AWS - EBS

AWS EBS (Amazon Elastic Block Store) is a cloud storage service that provides persistent block storage for Amazon EC2 instances. It allows you to store data that can be accessed like a hard drive. Key features include:

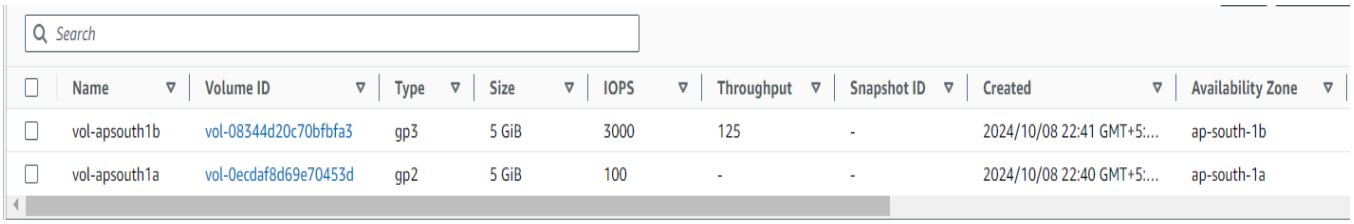
- Durability: EBS volumes are designed to be highly available and durable.
- Backup: You can create snapshots to back up your data.
- Performance: Offers different volume types for various workloads (e.g., SSD for high performance, HDD for cost-effective storage).
- Scalability: You can easily resize volumes and adjust performance as needed.

AWS EBS offers four main volume types:

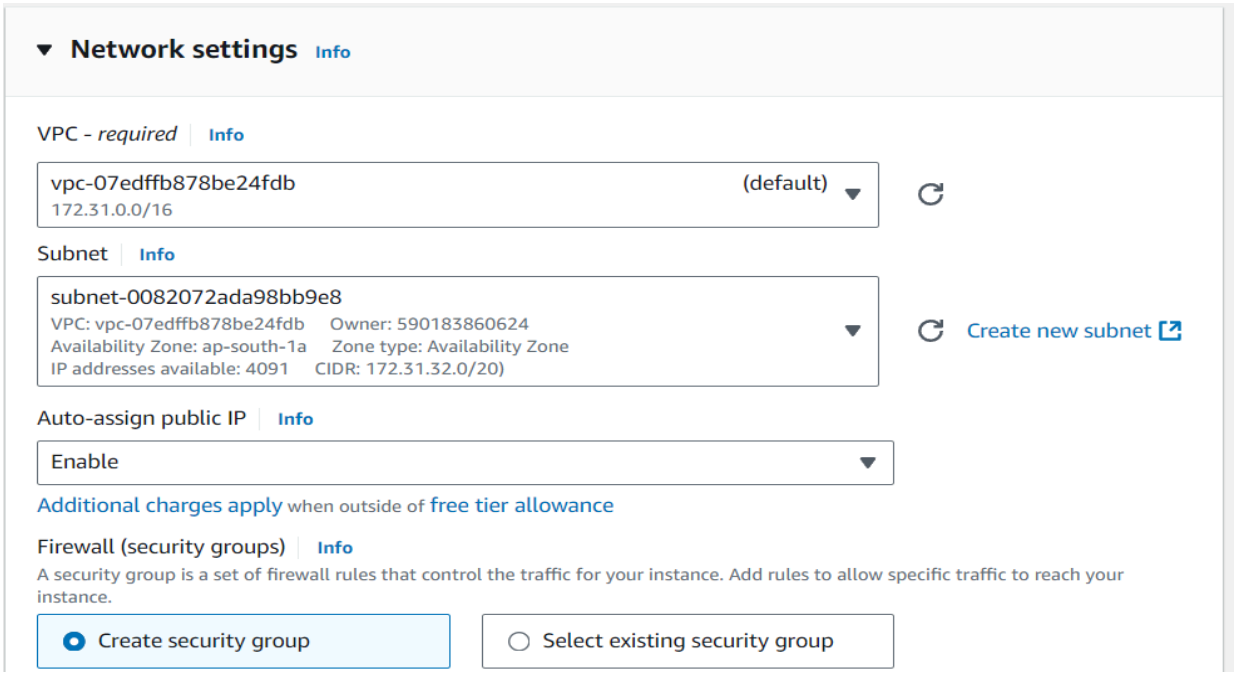
1. **General Purpose SSD (gp3/gp2)**:
   - Balanced performance and cost.
   - Best for most workloads like boot volumes, small databases.
2. **Provisioned IOPS SSD (io2/io1)**:
   - High performance and IOPS (Input/Output Operations per Second).
   - Ideal for mission-critical databases.
3. **Throughput Optimized HDD (st1)**:
   - Low-cost, high throughput for large, sequential data.
   - Suitable for big data, data warehouses.
4. **Cold HDD (sc1)**:
   - Cheapest, lower performance.
   - Best for infrequent access, like backups.

Each type suits different use cases based on performance and cost needs.

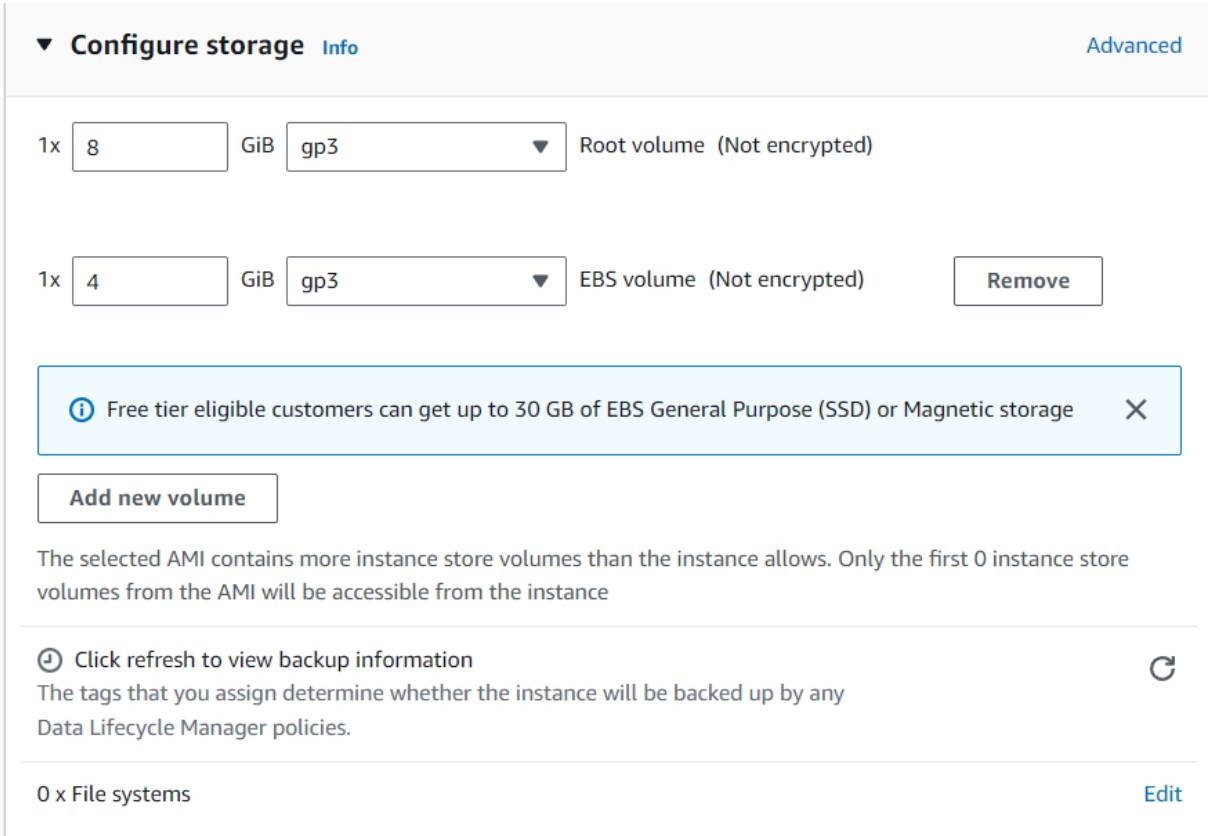
Creating Two volumes in 2 different Availability Regions



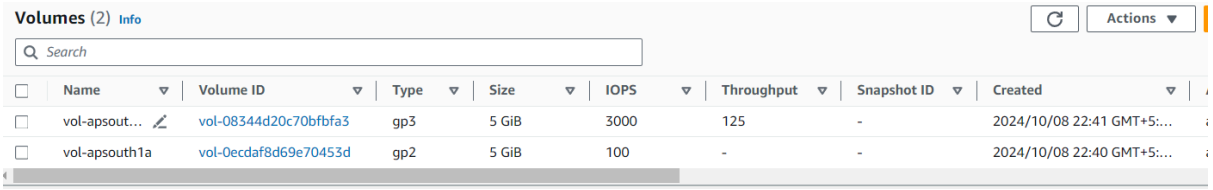
Creating an instance in South-1a.



Adding an EBC value of 4GB.



New Volumes will be created:



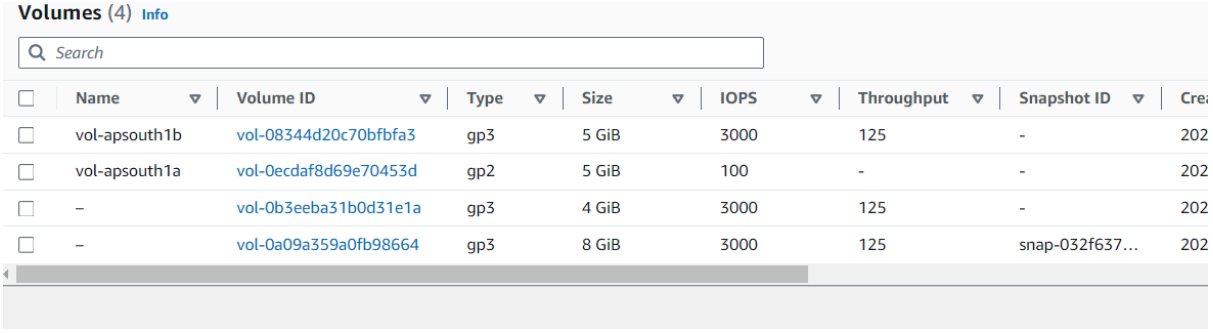
Volumes (2) Info

Actions

Search

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created
vol-apsout...	vol-08344d20c70bfbfa3	gp3	5 GiB	3000	125	-	2024/10/08 22:41 GMT+5:...
vol-apsouth1a	vol-0ecdaf8d69e70453d	gp2	5 GiB	100	-	-	2024/10/08 22:40 GMT+5:...

1. Default Storage. 2. EBS of 4GB



Volumes (4) Info

Search

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Crea
vol-apsouth1b	vol-08344d20c70bfbfa3	gp3	5 GiB	3000	125	-	202
vol-apsouth1a	vol-0ecdaf8d69e70453d	gp2	5 GiB	100	-	-	202
–	vol-0b3eeba31b0d31e1a	gp3	4 GiB	3000	125	-	202
–	vol-0a09a359a0fb98664	gp3	8 GiB	3000	125	snap-032f637...	202

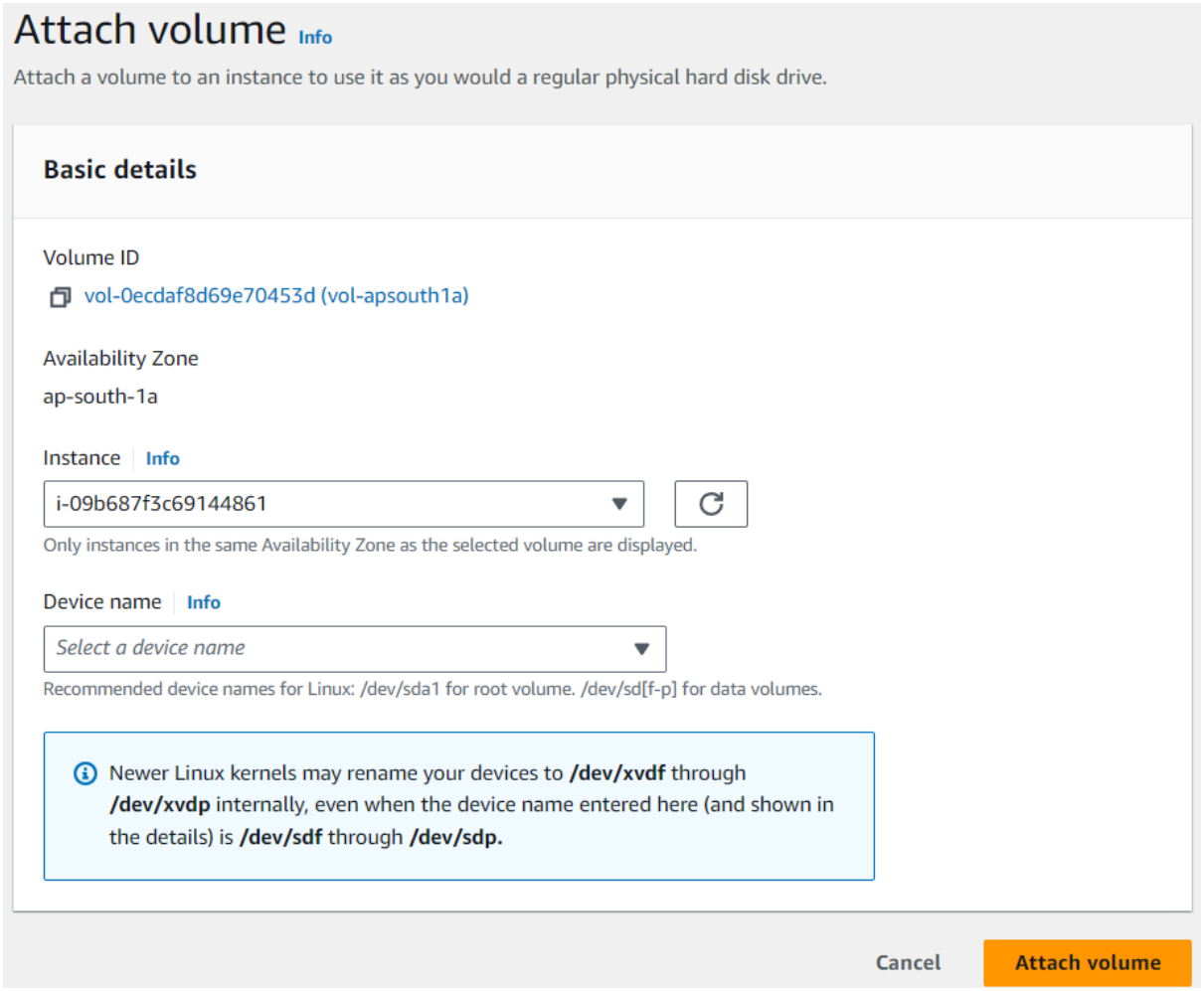
`lsblk` is a Linux command used to display information about block devices (like hard drives, SSDs, or attached storage). It shows details such as the device name, size, type (disk, partition), and mount point.

```
ubuntu@ip-172-31-41-78:~$ lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0       7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1       7:1    0 55.7M  1 loop /snap/core18/2829
loop2       7:2    0 38.8M  1 loop /snap/snapd/21759
xvda      202:0    0    8G  0 disk
├─xvda1   202:1    0    7G  0 part /
├─xvda14  202:14   0    4M  0 part
├─xvda15  202:15   0  106M  0 part /boot/efi
└─xvda16  259:0    0  913M  0 part /boot
xvdb      202:16   0    4G  0 disk
```

xvdb-4GB can be seen here at last.

Xvda 8GB - root storage..by default.

Connecting EBS Volume.
Apsouth-1b cant be connected to an instance in apsouth-1a.




```
ubuntu@ip-172-31-41-78:~$ lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0    8G  0 disk
├─xvda1  202:1    0    7G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdb     202:16   0    4G  0 disk
xvdf     202:80   0    5G  0 disk
```

Xvdf 5GB - EBS volume added.

Adding Data in this EBS.

`mkfs.ext4` is a Linux command used to format a disk or partition with the ext4 filesystem. It prepares the storage for use by creating an ext4 filesystem on it.
First lets format and then add data.

```
ubuntu@ip-172-31-41-78:~$ mkfs.ext4 /dev/xvdf
mke2fs 1.47.0 (5-Feb-2023)
mkfs.ext4: Permission denied while trying to determine filesystem size
ubuntu@ip-172-31-41-78:~$ sudo !!
sudo mkfs.ext4 /dev/xvdf
mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: a639d37d-5d6f-441a-a7d8-6762d17daa85
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Making a directory named TEST

```
root@ip-172-31-41-78:~# mkdir /test
root@ip-172-31-41-78:~# ls
snap
root@ip-172-31-41-78:~# cd /
root@ip-172-31-41-78:/# ls
bin             boot  etc   lib             lib64       media  opt   root  sbin             snap  sys   tmp  var
bin.usr-is-merged  dev   home  lib.usr-is-merged  lost+found  mnt    proc  run   sbin.usr-is-merged  srv   test  usr
```

mount /dev/xvdf /test/

The command `mount /dev/xvdf /test/` is used in Linux to mount a disk or partition (in this case, `/dev/xvdf`) to a specified directory (here, `/test/`).

Key Points:

- **`/dev/xvdf`**: This is the device you want to mount.
- **`/test/`**: This is the directory where the device will be mounted. You need to ensure that this directory exists before running the command.

Moving to test Directory and creating Files:

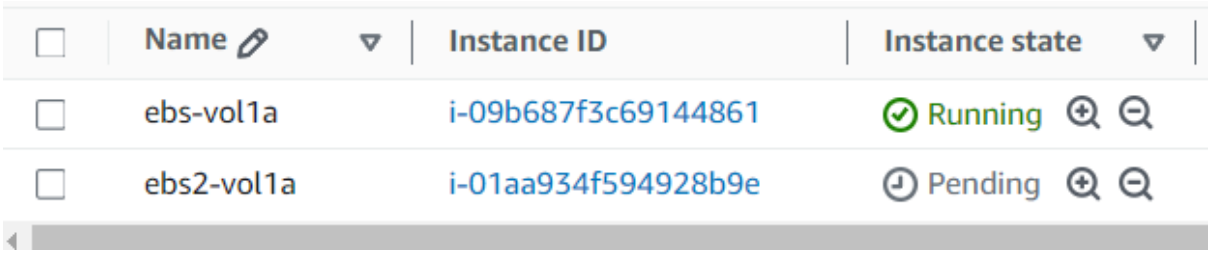
```
root@ip-172-31-41-78:/# cd /test
root@ip-172-31-41-78:/test# touch file1 file2
root@ip-172-31-41-78:/test# echo "hello irfan">irfan.txt
root@ip-172-31-41-78:/test# ls
file1  file2  irfan.txt  lost+found
root@ip-172-31-41-78:/test#
```

Going Back and Unmount the EBS

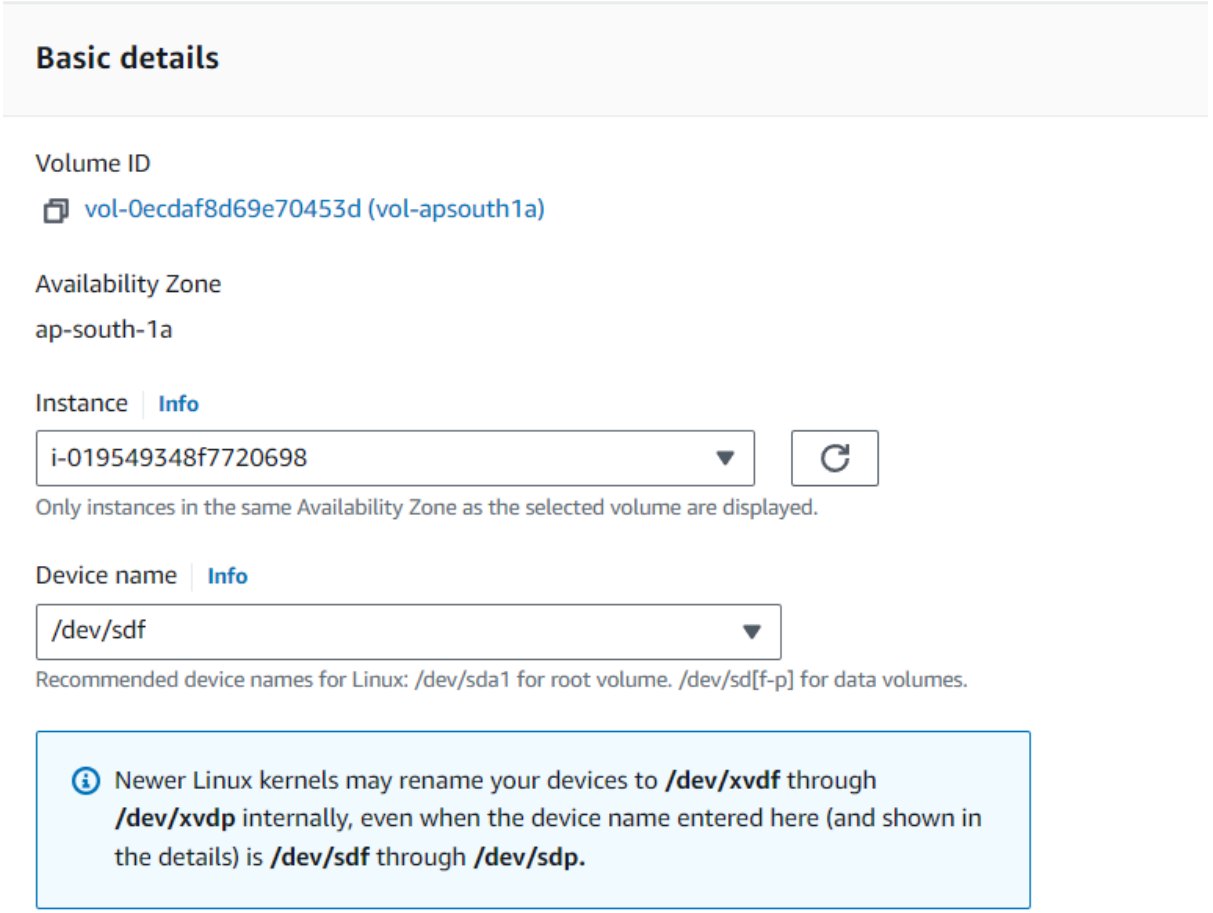
```
root@ip-172-31-41-78:/# cd /test
root@ip-172-31-41-78:/test# cd ..
root@ip-172-31-41-78:/# umount /test
root@ip-172-31-41-78:/# mountpoint /test
/test is not a mountpoint
```

Detach the EBS Volume xvdf to save cost.

Now to check if data persist in EBS volume created earlier.



New EC2 Instance created in south1a.



Attach the earlier EBS. To check if the data persists.

```
root@ip-172-31-42-212:~# lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0    8G  0 disk
├─xvda1  202:1    0    7G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdf     202:80   0    5G  0 disk
```

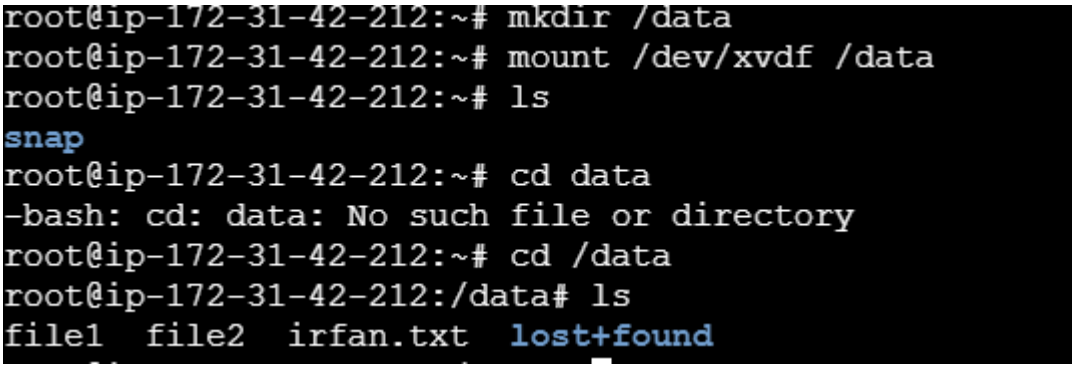
In 2nd EC2 instance
This time we will not format DATA.

The command `file -s /dev/xvdf` is used in Linux to check the type of data present on the block device (in this case, `/dev/xvdf`).

What it does:

- **`file`**: The command identifies the file type of a given file.
- **`-s`**: Tells `file` to read from a special block device like `/dev/xvdf` to determine its contents.

.ext4 file already present.



Mount new folder.
Check data present. Data Persists.

Resize EBS Volume and Resize the File System

Modify volume Info

Modify the type, size, and performance of an EBS volume.

Volume details

Volume ID

vol-0ecdaf8d69e70453d (vol-apsouth1a)

Volume type | Info

General Purpose SSD (gp2)

Size (GiB) | Info

7

Min: 1 GiB, Max: 16384 GiB. The value must be an integer.

IOPS | Info

100/3000

Baseline of 3 IOPS per GiB with a minimum of 100 IOPS, burstable to 3000 IOPS.

Size can be increased . Cant be decreased.

```
└─xvda16 259:0    0  913M  0 part /boot
xvdf     202:80   0    5G  0 disk /data
root@ip-172-31-42-212:/data# lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0    8G  0 disk
├─xvda1  202:1    0    7G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdf     202:80   0    7G  0 disk /data
```

The command `df -h` displays the disk space usage in a human-readable format (with sizes in MB, GB, etc.).

Breakdown:

- **`df`**: Displays information about disk space usage.
- **`-h`**: Makes the output human-readable (e.g., 10G instead of 10240000K).

```
root@ip-172-31-42-212:/data# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.8G  1.6G  5.2G  24% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  872K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
/dev/xvdf       4.9G   28K  4.6G   1% /data
```

Block Storage has grown. But file system has not grown.

The command `resize2fs /dev/xvdf` is used to resize an ext2/ext3/ext4 filesystem on a block device (in this case, `/dev/xvdf`).

Usage:

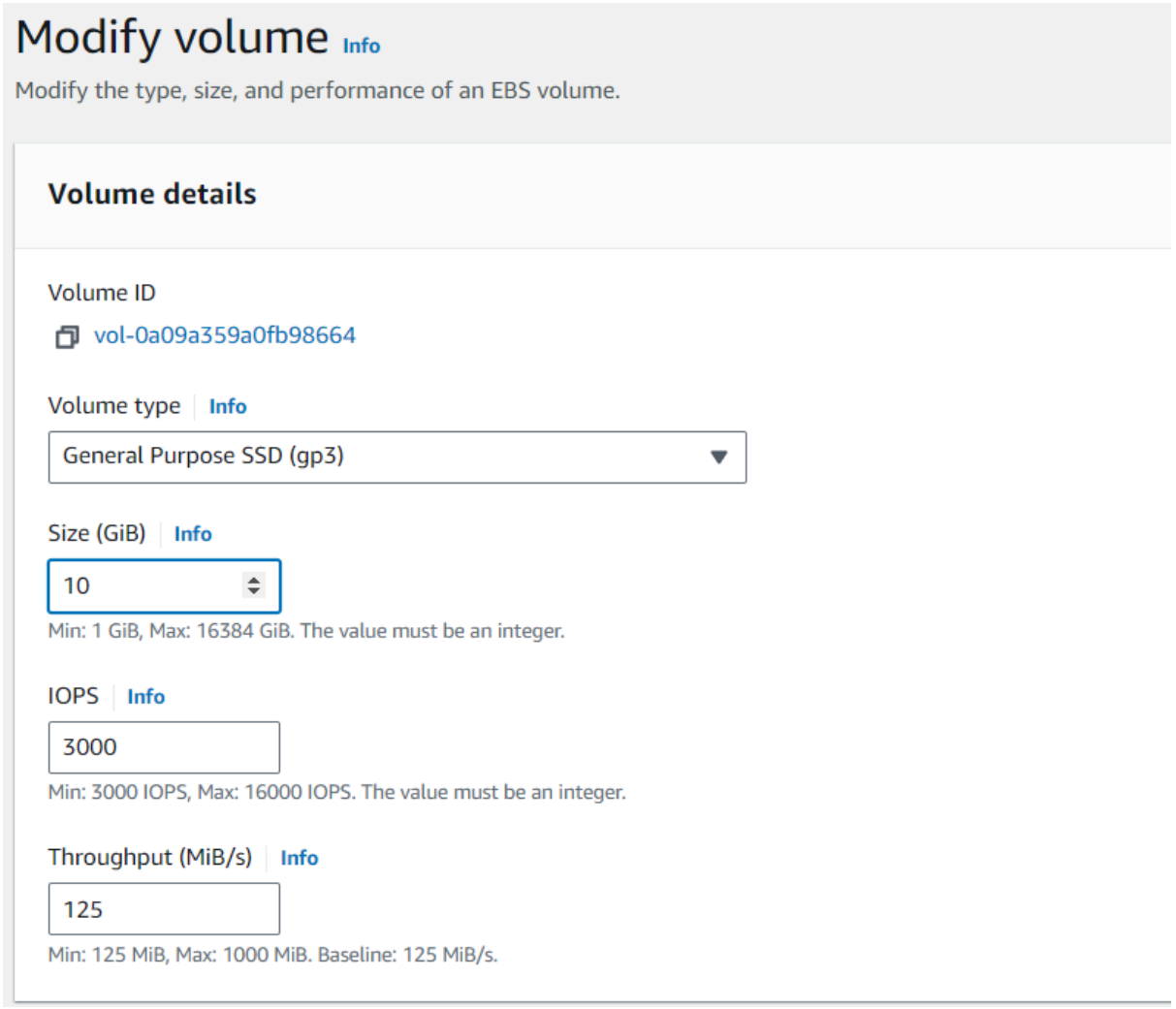
- If the block device has been extended, you can use this command to grow the filesystem to fill the new space.

```
root@ip-172-31-42-212:/data# resize2fs /dev/xvdf
resize2fs 1.47.0 (5-Feb-2023)
Filesystem at /dev/xvdf is mounted on /data; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/xvdf is now 1835008 (4k) blocks long.
```

Now , how to decrease Size.

Create a new EBS of smaller size. Transfer data. And unmount the larger EBS.

How to Resize ROOT EBS Volume




```
root@ip-172-31-42-212:/data# lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0   10G  0 disk
├─xvda1  202:1    0    7G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdf     202:80   0    7G  0 disk /data
```

```
root@ip-172-31-42-212:/data# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.8G  1.6G  5.2G  24% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  872K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
/dev/xvdf       6.9G   28K  6.5G   1% /data
```

File system didn't grow. /dev/root

```
root@ip-172-31-42-212:/data# resize2fs /dev/xvda1
resize2fs 1.47.0 (5-Feb-2023)
The filesystem is already 1834747 (4k) blocks long.  Nothing to do!
```

Grow the Partition.

The command `growpart /dev/xvda 1` is used to extend the size of a specific partition (in this case, partition 1 of `/dev/xvda`) without affecting the existing data.

Breakdown:

- **`growpart`**: Command used to resize a partition.
- **`/dev/xvda`**: The disk on which the partition exists.
- **`1`**: Refers to the first partition of the disk.

```
root@ip-172-31-42-212:/data# growpart /dev/xvda 1
CHANGED: partition=1 start=2099200 old: size=14677983 end=16777182 n
root@ip-172-31-42-212:/data# resize2fs /dev/xvda1
resize2fs 1.47.0 (5-Feb-2023)
Filesystem at /dev/xvda1 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 2
The filesystem on /dev/xvda1 is now 2359035 (4k) blocks long.

root@ip-172-31-42-212:/data# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       8.7G  1.6G  7.1G  18% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  872K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
/dev/xvdf       6.9G   28K  6.5G   1% /data
root@ip-172-31-42-212:/data# lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0   10G  0 disk
├─xvda1  202:1    0    9G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdf     202:80   0    7G  0 disk /data
```

One EBS can be attached to multiple EC2 instance.

Snapshot Overview - AWS Snapshot - EBS Snapshot

An **EBS Snapshot** is a backup of your Amazon EBS volume at a specific point in time. It captures the data stored on an EBS volume and saves it to Amazon S3, providing a way to restore the volume in case of data loss or corruption.

Key Points:

- **Incremental Backups**: After the first snapshot, only changes made since the last snapshot are saved, reducing storage costs.
- **Restore Volumes**: Snapshots can be used to create new EBS volumes in the same or different regions.

- **Automated Backups**: Snapshots can be created manually or automated using AWS Backup or Data Lifecycle Manager (DLM).
- **Durability**: Snapshots are stored in Amazon S3, ensuring high durability.

Commands:

Create Snapshot (from volume):
bash
Copy code

```
aws ec2 create-snapshot --volume-id vol-xxxxxxxx --description "My snapshot"
```

-

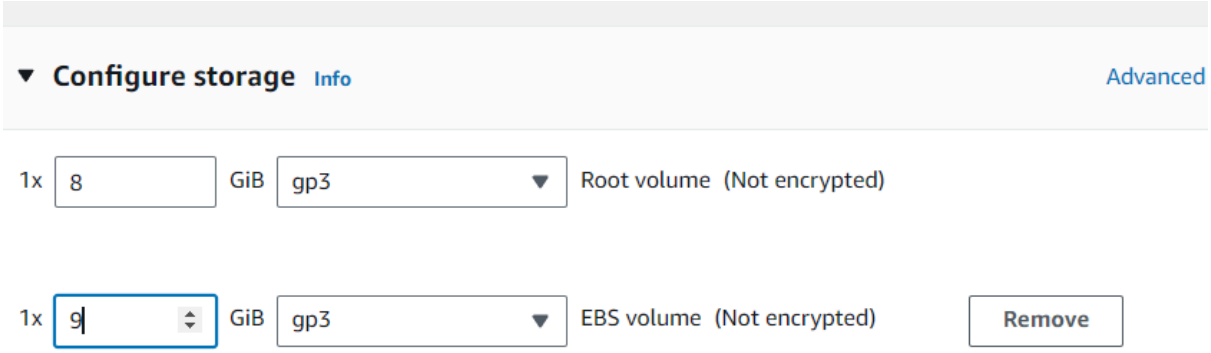
List Snapshots:
bash
Copy code

```
aws ec2 describe-snapshots --owner-ids self
```

-

Snapshots provide a reliable way to back up and restore EBS volumes for disaster recovery and data migration.

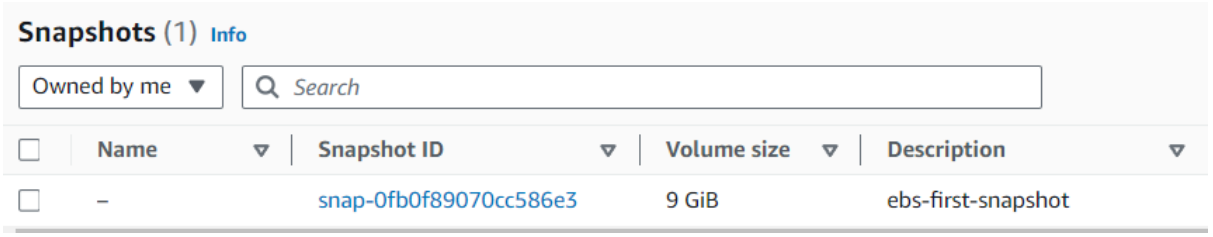
Creating an instance.



```
root@ip-172-31-14-152:~# file -s /dev/xvdb
/dev/xvdb: Linux rev 1.0 ext4 filesystem data, UUID=ce52742e-b4dd-4806-9
root@ip-172-31-14-152:~# mkdir folder_i
root@ip-172-31-14-152:~# mount /dev/xvdb /folder_i
mount: /folder_i: mount point does not exist.
       dmesg(1) may have more information after failed mount system call
root@ip-172-31-14-152:~# ls
folder_i  snap
root@ip-172-31-14-152:~# mount /dev/xvdb folder_i
root@ip-172-31-14-152:~# mountpoint folder_i
folder_i is a mountpoint
root@ip-172-31-14-152:~# cd folder_i
root@ip-172-31-14-152:~/folder_i# ls
lost+found
root@ip-172-31-14-152:~/folder_i# yes "irfan">sample.txt
^C
root@ip-172-31-14-152:~/folder_i# yes "irfan">>sample.txt
^C
root@ip-172-31-14-152:~/folder_i# yes "irfan">> sample.txt
ls -lh
^C
root@ip-172-31-14-152:~/folder_i# ls -lh
total 2.6G
drwx------ 2 root root  16K Oct  9 06:30 lost+found
-rw-r--r-- 1 root root 2.6G Oct  9 06:34 sample.txt
root@ip-172-31-14-152:~/folder_i#
```

Files and Folders added:

```
root@ip-172-31-14-152:~/folder_i# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.8G  1.6G  5.2G  24% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  872K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
/dev/xvdb       8.8G  2.6G  5.8G  31% /root/folder_i
root@ip-172-31-14-152:~/folder_i#
```

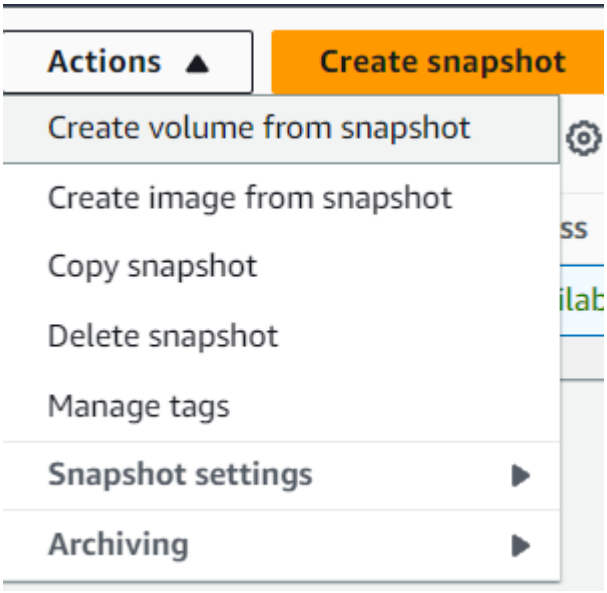


Snapshot Created.
Now make some changes the EBS. Add some new data. Go back to previous versions.

See if we can go back to old data using Snapshot.

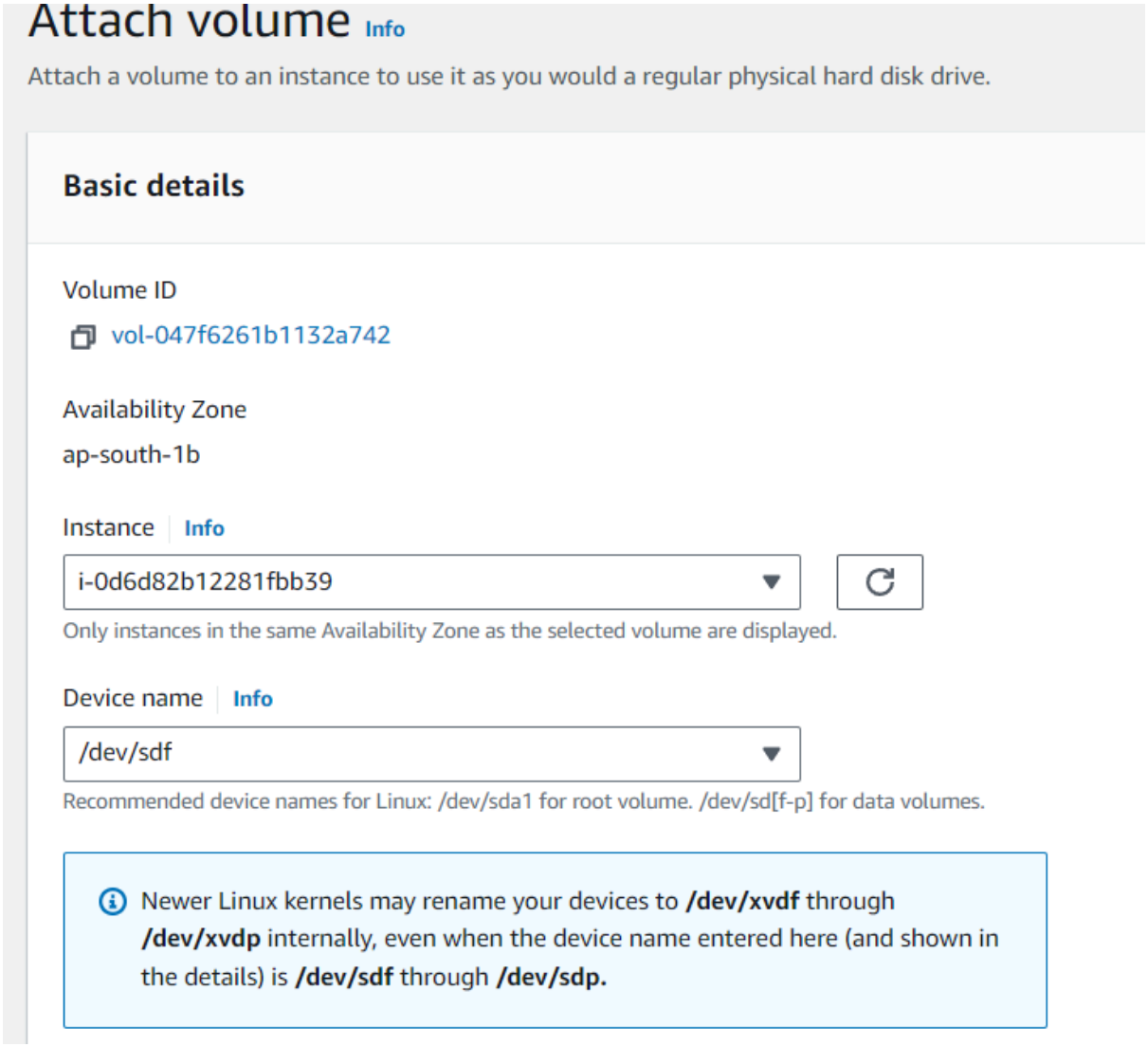
```
root@ip-172-31-14-152:~/folder_i# ls -lh
total 292M
-rw-r--r-- 1 root root   12 Oct  9 06:40 file1.txt
drwx------ 2 root root  16K Oct  9 06:30 lost+found
-rw-r--r-- 1 root root 292M Oct  9 06:37 sample.txt
```

New file1.txt added.



Creating new Volume from Snapshot.

Keeping same availability Zone . So i can mount this EBS volume in my EC2 instance which is in 1b.




```
root@ip-172-31-14-152:~/folder_i# lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0    8G  0 disk
├─xvda1  202:1    0    7G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdb     202:16   0    9G  0 disk /root/folder_i
xvdf     202:80   0    9G  0 disk
```

New Volume Added.

```
root@ip-172-31-14-152:~/folder_i# file -s /dev/xvdf
/dev/xvdf: Linux rev 1.0 ext4 filesystem data, UUID=ce52742e-b4dd-4806-930b-
root@ip-172-31-14-152:~/folder_i#
```

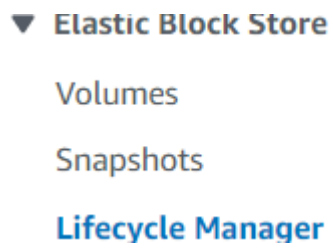
File system already exists in this.

```
root@ip-172-31-14-152:~/folder_i# cd ..
root@ip-172-31-14-152:~# mkdir folder_i2
root@ip-172-31-14-152:~# mount /dev/xvdf folder_i2
root@ip-172-31-14-152:~# mountpoint folder_i2
folder_i2 is a mountpoint
root@ip-172-31-14-152:~# cd folder_i2
root@ip-172-31-14-152:~/folder_i2# ls
lost+found  sample.txt
root@ip-172-31-14-152:~/folder_i2# ls -lh
total 292M
drwx------ 2 root root  16K Oct  9 06:30 lost+found
-rw-r--r-- 1 root root 292M Oct  9 06:37 sample.txt
root@ip-172-31-14-152:~/folder_i2#
```

That sample.txt of 292 mb exists.

The other folder which we added after creating snapshot do not exists.

Automate EBS Volume Backup - EBS Lifecycle Manager - EBS Backup



Enter LifeCycle Manager.

In AWS, **Lifecycle Manager** for EBS Snapshots helps automate the creation, retention, and deletion of snapshots based on predefined policies.

Retention Type: Count

When setting up a policy to automatically manage volume snapshots, **Retention Type: Count** refers to the number of snapshots to retain. Once the limit is reached, older snapshots are automatically deleted.

Example:

- **Keep 15 Snapshots**: If you set the policy to "Keep 15 snapshots", the system will automatically retain the most recent 15 snapshots of the EBS volume. Once 16 snapshots are created, the oldest one will be deleted to maintain the limit of 15.

Snapshots in Standard Tier:

- **Snapshots** are stored in the **Standard Tier**, which is the default storage class. This tier is cost-effective for regular snapshots.
- You can also move snapshots to the **Archive Tier** for long-term storage at lower costs, but retrieval times are slower.

How it Works:

1. **Snapshot Creation**: Lifecycle Manager creates snapshots at scheduled intervals (e.g., every day, week).
2. **Retention Policy**: The policy checks how many snapshots exist. If more than 15, it deletes the oldest ones.
3. **Cost Optimization**: Only the most recent 15 snapshots are kept in the Standard Tier, optimizing costs and storage management.

This approach ensures automated, reliable backups while managing costs by not over-accumulating snapshots.

Volume settings

Volume type | Info

General Purpose SSD (gp3)

Size (GiB) | Info

4

Min: 1 GiB, Max: 16384 GiB. The value must be an integer.

IOPS | Info

3000

Min: 3000 IOPS, Max: 16000 IOPS. The value must be an integer.

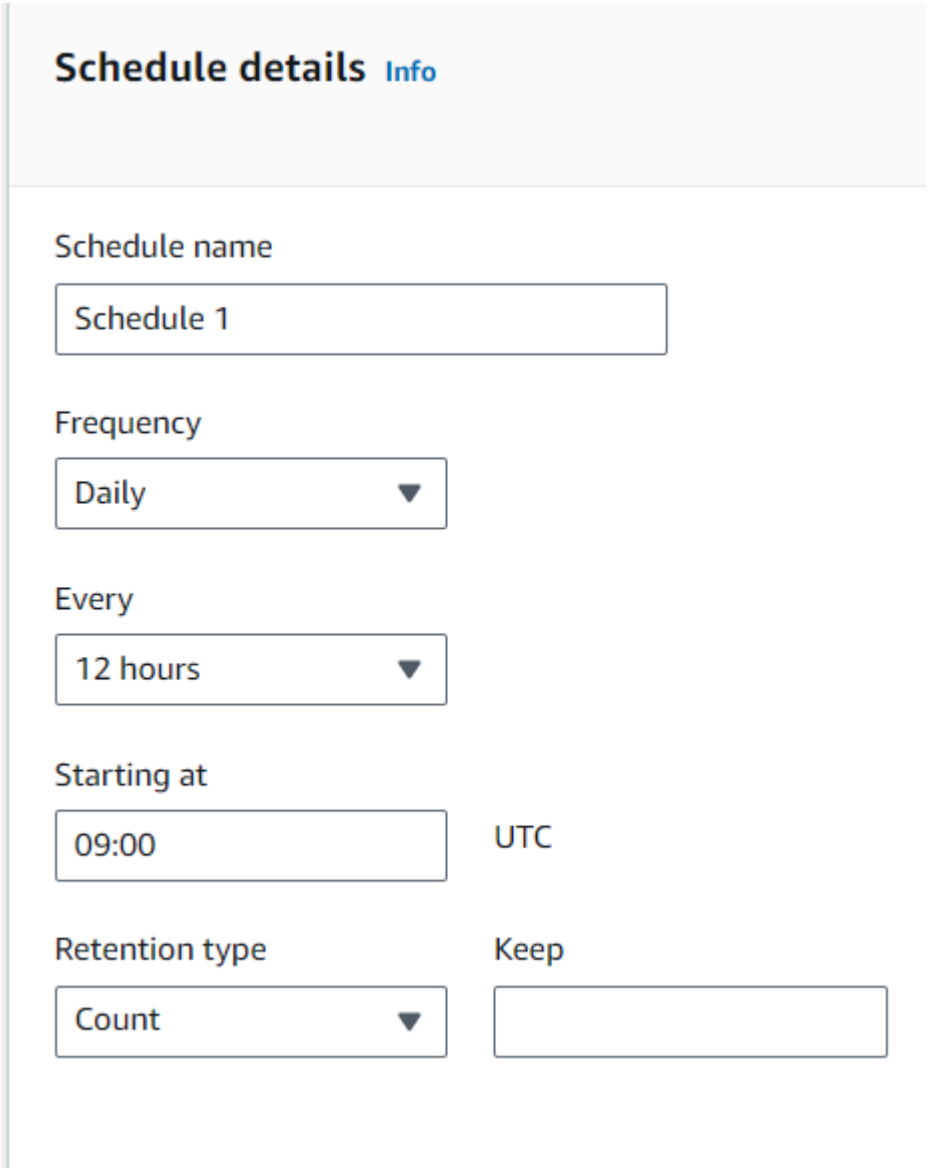
Throughput (MiB/s) | Info

125

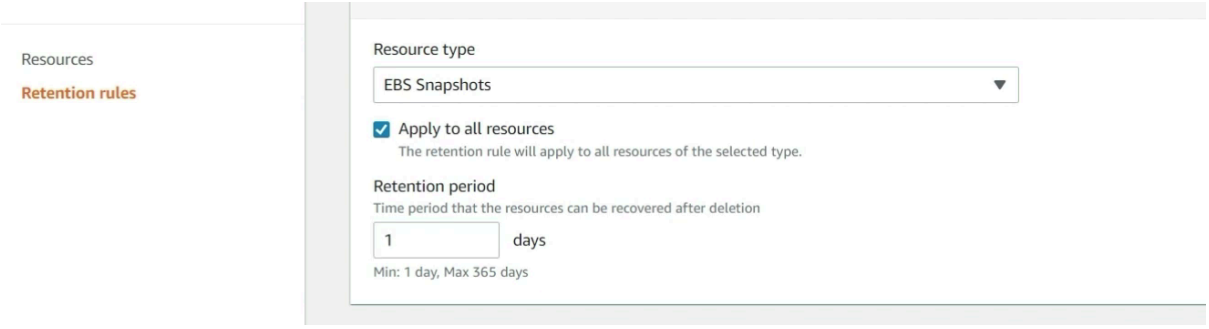
Min: 125 MiB, Max: 1000 MiB. Baseline: 125 MiB/s.

Availability Zone | Info

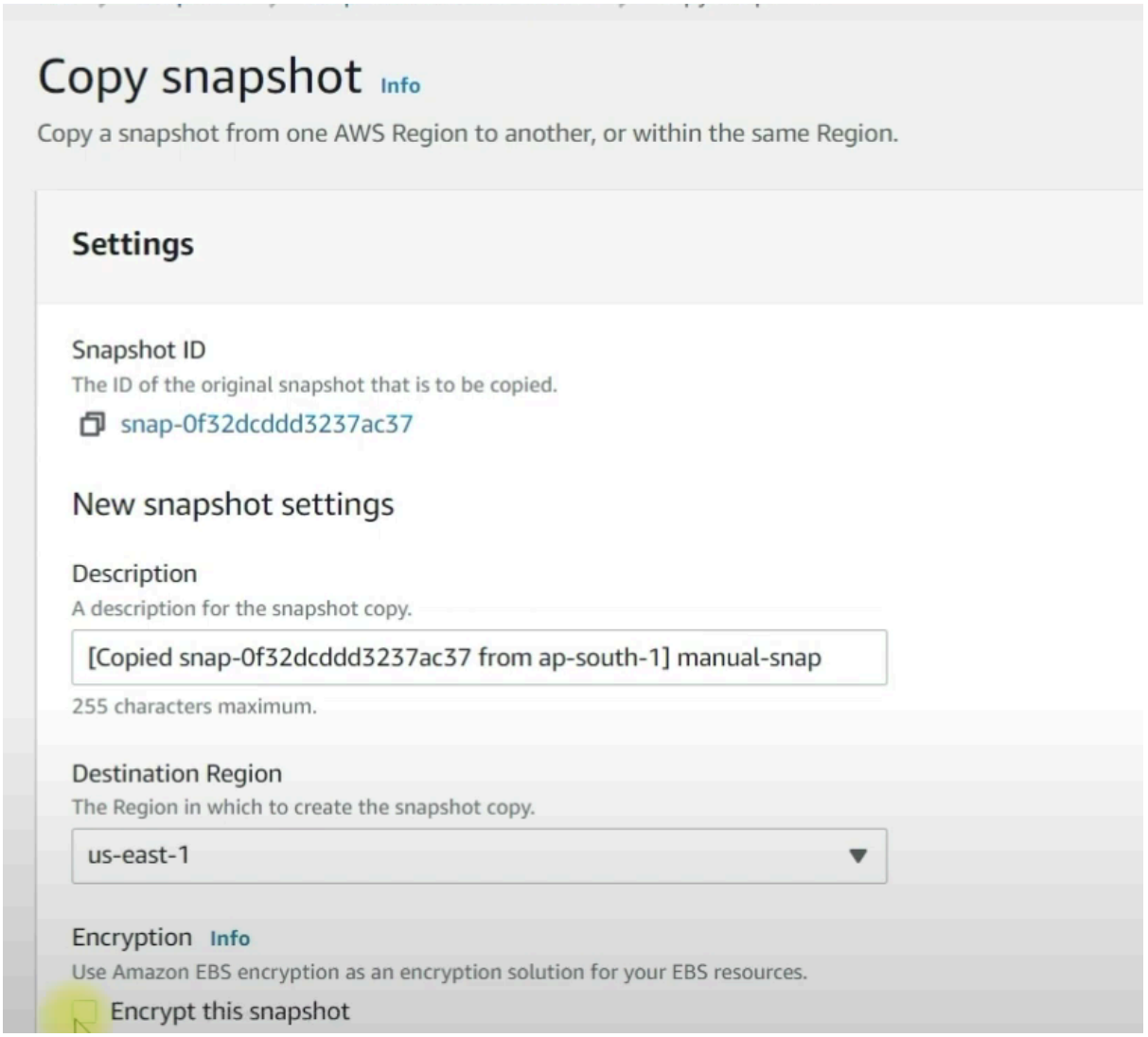
ap-south-1b



Snapshot and AMI Recycle Bin - Recycle Bin for Snapshot and AMI AWS



Copy Snapshot From One Region to Another- Copy Snapshot Cross Region/Account



AWS AMI - Amazon Machine Image - Create your Own AMI

An **AWS AMI (Amazon Machine Image)** is a pre-configured virtual machine image that contains the necessary information to launch instances (EC2 instances) in AWS. It includes the OS, application code, libraries, and data.

Benefits:

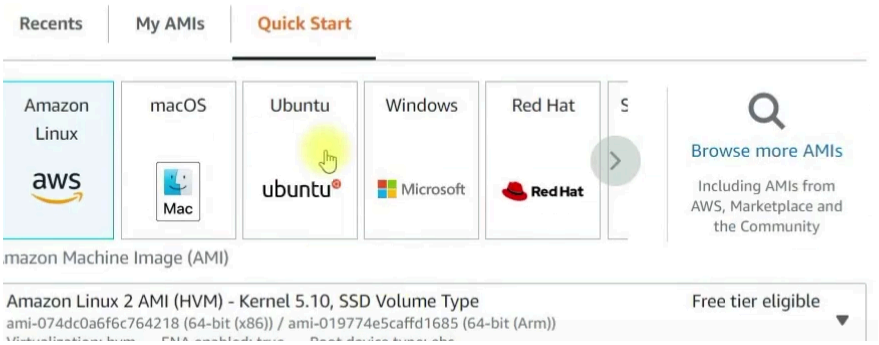
- **Consistency**: Use the same AMI across multiple instances for consistent environments.
- **Customization**: Tailor the AMI to your specific software needs.
- **Faster Deployment**: Launch pre-configured instances quickly without repeated setup.

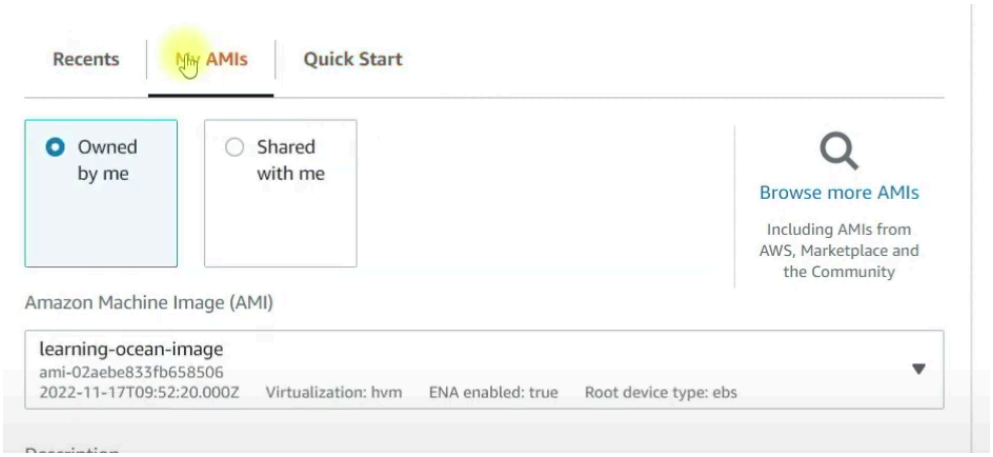
Creating custom AMIs is useful for scaling applications, maintaining backups, or deploying specific software environments.

Use Case: Launching Pre-configured Servers for a Web App

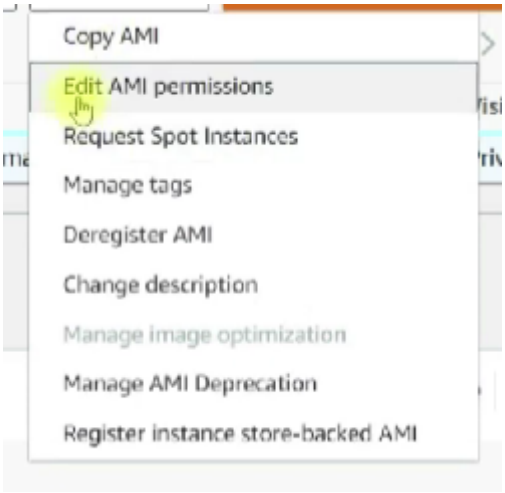
Imagine you are running a **web app** that requires specific software (like a web server, database, and custom settings). Every time you launch a new server, you would need to manually install and configure everything, which takes time.

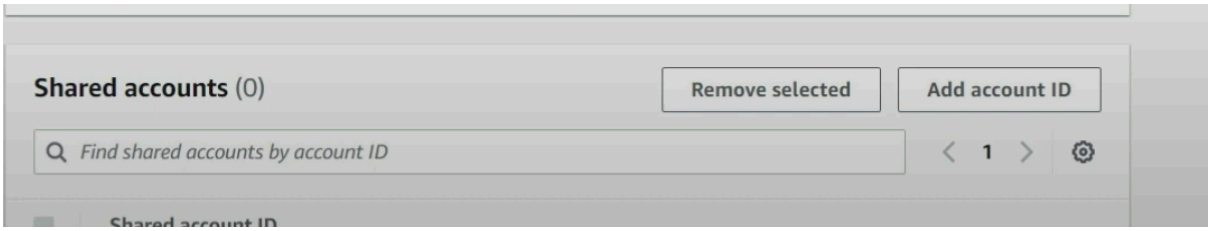
Pre-configured plan..just like a house design..many house can be built from that design.





Share AMI:


Copy AMI
Edit AMI permissions
Request Spot Instances
Manage tags
Deregister AMI
Change description
Manage image optimization
Manage AMI Deprecation
Register instance store-backed AMI


Shared accounts (0)
Remove selected
Add account ID
Find shared accounts by account ID
1