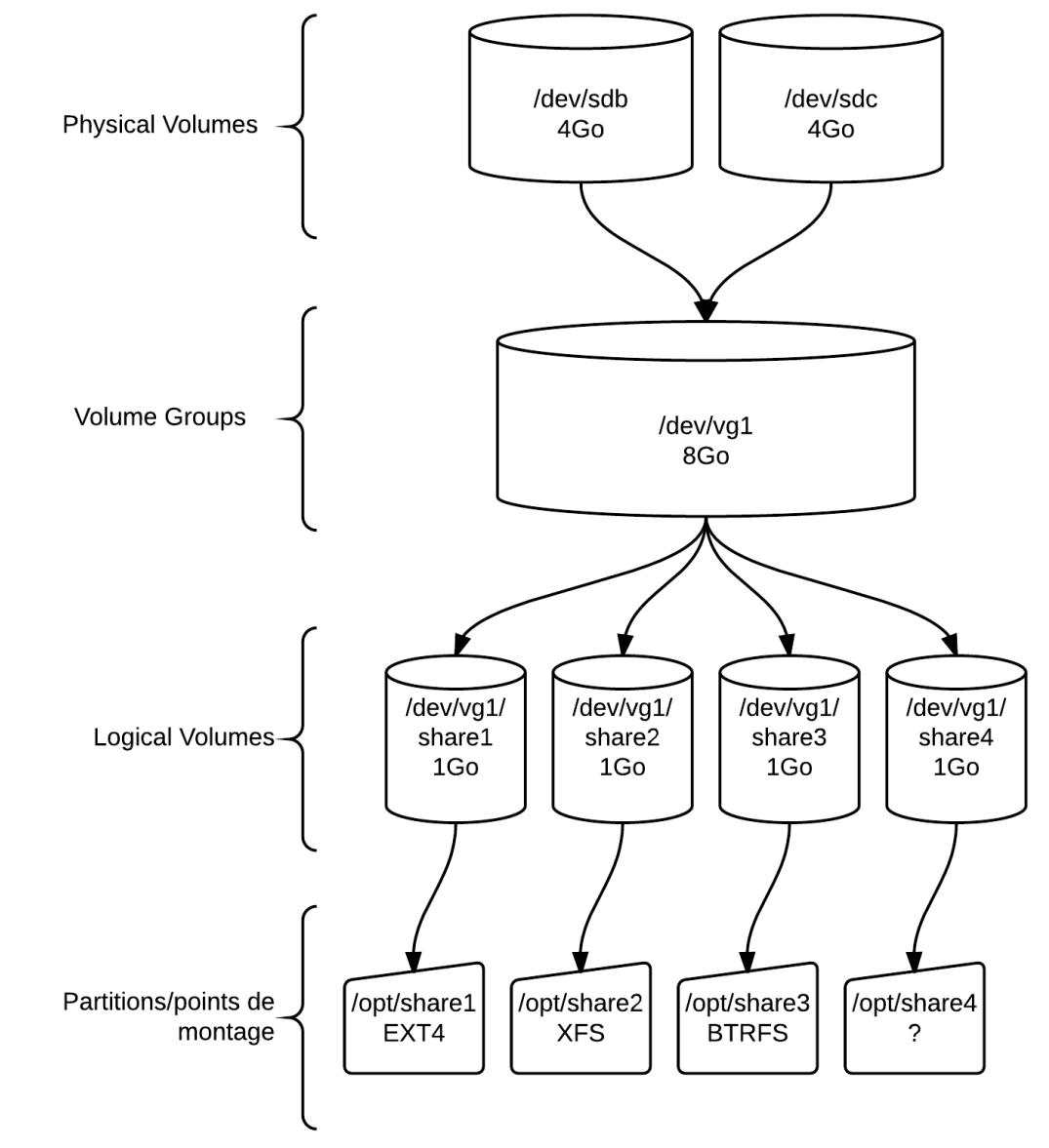


# LVM:

LVM stands for **Logical Volume Manager**. It is a system of managing logical volumes, or filesystems, in Linux. LVM provides a more flexible way of managing disk storage compared to traditional partitioning methods.

## **LVM Architecture:**



## ⊗ Key Concepts in LVM:

### 1. Physical Volume (PV)

- The actual hard drive or partition (e.g., `/dev/sdb`) that is initialized for LVM.
- Created using: `pvccreate /dev/sdb`

### 2. Volume Group (VG)

- Combines multiple physical volumes into one large pool of storage.
- Created using: `vgcreate my_vg /dev/sdb`

### 3. Logical Volume (LV)

- Created from a volume group. This is what you format with a file system and mount.
- Created using: `lvcreate -L 10G -n my_lv my_vg`

### 4. File System

- You can format the logical volume with a filesystem like ext4 using:  
`mkfs.ext4 /dev/my_vg/my_lv`

### 5. Mounting

- Mount the LV to access it:  
`mount /dev/my_vg/my_lv /mnt`

## ⊗ LVM Workflow Example

## ☒ Step 1: Add a New Disk in VMware

1. Open VMware Workstation, right-click the VM, and choose Settings.
2. Click Add > Hard Disk.
3. Choose:
  - **Create a new virtual disks**
  - **Set the size (e.g., 2 GB, 3 GB)**
  - **Choose SCSI or SATA as per your config (SCSI is recommended for LVM).**
4. Verify Disk Detection in Linux:
  - **lsblk**
  - **lsblk | grep -E 'sdb|sdc'**

## □ Step 2: Create Physical Volume (PV)

```
root@localhost:~ [root@localhost ~]# lsblk | grep -E 'sdb|sdc'  
sdb              8:16    0      3G  0 disk  
sdc              8:32    0      2G  0 disk  
[root@localhost ~]# [root@localhost ~]# pvcreate /dev/sd{b,c}  
Physical volume "/dev/sdb" successfully created.  
Physical volume "/dev/sdc" successfully created.  
[root@localhost ~]# [root@localhost ~]# █
```

```
[root@localhost ~]#
```

```
[root@localhost ~]# pvs
  PV          VG  Fmt Attr PSize  PFree
  /dev/sda3    rhel lvm2 a--  18.41g    0
  /dev/sdb      lvm2 ---   3.00g 3.00g
  /dev/sdc      lvm2 ---   2.00g 2.00g
[root@localhost ~]#
```

### Explanation:

This initializes the specified **raw disks** (`/dev/sdb` and `/dev/sdc`) as **Physical Volumes (PVs)** for use with **LVM (Logical Volume Manager)**.

## □ Step 3: Create Volume Group (VG)

```
[root@localhost ~]#
```

```
[root@localhost ~]# vgcreate appvg /dev/sdb /dev/sdc
  Volume group "appvg" successfully created
[root@localhost ~]#
[root@localhost ~]# vgs
  VG #PV #LV #SN Attr   VSize  VFree
  appvg   2   0   0 wz--n-  4.99g 4.99g
  rhel    1   2   0 wz--n- 18.41g    0
[root@localhost ~]#
```

### Explanation:

This creates a **Volume Group** named **appvg** using the two physical volumes `/dev/sdb` and `/dev/sdc`. You can later allocate **Logical Volumes** from this group.

## ☒ Step 4: Create Logical Volume (LV)

```
[root@localhost ~]# lvcreate -L 1G -n lv01 appvg
Logical volume "lv01" created.
[root@localhost ~]#
[root@localhost ~]# lvs
  LV   VG Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv01 appvg -wi-a----  1.00g
    root rhel -wi-ao---- 16.41g
    swap rhel -wi-ao----  2.00g
[root@localhost ~]#
```

### Explanation:

This creates a **Logical Volume** named **lv01** of size 4GB inside the **appvg Volume Group**. You can now format it and use it for file storage.

## □ Step 5: Format the Logical Volume

```
[root@localhost ~]# mkfs.xfs /dev/appvg/lv01
meta-data=/dev/appvg/lv01              isize=512      agcount=4, agsize=65536 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=262144, 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@localhost ~]#
```

```
[root@localhost ~]# blkid | grep appvg
/dev/mapper/appvg-lv01: UUID="6db719e1-85dd-4591-90f2-4ad10d204d15"
TYPE="xfs"
```

### Explanation:

Formats the **logical volume** with the **XFS file system**. You can use other file systems too, like **ext4**.

## □ Step 6: Mount the Logical Volume

```
root@localhost:~# [root@localhost ~]# mkdir /data1 [root@localhost ~]# [root@localhost ~]# mount /dev/appvg/lv01 /data1 [root@localhost ~]# [root@localhost ~]# df -h | grep data1 /dev/mapper/appvg-lv01 960M 39M 922M 5% /data1 [root@localhost ~]#
```

### Explanation:

- **mkdir /mnt/data1**: Creates a mount point (folder) where the volume will be accessed.
- **mount**: Mounts the LV to the mount point so you can read/write files there.
- **df-h** : Shows how much space is used and available on the mounted volume. The **-h** option shows human-readable sizes like GB/MB.

## □ Make the Mount Persistent

Add this line to **/etc/fstab**:

```
root@localhost:~# [root@localhost ~]# vi /etc/fstab [root@localhost ~]# [root@localhost ~]# cat /etc/fstab | grep /data1 UUID=6db719e1-85dd-4591-90f2-4ad10d204d15 /data1 xfs defaults 0 0 [root@localhost ~]#
```

### Explanation:

This line tells the system to automatically mount the **LV** at boot using its **UUID**.

## Test it:

```
root@localhost:~
```

```
[root@localhost ~]# mount -a  
[root@localhost ~]# █
```

### Explanation:

**mount -a:** Re-reads **/etc/fstab** and mounts everything.

## Differences:

Feature	LVM (Logical Volume Management)	Normal Partitions
Storage	Pools multiple physical volumes (PVs) into a volume group (VG), which can then be carved into logical volumes (LVs). LVs can span multiple PVs	Uses individual physical partitions on a single disk. Storage space is divided into fixed, unchangeable partitions.
Flexibility	Highly flexible. Allows dynamic resizing of LVs while the system is running. LVs can be moved to different PVs within a VG.	Less flexible. Resizing requires offline operations or the use of specialized tools. Partitions are typically fixed in size.
Disk Management	Supports multiple physical disks within a single volume group, allowing for flexible storage management and redundancy	Generally limited to individual disks. Each physical disk is typically partitioned separately.
Management	Requires managing LVM metadata, including PVs, VGs, and LVs. Can be more complex than managing normal partitions	Simpler to manage as each partition is typically treated as a separate unit.

## Conclusion

LVM is a powerful disk management tool every Linux admin and DevOps engineer should know. It enables dynamic volume resizing, easier backups via snapshots, and flexible disk usage.

Whether you're working in local VMs or cloud environments like AWS, mastering LVM will help you manage storage effectively and professionally.

## ⊕ Benefits of Using LVM

- **Resizing Volumes:** Easily resize logical volumes (expand or shrink).
- **Snapshots:** Take snapshots of volumes for backup or testing.
- **Flexibility:** Add more disks to VG and extend LVs as needed.
- **Better Disk Utilization:** Use the full capacity of multiple disks efficiently.