RAID 6

RAID 6: Striping with Dual Distributed Parity

- **Definition: RAID 6** stripes data across **four or more disks** and uses **two sets of parity blocks** distributed across the drives. This means RAID 6 can **tolerate the failure of any two disks** simultaneously.
- **Purpose: RAID 6** offers **greater fault tolerance than RAID 5** while maintaining a balance of **performance, efficiency, and reliability**.
- **Tradeoff:**
  - Can survive **2 disk failures**, making it more robust than RAID 5.
  - **Write performance is lower** than RAID 5 due to the extra parity calculation.
  - **Rebuild time** is longer but provides higher data protection.

Why RAID 6?

- **Efficient Redundancy:** Unlike RAID 1 or RAID 5, you get usable capacity of **(N – 2)** disks (e.g., 6×1TB = 4TB usable), while still protecting against **two simultaneous disk failures**.
- **Balanced Performance:**
  - **Reads:** Fast, like RAID 0 – data is striped across multiple disks.
  - **Writes:** Slower than RAID 5 – due to the need to calculate and store **dual parity** for each write operation

- **High Fault Tolerance:**
    - The array continues to operate even if **two disks fail**.
    - Dual parity blocks allow for full recovery of data from two failed disks, offering strong protection for critical systems.
- **Cost-Effective:**
    - More storage-efficient than mirroring (RAID 1), with better fault tolerance than RAID 5.
    - Ideal for **enterprise storage, backup systems, and read-heavy environments** where **data integrity is critical**.

RAID Storage Efficiency (Usable Space)

Raid level	Fault Tolerance	Usable Capacity	Efficiency
RAID 1	Can lose 1 disk	N = 50% (half the disks)	50%
RAID 5	Can lose 1 disk	(N - 1) disks	~75% (with 4 disks)
RAID 6	Can lose 2 disks	(N - 2) disks	~50–80% (depends on number of disks)

So, for RAID 6:

- Usable capacity = (N – 2) / N
- With 4 disks → (4 – 2) / 4 = 50%
- With 5 disks → (5 – 2) / 5 = 60%
- With 6 disks → (6 – 2) / 6 = 66.7%
- With 8 disks → (8 – 2) / 8 = 75%

To configure **RAID 6 (Striping with Dual Parity)** on Linux:

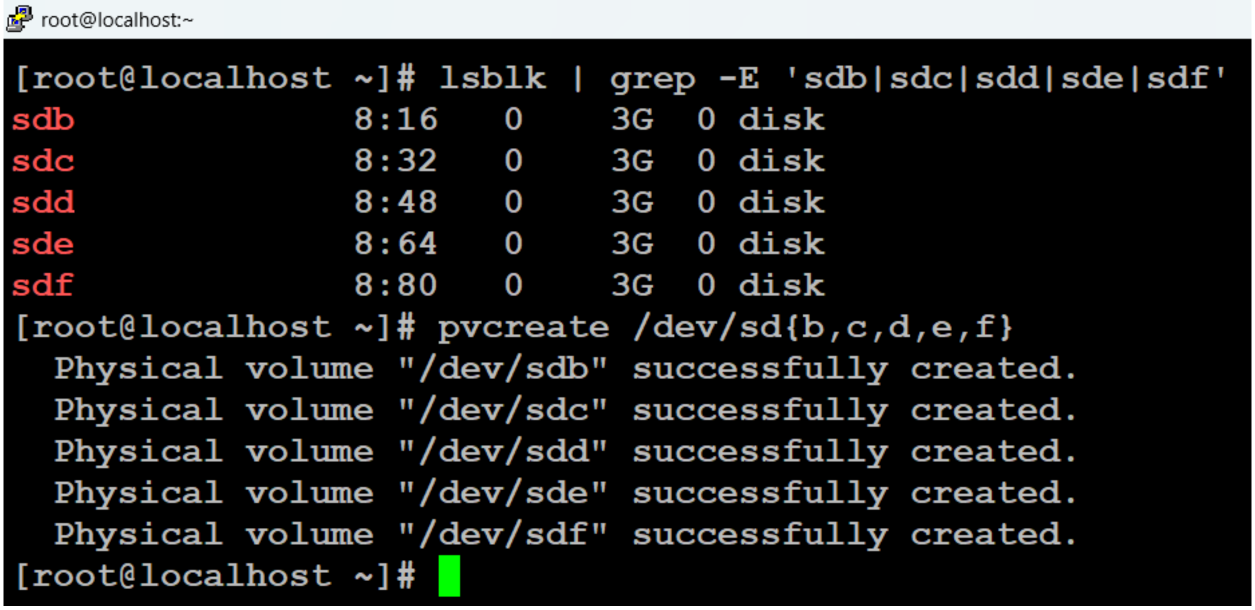
1. LVM (Logical Volume Manager)
2. MDADM (Multiple Device Administration)

RAID 6 Using LVM – Full Workflow

Step 1: Add New Disks & Initialize as PVs

After attaching your disks (e.g., /dev/sdb, /dev/sdc, /dev/sdd, /dev/sde, /dev/sdf), convert them into **LVM Physical Volumes**.

Command: **pvcreate /dev/sd{b,c,d,e,f}**



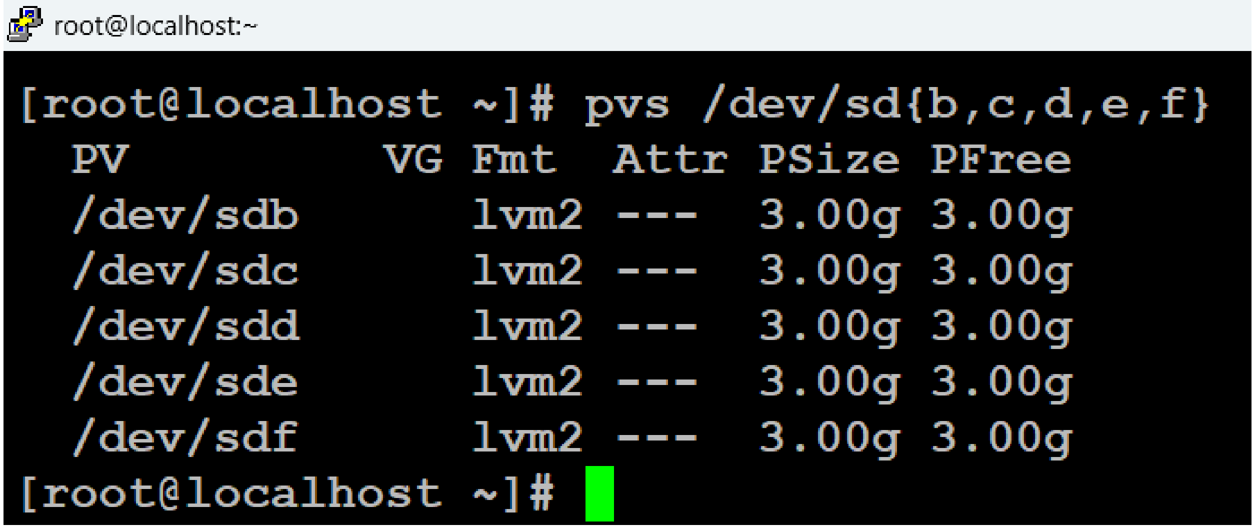
root@localhost:~

```
[root@localhost ~]# lsblk | grep -E 'sdb|sdc|sdd|sde|sdf'
sdb                 8:16   0    3G  0 disk
sdc                 8:32   0    3G  0 disk
sdd                 8:48   0    3G  0 disk
sde                 8:64   0    3G  0 disk
sdf                 8:80   0    3G  0 disk
[root@localhost ~]# pvcreate /dev/sd{b,c,d,e,f}
  Physical volume "/dev/sdb" successfully created.
  Physical volume "/dev/sdc" successfully created.
  Physical volume "/dev/sdd" successfully created.
  Physical volume "/dev/sde" successfully created.
  Physical volume "/dev/sdf" successfully created.
[root@localhost ~]#
```

Explanation:

This command initializes the raw disks and marks them as LVM-compatible **physical volumes (PVs)**. LVM cannot manage raw disks directly — they must be converted to PVs first.

Check with:

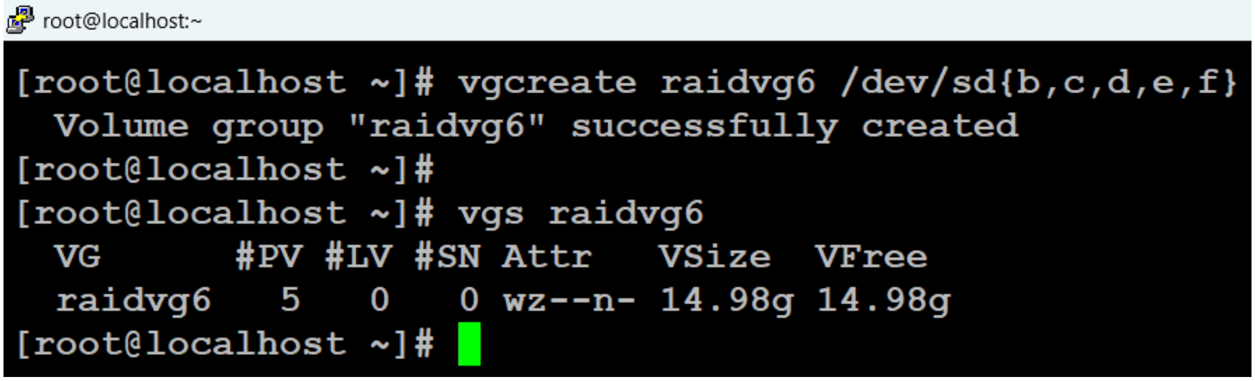


```
[root@localhost ~]# pvs /dev/sd{b,c,d,e,f}
  PV         VG Fmt  Attr PSize PFree
  /dev/sdb      lvm2 ---  3.00g 3.00g
  /dev/sdc      lvm2 ---  3.00g 3.00g
  /dev/sdd      lvm2 ---  3.00g 3.00g
  /dev/sde      lvm2 ---  3.00g 3.00g
  /dev/sdf      lvm2 ---  3.00g 3.00g
[root@localhost ~]#
```

Step 2: Create a Volume Group (VG)

Create a **Volume Group (VG)** that includes the five physical volumes

Command: vgcreate raidvg6 /dev/sd{b,c,d,e,f}



```
[root@localhost ~]# vgcreate raidvg6 /dev/sd{b,c,d,e,f}
  Volume group "raidvg6" successfully created
[root@localhost ~]#
[root@localhost ~]# vgs raidvg6
  VG      #PV #LV #SN Attr   VSize  VFree
  raidvg6   5   0   0 wz--n- 14.98g 14.98g
[root@localhost ~]#
```

Explanation:

This creates a Volume Group named **raidvg6** using the **five physical volumes**. All disks should be of similar size for optimal efficiency.

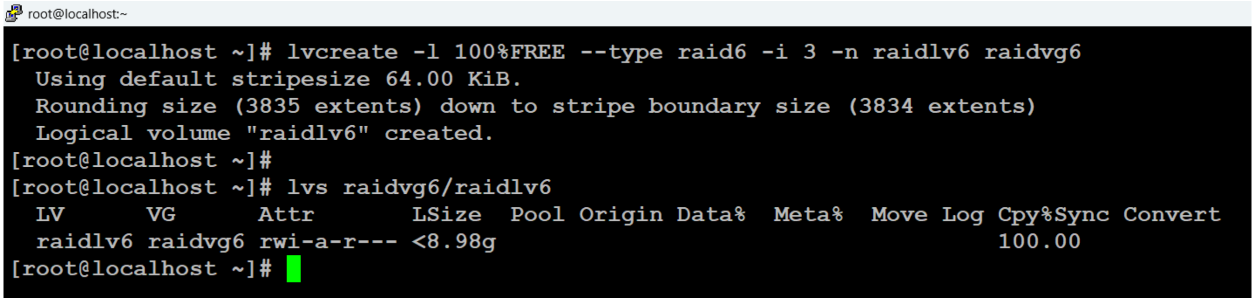
Step 3: Create a RAID 6 Logical Volume

To stripe across 5 PVs with dual parity:

Create a RAID 6 Logical Volume Using All Available Space

Command:

lvcreate -l 100%FREE --type raid6 -i 3 -n raidlv6 raidvg6



```
[root@localhost ~]# lvcreate -l 100%FREE --type raid6 -i 3 -n raidlv6 raidvg6
  Using default stripesize 64.00 KiB.
  Rounding size (3835 extents) down to stripe boundary size (3834 extents)
  Logical volume "raidlv6" created.
[root@localhost ~]#
[root@localhost ~]# lvs raidvg6/raidlv6
  LV      VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  raidlv6 raidvg6 rwi-a-r--- <8.98g                                    100.00
[root@localhost ~]#
```

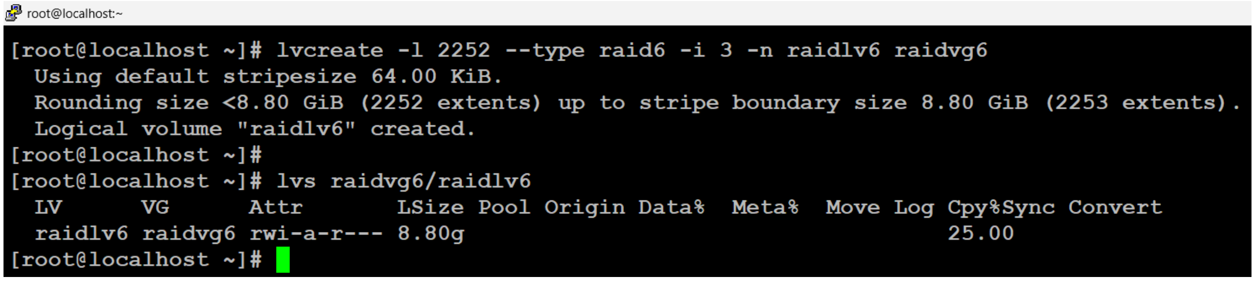
Description:

Creates a RAID 6 logical volume named raidlv6 that uses **100% of the free extents** available in the volume group raidvg6, with **3 data stripes** (total 5 disks: 3 data + 2 parity).

Create a RAID 6 Logical Volume Using Exactly 2252 Extents

Command:

lvcreate -l 2252 --type raid6 -i 3 -n raidlv6 raidvg6



```
[root@localhost ~]# lvcreate -l 2252 --type raid6 -i 3 -n raidlv6 raidvg6
  Using default stripesize 64.00 KiB.
  Rounding size <8.80 GiB (2252 extents) up to stripe boundary size 8.80 GiB (2253 extents).
  Logical volume "raidlv6" created.
[root@localhost ~]#
[root@localhost ~]# lvs raidvg6/raidlv6
  LV      VG      Attr       LSize Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  raidlv6 raidvg6 rwi-a-r--- 8.80g                                    25.00
[root@localhost ~]#
```

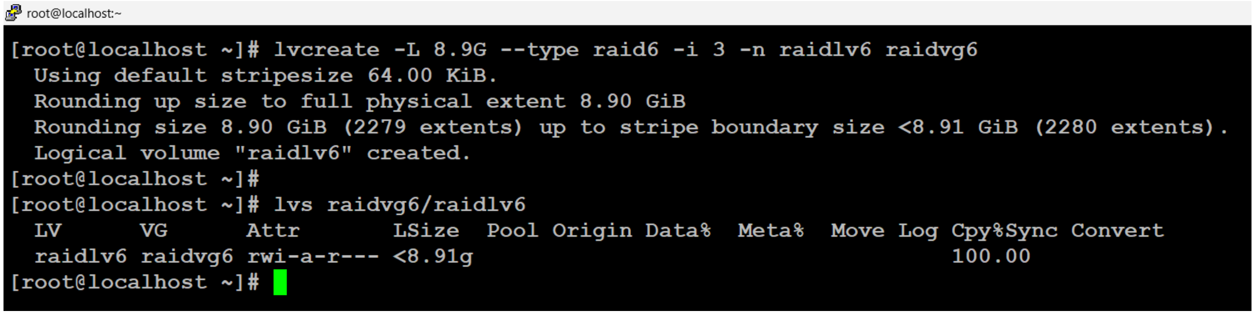
Description:

Creates a RAID 6 logical volume named raidlv6 with exactly **2252 logical extents** (~8.8 GB), with **3 data stripes** and automatic RAID 6 parity allocation.

Create a RAID 6 Logical Volume of Size 8.9 GB

Command

lvcreate -L 8.9G --type raid6 -i 3 -n raidlv6 raidvg6



```
[root@localhost ~]# lvcreate -L 8.9G --type raid6 -i 3 -n raidlv6 raidvg6
  Using default stripesize 64.00 KiB.
  Rounding up size to full physical extent 8.90 GiB
  Rounding size 8.90 GiB (2279 extents) up to stripe boundary size <8.91 GiB (2280 extents).
  Logical volume "raidlv6" created.
[root@localhost ~]#
[root@localhost ~]# lvs raidvg6/raidlv6
  LV      VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  raidlv6 raidvg6 rwi-a-r--- <8.91g                                    100.00
[root@localhost ~]#
```

Description:

Creates a RAID 6 logical volume named raidlv6 with a fixed size of **8.9 GB**, using **3 data stripes** and 2 parity stripes. Suitable when you want precise size control in GB.

Check with:

Command:

lvs raidvg6/raidlv6

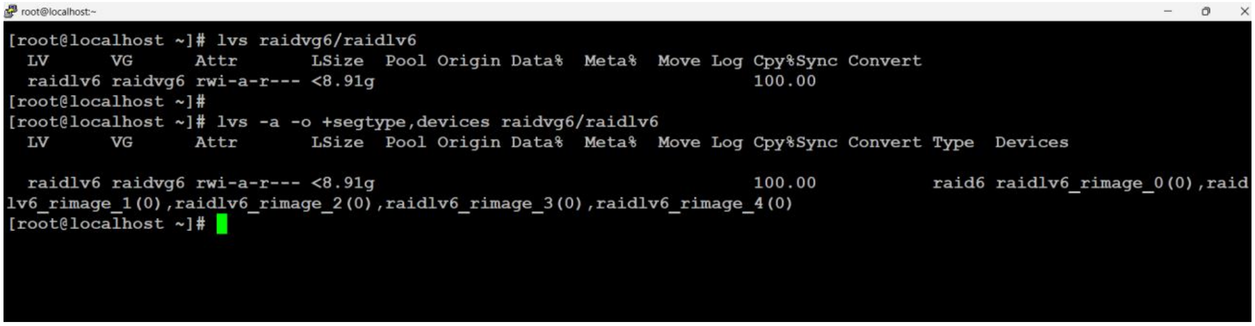
lvs -a -o +devices

lvs -a -o +segtype,devices raidvg6/raidlv6

lvs -a -o lv_name,vg_name,devices raidvg6/raidlv6

lvs --segment -o+segtype,stripes,devices raidvg6/raidlv6

lvdisplay -m /dev/raidvg6/raidlv6



```
[root@localhost ~]# lvs raidvg6/raidlv6
  LV      VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  raidlv6 raidvg6 rwi-a-r--- <8.91g                                    100.00
[root@localhost ~]#
[root@localhost ~]# lvs -a -o +segtype,devices raidvg6/raidlv6
  LV      VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert Type  Devices

  raidlv6 raidvg6 rwi-a-r--- <8.91g                                    100.00           raid6 raidlv6_rimage_0(0),raid
lv6_rimage_1(0),raidlv6_rimage_2(0),raidlv6_rimage_3(0),raidlv6_rimage_4(0)
[root@localhost ~]# 
```

Step 4: Format & Mount the LV

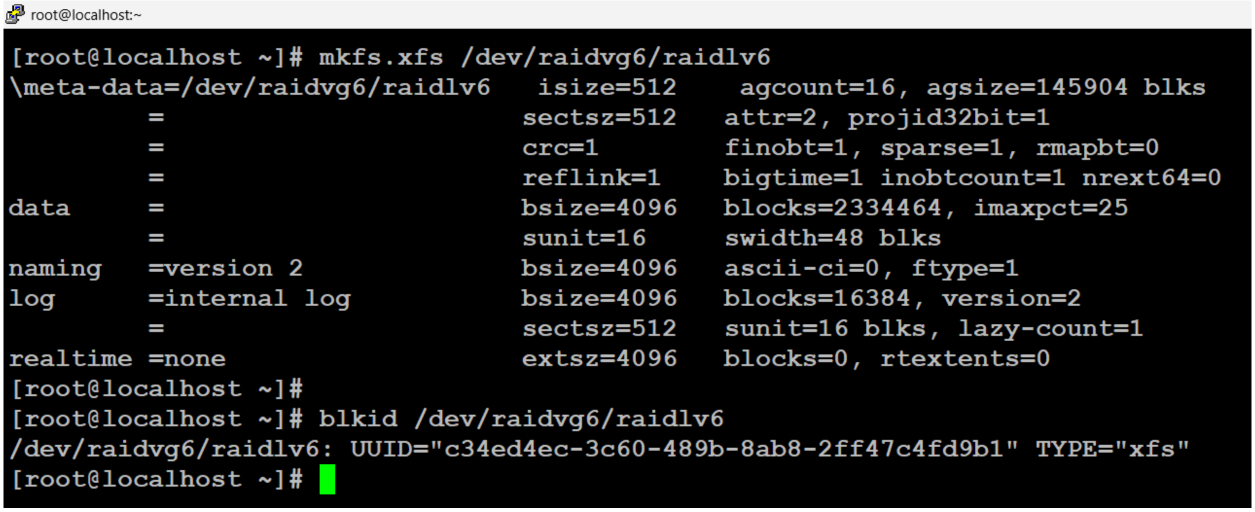
Commands:

mkfs.xfs /dev/raidvg6/raidlv6

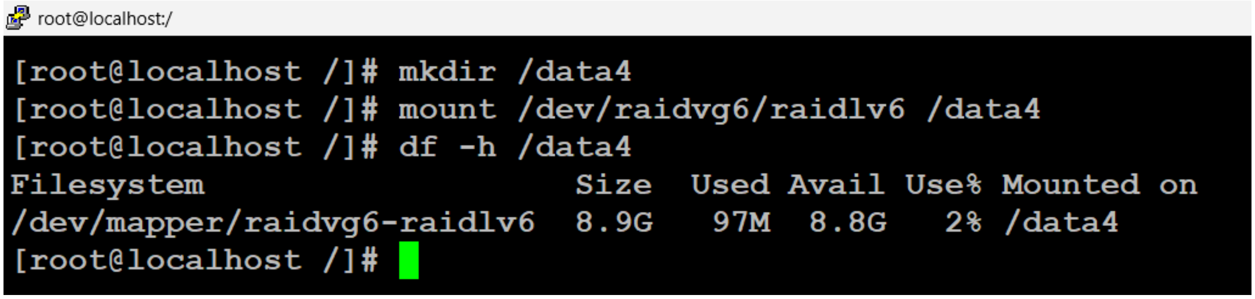
mkdir /data4

mount /dev/raidvg6/raidlv6 /data4

df -h /data4



```
[root@localhost ~]# mkfs.xfs /dev/raidvg6/raidlv6
\meta-data=/dev/raidvg6/raidlv6   isize=512    agcount=16, agsize=145904 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1 nrext64=0
data     =                       bsize=4096   blocks=2334464, imaxpct=25
         =                       sunit=16     swidth=48 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=16 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@localhost ~]#
[root@localhost ~]# blkid /dev/raidvg6/raidlv6
/dev/raidvg6/raidlv6: UUID="c34ed4ec-3c60-489b-8ab8-2ff47c4fd9b1" TYPE="xfs"
[root@localhost ~]#
```



```
[root@localhost /]# mkdir /data4
[root@localhost /]# mount /dev/raidvg6/raidlv6 /data4
[root@localhost /]# df -h /data4
Filesystem                   Size  Used Avail Use% Mounted on
/dev/mapper/raidvg6-raidlv6  8.9G   97M  8.8G   2% /data4
[root@localhost /]#
```

Explanation:

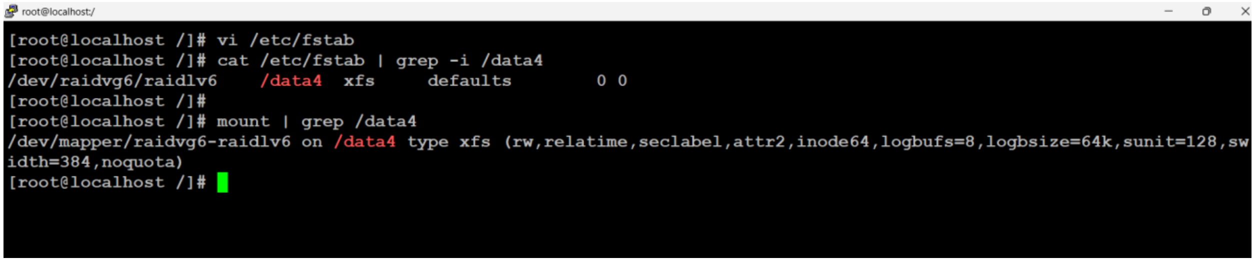
- mkfs.xfs = Formats the LV with the XFS file system (you can use ext4 if preferred)
- mkdir = Creates a mount point
- mount = Mounts the logical volume to the desired directory
- df -h = Verifies the mounted volume and available space

Step 5: Persist the Mount in /etc/fstab

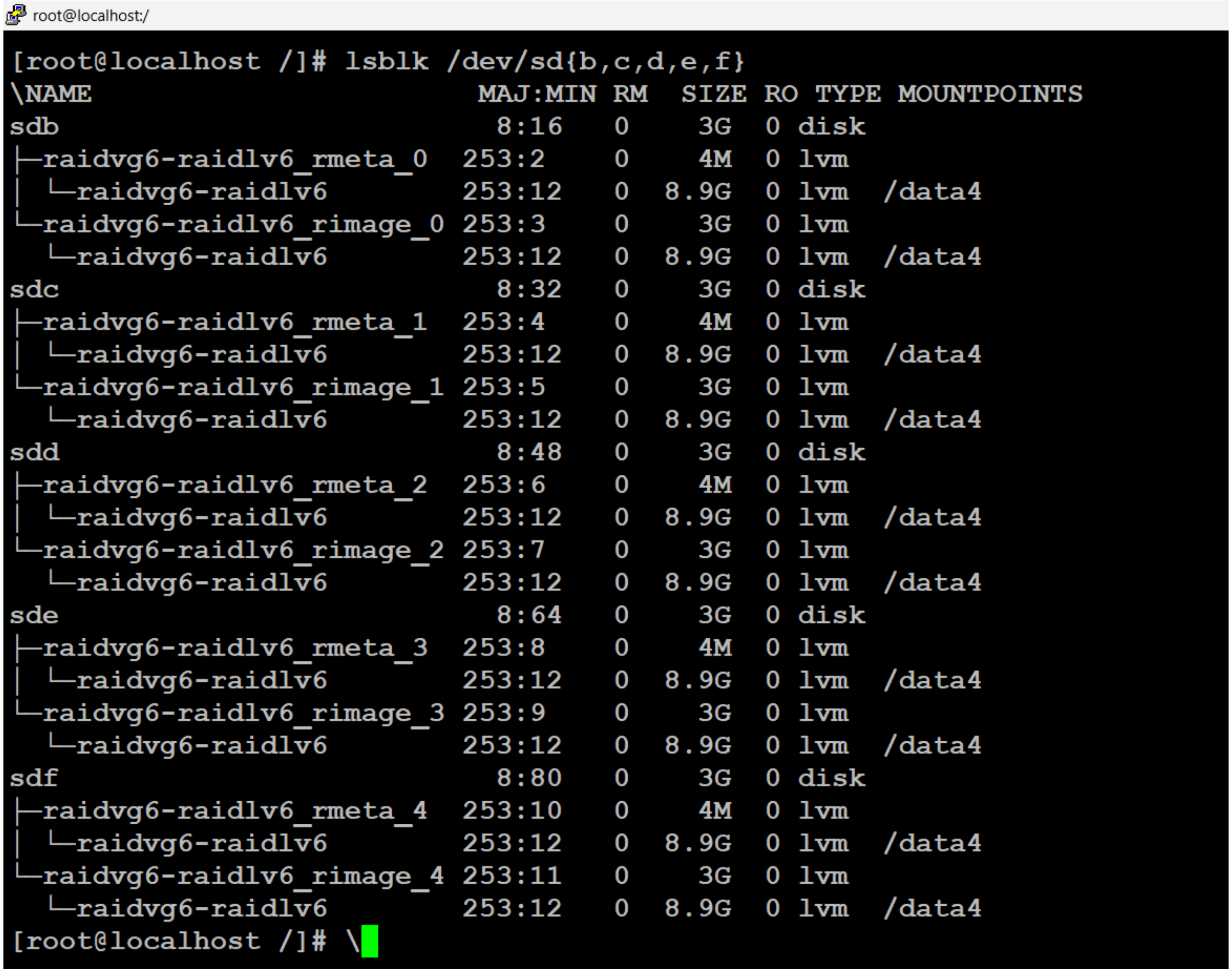
Ensure it auto-mounts after reboot:

Command:

```
vi /etc/fstab
cat /etc/fstab | grep -i /data4
mount | grep /data4
```



```
[root@localhost /]# vi /etc/fstab
[root@localhost /]# cat /etc/fstab | grep -i /data4
/dev/raidvg6/raidlv6    /data4  xfs     defaults        0 0
[root@localhost /]#
[root@localhost /]# mount | grep /data4
/dev/mapper/raidvg6-raidlv6 on /data4 type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=64k,sunit=128,sw
idth=384,noquota)
[root@localhost /]#
```



```
[root@localhost /]# lsblk /dev/sd{b,c,d,e,f}
\NAME                       MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sdb                          8:16   0    3G  0 disk
├─raidvg6-raidlv6_rmeta_0  253:2    0    4M  0 lvm
│ └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
└─raidvg6-raidlv6_rimage_0 253:3    0    3G  0 lvm
  └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
sdc                          8:32   0    3G  0 disk
├─raidvg6-raidlv6_rmeta_1  253:4    0    4M  0 lvm
│ └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
└─raidvg6-raidlv6_rimage_1 253:5    0    3G  0 lvm
  └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
sdd                          8:48   0    3G  0 disk
├─raidvg6-raidlv6_rmeta_2  253:6    0    4M  0 lvm
│ └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
└─raidvg6-raidlv6_rimage_2 253:7    0    3G  0 lvm
  └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
sde                          8:64   0    3G  0 disk
├─raidvg6-raidlv6_rmeta_3  253:8    0    4M  0 lvm
│ └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
└─raidvg6-raidlv6_rimage_3 253:9    0    3G  0 lvm
  └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
sdf                          8:80   0    3G  0 disk
├─raidvg6-raidlv6_rmeta_4  253:10   0    4M  0 lvm
│ └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
└─raidvg6-raidlv6_rimage_4 253:11   0    3G  0 lvm
  └─raidvg6-raidlv6        253:12   0  8.9G  0 lvm  /data4
[root@localhost /]# \
```

Final Notes:

- **RAID 6 with LVM provides redundancy, fault tolerance, and decent performance.**

- **Can tolerate two simultaneous disk failures.**

- **Write speed is slightly slower than RAID 5 due to dual parity calculations.**

- **Read speed remains fast thanks to striping.**

- **A modern and flexible solution for critical systems that demand both high availability and efficient storage.**