SCENARIO:-

Lab Prerequisites:
To successfully complete this lab, you should be familiar with basic Amazon EC2 usage and with basic Linux server administration. You should feel comfortable using the Linux command-line tools.

Background:
This experiment focuses on Amazon Elastic Block Store (Amazon EBS), a key underlying storage mechanism for Amazon EC2 instances. In this lab, you will learn how to create an Amazon EBS volume, attach it to an instance, apply a file system to the volume, and then take a snapshot backup.

Problem Scenario:
Steven is a Cloud Administrator of GotArray Techno Pvt Ltd, Their majority of clients are e-commerce and OTP service providers. Initially, they wanted to set up one virtual windows/Linux server using Amazon EC2 which can be resizable and provide compute capacity along with a web-scale cloud computing solution. After 6-month Steven and his team realized that they needed more cloud storage with an elastic, high performance block storage service. Design for EC2 machine due you to increase web traffic on their e-commerce client. Set the storage of 60 GB and attach them to the EC2 server and maintain all operations using logical volume management.

AIM:-

Create Terraform Configuration that can create EC2 instances, EBS volume (Customized Size) and attach it together.

Steps:-

1. **Configuring terraform script.**
   - **CODE:-** **(main.tf)**

```
terraform {
 required_providers {
   aws = {
     source  = "hashicorp/aws"
     version = "~> 3.0"
   }
 }
 required_version = ">= 0.13.5"
}
# Configure the AWS Provider
provider "aws" {
 region     = "us-east-1"
 access_key = "YOUR_ACCESS_KEY"
 secret_key = "YOUR_SECRET_KEY"
}

# Create a Instance
resource "aws_instance" "vm" {
 ami           = "ami-0cff7528ff583bf9a"
 instance_type = "t2.micro"
 key_name      = "MyKey"
```

```
 tags = {
   Name = "AMI Linux 2 ~ EBS storage"
 }
}

resource "aws_ebs_volume" "ebs_volume" {
 availability_zone = aws_instance.vm.availability_zone
 size              = 10
 type              = "gp2"
 tags = {
   Name = "ebs-volume-e4"
 }
}

resource "aws_volume_attachment" "ebs_volume_attachment" {
 device_name = "/dev/xvdh"
 volume_id   = aws_ebs_volume.ebs_volume.id
 instance_id = aws_instance.vm.id
}
output "EbsVolumeId" {
 value = aws_ebs_volume.ebs_volume.id
}
output "InstanceId" {
 value = aws_instance.vm.id
}
```

2. Launch terraform script.

- terraform init

```
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E3$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.0"...
- Installing hashicorp/aws v3.75.2...
- Installed hashicorp/aws v3.75.2 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E3$
```

- terraform validate

```
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E3$ terraform validate
Success! The configuration is valid.

satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E3$
```

- terraform plan

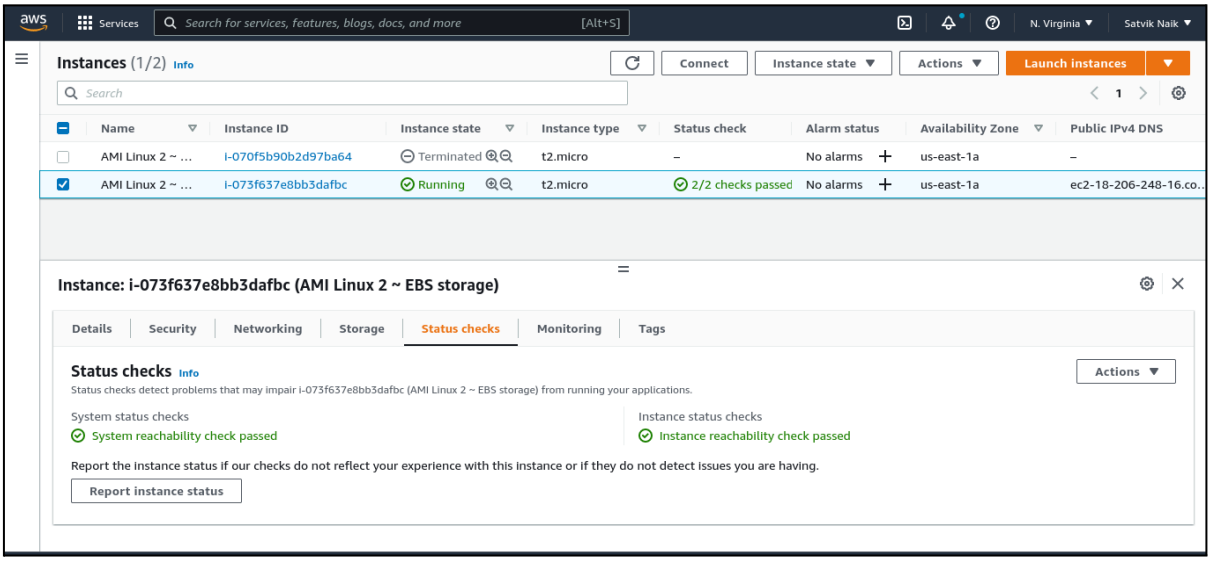
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER   GITLENS                                   bash - E3

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

  # aws_volume_attachment.ebs_volume_attachment will be created
  + resource "aws_volume_attachment" "ebs_volume_attachment" {
      + device_name = "/dev/xvdh"
      + id          = (known after apply)
      + instance_id = (known after apply)
      + volume_id   = (known after apply)
    }

Plan: 3 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run
"terraform apply" now.
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E3$
```

- terraform apply

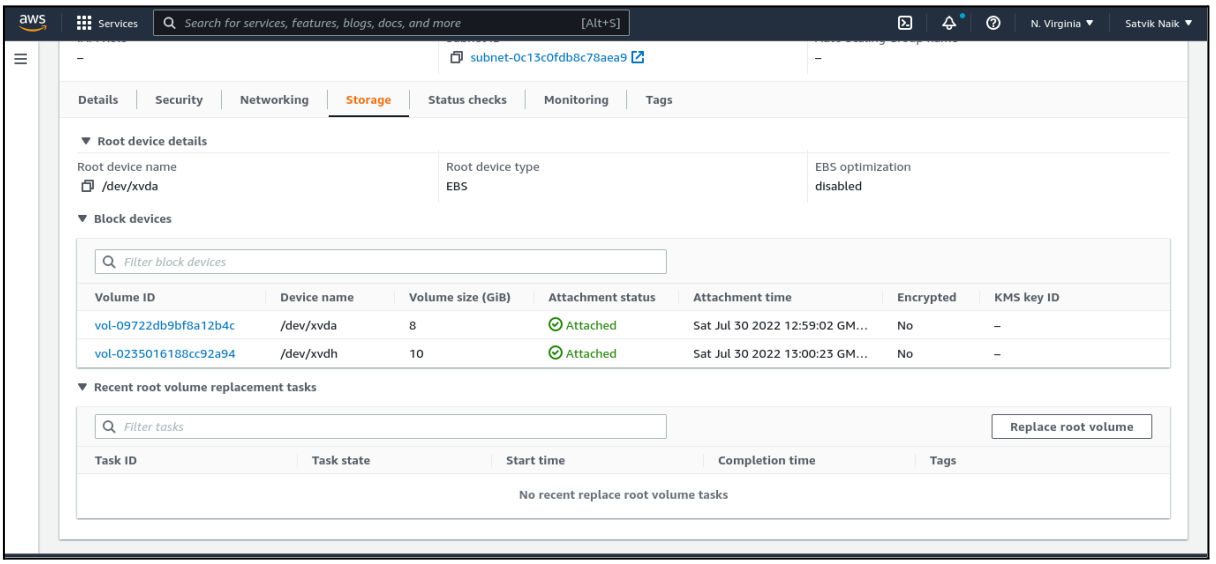
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER   GITLENS                                   bash - E3

Plan: 3 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + EbsVolumeId = (known after apply)
  + InstanceId  = (known after apply)
aws_instance.vm: Creating...
aws_instance.vm: Still creating... [10s elapsed]
aws_instance.vm: Still creating... [20s elapsed]
aws_instance.vm: Still creating... [30s elapsed]
aws_instance.vm: Still creating... [40s elapsed]
aws_instance.vm: Still creating... [50s elapsed]
aws_instance.vm: Creation complete after 56s [id=i-073f637e8bb3dafbc]
aws_ebs_volume.ebs_volume: Creating...
aws_ebs_volume.ebs_volume: Still creating... [10s elapsed]
aws_ebs_volume.ebs_volume: Creation complete after 13s [id=vol-0235016188cc92a94]
aws_volume_attachment.ebs_volume_attachment: Creating...
aws_volume_attachment.ebs_volume_attachment: Still creating... [10s elapsed]
aws_volume_attachment.ebs_volume_attachment: Still creating... [20s elapsed]
aws_volume_attachment.ebs_volume_attachment: Still creating... [30s elapsed]
aws_volume_attachment.ebs_volume_attachment: Creation complete after 37s [id=vai-2663629394]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

EbsVolumeId = "vol-0235016188cc92a94"
InstanceId = "i-073f637e8bb3dafbc"
satvik--naik@pop-os:/media/satvik--naik/MyDesk/CLG/SEM7/CCE/cloud-computing-essentials/E3$
```

3. Verifying resources created on AWS Cloud.

- Instance created successfully.

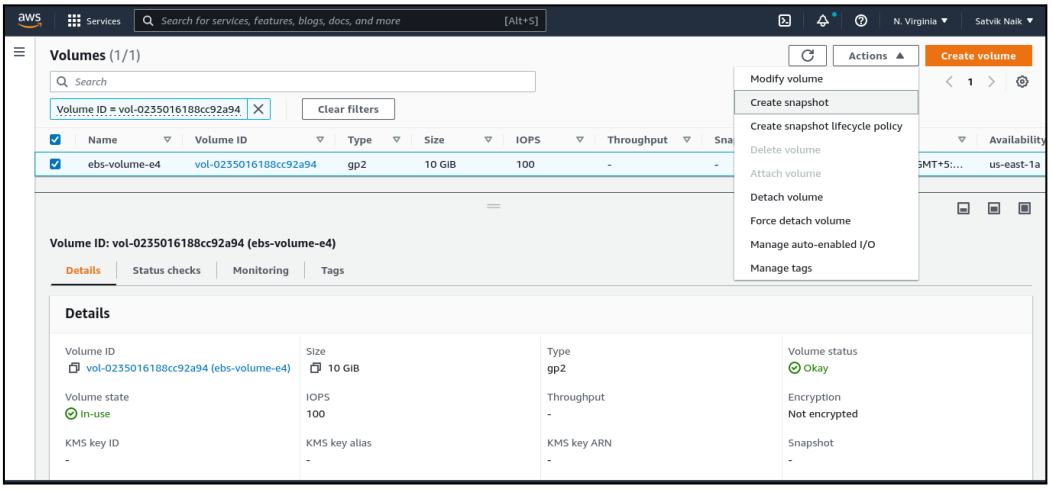


- Additional ***EBS volume*** attached of size ***10GB***.

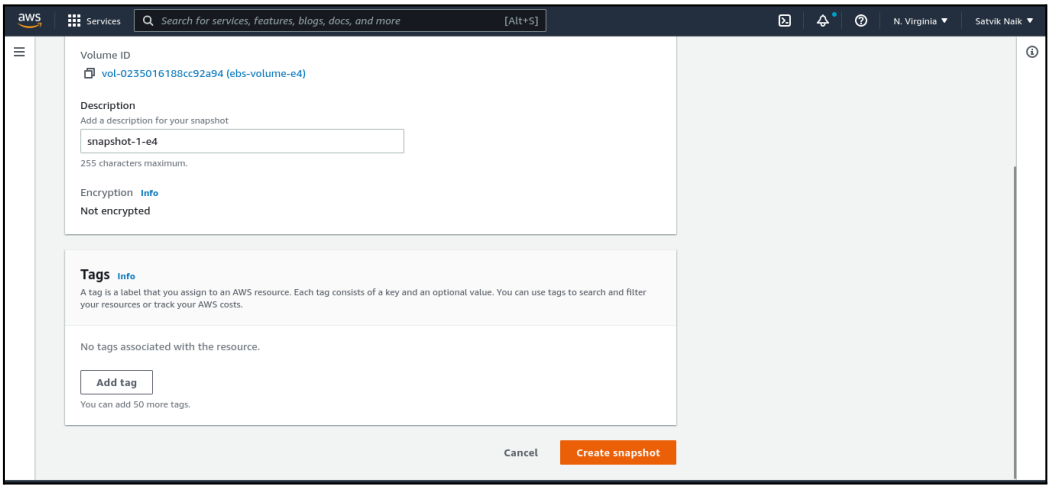


4. Create a snapshot of your volume.

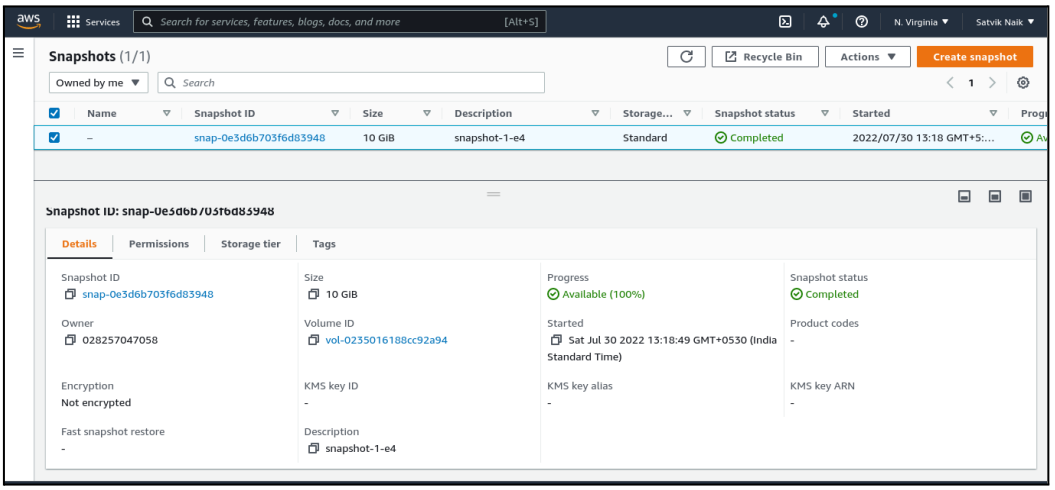
- Select the volume and go to ***Actions > Create snapshot***.



- Provide description & click on ***'Create snapshot'***.

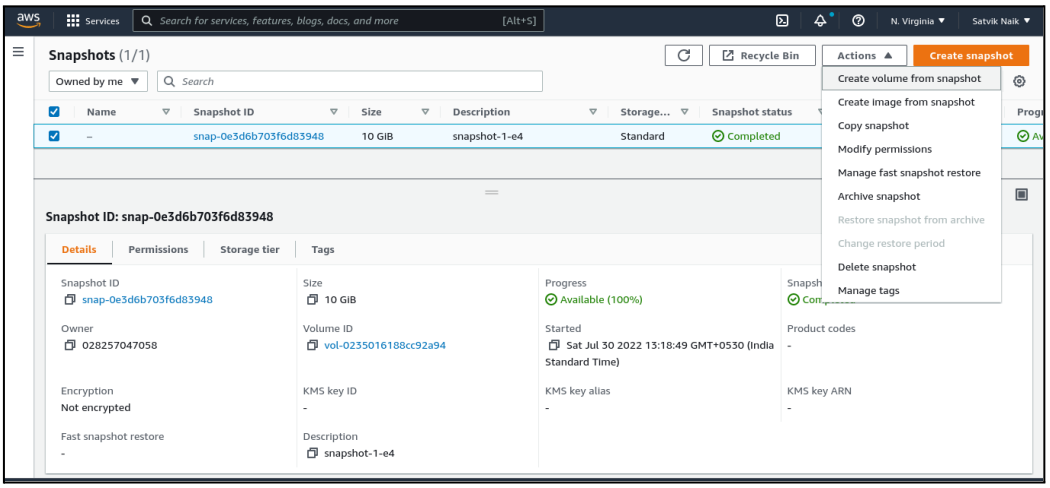


- Snapshot created successfully.

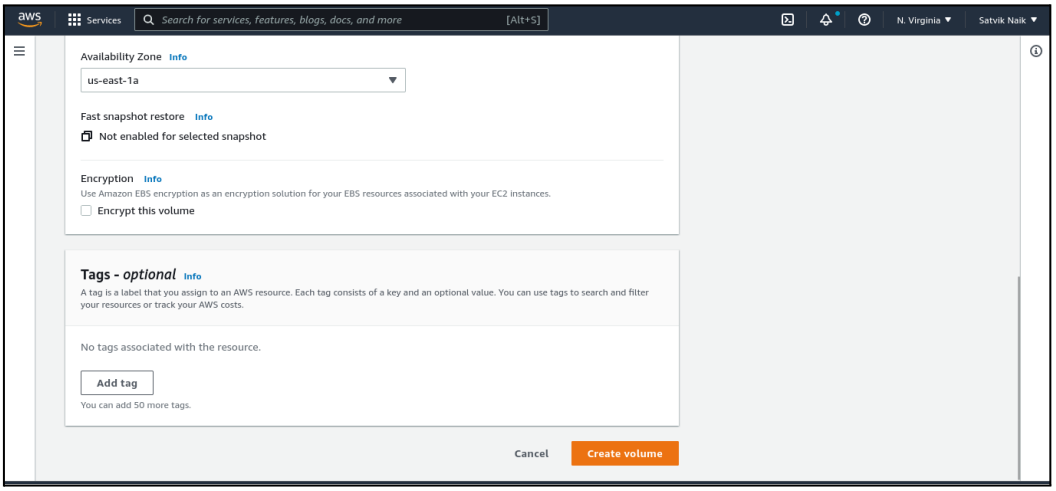


5. Create a new volume from your snapshot.

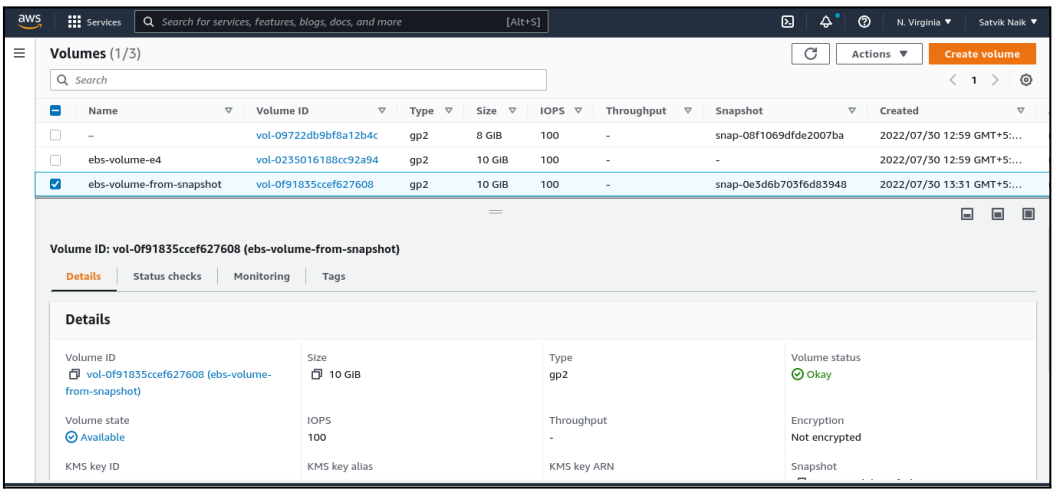
- Select the snapshot and go to ***Actions > Create volume from snapshot***.



- Verify details & click on ***'Create volume'***.

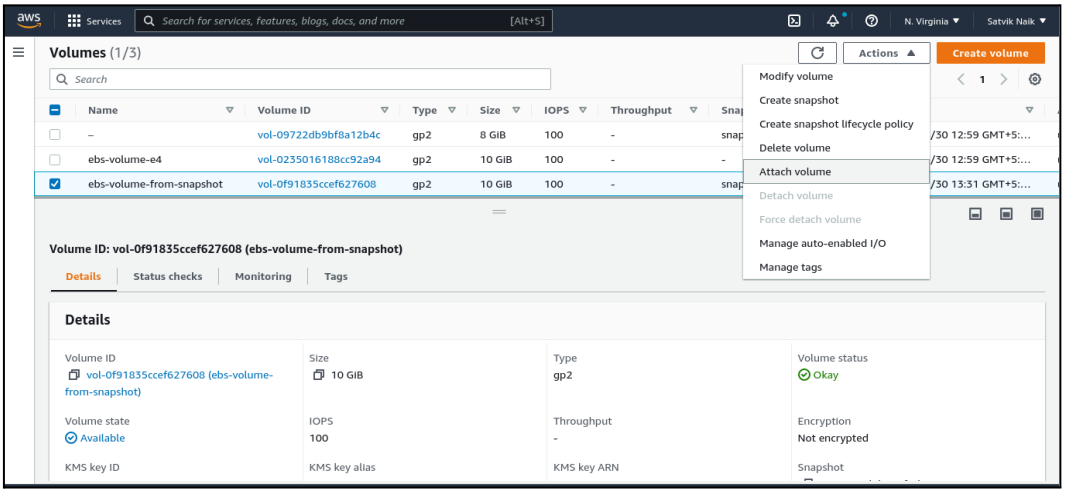


- Volume from snapshot created successfully.

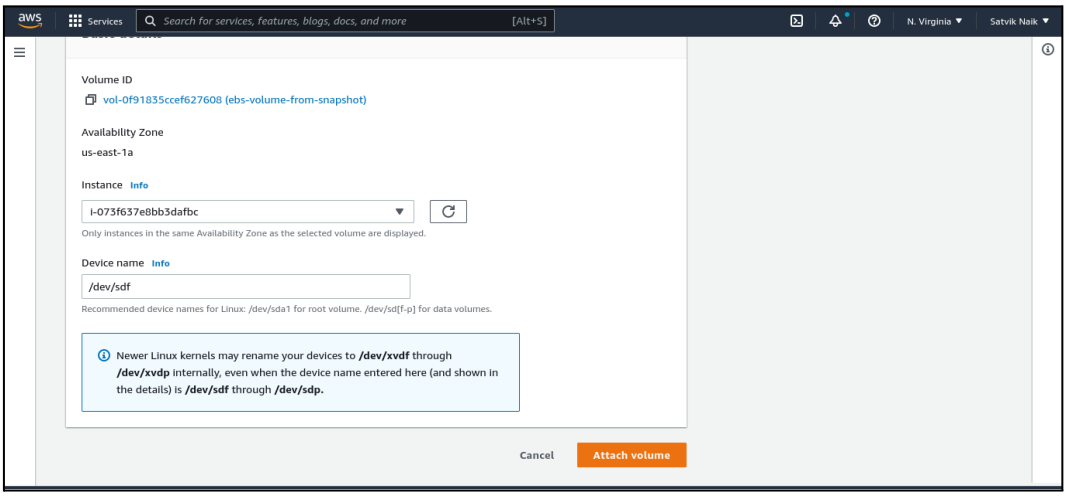


6. Attach and mount your new volume to your EC2 instance.

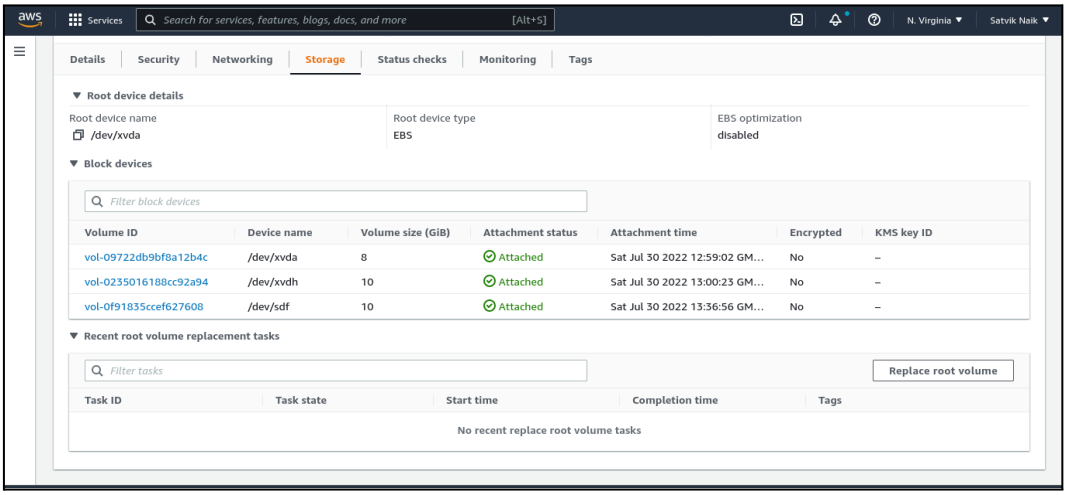
- Select the volume & go to ***Actions > Attach volume***.

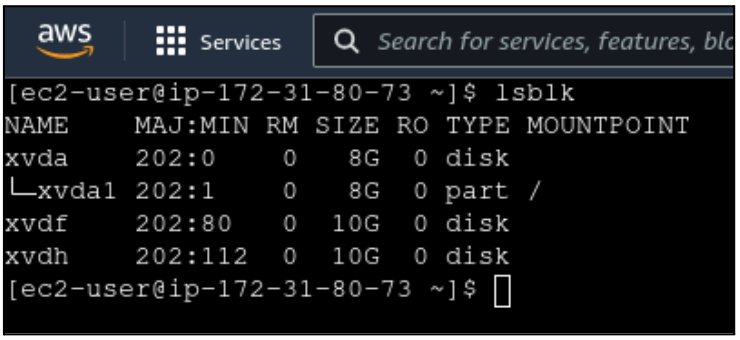


- Select the instance & click on ***'Attach Volume'***.



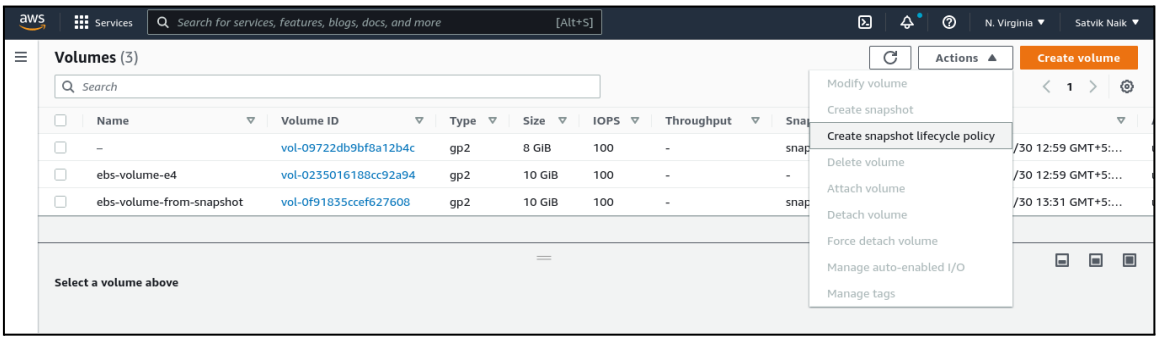
- New volume which was created from a snapshot attached successfully.



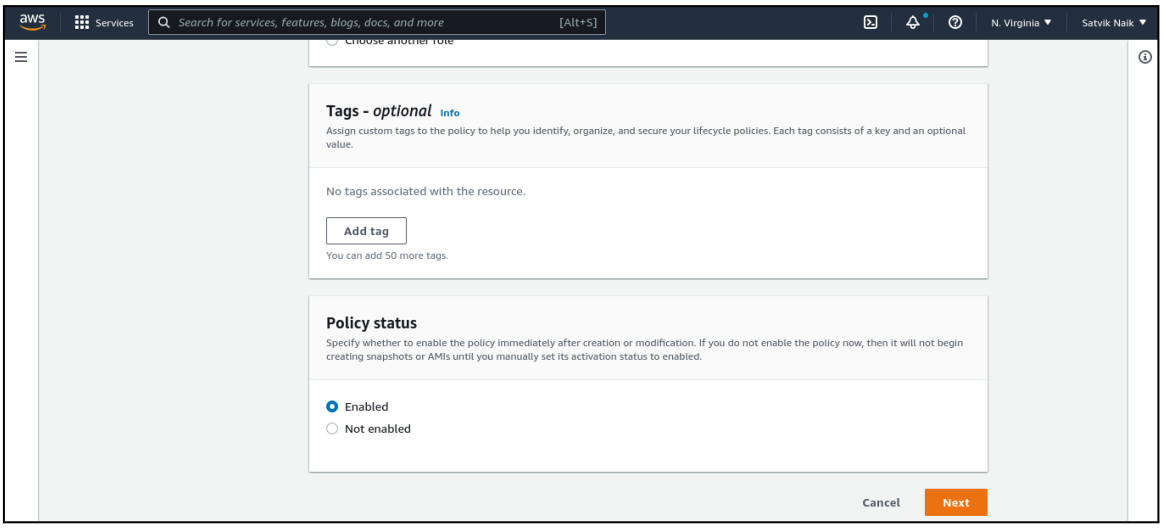


7. Schedule the snapshot.

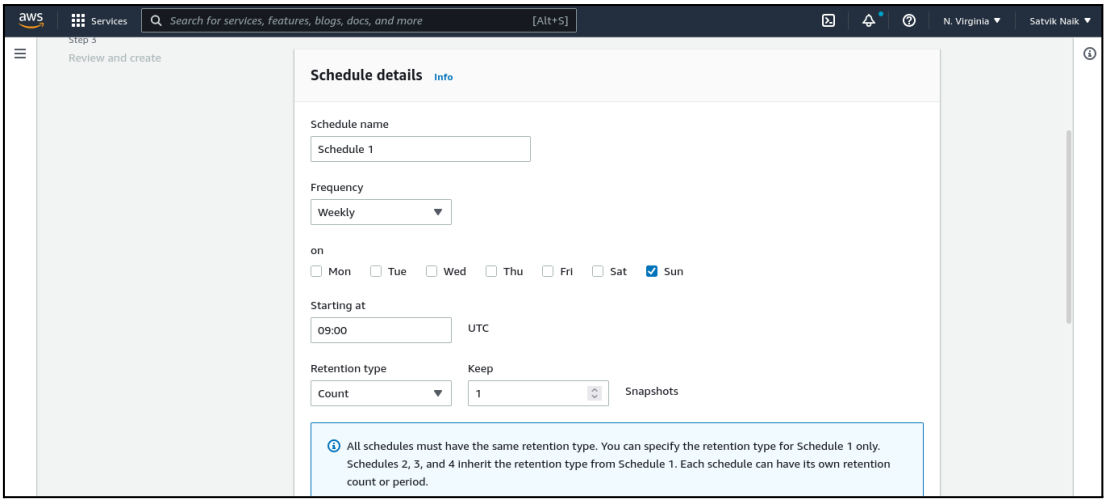
- Under the ***Volumes*** section, go to ***Actions > Create snapshot lifecycle policy***.



- Target resource types: ***Volumes***
- Target resource tags: ***ebs-volume-e4***
- Policy description:- ***Scheduling of ebs-volume-e4***

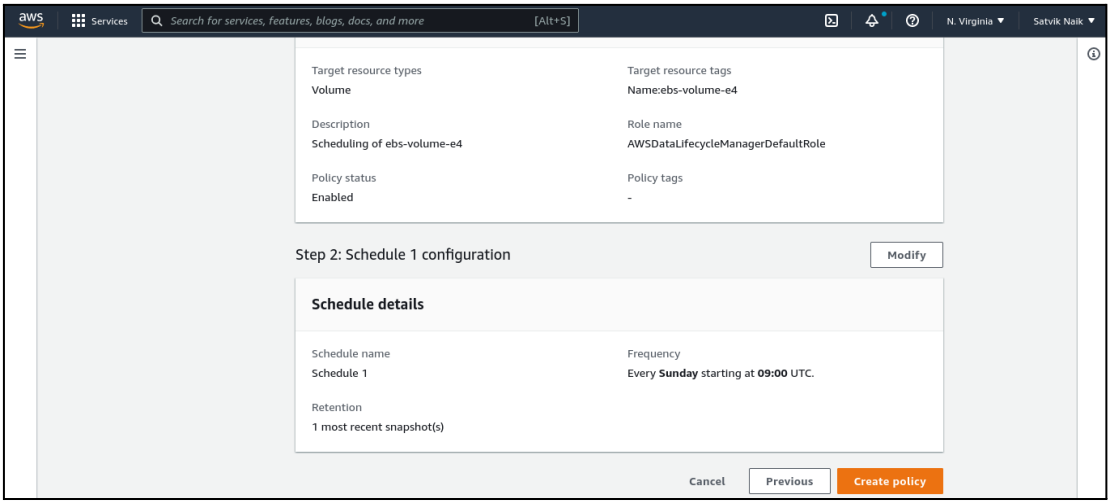


- Schedule details.



Click on ***'Review Policy'***.

- Review & Create Policy.



- Snapshot successfully scheduled.