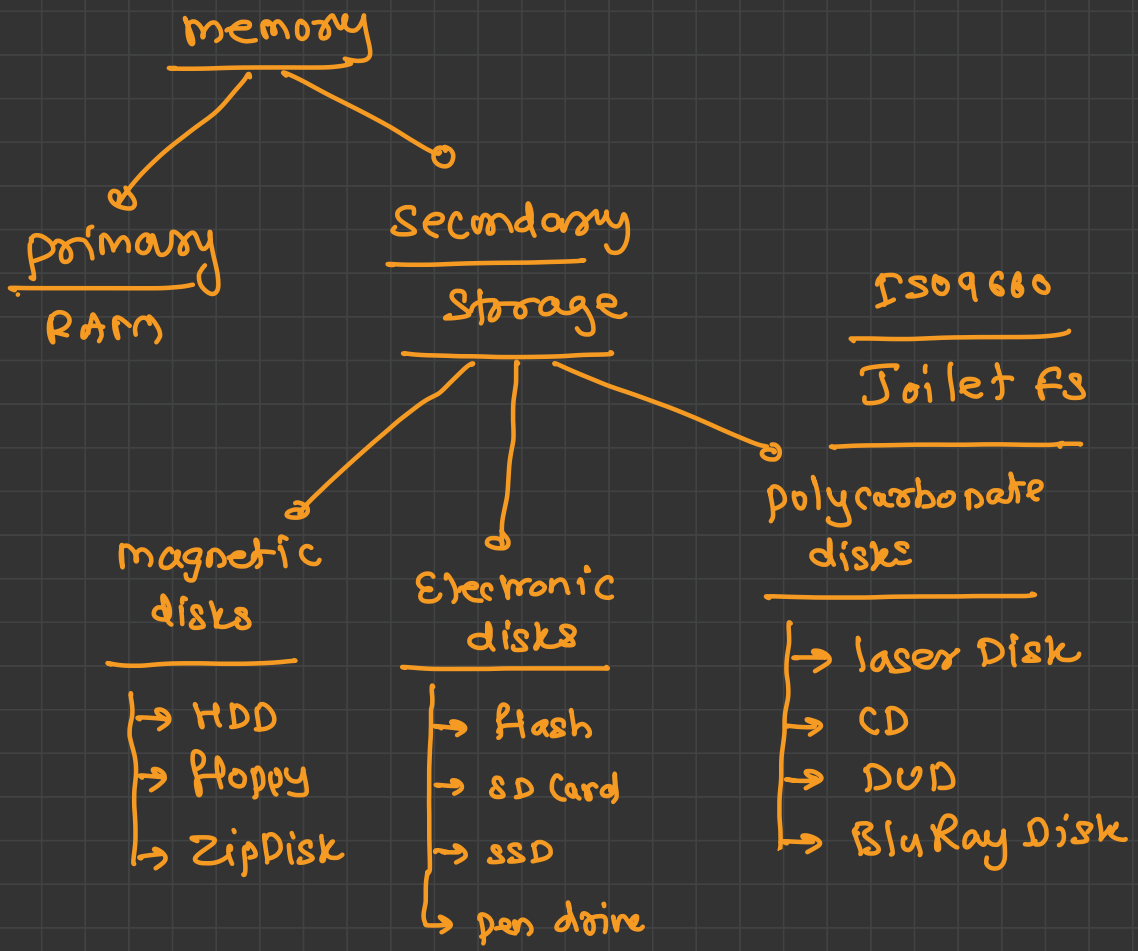
1] disk mgmt → add new disk, format (partition)

2] configure virtual disks → RAID / LVM ***

Storage Management


memory
primary
RAM
Secondary
Storage
magnetic
disks
HDD
Floppy
ZipDisk
Electronic
disks
Flash
SD Card
SSD
pen drive
ISO9660
Joilet FS
polycarbonate
disks
laser Disk
CD
DVD
BluRay Disk

Administering Storage

- Storage management involves planning, understanding hardware, and using many tools to accomplish storage goals
- Sysadmins must know what types of hardware are available and what type will work with the Linux system in question
- They must also know the performance requirements for data stored on any drives
- Direct-attached storage disks are managed as single entities with traditional partitions or can be combined into logical volumes for more flexible storage space
- Network-based storage, such as NAS, SAN, and cloud technologies, may also be used
- Finally, sysadmins must understand the available Linux tools for troubleshooting and performance monitoring

Understanding Storage

- Storage capacity for user data, databases, logs, configuration files, operating system and application executables, and other resources are of major concern to sysadmins. Capacity is only one part of the issue, however.
- Related is the ability to maintain cost-effective, reliable, and secure storage media that is accessible to users and services
- Some examples of storage media types include:
  - Hard disk drive (HDD): Spinning magnetic disks, usually inexpensive, large, and relatively slow
  - Solid-state disk (SSD): Flash memory storage, usually expensive, fast, small, and shorter -lived
  - Universal Serial Bus (USB): Connection protocol for various external devices, including storage drives
  - Thumb drive: Removable storage media, usually connects via USB
  - External: Removable storage disk, usually connects via USB

interfacing

Advanced Technology

- PATA → Parallel AT attached disks
  IDE → Integrated Device Electronics
- SATA → Serial ATA
- USB → Universal Serial Bus
  - A →
  - B →
  - C →

  1 – 100 Mbps
  2 – 400 Mbps
  3.x → 5 – 40 Gbps
- eSATA → enhanced SATA
- firewire 400 / 800
- M.2 → 6 – 8 Gbps
- thunderbolt → 40 – 50 Gbps
- PCIe → extended PCI
  Peripheral Component Interface

Common device files

Device	Description
/dev/sda	A hard disk that uses the SCSI driver. Used for SCSI and SATA disk devices. Common on physical servers but also in VMware virtual machines.
/dev/nvme0n1	The first hard disk on an NVM Express (NVMe) interface. NVMe is a server-grade method to address advanced SSD devices. Note at the end of the device name that the first disk in this case is referred to as *n1* instead of *a* (as in common with the other types).
/dev/hda	The (legacy) IDE disk device type. You will seldom see this device type on modern computers.
/dev/vda	A disk in a KVM virtual machine that uses the virtio disk driver. This is the common disk device type for KVM virtual machines.
/dev/xvda	A disk in a Xen virtual machine that uses the Xen virtual disk driver. You see this when installing RHEL as a virtual machine in Xen virtualization. RHEL 8 cannot be used as a Xen hypervisor, but you might see RHEL 8 virtual machines on top of the Xen hypervisor using these disk types.

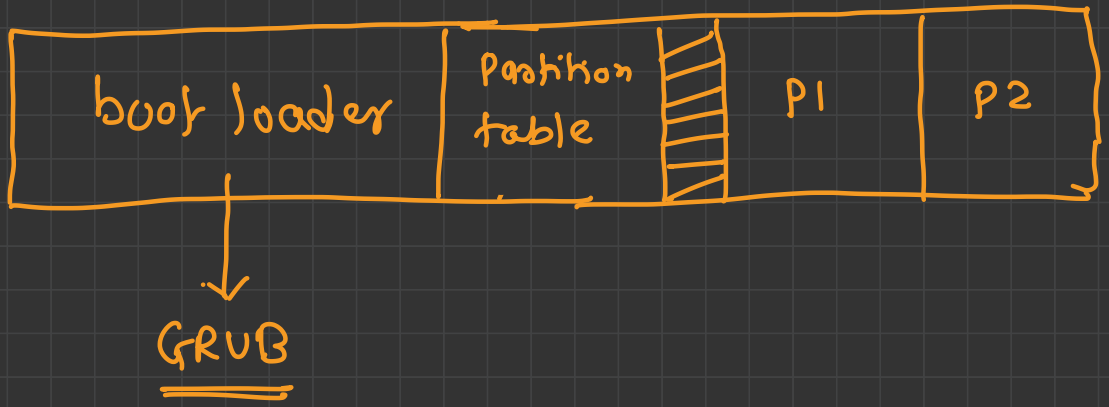
Type of Storage Design

- Stored data is usually managed via one of three different designs: file, block, and object storage
- While these don't tend to impact Linux on a day-to-day basis, it is important to recognize some basic differences
- File storage : EFS → NFS
  - It is the common approach to storing data on local drives, with data organized into discrete files stored in directories
  - These directories are organized in a hierarchical structure
  - Data is easy to retrieve and change, but file storage does not scale well: it must be scaled out (horizontal) rather than scaled up (vertical)
- Block storage : EBS
  - It is a good choice for large quantities of unstructured data stored across multiple platforms and storage types
  - Commonly used with SAN technologies, it is quick, reliable, and efficient
- Object storage : S3
  - It is cost-effective and very scalable but best for static data because it's difficult to modify data once written
  - This makes it particularly unsuited for databases

Storage Interfaces

- Internal storage devices use different protocols and interfaces to communicate with the rest of the system
- These designs vary by speed, communication type, and scalability
- Various solid-state drives, hard disk drives, and flash storage use these interfaces
- Serial Advanced Technology Attachment (SATA)
  - It is a common inexpensive storage media that uses a serial connection to communicate with the motherboard
  - This is the slowest of the three options discussed here but the least expensive
  - It is a good all-around solution for most systems
- Small Computer Systems Interface (SCSI)
  - It tends to be expensive with less capacity than most SATA drives but makes up for this by being very fast
  - For servers, RPMs of 10,000 and 15,000 are common, while desktop and laptop drives might spin at 7,200 or 5,400 RPM
  - Multiple SCSI drives can be attached to the same chain, making their scalability far more flexible than SATA
- Serial Attached SCSI (SAS)
  - It is an upgraded SCSI design with larger capacities, faster transfer speeds, and greater flexibility
  - However, these benefits come with a higher cost
  - The cost may be well worth it for mission-critical servers

MBR → bootabl



Partitions

partition Table
type / start / end / total / available

- Storage drives are divided into partitions for more efficient and organized use
- MBR
  - The master boot record (MBR) is the first physical sector on a storage drive and a type of partition structure
  - The MBR boot sector contains the boot loader that loads the operating system into memory
  - It also contains the partition table of the storage drive
  - MBR determines what sectors are available to each partition, as well as which partitions are bootable and which are not
  - It has three major disadvantages:
    - The maximum storage space of an MBR-partitioned drive is two terabytes
    - MBR-partitioned drives can have a maximum of four primary partitions
    - The boot data is stored all in one sector, which increases the risk of corruption

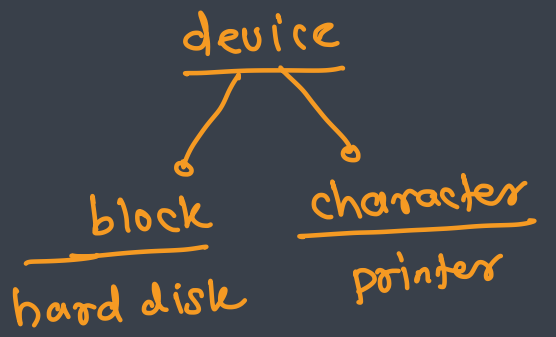
- GPT

Globally Unique Identifier

  - The GUID Partition Table (GPT) is a successor to MBR that makes up for the latter's shortcomings
  - Like MBR, it is a partition structure, but it employs a more modern design and is part of the UEFI standard
  - Every partition on a drive is assigned a globally unique identifier—a GUID—to distinguish it from every other partition on (theoretically) every drive
  - The storage space and partition number maximums are so large that they are not currently achievable, and any limitations are going to be imposed by the file system type or operating system kernel, rather than GPT itself
  - GPT also has the advantage of storing its boot data in multiple locations on a drive to enhance redundancy
  - If the primary location is corrupted, GPT can leverage one of the other copies to restore the boot data

Partitioning Commands

- lsblk: used to list the block devices
- blkid: get the guid of a block device
- partprobe: probing the new partitions
- fdisk: used to partition the disk
- parted: used to partition the disk



Partitioning the Drive

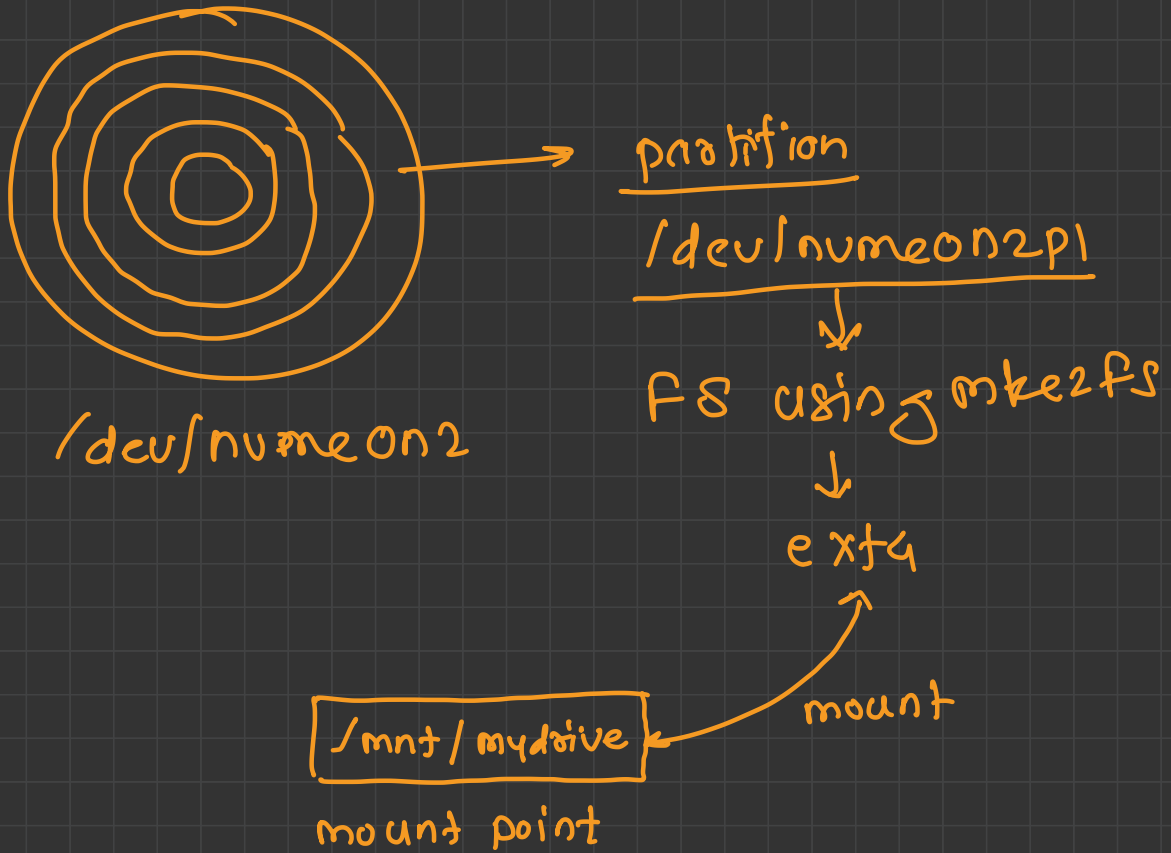
- Once you've installed the drive and it's been recognized by Linux, you must divide the storage capacity into usable space
- Creating partitions is the first step in organizing the storage space
- Considerations such as MBR vs. GPT must be taken into account, and then you'll partition the space using fdisk or parted
- After creating the partitions, update the system with partprobe and confirm the partitions in the /proc/partitions directory
- fdisk
  - Type fdisk <disk> to edit the partition table for the disk
  - Type m to display the menu of fdisk features
  - Type n to create a new partition
  - Type p to create a new primary partition, or type l to create a logical partition
  - Select Enter to start the partition at the first available sector
  - Type a value such as +10G to create a 10 GB partition
  - Type p to display the new partition information (note that no changes have been made at this point)
  - Type w to write (save) the changes to the partition table and exit fdisk
  - Type fdisk -l <disk> to confirm the partition information

Adding File System

- After creating the partitions on a new drive, you must add a filesystem to organize the storage of the actual data
- Different filesystems have different benefits, and there are many of them
- In general, most Linux distributions use either the ext4 or XFS filesystems.
- The ext4 Filesystem
  - The ext4 filesystem is the one of the oldest file systems
  - It provides journaling and is recognized by most distributions.
  - To place the ext4 filesystem on the second partition on the new disk, type:
    - mkfs.ext4 <partition>
  - The mkfs also recognizes a slightly different syntax.
    - mkfs -t ext4 <partition>
- The XFS Filesystem
  - While the ext4 filesystem has been the default for many Linux distributions, it is very common to find newer systems relying on XFS
  - XFS has many modern advantages over ext4 and relatively few disadvantages
  - XFS recognizes a larger partition size
  - XFS recognizes a larger file size
  - XFS avoids inode exhaustion

Manually Mounting FS

- Now that you have installed the drive, created the partitions, and placed the filesystem, the storage capacity is ready for use
- mount
  - The term for attaching storage to the FHS is referred to as "mounting," and the command is mount
  - To mount the partition, first create a directory
  - The directory is referred to as a mount point
  - Use the mount command to mount a fs to a directory
  - > sudo mount <partition> <mount point>
- findmnt
  - Find all the mounted file system
- umount
  - The logical reverse of this is to detach or unmount the storage area, and the related command is umount
  - Use umount command to unmount the mounted partition
  - > umount <partition>


/dev/nvme0n2
partition
/dev/nvme0n2p1
FS using mke2fs
ext4
/mnt/mydrive
mount
mount point

Testing the storage

- There are two approaches to testing
  - use reporting tools
  - actually storing data in the new space
- The df and du commands report storage information about specified partitions or directories
- Commands such as touch, mv, and cp place data in the new storage location

Making Mounts Persistent

- /etc/fstab is the main configuration file to persistently mount partitions
- /etc/fstab content is used to generate system mounts by the **system-fstab-generator** utility

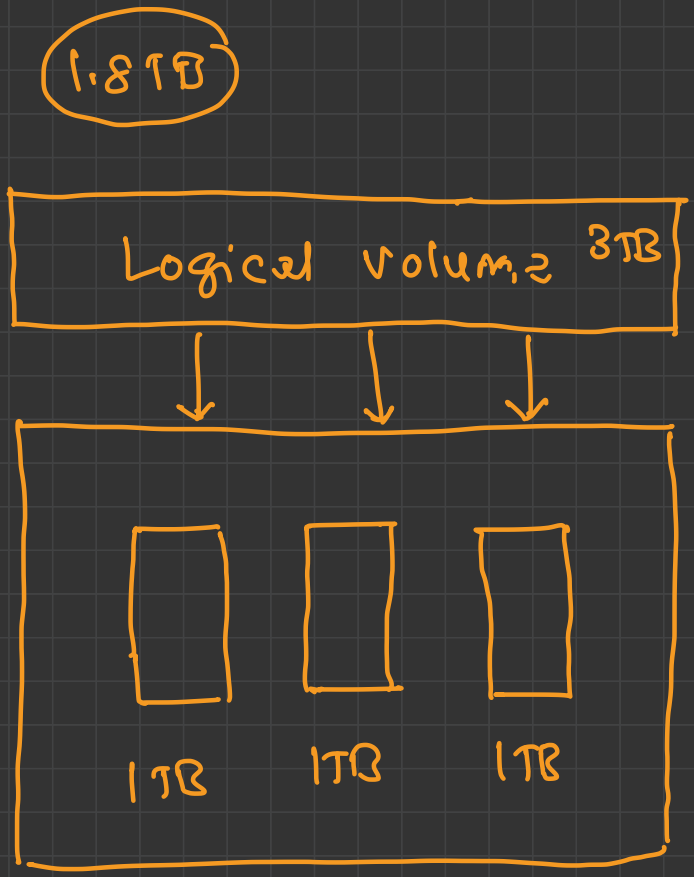
```
/dev/mapper/fedora_fedora-root /                       xfs     defaults        0 0
```

- Every entry contains:
  - Column 1: device to be mounted
  - Column 2: mount point
  - Column 3: file system
  - Column 4: extra mount options
  - Column 5 and 6: always 0
- To update systemd, make sure to use **systemctl daemon-reload** after editing /etc/fstab
- Use mount –a to check the entries are working from fstab file
- Note: Do not persist mount on /mnt as it is used to mount a FS temporarily

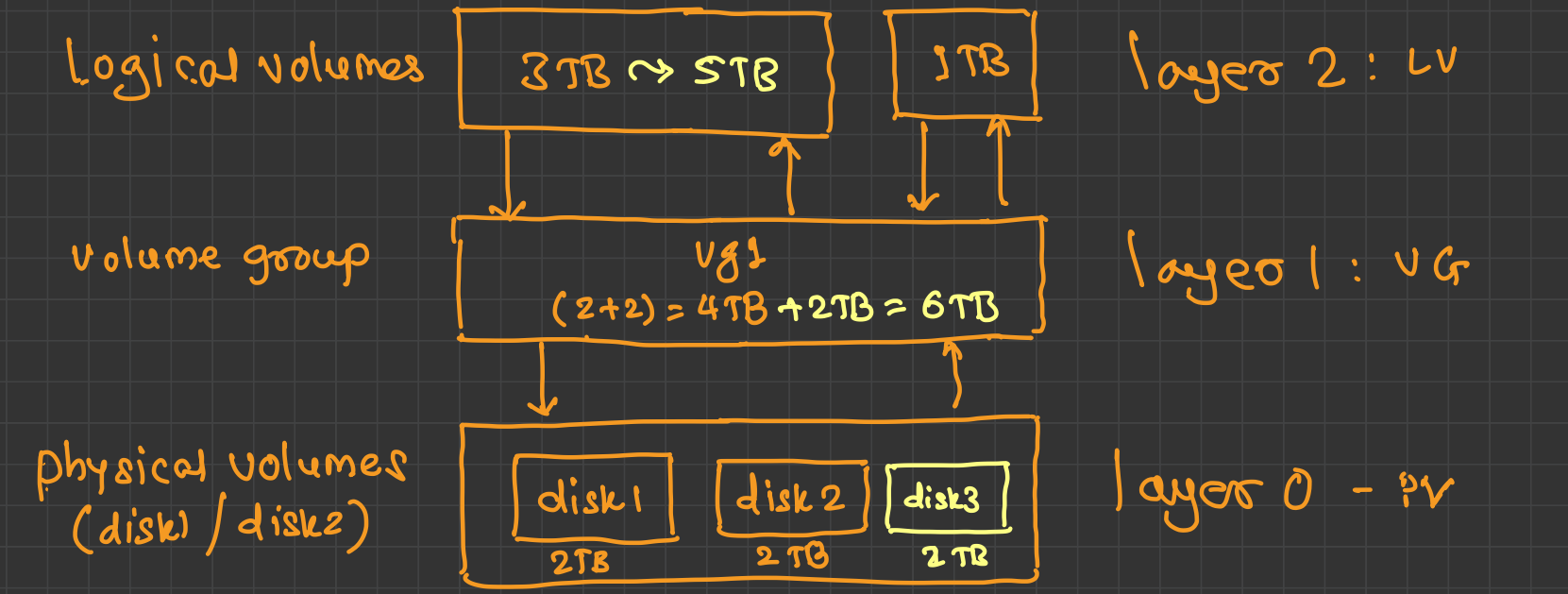
LVM

Volume

disk space



Similar to partitions



LVM

- It's common to add a single storage disk to a system to provide additional capacity
- The disk is then partitioned and its capacity used
- The Logical Volume Manager configuration is an alternative to this traditional partitioning scheme
- The LVM configuration is flexible and very scalable, often providing a better storage option than single storage disks
- Logical Volume Manager functions based on three configuration layers for storage
  - The first layer is one or more Physical Volumes (PV)
    - PVs are drives or partitions allocated to LVM
  - The next layer represents the aggregated storage capacity of any designated PVs that are added to the Volume Group (VG)
  - The top layer consists of one or more Logical Volumes (LV), which are treated by the system as if they were standard partitions
    - In reality, they are carved space from VGs that may store data on multiple PVs
- The /dev/mapper/ directory contains all of the logical volumes on the system that are managed by LVM
  - /dev/mapper/<volume group name>-<logical volume name>

LVM Advantages

- Dynamically create, delete, and resize volumes without having to reboot the system
- Day-to-day management of volumes is easier once everything is set up
- Map multiple logical volumes across multiple physical devices
- A logical volume can exceed the size of any one physical device (as long as it doesn't exceed the total size of devices in the volume group)
- Create virtual snapshots of each logical volume so you can quickly and easily revert a volume to a specific state

Commands for configuring LVM

- Physical Volume
  - pvcreate: creates new physical volume
  - pvscan: scans existing physical volumes
  - pvdisplay: displays attributes of selected physical volume
  - pvs: shows existing physical volumes
- Volume Group
  - vgcreate: creates mew volume group
  - vgscan: scans existing volume groups
  - vgdisplay: displays attributes of selected volume group
  - vgs: shows existing volume groups
- Logical Volume
  - lvcreate: created new logical volumes
  - lvscan: scans existing logical volumes
  - lvdisplay: displays attributes existing logical volumes
  - lvvs: shows existing logical volumes

RAID

RAID

- Individual storage disks are a single point of failure, putting data at risk of being lost or unavailable
- One way of mitigating this risk is to use redundant arrays of independent disks (RAID) or combinations of two or more disks to store data
- However, not all RAID standards mitigate the risk of lost data; some forms only provide performance benefits
- Types
  - Disk striping (RAID 0) relies on at least two disks. The disk partitions are divided into sections called stripes and data is written sequentially through the stripes. Because RAID 0 provides no fault tolerance, it is mainly used for performance benefits with application data that does not need to be preserved (such as caching).
  - Disk mirroring (RAID 1) duplicates data on two storage disks. This provides complete redundancy (100% of the data resides on each disk) but is a relatively inefficient use of storage capacity. RAID 1 usually results in faster reads than a single standalone disk.
  - Disk striping with parity (RAID 5) is a modification of RAID 0 that provides fault tolerance. Like RAID 0, stripes are created on the storage media, and data is distributed across the stripes. In addition to regular data, parity information is added to each disk. This parity information is used to recreate the missing data from any one failed storage drive. RAID 5 is particularly useful for file servers.