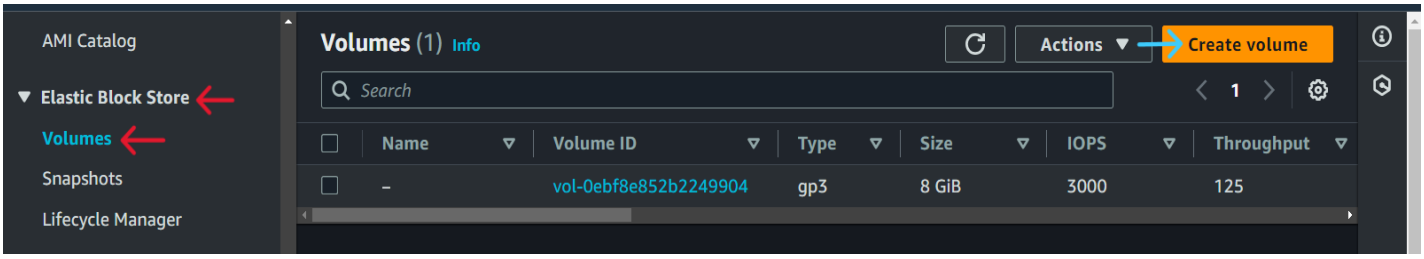
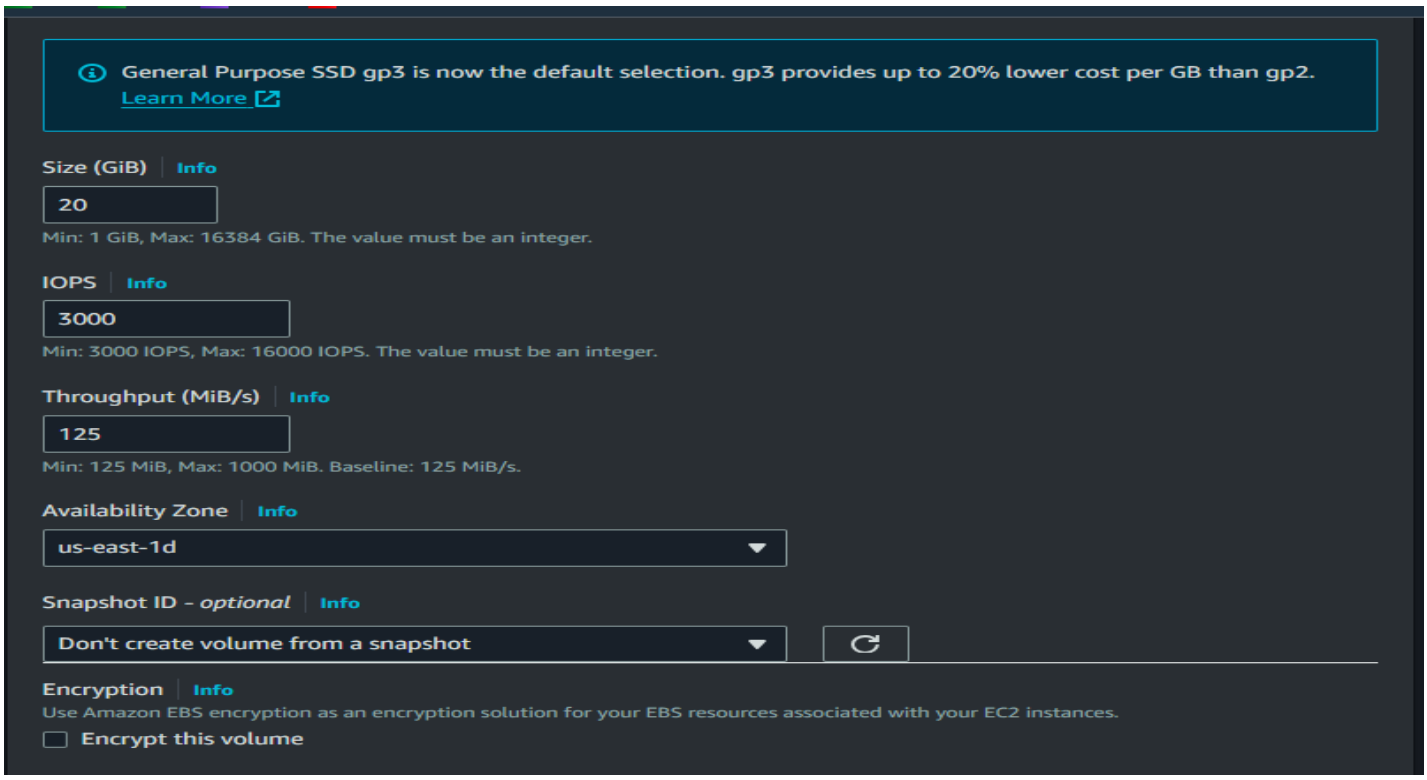
ELASTIC BLOCK STORAGE

Mount EBS Volume to EC2 Instance

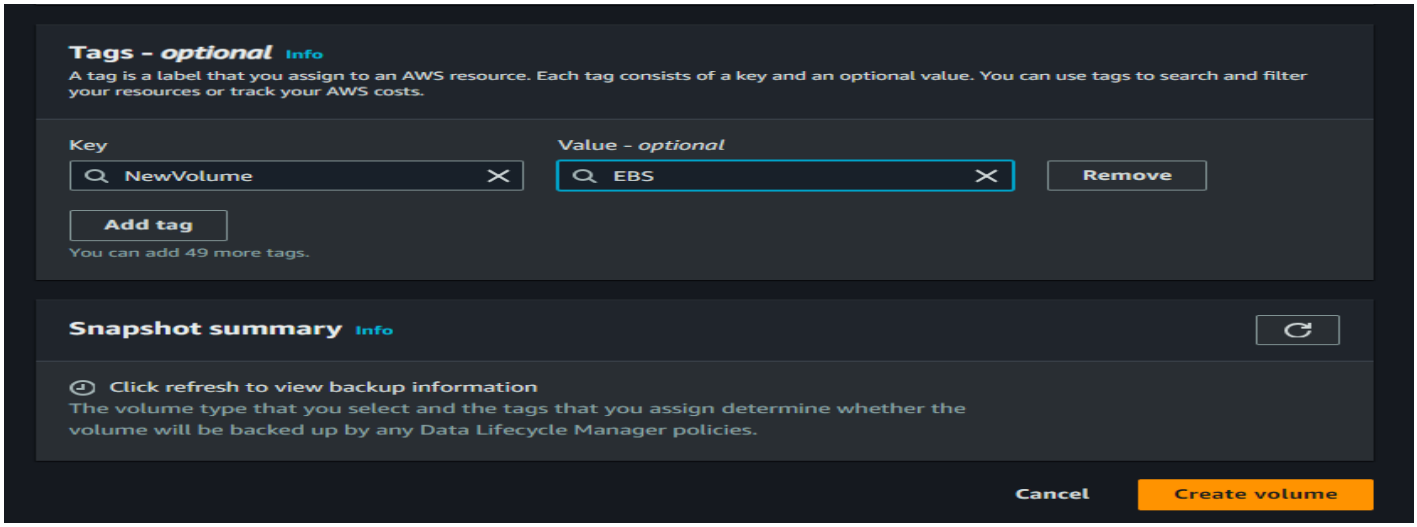
Step 1) Go EC2 dashboard and Click on the Volume option in the Elastic Block Store Menu. Then Click on the Create Volume Button.



Step 2) Now Configure your volume details like volume size, IPOS, Throughput, and availability zone, and you have the option to create a volume from a snapshot or not (Using a snapshot to create a volume has many use cases like data recovery, data replication, data migration between two deferent Instance or regions, Scaling storage, and testing & development)

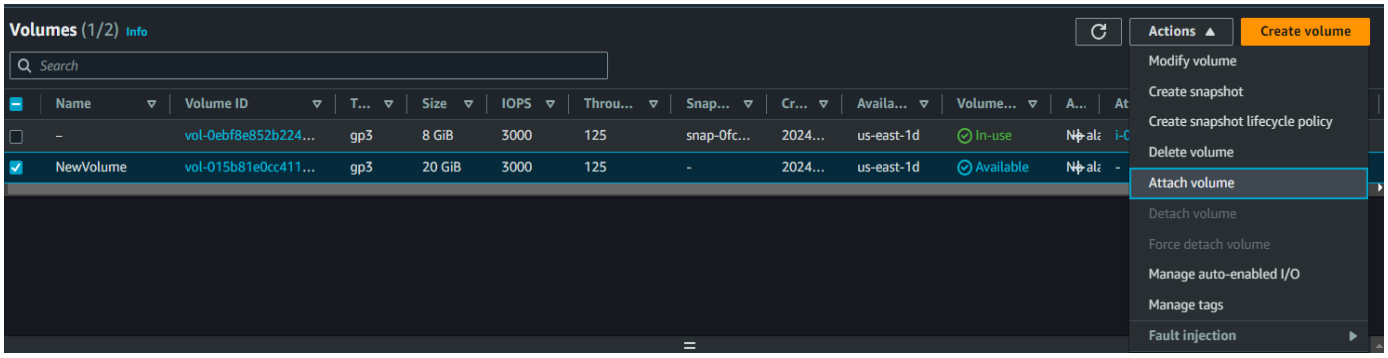


Step 3) Then simply add the tag (is optional), and click on the create volume button.

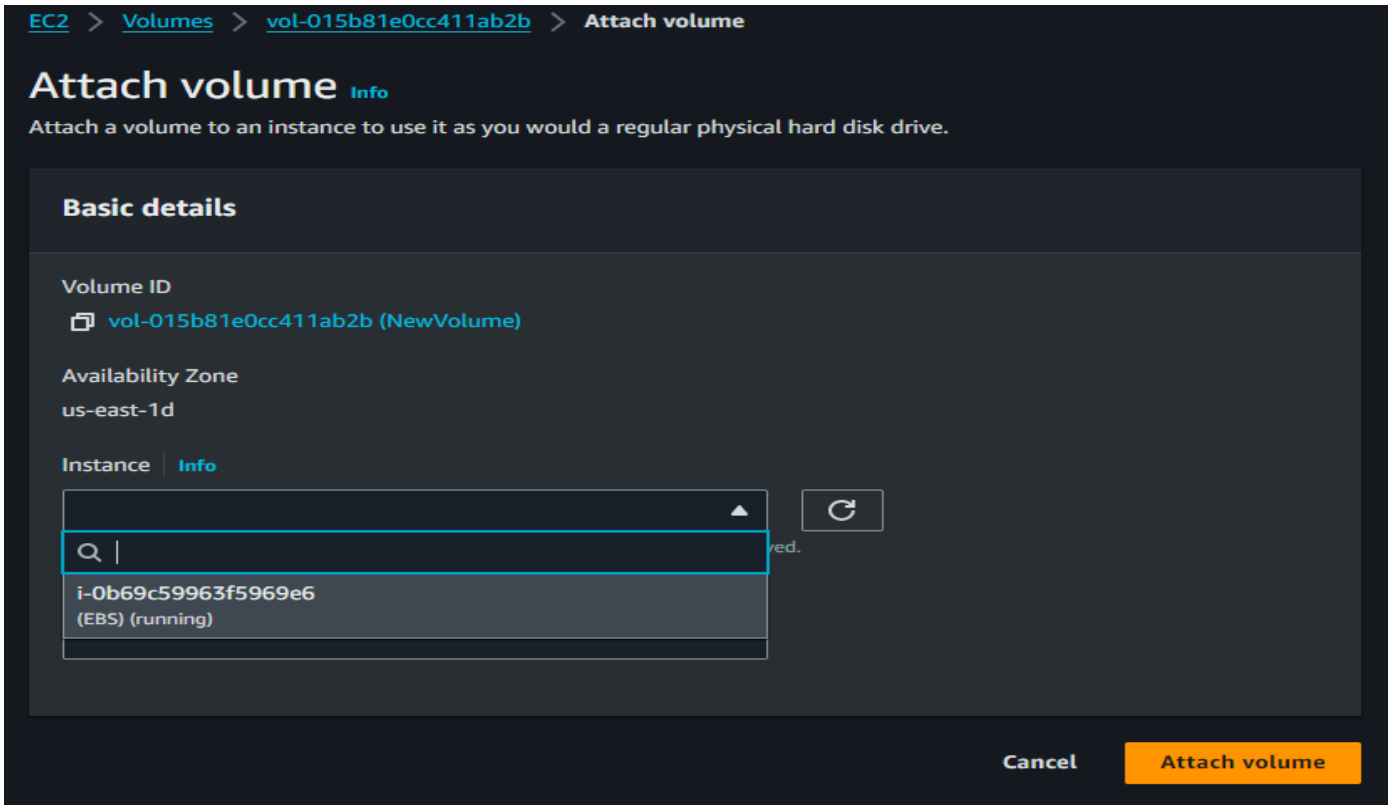


Now your volume is created. So we attach this volume to the Instance.

Step 4) Click on the Action Button and select the Attach volume option.



Step 5) Select your Instance ID where you want to attach this volume and click on Attach volume button.



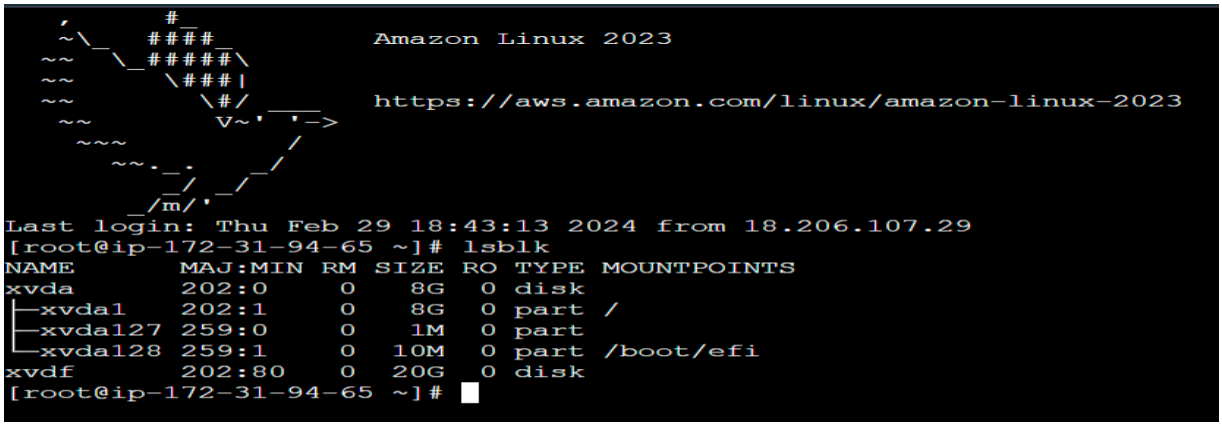
Now your volume is attached to your instance but this volume is not accessible, we need to initialize the volume which means formatting the volume with the file system (same as when we upgrade our PC's SSD) then we create a partition of volume (like our PC's C drive, D drive and E drive) after that we need to mount the volume to a directory within the instance file system.

Why we need this: - This makes the contents of the volume accessible to the operating system and any applications running on the instance.

So we need to access or connect to our Ec2 instance using SSH and run the following commands to mount the volume to the Ec2 instance.

Commands:-

- **lsbkl -> to list all block information and partitions**



Now we see our new volume **xvdf** attached but it does not have a partition like **xvda** or no files system mounted.

- **fdisk /dev/xvdf -> ** to make a partition of volume

```
[root@ip-172-31-94-65 ~]# fdisk /dev/xvdf

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x36d19c9e.

Command (m for help):
```

- **type m for help ->** all options list will open with description.

```
Help:

  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag

  Generic
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition

  Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)

  Script
   I   load disk layout from sfdisk script file
   O   dump disk layout to sfdisk script file

  Save & Exit
   w   write table to disk and exit
   q   quit without saving changes

  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table


Command (m for help):
```

- **type n for the new partition**
- **Then select partition type, by default on the primary just press Enter**
  We have up to **four primary partitions**, or Up to **three primary partitions and one extended partition** (which can then contain multiple logical partitions).
- **Then choose the partition number**
- **The option for partition starts from the first sector, simply press Enter, and proceed with the last sector.**
- **Type +10G and enter -> means create a 10 GB partition of disk and by default start from the last sector.**
- **Type wq and press Enter for save and exit**
- Run the **lsbkl** command to check the partition.

```
[root@ip-172-31-94-65 ~]# lsblk
NAME      MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
xvda      202:0    0   8G  0 disk
├─xvda1   202:1    0   8G  0 part /
├─xvda127 259:0    0   1M  0 part
└─xvda128 259:1    0  10M  0 part /boot/efi
xvdf      202:80   0  20G  0 disk
├─xvdf1   202:81   0  10G  0 part
└─xvdf2   202:82   0   5G  0 part
[root@ip-172-31-94-65 ~]#
```

Now you will see new partitions xvdf1 and xvdf2 that are successfully created.

Now we need to create the filesystem on that partition to store the files. Without creating a filesystem we can't use those partitions.

Linux File system types

Linux supports various file system types, each designed for different purposes. Some of the common file system types used in Linux are

1) ***ext4:*** *This is the default file system for many Linux distributions. It is a journaling file system and is backward compatible with its predecessors ext3 and ext2. ext4 offers improvements in performance, scalability, and reliability compared to ext3.*
2) ***XFS:*** *Known for its high performance and scalability, XFS is particularly well-suited for large-scale deployments and environments with heavy I/O workloads. It supports features like journaling, extended attributes, and large file systems.*
3) ***Btrfs (B-tree file system):*** *Btrfs is a modern copy-on-write (COW) file system that offers features such as snapshots, subvolumes, and built-in RAID support. It aims to provide better data integrity, scalability, and easier administration. While it's considered stable, it's still under active development.*
4) ***ZFS:*** *Although originally developed for Solaris, ZFS is available on Linux through projects like OpenZFS. ZFS is feature-rich, offering data integrity through checksums, built-in RAID, snapshots, and easy volume management. However, due to licensing issues, it's not I included in many Linux distributions by default.*
5) ***F2FS (Flash-Friendly File System):*** *Optimized for NAND flash storage devices like SSDs and eMMC, F2FS is designed to improve performance and extend the lifespan of flash-based storage. It employs techniques such as wear leveling and TRIM support.*
6) ***NTFS:*** *While primarily associated with Windows, Linux has NTFS support through the NTFS-3G driver. This allows Linux systems to read and write to NTFS-formatted drives, enabling interoperability with Windows systems.*
7) ***VFAT:*** *Another file system commonly associated with Windows, VFAT (Virtual File Allocation Table) is used for compatibility with older Windows systems and removable storage devices like USB drives and SD cards. Linux supports VFAT for read and write operations.*
8) ***ISO 9660:*** *This is the standard file system used for CD-ROMs and DVD-ROMs. It provides a read-only file system suitable for distributing software and data on optical media.*

To create a filesystem run the following commands:

- **mkfs –t xfs /dev/xvdf1 -> mkfs –t xfs - used to create a xfs filesystem**
- **mkfs –t ext4 /dev/xvdf2 -> mkfs –t ext4 - used to create a ext4 filesystem**

(As using same command mkfs –t [file type] /dev/sda you create other types of the filesystem)

- **blkid -> to check all blocks filesystem type and other information`rav**

```
[root@ip-172-31-19-205 ~]#
[root@ip-172-31-19-205 ~]# blkid
/dev/xvda128: SEC_TYPE="msdos" UUID="EA7D-FA7D" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="EFI System Partition" PARTUUID="7d97e
457-a415-478d-9207-ece727837f91"
/dev/xvda127: PARTLABEL="BIOS Boot Partition" PARTUUID="59d318c3-69e1-415c-bf1d-746623d6f25a"
/dev/xvda1: LABEL="/" UUID="81e4e009-191b-464c-8cc3-22de217d1136" BLOCK_SIZE="4096" TYPE="xfs" PARTLABEL="Linux" PARTUUID="dc
ea75b5-8895-48ba-96f4-e6ecd7175bba"
/dev/xvdf2: UUID="e2a2296c-c964-46af-976f-b47f4e2999ca" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="2a5f09e5-02"
/dev/xvdf1: UUID="a8a56205-75a0-47a8-ba72-b5b8377df7b3" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="2a5f09e5-01"
[root@ip-172-31-19-205 ~]#
```

- **xfs file system created**

```
[root@ip-172-31-19-205 ~]# mkfs -t xfs /dev/xvdf1
meta-data=/dev/xvdf1             isize=512    agcount=4, agsize=655360 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=2621440, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@ip-172-31-19-205 ~]#
```

- **ext4 file system created**

```
[root@ip-172-31-19-205 ~]# mkfs -t ext4 /dev/xvdf2
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: e2a2296c-c964-46af-976f-b47f4e2999ca
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
```

After the file system is created, we want to mount it to the directory in our file system so we can access it. So first we need to create a mount point in the [/mnt] directory this is the temporary mount point. (we can mount filesystem in any other directory)

- **mkdir /mnt/mydisk1**
- **mount /dev/xvdf1 /mnt/mydisk1**

Now you will see your filesystem is temporary mounted and available to use.

EC2 S3 EFS VPC IAM

```
[root@ip-172-31-19-205 ~]# mkdir /mnt/mydisk
[root@ip-172-31-19-205 ~]# mount /dev/xvdf1 /mnt/mydisk/
[root@ip-172-31-19-205 ~]# lsblk
NAME      MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
xvda      202:0    0   8G  0 disk
├─xvda1   202:1    0   8G  0 part /
├─xvda127 259:0    0   1M  0 part
└─xvda128 259:1    0  10M  0 part /boot/efi
xvdf      202:80   0  20G  0 disk
├─xvdf1   202:81   0  10G  0 part /mnt/mydisk
└─xvdf2   202:82   0   5G  0 part
[root@ip-172-31-19-205 ~]#
```

- **umount /dev/mnt/mydisk -> to unmount the volume**

To permanently mount the volume to the directory we need entry to add entry in "/etc/fstab" file.

- **vim /etc/fstab**
- **add entry [/dev/xvdf1 /mnt xfs default 0 0] in the file and save the file.**
- **mount –a -> to mount all file systems we listed.**
- **lsbkl -> to check file is mounted.**

EC2 S3 EFS VPC IAM

```
UUID=81e4e009-191b-464c-8cc3-22de217d1136     /           xfs    defaults,noatime  1   1
UUID=EA7D-FA7D        /boot/efi       vfat    defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2
/dev/xvdf1 /mnt xfs defaults 0 0
~
~
~
```

```
[root@ip-172-31-19-205 ~]# vim /etc/fstab
[root@ip-172-31-19-205 ~]# mount -a
[root@ip-172-31-19-205 ~]# lsblk
NAME      MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
xvda      202:0    0   8G  0 disk
├─xvda1   202:1    0   8G  0 part /
├─xvda127 259:0    0   1M  0 part
└─xvda128 259:1    0  10M  0 part /boot/efi
xvdf      202:80   0  20G  0 disk
├─xvdf1   202:81   0  10G  0 part /mnt
└─xvdf2   202:82   0   5G  0 part
[root@ip-172-31-19-205 ~]#
```