Date: July-19th-2025

ORIENTATION

PRESENTATION: Welcome to NIT

"Smallness of means, greatness of purpose and Astounding Results"

1

Date: July-19th-2025

CHAPTER 1
NIT ACADEMY – The Road to Career Change!

Using the rules and strategies in this section you will gain confidence and a deep insight in setting up yourself for success

- ❑Introductions
- ❑Training Team
- ❑Let us get to know you all!
- ❑Physical Machine Requirements
- ❑Virtual Machine Provisioning & Configuration
- ❑What you will learn?
- ❑Why will you learn?
- ❑Soft Skills
- ❑Fair Student Agreement
- ❑Interview and Job Placement

NEXUS TEAM

Date: July-19th-2025

Semeon Monehin

Executive Director

Nadeem Siddiqi

Director

Frederic Turlan

Executive Director

Bilal Mufti

Marketing & Sales

Bradford Laney

Attorney

Salmah Siddiqi

PR and Production

NIT TRAINING TEAM

Date: July-19th-2025

Yasin Sekabira

Lead Instructor

- Bachelors in Computer Science
- MSc in Computer Science
- CEO / CTO for number of IT Start-ups
- Fin-Tech Experience
- Data Center and Network Management

Nadeem Siddiqi

Program Lead

- Bachelors in Engineering
- Post Bachelors in ML/AI
- CEO / CTO for number of Start-ups
- Expertise in System Admin and Data Center
- Fin-Tech Experience
- Server and Network Management

Date: July-19th-2025

Welcome, New Students!

Introduce Yourself

Share your name, background, and interests to help everyone get acquainted and foster connections.

Course Goals and Expectations

Discuss your aspirations and what you hope to accomplish in this course to set a clear purpose.

Supportive Community Building

Building a supportive learning community enhances engagement and success in our PowerPoint-based Zoom sessions.

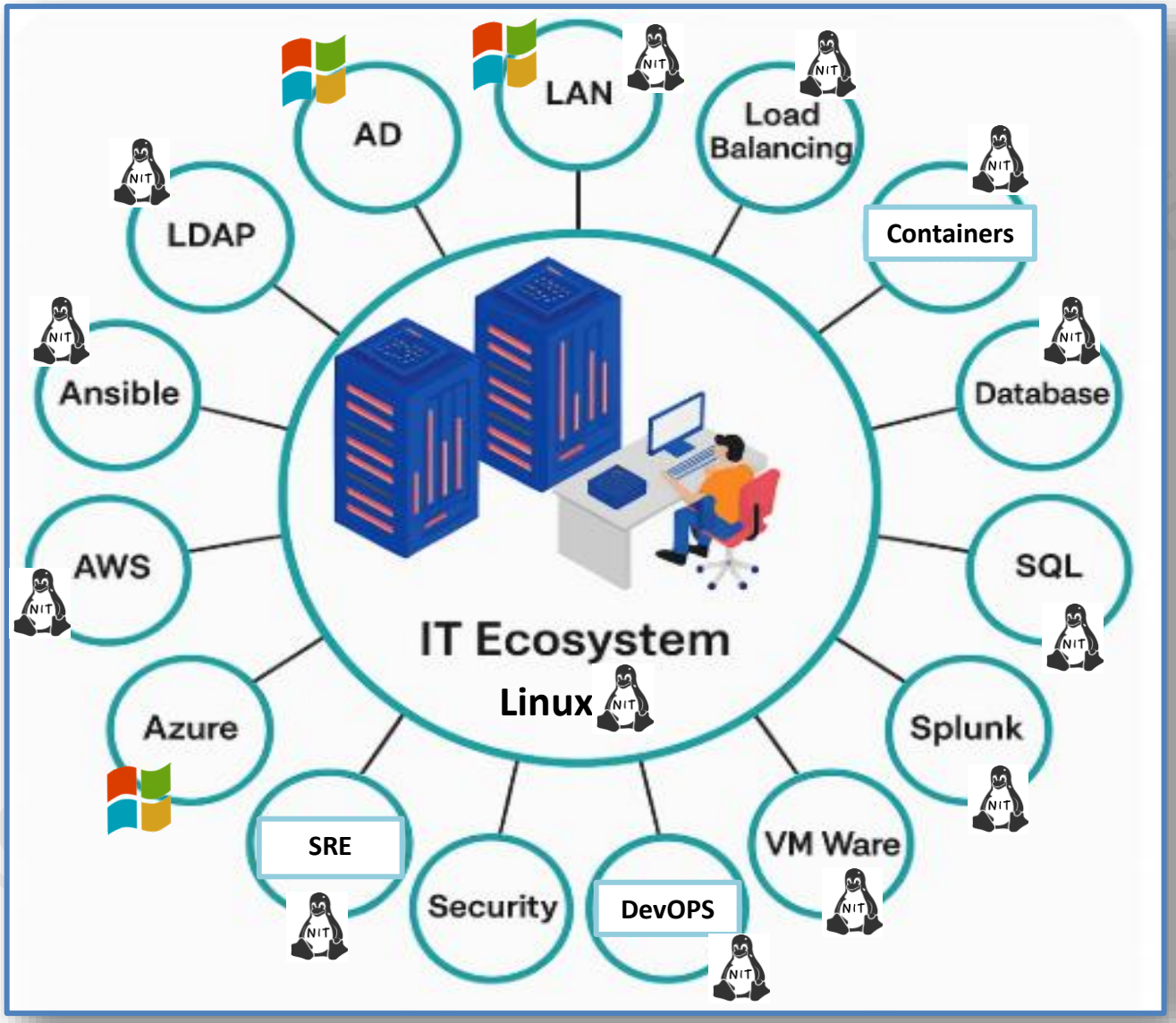
About our Program

10-12 weeks (learn absolutely free)

RHCSA – exam $500 taken at home

Empowering an IT Ecosystem Careers

Date: July-19th-2025



RedHat (RH124/RH134) Exam

Date: July-19th-2025

- **Access the command line**
  Log in to a Linux system and run simple commands from the shell.
- **Manage files from the command line**
  Copy, move, create, delete, and organize files from the Bash shell.
- **Get help in Red Hat Enterprise Linux**
  Resolve problems by using local help systems.
- **Create, view, and edit text files**
  Create, view, and edit text files from command output or in a text editor.
- **Manage local users and groups**
  Create, manage, and delete local users and groups, and administer local password policies.
- **Control access to files**
  Set Linux file-system permissions on files and interpret the security effects of different permission settings.
- **Monitor and Control services and daemons**
  Control and monitor network services and system daemons with the systemd service.
- **Configure and secure SSH**
  Configure secure command-line services on remote systems with OpenSSH.
- **Analyze and store logs**
  Locate and accurately interpret system event logs for troubleshooting purposes.
- **Manage networking**
  Configure network interfaces and settings on Red Hat Enterprise Linux servers.
- **Archive and transfer files**
  Archive and copy files from one system to another.
- **Install and Update Software Packages**
  Download, install, update, and manage software packages from Red Hat and DNF package repositories..
- **Access Linux file systems**
  Access, inspect, and use existing file systems on storage that is attached to a Linux server.

Date: July-19th-2025

Core Technologies and Certifications

Practical Linux Training

Students receive hands-on Linux experience, preparing for **RHCSA certification** and foundational IT skills essential for the industry.

Cloud Technology Training

Training includes AWS Cloud readiness and skill development, helping students secure competitive IT roles.

Automation With Ansible and Scripting

Students learn Ansible and scripting to automate IT tasks, enabling remote work and meeting high job market demand – **RHCSE Certification**

July

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	Linux Class Start-Date				19
20	21	22	23	24	25	26
27	28	29	30	31		

August

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

September

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	Linux Class End-Date					

October

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

KEY / LEGEND

19-July - 2025 Orientation

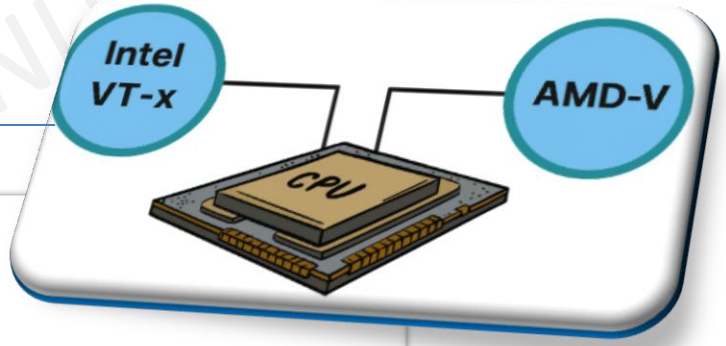
- Regular Classes / Modules
- Lab work / Practical's / Quiz / TBD
- Ready for Job - TBD

NOTE:
All other days will be for collaboration and discussion using "DISCORD CHANNEL"

VIRTUALIZATION SUPPORT

Date: July-19th-2025

Laptop (Windows or MAC)



Where This is Used:

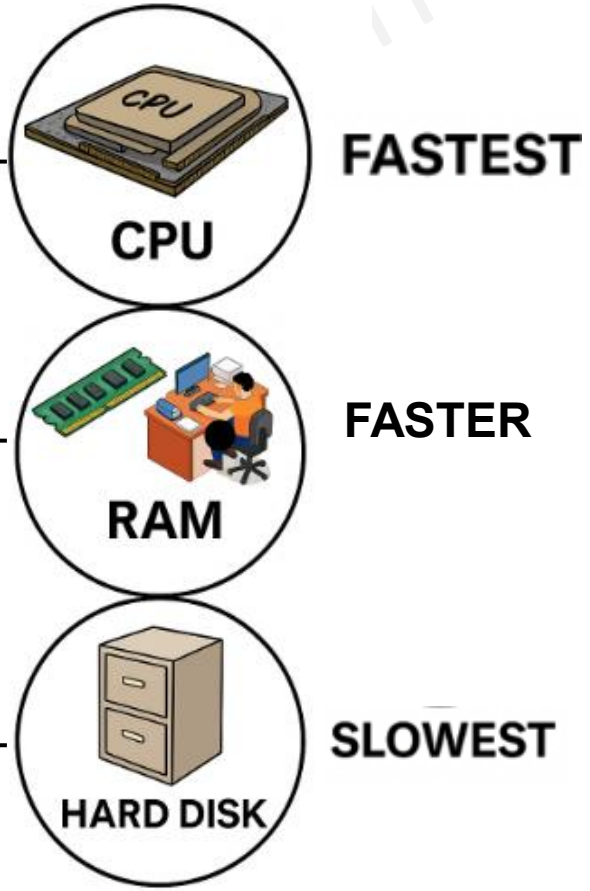
You'll see this label on:

- System requirements for installing **virtualization platforms**
- Requirements for running **64-bit operating systems** in virtual machines
- BIOS/UEFI settings where **VT-x/AMD-V needs to be enabled** for VM support

Inside the Computer: How Speed Affects Performance

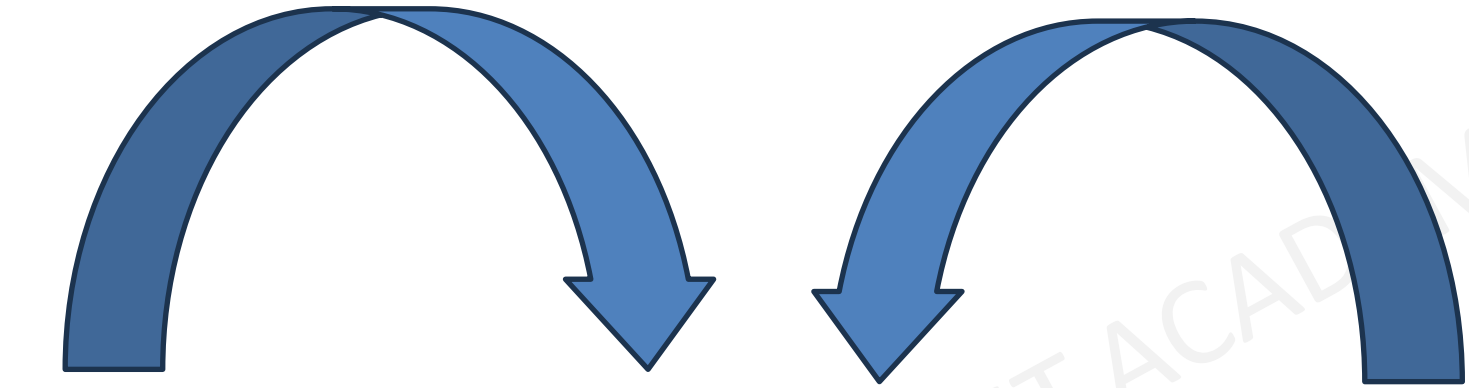
Date: July-19th-2025

Laptop (Windows or MAC)



VIRTUAL MACHINE REQUIREMENTS

Date: July-19th-2025



ORACLE VIRTUAL BOX

Laptop w/Windows

1 CPU

1GB RAM

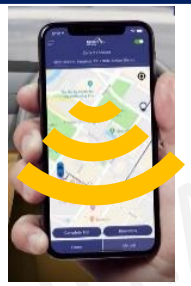
ISO Images:
RHEL 9
Rocky Linux 9

20GB Free Space

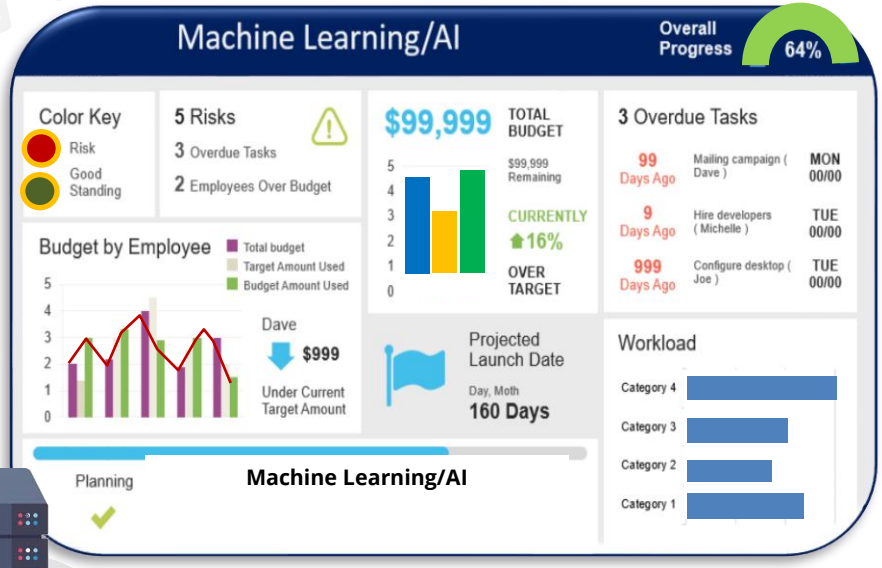
UBER/LYFT – HOW TECHNOLOGY WORKS

Date: July-19th-2025

1 Rider plans and initiates Request for Trip using Uber/Lyft APP



2 Uber/Lyft Command Control Center receives & processes request => SYS CALL

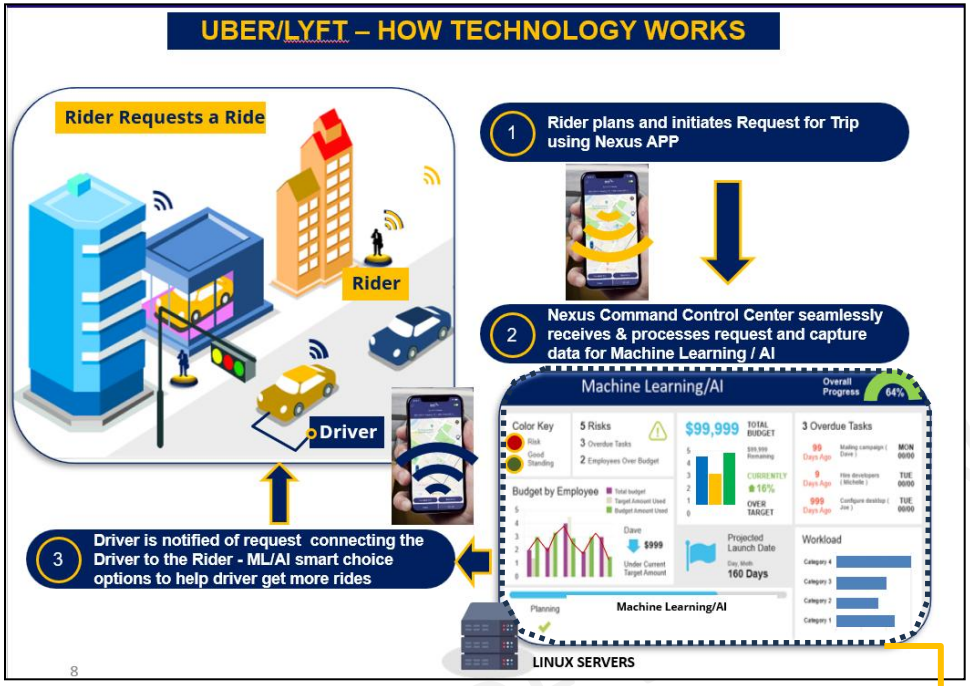


LINUX SERVERS

3 Driver is notified of request connecting the Driver to the Rider

WHY LEARN LINUX?

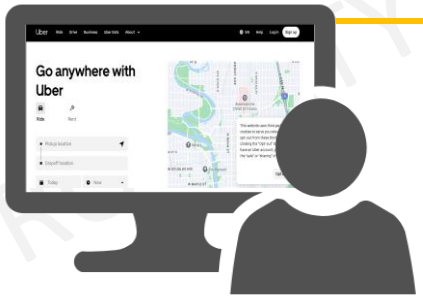
Date: July-19th-2025



(Linux System Admin)

App

App Servers



Web Servers

Linux Servers

FAIR STUDENT AGREEMENT

Date: July-19th-2025

A FAIR STUDENT AGREEMENT SHALL BE PROVIDED IN THE MONTH OF SEPTEMBER TO ALL STUDENTS.

THOS WHO WANT US TO HELP THEM WITH RESUME AND JOB INTERVIEWS WILL HAVE TO SIGN THE DOCUMENT.

YOU OWE US $6000 ONCE YOU HAVE YOUR JOB. PLEASE HONOR AND RESPECT OUR HARD WORK BY HONORING THE AGREEMENT ONCE YOU GET YOUR JOB.

Achieve IT Career Success

Resume Tailored for IT

- Receive expert help to create a resume that showcases your IT skills and experience for technology-focused roles.

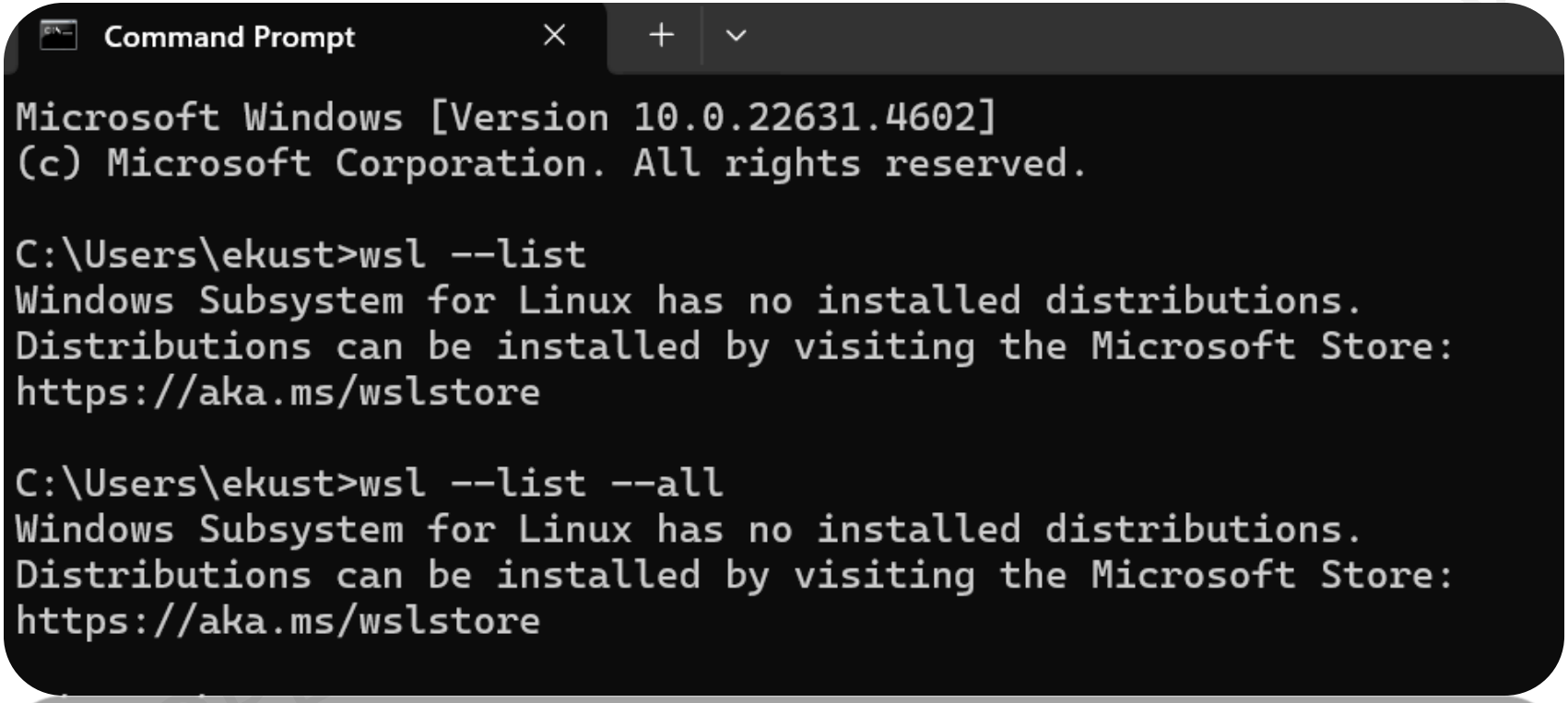
Interview Preparation Support

- Get prepared for IT job interviews with tailored practice, boosting your confidence and performance in interviews.

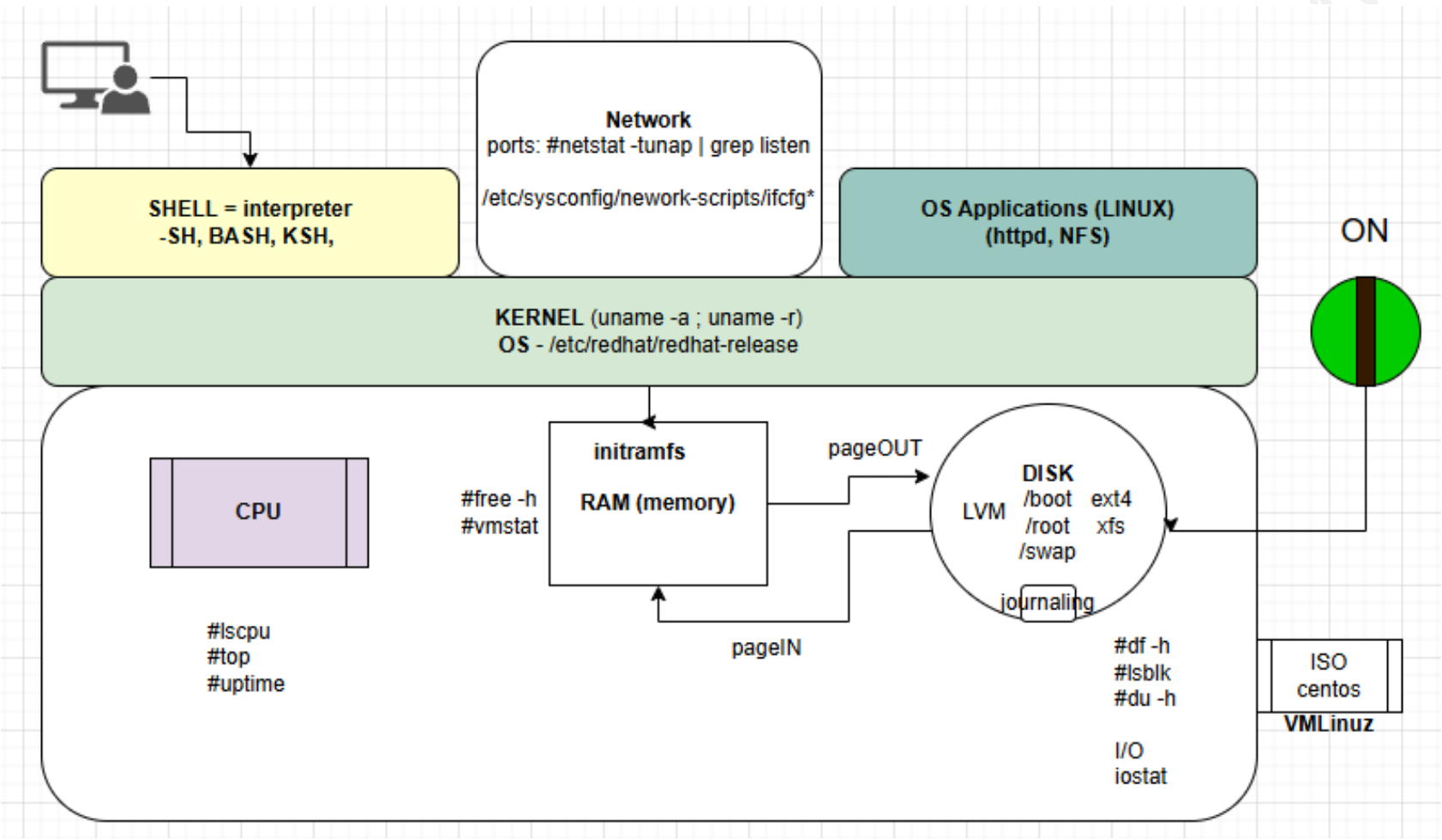
Job Search Strategies

- Learn proven strategies for finding and securing your desired IT job, increasing your chances of employment success.

WINDOWS ADOPTING LINUX!



OPERATING SYSTEM (OS):
USER MANAGEMENT
DISK (FILE) MANAGEMENT
MEMORY MANAGEMENT
PROCESS MANAGEMENT



INTRODUCTION TO SERVERS/VIRTUALIZATION

2

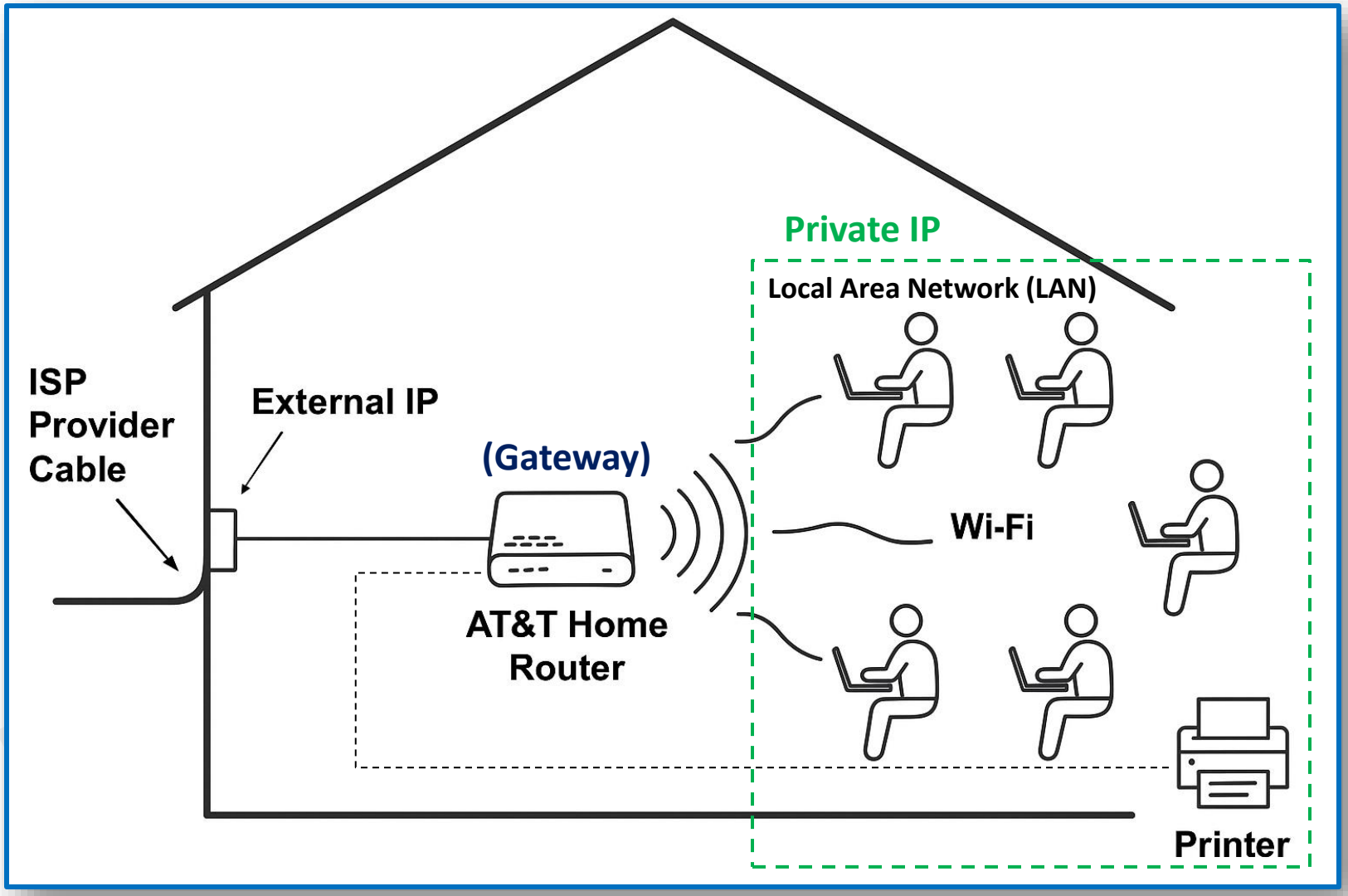
CHAPTER 2
INTRODUCTION TO SERVERS/COMPUTERS

Using the rules and strategies in this section you will gain confidence and a deep insight in setting up yourself for success

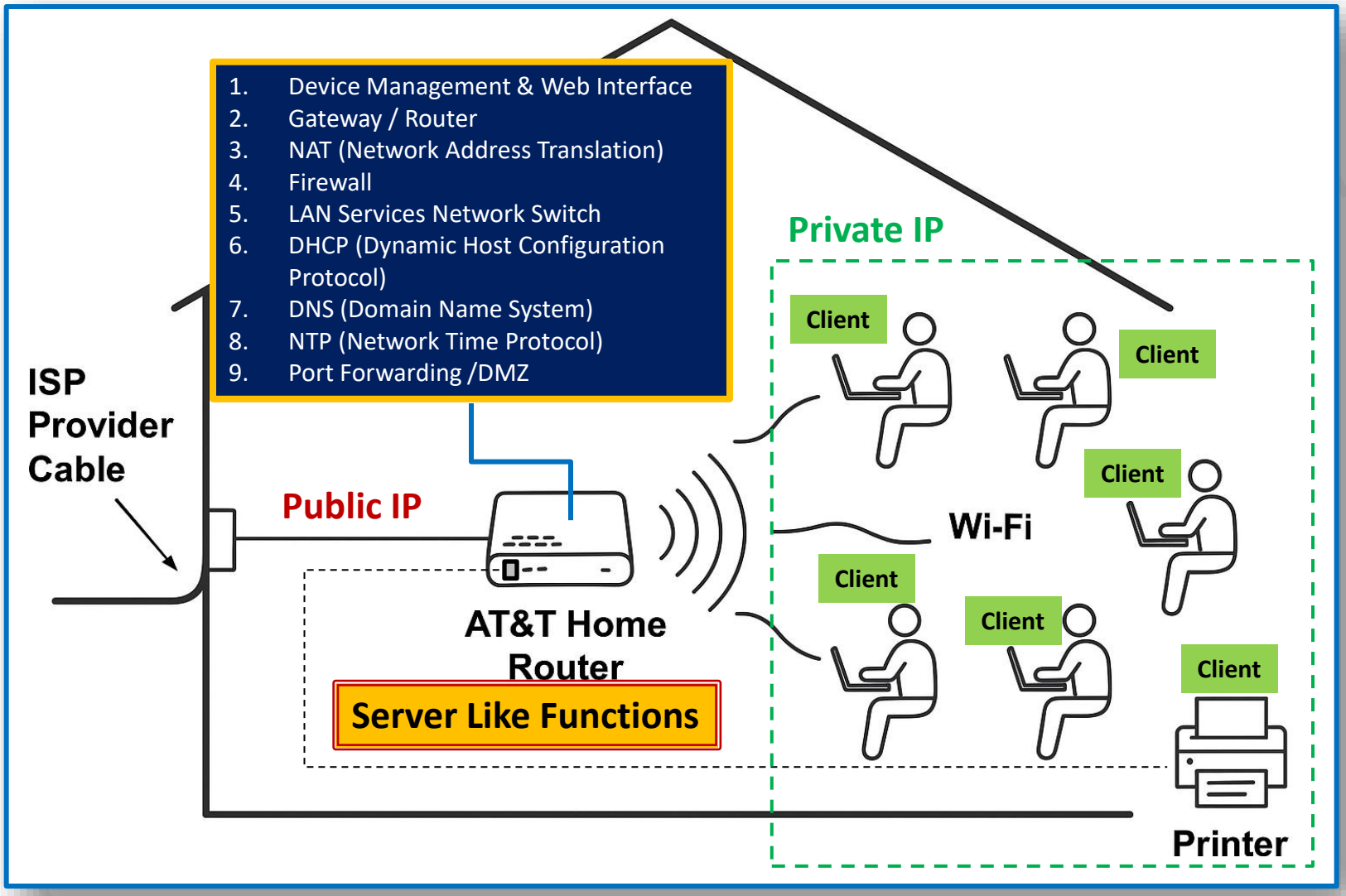
- Basics of Networking?
- Data Center and Servers?
- What is a Server?
- Types of Operating Systems
- Virtualization
- Linux Distros.
- DevOPS

BASIC NETWORK - HOME

A home AT&T router is **primarily a network device**, not a general-purpose server. However, it may provide **limited server-like functions** for network management.



WHERE IS THE SERVER?



1. Internet Gateway (WAN to LAN translation)
•Connects your home network to the AT&T internet backbone via fiber or DSL.
•Handles **external IP** on WAN (public) side and **local IPs** on LAN side.

2. Wi-Fi Access Point
•Provides **wireless connectivity** (Wi-Fi 2.4GHz and 5GHz or 6GHz on newer models).
•Allows devices like phones, laptops, smart TVs, and tablets to access the network.

3. DHCP Server
•Assigns **dynamic IP addresses** to devices in your home (e.g., 192.168.1.x).
•Manages IP leases and device identification.

4. NAT (Network Address Translation)
•Allows multiple home devices to share **one public IP**.
•Keeps internal IPs hidden from the outside world.

5. Firewall
•Filters incoming/outgoing traffic for **security**.
•Blocks unsolicited inbound traffic unless port forwarding or DMZ is configured.

6. DNS Forwarding
•Forwards DNS requests to AT&T's or user-specified DNS servers.
•Speeds up domain name resolution within the local network.

7. Port Forwarding / DMZ
•Enables hosting of services like game servers or security cameras.
•Allows custom rules for exposing specific ports to the internet.

8. LAN Services (Ethernet Switching)
•Has multiple **Ethernet ports** for connecting printers, PCs, or smart TVs.
•Functions as a basic **network switch**.

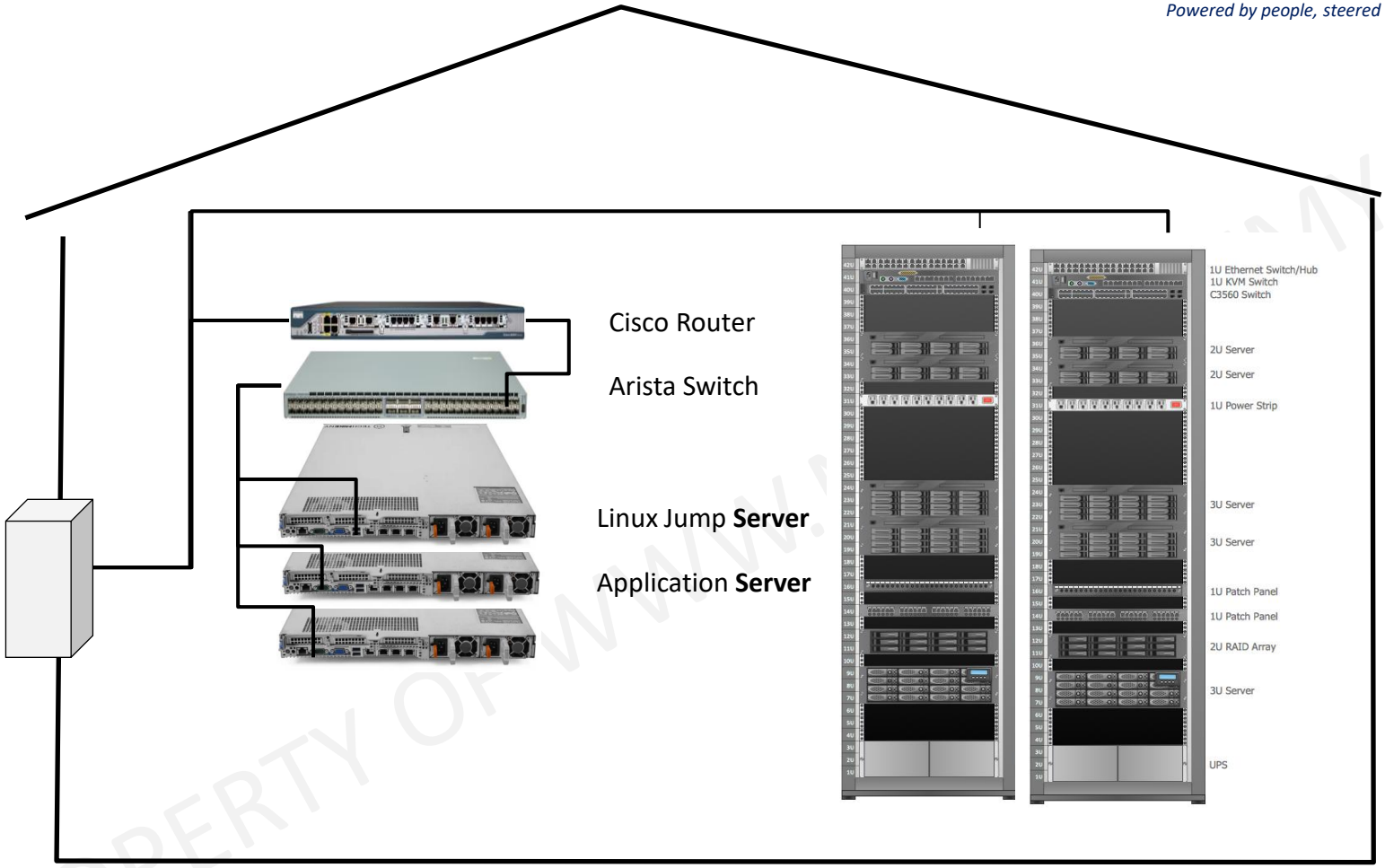
9. Parental Controls & QoS (limited)
•Some AT&T routers offer:
- •Content filtering
- •MAC filtering
- •**Bandwidth prioritization** (QoS) for devices like VoIP phones or gaming consoles

10. Device Management & Web Interface
•Accessed via http://192.168.1.254 (default)
•Lets you:
- •**View connected devices**
- •Change Wi-Fi settings
- •Update firmware
- •Monitor usage and errors

11. VoIP (if supported)
•Some AT&T routers provide ports for **digital home phones**.
•Uses AT&T's VoIP service over the internet.

DATA CENTER

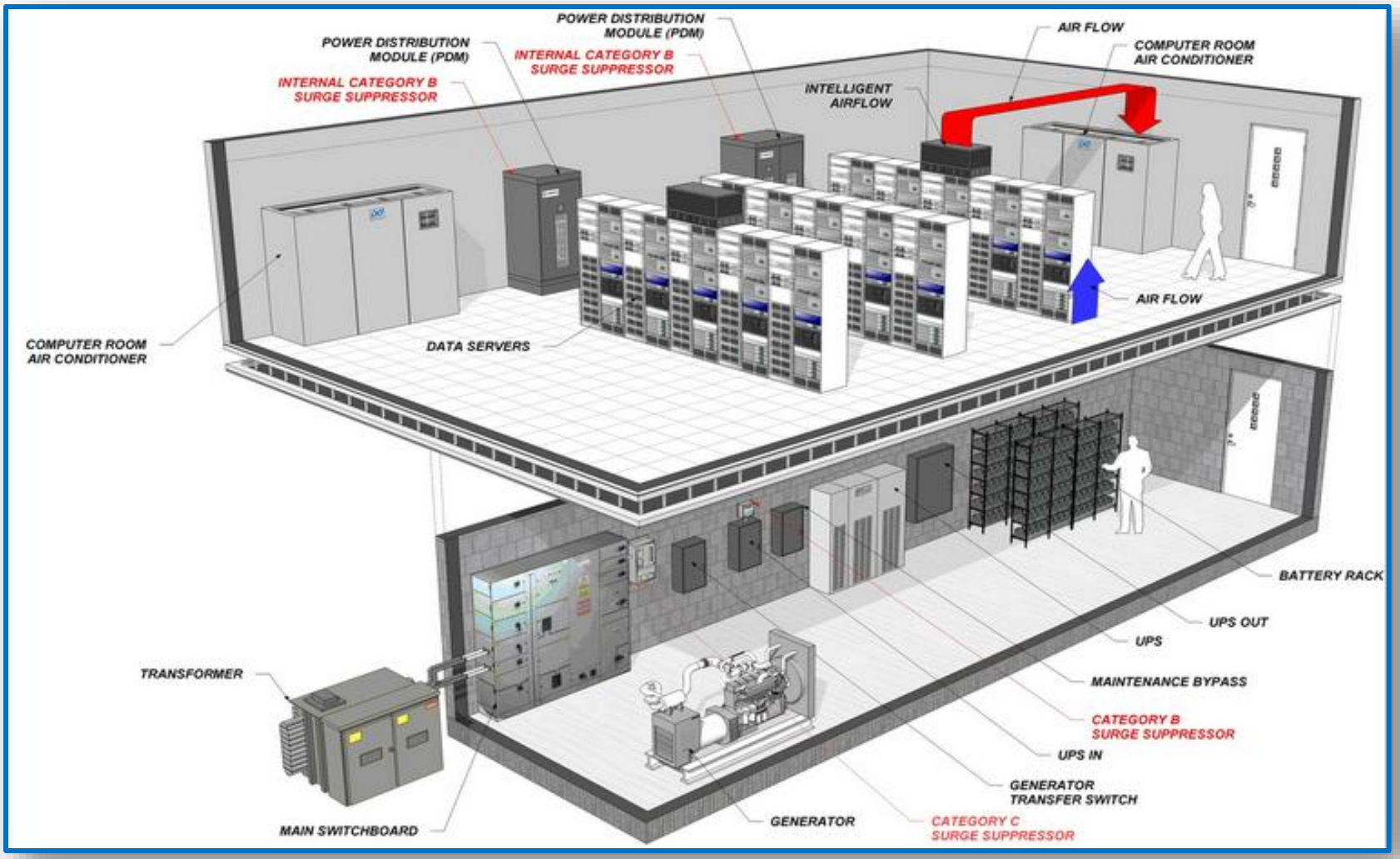
Powered by people, steered by safety™



A **server** is a device or **software** that provides **specific services** to clients on a network — such as:

1. File hosting (like an FTP server)
2. Web hosting (like Apache or Nginx)
3. Email services
4. Media streaming (like Plex),
5. DNS or DHCP services (beyond basic routing)

DATA CENTER (DC) = AWS = AZURE



COMPUTER = SERVER

Important Questions

- What is a Computer ?
- What is a Server ?

Server vs. Laptop: Key Differences

Performance and Reliability

Servers are built for continuous operation, high reliability, and to manage multiple user requests simultaneously.

Portability and Design

Laptops are designed for portability, featuring built-in displays, keyboards, and batteries for mobile computing.

Hardware and Storage Capacity

Servers typically offer more powerful hardware and greater storage capacity than laptops, supporting demanding workloads.

UNDERSTAND THE DIFFERENCE

LAPTOP / DESKTOP COMPUTER

SERVER

Aspect	Computer	Server
Purpose	Personal tasks	Serving resources to clients
Hardware	Consumer-grade	Server-grade (ECC RAM, redundancy)
Software	Desktop OS	Server OS
Usage	Individual	Multi-client
Uptime	Not designed for 24/7 operation	Designed for continuous operation

In short, a server is a specialized type of computer with hardware and software optimized for providing services reliably and efficiently over a network.

Server Overview

https://techmikeny.com/products/dell-poweredge-r640-10-12-bay-2-5-1u-server

HOST

MACHINE

BARE-METAL SERVER

Dell PowerEdge
R640 10-Bay 2.5"
1U Server

LINUX BOX

SERVER

What is inside a Server?

HARDWARE

1. Mechanical Parts

SOFTWARE

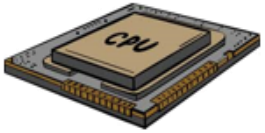
1. **Operating System**
2. Device Drivers
3. Applications /Packages / Programs

What Hardware is Important to Us?

NIC/NDC

Network Interface Card

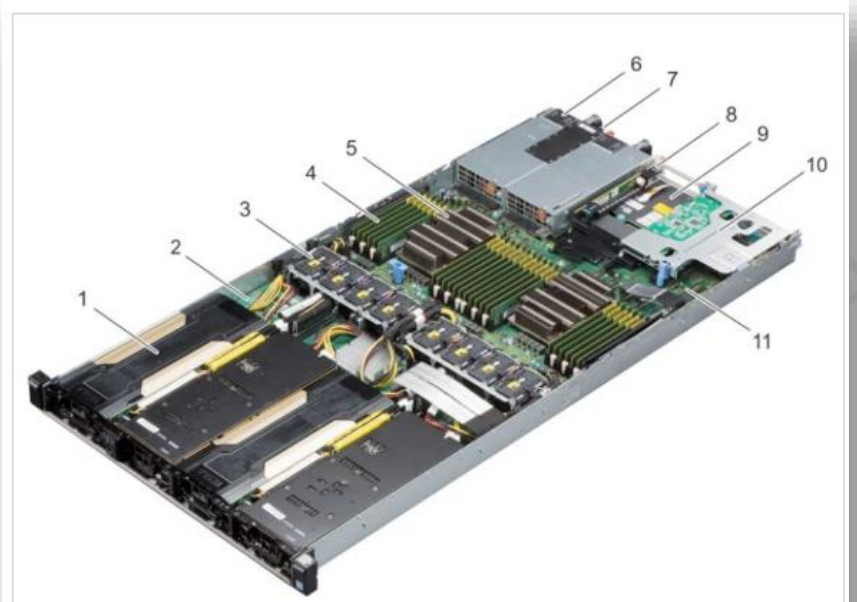
CPU



Central Processing Unit
(CPU)

Memory

Random Access Memory
(RAM)



1. PCIe GPU (4)
2. PCIe switch board
3. Cooling fan (8)
4. DIMMs (24)
5. Processor and heat sink (2)
6. PSU (2)
7. Information tag
8. Riser 2A (Low-profile PCIe expansion card: Slot 3)
9. Network daughter card (NDC)
10. Riser 1A (Low-profile PCIe expansion cards: Slots 1 and 2)
11. System board

GPU

Disk

Hard Disk
(HD)

iDRAC

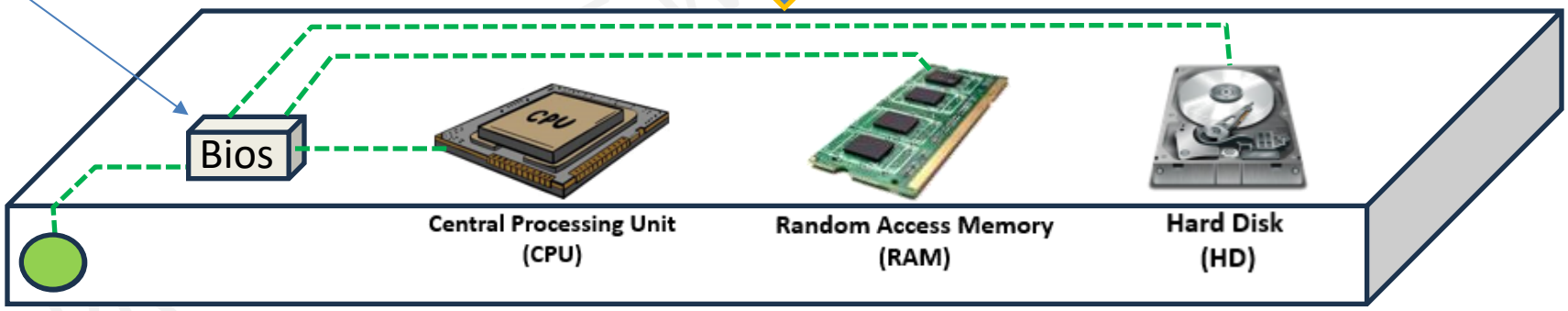
RAID

WHAT HAPPENS WHEN YOU TURN YOUR COMPUTER?

BIOS: **Input/Output System (BIOS)**: In computing, BIOS refers to a firmware interface that initializes hardware during the booting process and provides runtime services for operating systems. It's a critical component of a computer's startup routine.

Fundamental Similarities

BIOS Settings can be accessed by Powering the computer ON and pressing "ESC+F10" for HP



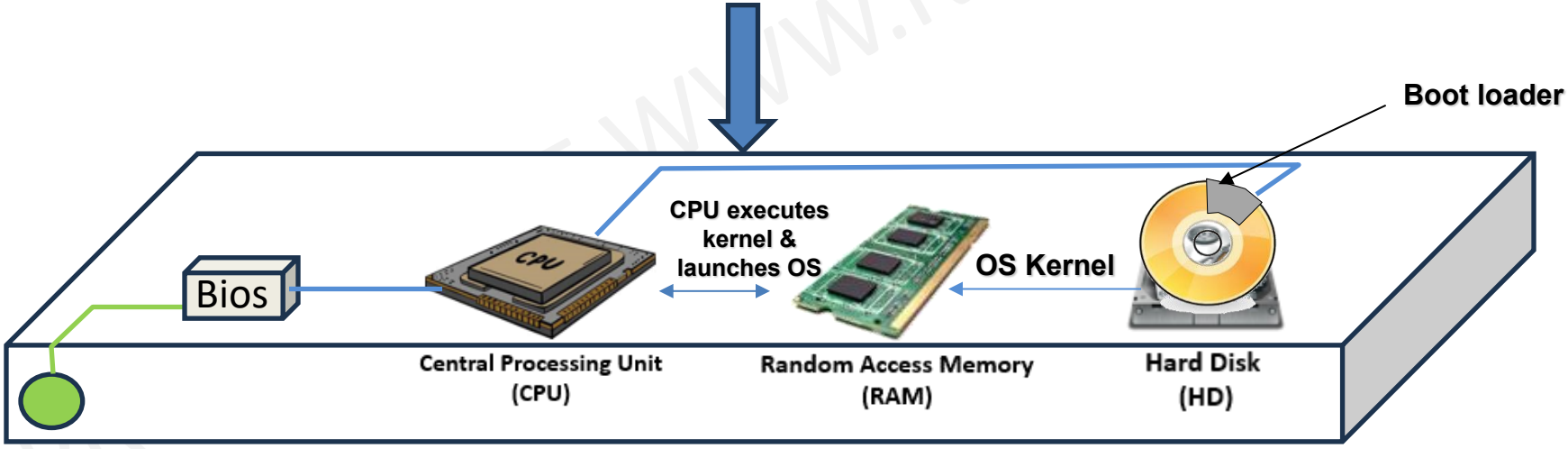
Power On Self–Test (POST): when the computer is powered ON, the BIOS performs a POST to ensure all hardware components (CPU, RAM, etc.) are functioning properly

FUNDAMENTAL DIFFERENCE BETWEEN PERSONAL COMPUTERS AND SERVERS?

Feature	Explanation
iDRAC Controller	Dell servers include a dedicated out-of-band management controller (iDRAC) to manage the system remotely—even if the OS is down. Desktops don't have this.
Advanced POST Diagnostics	PowerEdge servers perform **extensive hardware checks** (e.g., RAID controllers, ECC memory, redundant PSUs).
BIOS Boot Manager and One-Time Boot	Server BIOS includes more boot management options—booting from SAN, PXE over multiple NICs, virtual media, etc.
RAID Controllers (e.g., PERC)	Servers may boot from logical drives created by RAID controllers. RAID BIOS initialization is an extra step.
Lifecycle Controller	Dell servers include a Lifecycle Controller for firmware updates, system setup, and OS deployment—not found on desktops.
Redundancy Checks	Dual power supplies, failover NICs, ECC memory error logs—all these are validated during POST.
PXE Boot Common in Servers	Network booting using PXE (Preboot Execution Environment) is more common in data center servers for OS provisioning.

SIMPLIFIED SERVER BOOT PROCESS

Loading the Operating System



INTRODUCTION TO VIRTUALIZATION

Important Questions

- What is Virtualization ?
- Why use Virtualization ?

Single Operating System

USER (System Admin)

Application = > **processes running on a computer**

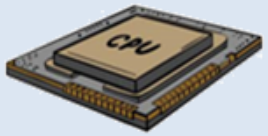
MS Word | Chrome | NodeJS | zoom — **Applications**

System Call **to?** | System Call | System Call

Running program issues request to kernel to interact with a piece of hardware

Operating System — **Kernel**

CPU

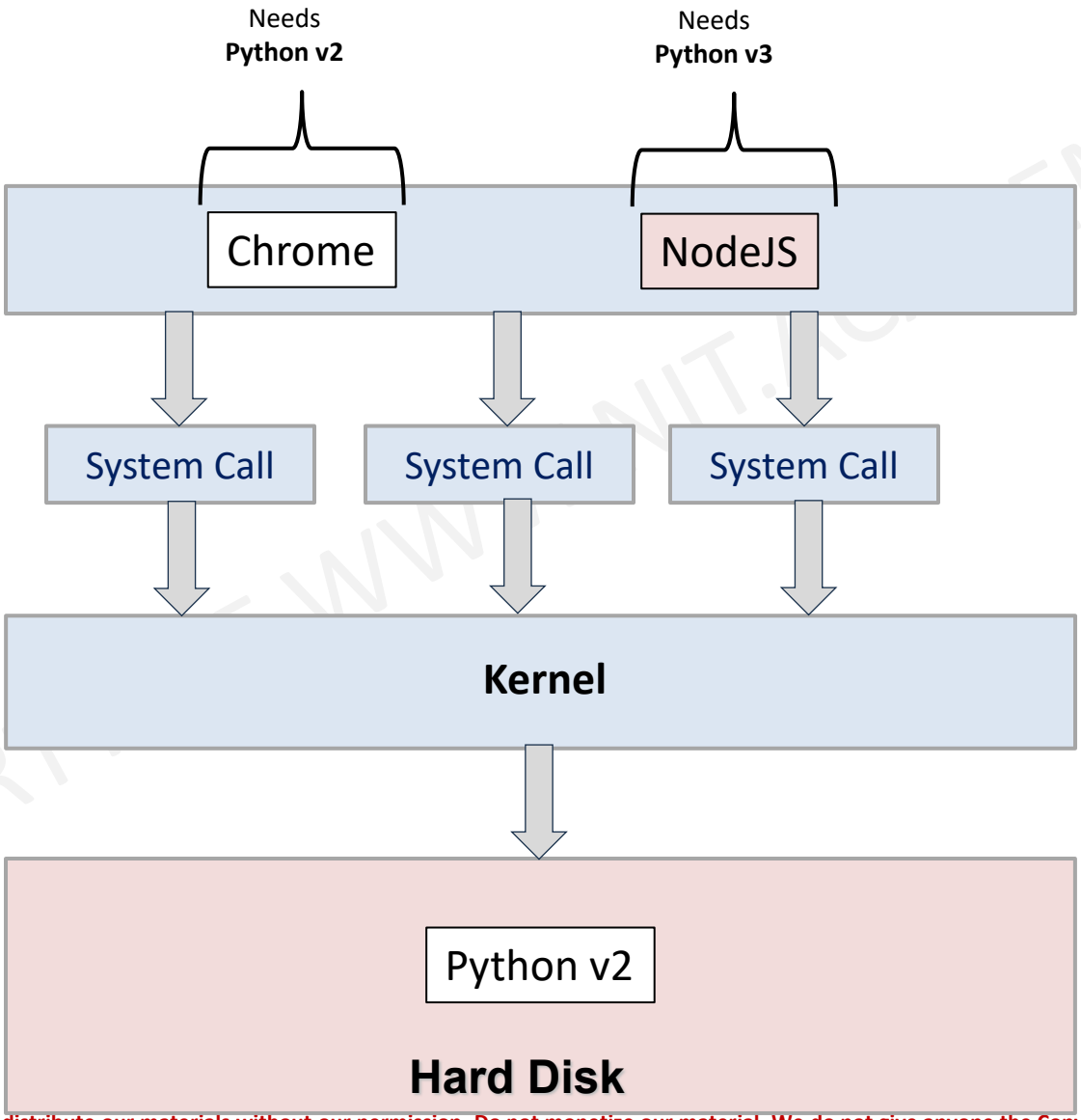


Memory

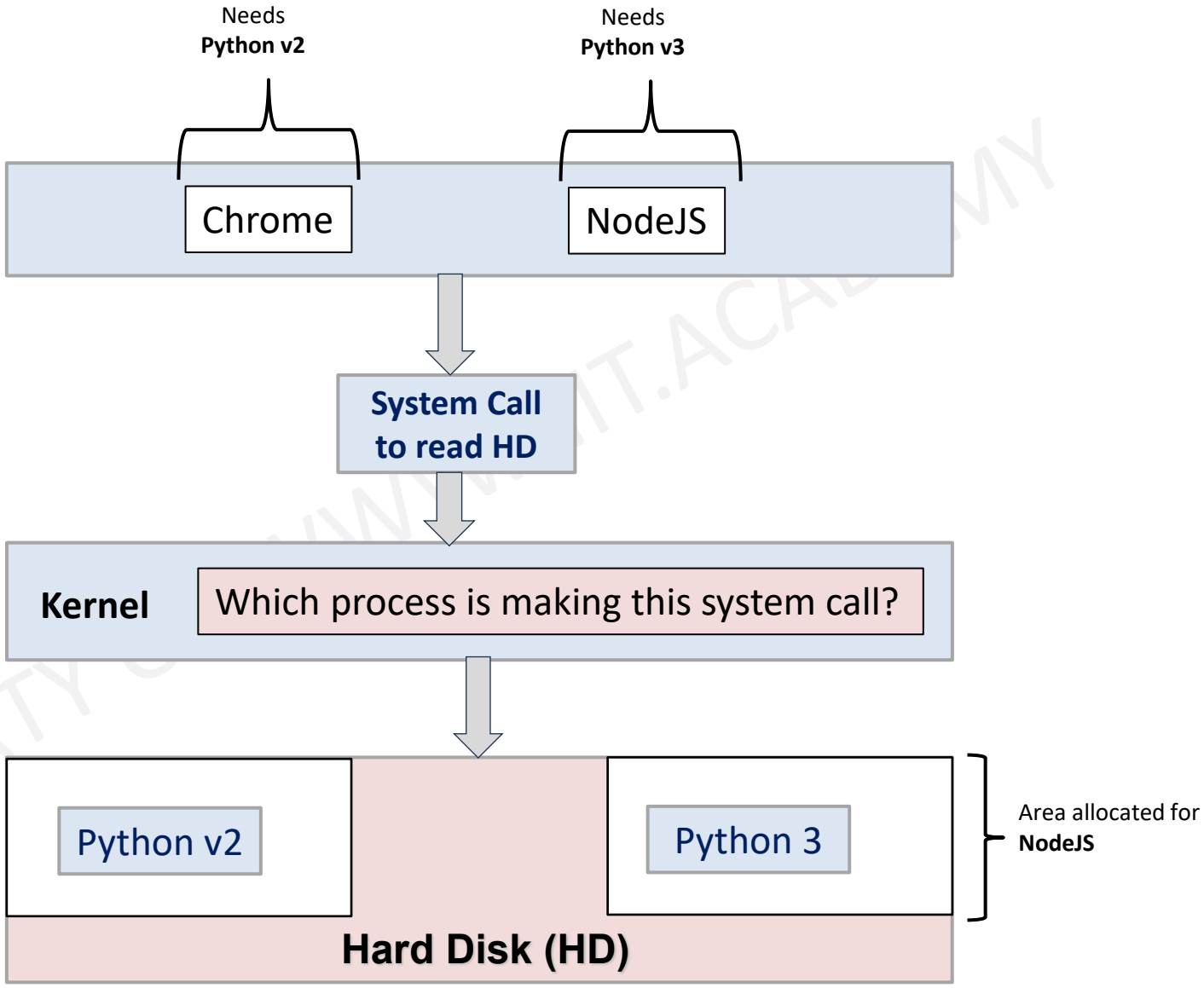
Disk

Hardware

The need for multiple operating systems



Partitioning / Virtualization



UNIX OPERATING SYSTEM (OS)

History of UNIX

1. Unix: The Beginning

• **1969**: Unix was developed at AT&T's Bell Labs by Ken Thompson, Dennis Ritchie, and others. It was created to be a simple, multitasking, multi-user operating system.

•**1971**: The first version of Unix was released. It was initially written in assembly language.

•**1973**: Unix was rewritten in the C programming language, making it portable across different types of hardware. This was revolutionary and laid the foundation for modern operating systems.

•**Late 1970s and 1980s**: Unix gained popularity in academic and research institutions. Various derivatives, such as BSD (Berkeley Software Distribution), emerged during this time.

•**1984**: AT&T divested its computing division due to antitrust regulations, leading to the commercialization of Unix. Different vendors began creating proprietary versions of Unix, leading to fragmentation.

.

UNIX/AIX OPERATING SYSTEM (OS)

History of Linux

2. Linux: Inspired by UNIX

•**1991**: Linux was created by Linus Torvalds, a Finnish computer science student. Inspired by Minix (a Unix-like system used for teaching), Torvalds started developing Linux as a free, open-source kernel.

- The first release, Linux 0.01, was shared on the Internet.
- The GNU Project, initiated by Richard Stallman in 1983, provided essential tools and utilities for Linux, forming what is often called "GNU/Linux."

•**1992**: Linux was licensed under the GNU General Public License (GPL), ensuring its continued freedom and openness.

3. Development and Adoption of Linux

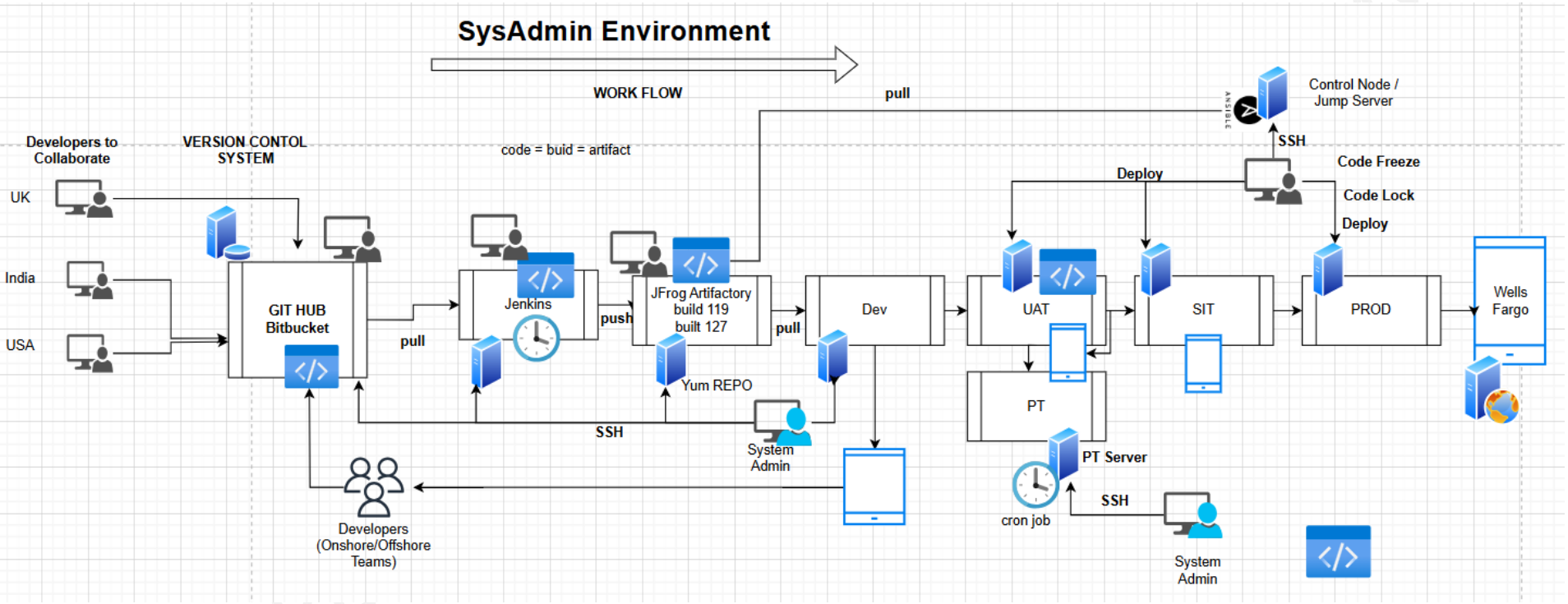
•**1990s**: Linux gained traction among hobbyists, academics, and developers. The release of distributions like Slackware (1993) and Debian (1993) made Linux easier to use and more accessible.

•**2000s**: Linux became a **dominant force in servers, supercomputing, and embedded systems. Enterprise-focused distributions like Red Hat Enterprise Linux (RHEL)** and SUSE Linux Enterprise were introduced.

•**2008**: **Android, a Linux-based operating system for mobile devices, was launched, significantly increasing Linux's presence worldwide.**

LINUX DISTROS/FLAVORS

DEVOPS - OVERVIEW



3

CHAPTER 3
Virtualization

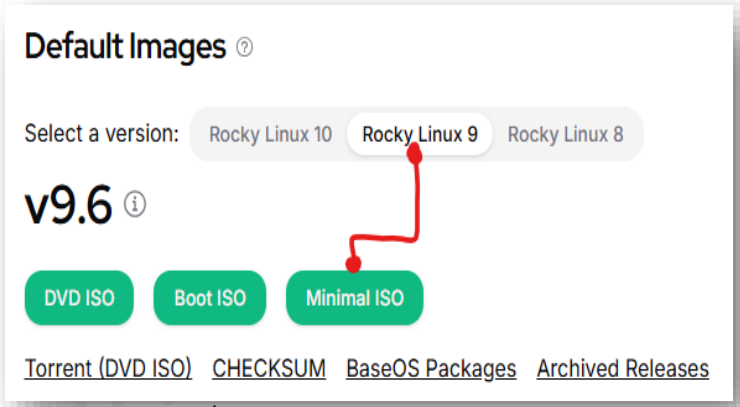
Date: July-26th-2025

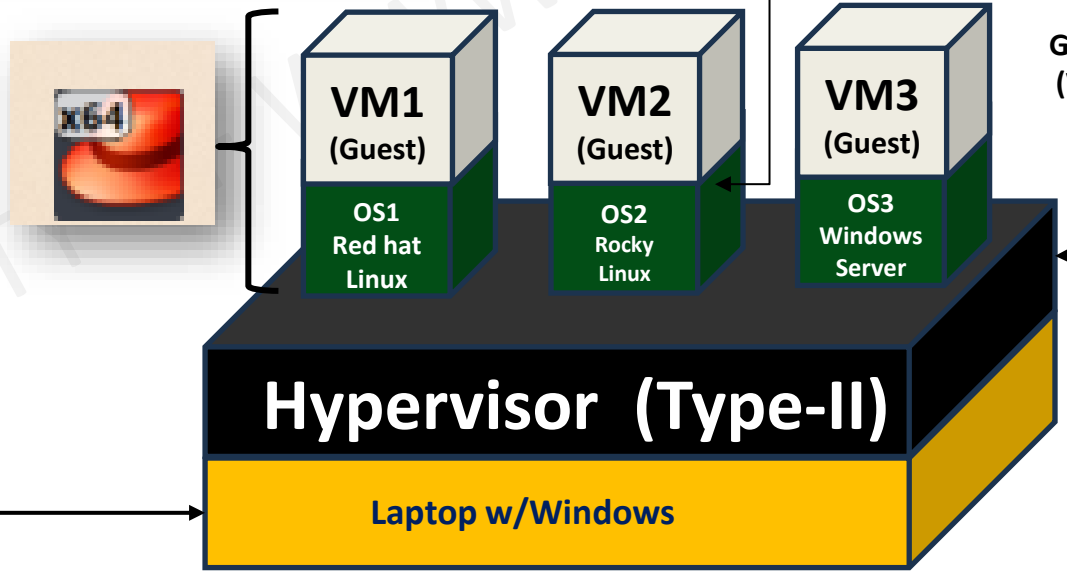
Using the rules and strategies in this section you will gain confidence and a deep insight in setting up yourself for success

- What is Virtualization ?
- Why Virtualization
- Virtual Box – What is it and why use it?
- Why do we download Redhat or Rocky linux 9 in the lab ?
- Hypervisors Type 1 and Type 2?
- Exploring Linux Commands with Shell Login

VIRTUAL MACHINE REQUIREMENTS

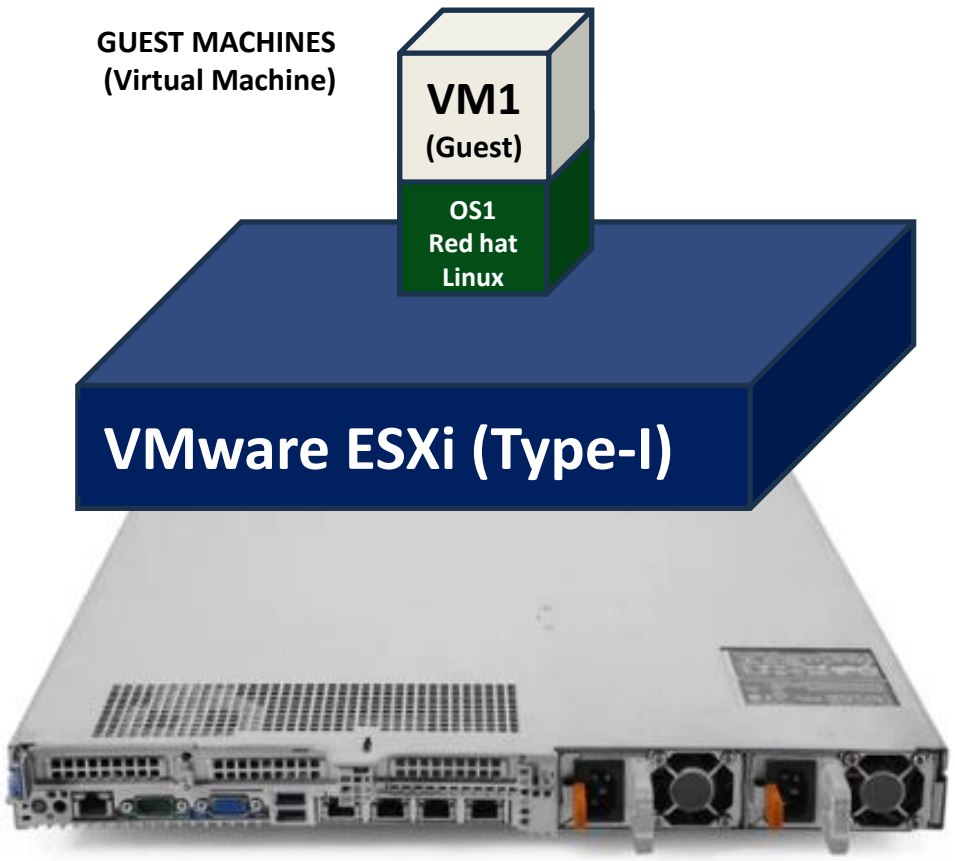
Date: July-26th-2025





VIRTUAL MACHINE REQUIREMENTS

Date: July-26th-2025



HYPERVISOR-1

Date: July-26th-2025

Run **directly on hardware** without needing a host OS. Used in enterprise environments.

Hypervisor	Vendor	Notes
VMware ESXi	VMware	Industry-leading, widely used in data centers
Microsoft Hyper-V (Server Core)	Microsoft	Built into Windows Server; native Type 1 option
Xen / XenServer (Citrix Hypervisor)	Citrix / Open Source	Used in AWS and other clouds
KVM (Kernel-based Virtual Machine)	Open Source (Linux)	Built into Linux kernel; used in Proxmox, RHEV, Nutanix AHV
Nutanix AHV (Acropolis Hypervisor)	Nutanix	Built on KVM, tailored for Nutanix HCI
Oracle VM Server for x86	Oracle	Based on Xen; being phased out
Red Hat Virtualization (RHV)	Red Hat	KVM-based; end-of-life as of 2024 (moving to OpenShift)
bhyve	FreeBSD	Lightweight hypervisor in FreeBSD
SmartOS (with KVM or Zones)	Joyent / Open Source	Hypervisor for containers and VMs

HYPERVISOR-2

Date: July-26th-2025

Run **on top of a host OS** like Windows, macOS, or Linux. Great for desktops and testing labs.

Hypervisor	Vendor	Notes
VMware Workstation	VMware	For Windows and Linux
VMware Fusion	VMware	For macOS
Oracle VirtualBox	Oracle	Free, cross-platform, widely used
Microsoft Hyper-V (Client)	Microsoft	Built into Windows 10/11 Pro (GUI mode)
Parallels Desktop	Parallels	macOS only, optimized for Mac hardware
QEMU	Open Source	Often combined with KVM; popular in embedded systems
GNOME Boxes	Linux (front-end for QEMU/KVM)	Simplified interface for running VMs on Linux

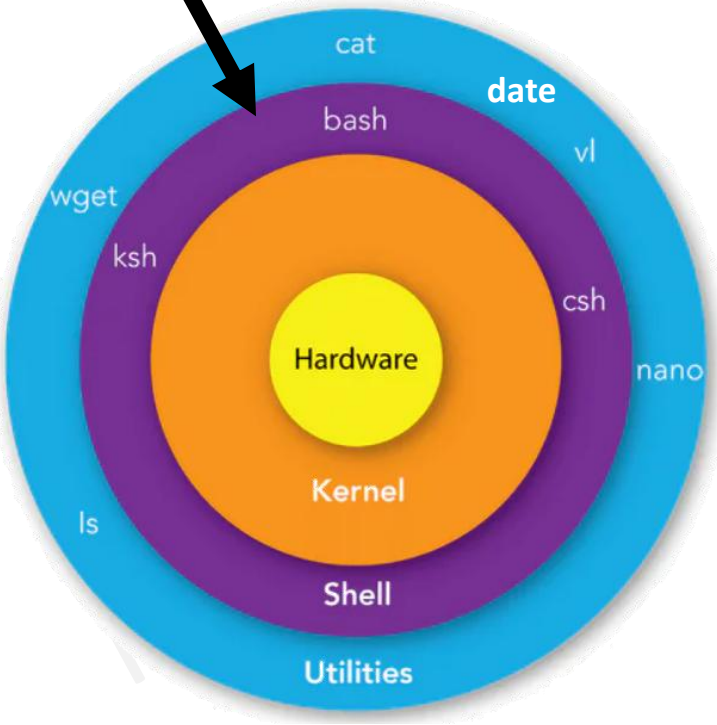
LINUX SHELL

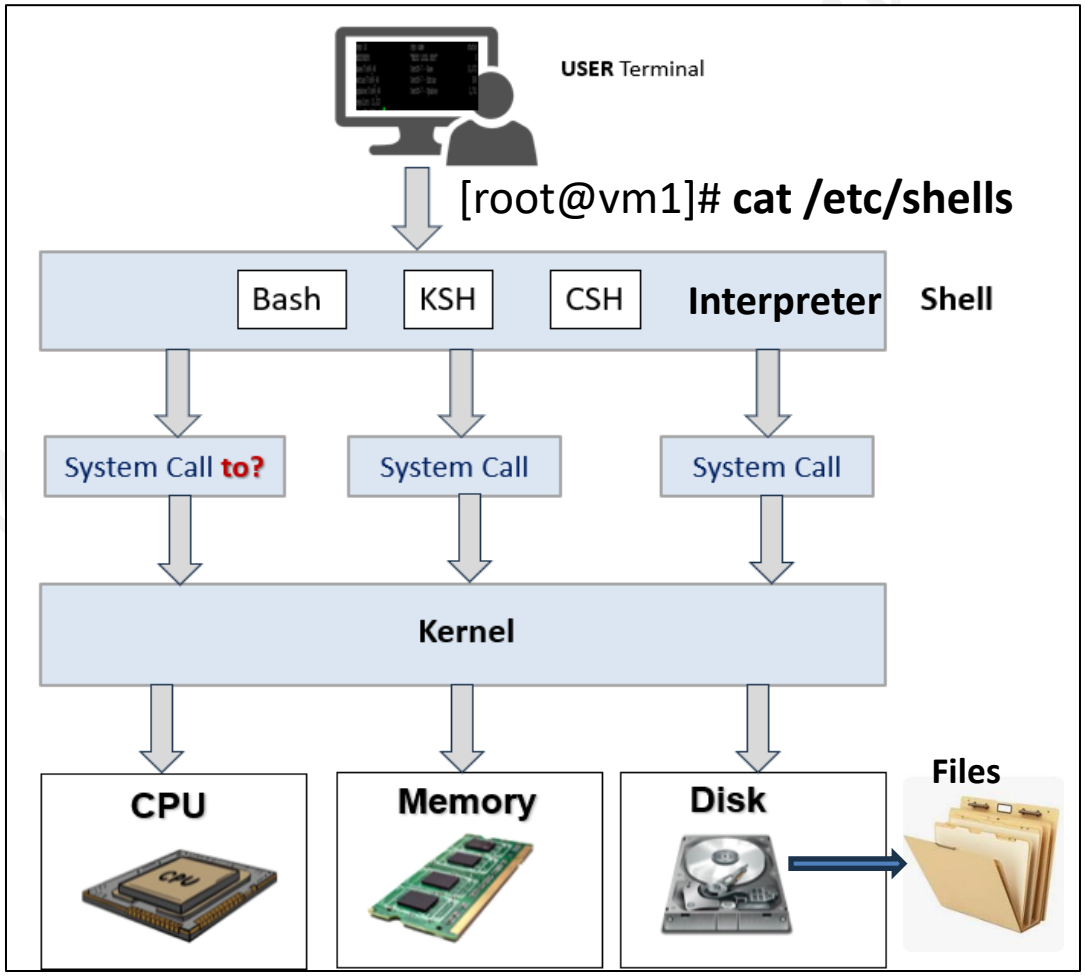
Date: July-25th-2025

Shell: Is a **Command-Line Interpreter** that connects a user to **Operating System** and allows to execute the commands

Terminal

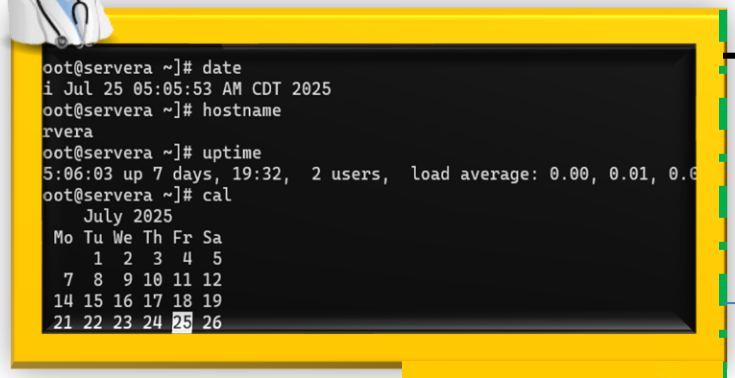
```
[root@vm1]# cat /etc/shells
[root@vm1]# echo $SHELL
[root@vm1]# ps $$
[root@vm1]# date
```





Date: July-26th-2025

LINUX OPERATING SYSTEM



Shell: Is a Command-Line Interpreter that connects a user to Operating System and allows to execute the commands

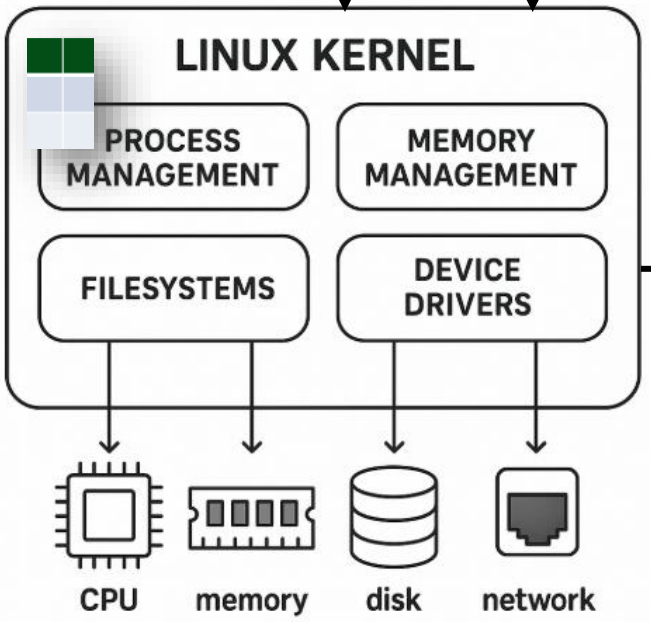
Binaries

Provide standard functions used by programs to interact with the kernel **/bin/bash /usr/bin/pwd**

Low-level core tools that do basic jobs. They're run as standalone programs

Programs and tools for system management (e.g., **systemctl, cron, useradd, journalctl**)

System Utilities



The **kernel** is the core software that directly interacts with hardware and manages system resources.

Shared Libraries

Date: July-25th-2025

LINUX LINE COMMANDS

Binaries

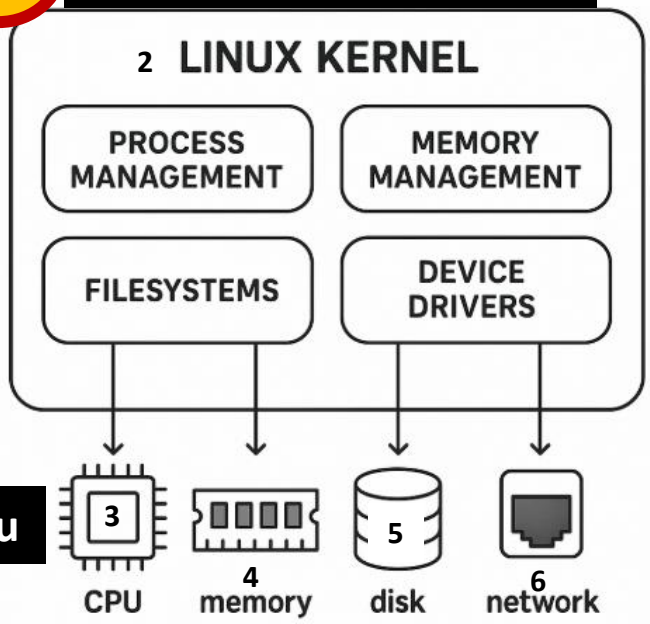
[root@vm1~]# pwd

7

1 [root@vm1~]#uname -a

```
oot@servera ~]# date
i Jul 25 05:05:53 AM CDT 2025
oot@servera ~]# hostname
rvera
oot@servera ~]# uptime
5:06:03 up 7 days, 19:32,  2 users,  load average: 0.00, 0.01, 0.0
oot@servera ~]# cal
     July 2025
 Mo Tu We Th Fr Sa
     1  2  3  4  5
  7  8  9 10 11 12
 14 15 16 17 18 19
 21 22 23 24 25 26
```

2 [root@vm1~]#uname -r



3 [root@vm1~]#lscpu

6

4 [root@vm1~]#free –m

[root@vm1~]#ip a

5 [root@vm1~]#df –h

Date: July-26th-2025

ALL SYSTEM INFORMATION

[root@myserver ~]# **uname -a**
Linux myserver 5.14.0-503.14.1.el9_5.x86_64 #1 SMP PREEMPT_DYNAMIC Fri Nov 15 12:04:32 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux

Field	Meaning
Linux	Kernel name (OS type)
myserver	Hostname of machine
5.14.0-503.14.1.el9_5.x86_64	Kernel version
#1	Kernel built version
SMP	Symmetric multiprocessing support
PREEMPT_DYNAMIC	Preemption model used by the kernel
Fri Nov 15 12:04:32 UTC 2024	Build time and date of the kernel
X86_64 (3 times)	Machine hardware name, Processor type, Hardware platform
GNU/Linux	Operating System

Date: July-26th-2025

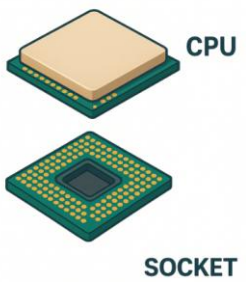
KERNEL RELEASE VERSION

[root@servera ~]# **uname -r**
5.14.0-503.14.1.el9_5.x86_64

Field	Meaning
5	Major Version of Linux Kernel
14	Minor Version of Linux Kernel
0	Patch Version of Linux Kernel
503.14.1	Red Hat-specific kernel release/patch level
el9_5	Built for Enterprise Linux 9.5 (RHEL 9.5)
x86	The original Intel CPUs (like 8086, 80286, 386, 486) ended in “86” – hence the name x86
64	AMD64 and Intel64 (Original names)
X86_64 (hardware name)	The machine is running a 64-bit processor (CPU)
X86_64 (architecture)	The kernel was compiled for a 64-bit system
X86_64 (platform)	The OS is also 64-bit capable

CPU

Date: July-26h-2025



(Actual Hardware)

vCPU

```
[root@myserver ~]# lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          39 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   1
  On-line CPU(s) list:    0
Vendor ID:                GenuineIntel
  Model name:             Intel(R) Core(TM) i3-N305
    CPU family:           6
    Model:                190
    Thread(s) per core:   1
    Core(s) per socket:   1
    Socket(s):            1
    Stepping:             0
    BogoMIPS:             3609.59
    Flags:                fpu vme de pse tsc msr pae mce cx8
```

NOTE: When **"lscpu"** or **"cat /proc/cpuinfo"** command is executed in a VM, it is **not showing the actual hardware**, but rather **what the VM is allowed to see**. Total **Logical Processors**, that is, **vCPU**

Term		Meaning	Example Value
Socket	A server with **1 CPU sockets** will have **1 chip** installed	Number of physical CPU chips	1
Core	An Intel CPU with 4 cores can handle 4 tasks **simultaneously**.	Processing units per CPU chip	4
Thread		Tasks each core can run (with SMT)	2 per core
Total Logical CPUs = `Sockets x Cores x Threads`		1 × 4 × 2 = 8	

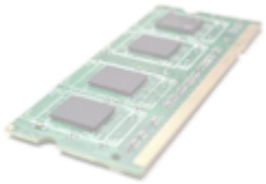
So the system has 8 logical CPUs (what `lscpu` calls `CPU(s)`), even though there's just 1 physical chip.

MEMORY

Date: July-26th-2025

Random Access Memory (RAM)

(Actual Hardware)



vRAM

RAM - Safe Allocation Rule

- **Never allocate** more than **50%–60% of your physical RAM** to a VirtualBox VM.

Example:

- If host has **16 GB**, allocate **max 8–10 GB** to a VM.
- If host has **32 GB**, allocate **max 16–20 GB**.

```
[root@myserver ~]# free -m
               total        used        free      shared  buff/cache   available
Mem:            1774         425        1021          32         508        1348
Swap:           2047           0        2047
```

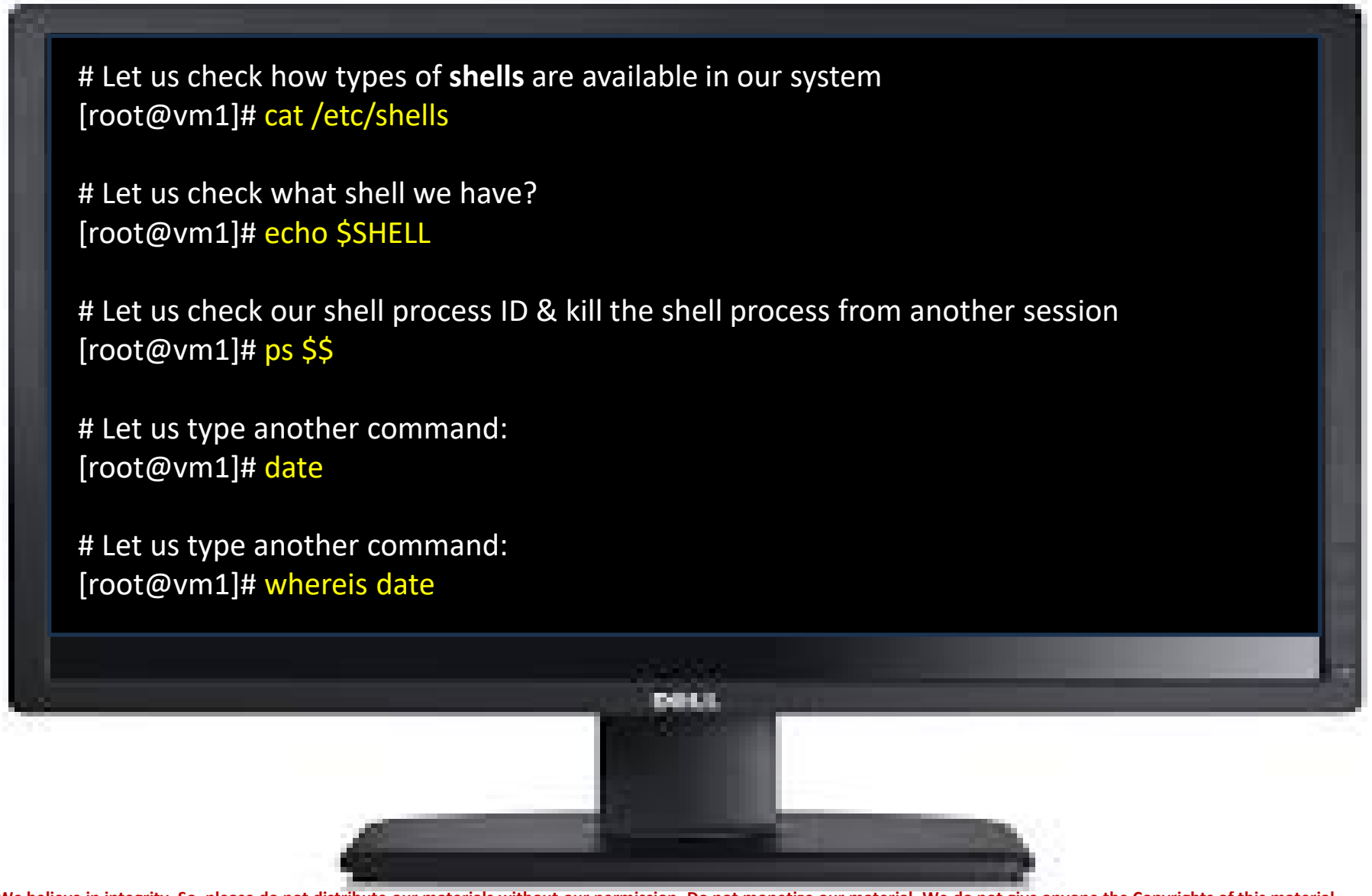
NOTE: This “1774 MB" is **vRAM**, but for the VM, it behaves like **real physical RAM**.

If your **host machine** has:

- **32 GB physical RAM**, and
- You allocate **4 GB RAM** to a VM,
- Then that VM will see 4 GB of RAM via **free –m** and use it just like real RAM — but it is **vRAM**, coming from the hypervisor's memory pool.

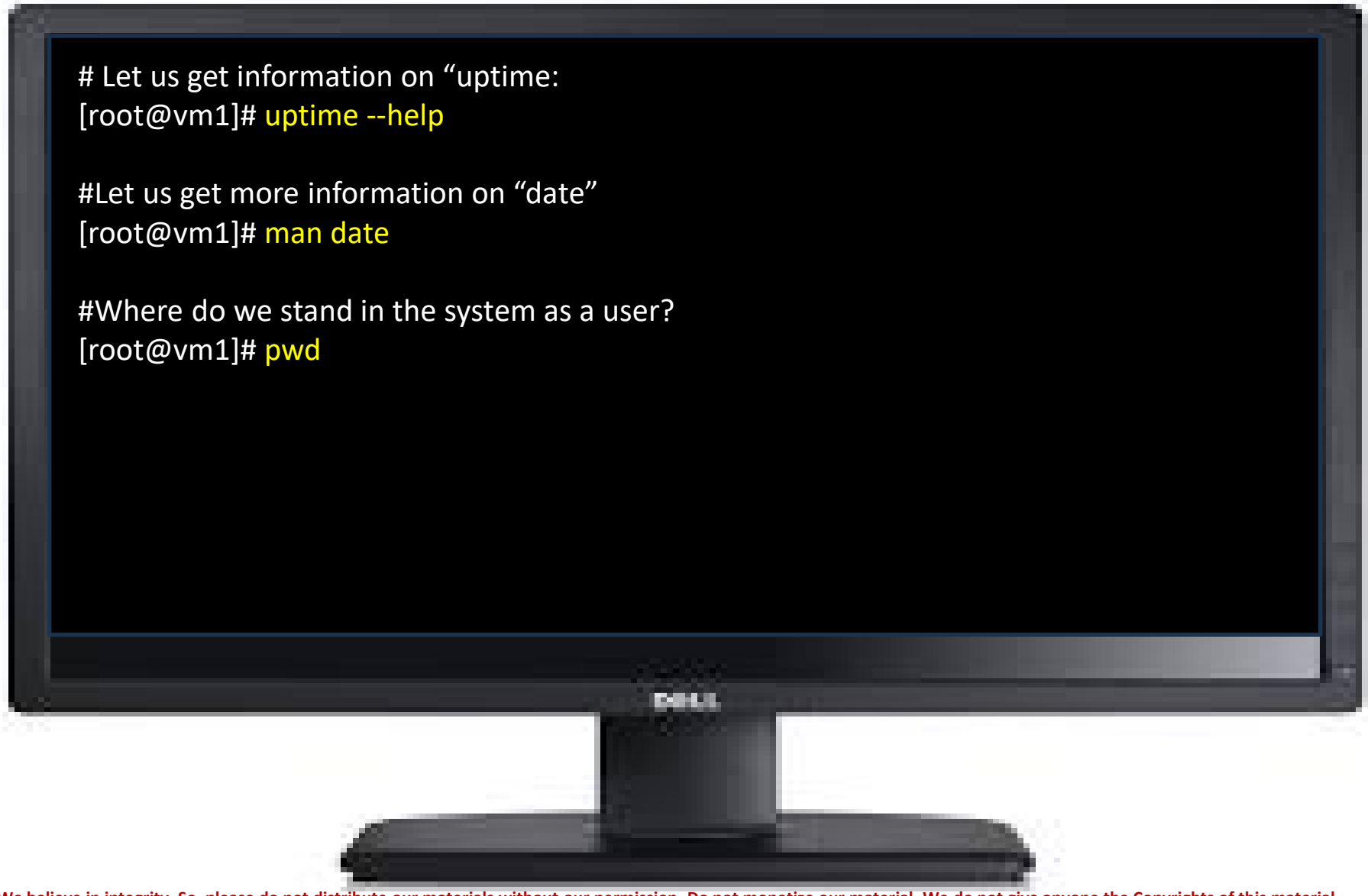
LET US EXPLORE LINUX

Date: July-26th-2025



HOW TO GET HELP IN LINUX?

Date: July-26th-2025



PROVISIONING & CONFIGURATION

Date: July-26th-2025

In **server management**:

Provisioning: refers to the initial setup and allocation of resources like hardware, operating systems, and basic software to create a functional server environment like creating a Virtual Machine.

Configuration involves customizing the settings and parameters of that provisioned server to meet specific needs, including network settings, application configurations, and user access levels; essentially, provisioning prepares the server to be used, while configuration fine-tunes it for its intended purpose.

PROVISIONING

Date: July-26th-2025

Provisioning of Virtual Machine:

- We created the basic infrastructure **(CPU, MEM, Disk Space)**
- Installed Linux operating system(s) on hypervisor **Type-2 (Oracle Virtual Box)**
- Set-up initial network connectivity **(Bridged vs NAT)**

Example of Provisioning:

We **deployed** a new virtual server (machine) with a standard Linux operating system and basic network settings.

- Enterprise Linux – Redhat
- Linux Community Based Linux Distro – Rocky Linux 9

NOTE: Once a machine is provisioned then STOPPING & STARTING that machine is called “REBOOT”

CONFIGURATION

Date: July-26th-2025

Configuration:

- We will customize settings based on specific requirements.
- We will configure application parameters.
- Manage user permissions and access.
- Adjustments to maintain desired state.

Example Configuration:

- Setting up a **STATIC IP** address and **YUM REPOSITORY**
- Configuring **Firewall Rules**
- Changing **File Permissions**

CONGRATS TO THE LEADERS!

Date: July-26th-2025

THANK YOU

Questions?

4

CHAPTER 4
Linux Initialization

Date: July-26th-2025

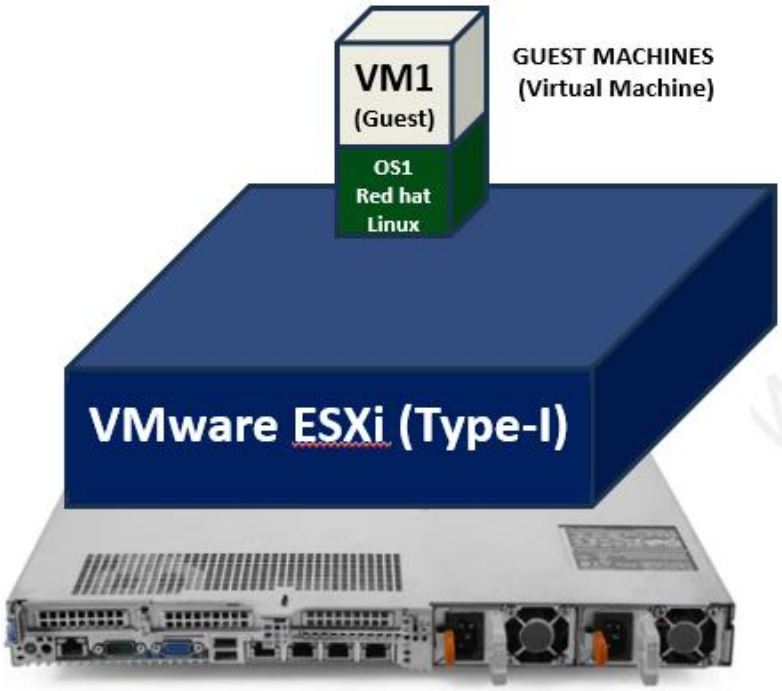
Using the rules and strategies in this section you will gain confidence and a deep insight in setting up yourself for success

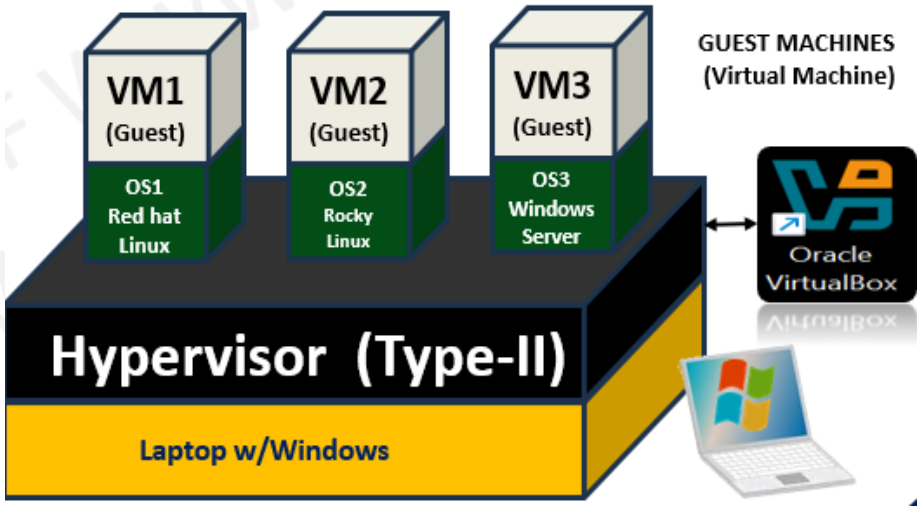
- ❑ Quick Overview of Chapter 3
- ❑ Login shell vs nologin shell
- ❑ Post login initialization
- ❑ User Home Directory
- ❑ Introduction to SSH
- ❑ Keyboard shortcuts
- ❑ Easy Bash Scripting

UNDERSTAND THE DIFFERENCE

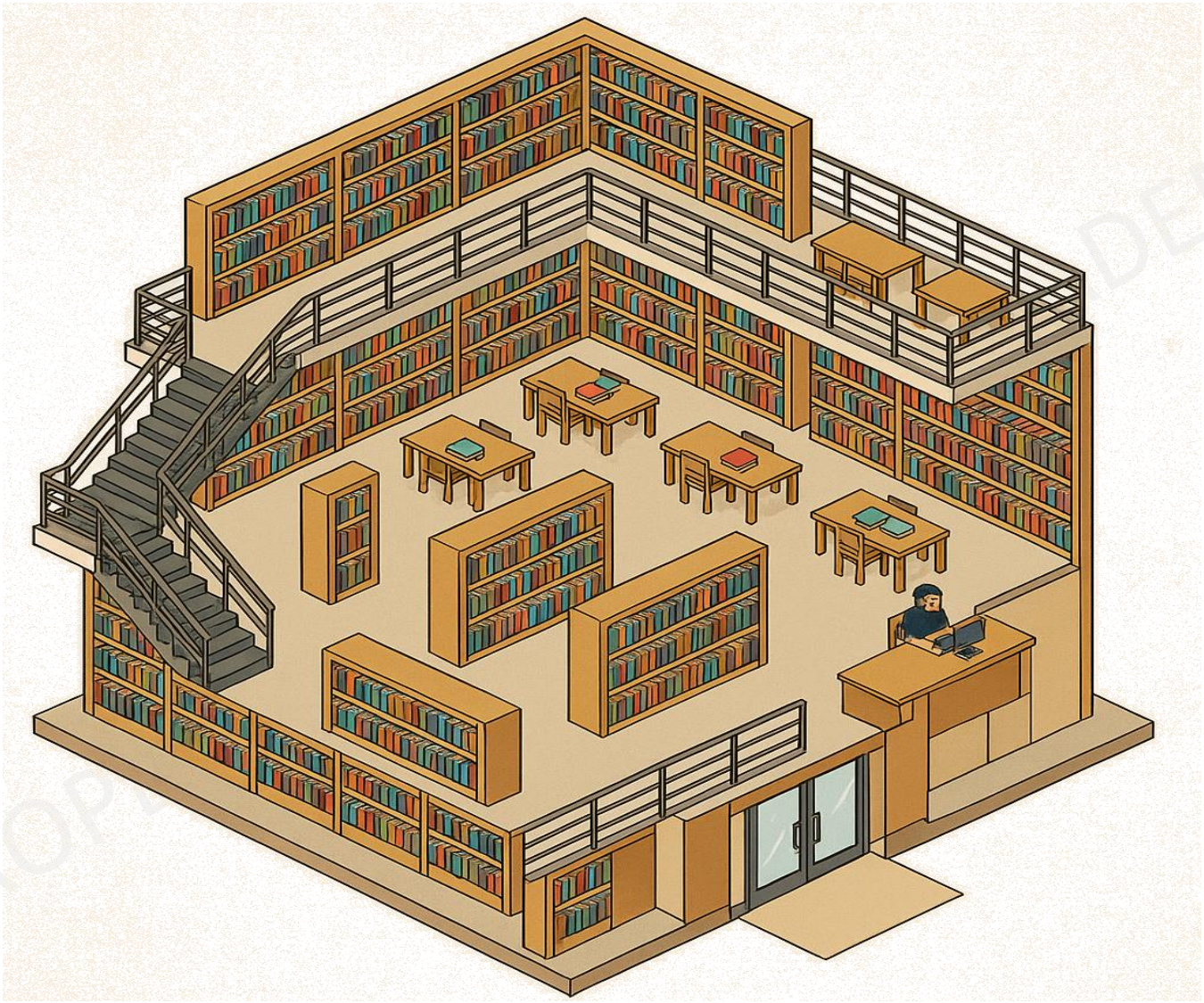
Date: July-27th-2025

Difference between Hypervisor 1 and Hypervisor 2



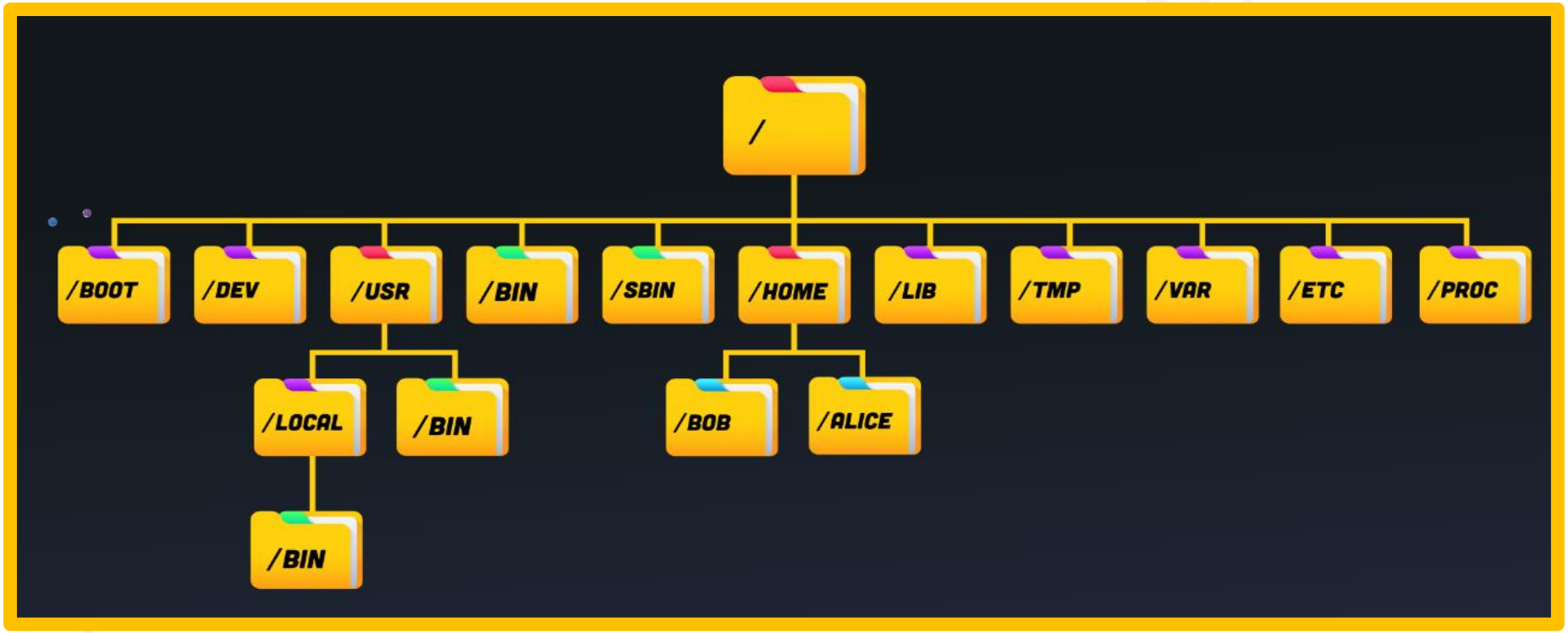


LINUX FILE SYSTEM



NAVIGATING FILES & DIRECTORIES IN LINUX

#pwd
#ls
#cd
#cat



Date: July-27th-2025

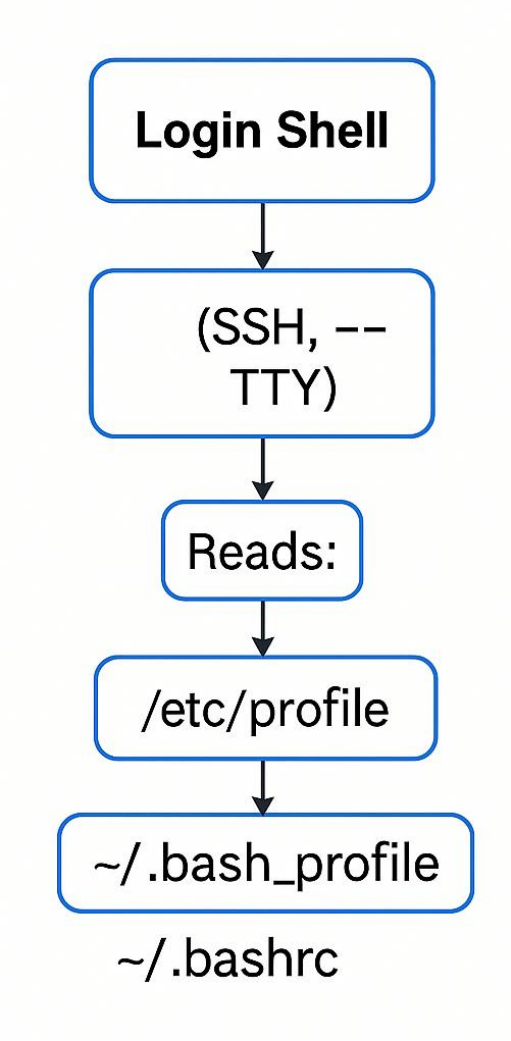
LOGIN SHELL

In essence, a **login shell** provides interactive access and a configured environment, while a **nologin interactive shell provides a limited evironment**.

Login Shell:

- A shell session started **when you log in** to the system (e.g., via SSH, virtual terminal Ctrl+Alt+F1, or console login).
- A **login shell is the first process** that executes after a user successfully logs into a system (either locally or remotely via SSH).
- It **reads configuration files**:
  - /etc/profile,
  - ~/.bash_profile,
  - ~/.profile.

Feature	Login Shell
Started via	SSH, `su -`, Console
Reads	`/etc/profile`, `~/.bash_profile`, `~/.bashrc`
Typical use case	User session on login
Example command	`su - username`

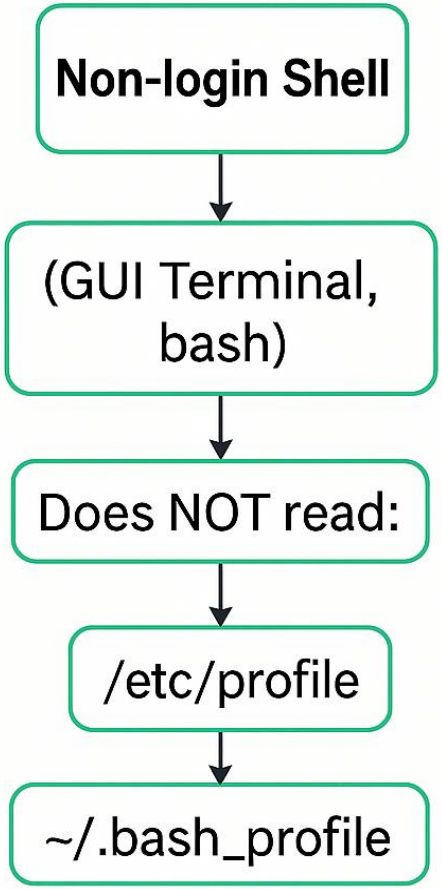


Date: July-27th-2025

In essence, a **login shell** provides interactive access and a configured environment, while a **nologin interactive shell provides a limited evironment**.

- An interactive **nologin shell** (often /sbin/nologin or /bin/false) **does not execute typical shell startup files for interactive sessions.**

Feature	Non-login Shell
Started via	Terminal in GUI, `bash`
Reads	`~/.bashrc` only
Typical use case	Running commands/scripts
Example command	`bash`



INITIALIZATION

Date: July-27th-2025

```
oot@servera ~]# date
i Jul 25 05:05:53 AM CDT 2025
oot@servera ~]# hostname
rvera
oot@servera ~]# uptime
5:06:03 up 7 days, 19:32,  2 users,  load average: 0.00, 0.01, 0.0
oot@servera ~]# cal
    July 2025
 Mo Tu We Th Fr Sa
     1  2  3  4  5
  7  8  9 10 11 12
 14 15 16 17 18 19
 21 22 23 24 25 26
```

1

Invoking Bash

1. Post Login **Bash** is launched as the shell
2. Typing /bin/bash
3. Run a shell script #!/bin/bash

/bin/bash: ELF binary (Executable and Linkable Format) compiled from C source code
Check using **[root@vm1~]# file /bin/bash**

Binaries

2

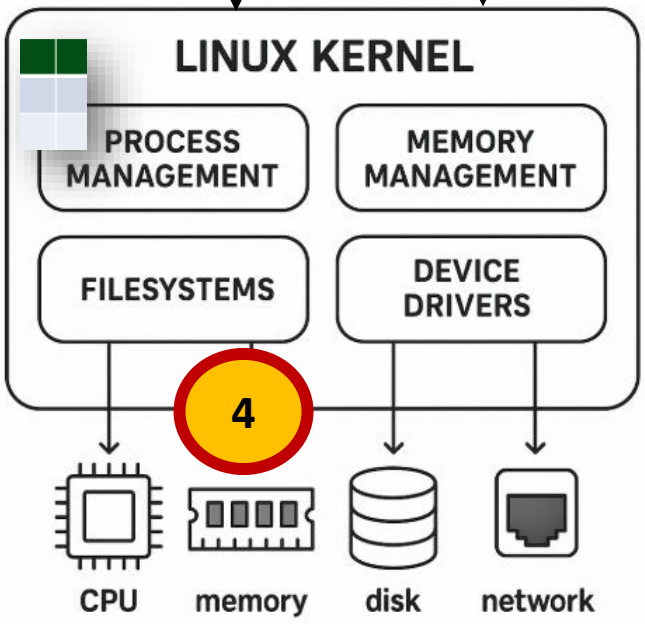
Provide standard functions used by programs to interact with the kernel **/bin/bash**

System Utilities

3

The Linux kernel sees that /bin/bash is an **ELF binary**, and uses the **executable loader** to:

1. Load program into memory
2. Set up a process in the kernel's process table
3. Link required shared files like libc.so.6

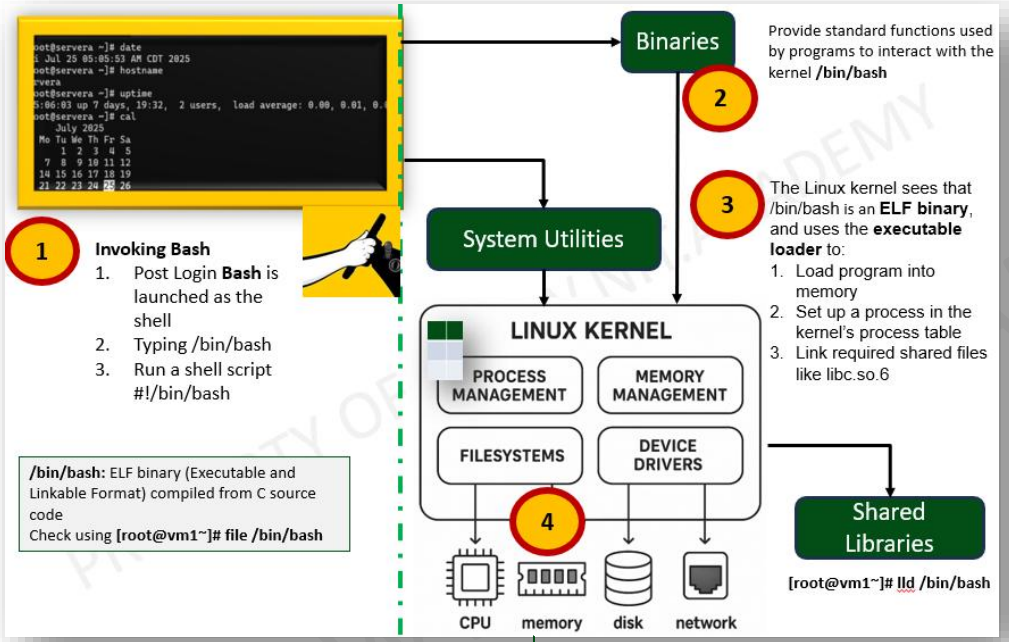


Shared Libraries

[root@vm1~]# lld /bin/bash

STANDARD I/O CONNECTED…

Initializing….
Once in memory



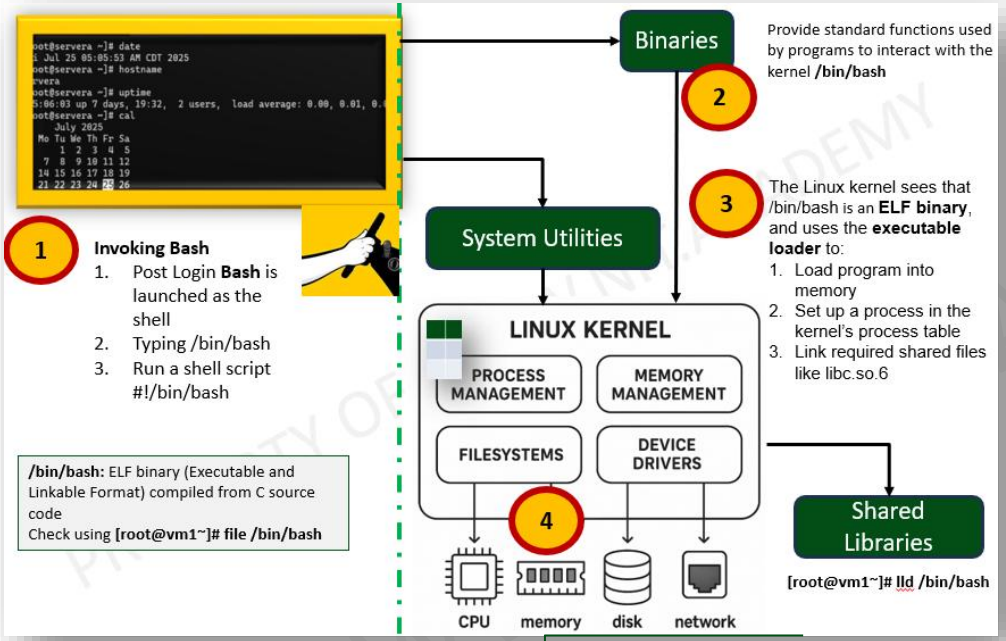
STDIN
(STANDARD INPUT)

STDERR
(STANDARD ERROR)

STDOUT
(STANDARD OUTPUT)

READS CONFIGURATION FILES…

Initializing….
Once in memory



5

[root@vm1~]# cat /etc/profile

What is this file?

1. Maintained by System Admin (and Root)
2. Information in this file **affects all users** who **log-in**
3. This file sets the shell (environmental) variables like **PATH, USER and HOSTNAME**

Commands
echo $PATH
echo $SHELL
echo $HOST
echo $USER

6

[root@vm1~]# cat /etc/bashrc

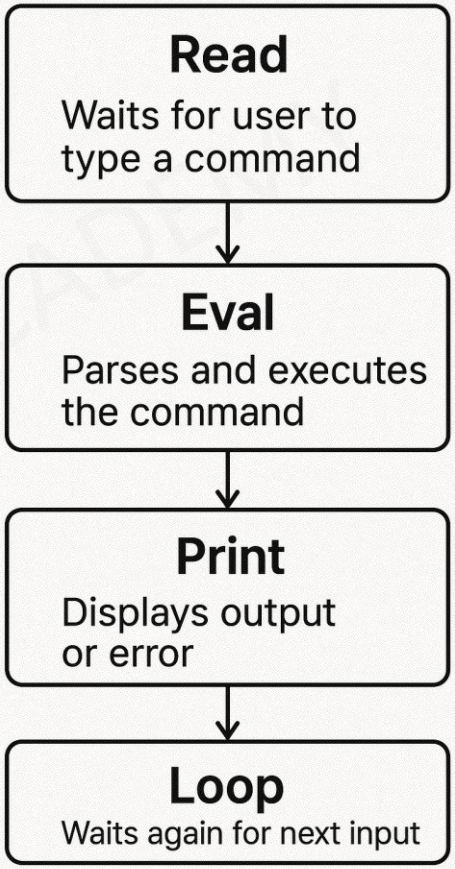
What is this file?

1. Maintained by System Admin (and Root)
2. Information in this file **affects all users** who are already logged-in and start another session by typing /bin/bash
3. When User logs in it is called **no Logon Shell**

REPL – INTERACTIVE LOOP

Bash now enters a loop called a **REPL** (Read-Eval-Print Loop):

Step	Action
Read	Waits for user to type a command
Eval	Parses and executes the command
Print	Displays output or error
Loop	Waits again for next input



Bash parses the command into:

- **ls** (program)
- -l and /etc (arguments)

Then it:

- Locates **ls** using the $PATH variable
- Creates a child process
- **Executes /bin/ls** with those arguments
- Waits for the command to finish
- Displays the output

Bash is just a shell (interface) — when you type a command like **uname -r**, it:

- Searches for it in PATH
- Runs it (binary or script)
- Passes arguments if any

SUMMARY

BASH STARTUP SCRIPTS: INITIALIZATION PROCESS

System Files (Global)

When Root logs in the SYSTEM FILES are executed. USER can not change these files:

1. /etc/profile
2. /etc/bashrc

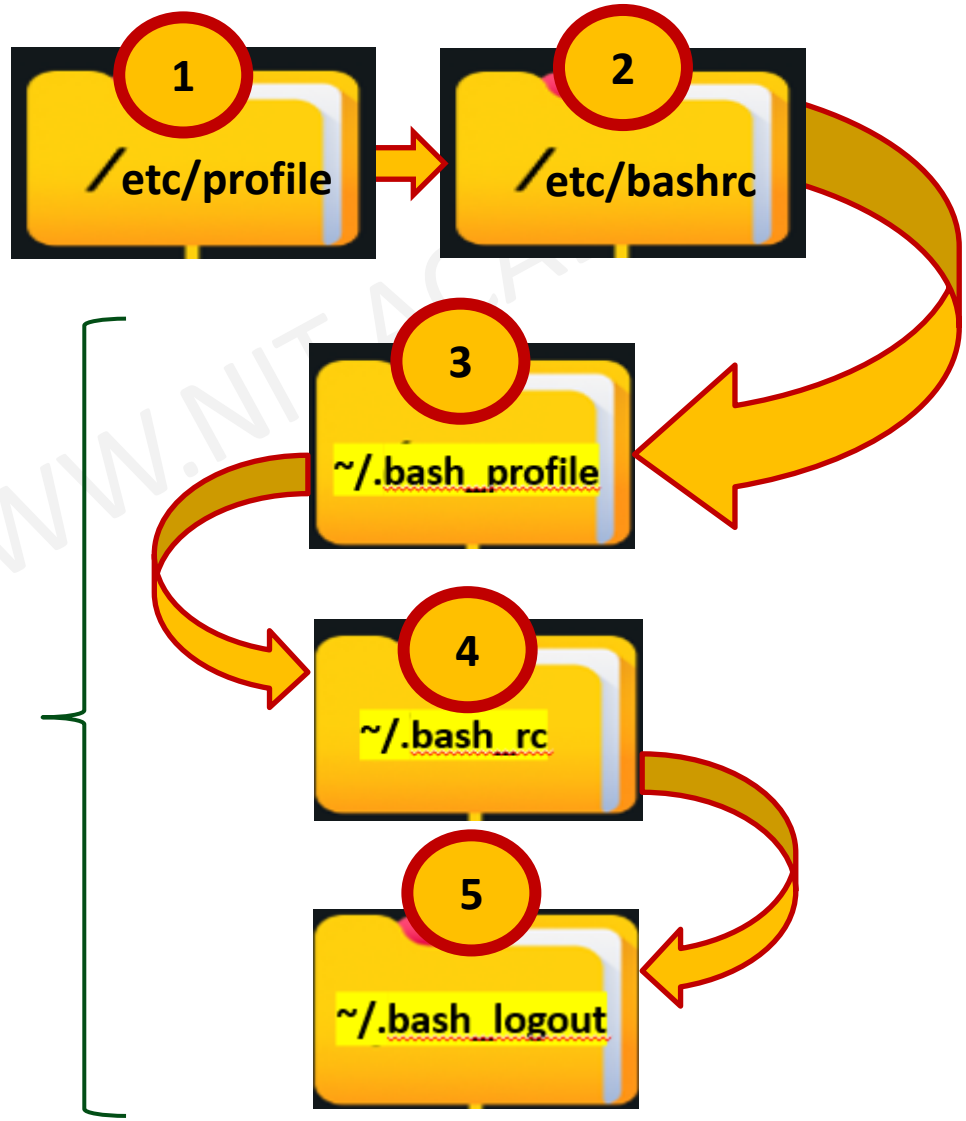
ROOT

User Files (Local)

User can change these files

1. USER initialization files
2. USER can make changes
3. These are environmental files that are found under #/etc/skell

USER



USER INITIALIZATION FILES

These are files that each USER can change / modify

Hey as a User I can edit this configuration file that I can modify and add extra stuff!

1

~/.bash_profile

Every time I the User open a new shell, this file is executed

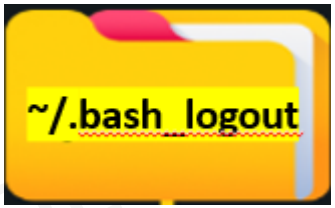
2

~/.bash_rc

.BASH_LOGOUT FILE…

These are files that each USER can change / modify

#vi /home/tom/.bash_logout




```
[tom@VM01 ~]$ cat .bash_logout
# ~/.bash_logout

# Clear my terminal Screen on logout
clear

#Remove all temporary files
rm -rf ~/tmp/*

# Unmount any NFS
#umount ~/network_share

#Kill Background processes
kill -9 $(jobs -p)
pkill -u $(whoami)

#Append a Timestamp
#date +"%Y-%m-%d %H:%M:%S Logout by $(whoami)" >> ~/.logout_history
```

LAB SESSION – HOME DIRECTORY

Login (Home Directory)

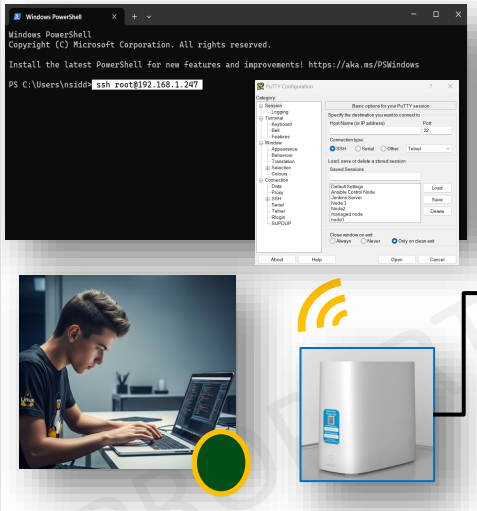
- A **home directory**, also called a **login directory**, is the directory on Linux-like operating systems that serves as the repository for a user's personal files, directories, and programs. It is also the directory that a user is first in after logging into the system.
- A **home directory** is created automatically for every ordinary user in the directory called **/home**. A **standard subdirectory of the root directory (/root)** and **/home** has the sole purpose of containing users' home directories.
- Apart from having a home directory to create and store files, **users need an environment** that gives them access to the tools and resources. When a user logs in to a system, the user's work environment is determined by the **initialization files.**
- These **initialization files are defined by the user's startup shell**, which can vary depending on the release. The default initialization files in your home directory enable you to customize your working environment.

SECURE SHELL (SSH)

Nexus Office

NOTE: All of these locations are REMOTE LOCATIONS

HOME

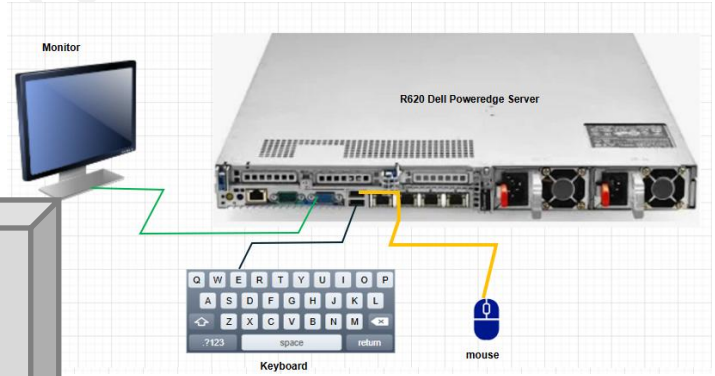


VPN, Tunnel, OpenVPN

- OpenVPN
- VPN
- Tunnel

SSH

DATA CENTER
"on-prem"



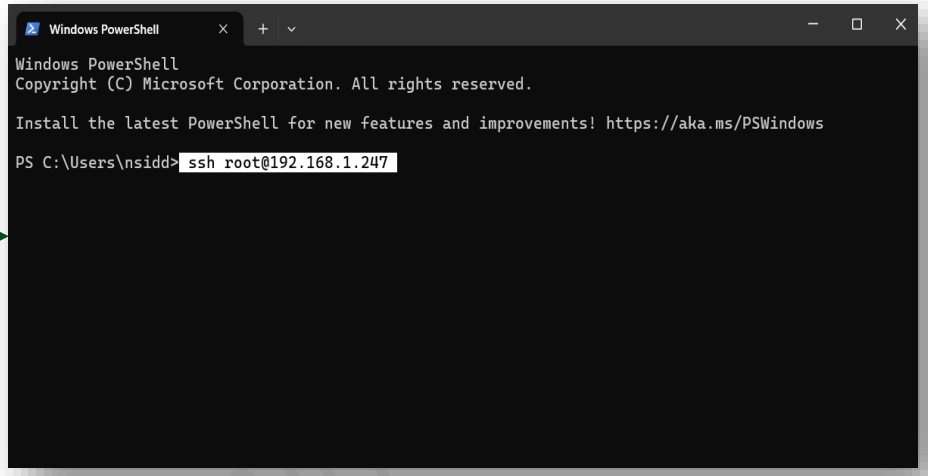
Client — Server

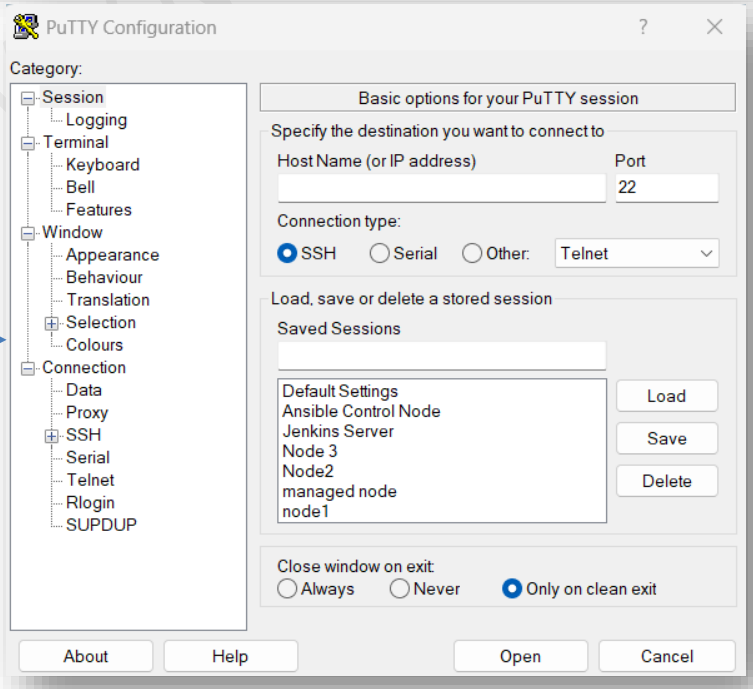
SSH – HOW MANY WAYS CAN YOU ACCESS A REMOTE SERVER

Date: July-27th-2025

- Terminal
- CMD
- Windows Powershell

PuTTy





Bash/Shell Scripting Essentials

Date: July-27th-2025

❑ Automating Repetitive Tasks

Shell scripting streamlines repetitive processes in Unix-based systems, saving time and reducing manual work for users.

❑ File and Process Management

Users can create scripts to manage files, automate processes, and monitor system status efficiently.

❑ Boosting Productivity

Learning shell scripting increases efficiency for developers, system administrators, and power users through automation and process optimization.

❑ What is Shell Scripting?

▶ A shell script is a file containing a series of commands.

The shell reads this file and carries out the commands as though they have been entered directly on the command line.

▶ It can be used to automate day-to-day various Unix/Linux tasks.

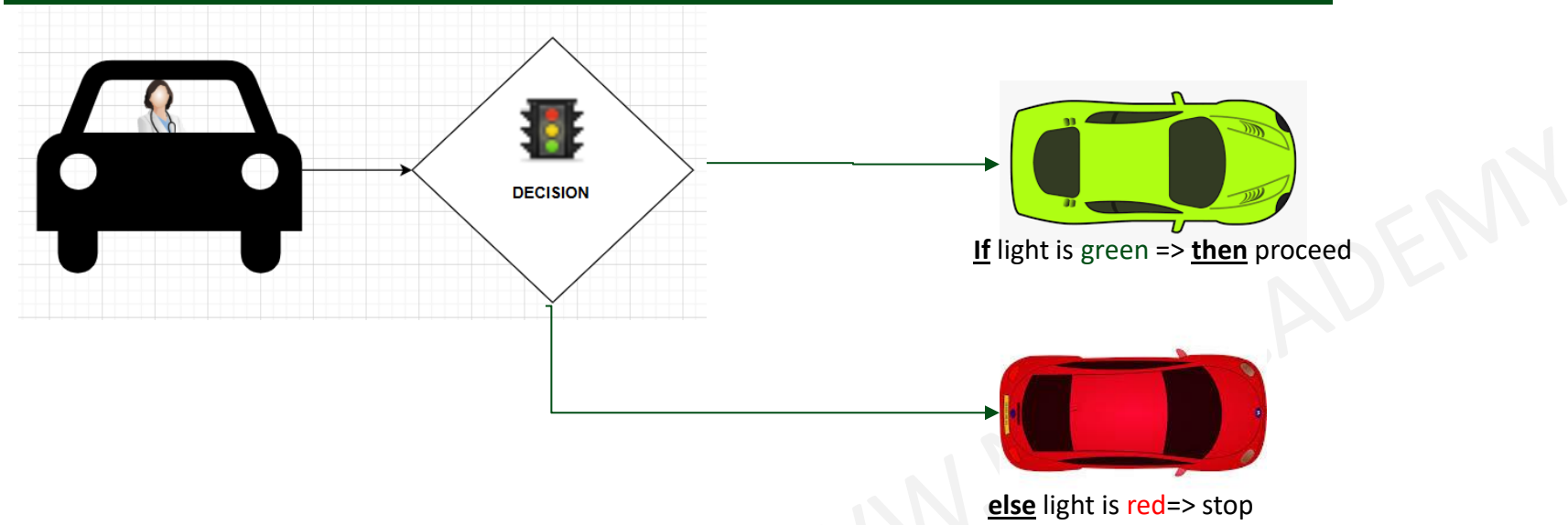
This can make Unix/Linux Administrators work easy, efficient and fast.

❑ Every shell script consists of

- Shell commands such as **pwd, echo, read, exit, type** etc.
- Shell keywords such as **if..else, for, while loop** etc.
- Linux binary commands such as **w, who, free** etc.
- Text processing utilities such as **sed, awk, cut, egrep** etc.
- Functions
- Control flow statements

if...else loop (logic)

Date: July-27th-2025



Explanation: This diagram illustrates an `if...else` statement. Imagine a car approaching a traffic light.

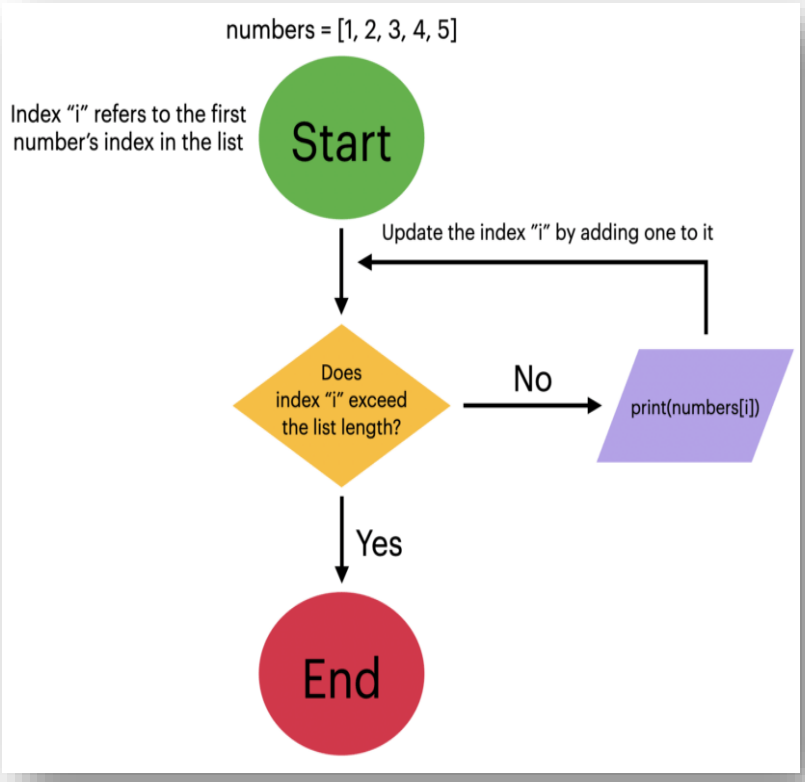
- **Condition:** The traffic light's color is the condition.
- **If (Green Light):** If the light is green (condition is true), the car proceeds straight through the intersection.
- **Else (Red Light):** If the light is red (condition is false), the car stops.
  This demonstrates how an `if...else` statement allows a program to choose between two different paths of execution based on whether a condition is true or false.

For loop (logic)

Date: July-27th-2025

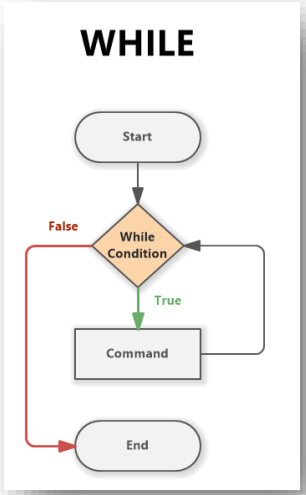
Explanation: This diagram represents a `for` loop. Think of a series of cars passing through a traffic light that stays green for a set number of cars.

- **Initialization:** The first car starts the process.
- **Condition Check:** Each car checks if there are more cars to go (the loop's condition).
- **Execution:** If the condition is true, the car passes through the light (the code inside the loop executes).
- **Increment/Decrement:** After passing, the next car in the series is ready (the loop's counter is updated).
  This loop continues for a predetermined number of iterations, much like a specific number of cars passing through.



while loop (logic)

Date: July-27th-2025

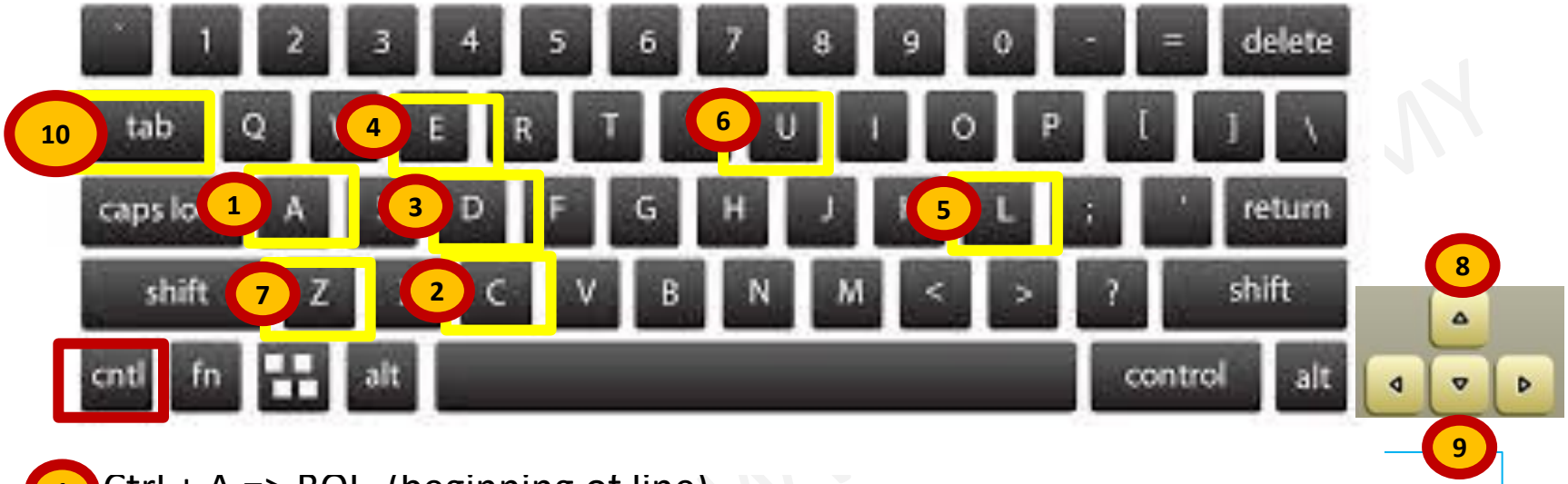


Explanation: This diagram shows a `while` loop. Consider a car driving around a traffic circle.

- **Condition:** The car continues to circle as long as a specific condition is true (e.g., a certain landmark hasn't been reached, or a traffic light at an exit remains red).
- **Execution:** The car keeps driving around the circle (the code inside the loop executes).
- **Loop Termination:** Once the condition becomes false (e.g., the landmark is reached, or the light turns green), the car exits the circle.
  A `while` loop continues to execute as long as its condition remains true, and it stops when the condition becomes false.

10 - KEYBOARD SHORTCUTS

Date: July-27th-2025



1. Ctrl + A => BOL (beginning of line)
2. Ctrl + C => Interrupt a process
3. Ctrl + D => exit
4. Ctrl + E => EOL (end of line)
5. Ctrl + L => This clears the screen
6. Ctrl + U => Clears line and takes you to BOL (acts like an eraser)
7. Ctrl + Z => STOPS a process (BG %1)

8.  **-------> scroll history UP**

9.  **-------> scroll history DOWN**

10. tab **----> saves time typing**

THANK YOU

Questions?

5

CHAPTER 5
Linux File System

Date: Aug-2nd-2025

Using the rules and strategies in this section you will gain confidence and a deep insight in setting up yourself for success

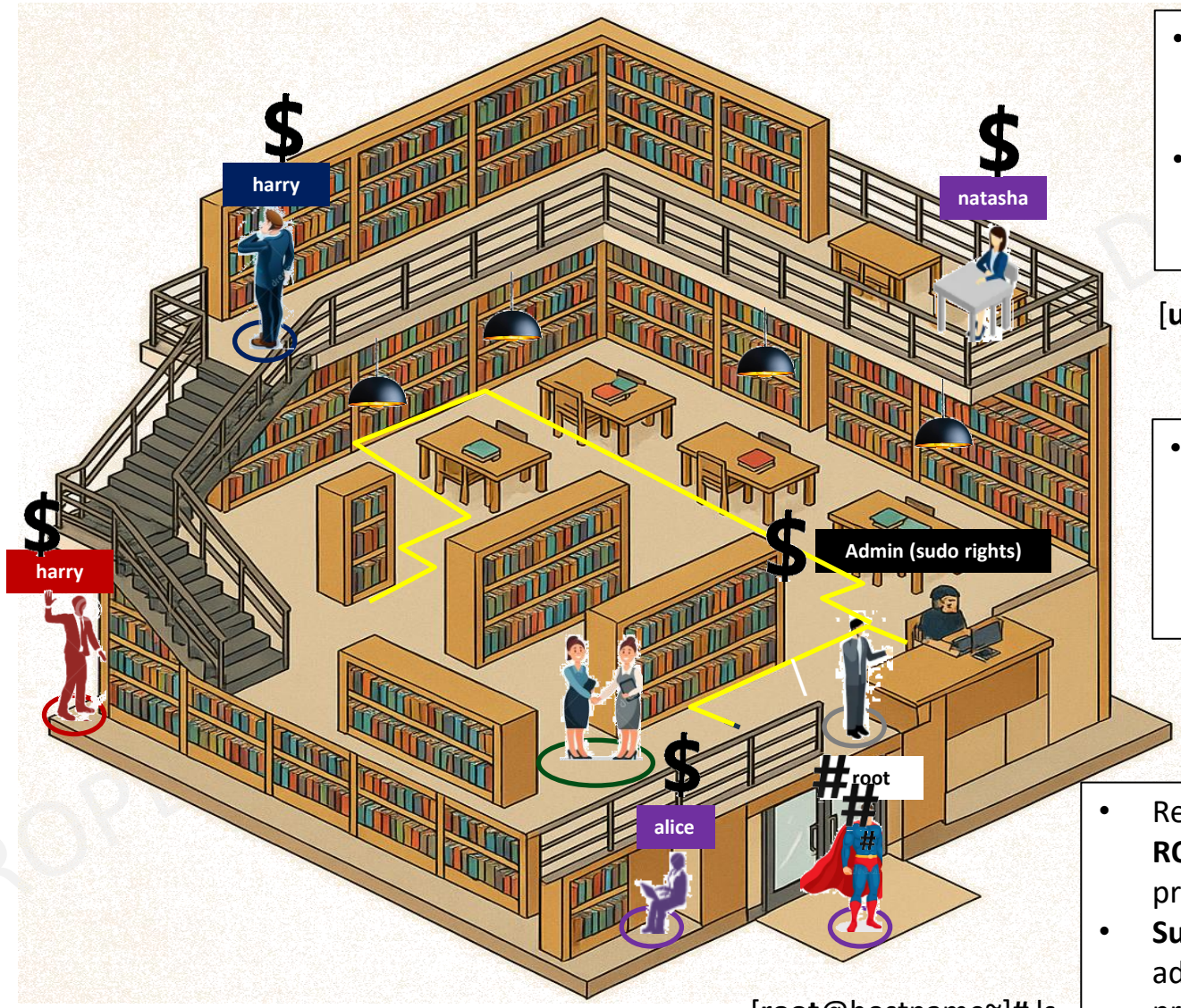
- Quick Overview of **Chapter 4**
- Linux Filesystem
  - Logical
  - Virtual
  - Physical
- Exploring Directories - /boot
  - Password Recovery – RCHSA Exam Question

- ACCESS THE COMMAND LINE
- MANAGE FILES FROM THE COMMAND LINE

INTRODUCTION TO LINUX FILESYSTEM

LINUX FILE SYSTEM

Date: Aug-2nd-2025



$

- Represents a **regular USER** shell prompt
- Non-root user / **normal user** account

[**user**@hostname~]**$** ls

- SUDO is a regular user with administrative privileges

#

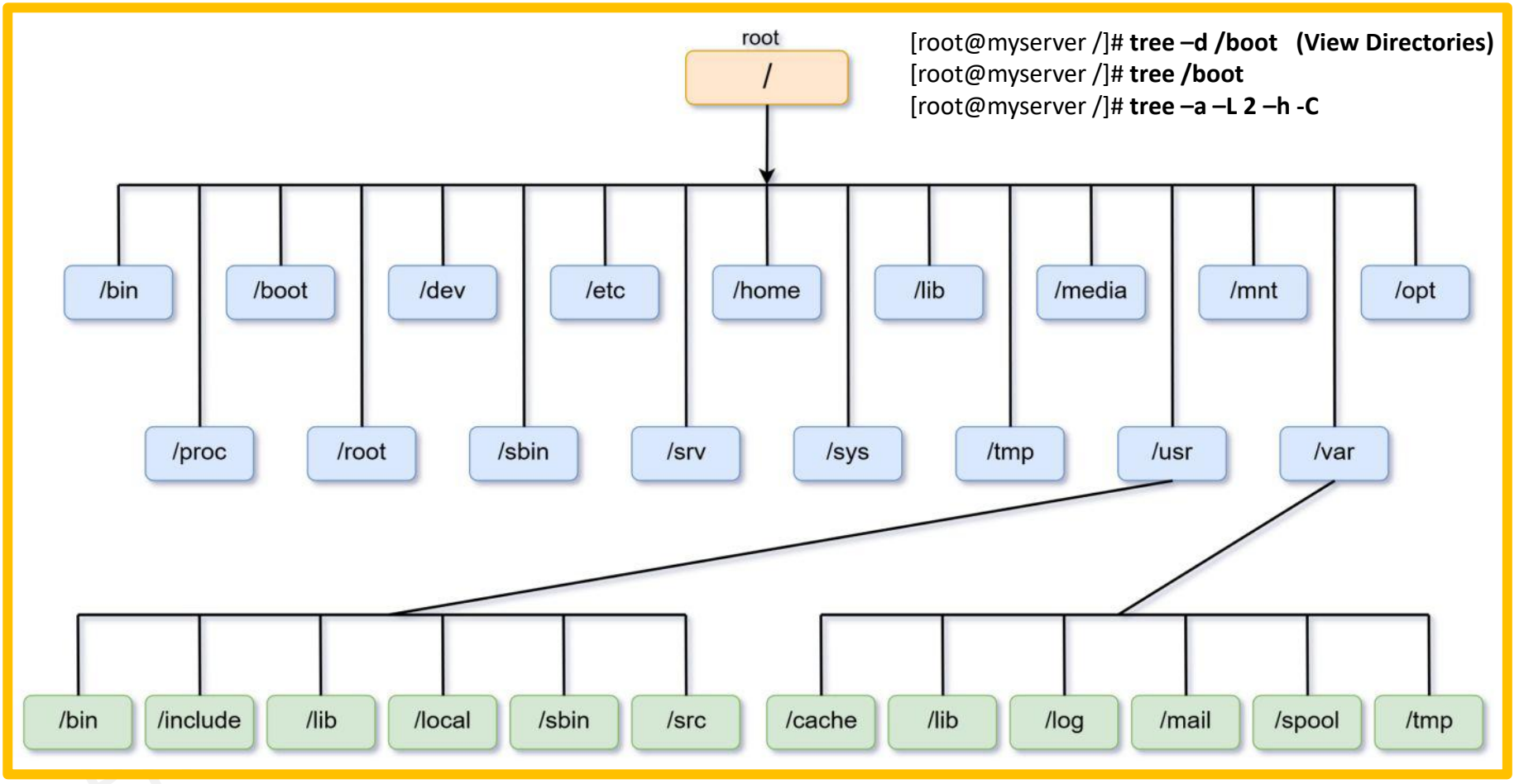
- Represents a **ROOT-USER** shell prompt
- **Superuser** with full administrative privileges

[**root**@hostname~]**#** ls

Manage Files

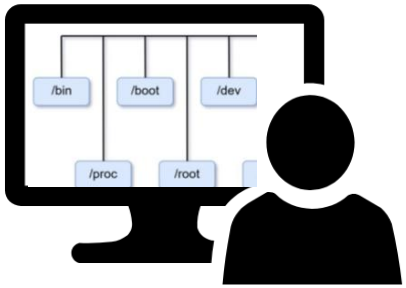
Date: Aug-2nd-2025

[root@myserver /]# **dnf install tree –y (let us install “tree” command)**



Linux File System Function...

Date: Aug-2nd-2025



(Logical File System)

[root@hostname~]# cat /proc/cpuinfo (Virtual File System)

Virtual Machine (VM)

Bash Shell

System Call

Kernel - OS

Logical File System

Virtual File System

Physical System

Linux File System

Partition 1

Partition 2

Partition 3

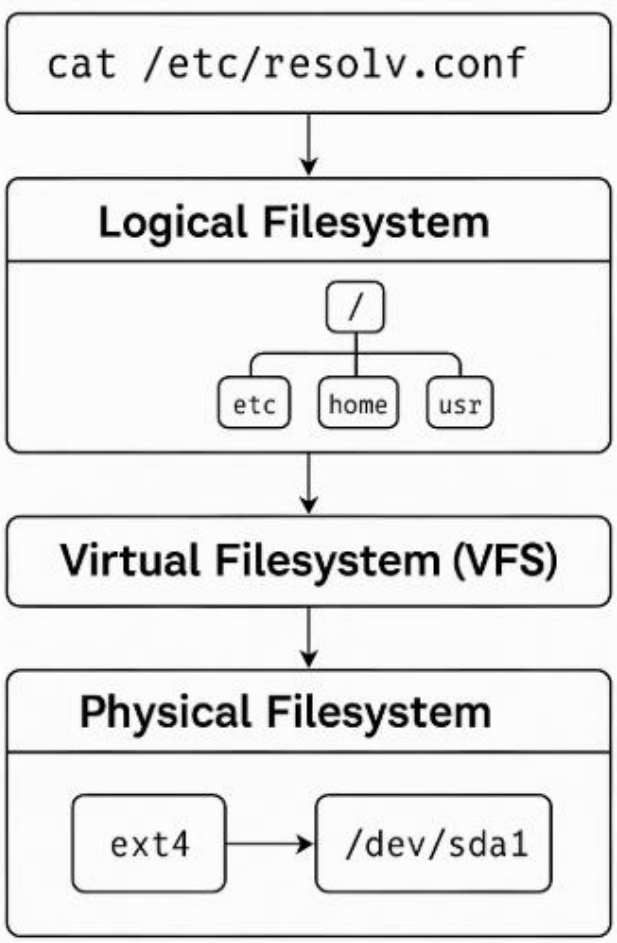
Hard Disk

HOW FILE SYSTEM WORKS?

Date: Aug-2nd-2025

The Linux file system is made up of **3-layers**:

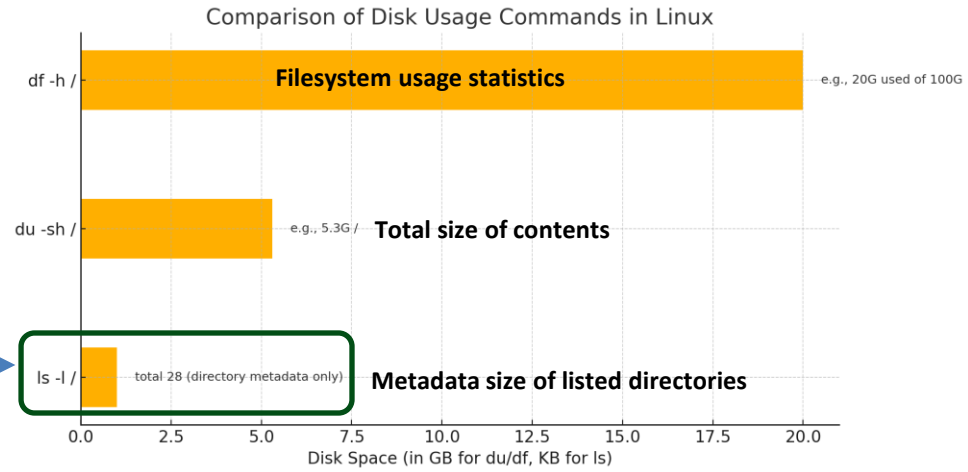
1. The **Logical File System** serves as the **interface** between user applications and the file system, managing operations like opening, reading, and closing files. Directory layout, LVM logical Volumes, Mounted Filesystems
2. **Virtual File System is an abstraction layer inside the Linux kernel.** Provides a unified interface to access all types of filesystems, for example, physical filesystems (ex4,xfs), Network filesystems (NFS) & Pseudo-filesystems (procfs, sysfs, tmpfs). Run cat/proc/cpuinfo which is DATA but looks like a normal file
3. **Physical File System is the actual data on the disk** organized into **blocks**, sectors, **inodes**, superblocks defined by formats like "**ext4**, xfs, btrfs, vfat" created using tools like "**mkfs**". It is what is physically **written to and read from** the storage device.

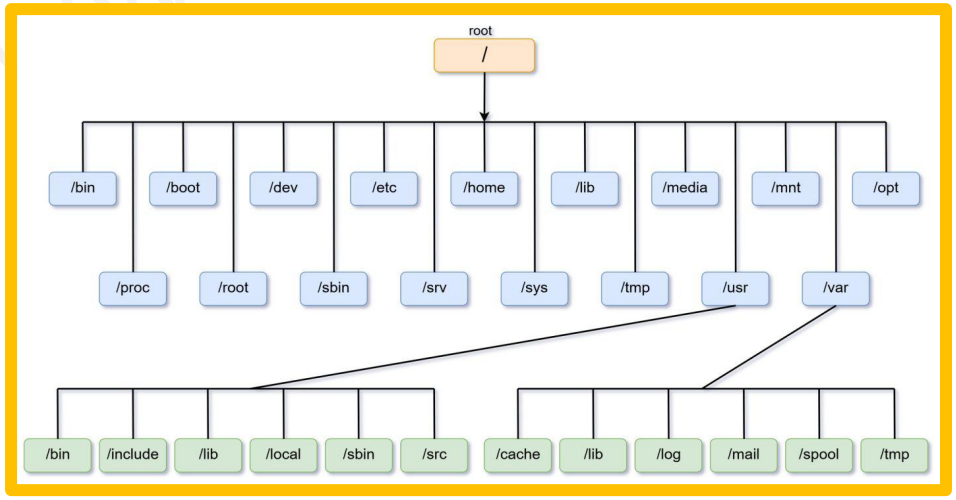


Linux File System (Logically)...

Date: Aug-2nd-2025

```
[root@myserver ~]# cd /
[root@myserver /]# ls -l
total 28 => 28KB of metadata (not actual data content)
dr-xr-xr-x.   2 root root    6 Nov  2  2024 afs
lrwxrwxrwx.   1 root root    7 Nov  2  2024 bin -> usr/bin
dr-xr-xr-x.   5 root root 4096 Aug  2 02:41 boot
drwxr-xr-x.  21 root root 3380 Aug  2 02:37 dev
drwxr-xr-x. 134 root root 8192 Aug  2 02:50 etc
drwxr-xr-x.   7 root root   77 Jul 27 12:53 home
lrwxrwxrwx.   1 root root    7 Nov  2  2024 lib -> usr/lib
lrwxrwxrwx.   1 root root    9 Nov  2  2024 lib64 -> usr/lib64
drwxr-xr-x.   2 root root    6 Nov  2  2024 media
drwxr-xr-x.   2 root root    0 Jul 28 01:23 misc
drwxr-xr-x.   2 root root    6 Nov  2  2024 mnt
drwxr-xr-x.   2 root root    0 Jul 28 01:23 net
drwxr-xr-x.   2 root root    0 Jul 28 01:23 netdir
drwxr-xr-x.   3 root root   22 Jul 19 20:09 netdir1
drwxr-xr-x.   4 root root   36 Jul 14 18:25 opt
dr-xr-xr-x. 183 root root    0 Jul 28 01:23 proc
dr-xr-x---.   4 root root 4096 Aug  2 02:51 root
drwxr-xr-x.  44 root root 1280 Aug  2 02:48 run
lrwxrwxrwx.   1 root root    8 Nov  2  2024 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 Nov  2  2024 srv
dr-xr-xr-x.  13 root root    0 Jul 28 01:23 sys
drwxrwxrwt.  11 root root 4096 Aug  2 09:13 tmp
drwxr-xr-x.  12 root root  144 Jul 13 00:28 usr
drwxr-xr-x.  21 root root 4096 Aug  2 02:32 var
```





LS COMMAND

Date: Aug-2nd-2025

LISTING
#ls
#ls –l /
#ls –ld /
#ls –al /
#ls –il /

```
[root@myserver /]# ls -ld /
dr-xr-xr-x. 22 root root 287 Jul 28 01:23 /
```

- 287bytes = This is the size of the **directory inode**, not what's inside it.
- This size may grow as more files/directories are added to /, but still, it is only the **directory's metadata**, not the sum of its contents.

```
[nitadmin2@nitadmin ~]$ #How can I see my home directories?
[nitadmin2@nitadmin ~]$ ls -al
total 20
drwx------. 2 nitadmin2 nitadmin2   99 Dec 21 15:50 .
drwxr-xr-x. 3 root      root        23 Dec 20 20:49 ..
-rw-------. 1 nitadmin2 nitadmin2 1086 Dec 25 01:23 .bash_history
-rw-r--r--. 1 nitadmin2 nitadmin2   18 Apr 30  2024 .bash_logout
-rw-r--r--. 1 nitadmin2 nitadmin2  141 Apr 30  2024 .bash_profile
-rw-r--r--. 1 nitadmin2 nitadmin2  492 Apr 30  2024 .bashrc
-rw-------. 1 nitadmin2 nitadmin2   20 Dec 21 15:50 .lesshst
```

ls -ld / shows the metadata of the '**/**' directory

This is different from **ls -l /**, which lists the metadata of the *contents inside* the /boot directory

Furthermore, **ls –il /** gives the inode numbers

What is Meta Data?

- **File type**
- **File size**
- **File permission**
- **File owner and group**
- **Timestamp**

INODES

Date: Aug-2nd-2025

When a filesystem is created a set amount of **Inodes are created.**
INODE is data structure that keeps track of all the information about a file.
You store your information in a file, and the operating system stores the information about a file in an inode (sometimes called an inode number).
Information about files(data) are sometimes called metadata. So, you can even say it in another way, **"An inode is metadata of the data."**

#ls -ld

Inode No 3470036	
. (DOT)	3470036
..(DOT DOT)	3470017
Folder 1	3470031
File 1	3470043
File 2	3470023
Folder 2	3470024
File 3	3470065

File and Directory Names

Inode corresponding to those names

#df - il

What is Meta Data?

- File type
- File size
- File permission
- File owner and group
- Timestamp

LINUX FILESYSTEM STRUCTURE

Date: Aug-2nd-2025

The **Linux filesystem** is structured as a hierarchical tree starting from the topmost directory called the **root (/) directory**. All other directories in Linux can be accessed from the root directory as they are “branches”.

Absolute Path

An **absolute path** is the complete path to a file or directory starting from the root (/) directory. It always begins with / and specifies the full location of the file or directory in the filesystem hierarchy, regardless of the current working directory.

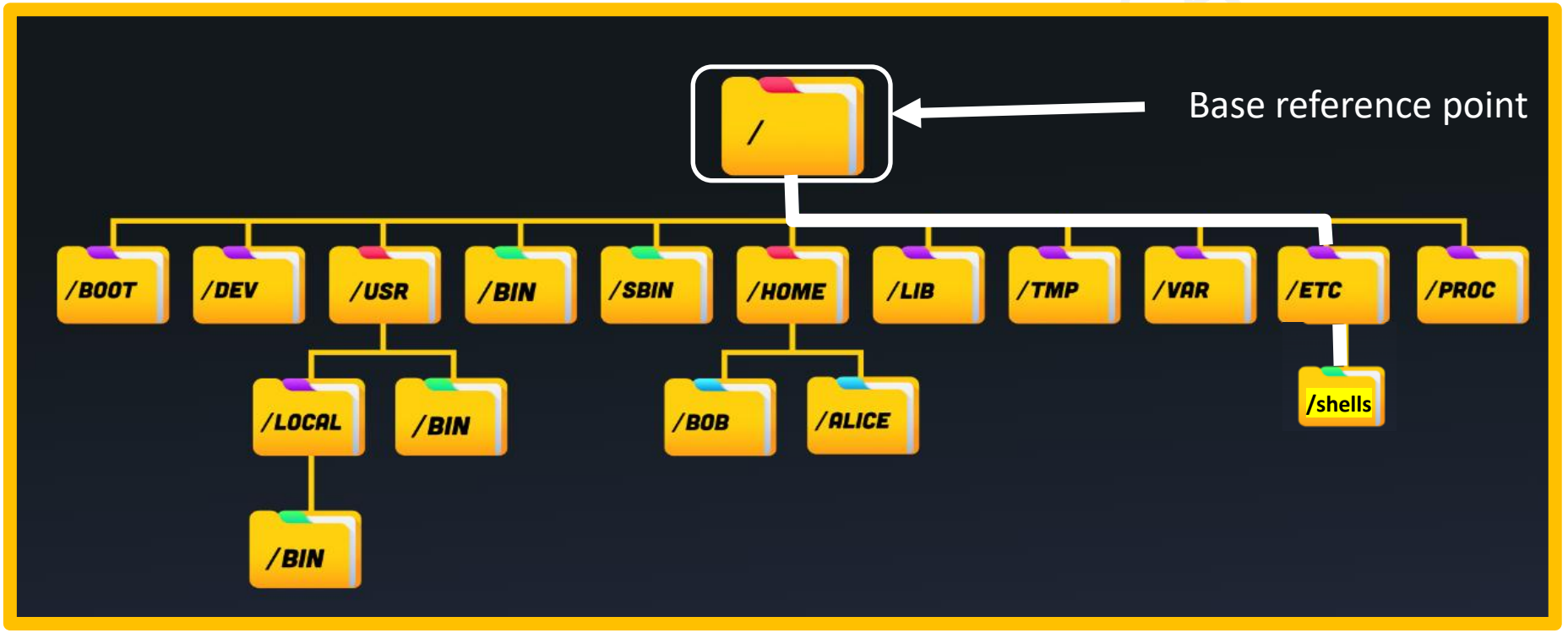
ABSOLUTE PATH

Date: Aug-2nd-2025

An **absolute path** is the complete path to a file or directory starting from the root (/) directory. It **always begins with "/"** and specifies the full location of the file or directory in the filesystem hierarchy, regardless of the current working directory.

Example: "/" – Leading when in beginning and "/" - trailing

[nitadmin2**]$cat /etc/shells**



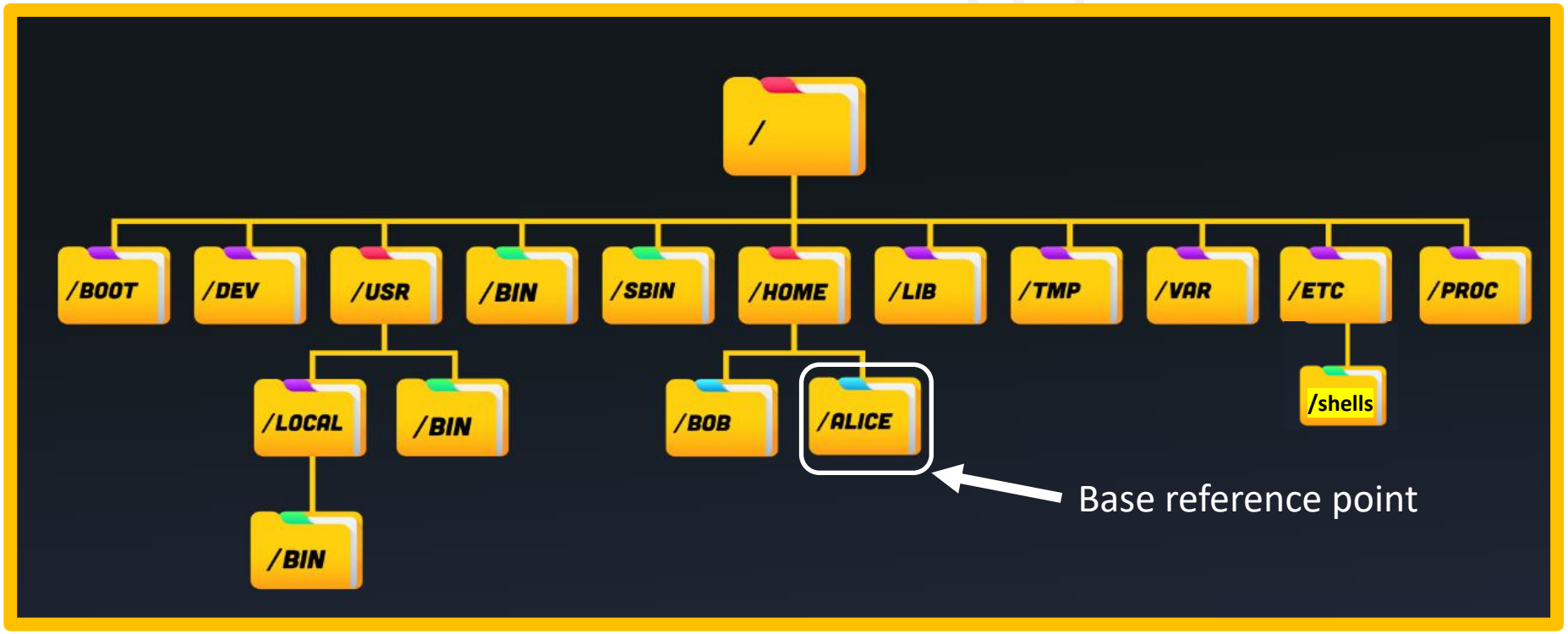
RELATIVE PATH

Date: Aug-2nd-2025

A Relative path specifies the location of a file or directory relative to the current working directory **(pwd)**. It **does not start with "/"** and instead uses the current directory as the **base reference point**.
Example: User is in "Home" directory and wants to navigate within "Home"

```
[root@myserver alice]# pwd
/home/alice
```

```
[root@myserver alice]# ll
total 4
drwxr-xr-x. 2 root root  6 Aug  2 12:07 dir1
-rw-r--r--. 1 root root 12 Aug  2 12:09 file1
[root@myserver alice]# cd dir1
[root@myserver dir1]#
```



LET US EXPLORE….

Date: Aug-2nd-2025

Permissions

User — read / execute

Group — read / execute

All — read / execute

We will learn later….

[root@myserver ~]# stat /boot

3

dr-xr-xr-x. 5 root root 4096 Jul 13 01:08 /boot

- dr-xr-xr-x. → File Type
- r-xr-x → Owner Permission
- r-x → Group Permissions
- r-x → Other Permissions
- 5 → Number of Hard Links
- root → User
- root → User's Primary Group
- 4096 → File Size in bytes
- Jul 13 01:08 → Modification Date
- /boot → File Name

1

Root

"/"

2

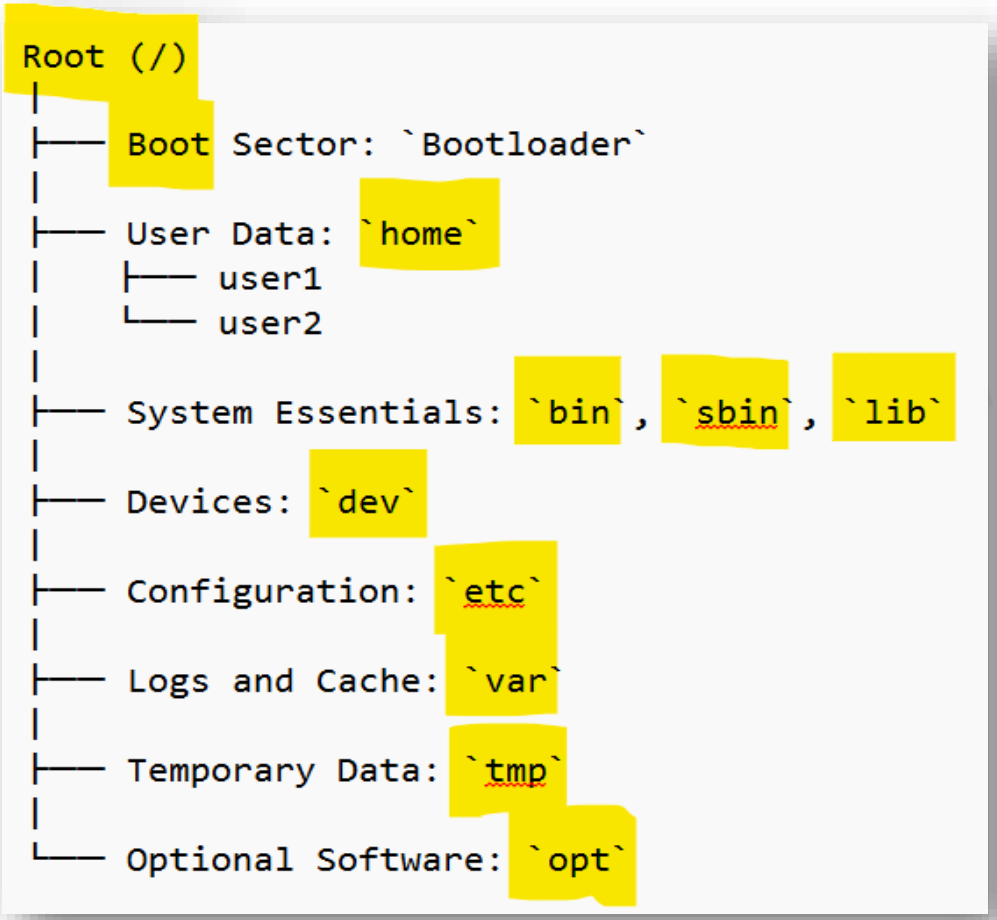
Step 1: Let us open the "/boot" directory:

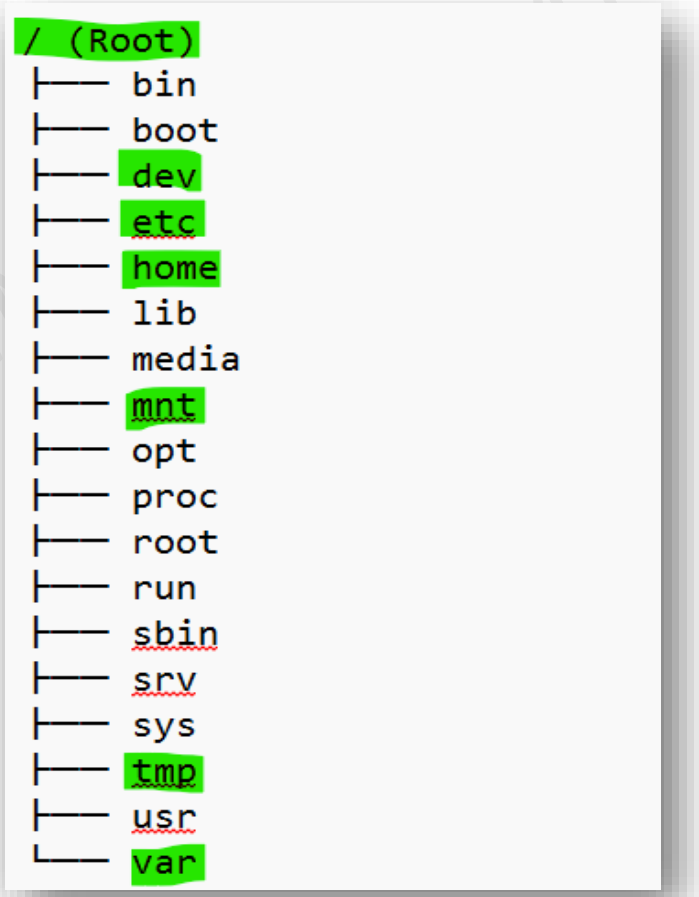
[root@myserver ~]# **ls -ld /boot**

DIRECTORY STRUCTURE

Date: Aug-2nd-2025

The **Linux filesystem** is structured as a hierarchical tree starting from the topmost directory called the **root (/) directory**. All other directories in Linux can be accessed from the root directory as they are “branches”.





Date: Aug-2nd-2025

Password Recovery – RHCSA EXAM QUESTION 1

- Crack Password of node2 Machine, and set password to **redhat**

/boot

1. Understand the booting process of Linux
2. How to set default target, used when booting and boot a system to a non-default-target
3. Login to a system and change root password
4. How to manually Repair the filesystem configuration or tackle corruption issue that stop the boot process.
5. Boot, reboot, and shut down a system **normally**

/BOOT

Date: Aug-2nd-2025

[root@myserver ~]# **ls -l /boot**

total 184536 <= File Size

```
[root@myserver kernel]# ls -l /boot
total 184536
-rw-r--r--. 1 root root   226284 Nov 15  2024 config-5.14.0-503.14.1.el9_5.x86_64
drwxr-xr-x. 3 root root       17 Jul 13 00:28 efi
drwx------. 5 root root       97 Jul 27 17:06 grub2
-rw-------. 1 root root 79130466 Jul 13 00:36 initramfs-0-rescue-4de79e0ace1b4500a934ade7bb133ed1.img
-rw-------. 1 root root 37206781 Jul 13 00:44 initramfs-5.14.0-503.14.1.el9_5.x86_64.img
-rw-------. 1 root root 34598400 Jul 13 01:08 initramfs-5.14.0-503.14.1.el9_5.x86_64kdump.img
drwxr-xr-x. 3 root root       21 Jul 13 00:30 loader
lrwxrwxrwx. 1 root root       52 Jul 13 00:31 symvers-5.14.0-503.14.1.el9_5.x86_64.gz -> /lib/modules/5.14.0-503.14.1.el
9_5.x86_64/symvers.gz
-rw-------. 1 root root  8876141 Nov 15  2024 System.map-5.14.0-503.14.1.el9_5.x86_64
-rwxr-xr-x. 1 root root 14457672 Jul 13 00:33 vmlinuz-0-rescue-4de79e0ace1b4500a934ade7bb133ed1
-rwxr-xr-x. 1 root root 14457672 Nov 15  2024 vmlinuz-5.14.0-503.14.1.el9_5.x86_64
[root@myserver kernel]# ls -ld /boot
dr-xr-xr-x. 5 root root 4096 Jul 13 01:08 /boot
```

Out of all these files, there are four that stand out:

1. The **Configuration** file
2. The **Initial RAM** Disk file
3. The **System Map** file
4. The **Linux Kernel** file

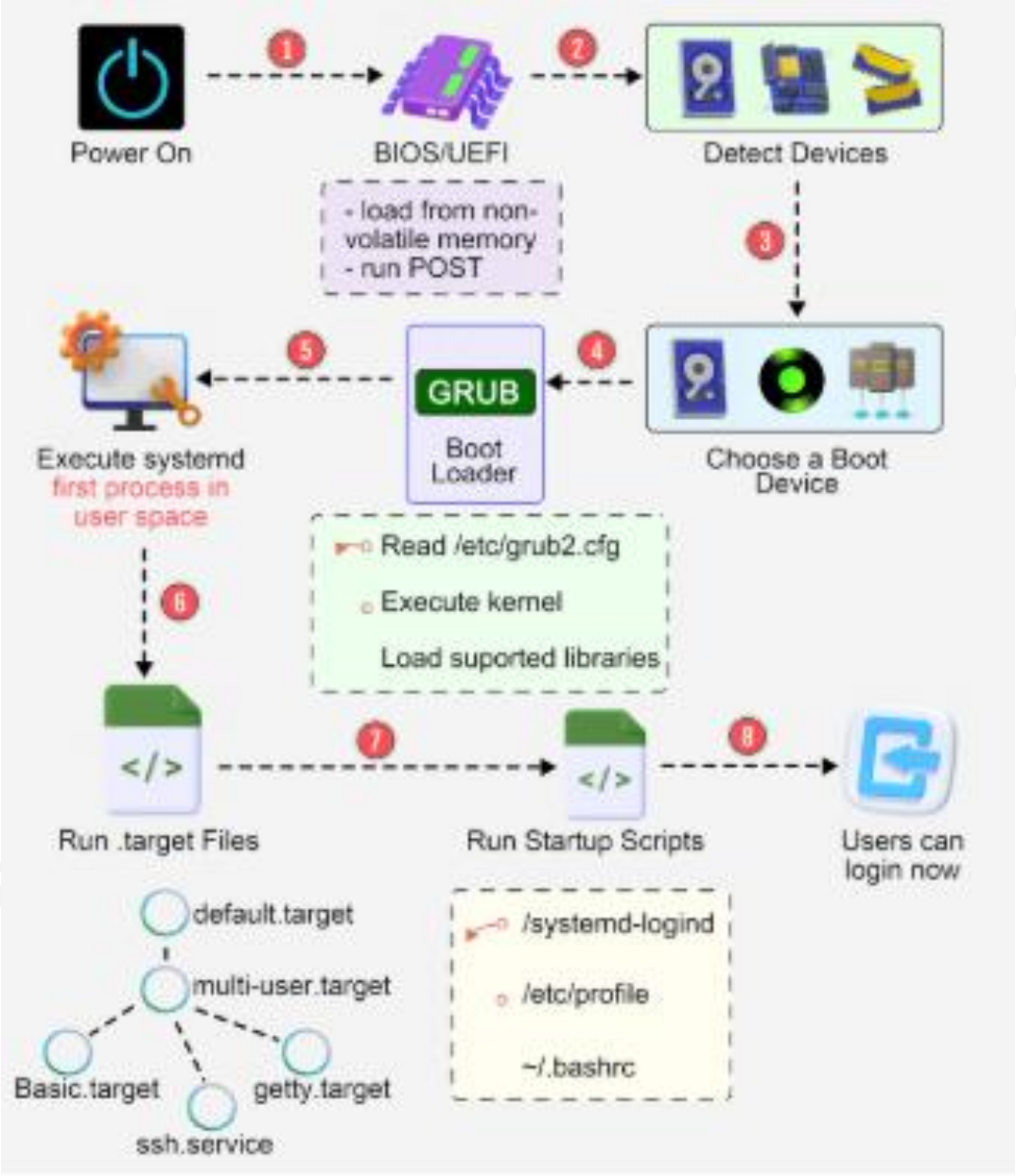
[root@myserver ~]# **tree -d /boot**

/boot

```
├── efi  (EFI partion exists, but does not imply UEFI boot
│   └── EFI
│       └── rocky
├── grub2
│   ├── fonts
│   ├── i386-pc
│   └── locale
└── loader
    └── entries
```

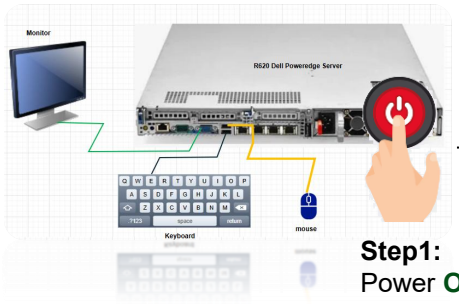
9 directories

Date: Aug-2nd-2025



CONTROLLING THE BOOT PROCESS

Date: Aug-2nd-2025



Step1: Power **ON**

BIOS

Step2: **Basic input/output**

HDD /dev/sda

Step3: **Search for bootable Device**

Fun activity: View MBR and Partition Table!

dd if=/dev/sda bs=512 count=1 | hexdump -C

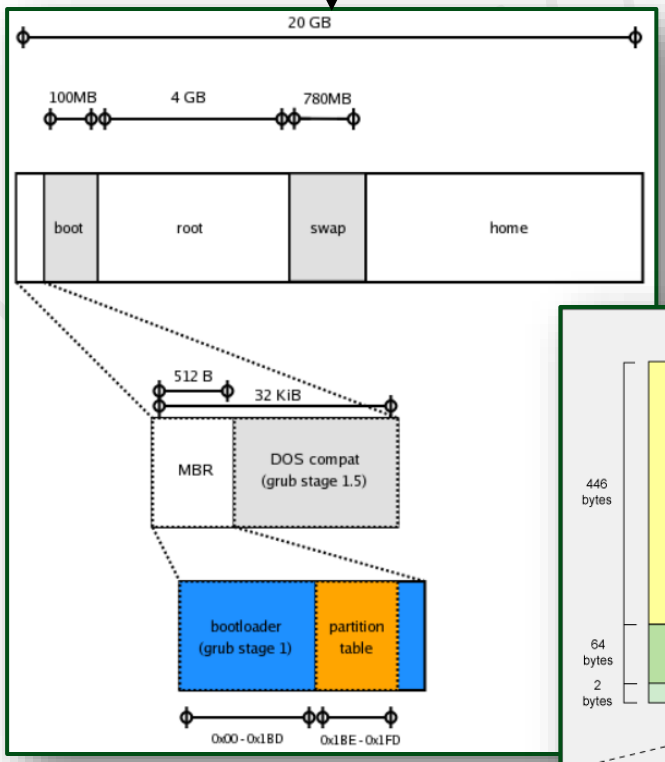
MBR

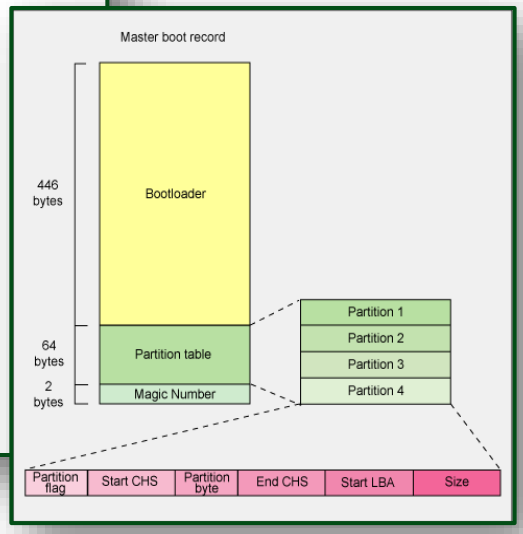
Grub2

[root@myserver boot]# fdisk -l

Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb25cfa22

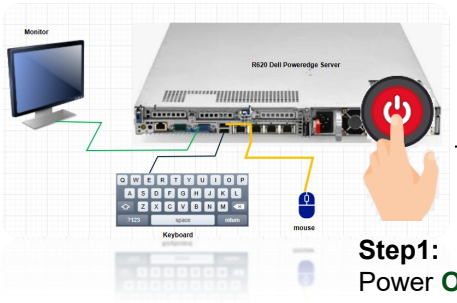
Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1	*	2048	2099199	2097152	1G	83	Linux
/dev/sda2		2099200	41943039	39843840	19G	8e	Linux LVM





CONTROLLING THE BOOT PROCESS

Date: Aug-2nd-2025



Step1: Power **ON**

BIOS

Step2: **Basic input/output**

HDD /dev/sda

Step3: **Search for bootable Device**

MBR

Grub2

Simply loads the the kernel into the memory

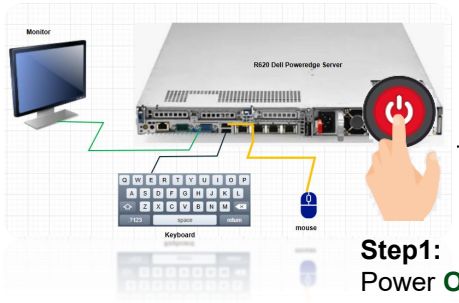
GRUB stands for Grand Unified Bootloader

- GRUB2 (Grand Unified Bootloader) Version 2.
- GRUB2 loads its configuration from the file /boot/grub2/grub.cfg file and displays a menu (splash screen) where you can select which kernel to boot.
- It loads the vmlinuz kernel Image (/boot/vmlinuz-4.18.0-80.el8.x86_64) and extract the content of Initramfs image (initramfs-4.18.0-80.el8.x86_64.img)

Random Access Memory (RAM)

CONTROLLING THE BOOT PROCESS

Date: Aug-2nd-2025



Step1: Power **ON**

BIOS

Step2: **Basic input/output**

HDD /dev/sda

Step3: **Search for bootable Device**

MBR

Grub2

Kernel, initramfs

Kernel Initializes all hardware

Executes /sbin/init

Systemd executes initrd.target

Mounting root fs on /sysroot directory

Root fs is switched

Systemd looks for default.target

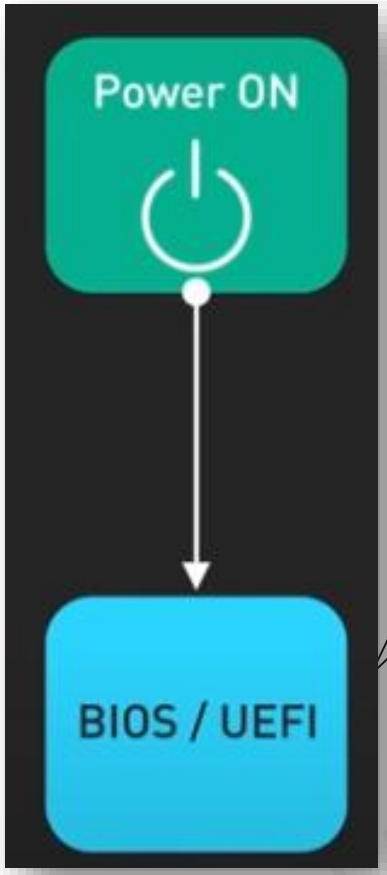
Systemd looks for default.target

/etc/system/system/default.target
/usr/lib/system/system/multi-user.target

BOOT PROCESS EXPLAINED

Date: Aug-2nd-2025

[-d /sys/firmware/efi] && echo "UEFI mode" || echo "Legacy BIOS mode"



BIOS	UEFI
- BIOS is tied to the Master Boot Record(MBR) system, which limits disk size to 2TB - Slower boot time - Less secure boot	- UEFI, on the other hand, uses the GUID Partition Table (GPT), removing these size constraints and offering a more flexible and modern solution - Faster boot time - Secure boot

[nitadmin1@nit firmware]$ **ls –l /sys/firmware/efi**

- If this directory exists, your system is using UEFI.
- If it does not exist, your system is using BIOS.

BIOS (Basic Input/Output System)

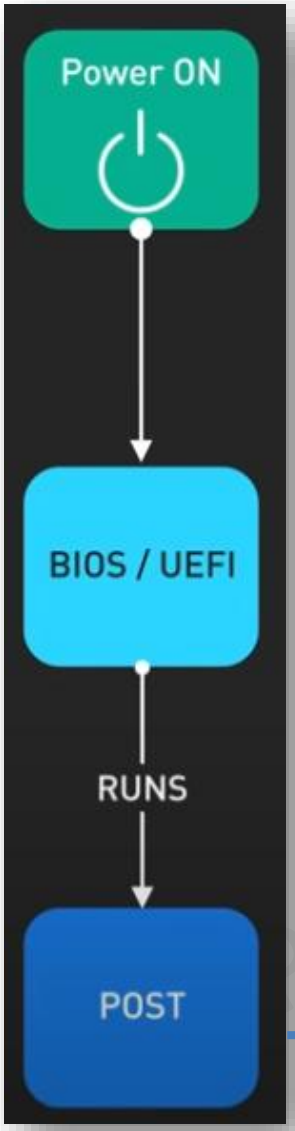
- **Older firmware standard** used in PCs before UEFI.
- Uses **MBR (Master Boot Record)** for booting, which has a 2TB disk size limit and supports up to **4 primary partitions**.
- **Runs in 16-bit mode**, meaning it has slower boot times and limited features.
- Uses a **text-based interface** for configuration.
- Relies on **Legacy Boot** mode, which loads the bootloader from a specific disk sector.

UEFI (Unified Extensible Firmware Interface)

- **Modern replacement for BIOS** with more features and better hardware compatibility.
- Uses **GPT (GUID Partition Table)**, which supports **disks larger than 2TB** and allows **unlimited partitions (typically up to 128)**.
- **Runs in 32-bit or 64-bit mode**, improving speed and efficiency.
- Provides a **graphical interface** with mouse support.
- Includes features like **Secure Boot**, which prevents unauthorized software from running at startup.

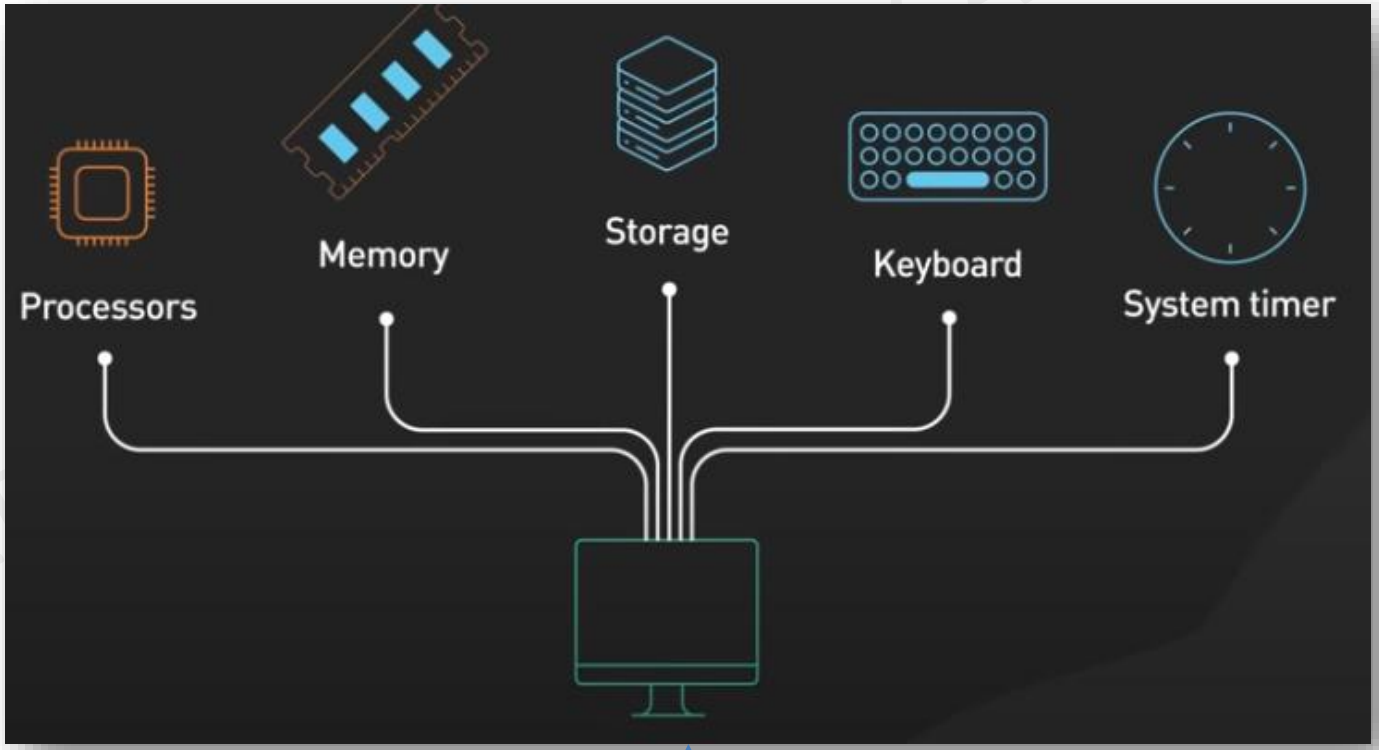
BOOT PROCESS - POST

Date: Aug-2nd-2025



Power **O**n **S**elf **T**est **(POST)**

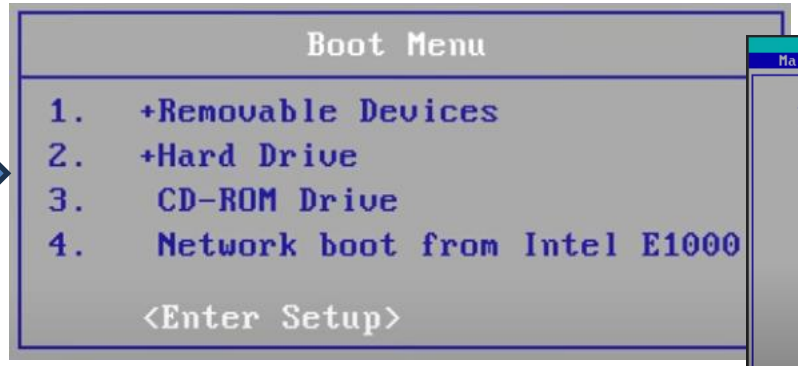
- **CPU Check** – Verifies if the processor is working.
- **Memory Test** – Checks if RAM is detected and functioning.
- **Storage Devices** – Ensures hard drives and SSDs are present.
- **Graphics Card Test** – Checks if the GPU is operational.
- **Input Devices** – Ensures the keyboard and mouse are detected.
- **Other Hardware Checks** – Includes network cards, sound cards, and other peripherals.

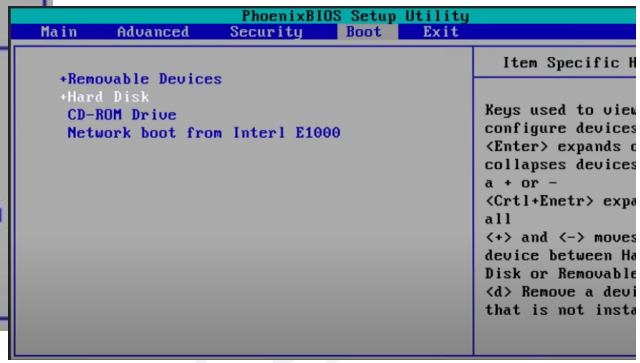


BOOT PROCESS – PARTITIONS

Date: Aug-2nd-2025

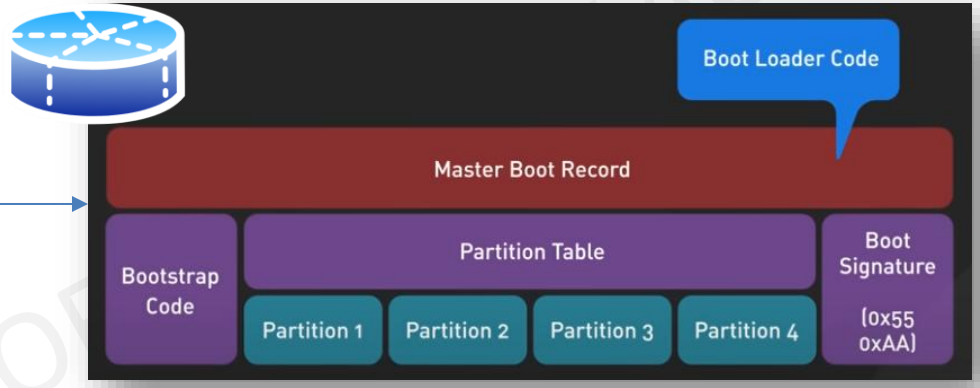
BIOS / UEFI



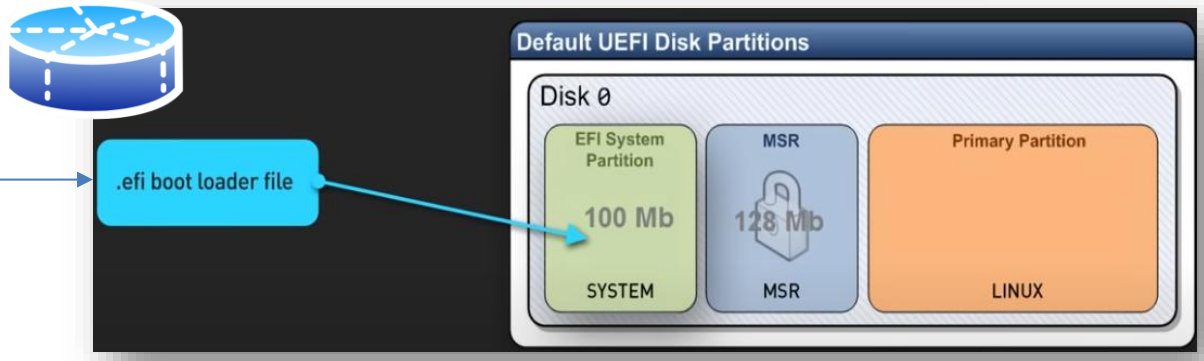


Partition

BIOS



UEFI



BOOT PROCESS – BOOT LOADER(S) -> KERNEL

BIOS / UEFI

MBR / efi

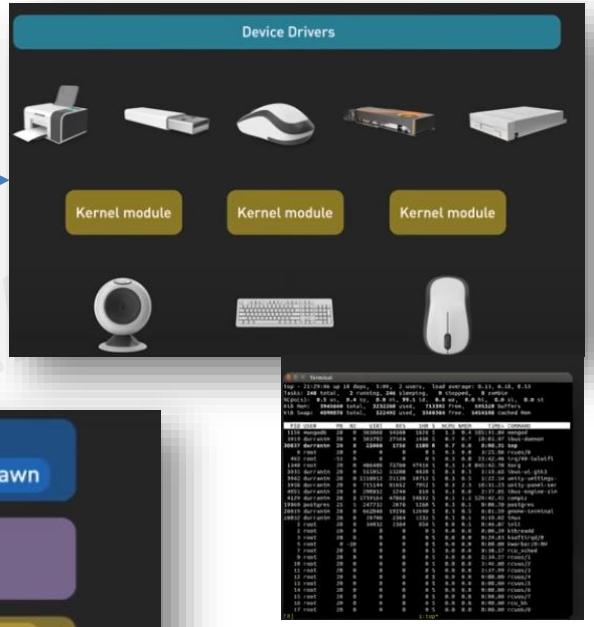
BOOTLOADER

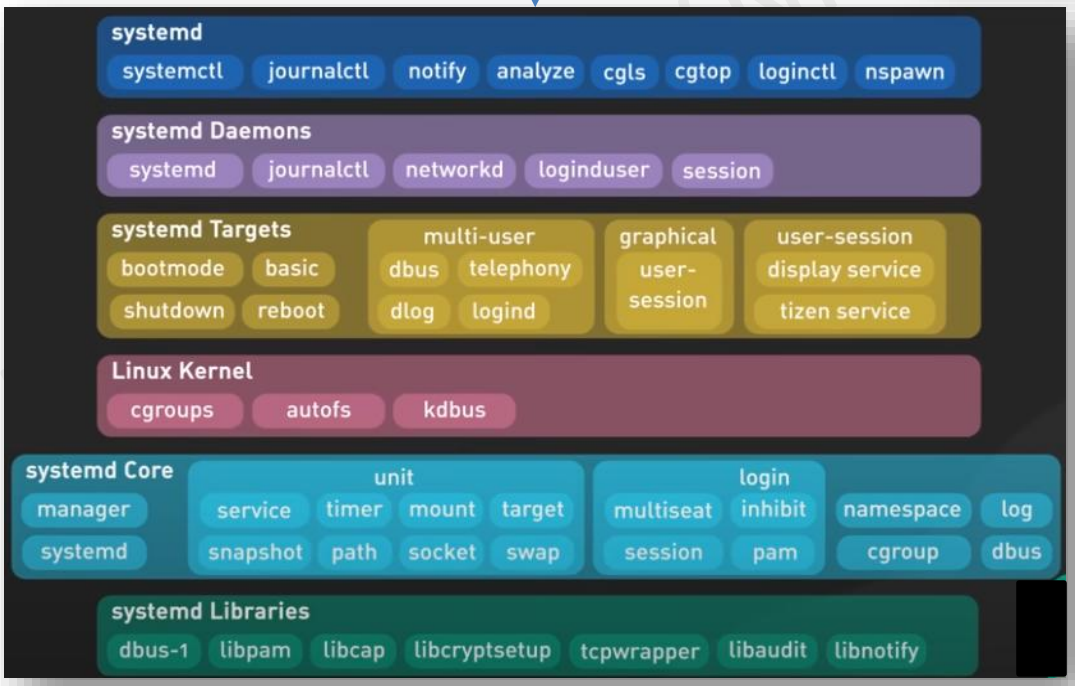
GRUB2 / ~~LILO~~

The key jobs for the boot loader are:

1. Locate the operating system kernel on the disk
2. Load the kernel into the computer's memory
3. Start running the kernel code

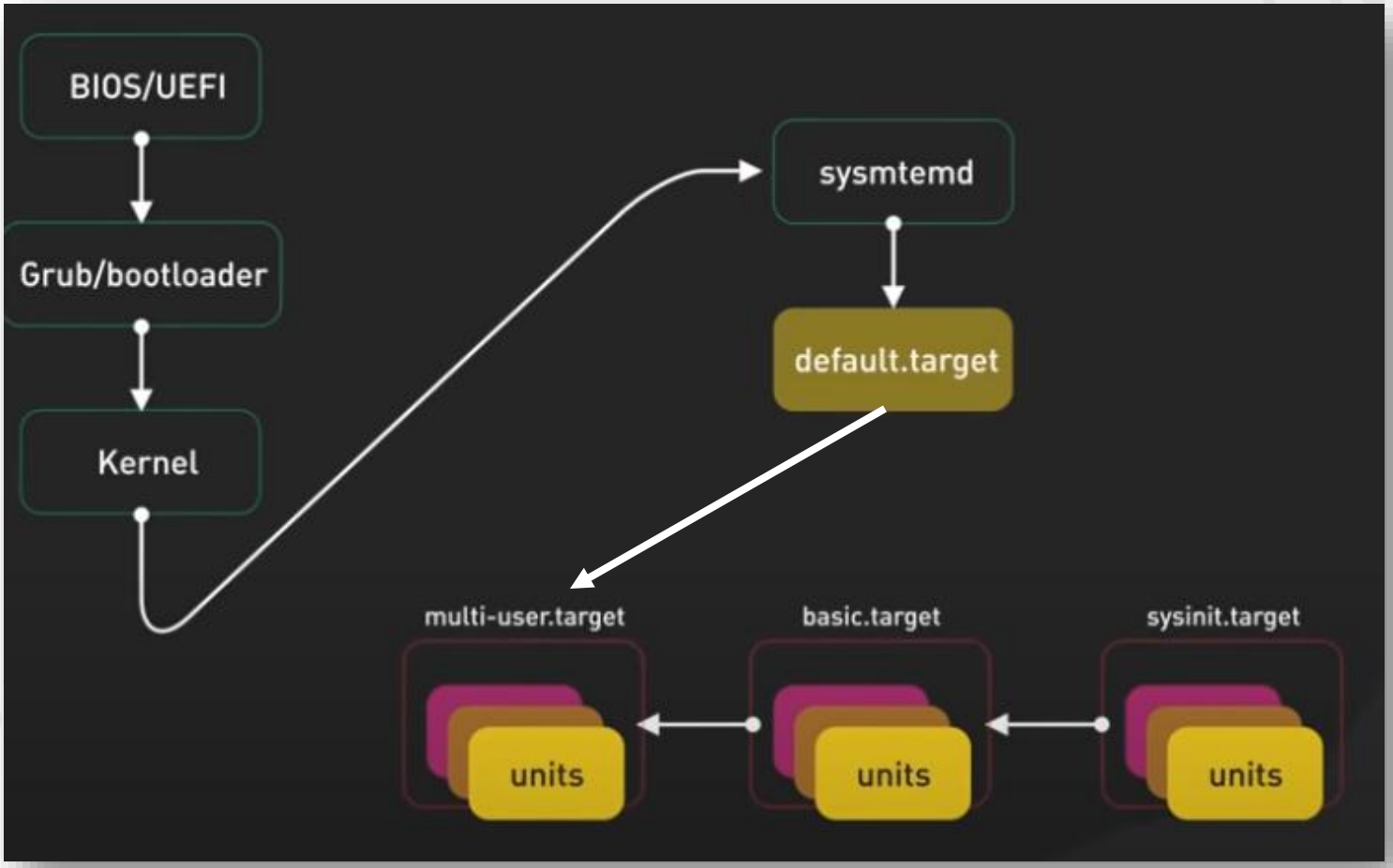
Kernel





BOOT PROCESS – TARGET

In **systemd**, a ***target*** is a group of services that define the state of the system. Targets replace traditional SysV runlevels and control what services and resources are available at any given time



TARGET – LINUX COMMANDS

How to check your systemd target
[nitadmin1@nit firmware]$ systemctl get-default
multi-user.target

To set default target permanently
To set a new default boot target:
#systemctl set-default multi-user.target

Switch to a different target in a running system
For example, to switch to **rescue mode**:
#systemctl isolate rescue.target

Run Level	Target Units	Description
0	runlevel0.target , poweroff.target	Shutdown and poweroff the system
1	runlevel1.target , rescue.target	Set up rescue shell
2	runlevel2.target, multi-user.target	Set up non-graphical multi-user system
3	runlevel3.target, multi-user.target	Set up non-graphical multi-user system
4	runlevel4.target, multi-user.target	Set up non-graphical multi-user system
5	runlevel5.target, graphical.target	Set up graphical multi-user system
6	runlevel6.target, reboot.target	Shut down and reboot the system

Here are some frequently used targets you can load with `systemd.unit=<target>` at boot:

Target	Description
`default.target`	The default target the system boots into (usually linked to `graphical.target` or `multi-user.target`).
`rescue.target`	Single-user mode with minimal services, useful for system recovery (like old **runlevel 1**).
`emergency.target`	Similar to `rescue.target`, but loads only the most basic system components.
`multi-user.target`	Command-line multi-user mode, similar to old **runlevel 3**.
`graphical.target`	Multi-user mode with a graphical user interface (GUI), like old **runlevel 5**.
`reboot.target`	Reboots the system immediately.
`poweroff.target`	Powers off the system.
`halt.target`	Stops the system without powering it off.
`suspend.target`	Suspends the system (sleep mode).
`hibernate.target`	Hibernates the system.
`hybrid-sleep.target`	Combination of suspend and hibernate.
`runlevelX.target`	Backward compatibility with SysV (`runlevel1.target` , `runlevel3.target` , etc.).
`network.target`	Ensures the network is available but does not configure it.

Run Level = init {0,1,2,3,4,5,6}

LINUX COMMANDS

Date: Aug-2nd-2025

Reboot, Shutdown and Suspending System

Command	Action/Description
systemctl reboot or shutdown -r now	To reboot System
systemctl halt or shutdown --halt	To halt the System
systemctl poweroff or shutdown --poweroff	To power off System
systemctl suspend	To suspend System (no-power mode)
systemctl hibernate	To hibernate System (low-power mode)
shutdown --halt HH:MM	To schedule System halt
shutdown --halt +10	To Schedule System halt after 10 minutes
shutdown --halt +10 “System will go down in 10 minutes”	To schedule System halt with message to users
shutdown -c	To cancel the scheduled shutdown command

Date: Aug-2nd-2025

ROOT PASSWORD RECOVERY

INTERVIEW QUESTION 1: HOW DO YOU RECOVER ROOT PASSWORD

Date: Aug-2nd-2025

Interrupt the boot process to set the root password as **"redhat"** and gain access to System. Procedure :

1. Start (Power on) the System.
2. Wait for GRUB menu to appear and then press e to edit.
3. Find the line starting with linux and type the **rd.break** at the end.
4. Press Ctrl+x to boot the system with these Kernel boot parameters.
5. Root file system is mounted on the disk as read only (this can be verified by mount command) mode => /sysroot [this is a chroot jail environment and must be remounted with r/w permissions].
6. To mount the root file system with r/w permissions
7. #mount -o remount,rw /sysroot
8. Activate chrooted root environment.
9. #chroot /sysroot
10. To set the root password
11. #passwd
12. To relabel the Selinux contexts (This step is important !)
13. #touch /.autorelabel
14. exit (To exit chrooted jail environment)
15. exit (To exit emergency maintenance mode)

INTERVIEW QUESTION 3: REPAIRING ROOT ISSUES AT BOOT TIME (RESCUE MODE)

Date: Aug-2nd-2025

1. **Reboot** the system
2. Wait for GRUB menu to appear and then press **e** to edit.
3. Find the line starting with linux and type the **rw init=/bin/bash** at the end
4. Press **Ctrl+x** to boot the system with **these** Kernel boot parameters
5. Once system is in the “bash shell” environment => fstab will be writable
6. **#vi /etc/fstab => fix the simulated problem**
7. **Exit**
8. **Turn VM off.**
9. **Start VM**

INTERVIEW QUESTION 2: REPAIRING A FILE SYSTEM ISSUE AT BOOT TIME (EMERGENCY MODE)

Interrupt the boot process to set the root password as **"password"** and gain access to System. Procedure :

1. Check your filesystems that you have => #df –h
2. Select a mounted fs and make some changes to FSTAB file => vi /etc/fstab
3. Reboot and **[FAILED]** Failed TO MOUNT /A
4. Type => #mount.
5. NOTICE => rhel-root MUST be read-write (rw) never (read-only, ro).
6. To mount the root file system with **r/w permissions**
7. **#mount -o remount,rw /sysroot**
8. Fix the issue in /etc/fstab
9. **#vi /etc/fstab => fix the simulated problem**
10. **#systemctl daemon-reload**
11. **exit** (To exit emergency maintenance mode)

CUSTOMIZING AND CONTROLLING THE BOOT PROCESS - GRUB

Follow the steps below to make changes to a boot process:

1. Launch your Red Hat Linux environment and open a terminal.
2. Configure Bootloader (GRUB):
   a) Identify the GRUB configuration file using the command:
   ls /etc/default/grub
   b) Edit the GRUB configuration file using a text editor:
   sudo vi /etc/default/grub (Note: sudo command is used by "Users" with root privileges.
   c) Modify the default boot options, such as the timeout or default entry, as desired.
   d) Save the changes and exit the text editor. e) Update the GRUB configuration using the command:
   sudo grub2-mkconfig -o /boot/grub2/grub.cfg
3. Modify Boot Options:
   a) Identify the current boot options set in the GRUB menu using the command:
   sudo cat /proc/cmdline
   b) Edit the boot options using a text editor:
   sudo vi /etc/default/grub
   c) Add or modify the desired boot options, such as kernel parameters or runlevels.
   d) Save the changes and exit the text editor. e) Update the GRUB configuration using the command:
   sudo grub2-mkconfig -o /boot/grub2/grub.cfg

4. Manage Services During Boot:
   - Identify the runlevel used for the default boot using the command:
     systemctl get-default
   - Disable a service from starting at boot using the command:
     sudo systemctl disable [service-name]
   - c) Enable a service to start at boot using the command:
     sudo systemctl enable [service-name]
5. Reboot the System:
   - Save all your work and close any open applications. b) Reboot the system using the command:
     sudo reboot
6. Observe and Verify Changes: a) During the boot process, observe any changes you made to the bootloader, boot options, or services. b) Check the status of specific services using the command:
   systemctl status [service-name]
7. Conclusion

In this exercise you have learned to control the boot process in Red Hat Linux. You have configured the **bootloader (GRUB),** modified boot options, and managed services during the boot process. By completing these steps, you have gained practical experience in controlling various aspects of the boot process in Red Hat Linux.

SUMMARY OF CONTROLLING BOOT PROCESS

1. **Install Bootloader**: Install and configure the bootloader (e.g., GRUB, LILO) on the primary disk.
2. **Modify Boot Options**: Edit the bootloader configuration file (e.g., /etc/default/grub) to change boot parameters.
3. **Update Bootloader**: Run commands like **update-grub or grub2-mkconfig** to apply the changes.
4. **Set Default OS**: Set the default boot entry by editing the bootloader configuration file.
5. **Manage Boot Services**: Configure system services to start or stop during the boot by modifying /etc/systemd/system or using systemctl.
6. **Enable/Disable Services**: Use commands like systemctl enable or systemctl disable to control service behavior during boot.
7. **Reboot System**: After making changes, reboot the system to verify the bootloader and service configurations.
8. **Troubleshoot Boot Issues**: Use tools like dmesg, journalctl, or boot logs for diagnosing boot problems.
9. **Test Recovery Mode**: Ensure that recovery options are available for troubleshooting and system recovery.
10. **Finalize Configuration**: Ensure all changes are persistent across reboots by saving configuration files.

Date: Aug-3rd-2025

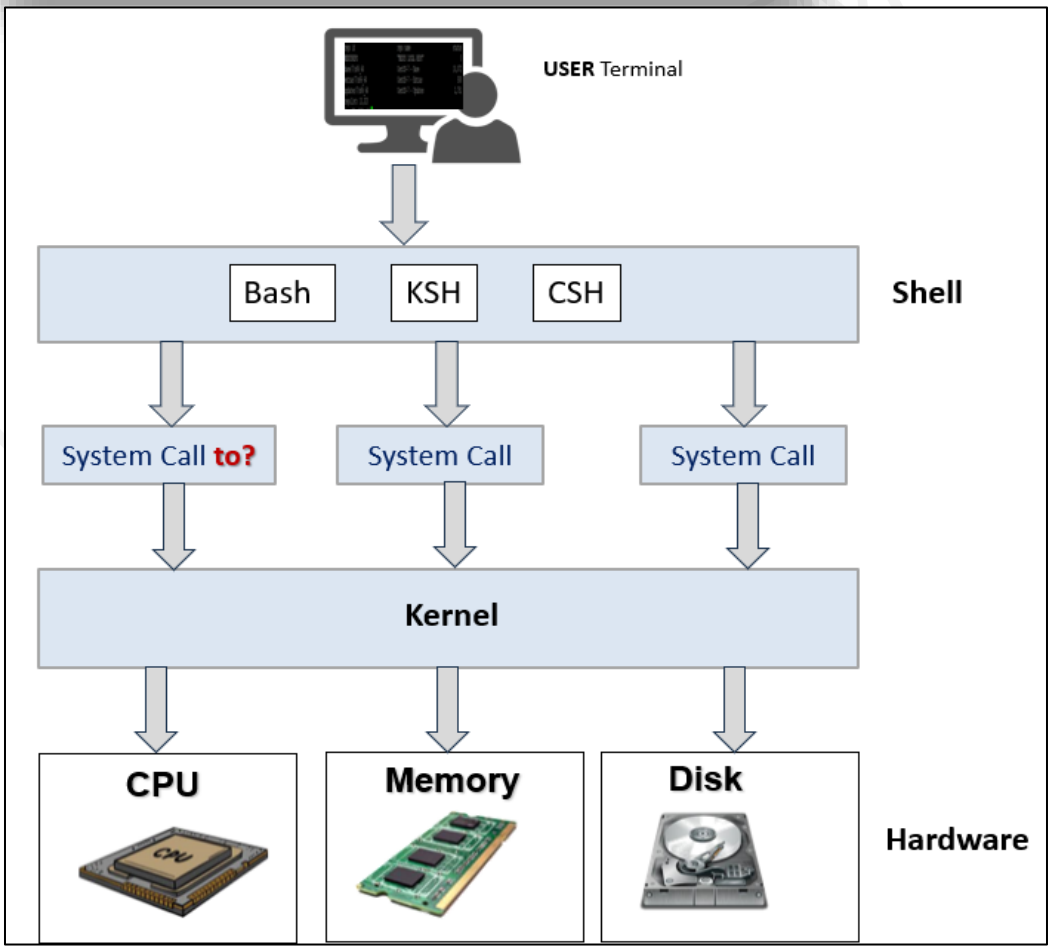
USERS - RHCSA EXAM QUESTION

/home

Date: Aug-3rd-2025

1. **Where are you when you login?**
   - Home directory
   - Login directory
   - Contains User personal files, directories and programs.
   - This is where a person lands after logging into the system
2. **Home Directory** is created automatically when a new user is added **with /home**
3. **Users need an environment** that gives them access to the tools and resources. When a user logs in to a system, the user's work environment is determined by the initialization files

```
Root (/)
|
├── User Directory: `home`
|    ├── user1
|    └── user2
```



Date: Aug-3rd-2025

USERS - RHCSA EXAM QUESTION

/dev

```
├── Devices: `dev`
```

Date: Aug-3rd-2025

- **Everything is a file in Linux. Even the devices are viewed as files**
- /dev stands for **"device"**. Reading from /dev/sda is like accessing the hard disk
- **Contains special files that represent hardware devices**
- **Managed dynamically by udev (device manager)**

1 **#ls /dev**

2 **#ls -l /dev/sda1**

```
[root@VM01 etc]# ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Jan 15 09:54 /dev/sda1
[root@VM01 etc]# df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                  4.0M     0  4.0M   0% /dev
tmpfs                     385M     0  385M   0% /dev/shm
tmpfs                     154M  4.3M  150M   3% /run
/dev/mapper/rl_vbox-root   17G  1.6G   16G  10% /
/dev/sda1                 960M  374M  587M  39% /boot
tmpfs                      77M     0   77M   0% /run/user/1000
```

Types of Device Files

Date: Aug-3rd-2025

Type	Description	Examples
Block Devices	Accessed in blocks (random I/O)	/dev/sda, /dev/sdb1
Character Devices	Accessed as a stream (sequential I/O)	**/dev/tty,** /dev/random

1. When accessing Data in Blocks (Block Devices)
Real-Life Devices:
- Filesystems (like ext4, xfs) use block devices
- Stored under **/dev** as **/dev/sda**, **/dev/xvda**, etc.

Examples:
- Hard drives **(/dev/sda)**
- SSDs
- USB drives

Behavior:
- Data is read or written in **fixed-size chunks**, called **blocks** (typically 512 bytes, 4KB, etc.)
- You **can jump to any block** directly without reading all data before it.

Analogy:
Like reading **pages from a book** — you can go to **page 100** directly without reading pages 1–99.

2. When accessing Data as a stream (Character Devices)
Real-Life Devices:
- Serial consoles, pipes, sensors
- Found as /dev/tty*, /dev/zero, /dev/random

Examples:
- Keyboard (/dev/tty)
- Mouse (/dev/input/mouse0)
- Serial ports (/dev/ttyS0)

Behavior:
- Data is accessed as a **continuous flow** (byte-by-byte or character-by-character).
- You **cannot seek** or skip ahead — you read data **in the order it comes in**.

Analogy:
Like **listening to a radio** — you hear sounds as they come; you can't skip ahead.

Types of TTYs in Your Screenshot:

Date: Aug-3rd-2025

TTY	Explanation
tty1	A **virtual console** directly on the machine
pts/0	A **pseudo-terminal** slave used for **remote or terminal emulator sessions**, like SSH

```
[root@myserver ~]# w
 01:03:26 up 15 min,  2 users,  load average: 0.03, 0.07, 0.09
USER     TTY      LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     00:48   14:30   0.01s  0.01s -bash
root     pts/0    01:01    0.00s  0.05s  0.02s w
```

Command	What it does	Terminal 1	Terminal 2
tty	Shows which terminal (e.g., /dev/pts/0)	**[root@myserver ~]# tty** /dev/pts/0	**[root@myserver ~]# tty** /dev/pts/1
w / who	Shows all logged-in users with TTY	**[root@myserver ~]# w** 01:46:22 up 57 min, 4 users, load average: 0.07, 0.06, 0.05 USER TTY LOGIN@ IDLE JCPU PCPU WHAT root tty1 00:48 38:06 0.26s 0.26s -bash root pts/0 01:01 5:49 0.05s 0.05s -bash root pts/1 01:12 0.00s 0.30s 0.13s w	
ls /dev/pts	Lists all pseudo terminals	[root@myserver ~]# ls /dev/pts 0 1 2 ptmx	
echo "msg" > /dev/pts/X	Sends a message to another terminal	echo "Hello from pts/0" > /dev/pts/1	

[root@myserver ~]# lsblk

Date: Aug-3rd-2025

```
[root@myserver ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   20G  0 disk
├─sda1        8:1    0    1G  0 part /boot
└─sda2        8:2    0   19G  0 part
  ├─rl-root 253:0    0   17G  0 lvm  /
  └─rl-swap 253:1    0    2G  0 lvm  [SWAP]
sr0          11:0    1 1024M  0 rom
```

Disk Commands

- **#df -hT**
- **#lsblk**
- **#fdisk -l**

Device	Description
/dev/sda	First hard disk (HD)
├─ /dev/sda1	First **partition** on first disk => 1GB mounted as /boot
├─ /dev/sda2	Second partition on first disk =>19 GB **not directly mounted**, but subdivided
│ ├─ rl-root	Logical volume (holds **/**) in a volume group called “rl”
│ ├─ rl-swap	Used as “swap” space
/dev/null	Discards all input
/dev/tty	Terminal (Console) device
/dev/pts/*	Pseudo terminals

Fun with Sensors in Data Center Environment!!!

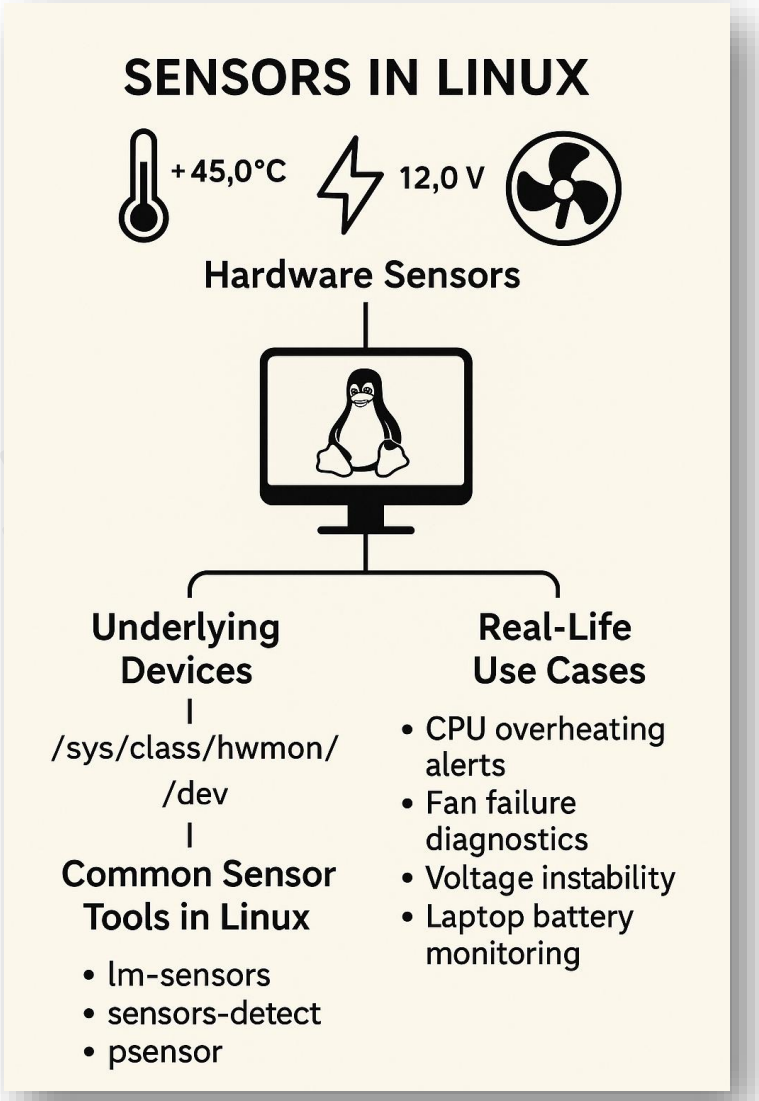
Date: Aug-3rd-2025

VMs on Virtual Box do not give real outputs. This is an important command used in data center equipment audit.

Step 1: Install lm-sensors
dnf install lm_sensors
Or yum install lm_sensors -y
For Debian / Ubuntu => apt install lm-sensors

Step 2: Detect sensors
sensors-detect
Answer **"YES"** to all prompts

Step 3: Check CPU temperature
sensors



Date: Aug-3rd-2025

USERS - RHCSA EXAM QUESTION

/etc

#ls /etc

Date: Aug-3rd-2025

Configuration Hub

The **/etc** directory is one of the most **critical system directories**. It holds **system-wide configuration files**

- If you're **troubleshooting or configuring** the system, /etc is where you'll spend most of your time.
- Stores **configuration files** for the system and installed services.
- Affects how the system behaves on startup and runtime.

System-Wide Settings

- Configurations in /etc apply to **all users**, unlike ~/.config (per user).

Contains Text Files

•Most config files are **readable text files** and can be edited with **vi editor**

Critical Subdirectories

- /etc/network/ – Network configuration (Debian)
- /etc/sysconfig/ – RHEL/CentOS network, firewall, and service configs
- /etc/ssh/ – SSH server/client settings
- /etc/systemd/ – systemd unit files
- /etc/cron.d/, /etc/crontab – Scheduled tasks
- /etc/yum.repos.d/ – YUM repository configurations

User and Authentication Files

- /etc/passwd – User account information
- /etc/shadow – Encrypted passwords
- /etc/group – Group definitions
- /etc/sudoers – sudo access control (edit with **visudo**).

Boot-Time Configuration

Contains files that control **startup behavior**, services, and kernel modules.

Back It Up!

Always **backup /etc** before making changes. A mistake can break the system

Date: Aug-3rd-2025

USERS - RHCSA EXAM QUESTION

/var

#ls /var

Date: Aug-3rd-2025

Configuration Hub

The /var directory stands for **"variable"** – it stores files that **change frequently** during system operation.

Dynamic Data Storage

- Holds **runtime data** that changes, such as logs, emails, and caches.
- Unlike /etc (static configs), /var content **grows over time**.
- \

Important Subdirectories

- /var/log/ – **System logs** (e.g., /var/log/messages, /var/log/secure)
- /var/spool/ – Jobs waiting (**cronjobs** included) to be processed (e.g., print or mail queue)
- /var/cache/ – Cached files for applications/package managers
- /var/tmp/ – Temporary files (more persistent than /tmp)
- /var/lib/ – State info for services (e.g., dpkg, rpm, mysql)

Used by Services and Daemons

- Web servers (like Apache) store runtime files here.
- Package managers store metadata in /var.

Can Grow Large

- Monitor disk usage in /var – especially /var/log/.
- **Uncontrolled growth can fill up disk and crash services**.

Critical for System Health

- Deleting or mismanaging files here can break logging, updates, or daemons.

How logs work in Linux

Date: Aug-3rd-2025

The Big Picture

Linux has **two major systems** for logging:

1. **rsyslog** – The **traditional** logging daemon
2. **systemd-journald** – The **newer** logging system used with **journalctl**

Both are involved in **handling logs** — many of which end up in the **/var/log/** directory.

NOTE: "Think of systemd-journald as the **collector**, rsyslog as the **file writer**, and /var/log/ as the **library** where logs live."

1. rsyslog – The Traditional Log Writer

What it does:

Listens for log messages from the kernel, services, and apps.

Where it writes:

Writes logs as **text files under /var/log/**, such as:

- /var/log/messages
- /var/log/secure
- /var/log/maillog

Configuration File (remember /etc folder!):

/etc/rsyslog.conf and /etc/rsyslog.d/

Example:

tail –f /var/log/messages

2. journalctl – The systemd Log Viewer

What it does:

Reads logs collected by **systemd-journald**, which captures logs **in binary format** (not plain text).

Where logs are stored:

- /var/log/journal/ → Persistent storage if enabled
- Or in memory (lost after reboot if not configured)

Command to view logs:

journalctl

Useful filters:

journalctl -u sshd # Logs for SSH service journalctl --since "1 hour ago"

How They Work Together

Component	Role	Storage Location
systemd-journald	Collects logs (new system)	/run/log/journal/ or /var/log/journal/
rsyslog	Processes + writes to text files	/var/log/
/var/log/	Directory where traditional logs live	Files created by rsyslog, cron, etc.
journalctl	Reads systemd-journald logs	Uses binary journal data

What is logrotate?

Date: Aug-3rd-2025

logrotate is a **log management tool**. It automatically:

- **Rotates** (renames) logs.
- **Compresses** old logs.
- **Deletes** logs older than X days.
- **Prevents /var/log from filling up.**

Practice: Basic logrotate Steps

Step 1: Check Existing Config

```
[root@myserver ~]# cat /etc/logrotate.conf
```

This is the **main config file**.

Step 2: See Service-Specific Configs

```
[root@myserver ~]# ls /etc/logrotate.d/
btmp  chrony  dnf  firewalld  kvm_stat  rsyslog  sssd  wtmp
```

You will find **individual config files** for apps like rsyslog, httpd, yum, etc.

ANALYZE AND STORE LOGS

Best Practices

1. Ensure proper permissions (logs often require root access).
2. Regularly rotate logs to prevent large file sizes (managed via **logrotate**).
3. **Logrotate** is used to manage log files, ensuring they don't grow too large or fill up disk space. It can automatically rotate, compress, and delete old logs based on specified rules.
4. Back up critical logs for post-incident analysis.

#cat /etc/logrotate.conf

[nitadmin1@localhost ~]$ cat /etc/logrotate.conf
see "man logrotate" for details
global options do not affect preceding include directives
rotate log files weekly
weekly
keep 4 weeks worth of backlogs
rotate 4
uncomment this if you want your log files compressed
#compress
packages drop log rotation information into this directory
include /etc/logrotate.d
system-specific logs may be also be configured here.

Errors and Warnings:

- Look for severity levels, for example, **error**, **critical**, **warn**, **fail**, or **denied**.
- Application logs may use unique severity codes (e.g., Apache logs use **INFO, WARNING, ERROR**).

#grep "error" /var/log/messages
Time and Date: Each entry has a Timestamp
Source: server or application
Message content: read the content

6

CHAPTER 6
WebServer / Website Hosting Project

Date: Aug-17th-2025

Using the rules and strategies in this section you will gain confidence and a deep insight in setting up yourself for success

- What is Internet?
- LAN, WAN, Client – Server
- Introduction to the concept of Domain Name System (DNS) Servers
  - How Does Linux work? (Hint: #cat /etc/nsswitch.conf)
  - Mastering the “ping” command
- What is “Cache” ?
- Provisioning of AWS EC2 Instance with key-pair to complete the Web Server / Website Hosting Project
- Downloading and Using PuTTy to connect an EC2 Instance
- How a package is installed in Linux - **1st Package => dnf install httpd”**
- How using “**systemctl”** - a command-line tool we can interact with and control systemd – the init system and service manager for Linux
- Learn and master copy command in Linux
- Learn and master delete command in Linux

Date: Aug-17th-2025

Application Support: What are we learning?

1. Introduction to Linux Service Management (system/systemctl/httpd)
2. How WinSCP is used in real work environment to move files from Window to Linux
3. How to really understand the Logical File System of Linux
4. How to copy files and move them around the system
5. How to delete files and directories
6. Introduction to “zip” and “unzip” which are part of a number of compression tools that we will learn like TAR.
7. Introduction to Web applications and how they work on Linux – Web Hosting
8. Mastering “Absolute Path” and “Relative Path”
9. Introduction to “cache” and “memory”

THE INTERNET

Date: Aug-17th-2025

“The internet is connected over wire/fiber/cables”

Local Area Network (LAN)

“Are you wondering how about over Ocean”

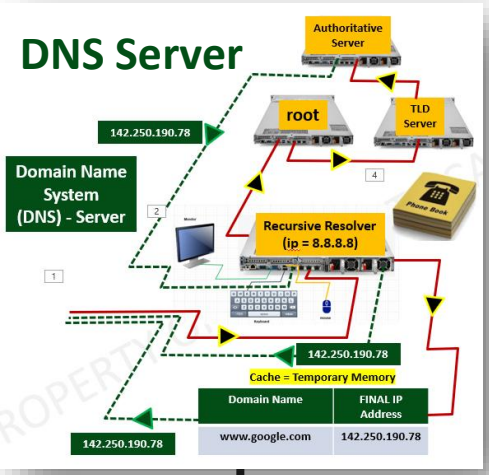
https://www.submarinecablemap.com/country/united-states

https://www.youtube.com/watch?v=WJqg3B2_Kwc

THE INTERNET

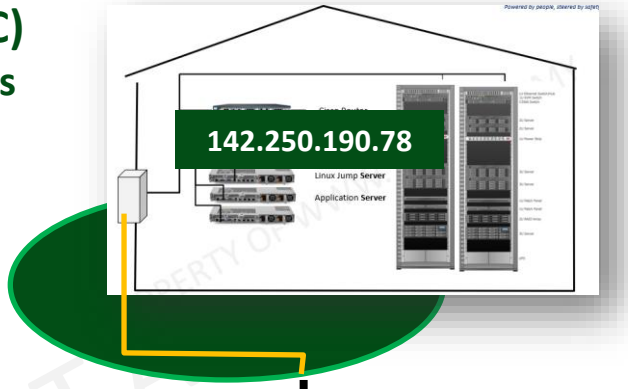
Date: Aug-17th-2025

DNS Server



Data Center (DC)

- Google Servers
- Files sent to client to view google home page



OPEN 24/7

Client

Making a direct Request

Google.com

What happens when we type a URL?

Date: Aug-17th-2025

www.google.com

Home/Work

www.google.com
name server 8.8.8.8

142.250.190.78

VPN, Tunnel, OpenVPN

Internet Service Provider (ISP)

Client

142.250.190.78

Domain Name System (DNS) - Server

Authoritative Server

root

TLD Server

142.250.190.78

Recursive Resolver (ip = 8.8.8.8)

Monitor

Keyboard

mouse

Phone Book

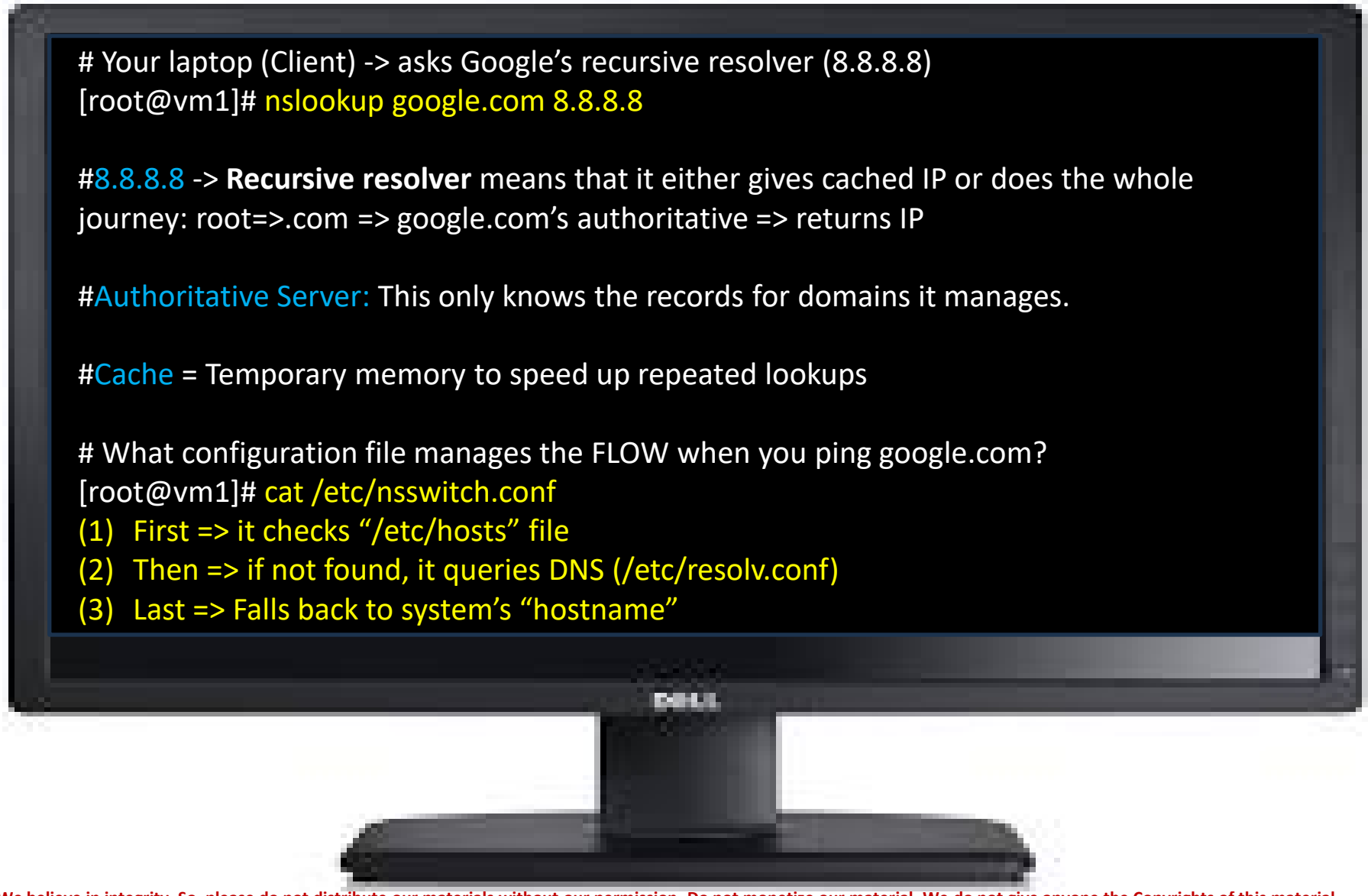
142.250.190.78

Cache = Temporary Memory

Domain Name	FINAL IP Address
www.google.com	142.250.190.78

DNS ON LINUX?

Date: Aug-17th-2025



DNS Resolution Flow Summary

Date: Aug-17th-2025

1. Client (Your Laptop) asks Recursive Resolver, send DNS Queries to:
   - 8.8.8.8 (Google) or 1.1.1.1 (Cloud Flare)
   - Linux Configuration File => **#cat /etc/resolv.conf** => give a list of recursive resolvers:
     - nameserver 8.8.8.8 (Recursive Resolver)
2. Resolver queries Root DNS Server
3. Root DNS Server directs to TLD Server (.com)
4. TLD Server directs to Authoritative Server (google.com)
5. Authoritative Server returns Final IP address
6. Resolver sends IP back to Client

Ping Resolution Flow Diagram

Date: Aug-17th-2025

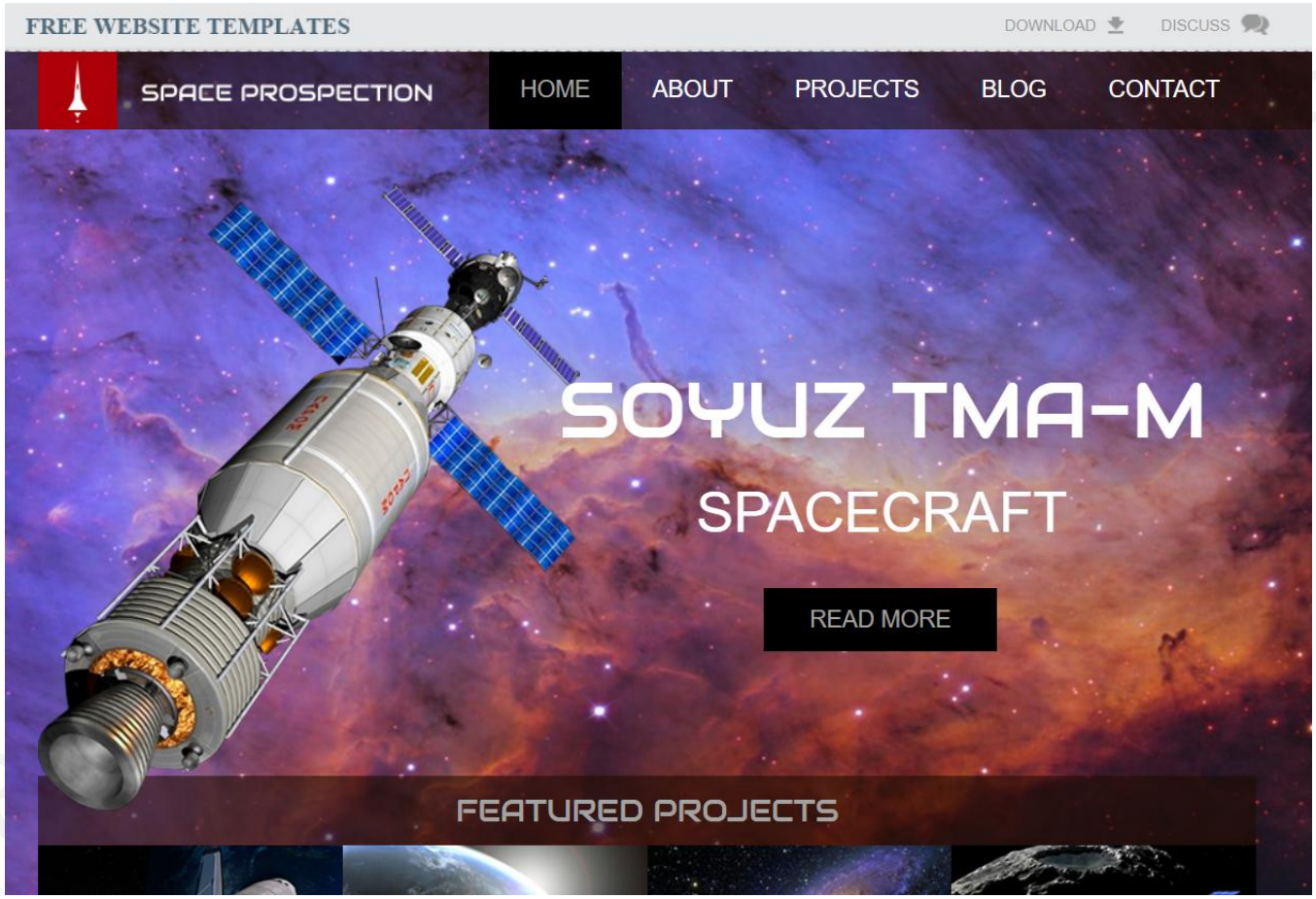
- **ping localhost (127.0.0.1 / ::1):** Tests only the OS TCP/IP stack via loopback **(green).**
- **ping <machine_IP>:** Adds test of NIC hardware & driver **(blue).**
- **ping 8.8.8.8**: Full path test: NIC → Router → ISP → Internet (red).
- **ping google.com**: Adds DNS resolution to full path **(purple).**

Command	What it Tests	What it Confirms	What it Does **NOT** Test
ping localhost (127.0.0.1 or ::1)	Loopback interface + local IP stack	✔ TCP/IP stack is up ✔ ICMP works locally ✔ OS networking is functional	❌ Physical NIC ❌ Router ❌ Internet connectivity
ping <machine_IP> (e.g. 192.168.1.100)	Local NIC + driver + OS network stack	✔ NIC card is working ✔ OS can talk through its own interface ✔ IP assigned correctly	❌ Router connectivity ❌ Internet access
ping 8.8.8.8 (Google DNS)	End-to-end connectivity	✔ Local NIC works ✔ Router/gateway works ✔ ISP path to Internet works	❌ DNS resolution (since IP is given directly)
ping google.com	End-to-end + DNS resolution	✔ Local NIC works ✔ Router/ISP path works ✔ Internet reachable ✔ DNS is working	❌ Nothing major — this is the *full test*

How a Browser Opens a Website hosted on Linux?

Date: Aug-17th-2025

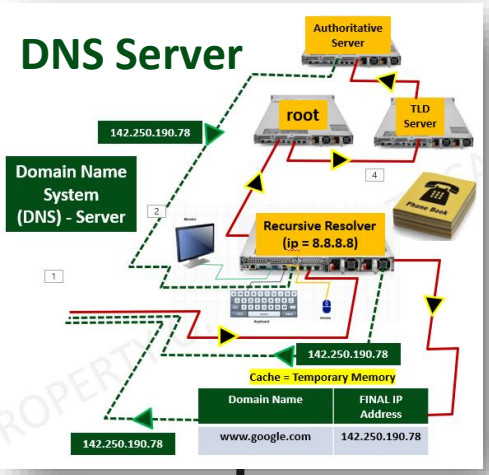
- HTML, CSS, JavaScript with **Apache (httpd)**



THE INTERNET

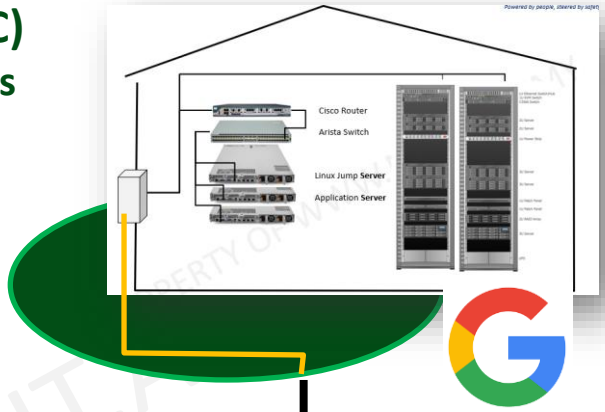
Date: Aug-17th-2025

DNS Server



Data Center (DC)

- Google Servers
- Files sent to client to view google home page

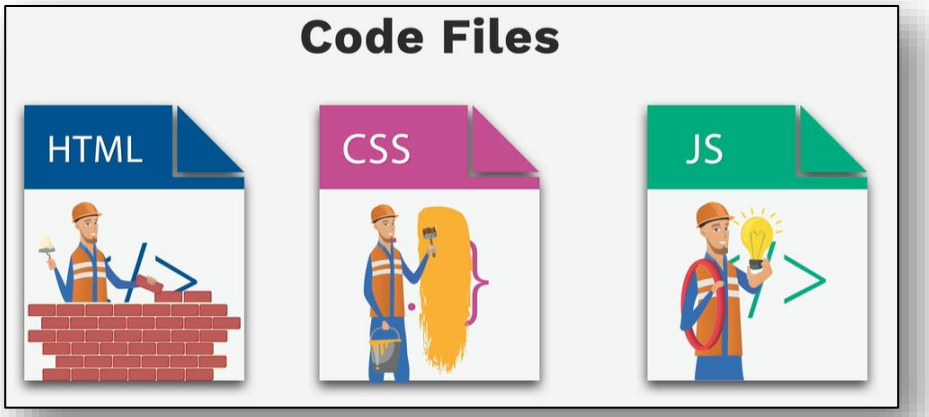


Client

142.250.190.78

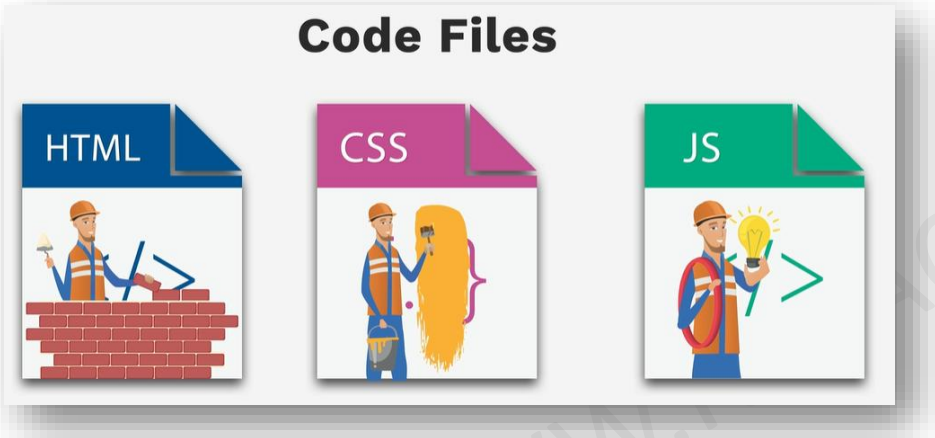
Making a direct Request

Code Files



Let us Explore Developer Tools

Date: Aug-17th-2025



- Inspect a page
- Chrome Developer Tool
- Information about a page

Date: Aug-17th-2025

Monitor and Control Services / Daemon

SERVICES / DAEMONS

Systemd Overview

Date: Aug-17th-2025

- **Systemd (daemon)** was designed as an upgrade to **init**.
- It is a process that runs in the background, and it starts in parallel when a system boots.
- **Systemd** is now part of RedHat enterprise Linux and is responsible for **starting and stopping** services. Not only services but a variety of other items as well.
- **Systemd** combined with Service Manager start/stop service that are called units.
- Units can be many things. One of the most important unit types is the service.
- **Services** are processes that provide specific functionality and allow connections from external clients coming in, such as **SSH service**, the **Apache web service**, and many more.
- Apart from services, other unit types exist, such as **socket**, **mount**, and **target**. To display a list of available units, type **systemctl -t help**.

```
[nitadmin1@System-Admin ~]$ systemctl -t help
Available unit types:
service
mount
swap
socket
target
device
automount
timer
path
slice
scope
```

SYSTEMCTL

Date: Aug-17th-2025

The **systemctl** command has six options to control a **systemd service**. These options are **start**, **stop, restart, reload, enable** and **disable**.

We can categorize these options into **two types:**

Boot time - **Enable** and **disable** are the boot time options. These options control a service at boot time.

Run time - **Start**, **stop**, **restart**, and **reload** are the run-time options. These options control a service on a running system in the current

```
[root@localhost nitadmin1]# systemctl start httpd
```

Action/operation	Command
Start	#systemctl start [name-of-service].service
Stop	#systemctl stop [name-of-service].service
Restart	#systemctl restart [name-of-service].service
Reload	#systemctl reload [name-of-service].service
Enable	#systemctl enable [name-of-service].service
Disable	#systemctl disable [name-of-service].service

Date: Aug-17th-2025

```
[nitadmin1@localhost ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
     Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-01-25 18:03:41 CST; 14h ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 692 (sshd)
      Tasks: 1 (limit: 4643)
     Memory: 5.1M
        CPU: 109ms
     CGroup: /system.slice/sshd.service
             └─692 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Jan 25 18:03:41 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
Jan 25 18:03:41 localhost.localdomain sshd[692]: Server listening on 0.0.0.0 port 22.
Jan 25 18:03:41 localhost.localdomain sshd[692]: Server listening on :: port 22.
Jan 25 18:03:41 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
Jan 26 08:45:39 localhost.localdomain sshd[1456]: Accepted password for nitadmin1 from
192.168.1.217 port 49184 ssh2
Jan 26 08:45:39 localhost.localdomain sshd[1456]: pam_unix(sshd:session): session opened for user
nitadmin1(uid=1000) by nitadmin1(uid=0)
```

Practice Lab

```
[root@localhost nitadmin1]# netstat -tunap | grep -i listen
```

```
[root@localhost nitadmin1]# dnf/yum install net-tools -y
```

Date: Aug-17th-2025

CLI Command "systemctl" -> "systemd"

We are talking about Linux Service Management

Explanation

1. **User/Admin**: You (client) type commands like #**systemctl start httpd**.
   - **systemctl**: This command forwards your request to **systemd**.
2. **systemd**: The core service manager then **interprets the request**, manages dependencies, logs events, etc.
3. **"systemd interprets the request => Services/Units** which is the actual programs (**httpd**, sshd, nginx, etc.) that you (the client) is trying to either enable/start/stop/restart done by systemd.

Car Analogy

- **systemctl** → Car Remote (you press buttons)
- **systemd** → Car ECU / Brain (executes commands, controls dependencies)
- **Services** → Car Components (engine, lights, AC, radio)

systemctl = The Remote Control

Date: Aug-17th-2025

#systemctl start httpd.service

- Command-line tool
- You (Client) sends instructions: start, stop, status, restart

Example:

- **#systemctl start httpd** → tells systemd to run Apache
- $sudo systemctl **enable --now sshd.service** → tells system to enable and start sshd service daemon
- $sudo systemctl **stop firwald.service** → tells systemctl to stop firwald service.

Date: Aug-17th-2025

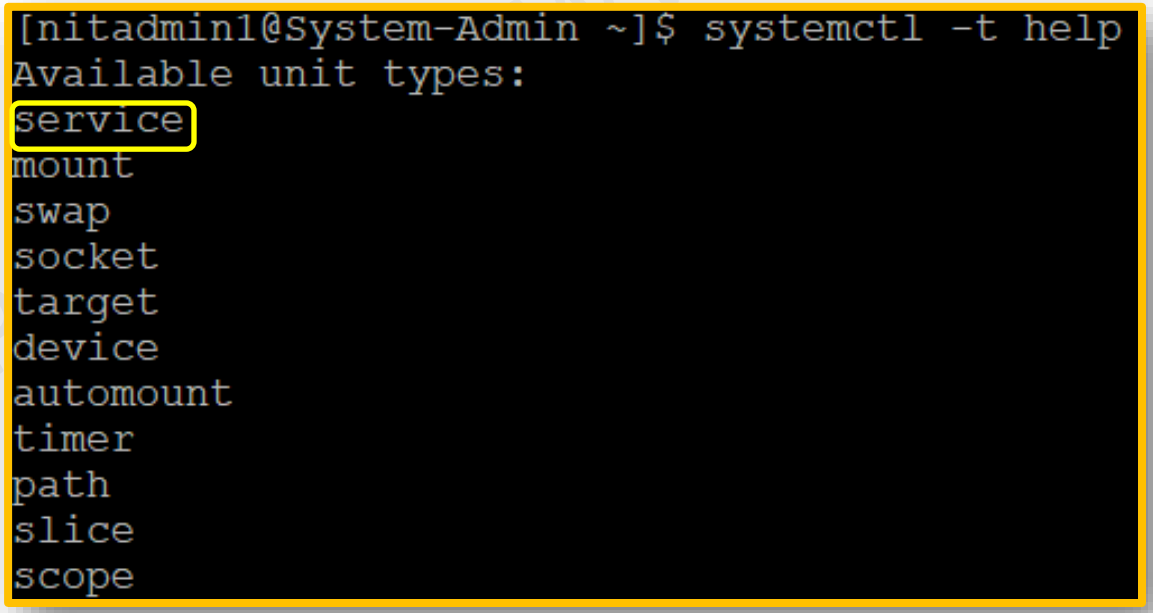
systemd = The Brain / Engine Controller

- **Core init** system & service manager (PID 1)
- Receives requests from **systemctl**
- Starts/stops services, manages dependencies, handles logs
- Ensures services start in the right order

Date: Aug-17th-2025

Services = The Car Parts

- Actual programs managed by **system that run when the brain (system) tells to take action (start,stop etc):**
  - Httpd (Apache Web Services)
  - Sshd (secure shell login services)
  - Firwald Services
  - Nginx services
  - Crond cron services / scheduling within the linux system



Typing a URL

Date: Aug-17th-2025

1. In our project the User (client) types a URL in browser
2. Example: http://192.168.1.100/index.html
3. Browser prepares to request this file from the server
4. Browser says: 'Hey server, send me index.html!’
5. Request goes over HTTP protocol (port 80)

WebServer - httpd (Apache) Receives the Request

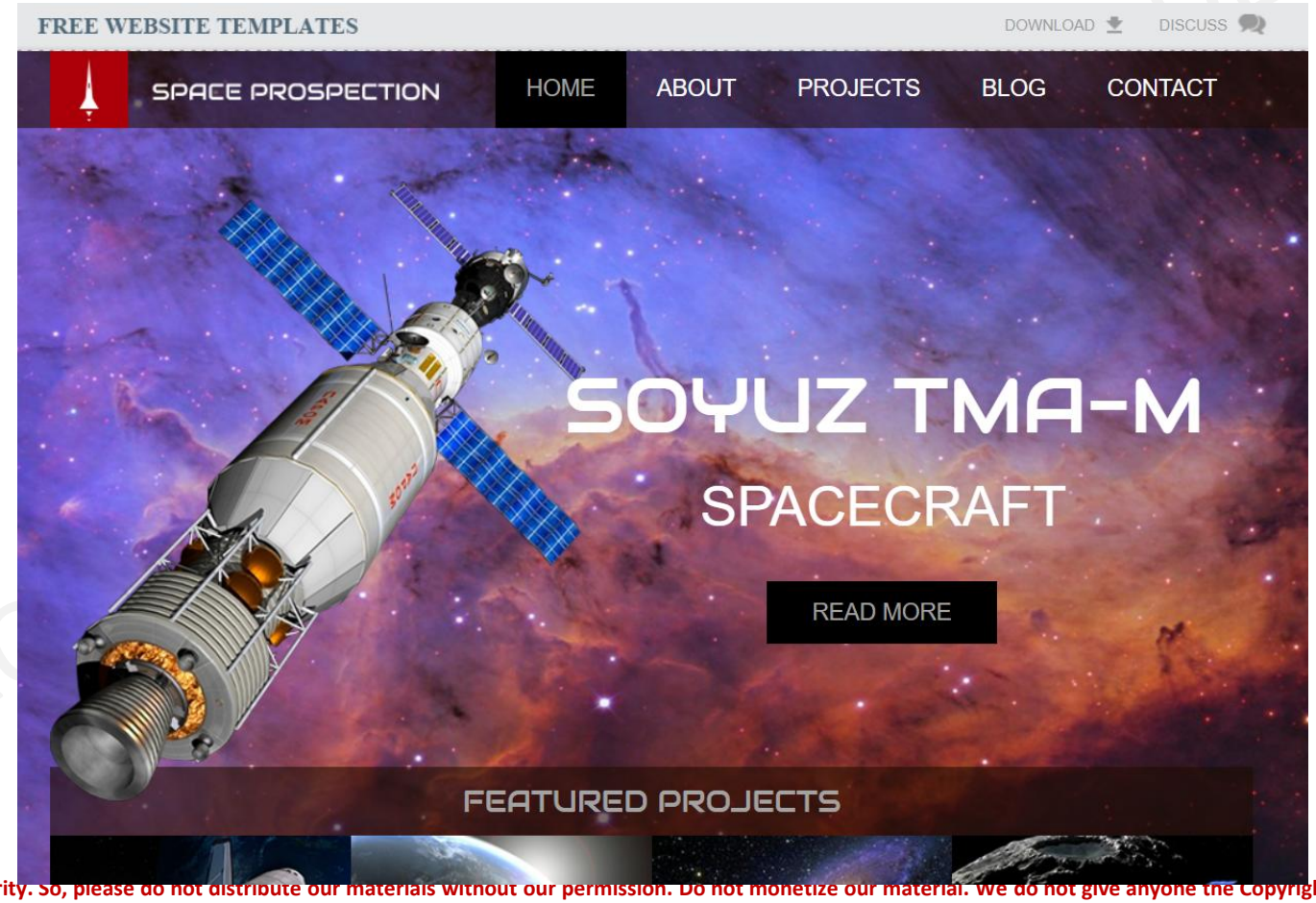
Date: Aug-17th-2025

- httpd service must be running on the Webserver
- Listening on port 80
- Looks into **web root folder→** /var/www/html/
- Finds **index.html**
- Apache sends index.html file to the browser
- Browser starts reading HTML structure
- HTML may include links to CSS & JS:
- <link rel='stylesheet' href='style.css'>
- <script src='app.js'></script>
- Browser makes new requests for these files
- Apache sends them back from /var/www/html/

Browser Builds the Page

Date: Aug-17th-2025

- HTML → Structure of page
- CSS → Styling (colors, layout)
- JavaScript → Interactivity (buttons, forms, animations)



Quiz: How Does a Browser Open a Website?

Date: Aug-17th-2025

1. What service must be running for the browser to access /var/www/html?
2. Which protocol and port are used by the browser to send the request?
3. Where does Apache look for index.html by default?
4. What role does HTML play? CSS? JavaScript?
5. In the restaurant analogy, what does httpd represent?

🌐 What is HTTP?

Date: Aug-17th-2025

- HTTP = **HyperText Transfer Protocol**
- A set of rules (protocol) for browser ↔ server communication
- Used to request and deliver webpages, images, CSS, JS, etc.
- Default port: 80 (HTTPS uses port 443)

Example:

- Browser: GET /index.html
- Server: Here's index.html

Analogy:

- Like speaking the same language with a waiter in a restaurant

🔒 What is HTTPS?

Date: Aug-17th-2025

- HTTPS = HyperText Transfer Protocol Secure
- Same as HTTP, but with encryption (SSL/TLS)
- Ensures data cannot be read or changed during transfer
- Default port: 443

Benefits:

- Security: Protects passwords, credit cards, personal info
- Trust: Browser shows a padlock icon
- Required by most modern websites

Analogy:

- Like speaking to the waiter in a private, encrypted language
- so no eavesdropper in the restaurant can understand you

HTTP vs HTTPS Comparison

Date: Aug-17th-2025

Feature	HTTP	HTTPS
Full Form	HyperText Transfer Protocol	HyperText Transfer Protocol Secure
Port Number	**80**	**443**
Security	No encryption Data can be intercepted	**Encrypted using SSL/TLS** **Protects data**
Browser Indicator	No padlock icon	Padlock icon shown

http://example.com

No padlock (Not secure)

https://example.com

Padlock shown (Secure)

Summary: How a Website Loads

Date: Aug-17th-2025

- Browser sends an HTTP/HTTPS request when you type a URL
- Apache (httpd) receives the request on port 80/443
- Apache looks inside /var/www/html/ for the requested files
- HTML = Structure | CSS = Styling | JavaScript = Interactivity
- HTTP = Not secure (port 80) | HTTPS = Secure (SSL/TLS, port 443)

- Big Picture:
- Browser ↔ (HTTP/HTTPS) ↔ Apache ↔ /var/www/html ↔ Webpage

INSTALL AND UPDATE SOFTWARE PACKAGES

VM CONFIGURATION
LINUX - PACKAGE MANAGEMENT
(dnf/yum & rpm)

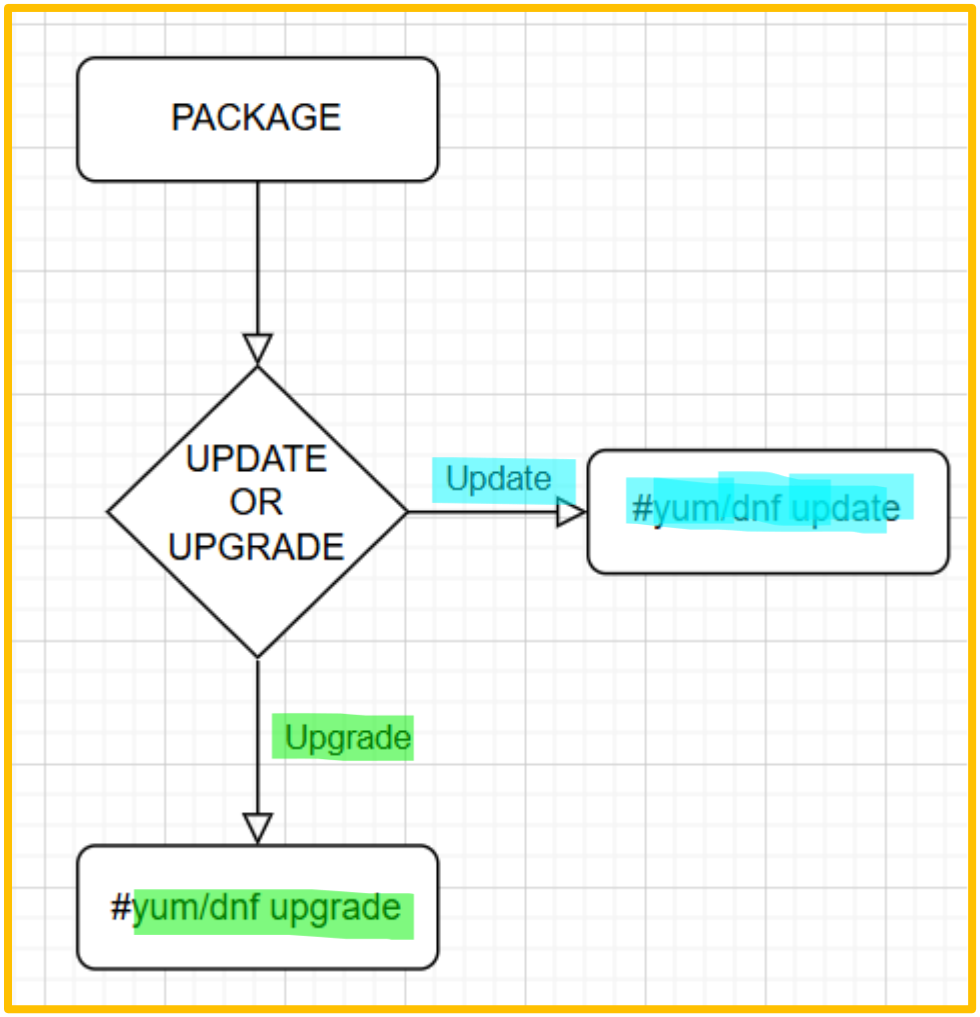
A package in Linux means Software

RPM vs YUM/DNF

- **RPM** is a low-level tool. You must manually handle dependencies.
- **YUM** (Yellowdog Updater Modified) and **DNF** (Dandified Yum) are higher-level tools that use RPM but automatically resolve dependencies.

PACKAGE UPDATE / UPGRADE / MAJOR / MINOR

Date: Aug-17th-2025



Update

Applying minor changes or patches to installed packages - bug fixes, security patches, or small enhancements within the same version of the software.
It does not change the software's major version. Minor Updates within the same major version (httpd-2.4.6-97 → httpd-2.4.6-100 but does not install 2.6)

Upgrade

Replacing installed packages with a newer major version- introducing significant changes, new features, or backward-incompatible changes.
Changes the major version (e.g., httpd-2.4.6-97 → httpd-2.4.6-100 and might replace httpd-tools with httpd-tools-new). May introduce new features, significant changes, and possibly require configuration updates or compatibility checks.

RPM

Date: Aug-17th-2025

How do you Query RPM to see the package that has Linux commands?

```
# which <command name>     (to display the location of that command)
# rpm -qf <location of the command> (to check the package name for that command)
# rpm -V <package name>    (to verify that package, ie., 100% package is
there or not, if any files missed in that package, those are displayed as a list)
```

Practice Lab

STEP 1: Locate the comamnd

[nitadmin2@VM01 etc]$ which ls
/usr/bin/ls

STEP 2: Check package name for "ls" command

[nitadmin2@VM01 etc]$ rpm -qf /usr/bin/ls
coreutils-8.32-36.el9.x86_64 **<= This is the package / software**

STEP 3: Check package has nothing missing

[nitadmin2@VM01 etc]$ rpm -V bash-5.1.8-9.el9.x86_64
[nitadmin2@VM01 etc]$

RPM

Date: Aug-17th-2025

How do you Query RPM to see all installed Packages?

rpm -qa (to list all installed packages)
rpm -qa |wc -l (to count how many packages already installed)
rpm -qa --last | less (to check last installed packages)
rpm -qa |grep -i <package name> (to check the specified package is installed or not)

Installing Packages?

 # rpm -ivh coreutils-8.32-36.el9.x86_64 (to install the package. Common in Patching)
rpm -ivh --test <package name> (to check the package consistency)
rpm -ivh <package name> --force (to install the package forcefully)
NOTE: If the installation status shows 100%, then the package is in good condition or consistent. But while showing the hash progress if it shows any error, then the package is in inconsistent state.
rpm --test -ivh (to test the package before installing ie., whether the package is suitable or not)

RPM

Date: Aug-17th-2025

How do you remove a Package

rpm -e coreutils-8.32-36.el9.x86_64 (to erase or remove or uninstall the package)
rpm -evv <package name> (to remove the package in verbose mode)

What are the dependencies of a Package?

rpm -qR coreutils-8.32-36.el9.x86_64 (to list the dependencies of that package)
rpm -qip <package full name> (to display the package information before installation)

How do you see information or details about a Package?

rpm -qi <package name> (to see the details or information on the installed package)
rpm -ql <package name> (to list all package related files)
rpm -qlc <package name> (to list all the configuration files of that package)
rpm -qd <package name> (to list all the document files of that package)

DNF/YUM - DOCUMENTATION

Date: Aug-17th-2025

vm1@System-Admin:~

```
DNF(8)                                  DNF                                  DNF(8)

NAME
       dnf - DNF Command Reference

SYNOPSIS
       dnf [options] <command> [<args>...]

DESCRIPTION
       DNF is the next upcoming major version of YUM, a package manager for
       RPM-based Linux distributions. It roughly maintains CLI compatibility
       with YUM and defines a strict API for extensions and plugins.

       Plugins can modify or extend features of DNF or provide additional CLI
       commands on top of those mentioned below. If you know the name of such
       a command (including commands mentioned below), you may find/install
       the package which provides it using the appropriate virtual provide in
       the form of dnf-command(<alias>), where <alias> is the name of the com□
       mand; e.g.``dnf install 'dnf-command(versionlock)'`` installs a ver□
       sionlock plugin. This approach also applies to specifying dependencies
       of packages that require a particular DNF command.

       Return values:
```

YUM/DNF

Date: Aug-17th-2025

https://access.redhat.com/articles/yum-cheat-sheet

How to INSTALL a package

yum install tree
yum install tree -y

Clear out Cached Memory

yum clean packages

List Installed Packages and check for any updates

yum list installed or # yum list all
yum check-updated (Check for updates)

Delete / Remove a Package

yum remove <package_name> (package name = vsftpd, httpd

DELETING (REMOVE) IN LINUX

Date: Aug-17th-2025

#rm - file

#rm –rf - directory

```
[vm2@System-Admin ~]$ ls
a.txt  b.txt  dir1  file10  my_backup
[vm2@System-Admin ~]$ rm file10
[vm2@System-Admin ~]$ rm my_backup
rm: cannot remove 'my_backup': Is a directory
[vm2@System-Admin ~]$ rm -rf my_backup
[vm2@System-Admin ~]$ ls
a.txt  b.txt  dir1
```

2> /dev/null

Use **2> /dev/null** to silence permission errors (or use sudo to gain all permissions)