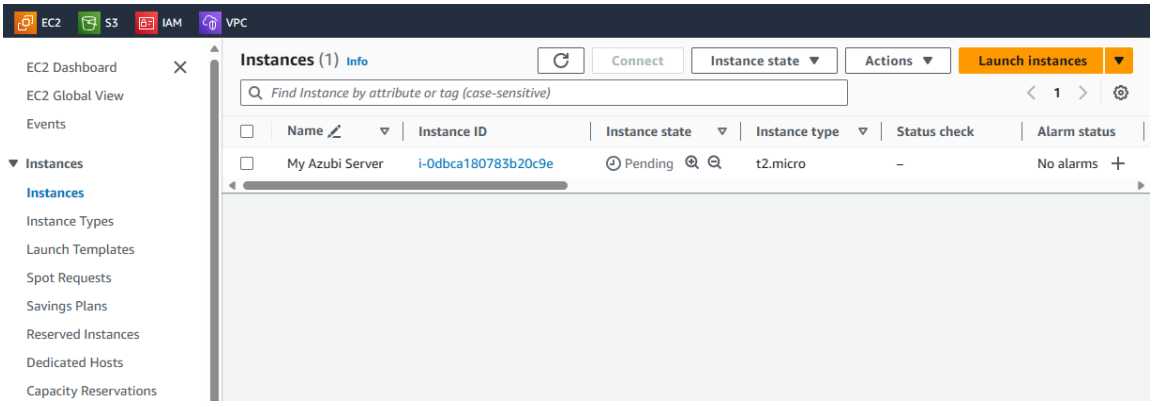
AWS Storage Project

Author	Olatunji Olayinka Oluwaseun
LinkedIn	https://www.linkedin.com/in/olatunji-olayinka-coder/
GitHub	https://github.com/olatunji-weber
Status	Done

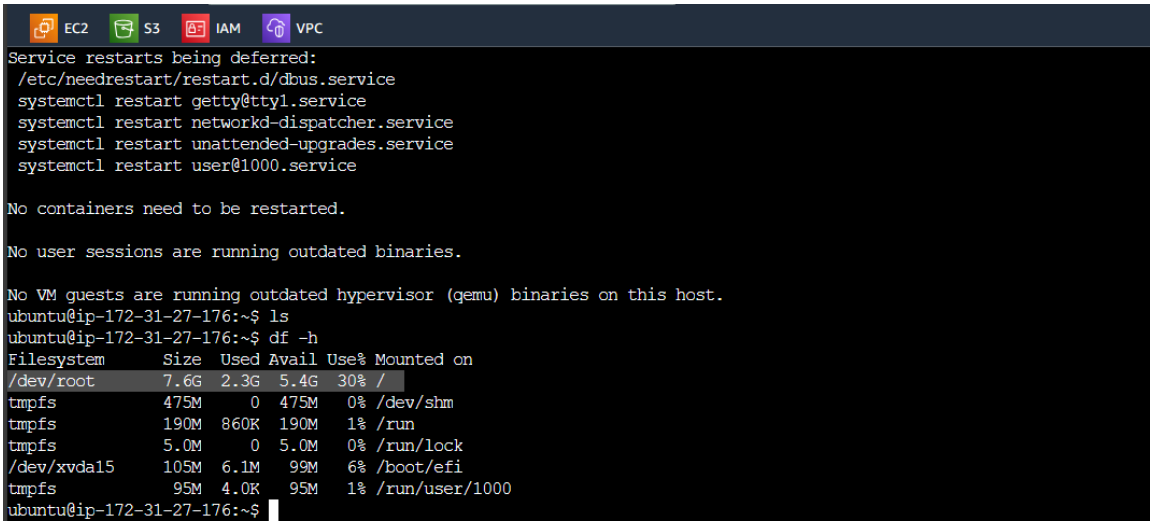
This project involves setting up Amazon Elastic Block Store (EBS) and Amazon Elastic File System (EFS) in AWS. Below are the key steps and findings:

1. **Amazon EBS Setup:**

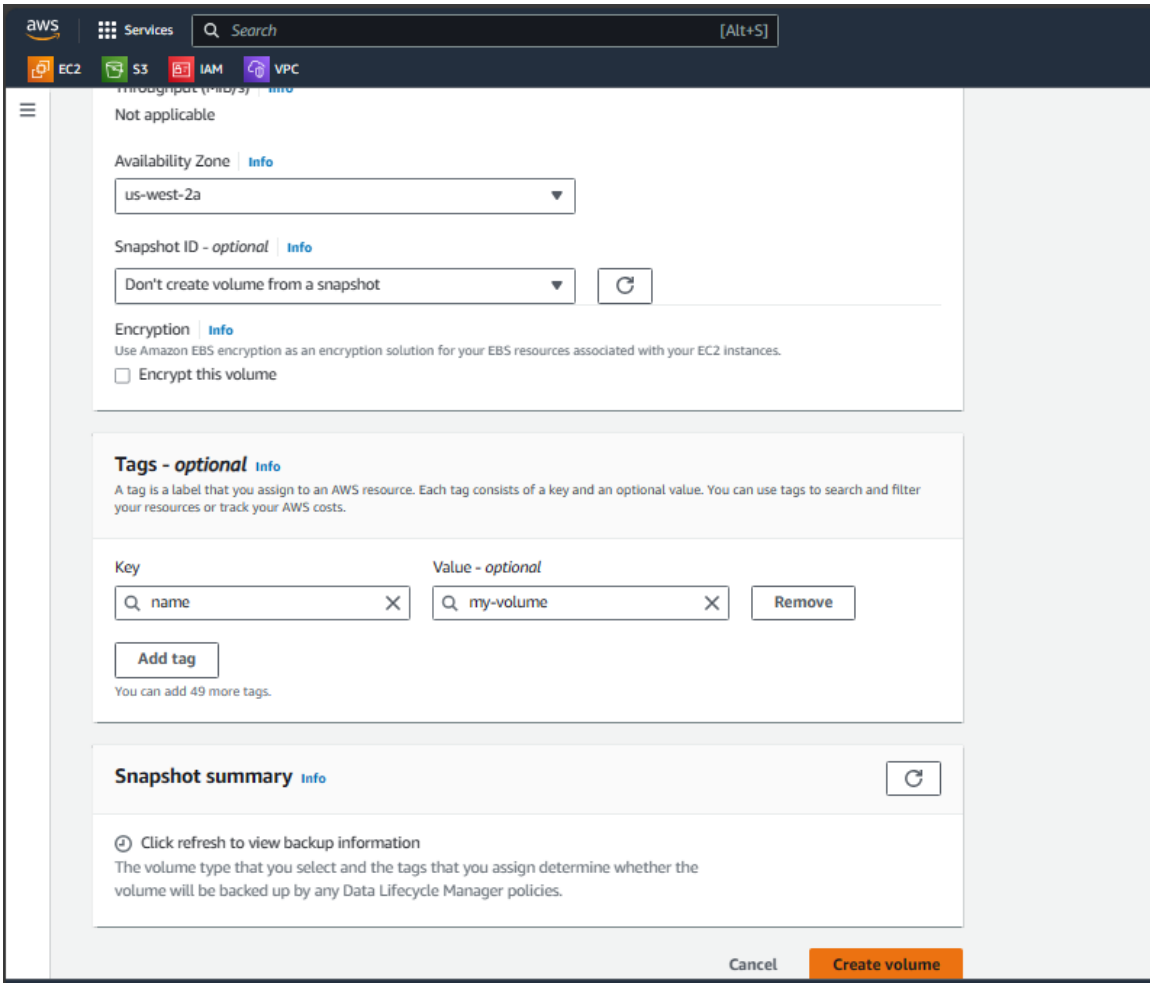
   - Create an EC2 Instance

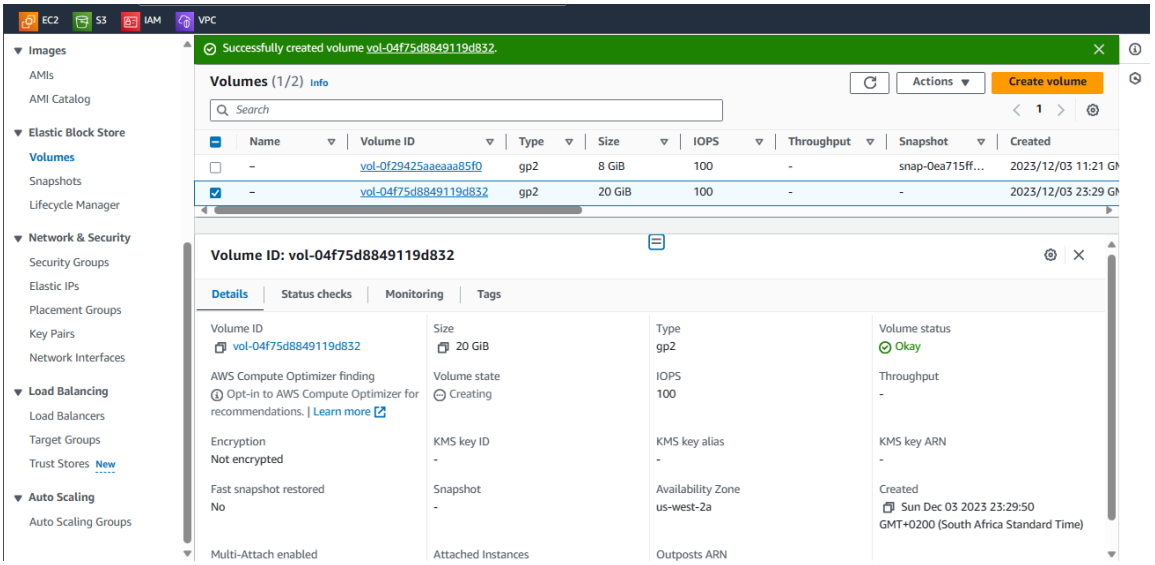


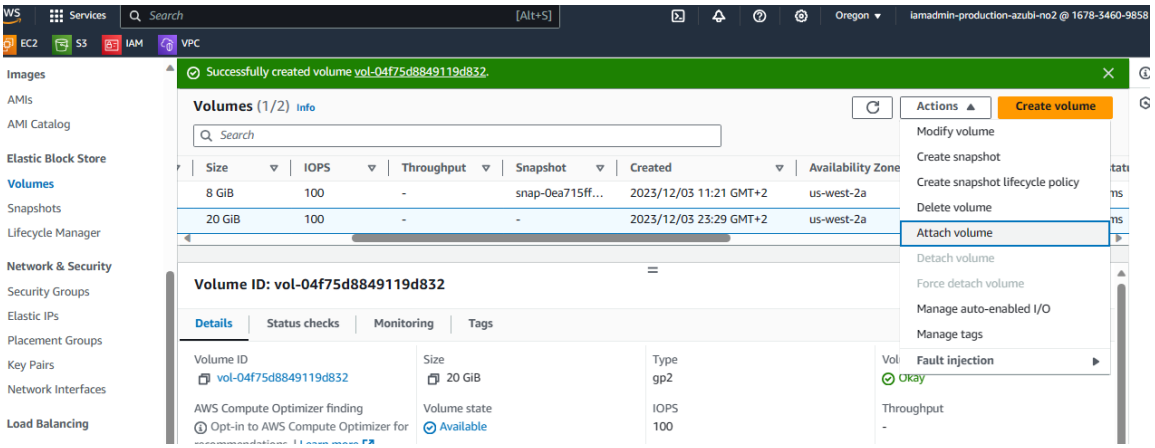
   - The command “df -h” (Will list all the volumes present and attached to the ec2 instance)



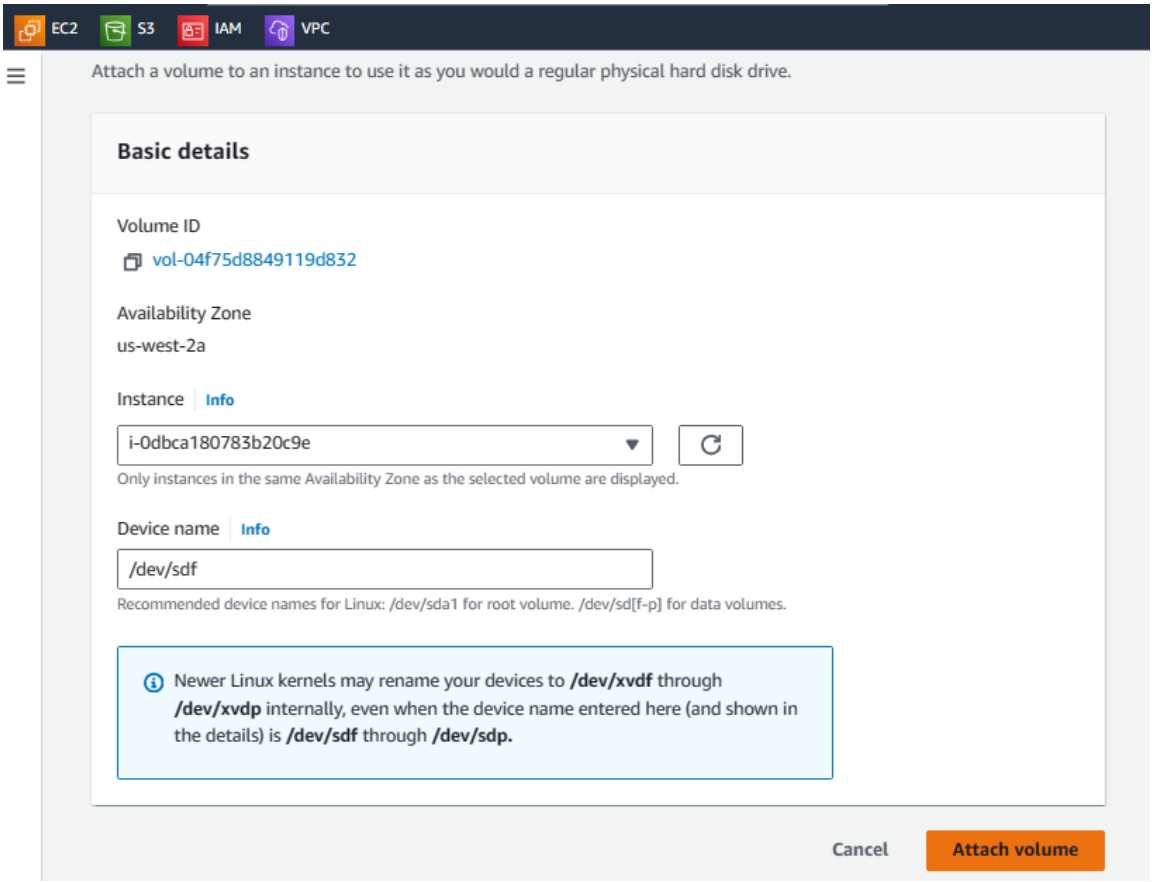
- Create an Amazon EBS volume and attach it to an EC2 instance.



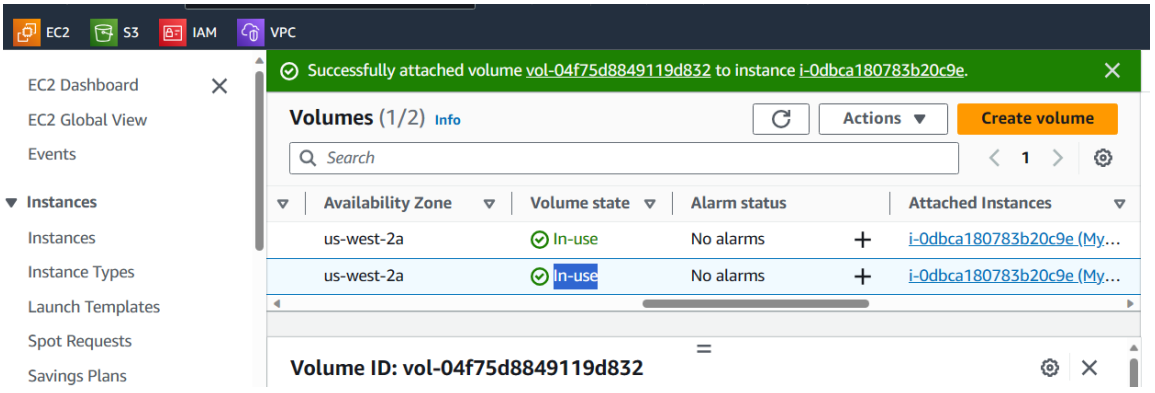




Next, you must attach the EC2 instance to the EBS volume that you created.



Then you will see that the “Volume state” is now: “In-use”



- Run “lsblk” to List all the Block devices on the Linux machine

```
EC2   S3   IAM   VPC
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
ubuntu@ip-172-31-27-176:~$ lsblk
NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0       7:0    0  24.6M  1 loop /snap/amazon-ssm-agent/7528
loop1       7:1    0  55.7M  1 loop /snap/core18/2790
loop2       7:2    0  63.5M  1 loop /snap/core20/2015
loop3       7:3    0 111.9M  1 loop /snap/lxd/24322
loop4       7:4    0  40.8M  1 loop /snap/snapd/20092
loop5       7:5    0  40.9M  1 loop /snap/snapd/20290
loop6       7:6    0  55.7M  1 loop /snap/core18/2796
loop7       7:7    0  24.9M  1 loop /snap/amazon-ssm-agent/7628
xvda      202:0    0     8G  0 disk
├─xvda1   202:1    0   7.9G  0 part /
├─xvda14  202:14   0     4M  0 part
└─xvda15  202:15   0   106M  0 part /boot/efi
xvdf      202:80   0    20G  0 disk
ubuntu@ip-172-31-27-176:~$
```

- We can check of there is any file system on this new volume using “$sudo file -s /dev/xvdf”. If we see “data”, it means you need to setup file system for this block device. You need to have a file system in your volume, only then can it be mounted into your EC2 instance.

```
ubuntu@ip-172-31-27-176:~$ sudo file -s /dev/xvdf
/dev/xvdf: data
ubuntu@ip-172-31-27-176:~$
```

- You can run this command “mkfs -t xfs /dev/xvdf”

```
ubuntu@ip-172-31-27-176:~$ sudo mkfs -t xfs /dev/xvdf
meta-data=/dev/xvdf              isize=512    agcount=4, agsize=1310720 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=5242880, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
ubuntu@ip-172-31-27-176:~$
```

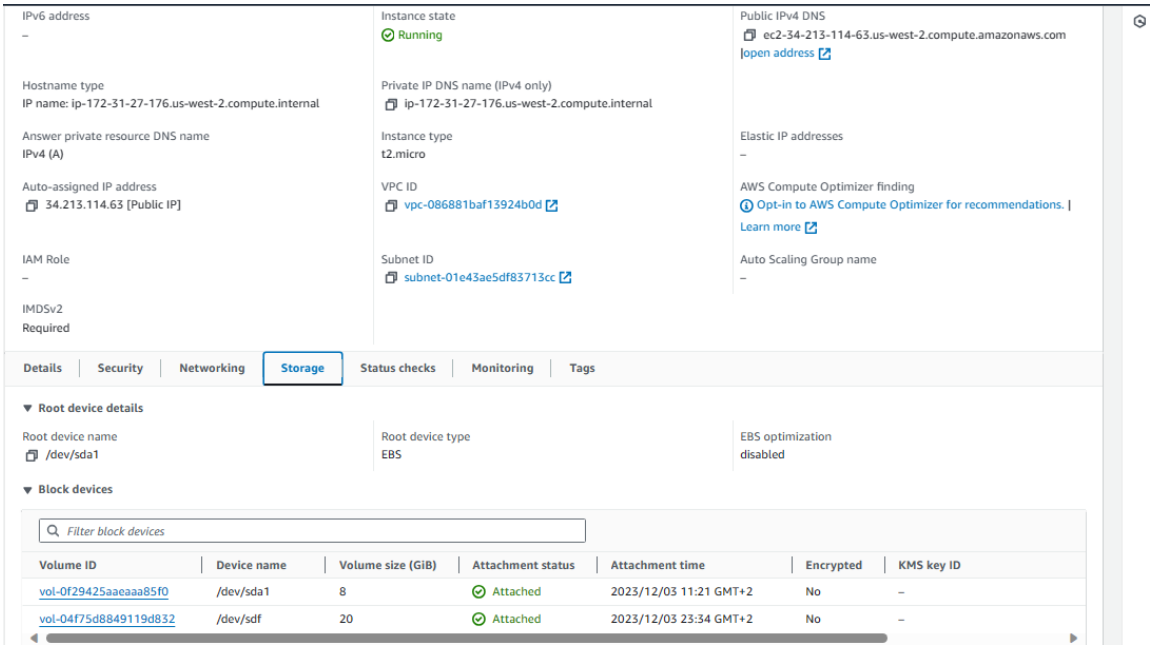
- And run the command “sudo file -s /dev/xvdf” to check the file system again and you will see that the SGI XFS file system is now present.

```
ubuntu@ip-172-31-27-176:~$ sudo file -s /dev/xvdf
/dev/xvdf: SGI XFS filesystem data (blksz 4096, inosz 512, v2 dirs)
ubuntu@ip-172-31-27-176:~$
```

- Now, we must mount to a directory in our EC2 instance, but first we must create a directory.
- Using the command “sudo mount /dev/xvdf apps/volume/new-volume” in order to format and mount the volume to a specific directory will help us mount the EBS volume to the EC2 instance without even restarting our EC2 instance.

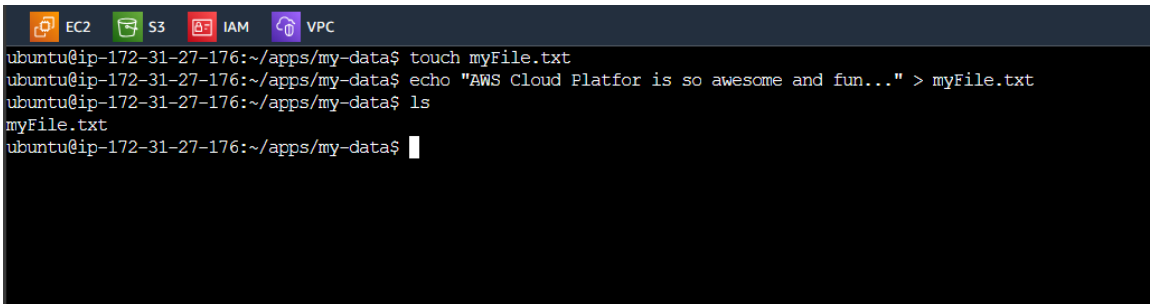
```
ubuntu@ip-172-31-27-176:~$ sudo mount /dev/xvdf apps/volume/new-volume
ubuntu@ip-172-31-27-176:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  2.3G  5.4G  30% /
tmpfs           475M     0  475M   0% /dev/shm
tmpfs           190M  864K  190M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
/dev/xvdf        20G  175M   20G   1% /home/ubuntu/apps/volume/new-volume
ubuntu@ip-172-31-27-176:~$
```

- And if you check the running EC2 instance, you will find the attached EBS block storage is reflecting now.

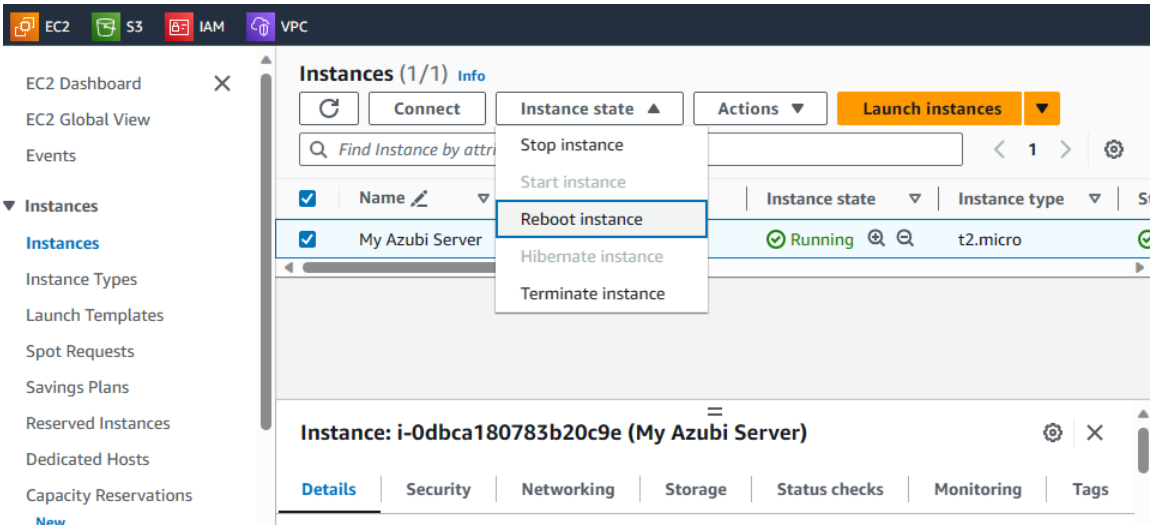


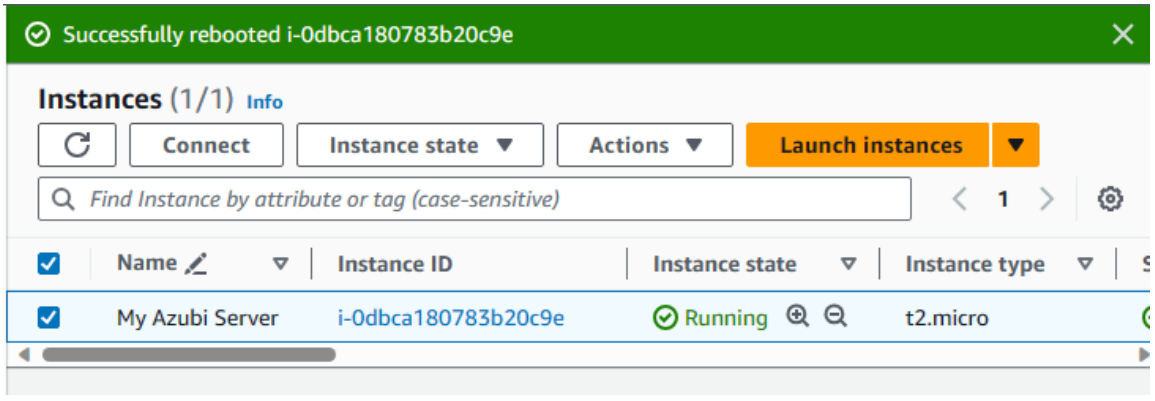
2. **Use EBS for Application Data:**

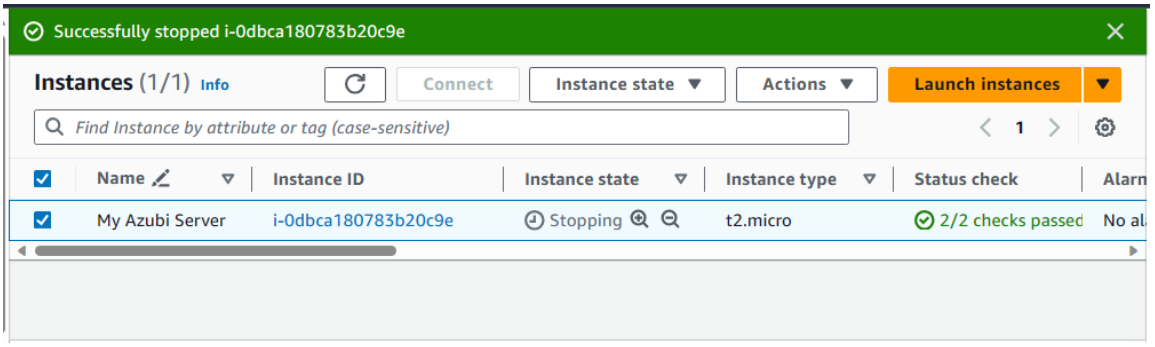
- Create a simple text file on the EBS volume.



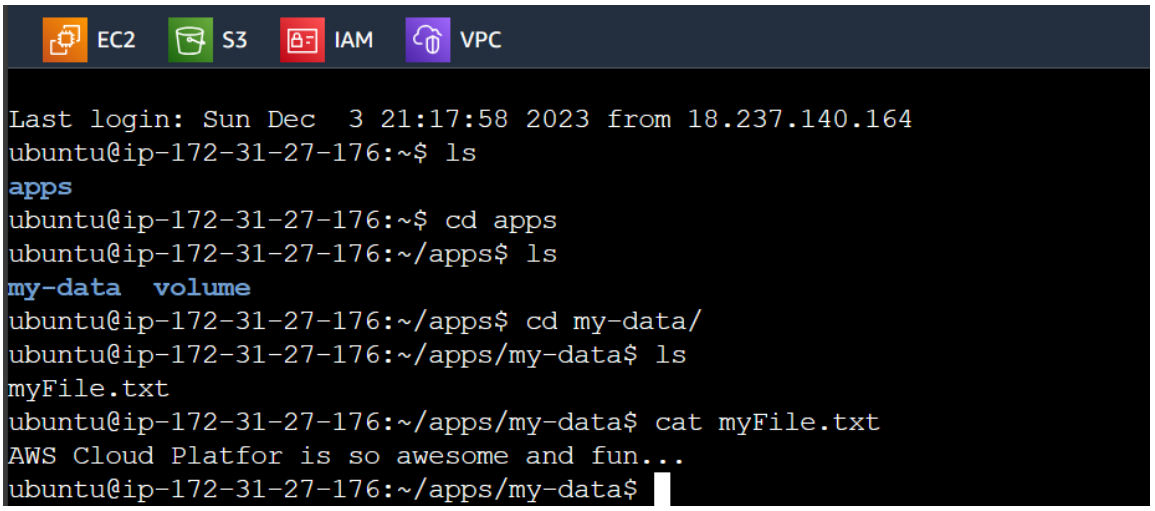
- Ensure the data persists even if the instance is stopped and started.



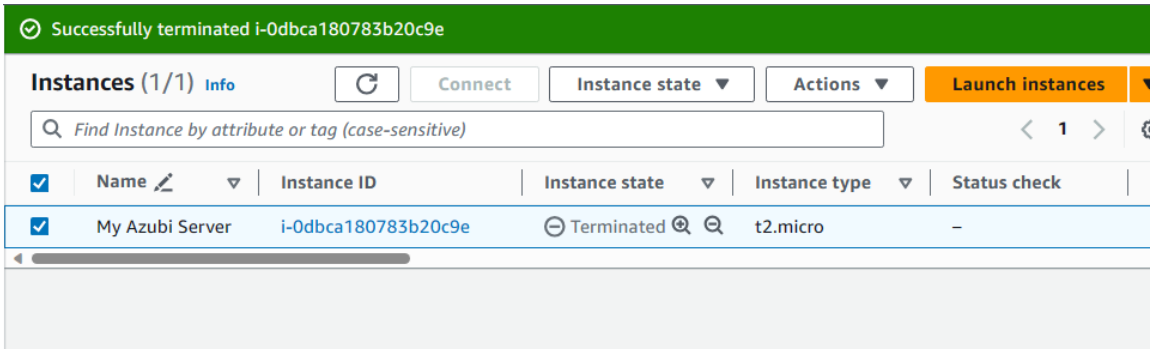




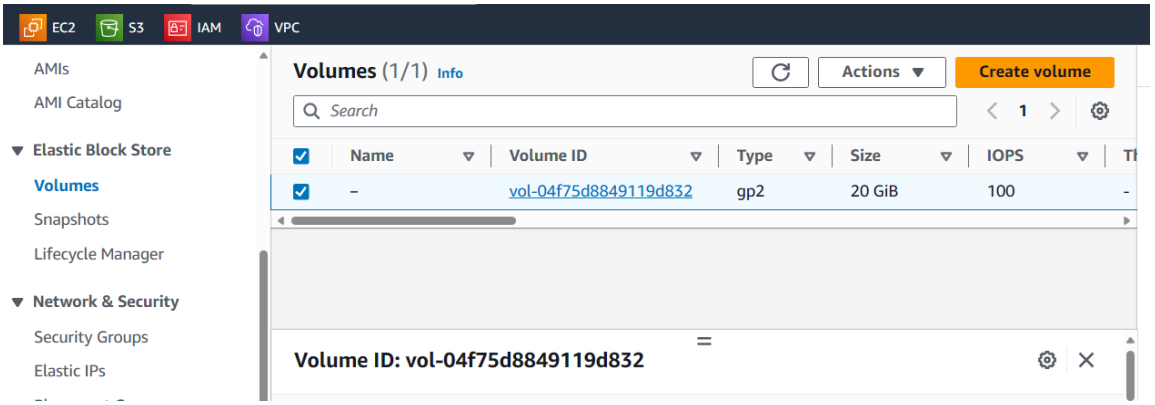
- After Reboot action, connecting to the EC2 instance and being able to access the folders and files that were created before the reboot signifies that the contents of the EBS volume persisted after the EC2 instance is rebooted, or stopped and restarted.



And even if you terminate the EC2 instance, the EBS volume will not be delete/removed except you specifically delete/remove it.

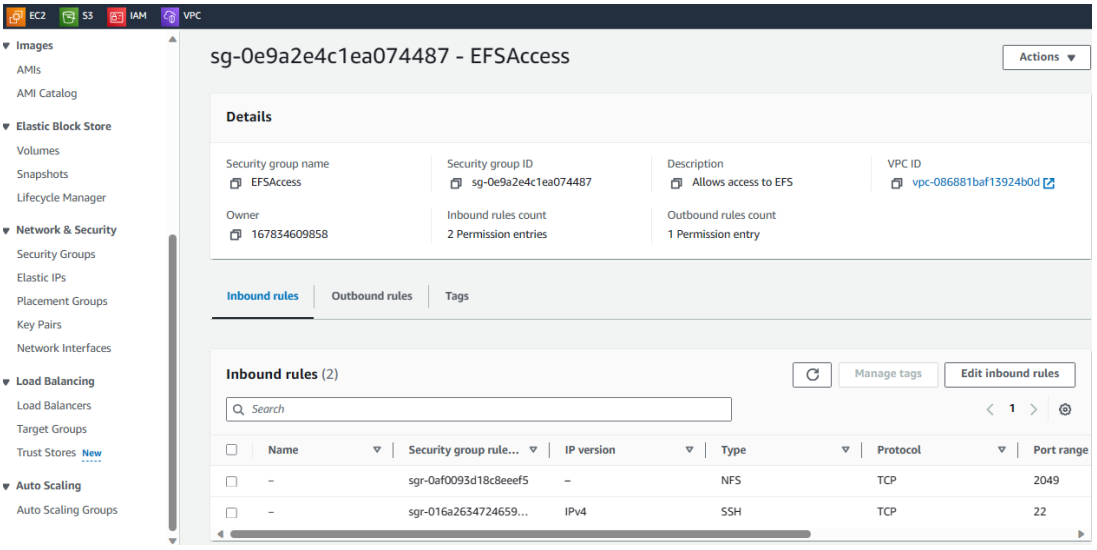


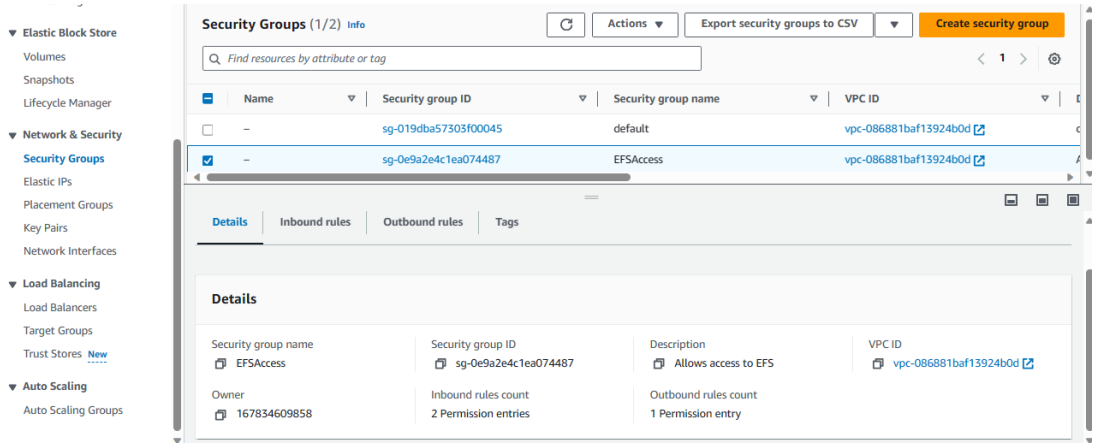
The EBS volume remaining even after the EC2 instance has been terminated. Which means that if you create a new EC2 instance, you can literally re-attach the EBS volume so that you can have access to the data on it.



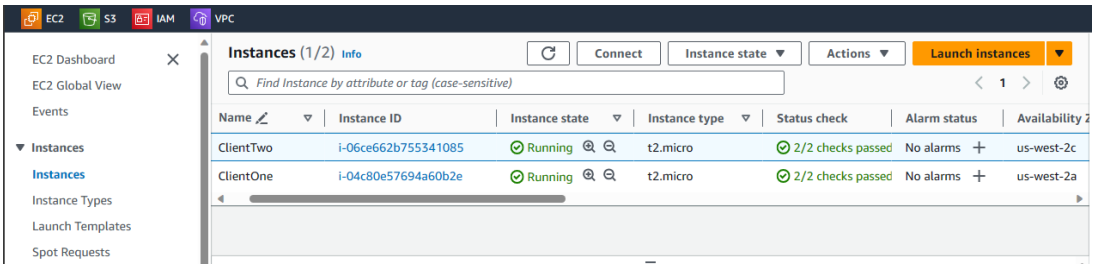
1. **Amazon EFS Setup:**

   - First of all, create a Security Group that will be used for both of the EC2 instances that will be created

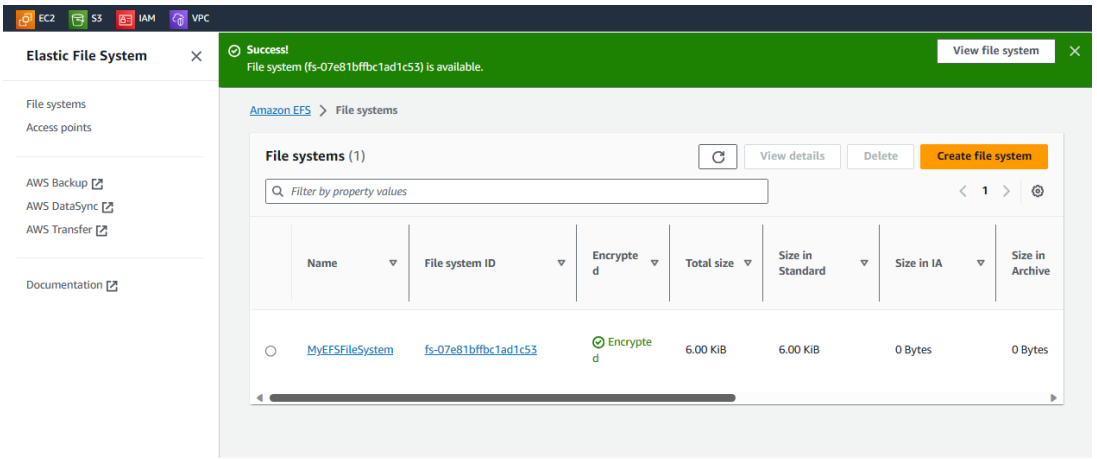




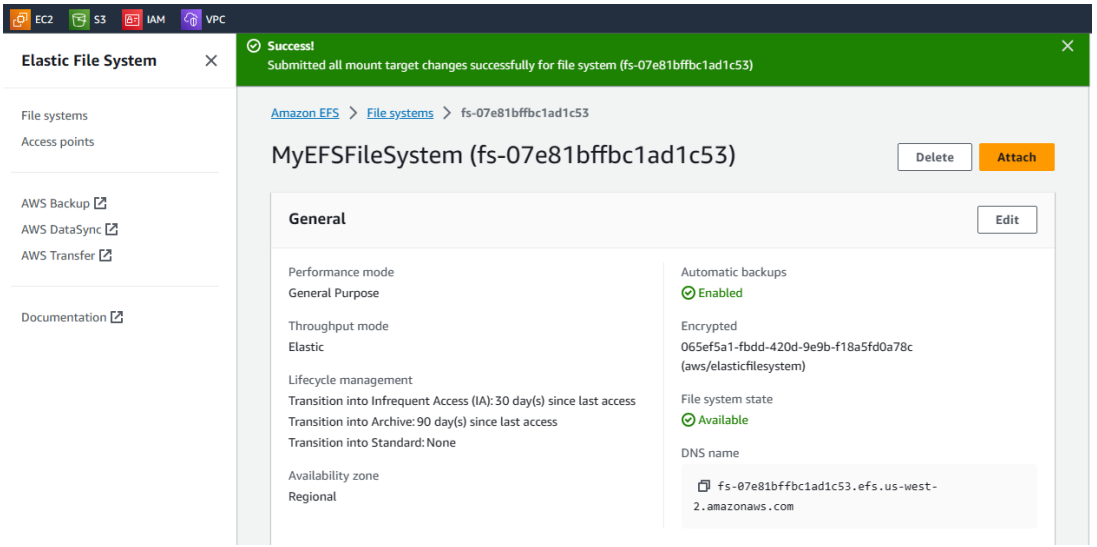
- Create 2 EC2 instances



- Create an Amazon EFS file system.



- Alter the EFS Network configuration to be associated with the Security Group that was created.



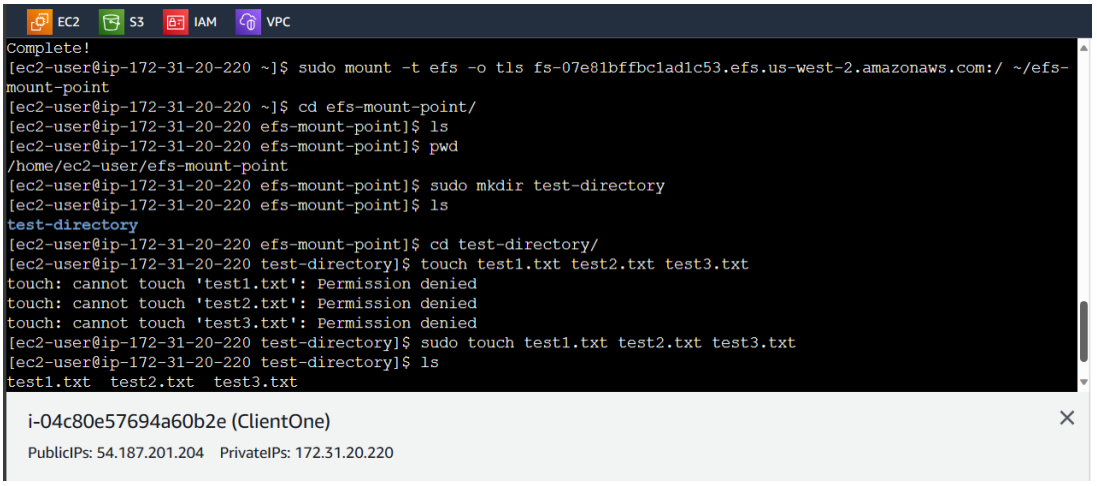
- Mount the file system on multiple EC2 instances using this command: “sudo mount -t efs -o tls fs-07e81bffbc1ad1c53.efs.us-west-

2.amazonaws.com:/ ~/efs-mount-point". And you will see that you can literally access the files created on the mounted directory on both EC2 instances. Here a directory with 4 different files are created on the first instance. We will then find out that the directory and files created from the first instance can be accessed seen from the second instance.

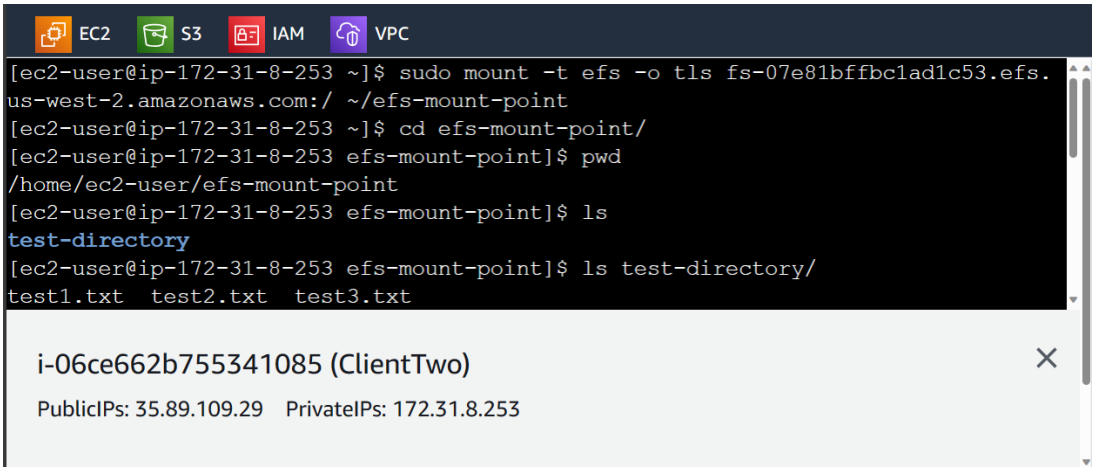
2. **Use EFS for Shared Data:**

- Create a file on one instance and verify its presence on another.

This is the file system as seen from the first instance.



- This is the file system as seen from the second instance.



- And likewise, if some files are created from the second instance, they will be accessible from the first instance. Create some new files in the second EC2 instance. And view access it in the first EC2 instance.

```
[ec2-user@ip-172-31-8-253 efs-mount-point]$ sudo touch another.txt yetAnother.txt
[ec2-user@ip-172-31-8-253 efs-mount-point]$ ls
another.txt  test-directory  yetAnother.txt
[ec2-user@ip-172-31-8-253 efs-mount-point]$ mv another.txt test-directory/another.txt
mv: cannot move 'another.txt' to 'test-directory/another.txt': Permission denied
[ec2-user@ip-172-31-8-253 efs-mount-point]$ sudo mv another.txt test-directory/another.txt
[ec2-user@ip-172-31-8-253 efs-mount-point]$ sudo mv yetAnother.txt test-directory/yetAnother.txt
[ec2-user@ip-172-31-8-253 efs-mount-point]$ cd test-directory/
[ec2-user@ip-172-31-8-253 test-directory]$ ls
another.txt  test1.txt  test2.txt  test3.txt  yetAnother.txt
[ec2-user@ip-172-31-8-253 test-directory]$
```

i-06ce662b755341085 (ClientTwo)

PublicIPs: 35.89.109.29 PrivateIPs: 172.31.8.253

```
[ec2-user@ip-172-31-20-220 test-directory]$ ls
another.txt  test1.txt  test2.txt  test3.txt  yetAnother.txt
[ec2-user@ip-172-31-20-220 test-directory]$
```

i-04c80e57694a60b2e (ClientOne)

PublicIPs: 54.187.201.204 PrivateIPs: 172.31.20.220

- Observe how changes to the file on one instance are reflected on the other. Here is adding a line of text to a file from the first EC2 instance.

```
[ec2-user@ip-172-31-20-220 test-directory]$ sudo echo "All things are bright and beautiful..." > test1.txt
-bash: test1.txt: Permission denied
[ec2-user@ip-172-31-20-220 test-directory]$ sudo chmod 777 test1.txt
[ec2-user@ip-172-31-20-220 test-directory]$ echo "All things are bright and beautiful..." > test1.txt
[ec2-user@ip-172-31-20-220 test-directory]$ cat test1.txt
All things are bright and beautiful...
[ec2-user@ip-172-31-20-220 test-directory]$
```

i-04c80e57694a60b2e (ClientOne)

PublicIPs: 54.187.201.204 PrivateIPs: 172.31.20.220

And we find that we will literally be able to access and add additional lines of text into the same file in the second EC2 instance.

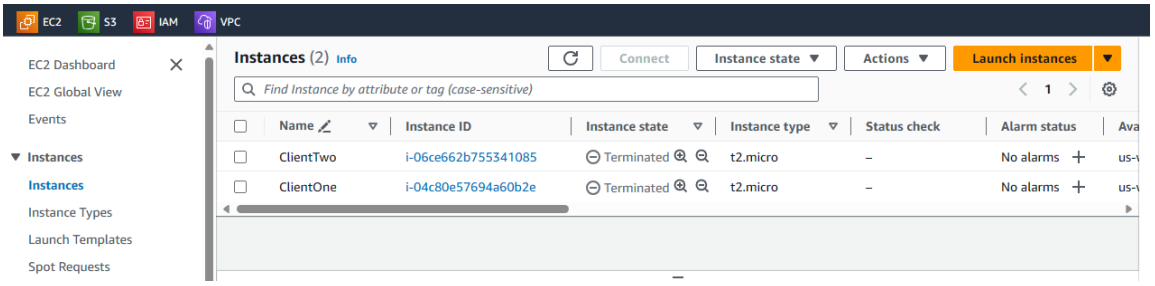
```
[ec2-user@ip-172-31-8-253 test-directory]$ ls
another.txt  test1.txt  test2.txt  test3.txt  yetAnother.txt
[ec2-user@ip-172-31-8-253 test-directory]$ cat test1.txt
All things are bright and beautiful...
[ec2-user@ip-172-31-8-253 test-directory]$ echo "The potential of the human mind knows no bounds..." >> test1.tx
t
[ec2-user@ip-172-31-8-253 test-directory]$ cat test1.txt
All things are bright and beautiful...
The potential of the human mind knows no bounds...
[ec2-user@ip-172-31-8-253 test-directory]$
```

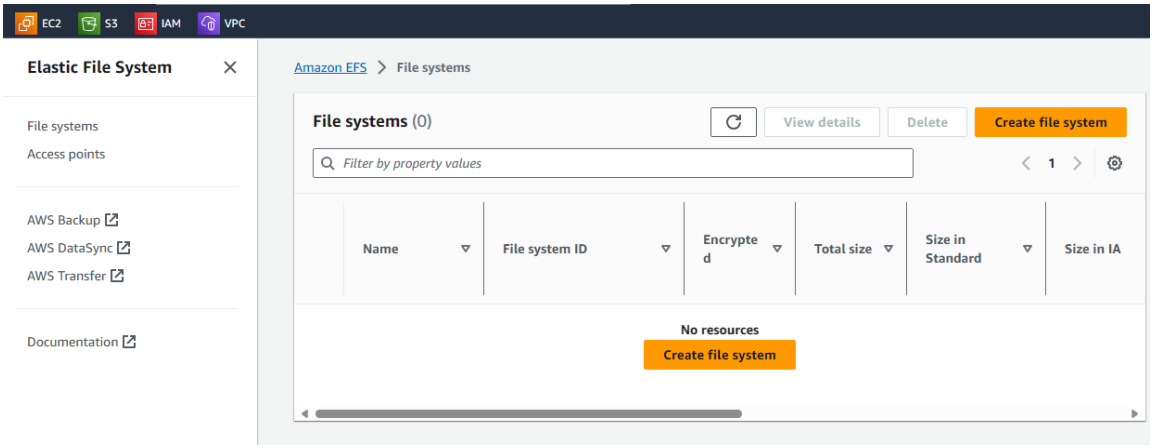
i-06ce662b755341085 (ClientTwo)

PublicIPs: 35.89.109.29 PrivateIPs: 172.31.8.253

3. Delete all created resources.

Now, ensure to terminate the 2 EC2 instances and delete the EFS file system in order not to incur unnecessary costs.





And you can view the associated GitHub here - AWS Storage Project