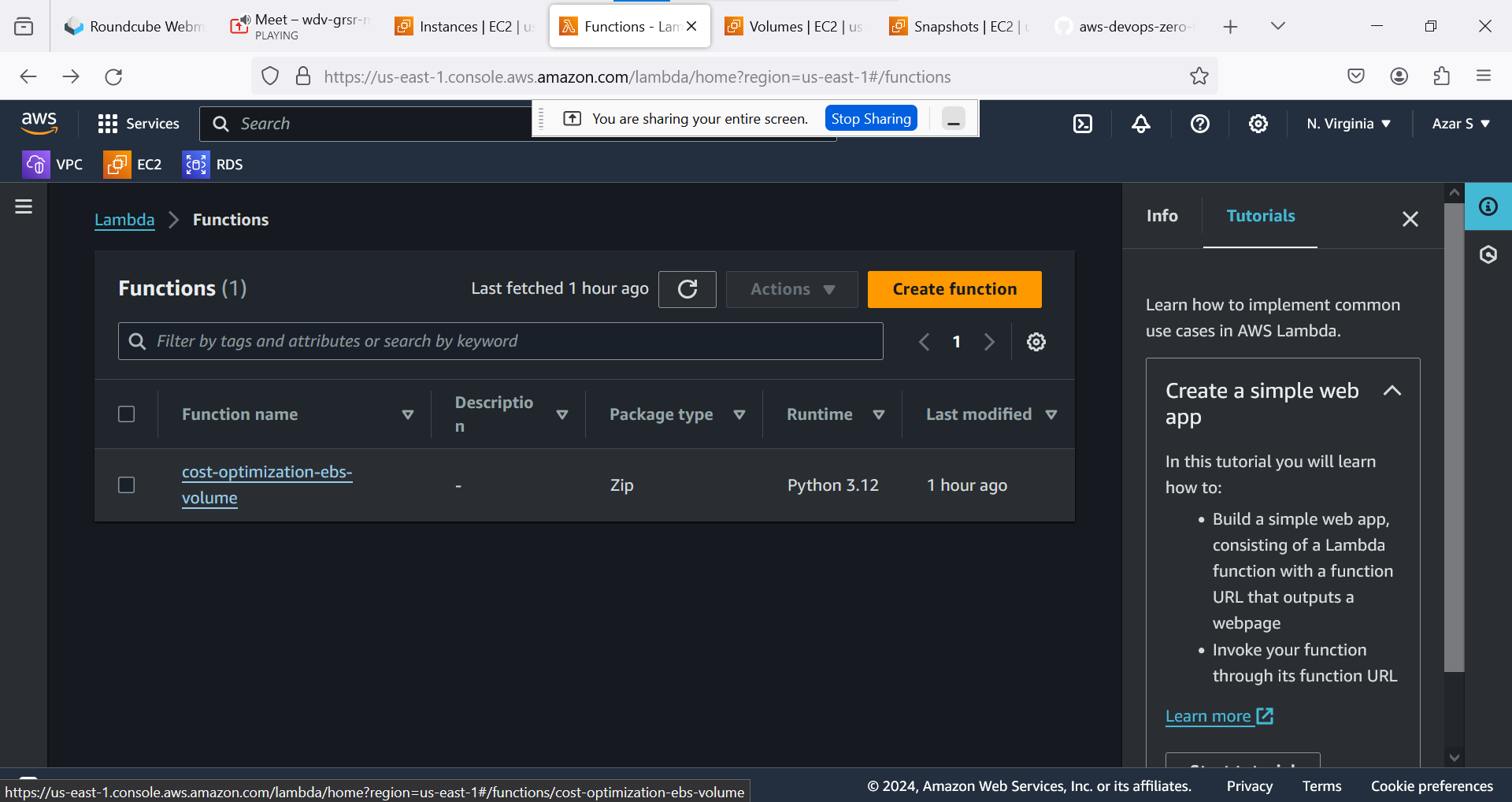
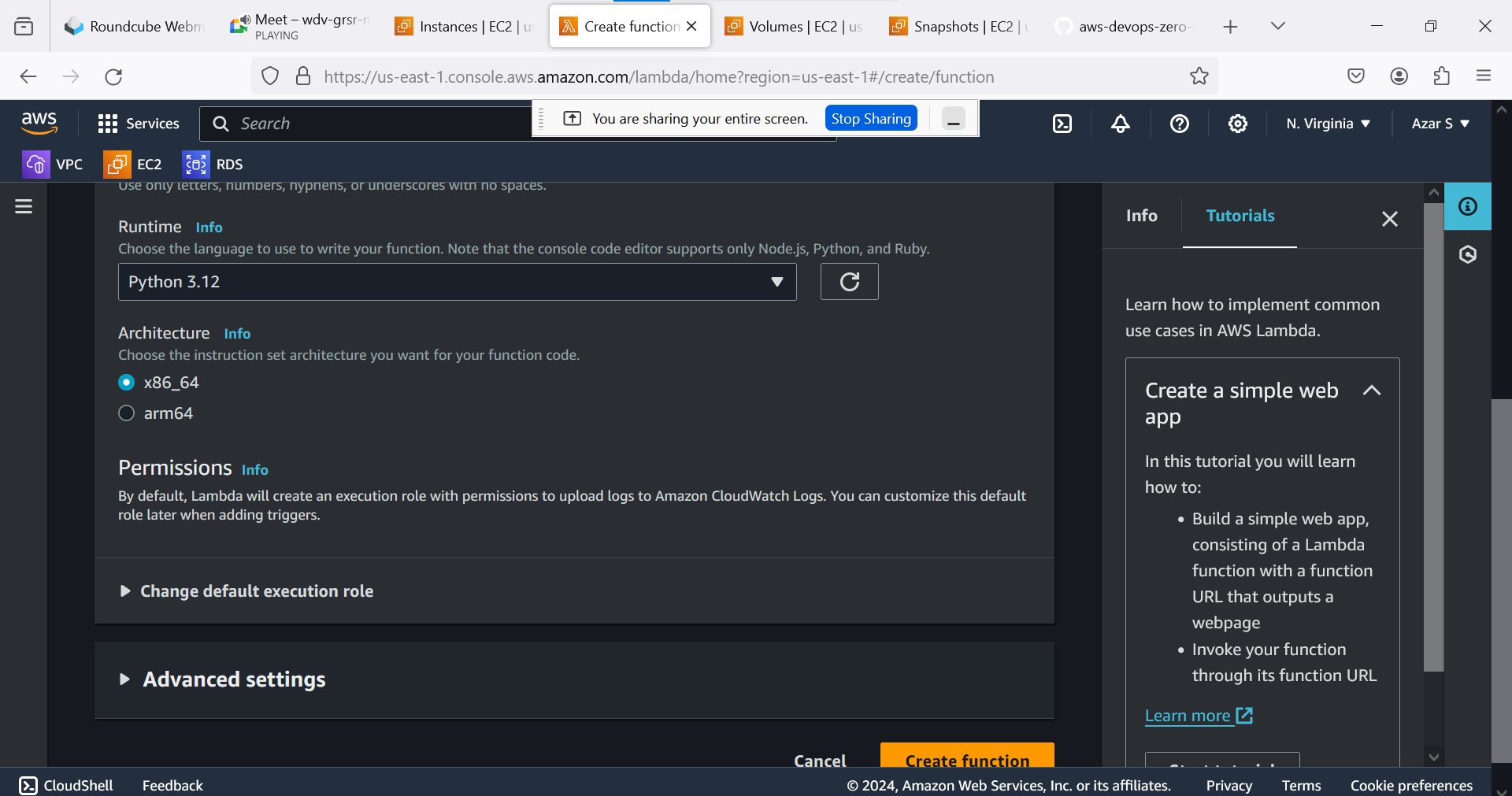
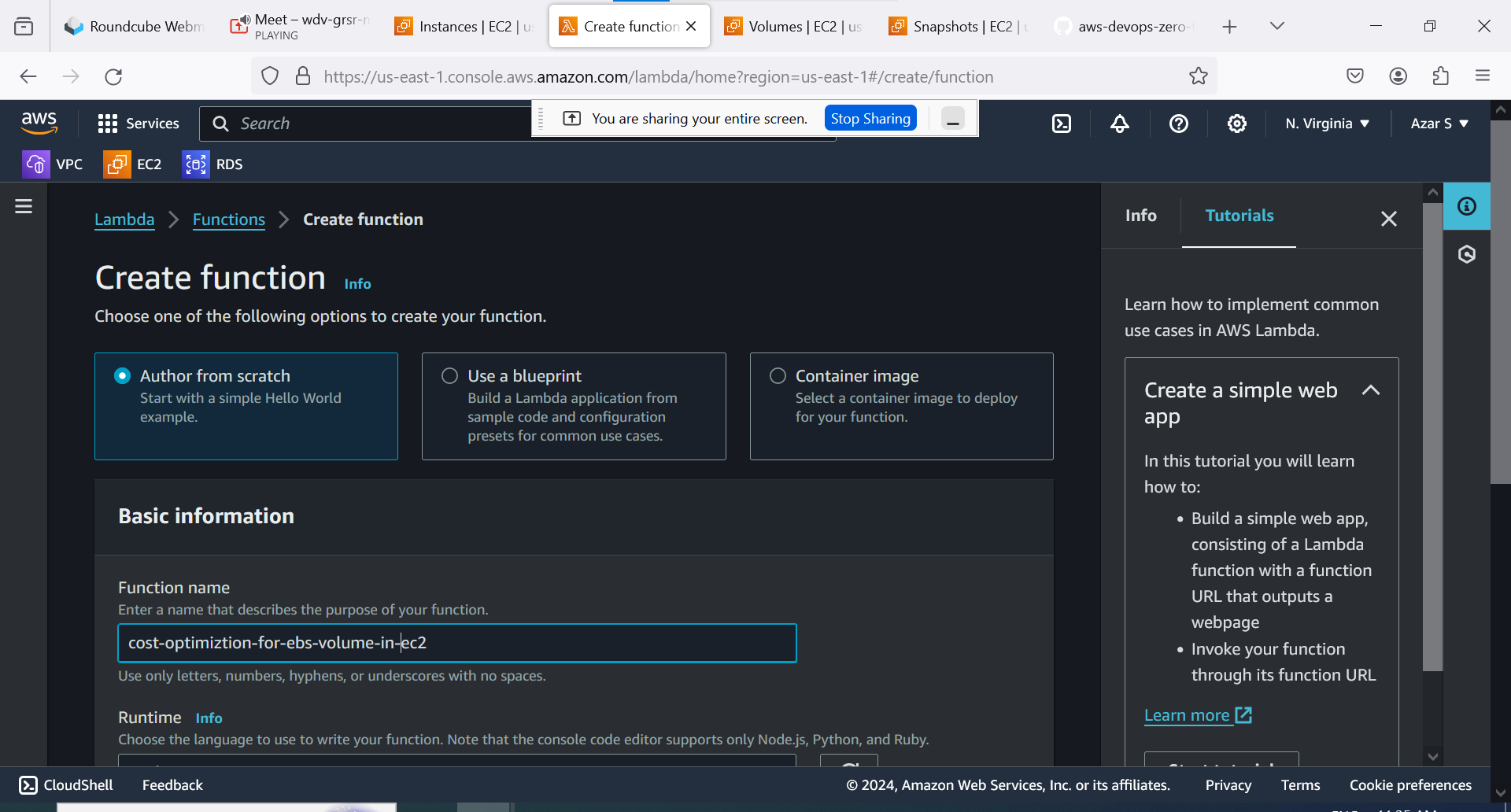
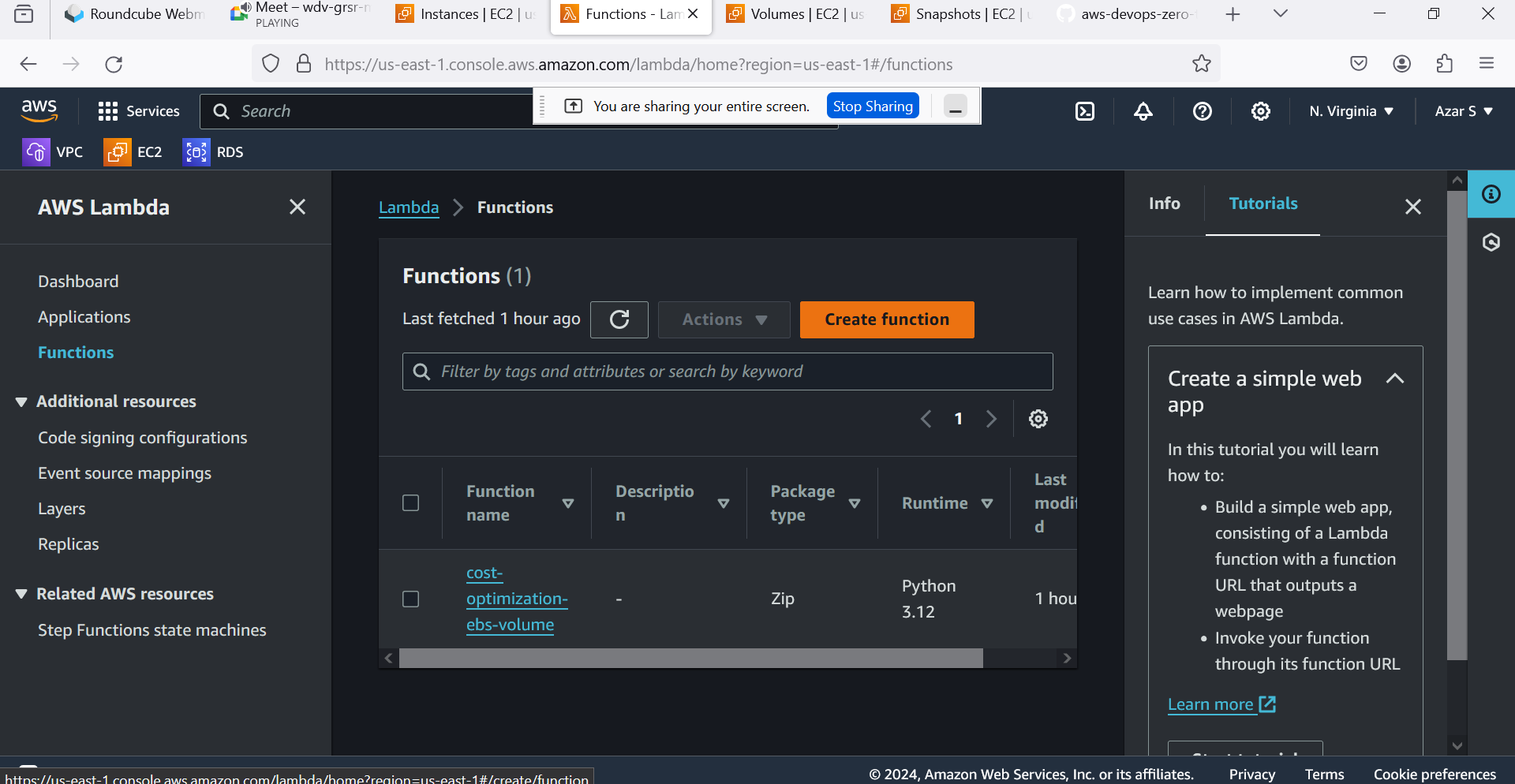
**AWS Cloud Cost Optimization - Identifying Stale Resources**

**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:**

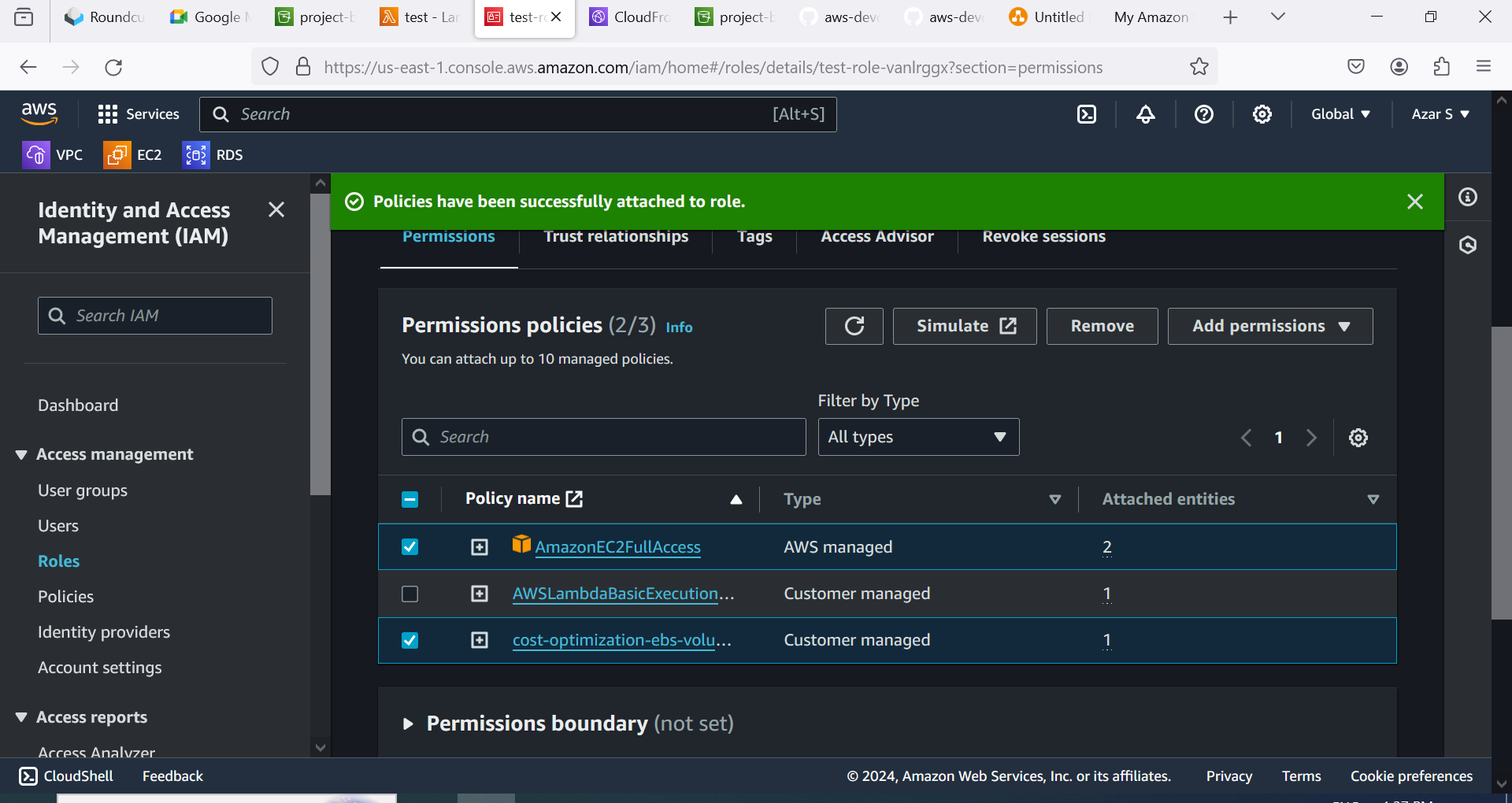
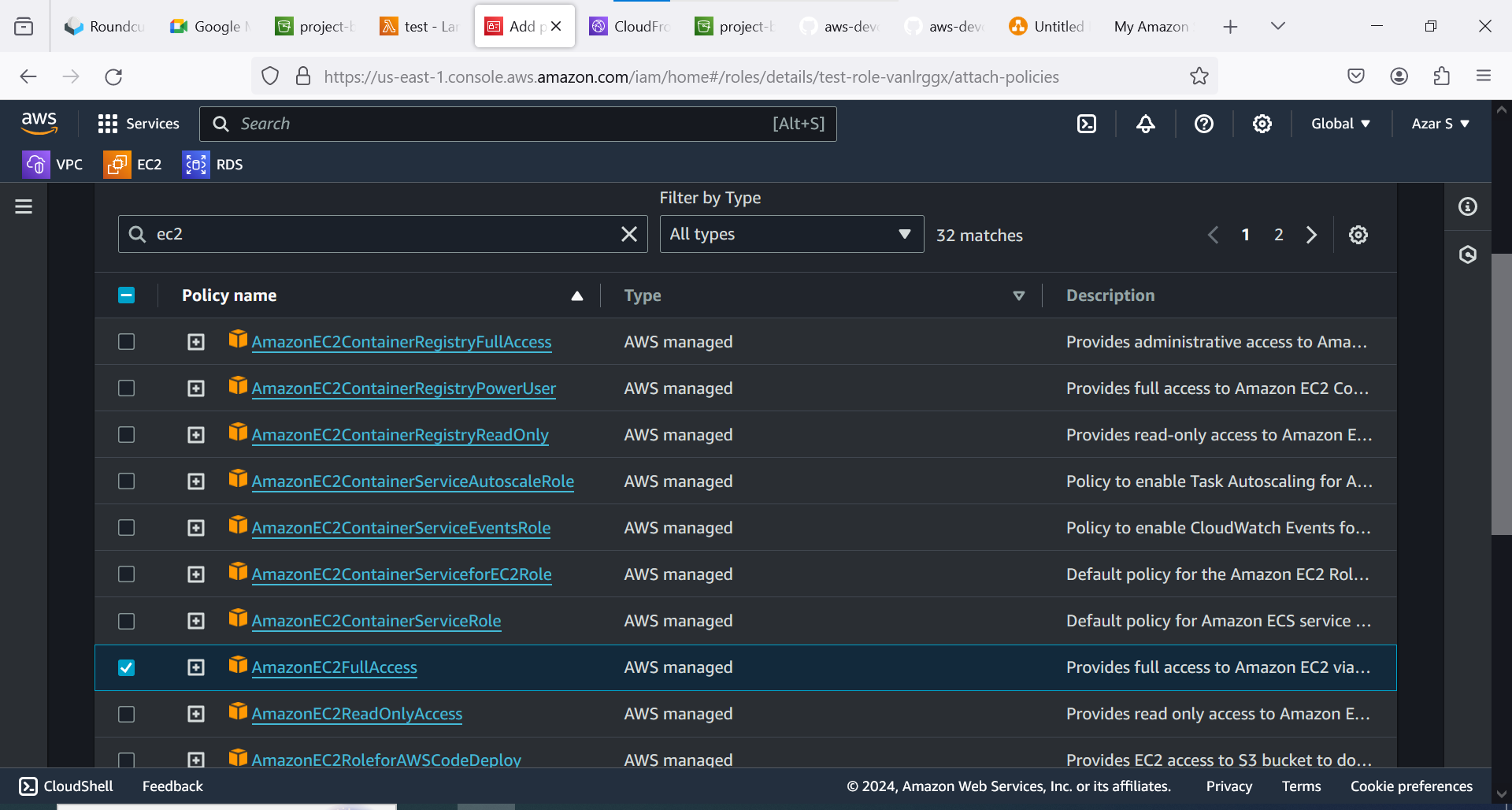
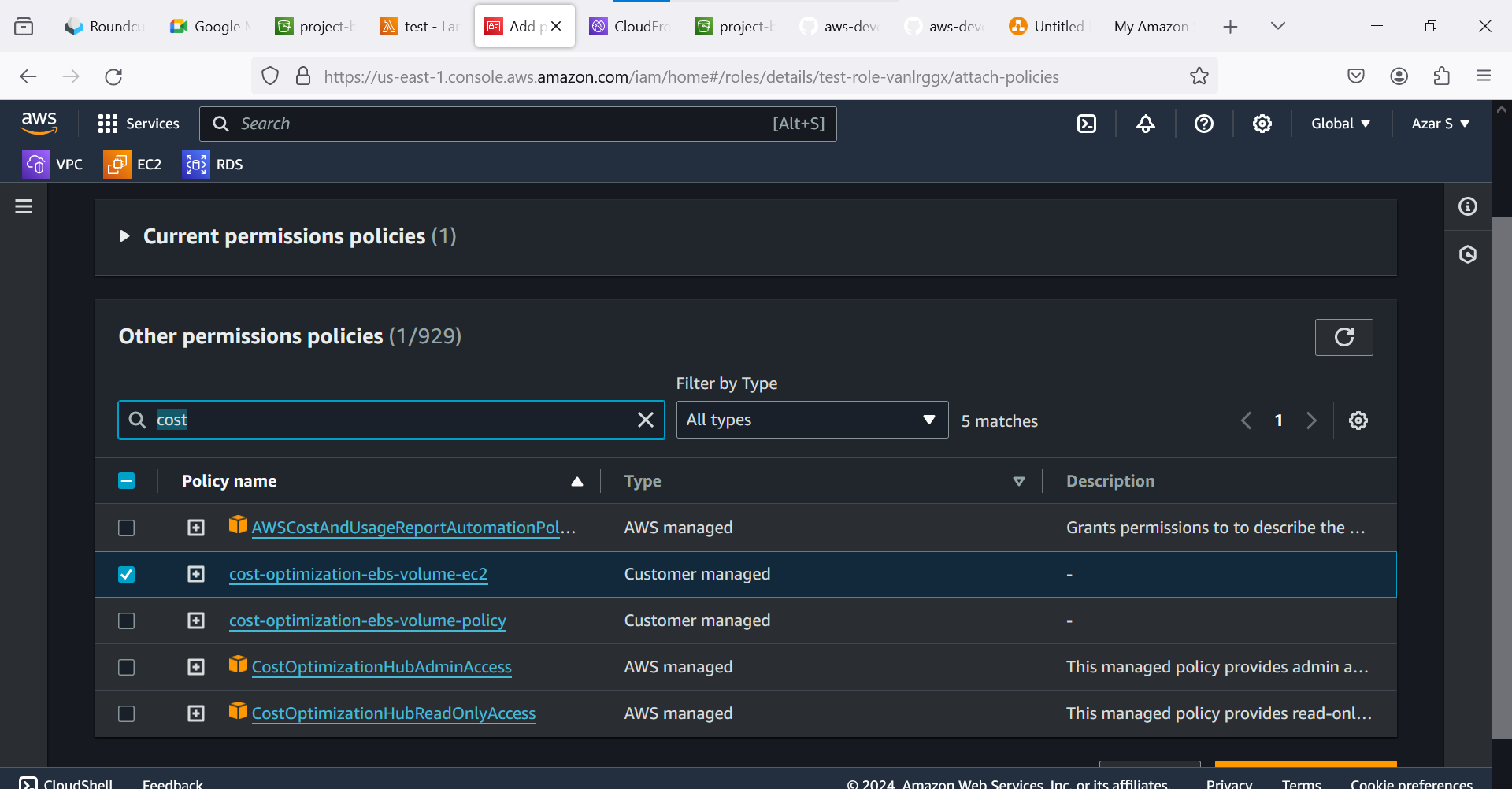
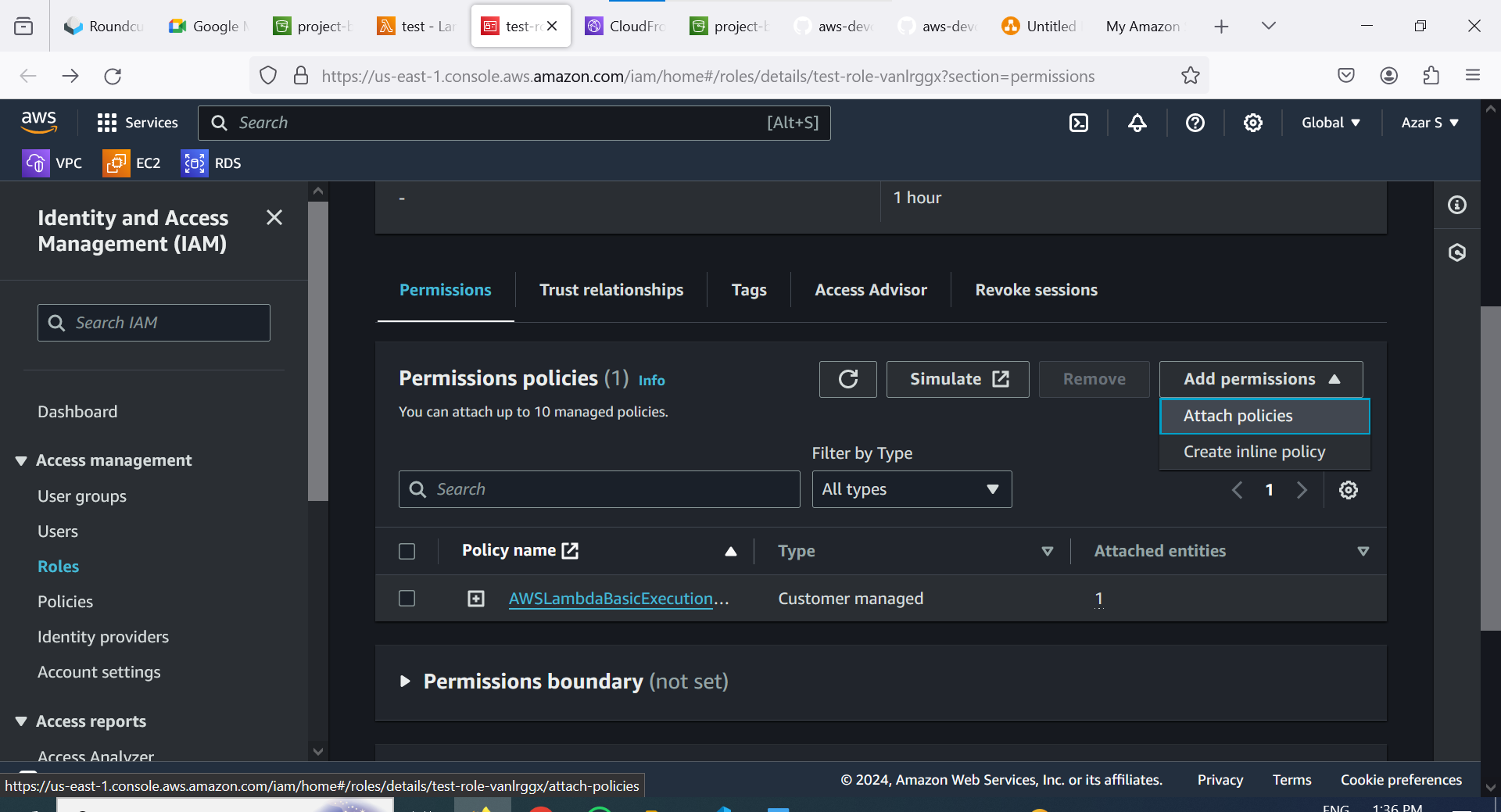
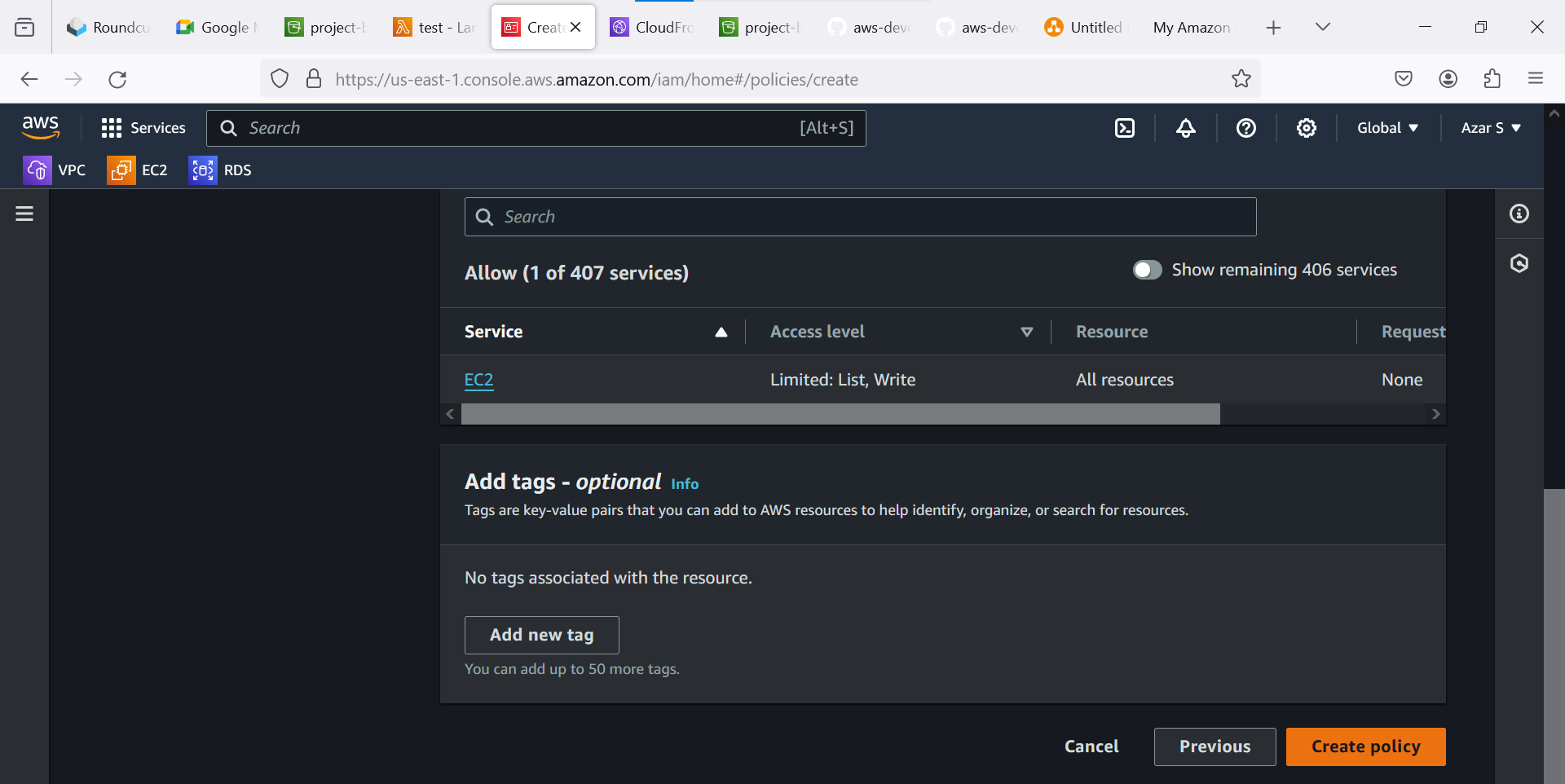
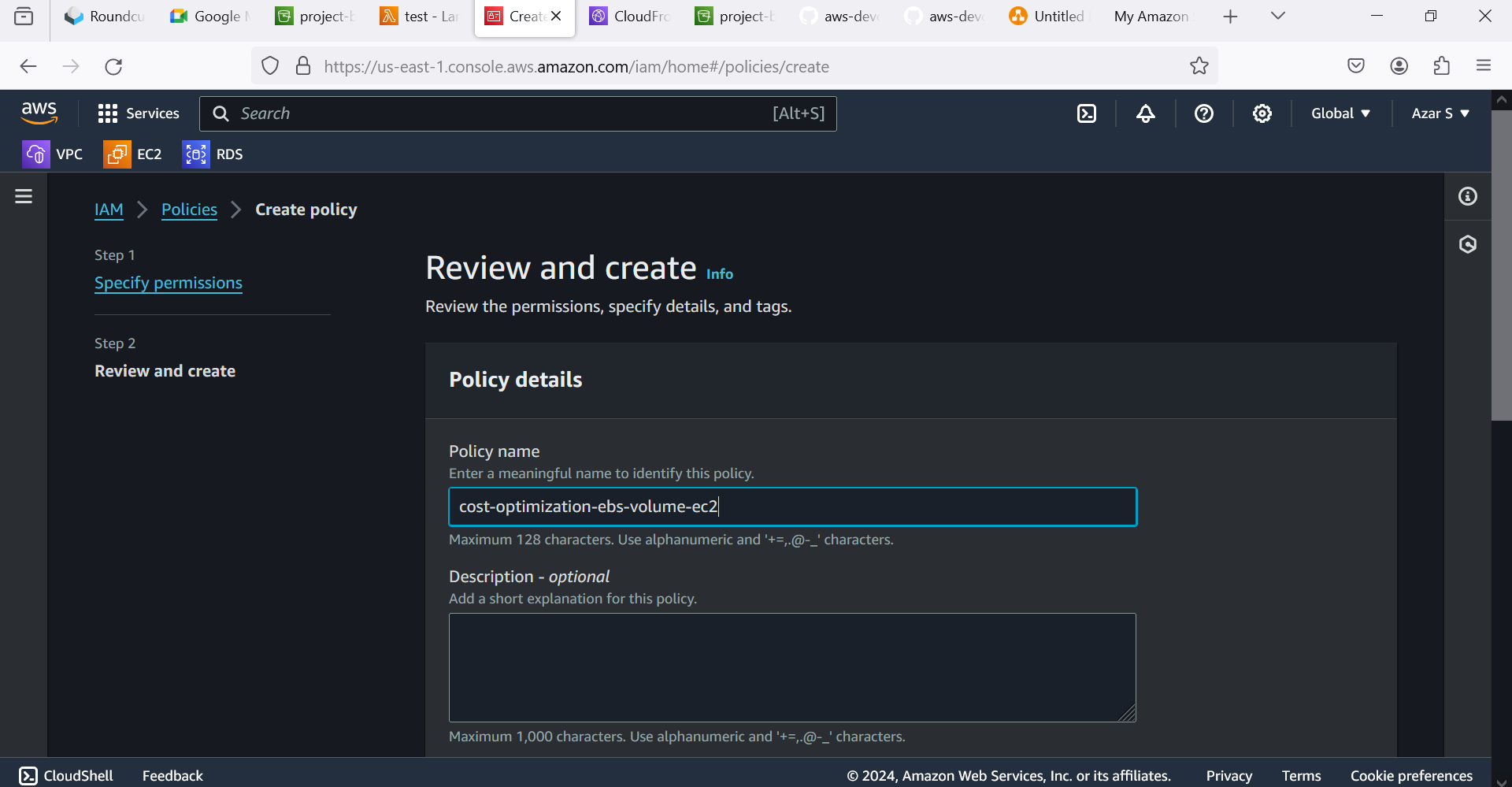
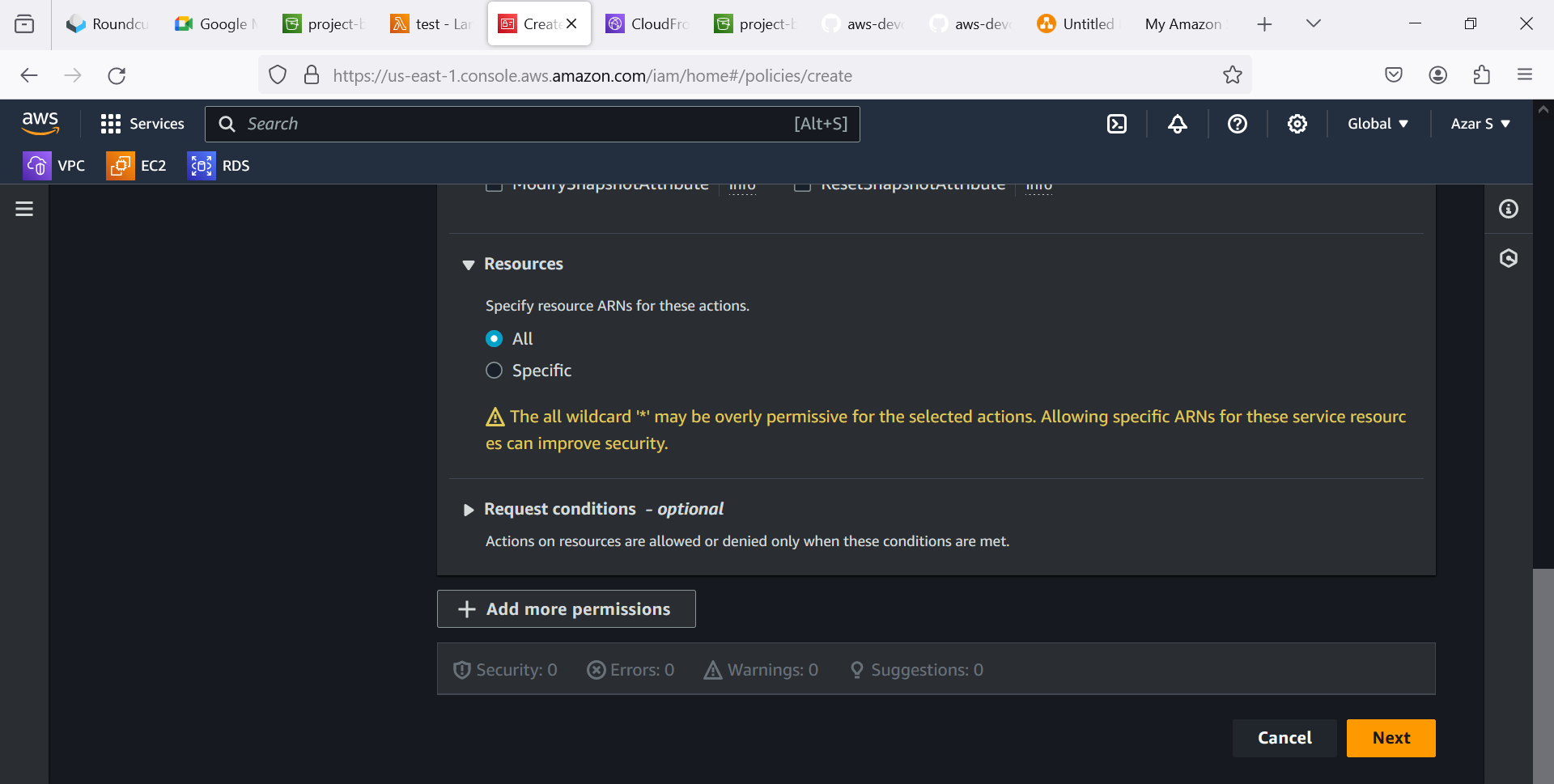
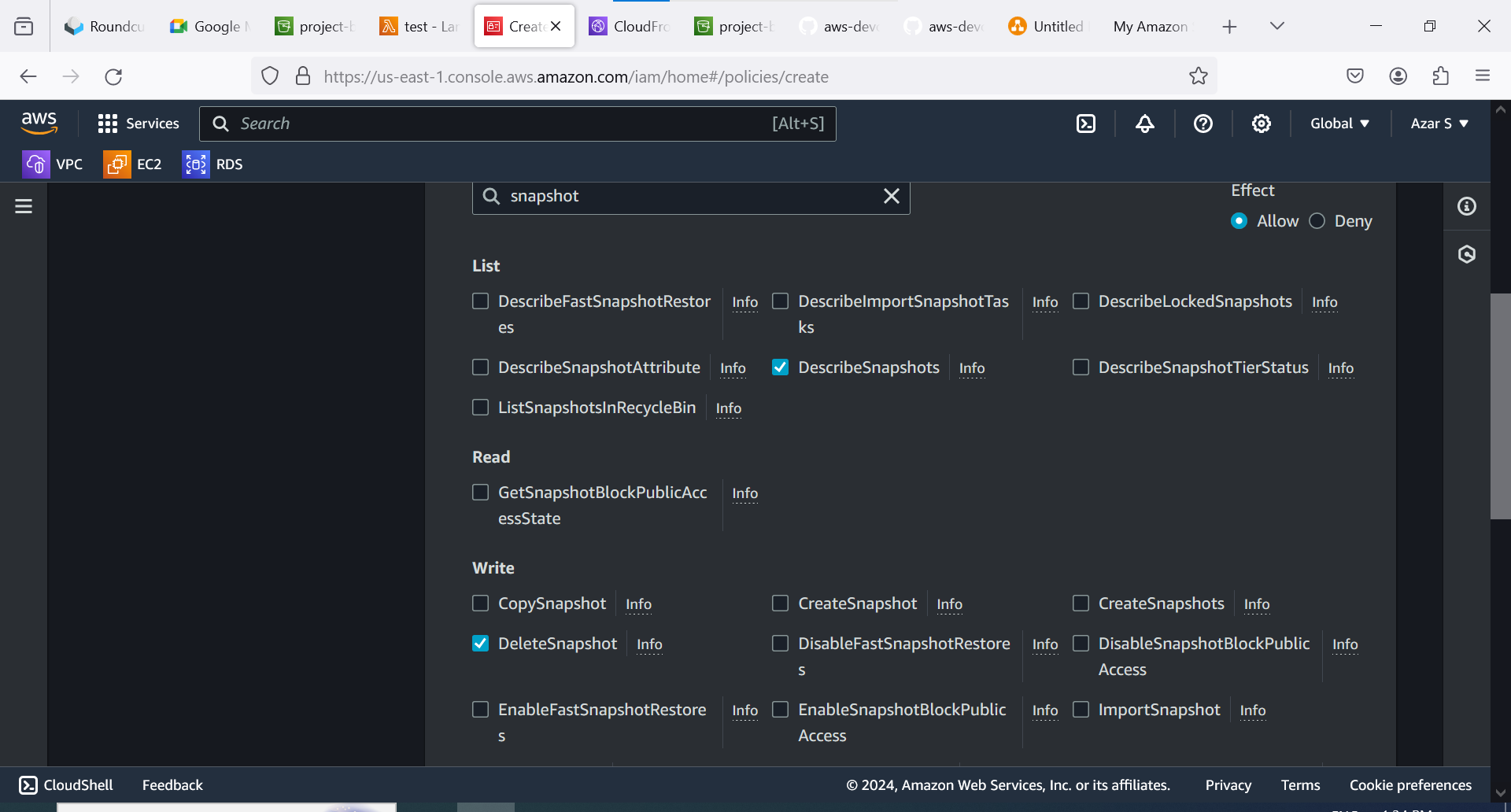
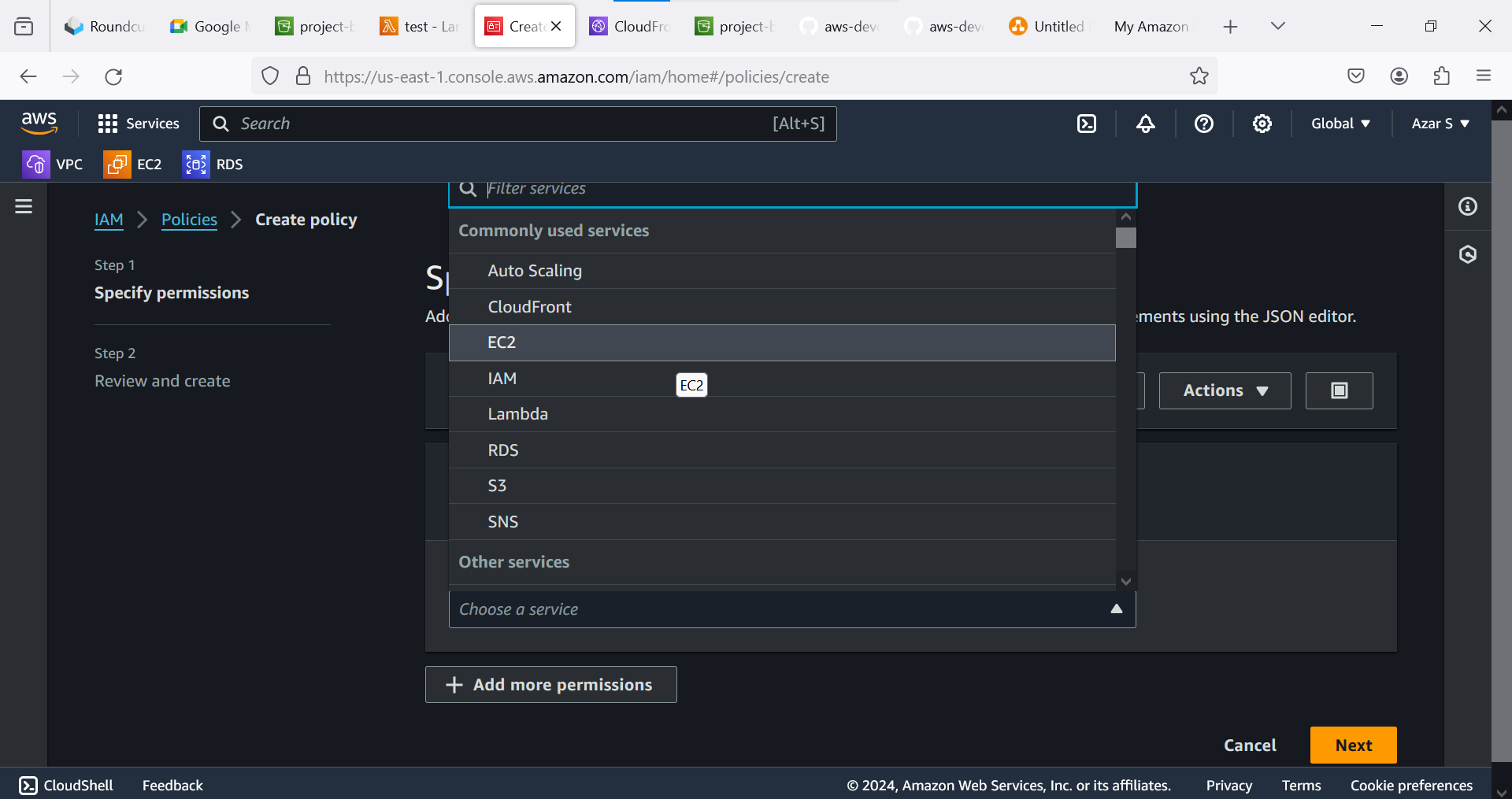
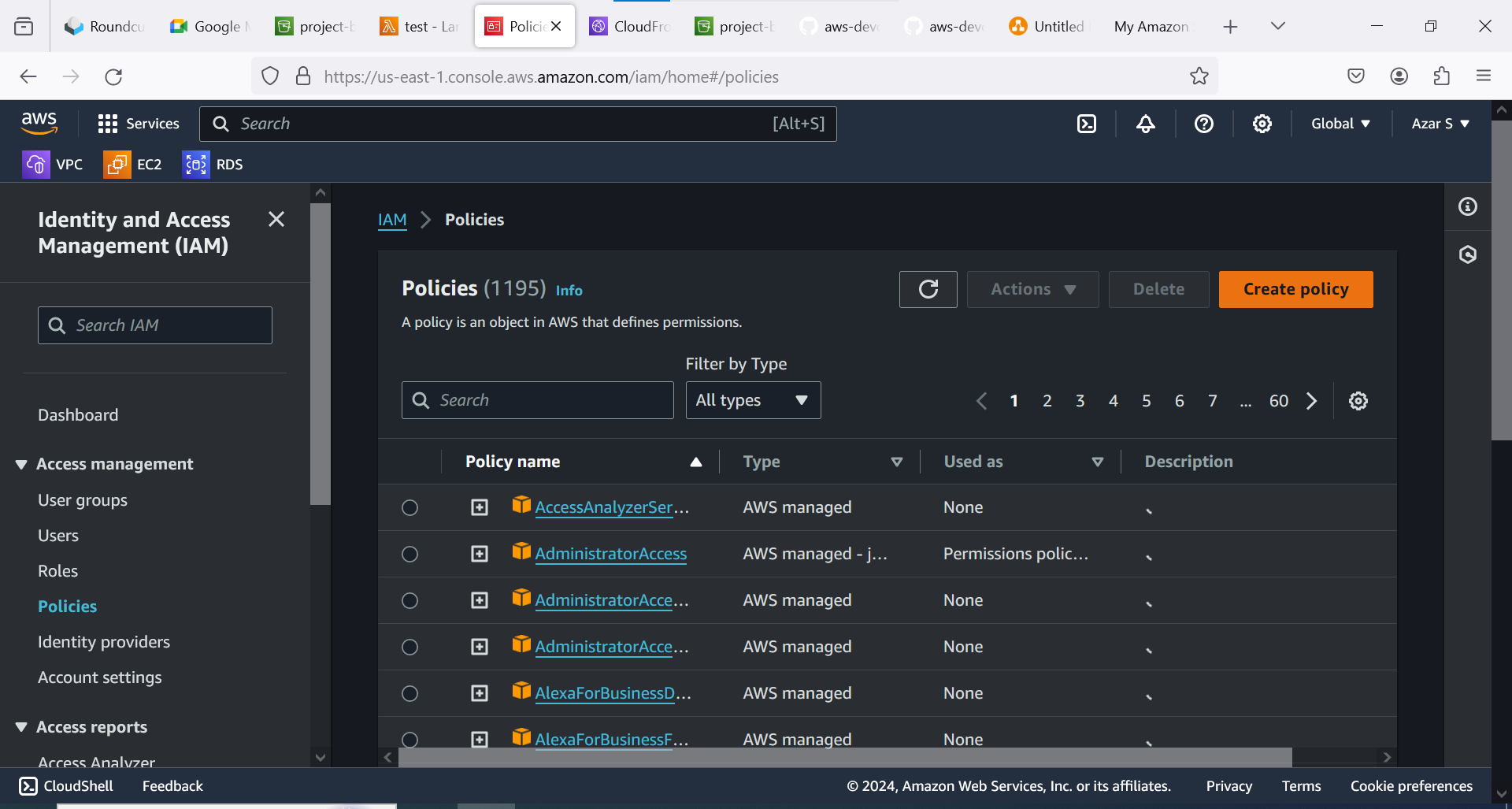
The Lambda function fetches all EBS snapshots owned by the same account ('self') and also retrieves a list of active EC2 instances (running and stopped). For each snapshot, it checks if the associated volume (if exists) is not associated with any active instance. If it finds a stale snapshot, it deletes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

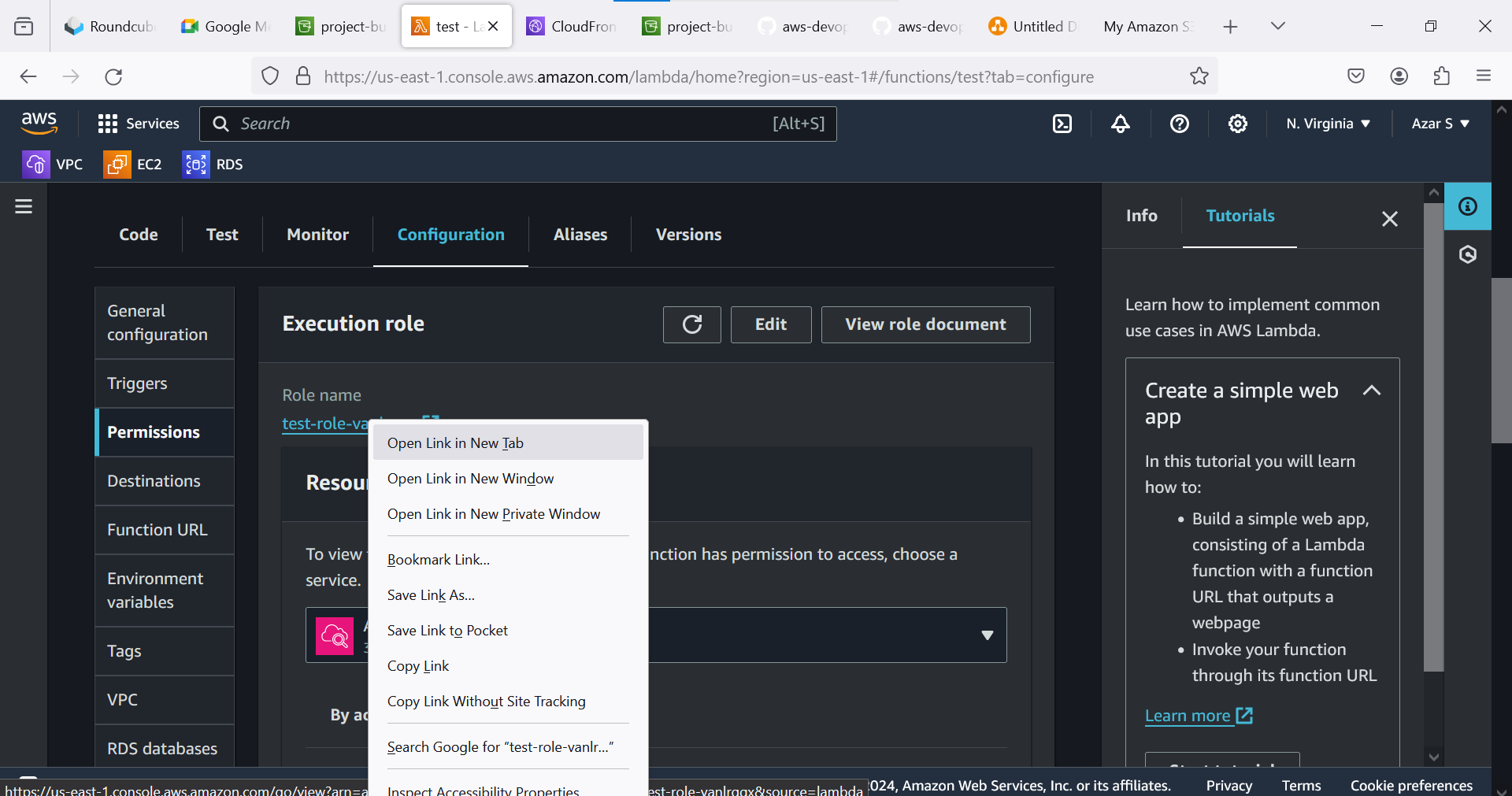


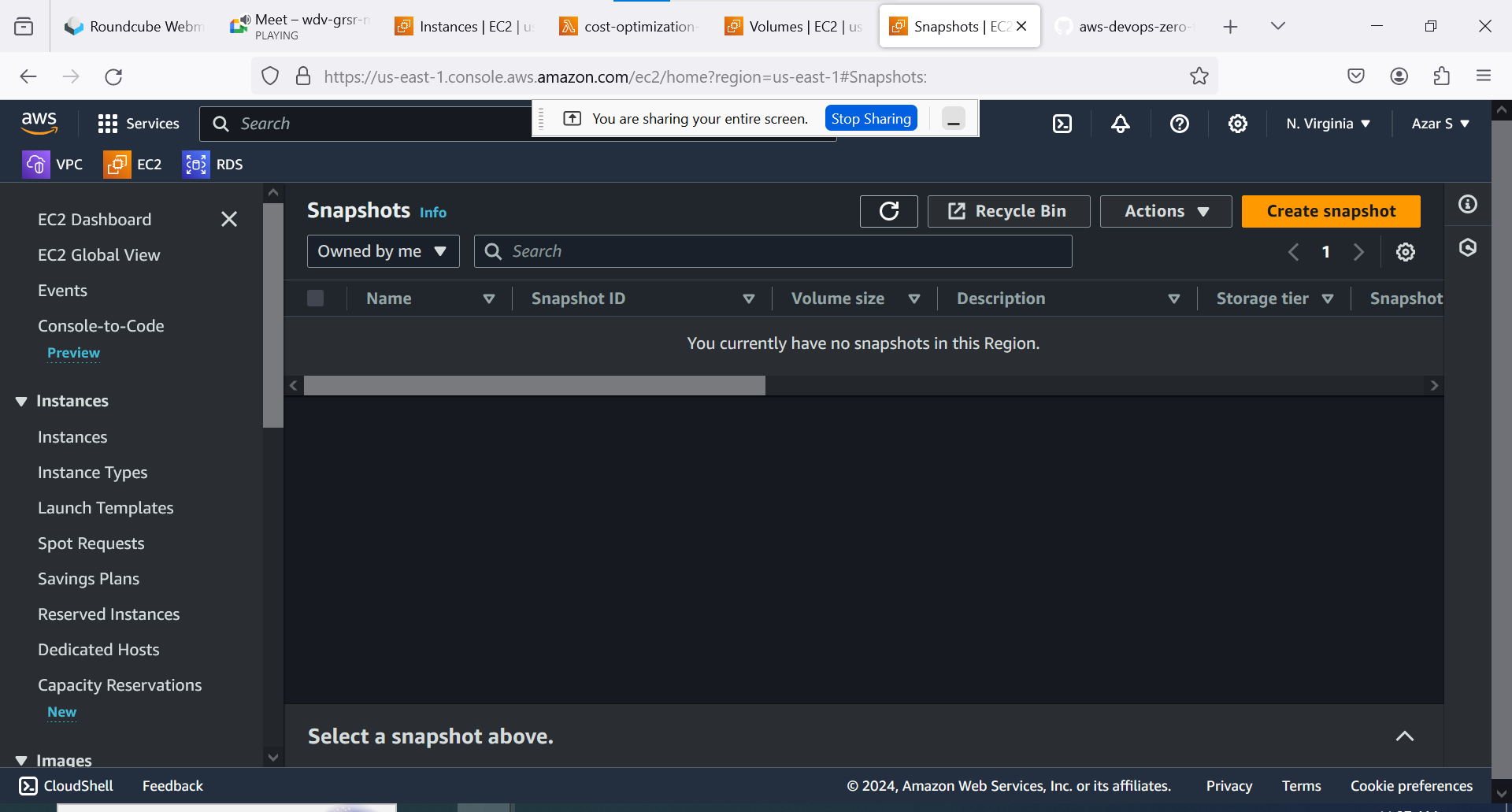
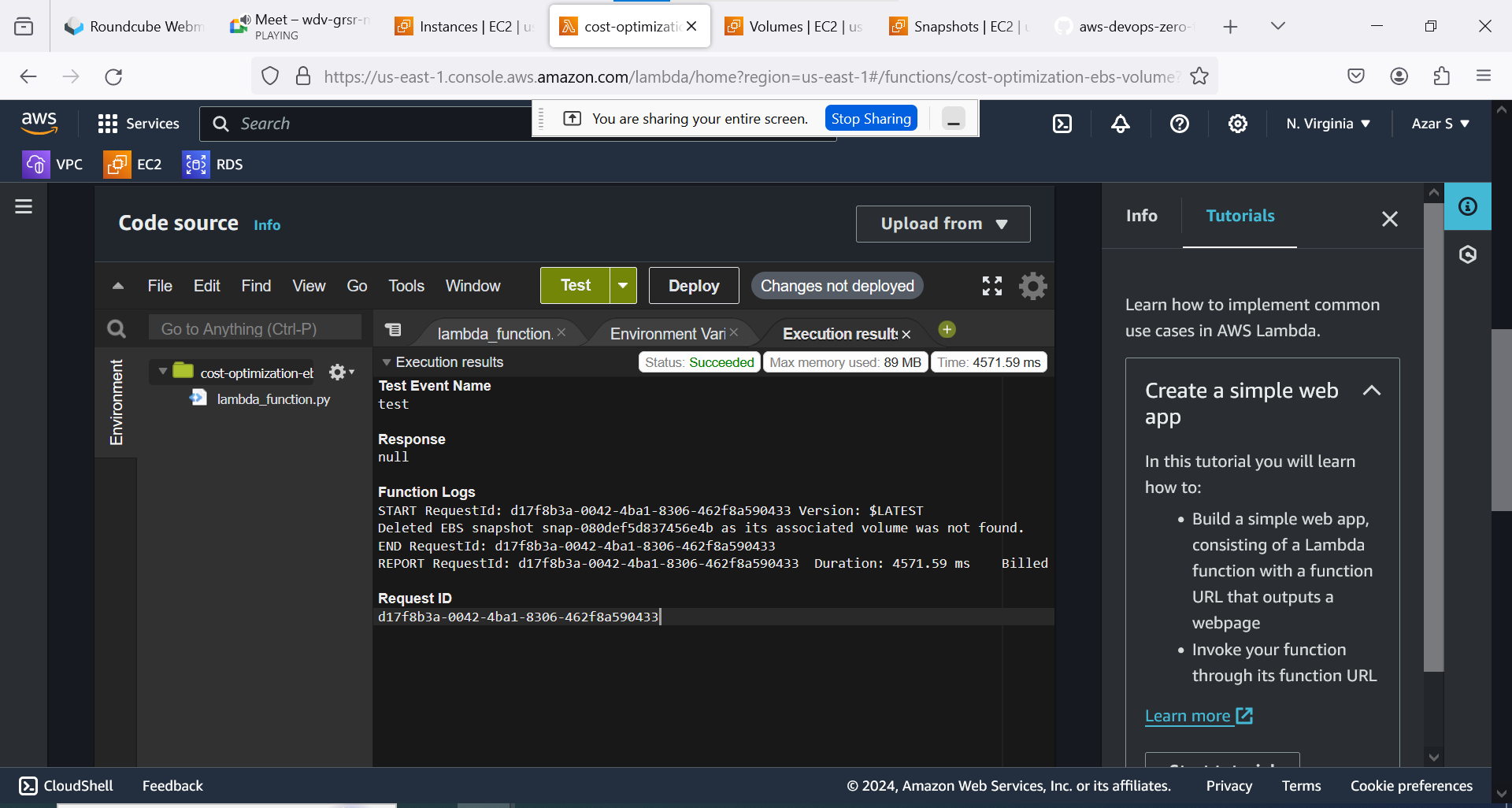
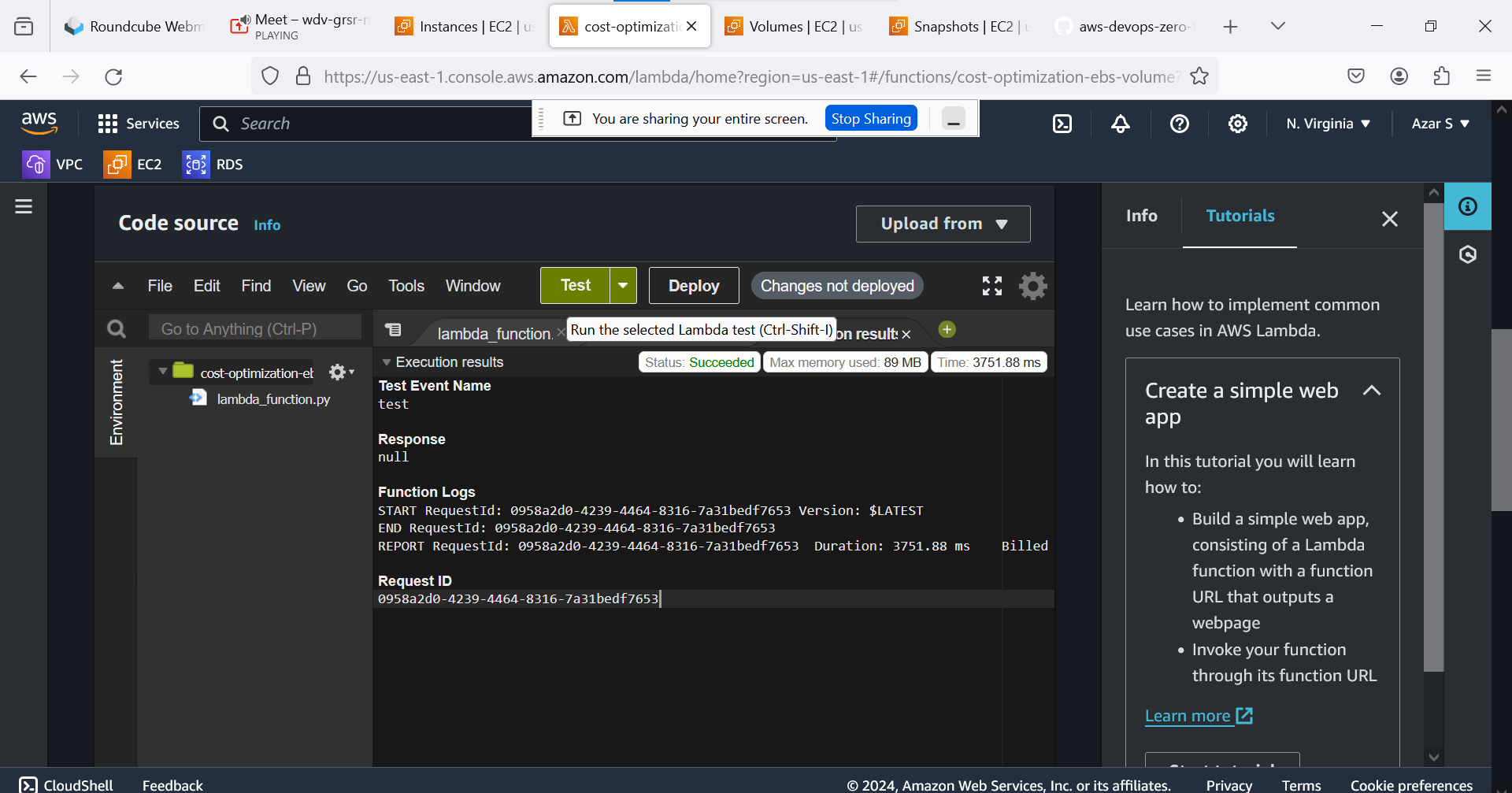
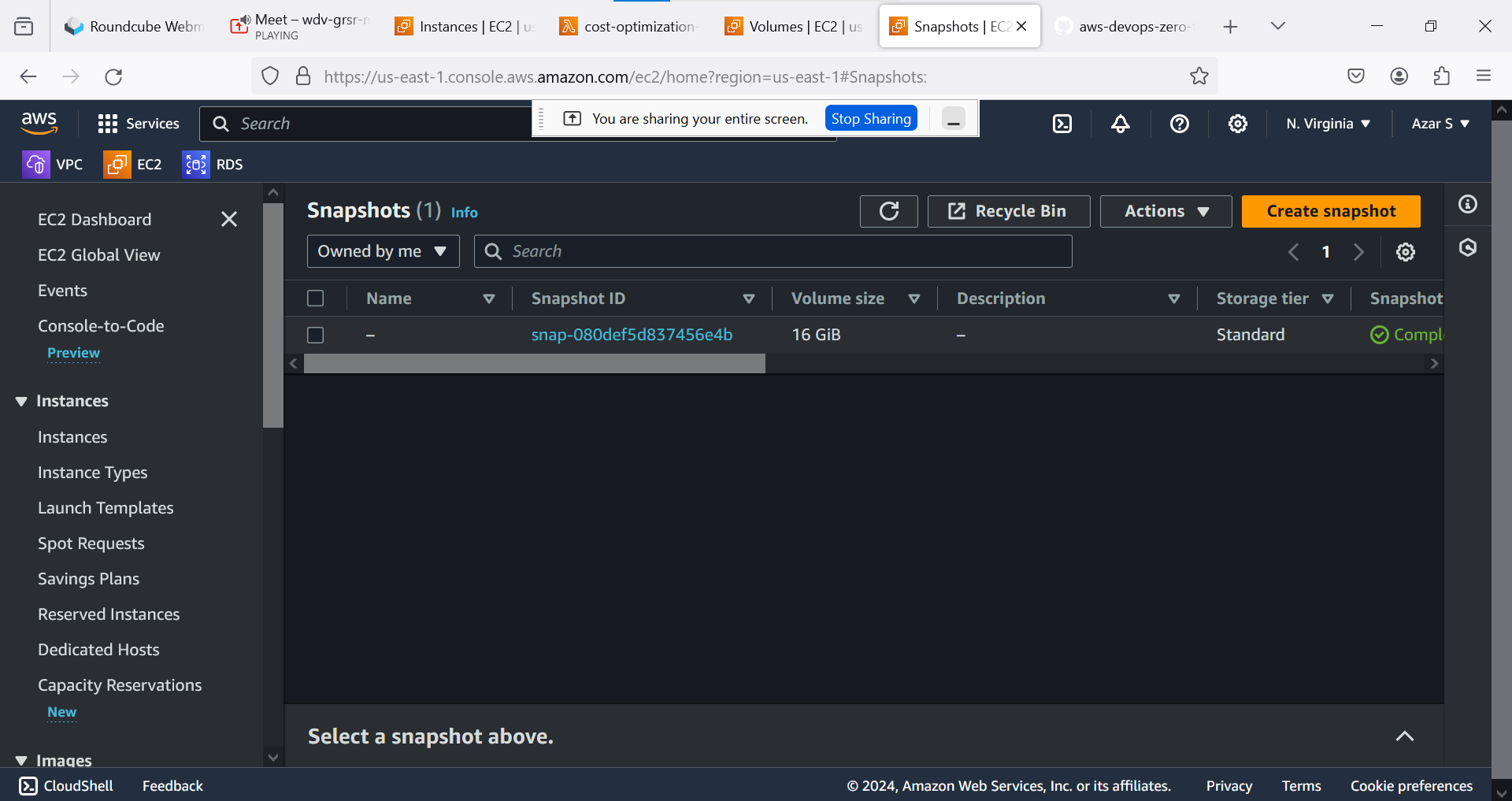
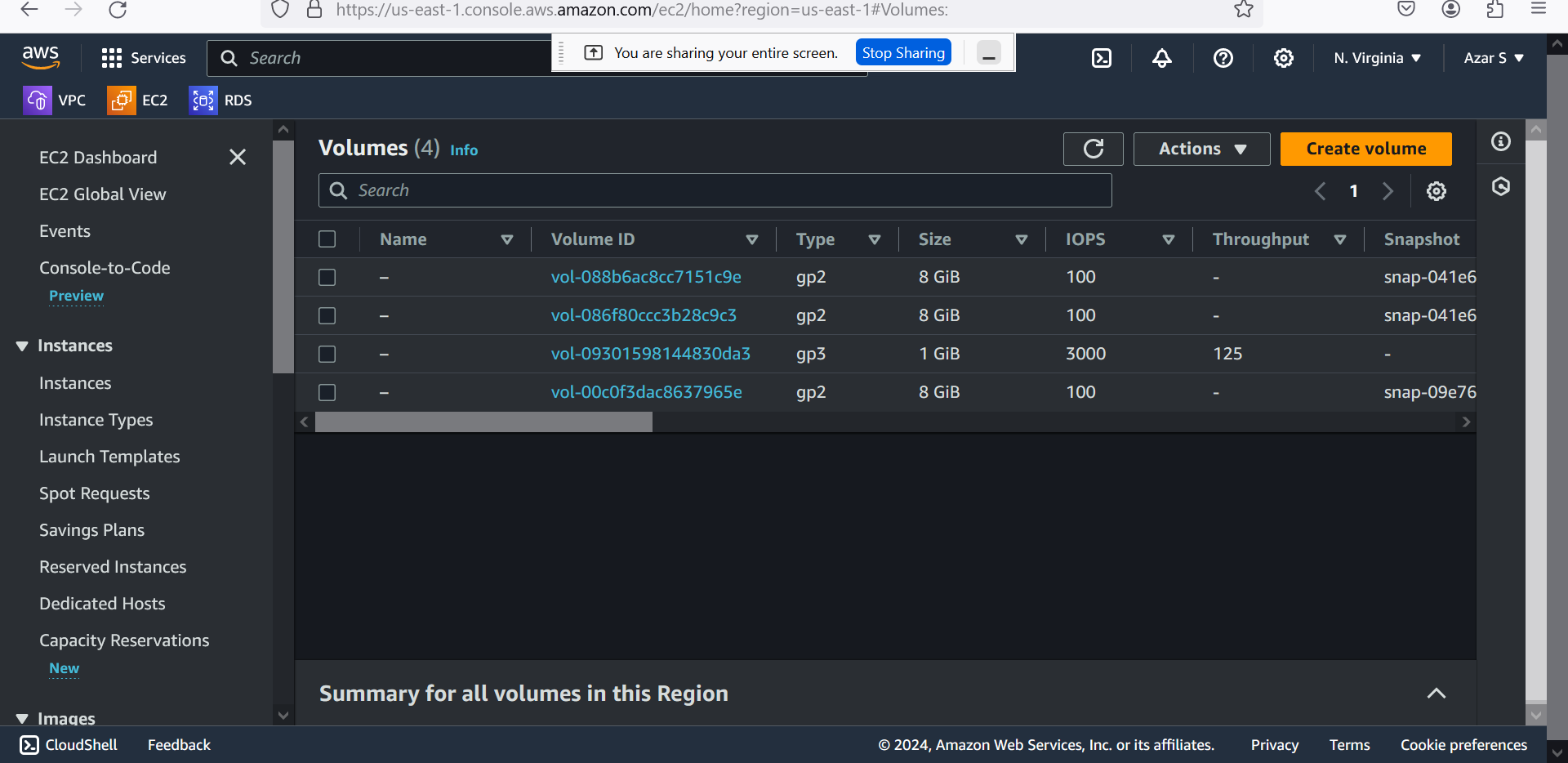
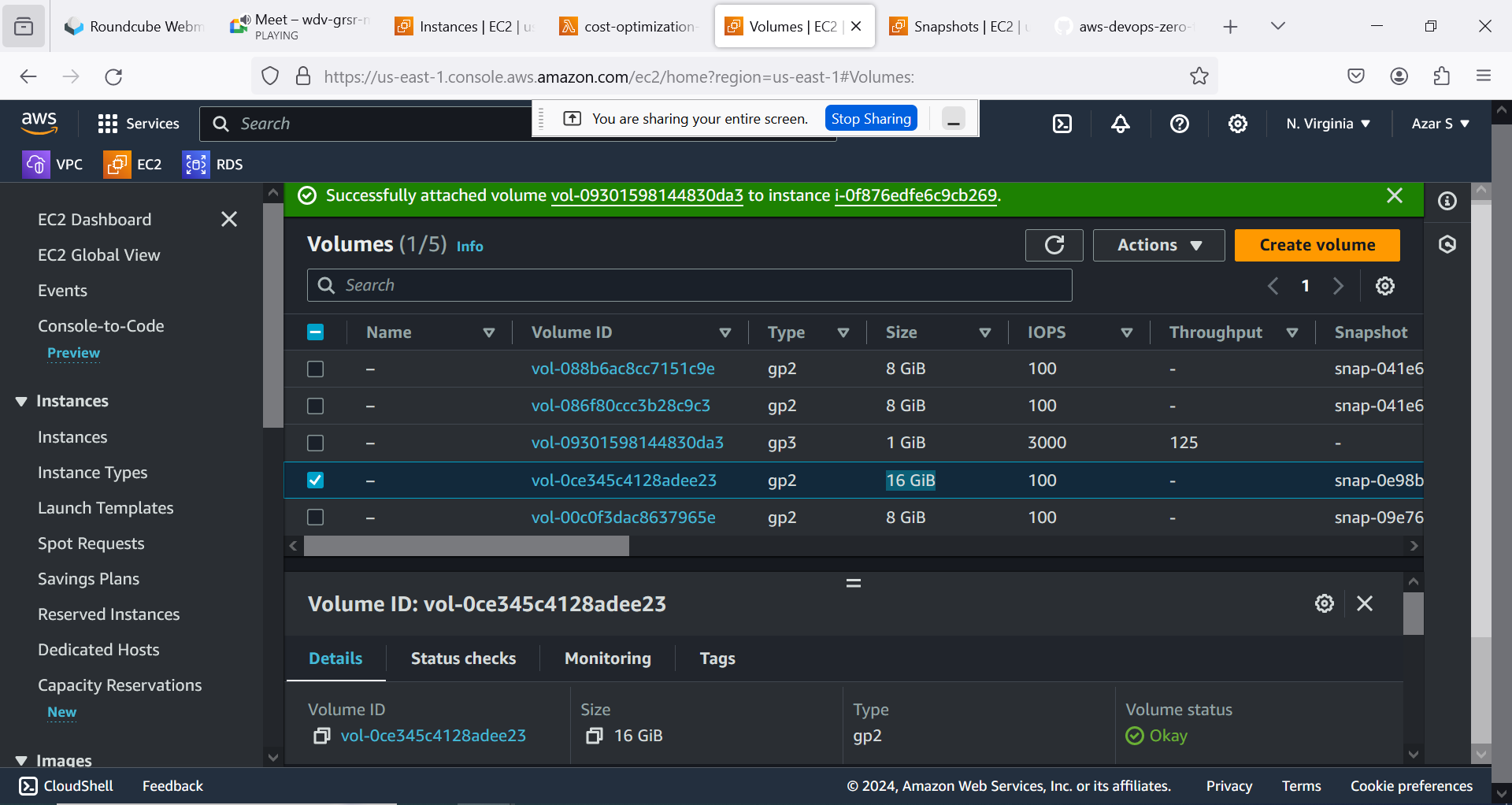
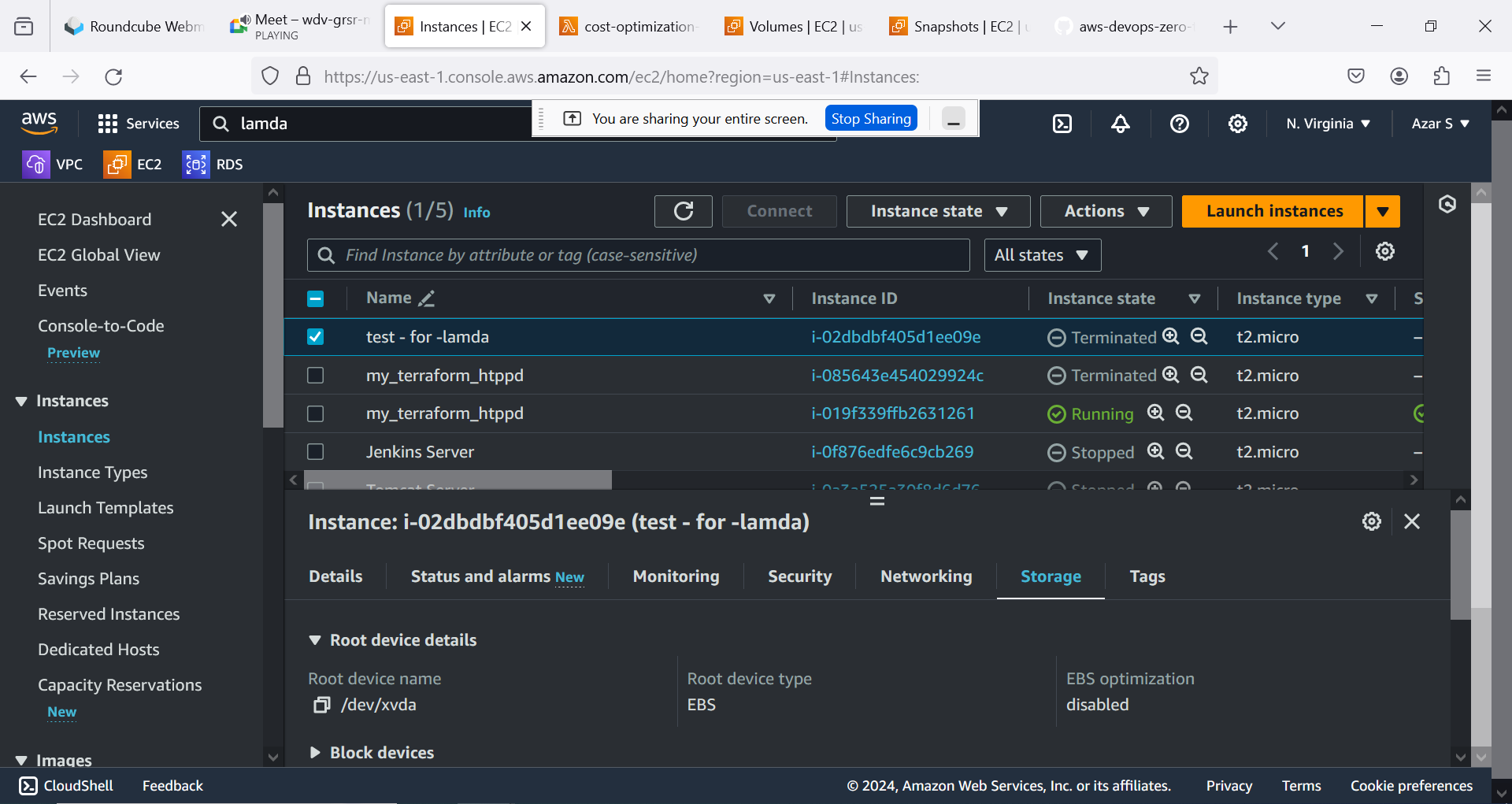
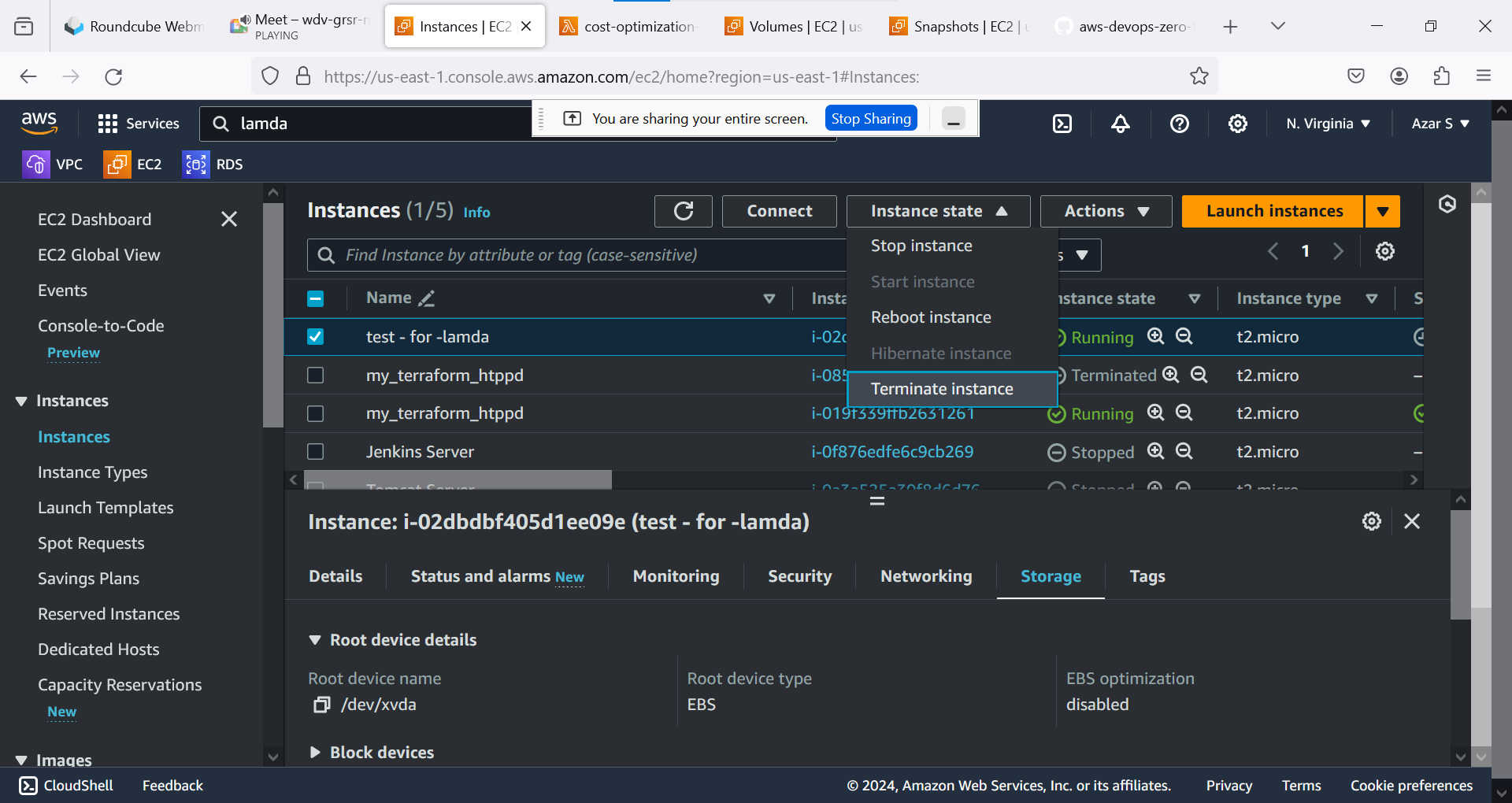
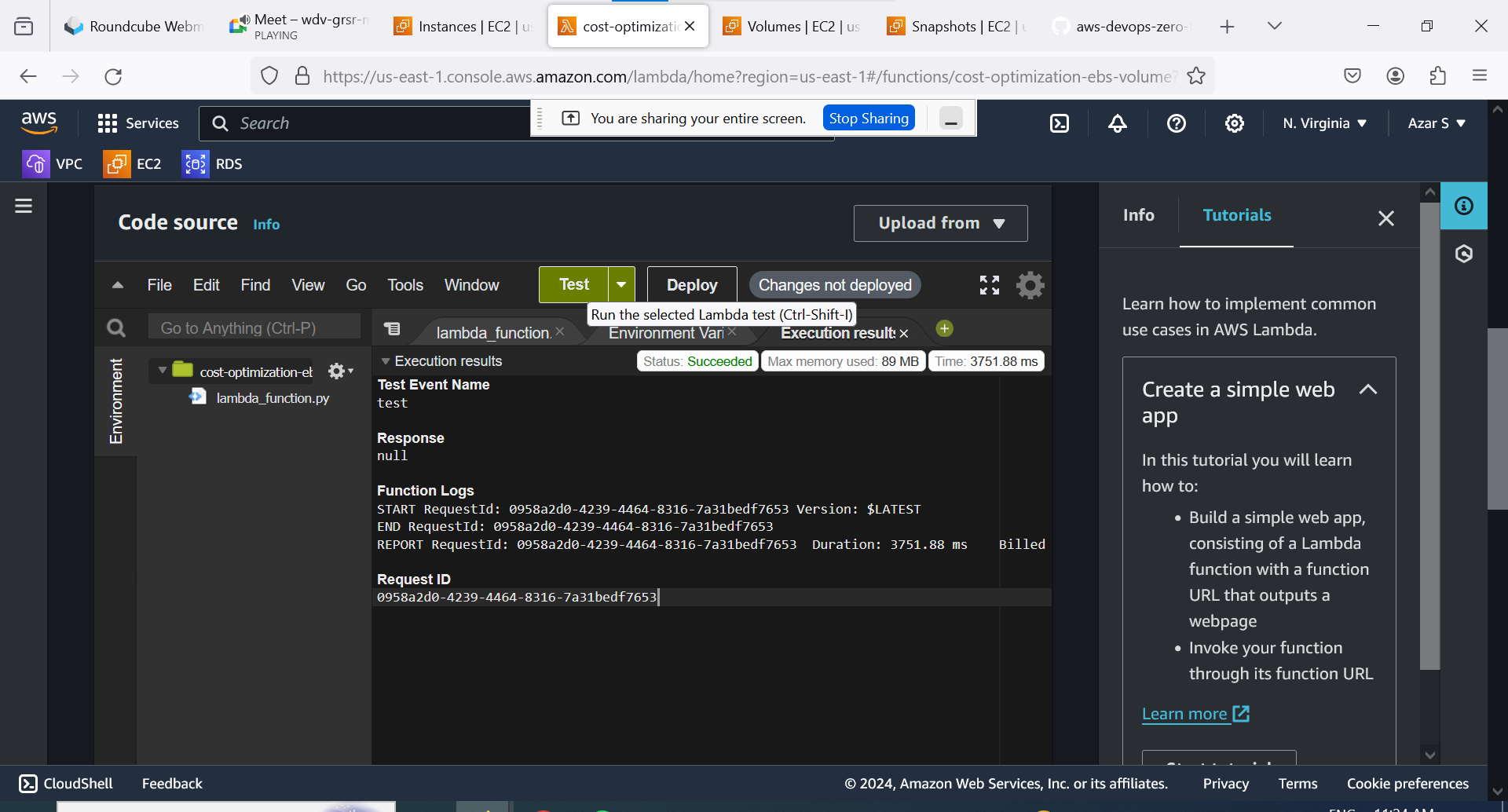
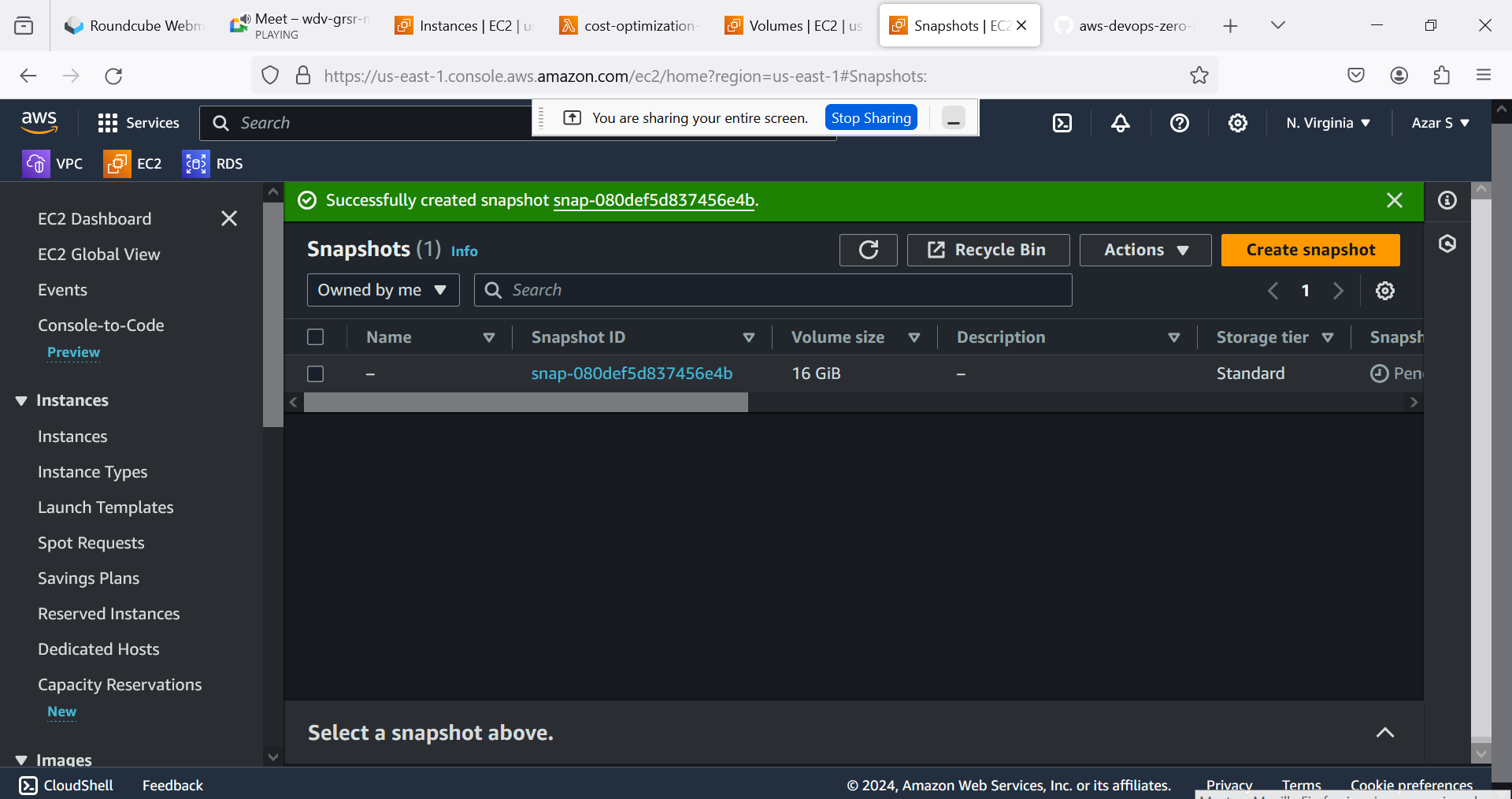
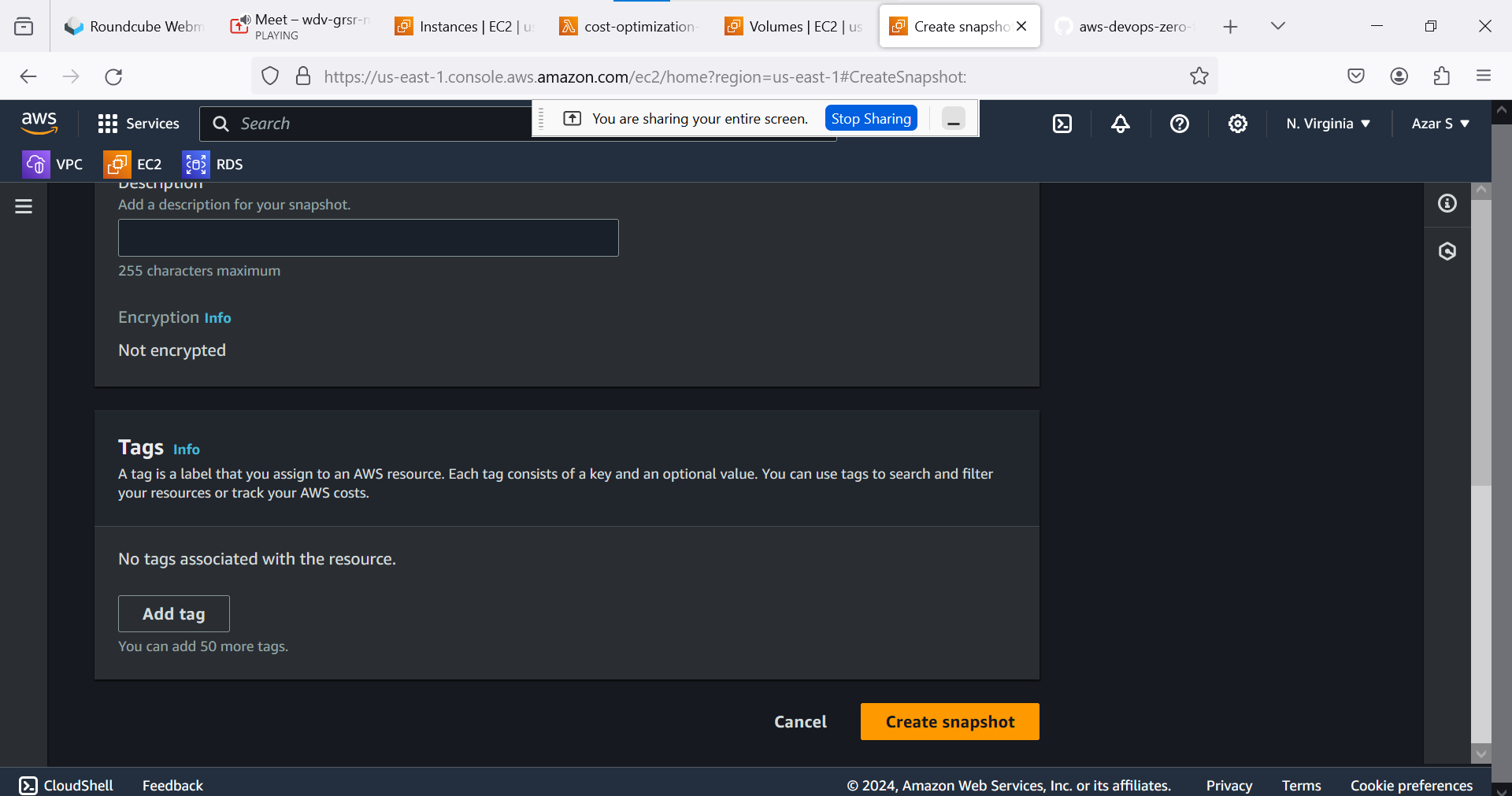
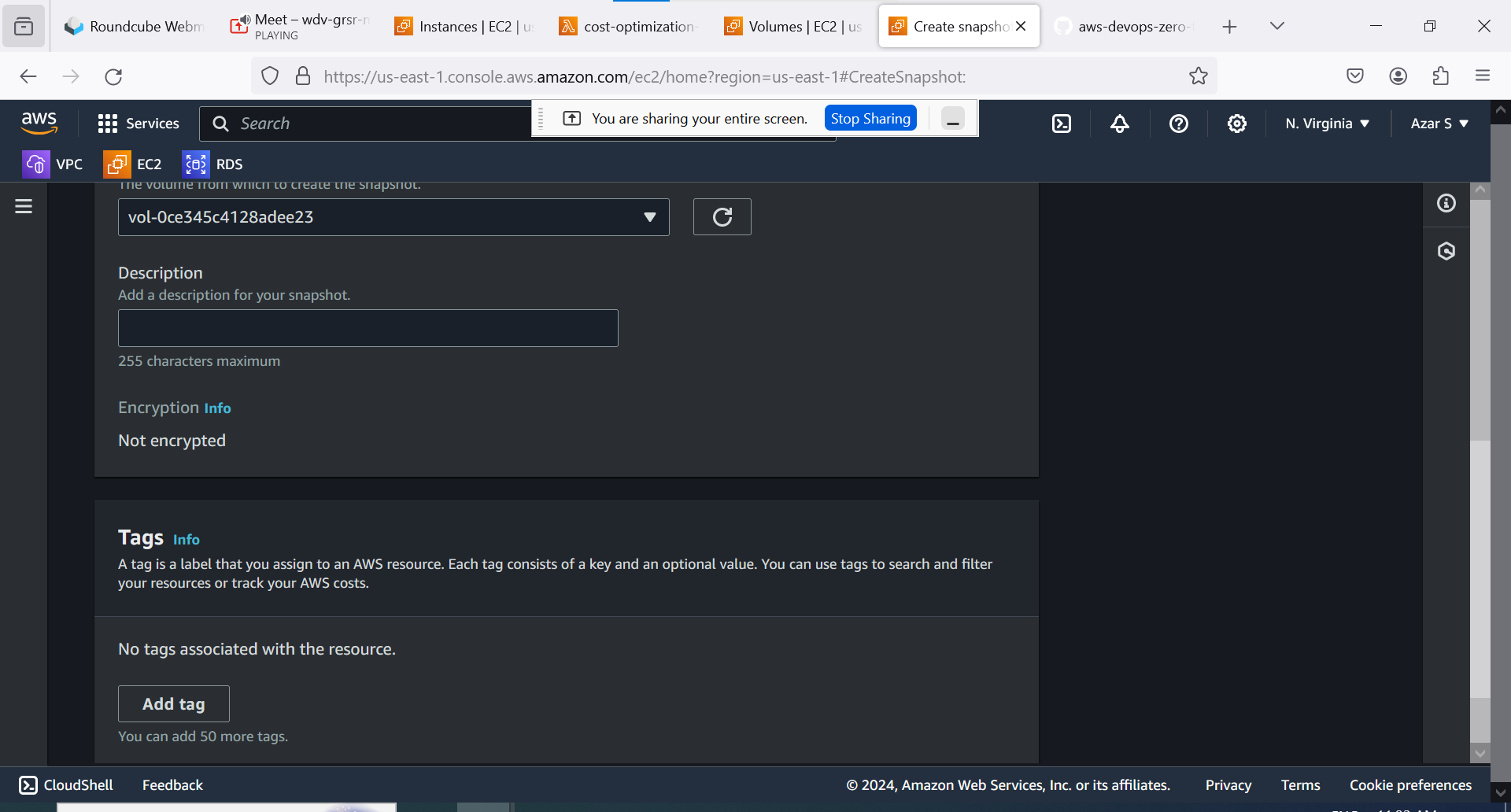
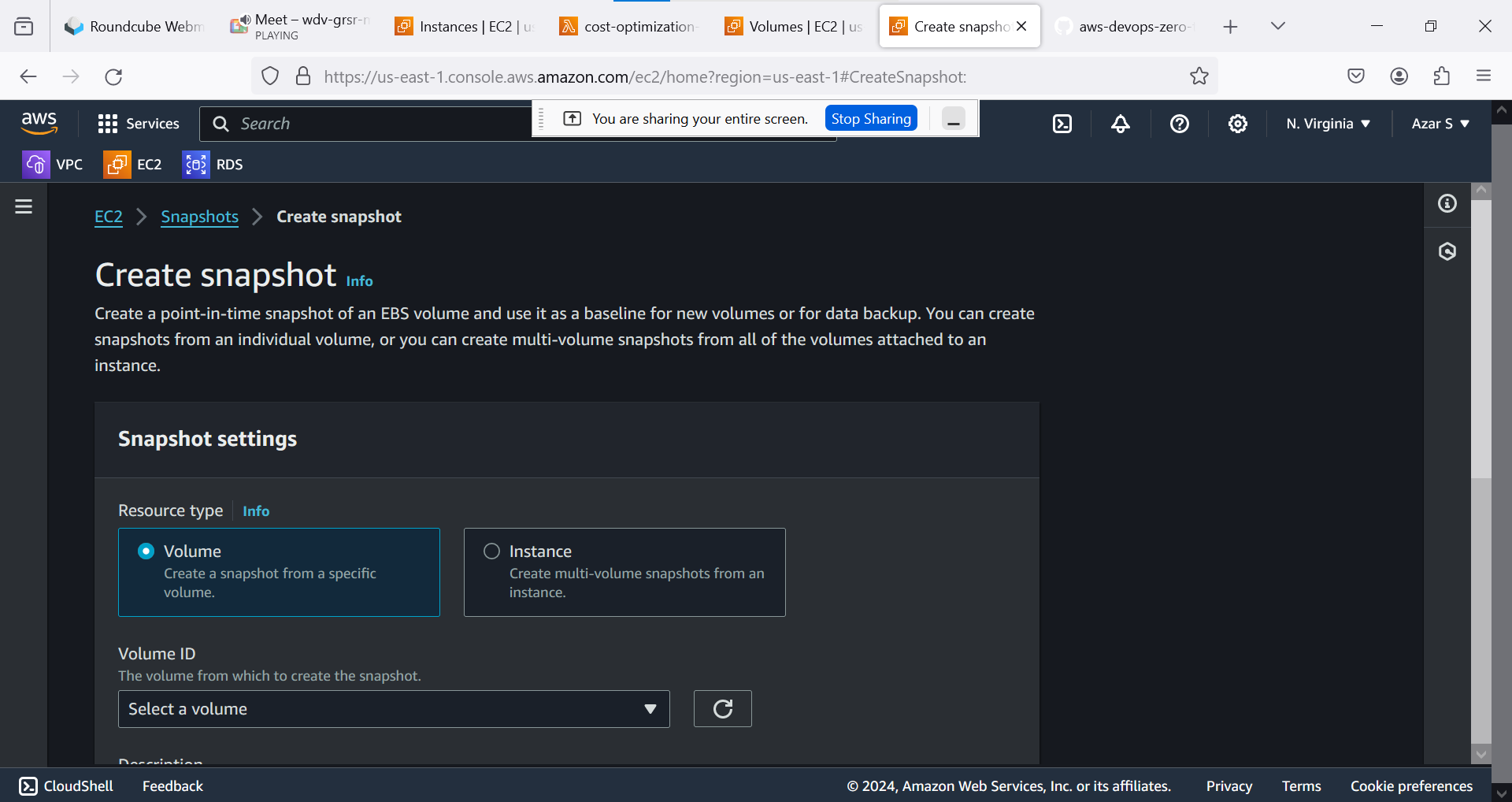
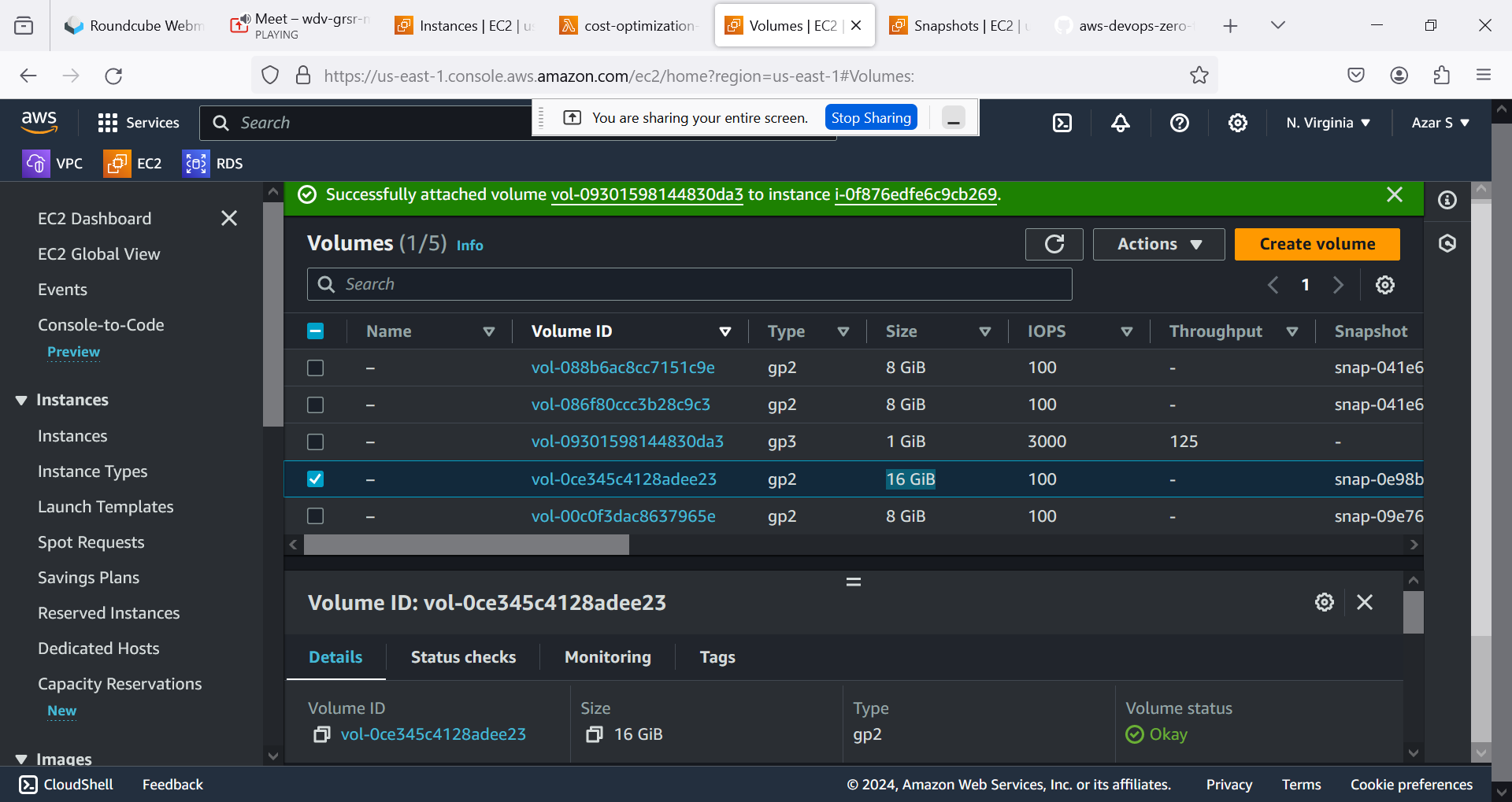
**ATTACH ROLES TO LAMBDA:**

**CREATE CUSTOMIZE POLICY:**

1. **Delete-Snapshot**
2. **Describe-snapshot**
3. **Describe-Volume**
4. **Describe-instance**
5. **Instead 3 &4 directly attach ec2 full access role**

****





GITHUB LINK: https://github.com/iam-veeramalla/aws-devops-zero-to-hero/tree/main/day-18

**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')

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.")