🐧 Introduction to Linux Shell 🖥️ 🏆

What is a Shell?

A **shell** is a **command-line interface (CLI)** that allows users to interact with the **operating system**by executing commands. It serves as a bridge 🏗️ between the **user** and the **kernel**.

🔍 Types of Shells

🏷️ **Shell Type**	📌 **Description**
Bourne Shell (sh)	The original Unix shell.
Bash (Bourne Again Shell)	🔥 Improved version of Bourne Shell, widely used in Linux.
C Shell (csh)	🏗️ Uses C-like syntax.
Korn Shell (ksh)	🎭 Combines features of Bourne and C Shell.
Z Shell (zsh)	⚡ Extended version of Bash with extra features.

🏗️ Basic Linux Commands

📂 File and Directory Management

🔧 **Command**	📌 **Description**
`ls`	📋 List files and directories.
`pwd`	📍 Show current directory.
`cd <directory>`	🚀 Change directory.
`mkdir <directory>`	📁 Create a new directory.

`rmdir <directory>`	🗑️ Remove an empty directory.
`rm <file>`	❌ Delete a file.
`rm -r <directory>`	❌ Delete a directory and its contents.
`cp <source> <destination>`	📤 Copy files or directories.
`mv <source> <destination>`	🔀 Move or rename files and directories.

📜 File Viewing

🔍 **Command**	📌 **Description**
`cat  <file>`	📖 Display file contents.
`less <file>`	📜 View file page by page.
`head <file>`	📄 Show the first 10 lines.
`tail <file>`	📑 Show the last 10 lines.

🔒 File Permissions and Ownership

🔧 **Command**	📌 **Description**
`chmod <permissions> <file>`	🛡️ Change file permissions.
`chown <user>:<group> <file>`	👤 Change file ownership.

⚙️ Process Management

🔍 **Command**	📌 **Description**
`ps`	📋 Show running processes.
`top`	📊 Display system resource usage.

`kill <PID>`	🔫 Terminate a process by its ID.
`pkill <process_name>`	🛑 Kill a process by name.

🖥️ System Information

📌 **Command**	📊 **Description**
`uname -a`	💻 Show system details.
`df -h`	💾 🧠Display disk usage.
`free -m`	Check memory usage.

🌐 Networking Commands

🌍 **Command**	📡 **Description**
`ping <host>`	📡 Check network connectivity.
`ifconfig / ip addr show`	🔎 View network interfaces.
`netstat -tulnp`	🔥 Show active network connections.
`curl <URL>`	🌍 Fetch data from a URL.

📚 Command Line Help

📖 Getting Help on Commands

🛠️ **Command**	📌 **Description**
`man <command>`	📕 Show manual for a command.
`info <command>`	📖 View detailed documentation.

`<command> --help`	📌 Quick command guide.
`whatis <command>`	📌 One-line command summary.

🕵️ Searching for Commands

🔍 **Command**	📌 **Description**
`apropos <keyword>`	🔎 Find commands related to a keyword.
`which <command>`	📌 Show full command path.

🐚 Bash Shell

🏆 What is Bash?

Bash (**Bourne Again Shell**) is the most commonly used **Linux shell**. It provides powerful **scripting**, **command history**, and **job control** features.

🚀 Bash Features

✅ **Command history** – Use `history` or arrow keys to navigate previous commands.
✅ **Auto-completion**– Press `Tab` to autocomplete commands.
✅ **Aliases**– Define shortcuts using `alias name='command'`.

📝 Bash Scripting Basics

✍️ Creating a Bash Script

1 **Create a new script file:**

```
nano script.sh
```

2 **Add script content:**

```
#!/bin/bash
echo "Hello, World!"
```

3 **Make it executable:**

```
chmod +x script.sh
```

4 **Run the script:**

```
./script.sh
```

📦 Variables in Bash

```
name="John"
echo "Hello, $name"
```

🔁 Conditional Statements

```
if [ "$name" == "John" ]; then
    echo "Name is John"
else
    echo "Name is not John"
fi
```

🔄 Loops in Bash

```
for i in {1..5}; do
    echo "Number: $i"
done
```

🖥️ Linux Core Concepts 🛠️

🔍 1. Linux Kernel

What is the Linux Kernel?

The **Linux Kernel** is the **core**of the operating system. It acts as a **bridge** between **hardware** and **user applications**, managing:

✅ **Processes**
✅ **Memory** 🧠
✅ **Devices** 🎧📱
✅ **File Systems**📁
✅ **Networking** 🌐

📌 Checking Kernel Information

Use these commands to check kernel details:

```
uname  -r       # Display kernel version
uname -a        # Show complete system information
cat /proc/version # Detailed kernel version info
```

📌 Updating the Kernel

To check and upgrade your kernel:

```
sudo apt update && apt list --upgradable | grep linux-image # Check
for kernel updates
sudo apt install linux-image-generic # Upgrade the kernel
```

💻 2. Working with Hardware

🖥️ Device Files in Linux

Linux represents hardware as files in the /dev/ directory:

🔹 **Character Devices** – Process data **one character at a time** (e.g., `/dev/tty` for terminals).

🔹 **Block Devices** – Process data **in blocks** (e.g., /dev/sda for storage).

📌 Checking Hardware Information

Use these commands to check system hardware:

```
lscpu        # Display CPU details
lsblk        # Show storage devices and partitions
lspci        # List PCI devices (graphics, network cards, etc.)
lsusb df     # List connected USB devices
fhee         # Show disk space usage
-m           # Display RAM usage in MB
```

📌 Managing Devices

📌 **Mount a device:**

```
sudo mount /dev/sdb1 /mnt # Mount device at /mnt
```

📌 **Unmount a device:**

```
sudo umount /mnt # Unmount device from /mnt
```

📌 **Load a kernel module:**

```
sudo modprobe <module_name> # Load a specific kernel module
```

📌 **Remove a kernel module:**

```
sudo rmmod <module_name> # Remove a kernel module
```

🚀 3. Linux Boot Sequence

The **Linux Boot Process** consists of:

1 **BIOS/UEFI**– Initializes hardware and loads bootloader.
2 **Bootloader (GRUB, LILO, Syslinux, etc.)**– Loads the Linux kernel.
3 **Kernel Initialization**– Detects hardware and mounts the root filesystem.
4 **Init System (Systemd, SysVinit, Upstart)**– Starts system processes.
5 **Runlevel/Target Execution**– Loads user processes and services.

📌 Checking Boot Information

```
dmesg | less       # Show boot messages
journalctl -b      # View system logs from the last boot
cat /var/log/syslog # Check system boot logs
```

🔄 4. Runlevels (SysVinit) and Systemd Targets

📌 Runlevels in SysVinit Systems

🏷️ **Runlevel**	📌 **Description**
0	Shutdown 🔴
1	Single-user mode 🔧
2	Multi-user (No Networking) 🔌
3	Multi-user (With Networking) 🌐
4	Unused/Custom ⚙️
5	Graphical mode (GUI) 🎨
6	Reboot 🔄

📌 **Check & Change Runlevels:**

```
runlevel        # Check current runlevel
sudo init 3     # Change to runlevel 3 (multi-user mode)
```

📌 Systemd Targets (Modern Linux Systems)

Systemd Target	**Equivalent Runlevel**

poweroff.target	Runlevel **0** (Shutdown)
rescue.target	Runlevel **1**(Single-user mode)
multi-user.target	Runlevel **3**(Multi-user mode)
graphical.target	Runlevel **5**(Graphical mode)
reboot.target	Runlevel **6** (Reboot)

📌 **Managing Systemd Targets:**

```
systemctl get-default # Check the current target
sudo systemctl set-default multi-user.target # Set default target
sudo systemctl isolate graphical.target # Switch to a specific target
```

📂 5. Linux File Types

🏷️ **File Type**	📌 **Description**	📁 **Example**
Regular File	Normal text, binary, or script file	`/etc/passwd`
Directory	Folder containing files	`/home/user/`
Symbolic Link	Shortcut to another file	`/usr/bin/python -> /usr/bin/python3`
Character Device	Reads/Writes one character at a time	`/dev/tty`
Block Device	Reads/Writes data in blocks	`/dev/sda`
Named Pipe	Inter-process communication	`/tmp/mypipe`
Socket	Network communication endpoint	`/var/run/docker.sock`

📌 **Checking File Types:**

```
ls -l # Display file types
file /dev/sda # Check type of a specific file
```

📌 **Example output of `ls -l`:**

```
-rw-r--r-- 1 user user 1024 Mar 07 12:00 file.txt    # Regular file
drwxr-xr-x 2 user user 4096 Mar 07 12:00 mydir/      # Directory
lrwxrwxrwx 1 user user    10 Mar 07 12:00 link -> file.txt # Symbolic link
```

🗂️ 6. Filesystem Hierarchy

📌 The **Filesystem Hierarchy Standard (FHS)** organizes Linux directories:

📁 **Directory**	📌**Purpose**	🔍 **Example Command**
`/`	**Root directory** (everything starts here)	`ls /`
`/bin`	Essential binaries (e.g., `ls`, `cat`)	`ls /bin`
`/boot`	Boot files (Kernel, GRUB)	`ls /boot`
`/dev`	Device files	`ls /dev`
`/etc`	Configuration files	`ls /etc`
`/home`	User home directories	`ls /home`
`/lib`	Shared libraries & kernel modules	`ls /lib`
`/media`	Mount points for removable media	`ls /media`
`/mnt`	Temporary mount points	`ls /mnt`
`/opt`	Optional software packages	`ls /opt`
`/proc`	Virtual filesystem (process info)	`ls /proc`
`/root`	Home directory for root user	`ls /root`
`/sbin`	System binaries (for admin tasks)	`ls /sbin`
`/srv`	Data served by the system	`ls /srv`
`/tmp`	Temporary files	`ls /tmp`

`/usr`	User applications and libraries	`ls /usr`
`/var`	Variable data (logs, caches, databases)	`ls /var`

📌 **Checking Disk Usage:**

```
df -h # Show available disk space
du -sh /var/log # Display the size of the /var/log directory
```

📦 Linux Package Management 🛠️

🏗️

🔍 1. Introduction to Package Management

What is Package Management?

Package management is how Linux handles **software installation, updates, and removal**. Different distributions use different package management tools.

📌 Types of Package Managers

🔹 **Low-Level Package Managers (work directly with package files):**

- rpm → Used in **Red Hat-based** systems
- dpkg → Used in **Debian-based** systems

🔹 **High-Level Package Managers (handle dependencies & repositories):**

- yum / dnf → Used in **Red Hat-based** systems
- apt / apt-get → Used in **Debian-based** systems

📌 Package Manager Categories by Distribution

🏷️ **Distribution**	📦 **Low-Level Package Manager**	📦 **High-Level Package Manager**
Debian-based (Ubuntu, Debian, Mint)	`dpkg`	`apt, apt-get`
Red Hat-based (RHEL, CentOS, Fedora)	`rpm`	`yum, dnf`

🔴 2. RPM and YUM (For Red Hat-based Systems)

📌 RPM (Red Hat Package Manager)

RPM is a **low-level package manage r**used in Red Hat-based distributions like **CentOS** and **Fedora**.

⚡ Common RPM Commands

📌 **Install a package:**

```
sudo rpm -ivh package.rpm
```

📌 **Upgrade a package:**

```
sudo rpm -Uvh package.rpm
```

📌 **Remove a package:**

```
sudo rpm -e package-name
```

📌 **Query installed packages:**

```
rpm -qa | grep package-name
```

📌 **Show package details:**

```
 rpm -qi package-name
```

📌 YUM (Yellowdog Updater, Modified)

`YUM` is a **high-level package manager** that resolves dependencies automatically.

⚡ Common YUM Commands

📌 **Install a package:**

```
sudo yum install package-name
```

📌 **Remove a package:**

```
sudo yum remove package-name
```

📌 **Update all packages:**

```
sudo yum update
```

📌 **Check if a package is installed:**

```
yum list installed | grep package-name
```

📌 **Show package info:**

```
yum info package-name
```

🔹 **Note:** `dnf`is the newer version of `yum` used in Fedora and CentOS 8+.

🔵 3. DPKG and APT (For Debian-based Systems)

📌 DPKG (Debian Package Manager)

`dpkg` is a **low-level** package manager for **.deb** packages and **does not handle dependencies** automatically.

⚡📌 Common DPKG Commands

Install a package:

```
sudo dpkg -i package.deb
```

📌 **Remove a package:**

```
sudo dpkg -r package-name
```

📌 **Reconfigure a package:**

```
sudo dpkg-reconfigure package-name
```

📌 **List installed packages:**

```
dpkg -l | grep package-name
```

📌 APT (Advanced Package Tool)

APT is a **high-level package manager** for **Debian-based** systems that resolves dependencies automatically.

⚡ Common APT Commands

📌 **Update package lists:**

```
sudo apt update
```

📌 **Install a package:**

```
sudo apt install package-name
```

📌 **Remove a package:**

```
sudo apt remove package-name
```

📌 **Upgrade all packages:**

```
sudo apt upgrade
```

📌 **Show package details:**

```
apt show package-name
```

📌 **Search for a package:**

```
apt search package-name
```

⚖️ 4. APT vs APT-GET

🔹 **Both `apt` and `apt-get` are used in Debian-based distributions, but `apt` is newer and more user-friendly.**

📌 Key Differences

🏷️ **Feature**	📦 `apt-get`	📦 `apt`
Introduced in	**Debian**	**Ubuntu 16.04+**
Handles package management	✅ Yes	✅ Yes
Displays progress bar	❌ No	✅ Yes
Simplified syntax	❌ No	✅ Yes

📌 Example Commands Comparison

Task	**apt-get (Old)**	**apt (New)**
Update package lists	`sudo apt-get update`	`sudo apt update`
Install a package	`sudo apt-get install package-name`	`sudo apt install package-name`
Upgrade all packages	`sudo apt-get upgrade`	`sudo apt upgrade`

🔹 **Note:** `apt-get` is still available for **backward compatibility**, but `apt` is recommended for newer systems.

🚀 Conclusion

✅ **Red Hat-based systems** use rpm (low-level) and yum/dnf (high-level).
✅ **Debian-based systems** use dpkg (low-level) and apt/apt-get (high-level).
✅ **High-level package managers** (apt, yum, dnf) **handle dependencies automatically**.
✅ **Low-level package managers**(rpm, dpkg) **work directly with package file. s**

📌 Choosing the Right Package Manager:

- If you're on **Ubuntu/Debian**, use apt.
- If you're on **Fedora/CentOS**, use dnf.
- If you want **manual control**, use dpkgor rpm.

🐚 Working with Shell in Linux

1. File Compression and Archival

📌 What is File Compression and Archival?

- **Compression** reduces file sizes, making them easier to store or transfer.
- **Archiving**bundles multiple files into a single file.

🛠️ Common Compression and Archival Commands

Command	Description
`tar`	Archive multiple files into one file
`gzip`	Compress files using the Gzip algorithm
`bzip2`	Compress files using the Bzip2 algorithm
`xz`	Compress files using the XZ algorithm
`zip`	Compress files into a `.zip` format
`unzip`	Extract `.zip` archives

📦 Archiving with tar

Create an archive (without compression):

```
tar -cvf archive.tar file1 file2 directory/
```

Extract an archive:

```
tar -xvf archive.tar
```

List contents of an archive:

```
tar -tvf archive.tar
```

🗜️ Compressing with gzip, bzip2, and xz

Compress a file using gzip:

```
gzip file.txt # Produces file.txt.gz
gunzip file.txt.gz # Decompress
```

Compress a file using bzip2:

```
bzip2 file.txt # Produces file.txt.bz2
bunzip2 file.txt.bz2 # Decompress
```

Compress a file using xz:

```
xz file.txt # Produces file.txt.xz
unxz file.txt.xz # Decompress
```

📁 Combining tar with Compression

Create a gzip-compressed archive:

```
tar -czvf archive.tar.gz file1 file2 directory/
```

Create a bzip2-compressed archive:

```
tar -cjvf archive.tar.bz2 file1 file2 directory/
```

Create an xz-compressed archive:

```
tar -cJvf archive.tar.xz file1 file2 directory/
```

Extract compressed archives:

```
tar -xzvf archive.tar.gz # For gzip
tar -xjvf archive.tar.bz2 # For bzip2
tar -xJvf archive.tar.xz # For xz
```

📌 Working with Zip Files

Create a zip archive:

```
zip archive.zip file1 file2 directory/
```

Extract a zip archive:

```
unzip archive.zip
```

---

2. Searching for Files and Patterns

🔍 Why Search for Files?

Searching for files and text patterns is **essential for managing a Linux system** effectively.

🔎 Finding Files with find

Find a file by name:

```
find /home -name "file.txt"
```

Find files larger than 100MB:

```
find / -size +100M
```

Find all `.log` files in `/var/log`:

```
find /var/log -name "*.log"
```

Find and delete files older than 7 days:

```
find /tmp -type f -mtime +7 -exec rm {} \;
```

📌 Searching for Text in Files with grep

Find a pattern in a file:

```
grep "error" /var/log/syslog
```

Case-insensitive search:

```
grep -i "warning" /var/log/syslog
```

Recursive search in a directory:

```
grep -r "TODO" /home/user/projects/
```

Show line numbers for matches:

```
grep -n "failed" /var/log/auth.log
```

⚡ Searching Faster with locate

Find files using `locate` (requires `updatedb` first):

```
locate bashrc
```

Update the locate database:

```
sudo updatedb
```

---

3. I/O Redirection

🔄 What is I/O Redirection?

I/O redirection allows you to control where input and output streams go in the shell.

Stream	Description	File Descriptor
`stdin`	Standard input	0
`stdout`	Standard output	1
`stderr`	Standard error	2

📤 Redirecting Output to a File

Redirect stdout to a file (overwrite):

```
ls > output.txt
```

Redirect stdout to a file (append):

```
ls >> output.txt
```

Redirect stderr to a file:

```
ls non_existing_file 2> error.txt
```

Redirect both stdout and stderr:

```
ls non_existing_file > output.txt 2>&1
```

Suppress error messages:

```
command 2>/dev/null
```

📥 Redirecting Input from a File

Read from a file instead of standard input:

```
sort < unsorted.txt
```

🔗 Using Pipes (`|`) to Redirect Output Between Commands

Count lines in a file:

```
cat file.txt | wc -l
```

Find all `.log` files and search for "error":

```
find /var/log -name "*.log" | xargs grep "error"
```

---

4. Vi Editor

📝 What is the Vi Editor?

`vi` is a **powerful text editor** used in Unix/Linux systems.

📂 Opening and Closing Files

Open a file in vi editor:

```
vi filename
```

Exit without saving:

```
:q!
```

Save and exit:

```
:wq
```

🛠️ Vi Modes

Mode	Description
Normal Mode	Default mode for navigation and commands
Insert Mode	Used for text editing (press `i` or `a` to enter)
Command Mode	Used for executing commands (`:` followed by command)

📌 Basic Navigation

- **Move left:** `h`
- **Move right:** `l`
- **Move down:** `j`
- **Move up:** `k`

✏️ Editing in Insert Mode

- **Insert before cursor:** `i`
- **Append after cursor:** `a`
- **Open a new line below:** `o`
- **Return to Normal mode:** `ESC`

📝 Copy, Cut, and Paste

- **Copy a line:** `yy`
- **Cut (delete) a line:** `dd`
- **Paste after the cursor:** `p`
- **Paste before the cursor:** `P`

🔍 Searching in Vi

- **Search forward for "word":** `/word`
- **Search backward for "word":** `?word`

- **Repeat last search forward:** `n`
- **Repeat last search backward:** `N`

💾 Saving and Exiting

- **Save:** `:w`
- **Quit:** `:q`
- **Save and quit:** `:wq`
- **Quit without saving:** `:q!`

Linux Networking

1 Network Issues

📌 Overview

Networking issues in Linux can arise due to misconfigurations, hardware failures, or connectivity problems. Below are **common network issues, their causes, and solutions**.

🛠️ Common Network Issues

Issue	Cause	Solution
No Internet	Missing IP address, no default gateway	Check with `ip a`, `route -n`
Slow Connection	High network traffic, DNS issues	Test with `ping`, `traceroute`
Cannot Reach Host	Firewall, incorrect routes	Check `iptables`, `firewalld`, `netstat -r`
Hostname Not Resolving	DNS issues	Verify `/etc/resolv.con` ufse `nslookup`, `dig`
No Network Interface	Driver issue	Use `lspci`, `lsmod`, reinstall drivers

📌 Basic Network Commands

```
ip a ip r ping 8.8.8.8  # Show network interfaces
google.com              # Display routing table
traceroute
google.com              # Test connectivity
nslookup
netstat -tulnp          # Trace route to a host # Resolve domain
                        to IP
                        # Show open ports and listening services
```

2 DNS (Domain Name System)

📌 Overview

DNS translates domain names (e.g., `google.com`) into IP addresses.

🛠 Checking DNS Configuration

View DNS servers in use:

```
cat /etc/resolv.conf
```

Test domain resolution:

```
nslookup google.com
dig google.com
host google.com
```

📌 Flushing DNS Cache

If using `systemd-resolved`:

```
sudo systemctl restart systemd-resolved
```

For `nscd` (Name Service Cache Daemon):

```
sudo systemctl restart nscd
```

📌 Manually Setting DNS

Edit `/etc/resolv.conf`(temporary change):

```
nameserver 8.8.8.8
nameserver 1.1.1.1
```

For permanent changes, modify `/etc/systemd/resolved.conf`:

```
[Resolve]
DNS=8.8.8.8 1.1.1.1
```

Then restart the service:

```
sudo systemctl restart systemd-resolved
```

---

3 Networking Basics

📌 Overview

Linux networking follows a layered approach similar to the **OSI model**.

🛠️ Key Networking Components

Component	Description
IP Addressing	Identifies a device in a network (e.g., `192.168.1.1`)
Subnet Mask	Defines the network range (e.g., 255.255.255.0)
Gateway	Router address for external traffic
DNS Server	Resolves domain names (e.g., 8.8.8.8 for Google DNS)
MAC Address	Unique hardware address of a network interface

📌 Viewing Network Configuration

```
ip  a  ip   # Show IP addresses
r      ip   # Show routing table
link        # Show network interfaces
```

📌 Assigning a Static IP (Temporary)

```
sudo ip addr add 192.168.1.100/24 dev eth0
sudo ip route add default via 192.168.1.1
```

📌 Assigning a Static IP (Permanent - Ubuntu)

Edit `/etc/netplan/*.yaml`:

```
network:
  ethernets:
    eth0:
      addresses:
        - 192.168.1.100/24
      gateway4: 192.168.1.1
      nameservers:
      addresses:
          - 8.8.8.8
          - 1.1.1.1
  version: 2
```

Apply the changes:

```
sudo netplan apply
```

---

4 Network Troubleshooting

📌 Overview

Troubleshooting involves diagnosing and resolving network issues **systematically**.

🛠️ Step-by-Step Troubleshooting

🟢 Step 1: Check Network Interfaces

```
ip a         # List all interfaces
ifconfig -a # Show interface details
```

🟢 Step 2: Check Connectivity

```
ping 8.8.8.8            # Test internet access
ping -c 5 google.com    # Test DNS resolution
```

● Step 3: Verify Routing Table

```
ip r       # Show routing table
route -n   # Display routes
```

● Step 4: Check DNS Resolution

```
nslookup google.com
dig google.com
host google.com
```

● Step 5: Check Open Ports

```
netstat -tulnp # Show listening ports
ss -tulnp       # Alternative to netstat
```

● Step 6: Restart Networking Services

```
sudo systemctl restart networking          # Restart networking service
sudo systemctl restart systemd-resolved    # Restart DNS resolver
```

● Step 7: Inspect Logs

```
journalctl -u networking --no-pager # View networking logs
dmesg | grep eth0                     # Check for interface errors
```

✅ Summary of Key Commands

Action	Command
Show network interfaces	`ip a ip r ping 8.8.8.8`
Display routing table	`traceroute google.com`
Test connectivity	
Trace route to a host	

Check DNS resolution	`nslookup google.com`
Restart networking service	`sudo systemctl restart networking`
Assign a static IP (temporary)	`sudo ip addr add 192.168.1.100/24 dev eth0`
Set a static IP permanently (Ubuntu)	Edit `/etc/netplan/*.yaml` and run `sudo netplan apply`
Show open ports	`netstat -tulnp` or `ss -tulnp`
View networking logs	`journalctl -u networking --no-pager`

🔒 Linux Security and File Permissions

1 Security Incidents

📌 Overview

A security incident in Linux refers to unauthorized access, malware, data breaches, or misconfigurations that compromise system integrity.

🛠️ Common Security Incidents in Linux

Incident Type	Description	Detection Method
Unauthorized Access	An attacker gains unauthorized system access	Check `/var/log/auth.log`, `last`, `w`
Privilege Escalation	User gains unauthorized root access	Audit sudo logs (`cat /var/log/auth.log`)
Malware Infection	Malicious software compromises the system	Scan with `chkrootkit`, `clamav`
File Integrity Breach	Unauthorized modification of files	Use `tripwire` or `aide` for integrity checking
Denial of Service (DoS)	Overloading services to cause downtime	Monitor `netstattop`, `fail2ban`

📌 Basic Security Audit Commands

```
last                            # Show last logins
who                             # Show logged-in users
ps aux --sort=-%mem             # Find processes using high memory
netstat -tulnp                  # Check open ports
```

2 Linux Accounts

📌 Overview

Linux has different types of accounts to manage access and permissions.

🛠 Types of Linux Accounts

Account Type	Description
Root User (UID=0)	Full system access
Regular User	Restricted access, used for daily tasks
System Accounts (UID<1000)	Used by services (e.g., `www-data`, `mysql`)

📌 Managing User Accounts

```
whoami                         # Show current user
id user1                       # Show user ID and group ID
cat /etc/passwd                # List all users
```

---

3 Access Control Files

📌 Overview

Linux uses access control files to manage permissions for users and groups.

🛠 Important Access Control Files

File	Description
`/etc/passwd`	Stores user account details
`/etc/shadow`	Stores encrypted passwords
`/etc/group`	Manages group memberships
`/etc/sudoers`	Controls sudo access

📌 Checking User Access

```
cat /etc/passwd | grep user1   # View user account details
cat /etc/group | grep sudo     # Check if a user is in the sudo group
```

---

4 User Management

📌 Creating and Managing Users

```
# Create a new user
sudo useradd -m user1

# Set a password for the user
sudo passwd user1

# Add a user to a group
sudo usermod -aG sudo user1

# Delete a user
sudo userdel -r user1
```

📌 Group Management

```
# Create a new group
sudo groupadd developers

# Add a user to a group
sudo usermod -aG developers user1

# Remove a user from a group
sudo gpasswd -d user1 developers
```

---

5 File Permissions and Ownership

📌 Overview

Linux file permissions determine who can read, write, and execute files.

🛠️ Understanding File Permissions

To view file permissions, use:

```
ls -l file.txt
```

Example output:

```
-rwxr-xr-- 1 user1 group1 1024 Mar 7 10:00 file.txt
```

Permission Symbol	**Meaning**
r (read)	Can read the file
w (write)	Can modify the file
x (execute)	Can run the file as a program

- **First three characters:**Owner permissions
- **Next three:**Group permissions
- **Last three:**Others' permissions

📌 Changing Permissions

```
chmod 755 file.txt        # Owner: rwx, Group: r-x, Others: r-x
chmod u+r file.txt        # Add read permission for owner
```

📌 Changing Ownership

```
chown user1 file.txt      # Change file owner
chgrp group1 file.txt     # Change file group
```

---

6 SSH and SCP

📌 Secure Shell (SSH)

SSH allows secure remote login:

```
ssh user@remote-server
```

📌 Copying Files Securely with SCP

```
# Copy file to a remote server
scp file.txt user@remote-server:/home/user/

# Copy file from a remote server
scp user@remote-server:/home/user/file.txt .
```

---

7 IPTables Introduction

📌 Overview

iptables is a firewall used to filter traffic.

🛠️ Basic IPTables Commands

```
# List current rules
sudo iptables -L -v

# Block an IP address
sudo iptables -A INPUT -s 192.168.1.10 -j DROP

# Allow SSH (port 22)
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Save rules
sudo iptables-save > /etc/iptables.rules
```

---

8 Securing the Environment with IPTables

📌 Securing Common Services

```
# Allow web traffic (HTTP and HTTPS)
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow internal network access
sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT

# Block all other incoming traffic
sudo iptables -P INPUT DROP
```

📌 Saving and Reloading Rules

```
sudo iptables-save > /etc/iptables.rules
sudo iptables-restore < /etc/iptables.rules
```

---

9 Cron Jobs

📌 Overview

Cron jobs automate scheduled tasks in Linux.

📌 Creating a Cron Job

```
crontab -e # Edit cron jobs
```

🛠️ Example Cron Jobs

Task	Cron Expression
Run a script every minute	`* * * * * /path/to/script.sh`
Run a script daily at midnight	`0 0 * * * /path/to/script.sh`

Run a script every Sunday at 2 AM	`0 2 * * 0 /path/to/script.sh`

📌 List and Remove Cron Jobs

```
crontab -l # List scheduled jobs
crontab -r # Remove all cron jobs
```

✅ Summary of Key Security Commands

Action	Command
Show last logins	`last who netstat -tulnp chmod`
Show logged-in users	`chown scp sudo iptables -L -v`
Check open ports	`sudo iptables -A INPUT -s <IP> -j DROP`
Change file permissions	`sudo systemctl restart sshd`
Change file ownership	
Securely copy files	
List current firewall rules	
Block an IP address	
Restart SSH service	

📌 Linux System Management: systemd Services

🔥

🛠 1. Creating a systemd Service

What is systemd?

systemd is an init system that manages services, processes, and system states. It uses **unit files**to control how services are started, stopped, and restarted.

🚀📝 Steps to Create a Custom systemd Service

1. Create the Service File

systemd service files are stored in /etc/systemd/system/.

```
sudo nano /etc/systemd/system/myservice.service
```

⚙ 2. Define the Service Configuration

A basic **systemd** service file has three sections:
✅ **[Unit]**– Describes the service and dependencies
✅ **[Service]**– Defines how the service runs
✅ **[Install]**– Specifies when to start the service

📜 Example systemd Service File

```
[Unit]
Description=My Custom Service
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/user/myscript.py
Restart=always
User=user
```

```
Group=user
WorkingDirectory=/home/user
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

🔄 3. Reload systemd to Recognize the New Service

```
sudo systemctl daemon-reload
```

▶4. Enable and Start the Service

```
sudo systemctl enable myservice
sudo systemctl start myservice
```

📌 5. Check Service Status

```
sudo systemctl status myservice
```

⛔ 6. Stop and Disable the Service

```
sudo systemctl stop myservice
sudo systemctl disable myservice
```

🛠️ 2. systemd Service Management and Tools

🎛️ Managing Services

Action	Command
Start a service	`sudo systemctl start myservice`

Stop a service	`sudo systemctl stop myservice`
Restart a service	`sudo systemctl restart myservice`
Reload a service	`sudo systemctl reload myservice`
Enable at boot	`sudo systemctl enable myservice`
Disable at boot	`sudo systemctl disable myservice`
Check service status	`sudo systemctl status myservice`
View service logs	`journalctl -u myservice`

📋 Checking System Services

```
systemctl list-units --type=service     # List all active services
systemctl list-unit-files --type=service # List all installed services
```

⚡ 3. Service Unit File Options

🏗️ [Unit] Section (Describes the service)

Option	Description
`Description=`	Short description of the service
`After=`	Services that should start **before** this one
`Before=`	Services that should start **after** this one
`Requires=`	**Hard dependency** – If this service fails, dependent services fail too

`Wants=`	**Soft dependency** – If this service fails, dependent services may continue

🔧 [Service] Section (Defines how the service runs)

Option	Description
`ExecStart=`	Command to start the service
`ExecStop=`	Command to stop the service
`ExecReload=`	Command to reload the service
`Restart=`	Restart policy (`always`, `on-failure`, `no`, `on-success`)
`User=`	Specifies which user runs the service
`Group=`	Specifies which group runs the service
`WorkingDirectory=`	Specifies the working directory
`Environment=`	Sets environment variables

🎯 [Install] Section (Specifies startup behavior)

Option	Description
`WantedBy=`	Defines target where the service should start (e.g., `multi-user.target`)
`RequiredBy=`	Services that depend on this service

🛠️ 4. systemctl Command Options

Command	Description

`systemctl start <service>`	Starts a service
`systemctl stop <service>`	Stops a service
`systemctl restart <service>`	Restarts a service
`systemctl reload <service>`	Reloads the configuration of a running service
`systemctl enable <service>`	Enables a service at boot
`systemctl disable <service>`	Disables a service at boot
`systemctl status <service>`	Shows service status
`systemctl is-active <service>`	Checks if a service is running
`systemctl is-enabled <service>`	Checks if a service is enabled
`systemctl mask <service>`	Prevents a service from starting
`systemctl unmask <service>`	Allows a masked service to start

📜 5. Checking systemd Logs

```
journalctl -xe   # Show detailed logs
journalctl -u myservice --since "1 hour ago" # View logs for a specific service
journalctl --boot -1 # Show logs from the previous boot
```

✏️ 6. Editing and Reloading Services

```
sudo systemctl edit myservice # Edit service configuration
sudo systemctl daemon-reexec    # Restart systemd itself
sudo systemctl reset-failed     # Clear failed services list
```

✅ Conclusion

- systemd provides a powerful way to manage services in Linux.
- You can create, enable, disable, start, stop, and monitor services with systemctl.
- Logs can be viewed with journalctl to debug issues.
- Understanding systemd helps in automating processes, running background jobs, and optimizing system performance.

🚀 **Mastering systemd = Better Linux System Administration!** 🔥💻

💾 Linux Storage Management

🔥 1. Disk Partitions

A **disk partition** is a logical division of a physical storage device that allows better management of data. In Linux, partitions are managed using tools like fdisk, parted, and lsblk.

🗂️ Types of Partitions

Partition Type	Description
Primary	Can contain a bootable OS, limited to 4 partitions
Extended	Used to create more than 4 partitions, acts as a container
Logical	Partitions inside an extended partition

📋 Checking Disk Partitions

```
l#sbLlikst  block  devices  and  partitions  #
Display  fpadritsikti-oln  details  parted  -l  #
Show partition table info
```

🛠️ Creating a New Partition with fdisk

```
sudo fdisk /dev/sdb
# Press 'n' → Create a new partition
# Press 'p' → Make it a primary partition
# Enter partition number and size
# Press 'w' → Write changes
```

🔧 Formatting a Partition

```
sudo mkfs.ext4 /dev/sdb1   # Format as ext4
```

📂 Mounting a Partition

```
sudo mount /dev/sdb1 /mnt/data
```

To make it permanent, add to `/etc/fstab`:

```
/dev/sdb1 /mnt/data ext4 defaults 0 2
```

🗄️ 2. File Systems in Linux

A **file system** defines how data is stored and accessed on a disk.

🔍 Common File Systems

File System	Description
ext4	Default Linux file system, supports journaling
xfs	High-performance file system, used in RHEL
btrfs	Supports snapshots and RAID features
vfat	Used for USB drives, compatible with Windows
ntfs	Windows file system, needs `ntfs-3g` to write

🛠️ Checking File System Type

```
df -T /mnt/data   # Check file system type of a mount point
lsblk -f          # Show file system type of all partitions
```

🔄 Converting File Systems

```
sudo mkfs.xfs /dev/sdb1   # Convert to XFS
sudo mkfs.btrfs /dev/sdb1 # Convert to Btrfs
```

💾 3. DAS, NAS, and SAN

Linux storage can be categorized into:
✔ **Direct-Attached Storage (DAS)**
✔ **Network-Attached Storage (NAS)**
✔ **Storage Area Network (SAN)**

🖥 3.1 Direct-Attached Storage (DAS)

DAS refers to storage devices that are directly connected to a single computer without a network.

Examples of DAS

- **Internal Hard Drives (HDDs & SSDs)**: SATA, NVMe
- **External USB Drives**: USB hard drives, flash drives
- **RAID**: Locally attached RAID controllers

✅ Advantages of DAS

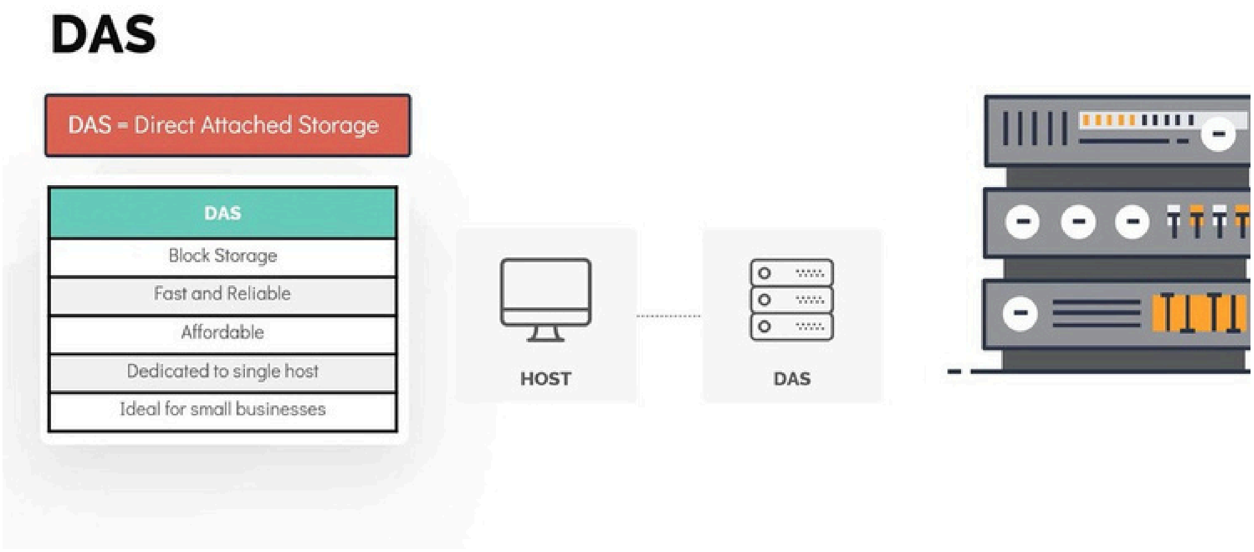
✔ High speed (direct connection)
✔ No network dependency
✔ Lower cost

❌ Disadvantages of DAS

✖ Limited to a single machine
✖ Not easily shareable

🛠 Checking DAS Devices in Linux

```
lsblk       # Show block devices
fdisk -l    # List partitions
df -h       # Show mounted file systems
```



DAS
Block Storage
Fast and Reliable
Affordable
Dedicated to single host
Ideal for small businesses

🌐 3.2 Network-Attached Storage (NAS)

NAS is a storage device connected to a network, accessible by multiple clients over protocols like **NFS (Linux/Unix)**and **SMB (Windows)**.

Examples of NAS

- **Dedicated NAS appliances** (e.g., Synology, QNAP)
- **Linux-based file servers**

✅ Advantages of NAS

✓ Centralized storage
✓ Multiple users can access files
✓ Easy to scale

❌ Disadvantages of NAS

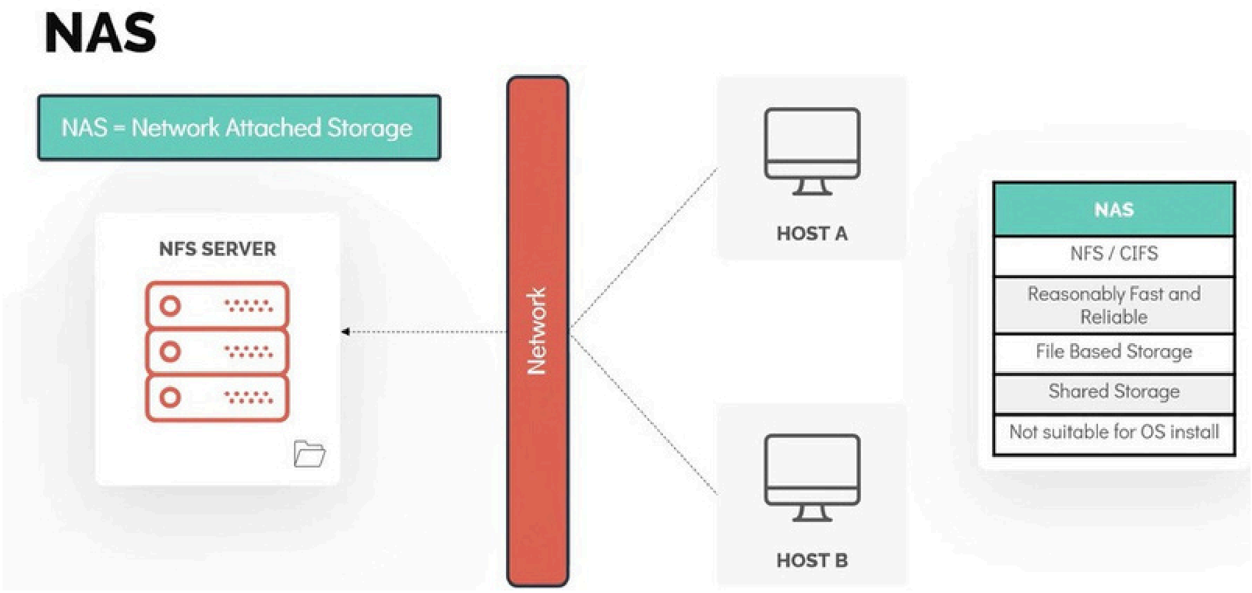
✕ Slower than local DAS
✕ Dependent on network performance

🛠️ Mounting a NAS Share via NFS

```
sudo mount -t nfs 192.168.1.100:/shared_folder /mnt/nas
```

To make it persistent, add to `/etc/fstab`:

```
192.168.1.100:/shared_folder /mnt/nas nfs defaults 0 0
```



NAS
NFS / CIFS
Reasonably Fast and Reliable
File Based Storage
Shared Storage
Not suitable for OS install

⚡ 3.3 Storage Area Network (SAN)

SAN is a high-speed network dedicated to providing block-level storage to multiple machines, often used in data centers.

Examples of SAN Technologies

- **Fibre Channel (FC)** – High-speed storage networking
- **iSCSI (Internet Small Computer System Interface)**– Block storage over TCP/IP

✅ Advantages of SAN

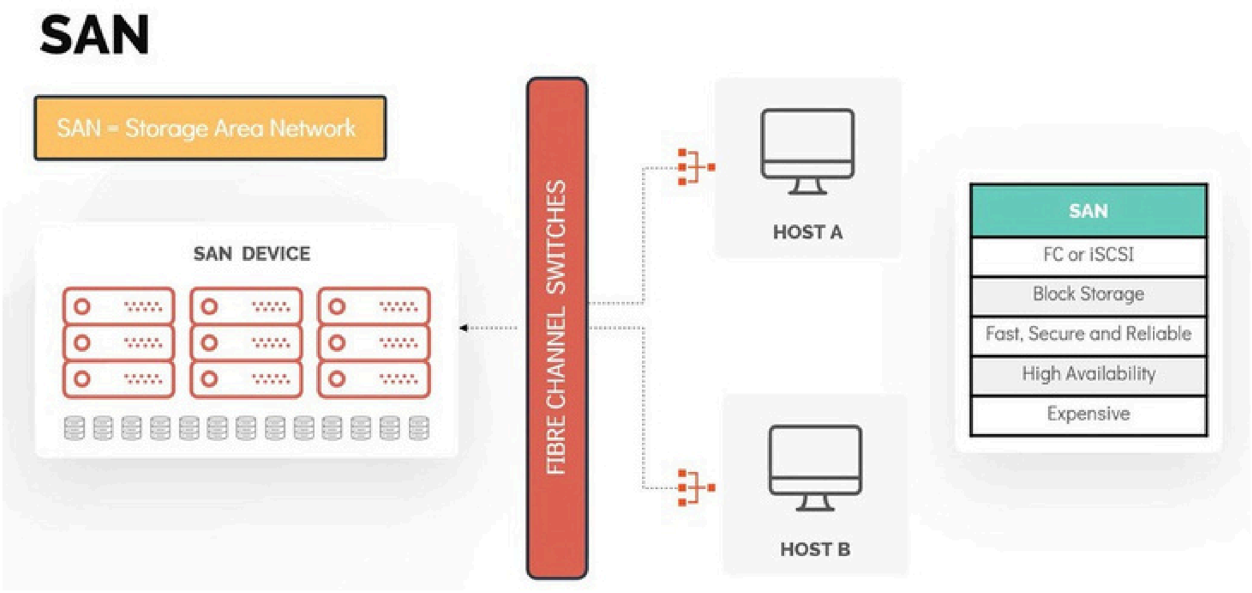
✓ High-speed performance
✓ Redundancy and fault tolerance
✓ Supports virtualization

❌ Disadvantages of SAN

✕ Expensive setup
✕ Requires specialized hardware

🛠️ Checking SAN/iSCSI Devices in Linux

```
iscsiadm -m session # List active iSCSI sessions
lsblk                # Show block devices
multipath -ll        # Show multi-path storage devices
```



📡 4. NFS (Network File System) in Linux

NFS allows Linux systems to share files over a network using a **client-server architecture**.

🔧 4.1 Installing NFS

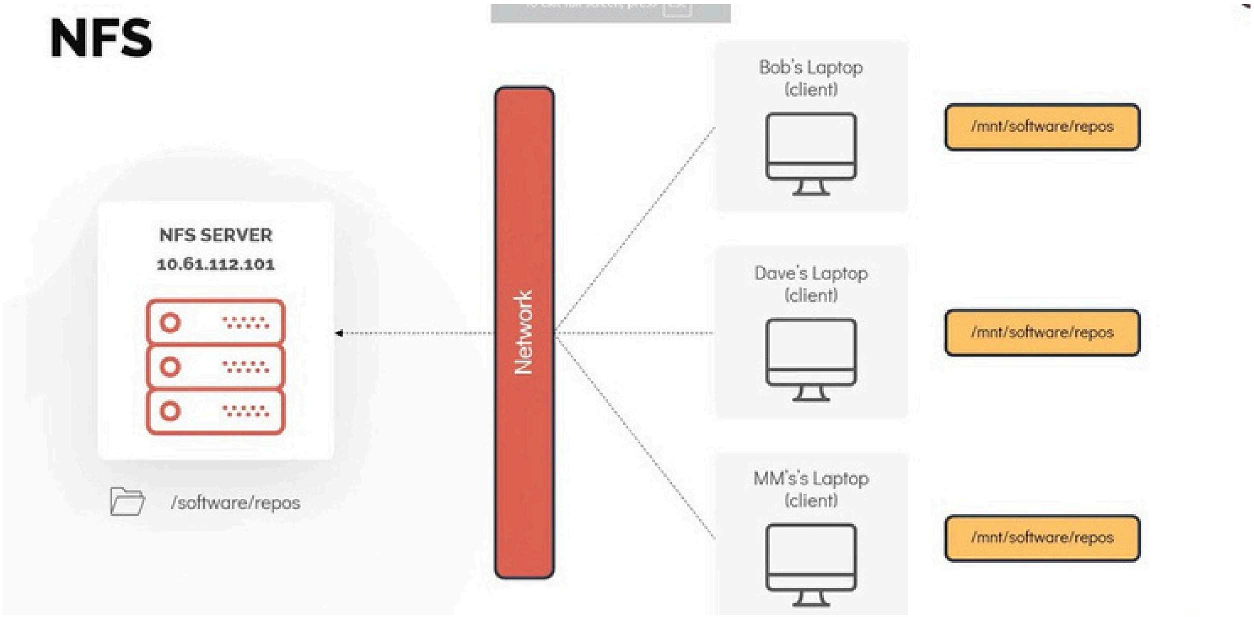
On the NFS Server:

```
sudo apt install nfs-kernel-server   # Debian-based

sudo yum install nfs-utils           # RHEL-based
```

On the NFS Client:

```
sudo apt install nfs-common   # Debian-based
sudo yum install nfs-utils    # RHEL-based
```



🛠️ 4.2 Configuring an NFS Server

Create a Shared Directory:

```
sudo mkdir -p /mnt/shared
sudo chmod 777 /mnt/shared
```

Edit the NFS Exports File:

```
sudo nano /etc/exports
```

Add:

```
/mnt/shared 192.168.1.0/24(rw,sync,no_root_squash)
```

Restart NFS Service:

```
sudo systemctl restart nfs-server
```

📂 4.3 Mounting an NFS Share on a Client

```
sudo mount -t nfs 192.168.1.100:/mnt/shared /mnt/client
```

To make it persistent, add to `/etc/fstab`:

```
192.168.1.100:/mnt/shared /mnt/client nfs defaults 0 0
```

📊 4.4 Checking NFS Status

```
showmount -e 192.168.1.100 # Show available NFS shares
systemctl status nfs-server # Check NFS service status
```

📦 5. LVM (Logical Volume Manager)

LVM allows **flexible disk management**, enabling resizing and snapshots.

📜 5.1 LVM Components

Component	Description
Physical Volume (PV)	Raw storage device (e.g., `/dev/sdb`)
Volume Group (VG)	Collection of physical volumes
Logical Volume (LV)	Partition inside a volume group

5.2 Creating an LVM Partition

Step 1: Initialize the Physical Volume

```
sudo pvcreate /dev/sdb
```

Step 2: Create a Volume Group

```
sudo vgcreate my_vg /dev/sdb
```

Step 3: Create a Logical Volume

```
sudo lvcreate -L 10G -n my_lv my_vg
```

Step 4: Format and Mount the LVM Volume

```
sudo mkfs.ext4 /dev/my_vg/my_lv


sudo mkdir /mnt/lvm

sudo mount /dev/my_vg/my_lv /mnt/lvm
```

To make the mount persistent, add it to `/etc/fstab`:

```
/dev/my_vg/my_lv /mnt/lvm ext4 defaults 0 2
```

🔄 5.3 Resizing an LVM Volume

Increase LVM Size

```
sudo lvextend -L +5G /dev/my_vg/my_lv # Increase by 5GB


sudo resize2fs /dev/my_vg/my_lv        # Resize file system
```

Reduce LVM Size (Unmount First!)

```
sudo umount /mnt/lvm


sudo lvreduce -L -5G /dev/my_vg/my_lv

sudo resize2fs /dev/my_vg/my_lv

sudo mount /dev/my_vg/my_lv /mnt/lvm
```

📸 5.4 Creating LVM Snapshots

Create a Snapshot

```
sudo lvcreate -L 1G -s -n my_snapshot /dev/my_vg/my_lv
```

Restore from a Snapshot

```
sudo lvconvert --merge /dev/my_vg/my_snapshot
```

📊 5.5 Checking LVM Status

Show Physical Volumes

```
sudo pvs
```

Show Volume Groups

```
sudo vgs
```

Show Logical Volumes

```
sudo lvs
```