Linux Operating System

by
Jitendra Singh Tomar

Index

What is UNIX?

UNIX is a multiuser, multitasking, portable, and time-sharing operating system originally developed in the late 1960s and early 1970s at AT&T's Bell Labs by Ken Thompson, Dennis Ritchie, and others.

Key Characteristics of UNIX:

Feature	Description
Multiuser	Multiple users can use the system simultaneously.
Multitasking	Can run multiple processes at the same time.
Portability	Written in C, so it can be easily modified and ported to different hardware.
Security	Has built-in user authentication, file permissions, and encryption tools.
Stability	Known for being stable and running for years without reboot.
Networking	Strong TCP/IP networking support-built in.
Shell Interface	Provides a command-line interface (CLI) using shells like **sh**, **bash**, **ksh**, **csh**, etc.

History and Evolution:

1. **1969** – Developed at Bell Labs.
2. **1971** – First edition of UNIX released.
3. **1973** – Rewritten in **C language** (made it portable).
4. **1980s** – Many variants emerged: **System V (AT&T)**, **BSD (Berkeley Software Distribution)**.
5. **1991** – Inspired **Linux**, a UNIX-like system.

Popular UNIX & UNIX-like Systems:

- **True UNIX systems**:
    - IBM AIX
    - HP-UX
    - Solaris (originally by Sun Microsystems)
    - SCO UNIX
- **UNIX-like systems (not certified but similar)**:
    - **Linux** (Ubuntu, CentOS, RHEL, Debian, etc.)
    - **macOS** (based on BSD)
    - **FreeBSD**, **NetBSD**, **OpenBSD**

What is Linux?

Linux is a free and open-source, UNIX-like operating system kernel originally created by Linus Torvalds in 1991.

Today, the term "Linux" often refers to Linux-based operating systems (called distributions) that use the Linux kernel along with other software (mostly from the GNU Project).

Key Characteristics of Linux:

Feature	Description
Open-source	Source code is freely available for anyone to use, modify, and distribute.
UNIX-like	Follows UNIX design principles (but is not derived from original UNIX code).
Multiuser & Multitasking	Supports multiple users and tasks at the same time.
Modular Kernel	Kernel supports modules (drivers, file systems) loaded at runtime.
Security	Strong permission model, SELinux/AppArmor, encryption tools.
Customizability	Users can tweak every part of the system.
CLI & GUI Support	Offers powerful command-line interface and desktop environments (GNOME, KDE, etc.)
Portable	Runs on desktops, servers, smartphones, routers, supercomputers, etc.

Brief History:

- **1991**: Linus Torvalds releases Linux kernel v0.01
- **1992**: Kernel licensed under GNU General Public License (GPL)
- **1993 onward**: Various Linux distributions (distros) like Slackware, Debian, Red Hat appear

Major Components of a Linux OS:

1. **Kernel** – Core that manages hardware and system processes.
2. **Shell** – Interface between user and OS (e.g., bash, zsh).
3. **File System** – Organizes data into a hierarchy (/, /home, /etc, etc.).
4. **System Libraries** – Standard libraries used by applications.
5. **Utilities and Applications** – Core tools and optional apps for productivity, networking, etc.

Why is Linux So Popular?

- Free and open-source, Highly-secure and stable, huge community and commercial support
- Flexible (customizable for any use case), Preferred OS for servers and developers

UNIX vs Linux:

Feature	UNIX	Linux
Origin	Developed in 1969 at Bell Labs	Developed by Linus Torvalds in 1991
Source Code	Closed-source (mostly proprietary)	Open-source and free
Cost	Commercially licensed	Mostly free
Usage	Used in servers, mainframes, workstations	Used in servers, desktops, embedded systems
Development Model	Vendor-developed (IBM, HP, Oracle, etc.)	Community and vendor-supported (Red Hat, Canonical, etc.)
Hardware Support	Limited to specific hardware	Wide hardware support
Security	Secure, but patching depends on vendors	Strong community-driven security and updates
Filesystems	UFS, JFS, etc.	ext4, XFS, Btrfs, ZFS, etc.
GUI Support	Minimal or optional	Multiple GUI options (GNOME, KDE, etc.)
Examples	AIX, HP-UX, Solaris	Ubuntu, CentOS, RHEL, Debian, Fedora, Arch
Certification	POSIX-certified	Many distros are POSIX-compliant (not certified)

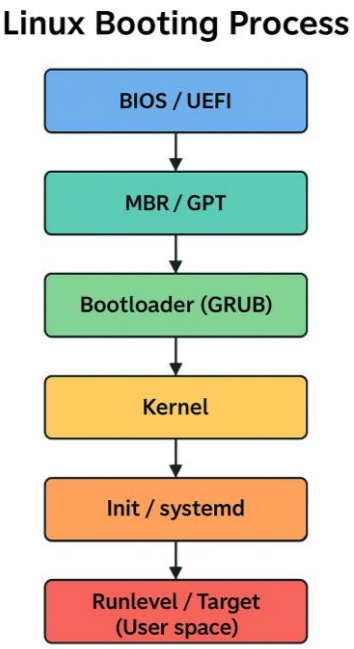
Linux Booting Process – Step by Step

1. **BIOS / UEFI**
2. **MBR / GPT**
3. **Bootloader (GRUB/LILO/etc.)**
4. **Kernel**
5. **Init / systemd**
6. **Runlevel / Target (User space)**

Booting process Step-by-Step Breakdown

1. BIOS / UEFI (Firmware Initialization)

- Performs POST (Power-On Self-Test).
- Detects hardware (CPU, RAM, disk, keyboard).
- Looks for a bootable device (HDD, USB, CD/DVD).
- Passes control to the **MBR (or UEFI partition)**.



2. MBR / GPT (Boot Sector on Disk)

- MBR (first 512 bytes of disk) contains:
  - **Bootloader info**
  - **Partition table**
- In UEFI systems, this is handled via **EFI system partition (ESP)**.
- Loads the **bootloader** (like GRUB).

3. Bootloader (e.g., GRUB2)

- Allows choosing OS/kernel (dual boot if configured).
- Loads the selected **Linux kernel** into memory.
- Passes **initrd/initramfs** and **kernel parameters**.
- Transfers control to the **kernel**.

4. Kernel Initialization

- Mounts root file system (from initramfs).
- Initializes low-level hardware and drivers.
- Starts **init** or **systemd** process (PID 1).
- Transitions from kernel space to **user space**.

5. Init / systemd (First User Space Process)

- init (SysVinit) or systemd (modern distros) is launched.
- Loads system configuration and targets (runlevels).
- Starts background services (network, SSH, etc.).

6. Runlevel / Target (Multi-user Environment)

- Brings the system to a usable state:
  - CLI login (runlevel 3 or multi-user.target)
  - GUI login (runlevel 5 or graphical.target)
- Displays login prompt (getty, gdm, lightdm, etc.)

Principles of Linux:

1. Everything is a File

- In Linux, almost everything (devices, processes, sockets) is treated as a file.
- E.g., /dev/sda (disk), /proc/cpuinfo (CPU info), /etc/passwd (user info).
- This simplifies interaction and access via standard tools (e.g., cat, ls).

2. Small, Single-Purpose Programs

- Linux uses **small utilities** that do one task well.
- Examples:
    - grep – Search text
    - awk – Pattern scanning
    - sed – Stream editing
- These tools can be combined in powerful ways using **pipes (|)**.

3. Chain Programs Together (Pipes and Redirection)

- Programs can be connected using **pipes** to form complex operations.
- Example:

```
ps aux | grep apache | awk '{print $2}'
```

4. Modularity

- The system is built from independent components (kernel, shells, daemons).
- You can swap out or replace parts (e.g., systemd ↔ OpenRC).

5. Open Source and Freedom

- Linux is licensed under **GPL (GNU Public License)**.
- Users have freedom to:
    - View and modify source code
    - Distribute copies
    - Customize the OS as needed

6. Multiuser and Multitasking

- Supports multiple users simultaneously.
- Each user has separate permissions and home directories.
- Can run many tasks (processes) at the same time efficiently.

7. Security and Permissions

- File system permissions: Read, write, execute (for user/group/others).
- Additional security with:
    - sudo, su for privilege escalation
    - **SELinux** or **AppArmor**
    - Firewall (iptables, firewalld)

8. Portability

- Written in **C**, allowing Linux to run on:
    - Desktops
    - Servers
    - Supercomputers
    - Smartphones (Android)
    - Embedded devices (routers, IoT)

9. Filesystems and Hierarchy

- Uses a **hierarchical file system** starting from root (/).
- Well-organized structure: /home, /etc, /usr, /var, etc.

10. Community Collaboration

- Developed and improved by a global community.
- Frequent updates, patches, and improvements.

What is FHS (File System Hierarchy Standard)?

The File System Hierarchy Standard (FHS) defines the directory structure and content in Unix-like operating systems, including Linux. It ensures that files and directories are organized logically and consistently, making it easier for users, applications, and administrators to work across different Linux distributions.

Top-Level FHS Directories and Their Purpose:

Directory	Purpose
/	**Root** directory — the top of the hierarchy. All files and directories stem from here.
/bin	Essential **user binaries** (e.g., ls, cp, mv). Needed in single-user mode and for all users.
/boot	Boot loader files, **Linux kernel**, and **initrd** image. (e.g., vmlinuz, grub/)
/dev	Device files (e.g., /dev/sda, /dev/null). Interface to hardware via files.
/etc	System-wide **configuration files**. (e.g., /etc/passwd, /etc/fstab)
/home	Home directories of regular users (e.g., /home/jitendra).
/lib	Essential shared **libraries** for binaries in /bin and /sbin.
/media	Mount point for **removable media** (USB, CD/DVD).
/mnt	Temporary **mount point** for manual mounting.
/opt	Add-on **optional software packages** (e.g., /opt/google/).
/proc	Virtual filesystem with **kernel and process info** (e.g., /proc/cpuinfo).
/root	Home directory of the **root user**.
/run	Runtime data for processes started since boot.
/sbin	System **binaries for administration** (e.g., reboot, ifconfig).
/srv	Data for **system services** (e.g., web server files).
/sys	Virtual filesystem with info about **hardware and devices** (sysfs).
/tmp	Temporary files. May be cleared on reboot.
/usr	**User-level software**, libraries, and documentation.
/var	Variable data like **logs**, **mail**, **spools**, **cache**, etc.

Breakdown of Important Directories

/bin

- Essential commands available in single-user mode.
- Examples: ls, cat, cp, mv, rm, echo

/etc

- Hostname: /etc/hostname
- Network config: /etc/network/interfaces or /etc/netplan/
- Services: /etc/systemd/system/
- Users: /etc/passwd, /etc/shadow

/usr

- /usr/bin: Most user binaries
- /usr/lib: Shared libraries
- /usr/share: Architecture-independent files (icons, docs)
- /usr/local: Locally installed software

/var

- /var/log: System logs (e.g., /var/log/syslog, /var/log/auth.log)
- /var/spool: Mail, print queues
- /var/cache: Cached data
- /var/tmp: Temporary files (preserved between reboots)

In simple terms:

/

├— bin -> Essential binaries

├— boot -> Boot files (kernel, GRUB)

├— dev -> Device files

├— etc -> System config files

├— home -> User directories

├— lib -> System libraries

├— media -> Mounts for removable devices

├— mnt -> Temporary mounts

├— opt -> Optional apps

├— proc -> Kernel and process info

├— root -> Root user's home

├— run -> Runtime variable data

├— sbin -> Admin/system binaries

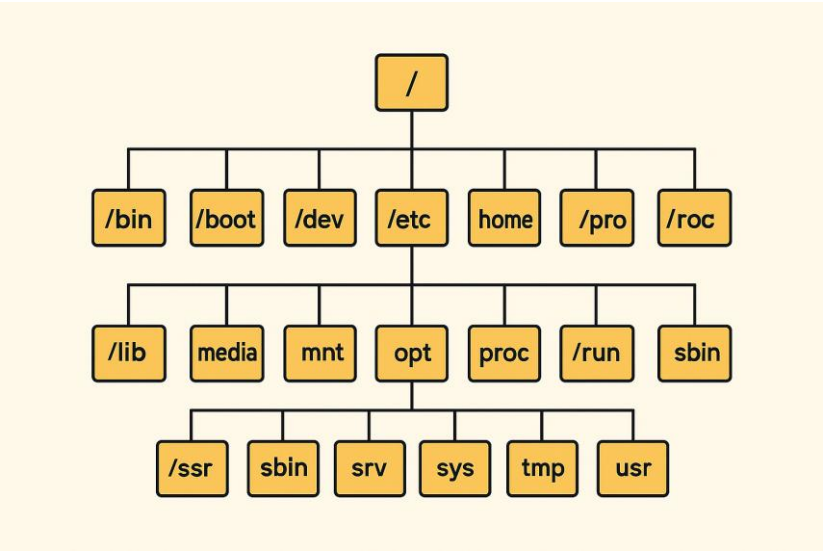
├— srv -> Service data (web, FTP, etc.)

├— sys -> Hardware info

├— tmp -> Temporary files

├— usr -> Secondary hierarchy (user binaries, libraries)

├— var -> Variable data (logs, spool, etc.)



Major Differences between RHEL 6/7/8/9/10

Feature / Component	RHEL 6	RHEL 7	RHEL 8	RHEL 9	RHEL 10 *(Upcoming)*
Init System	**SysV init**	**systemd**	systemd	systemd	systemd (latest version)
Default File System	ext4	XFS	XFS	XFS + Stratis + Btrfs (optional)	Advanced FS support (TBD)
Kernel Version	2.6.32	3.10	4.18	5.14	6.x series (expected)
Desktop Environment	GNOME 2	GNOME 3 (Classic Mode)	GNOME 3.28	GNOME 40+	GNOME 44+ (expected)
Default Firewall	iptables	firewalld	firewalld/nftables	nftables	nftables
Network Manager	Network scripts	nmcli, nmtui, NetworkManager	NetworkManager	NetworkManager	NetworkManager + GUI upgrades
Package Manager	yum	yum	dnf + yum-compat	dnf	dnf5 (expected)
SELinux	Enforcing	Enforcing	Enforcing	Enforcing	Enhanced with AI-based tuning
Rootless Containers	Not available	Limited via Docker	Podman introduced	Podman & Buildah	Podman (advanced), AI hooks
Cockpit Web GUI	Not included	Optional	Default	Default	Default
Security Enhancements	Basic	OpenSSL 1.0.1, auditd	OpenSSL 1.1, fapolicyd	OpenSSL 3, Integrity Measurement	TPM 2.0+ & confidential compute
System Roles	Not available	Basic via Ansible	Extensive via RHEL System Roles	Enhanced System Roles	Expanded Roles + AI Assistants
Cloud Readiness	Low	Medium	Cloud-native support	Full hybrid cloud support	Cloud-native, AI workloads

Linux File Management

Creating Files

- Files are containers for storing data. Creating files is one of the most basic tasks in Linux.
- Commands:
    - touch filename — Creates an empty file or updates the timestamp if it exists.
    - echo "text" > filename — Creates a file with text content.
    - cat > filename — Creates a file and lets you input content manually (Ctrl+D to save).

Modifying Files

- You can modify files by editing their content using command line editors or redirection operators.
- Commands:
    - echo "new text" >> filename — Append text to the file.
    - Use editors like nano, vim, or vi.

Deleting Files

- Deleting removes files from the filesystem.
- Command**:**
    - rm filename — Remove file.

Renaming Files

- Renaming changes the name of the file.
- Command**:**
    - mv oldname newname — Move or rename file.

Copying Files

- Copying duplicates a file.
- Command:
    - cp sourcefile destinationfile — Copies file.

Moving Files

- Moving transfers the file from one location to another or renames it.
- Command**:**
    - mv source destination

Hiding and Unhiding Files

- In Linux, any file or directory starting with a dot (.) is hidden by default.
- Commands**:**
- Rename file to start with “.” to hide.
    - Rename file to remove the leading “.” to unhide.

Table of Commands:

Operation	Command	Notes
Create file	touch filename	Creates empty file
Create file with text	echo "text" > filename	Writes text to file
Modify file (append)	echo "text" >> filename	Append text
Delete file	rm filename	Deletes file
Rename file	mv oldname newname	Moves or renames file
Copy file	cp source dest	Copies file
Move file	mv source dest	Moves file
Hide file	mv filename .filename	Prefix dot to hide
Unhide file	mv .filename filename	Remove dot to unhide
List files	ls -a	Show hidden files
Create directory	mkdir directoryname	Creates folder
Delete empty dir	rmdir directoryname	Deletes only empty folder
Delete dir + content	rm -r directoryname	Recursively delete folder

Linux Permissions

What are Linux Permissions?

Linux permissions control who can access and manipulate files and directories. This is essential for security and multi-user environments.

Permissions define what actions users can perform on files or directories:

- Read (r) — View or read the file contents.
- Write (w) — Modify or delete the file.
- Execute (x) — Run the file as a program or script, or enter/search a directory.

Permission Model in Linux:

User Type	Description
Owner	The user who owns the file.
Group	Users belonging to the file's group.
Others	Everyone else (all other users).

File Types in linux:

The first character in permissions shows the type:

- - : Regular file
- d : Directory
- l : Symbolic link
- c : Character device file
- b : Block device file
- s : Socket
- p : Named pipe (FIFO)

Understanding Permissions Symbols:

Symbol	Meaning
`r`	Read permission
`w`	Write permission
`x`	Execute permission
`-`	Permission not granted
`s`	Setuid or setgid bit (special execute permission)
`t`	Sticky bit (special directory permission)

Numeric (Octal) Permission Representation

Permissions can also be represented numerically (octal):

Permission	Value
Read (r)	4
Write (w)	2
Execute(x)	1

Add the values for owner, group, and others:

Example	Explanation
755	Owner = 7 (r+w+x = 4+2+1)
	Group = 5 (r+x = 4+0+1)
	Others = 5 (r+x = 4+0+1)
644	Owner = 6 (r+w = 4+2)
	Group = 4 (r only)
	Others = 4 (r only)

Changing Permissions: chmod

```
chmod u+x script.sh    # Add execute permission to user (owner)
chmod go-w file.txt    # Remove write permission from group and others
chmod a+r file.txt     # Give read permission to all (user, group, others)
```

Changing Ownership: chown

```
chown username:developers file.txt
```

Changing Group: chgrp

```
chgrp groupname filename
```

Special Permissions

1. Setuid (s on owner's execute bit)

- When set on executable files, runs the file with the permissions of the file owner.
- Used for programs needing temporary elevated privileges.
- Ex: chmod u+s program

2. Setgid (s on group execute bit)

- On files: runs with group privileges.
- On directories: files created inside inherit the directory's group.
- Ex: chmod g+s directory

3. Sticky Bit (t on others' execute bit)

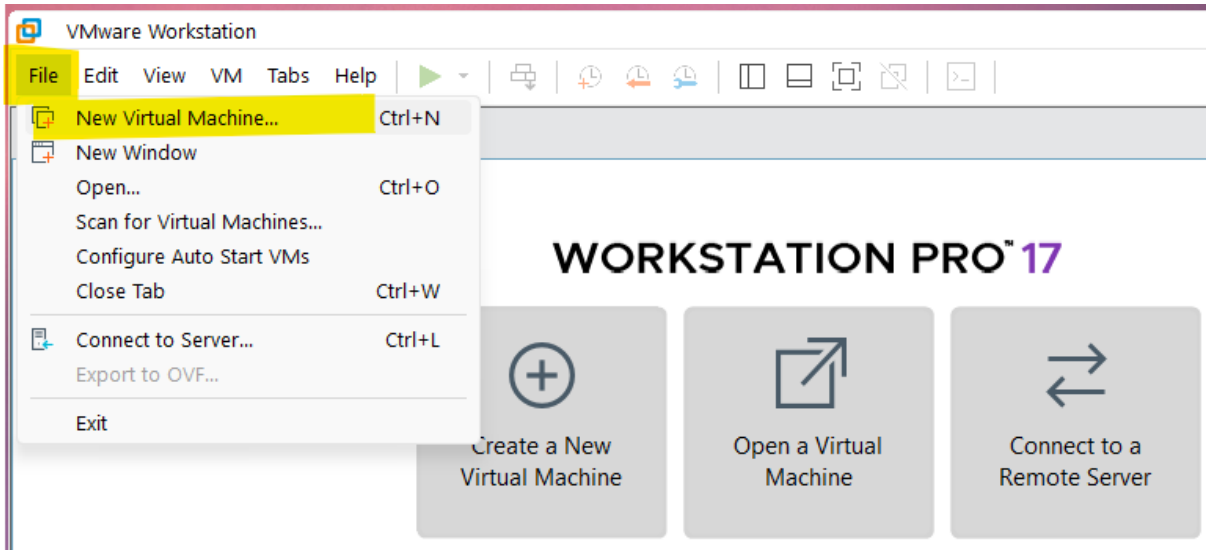
- Mostly used on shared directories (e.g., /tmp).
- Users can only delete their own files, even if others have write permissions on the directory.
- Ex: chmod +t /tmp

Common Commands:

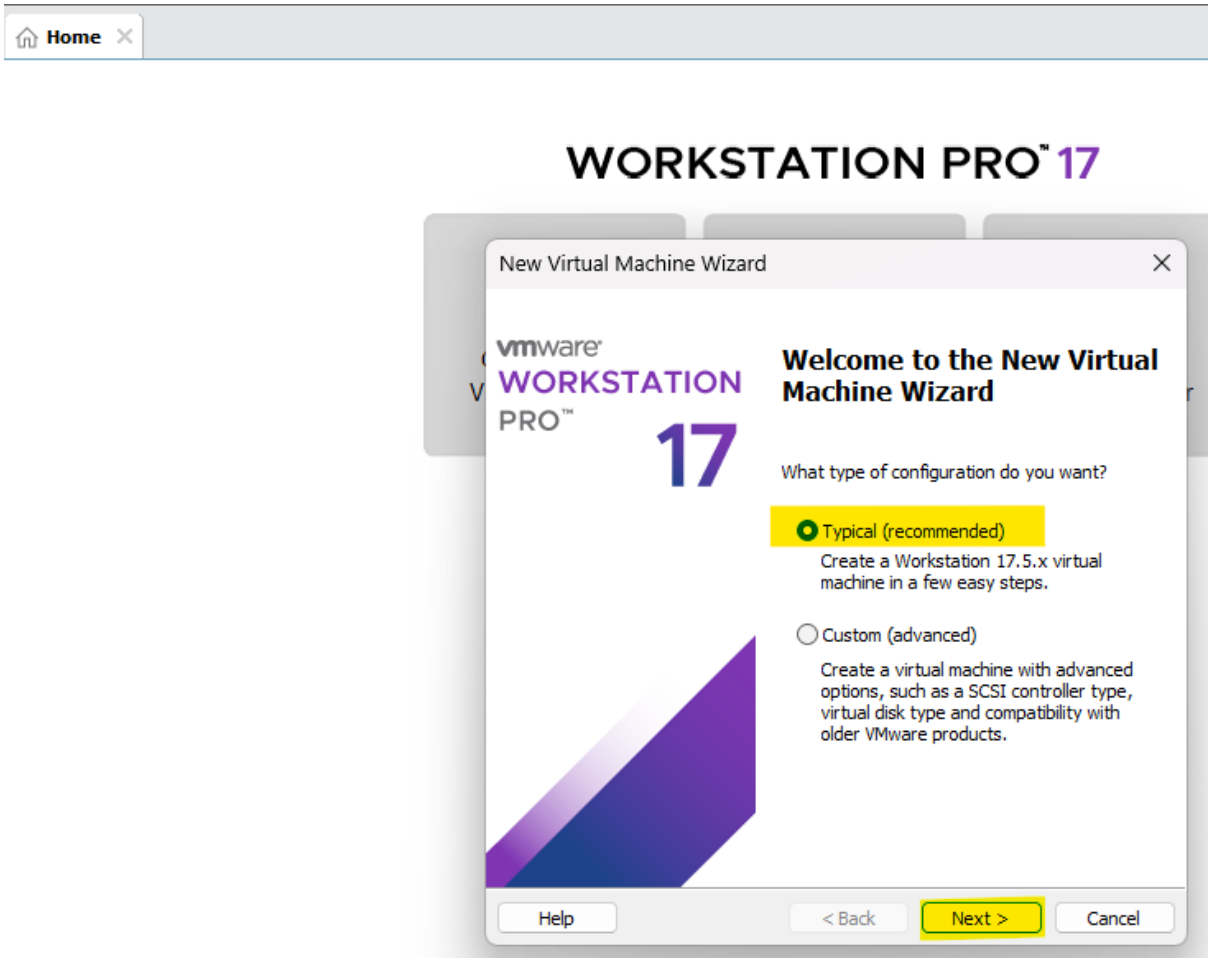
Purpose	Command Examples
View permissions	ls -l filename
Change permissions (symbolic)	chmod u+x file
Change permissions (numeric)	chmod 755 file
Change owner & group	chown user:group file
Change group only	chgrp group file
Set setuid	chmod u+s file
Set setgid	chmod g+s directory
Set sticky bit	chmod +t directory

Creating CentOS 9 configuration on VMWare Workstation

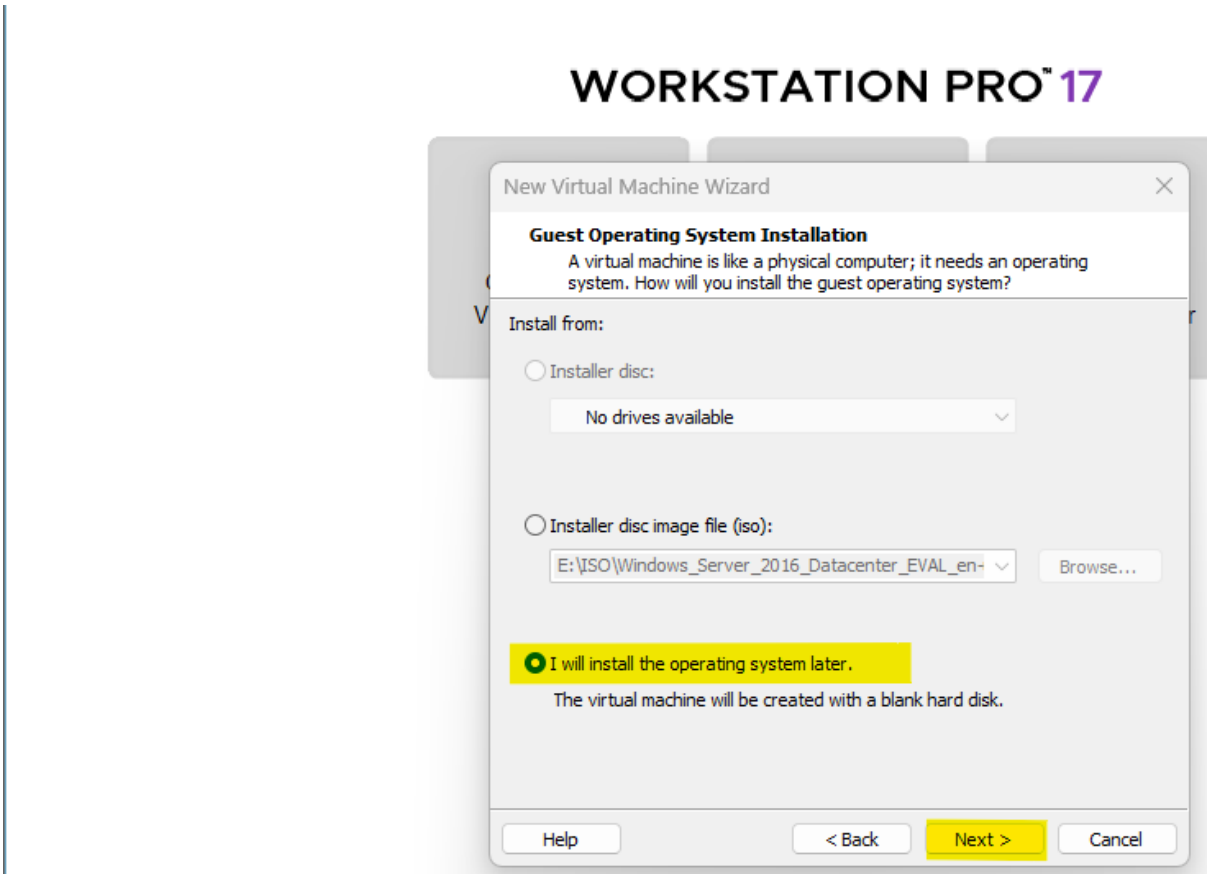
Go to VMWare workstation → File → New Virtual Machine



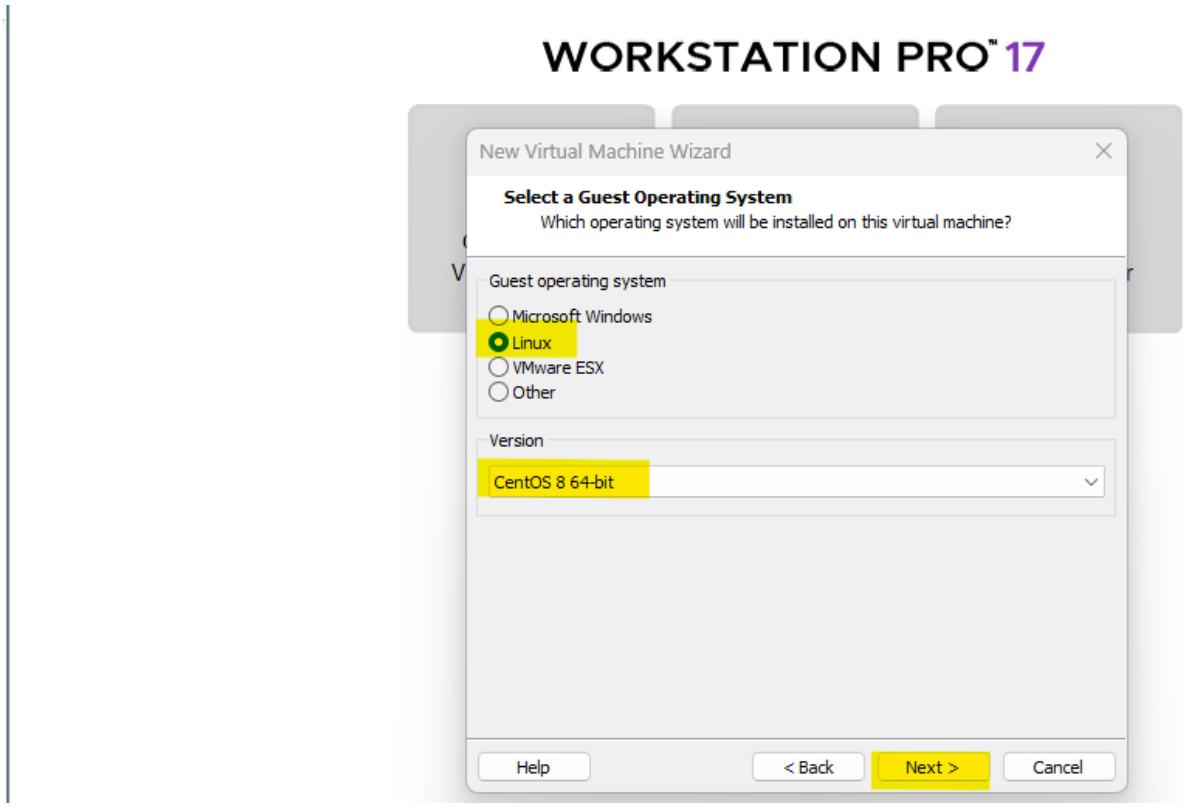
Select typical and click Next.



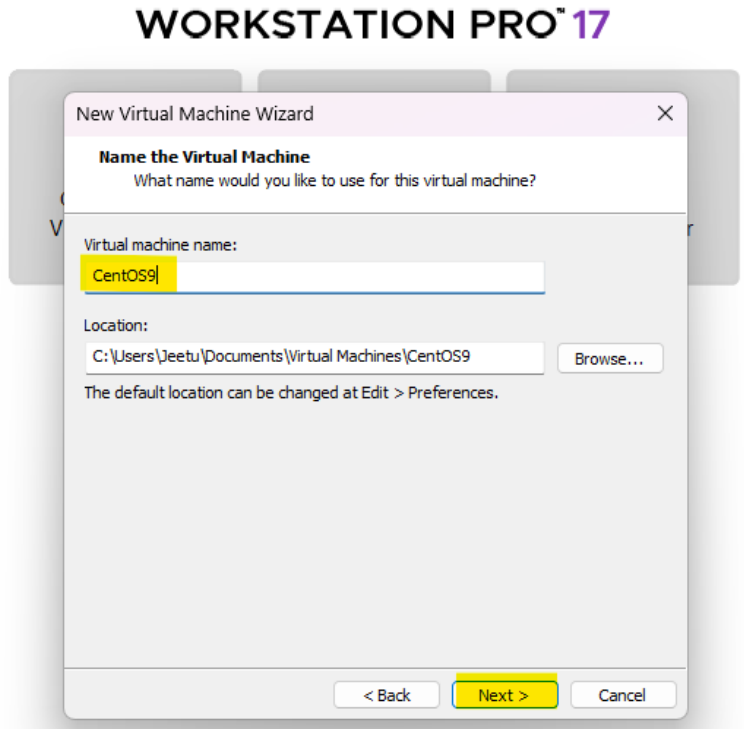
Select “I will install OS later”



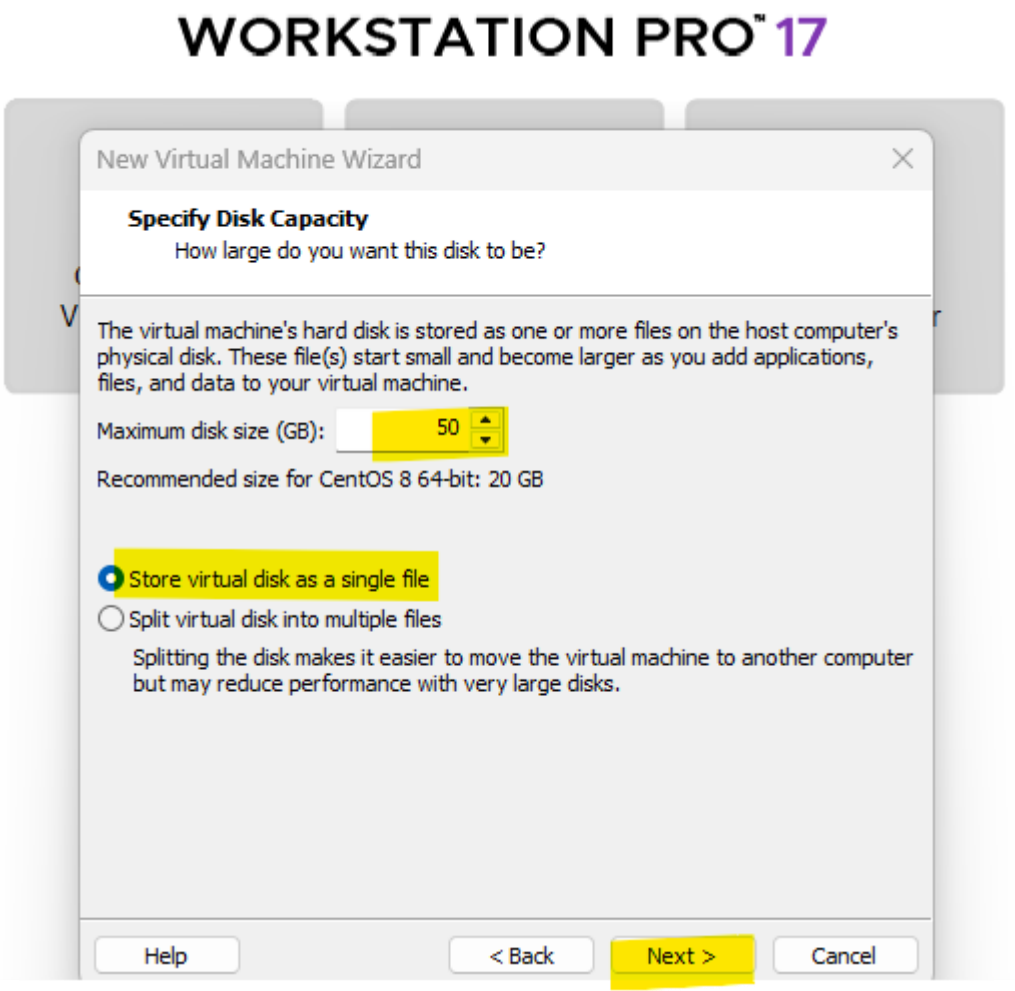
Select the following and click Next



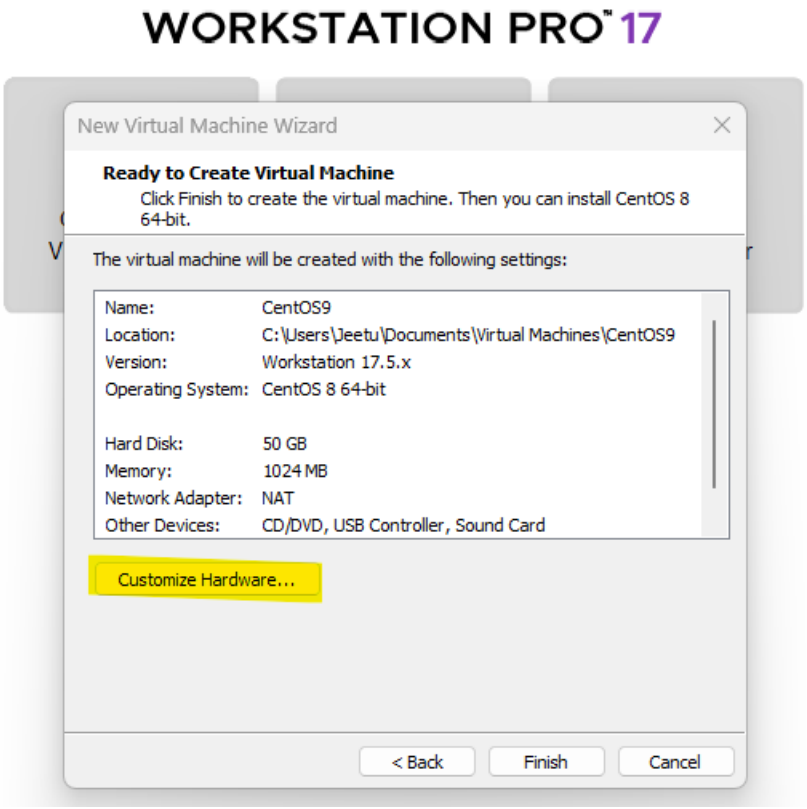
Provide a name to the virtual machine and click Next.



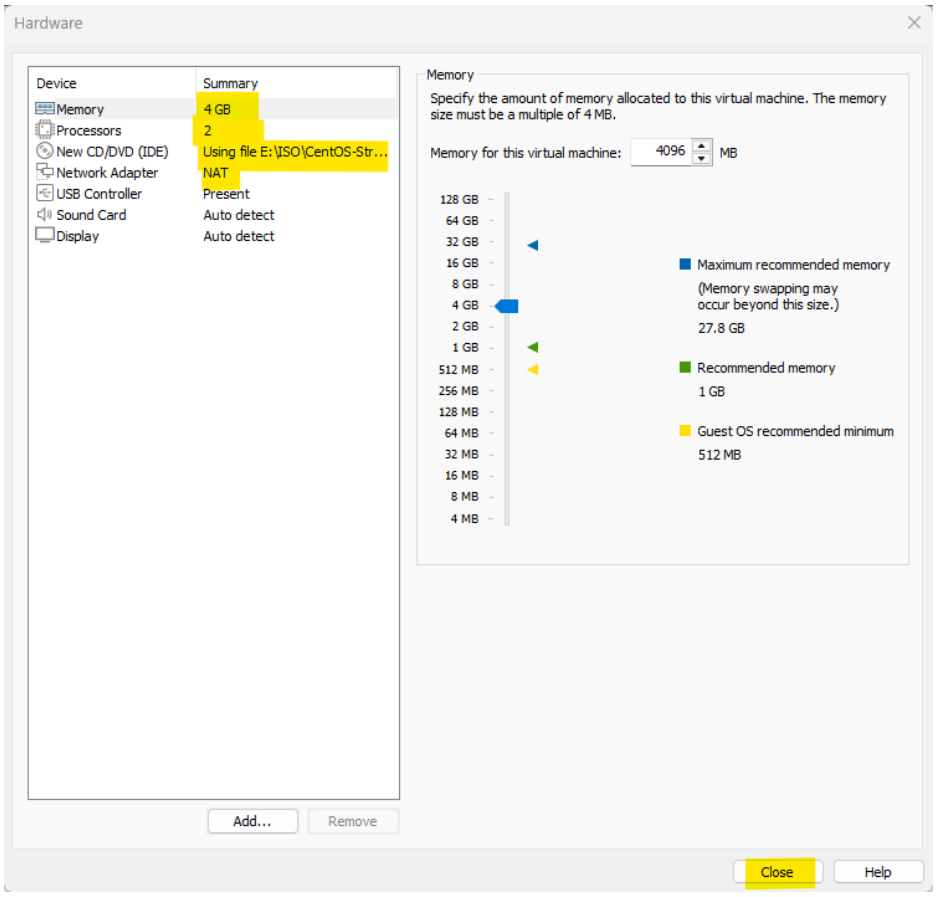
Provide disk size as 50GB and select “Store virtual disk as a single file”.



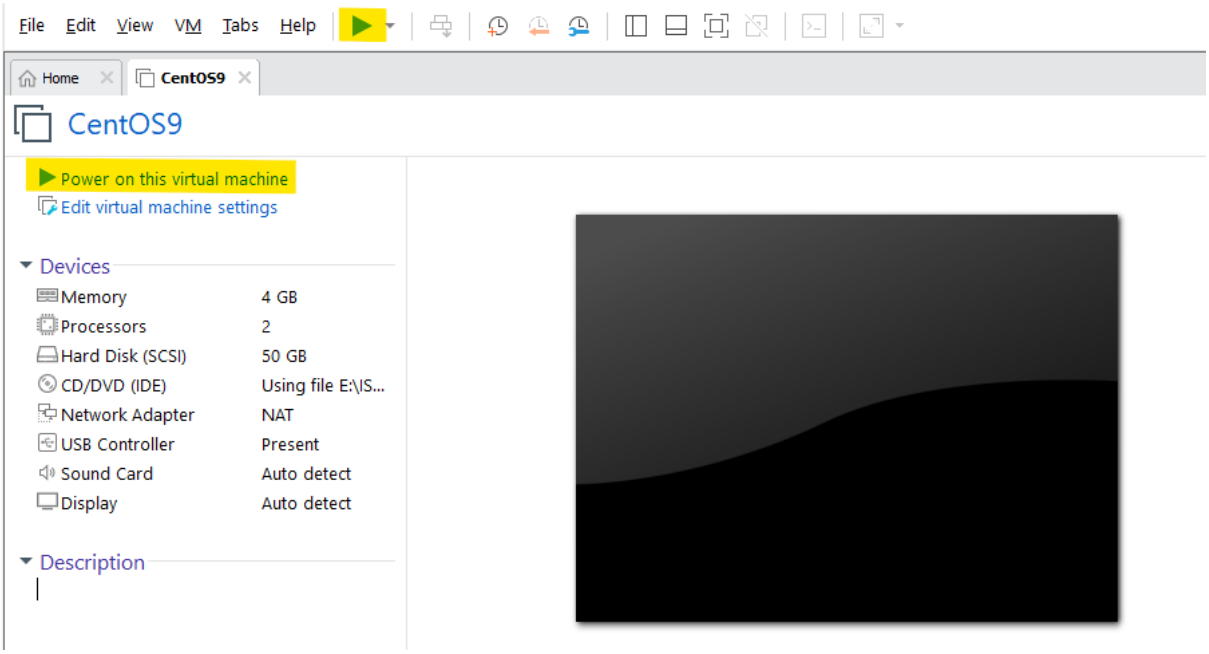
Now, click on “Customize Hardware”



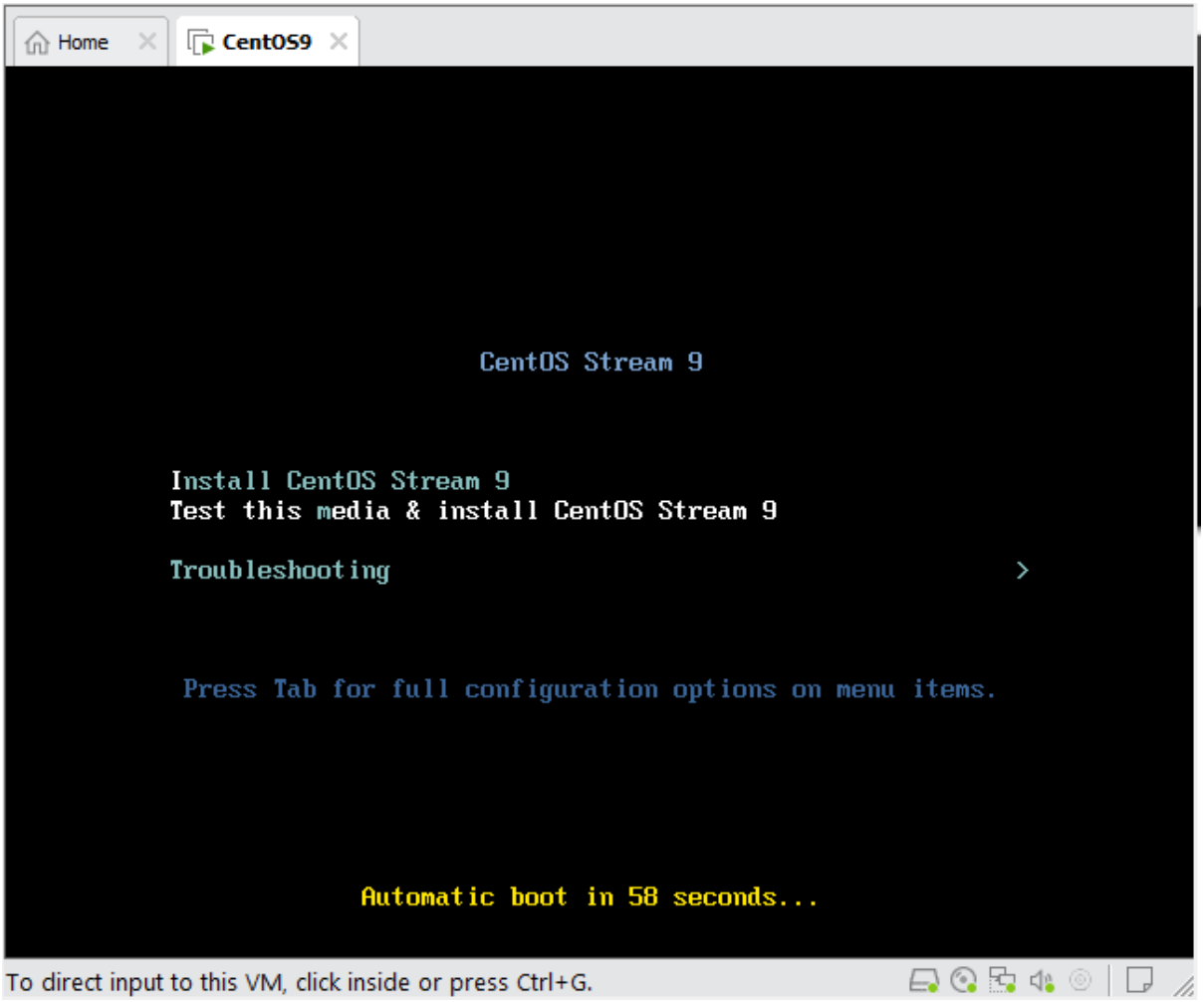
Provide the following and click “close” and then “Finish:



Powering on the virtual machine, use either of the high-lighted options.

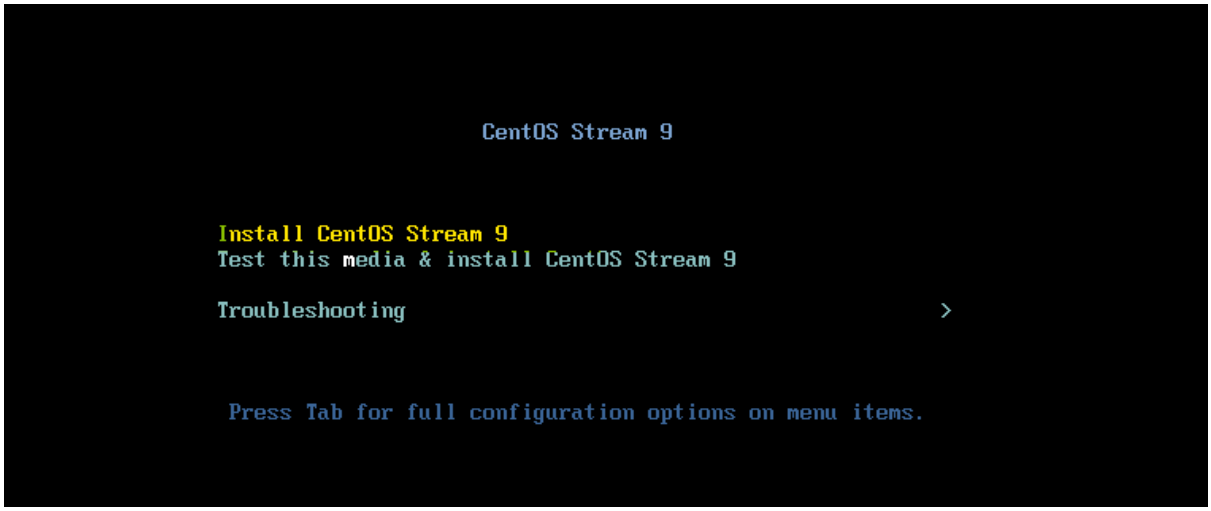


Click inside the VM (in the black area), and press up or down keys from keyboard to break timer.

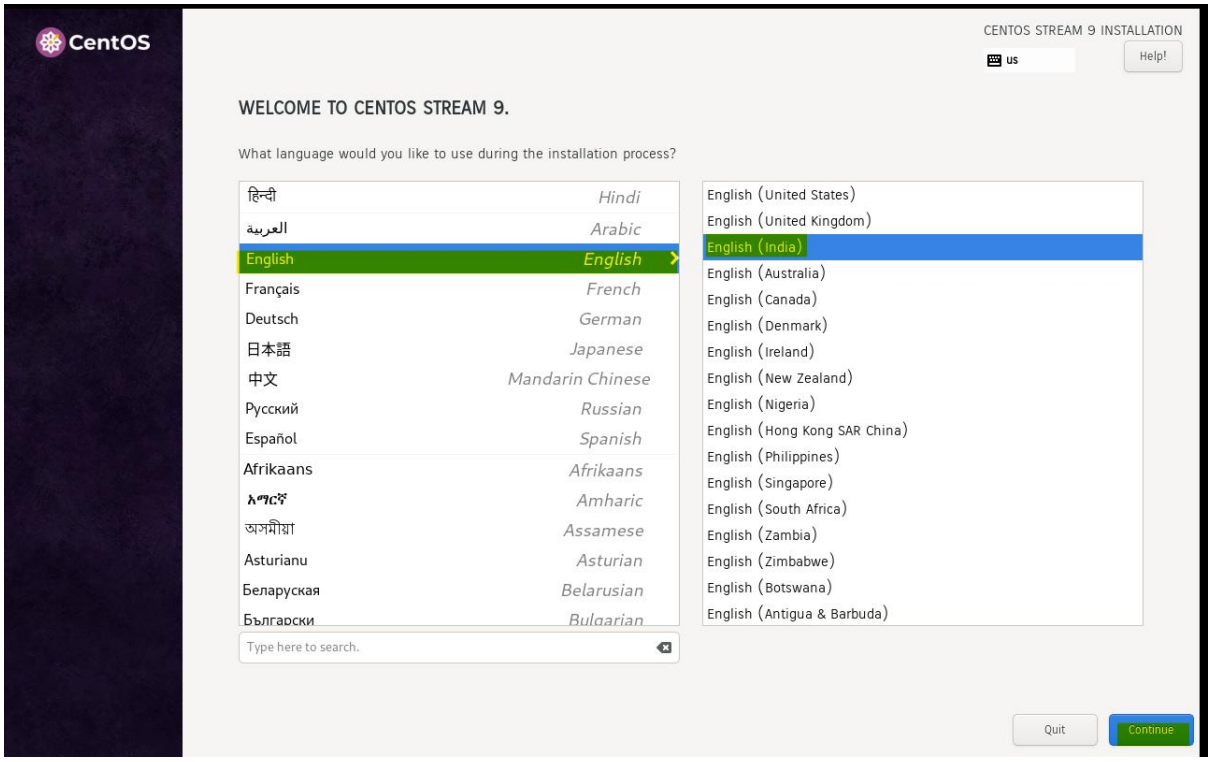


Installing CentOS 9

By selecting “Install CentOS Stream 9” option.

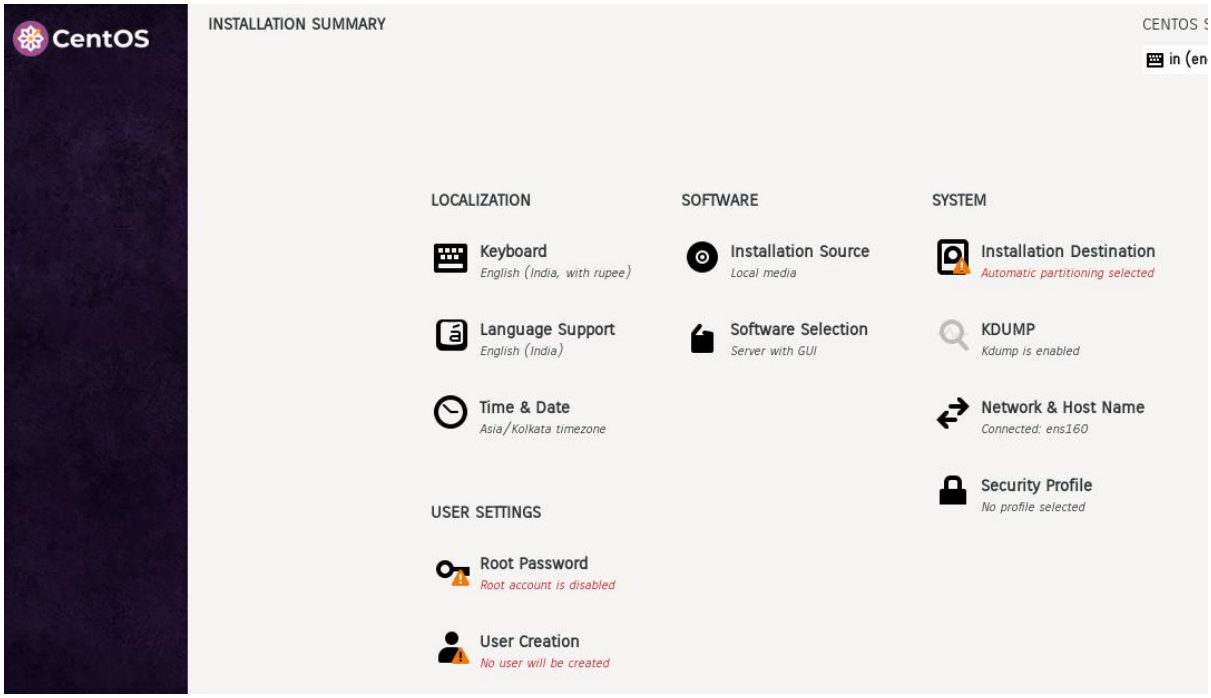


Wait until you get this language selection page:

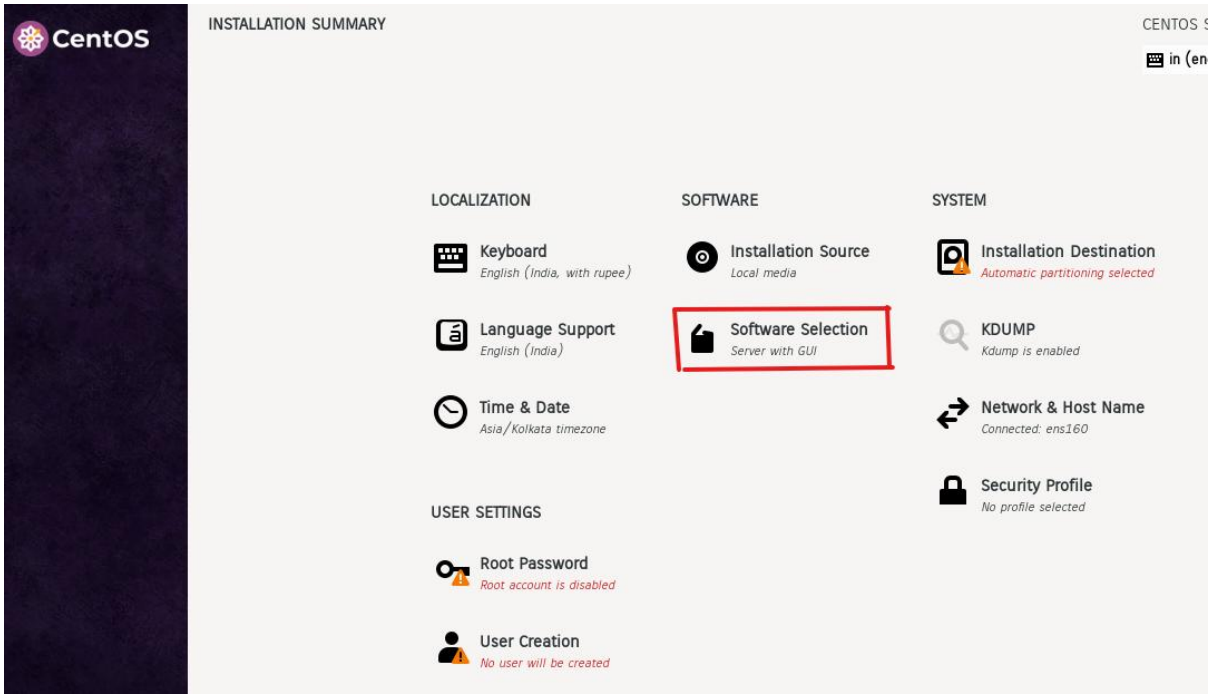


Select “*English* (left side) then *English (India)*”

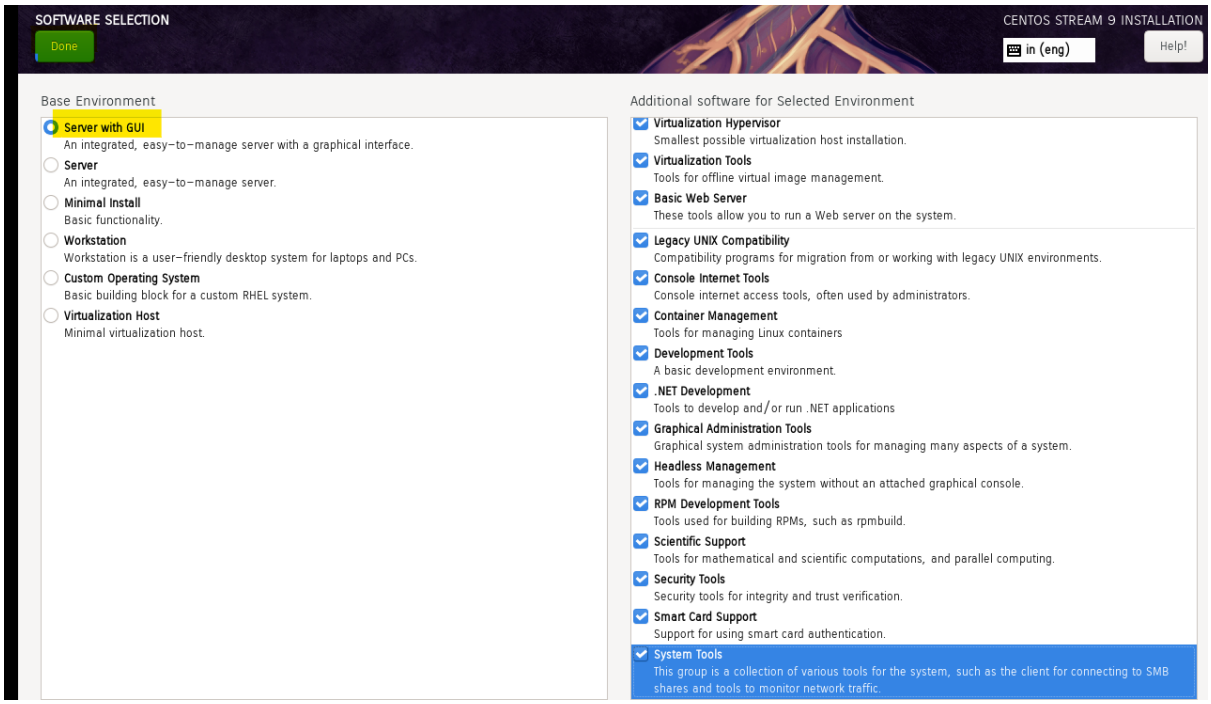
Now fill the following options one by one:



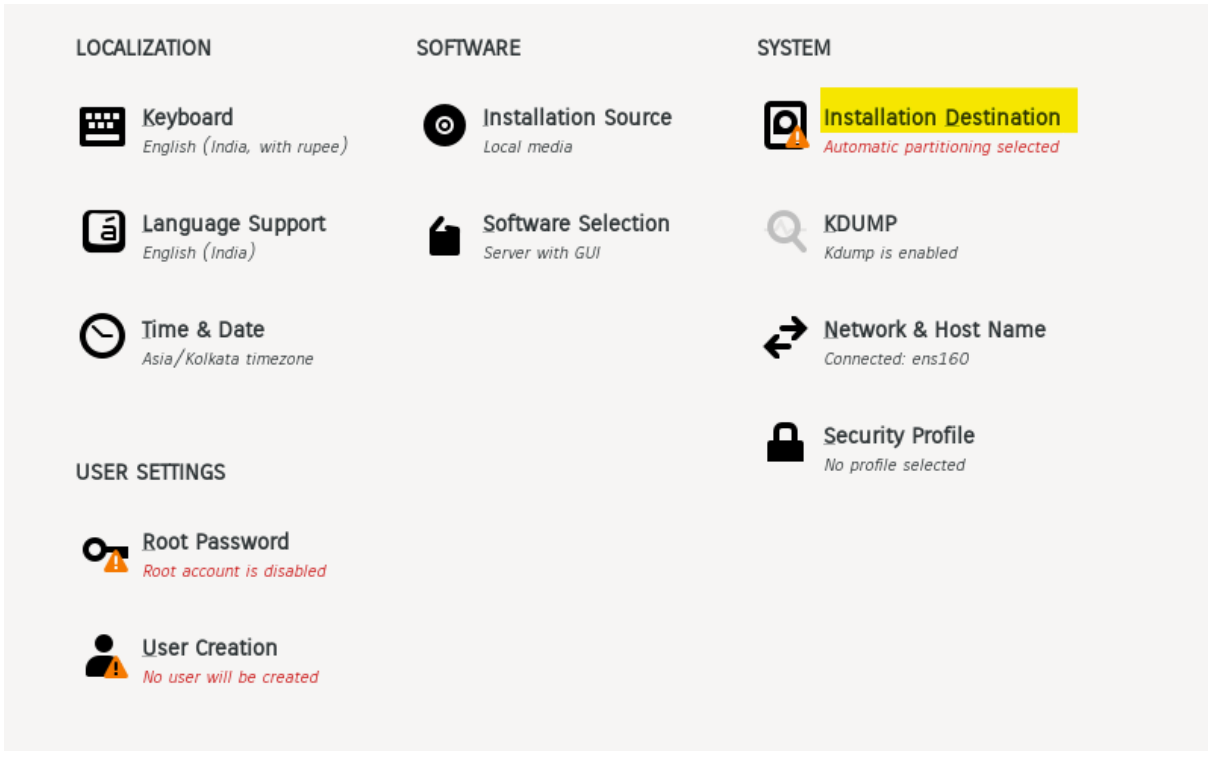
Go to "*Software*" → "*Software Selection*"



Under “Base Environment” select “Server with GUI” and select all additional software (on right-side)



Then click on “Done”. Now go to “System” → “Installation Destination”

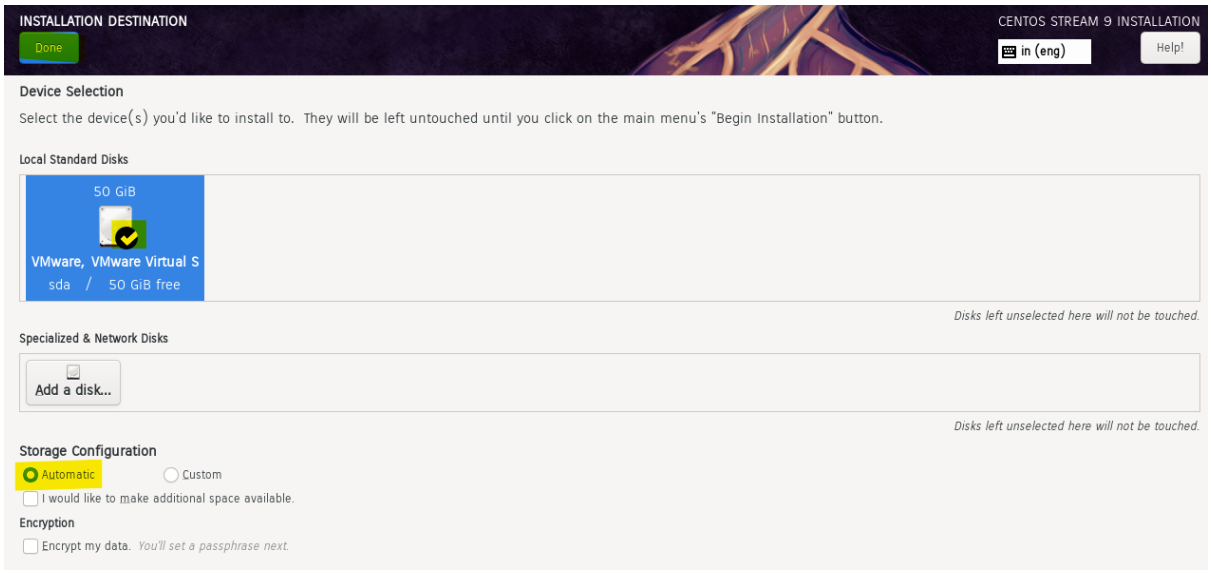


Here, provide the following disk partitions (for custom partitioning)

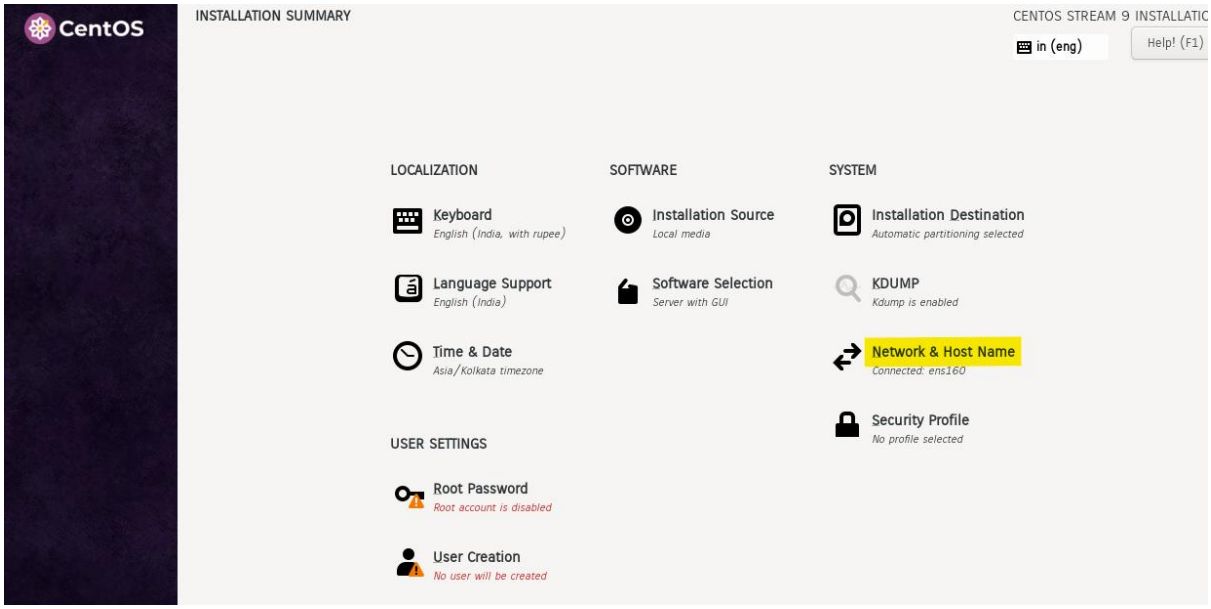
- / = 35GB
- /boot = 500 MB
- Swap = 2xRAM = 2x4 = 8GB

Or you can also select “Automatic partitioning”, I will proceed with “Automatic partitioning”

Select the disk twice (with blue background and black-white check)

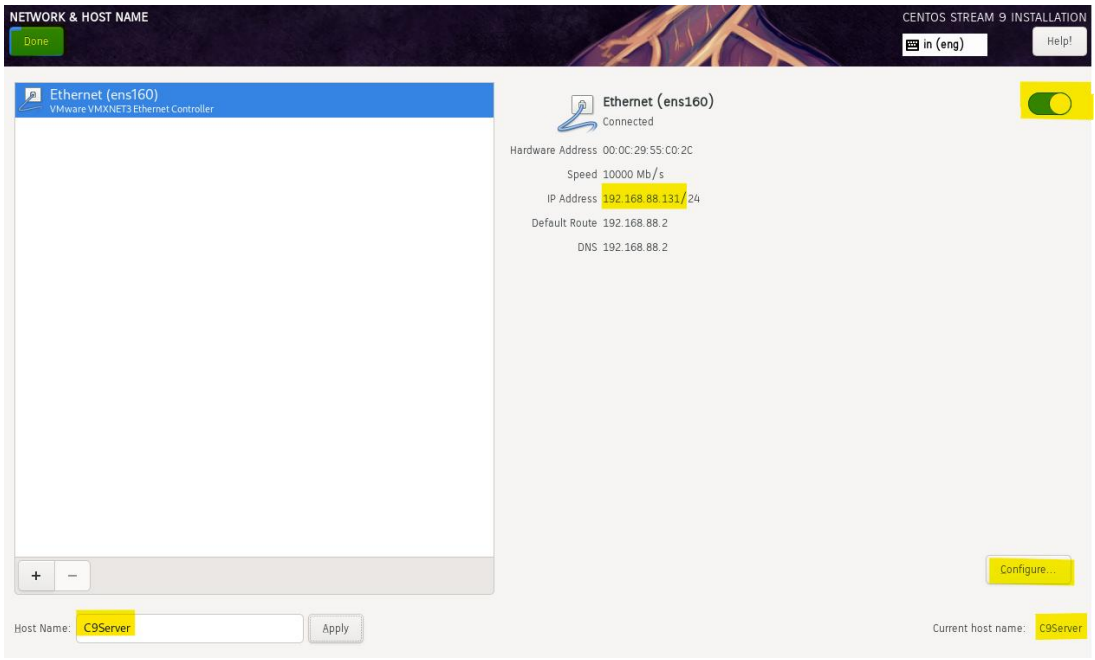


Now go to "System" → "Network & Hostname".

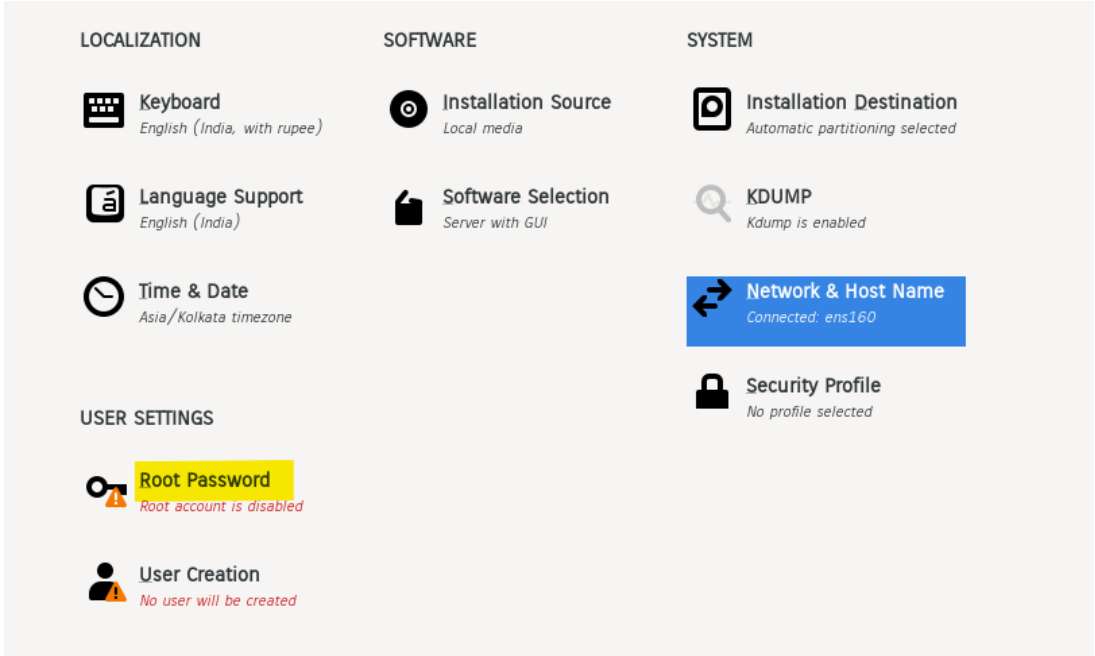


Since my VM is set on NAT, I will be receiving dynamic IP address. In case you need static (manual) IP address, you can click on "configure" and set the IP to "Manual".

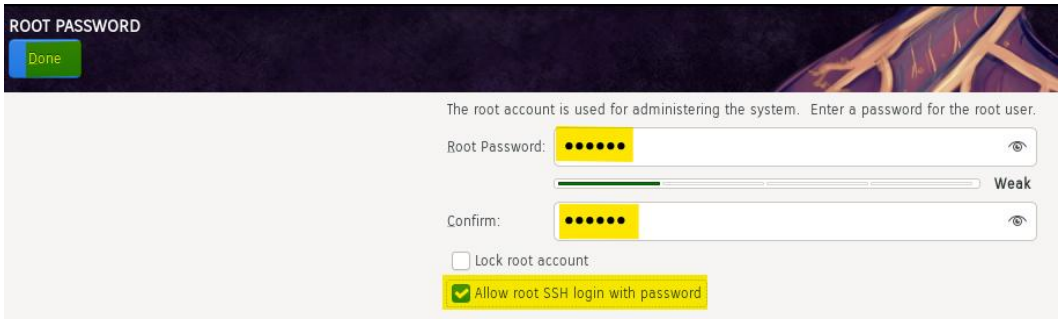
Provide a hostname “C9Server” in my case.



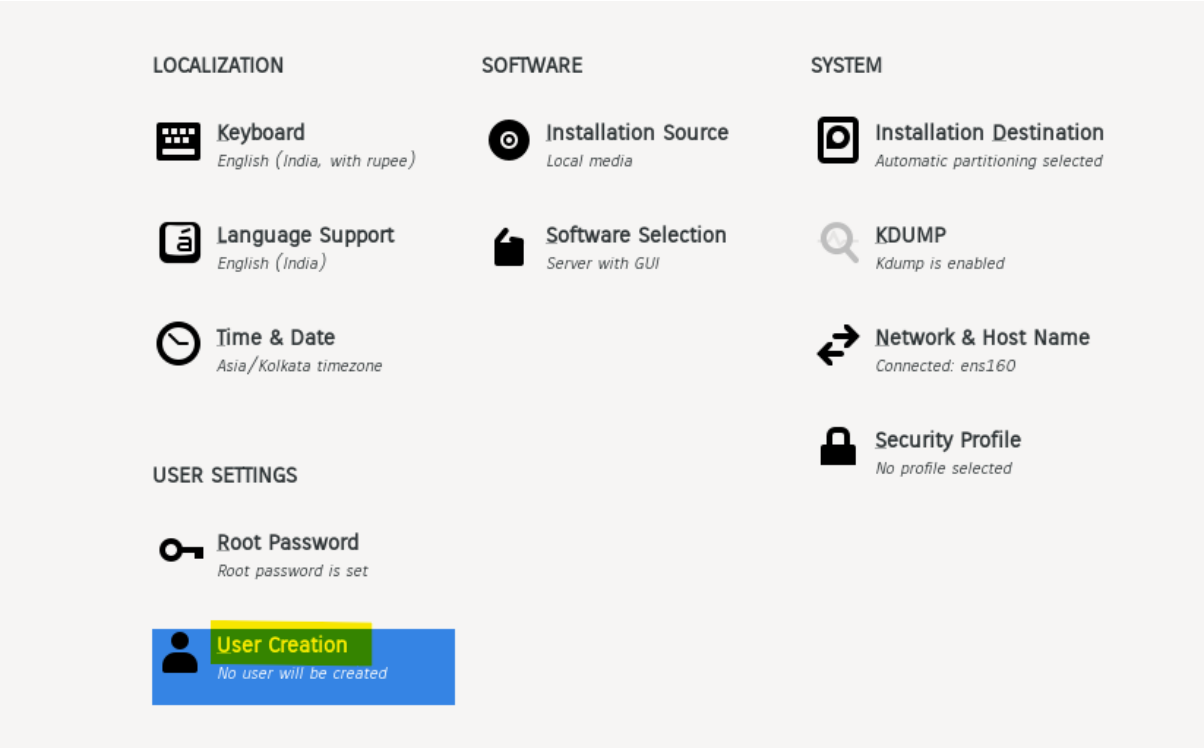
Now go to User settings → Root Password – and set root user password (centos, in my case).



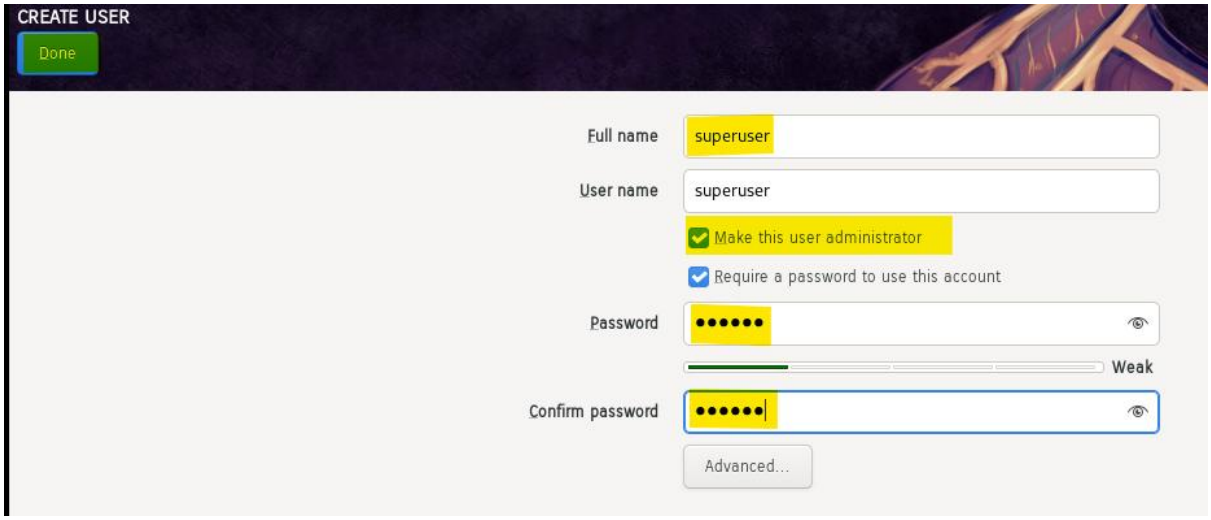
Perform the following and click on “Done” twice.



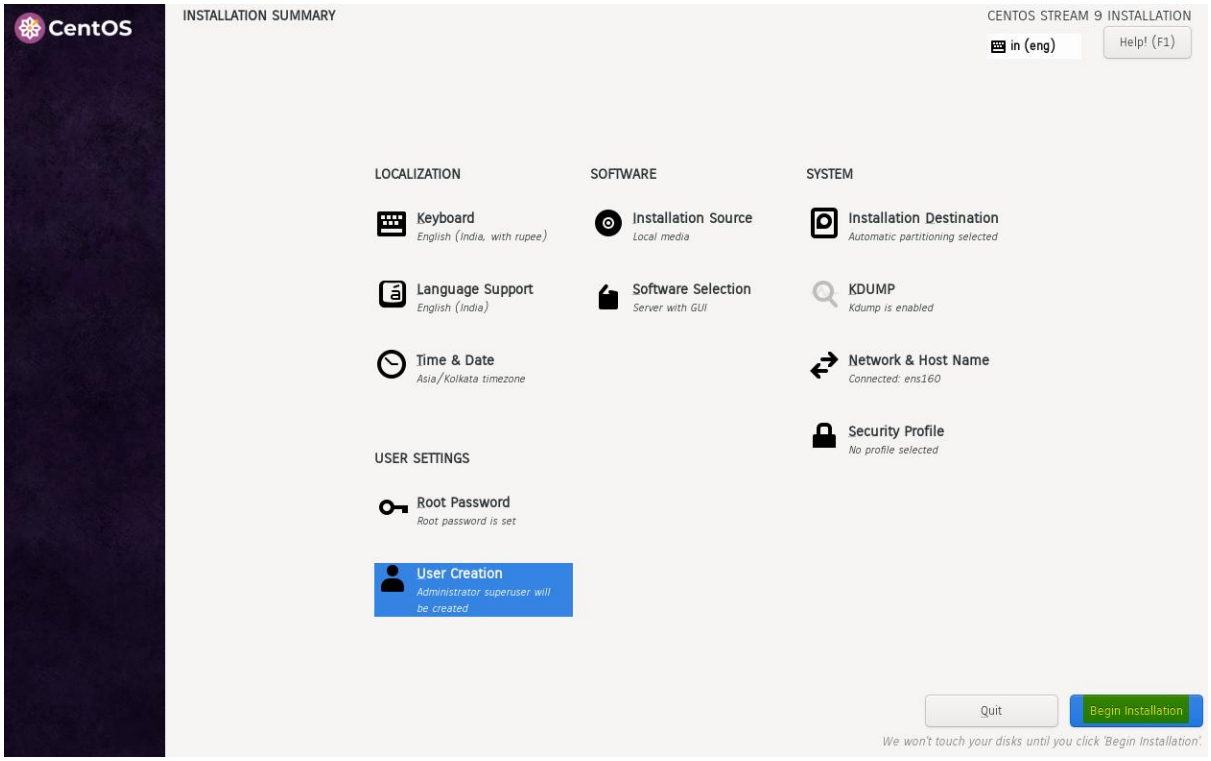
Now go to “User Settings” and click on “user creation” and fill your details.



Provide the details and click on “Done” twice.

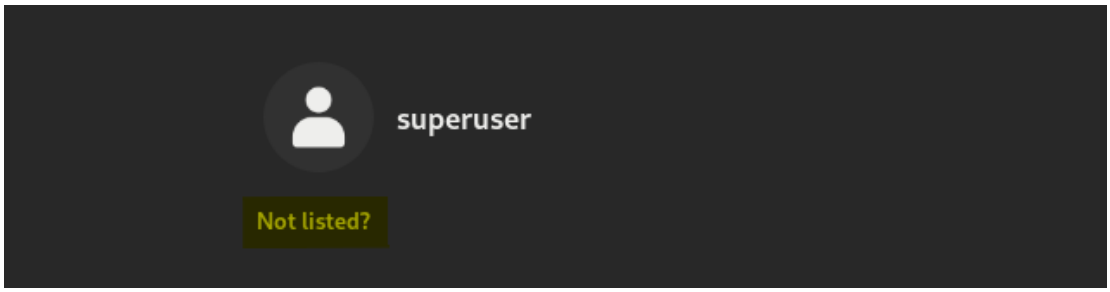


Verify all the warnings/errors are resolved, now click on “Begin Installation” to start installation.

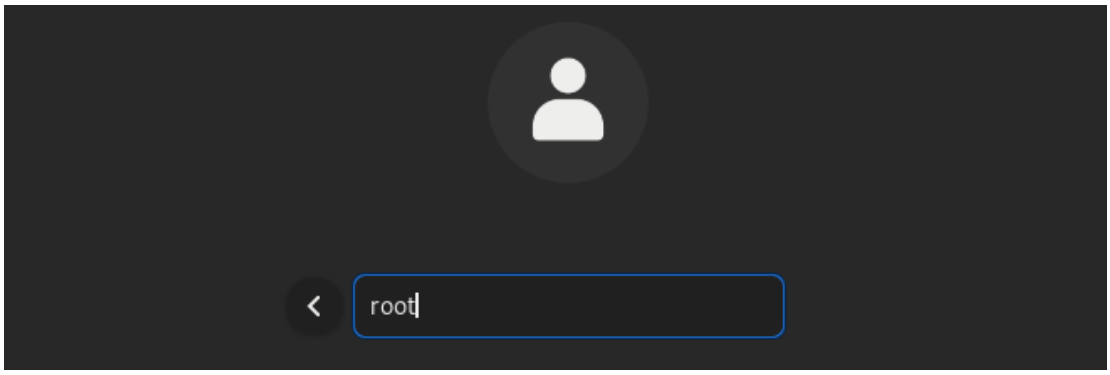


Wait until, you get the “Reboot System” button (this installation will take around 5-10Mins).

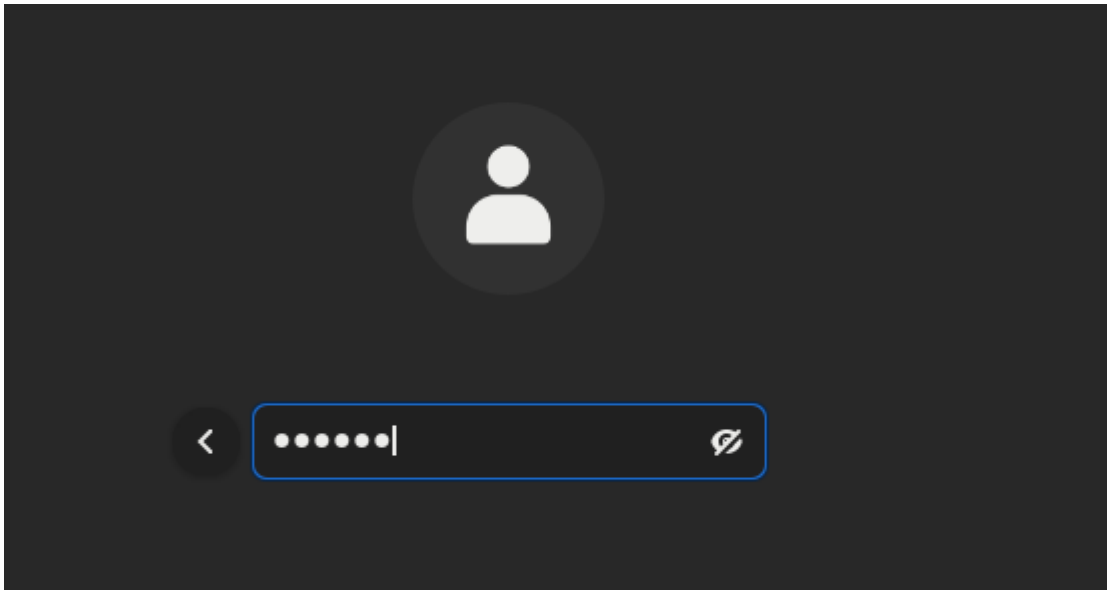
Once rebooted, wait until you get this window. Click on “Not listed?” for root user.



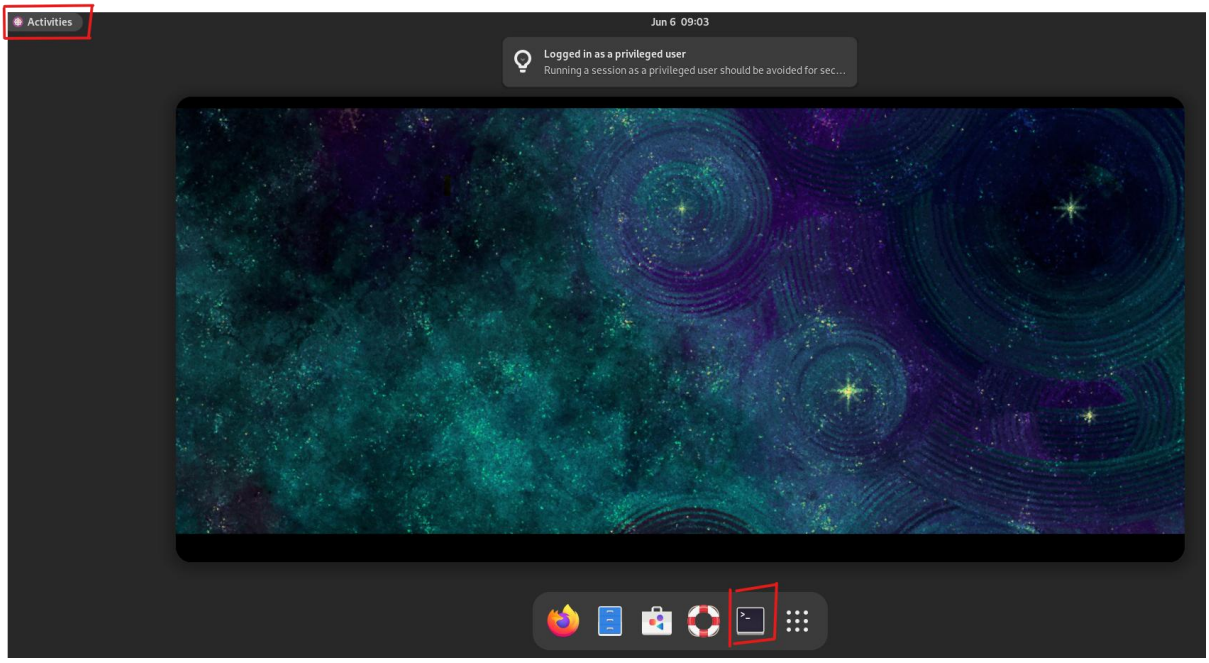
Type username as “root”



Type password and press enter button.



Once you logged-in, you will see this page.



To open terminal click on the  , at the bottom and type following commands.

```
[root@C9Server ~]# hostname
C9Server
[root@C9Server ~]#
[root@C9Server ~]# whoami
root
[root@C9Server ~]#
[root@C9Server ~]# uptime
 09:07:09 up 6 min,  2 users,  load average: 0.20, 0.27, 0.16
[root@C9Server ~]#
[root@C9Server ~]#
```

Hostname → displays the computer/host name.

Whoami → displays the login user name.

Uptime → displays the overall up time of linux machine.

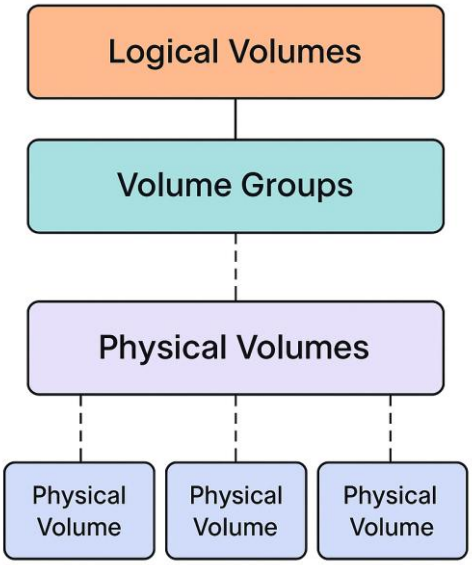
Logical Volume Manager (LVM)

Logical Volume Manager (LVM) is a device mapper framework that provides logical abstraction over physical storage. It allows flexible disk management — you can create, resize, shrink, extend, remove partitions without unmounting or rebooting the system.

Advantages of LVM:

Feature	Description
Dynamic resizing	Increase or shrink volumes without rebooting
Snapshots	Create a copy of volume state at a moment
Flexibility	Add/remove disks on-the-fly
Efficient storage	Create volumes of required sizes anytime
Volume grouping	Combine multiple disks into one storage pool

Key LVM Components



1. Physical Volume (PV)

A raw disk or partition initialized for use by LVM.

- Command: pvcreate

2. Volume Group (VG)

A pool of storage that aggregates multiple PVs.

- Command: vgcreate

3. Logical Volume (LV)

The equivalent of a partition created from the VG space.

- Command: lvcreate

4. Physical Extents (PE)

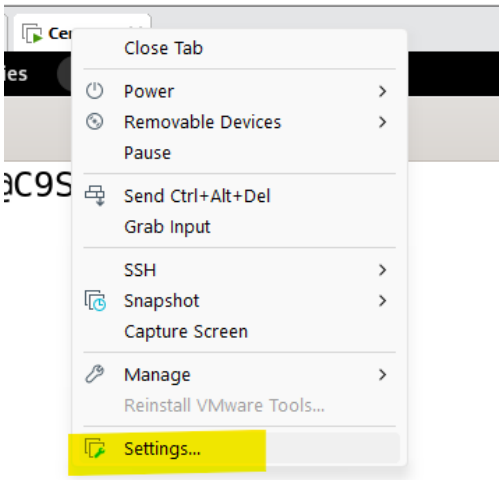
- Fixed-size blocks (default 4MB) allocated on PVs. VGs and LVs are made of PEs.

Steps to create Logical Volume Manager (LVM)

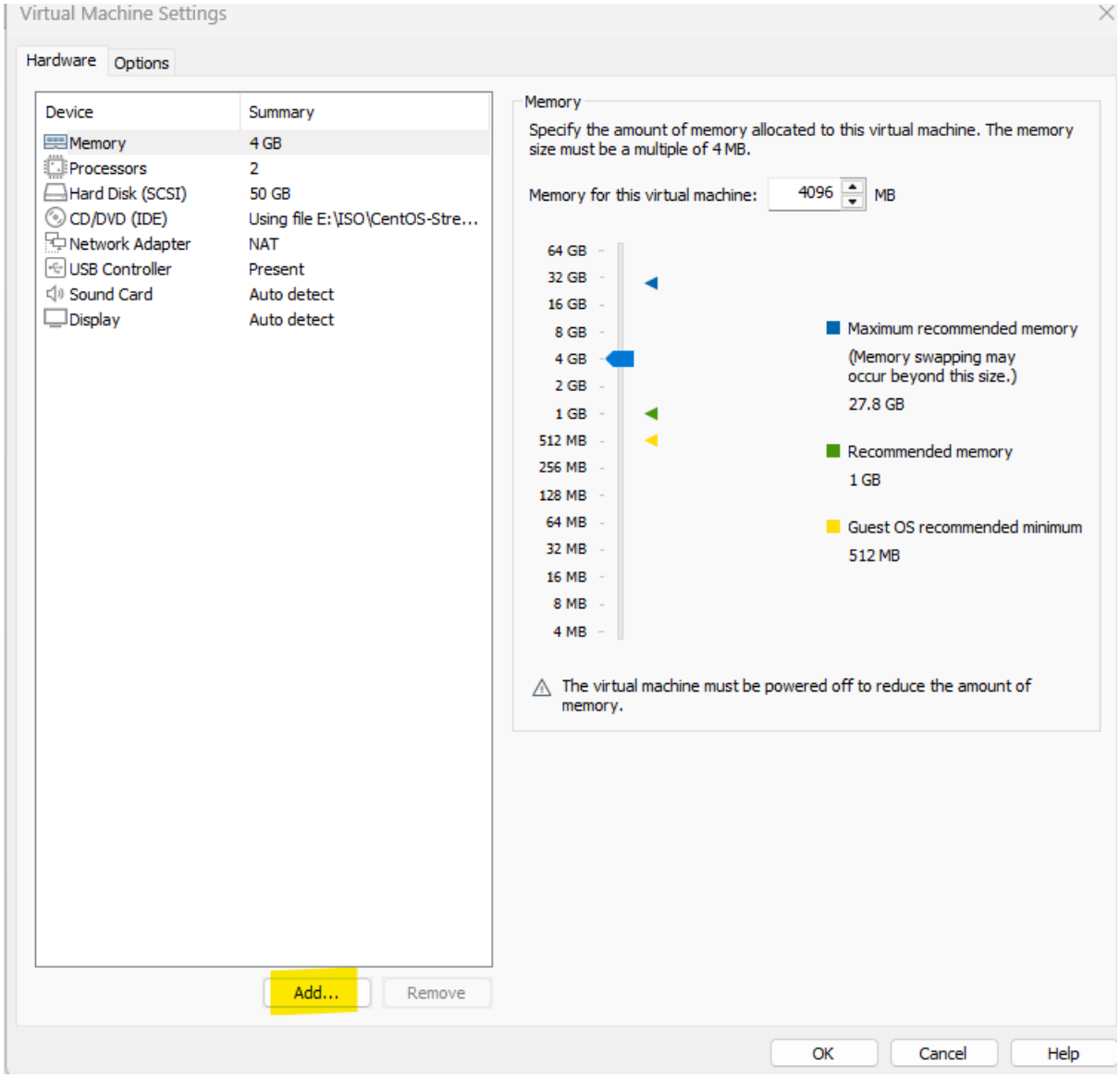
- Add 2-3 disks using VMWare workstation
- Create partitions using fdisk utility
- Create physical volume (PV)
- Create volume group (VG)
- Create logical group (LV)
- Create a file system for this logical volume
- Create a mount point
- Mount this logical volume

Add 2-3 disks using VMWare workstation

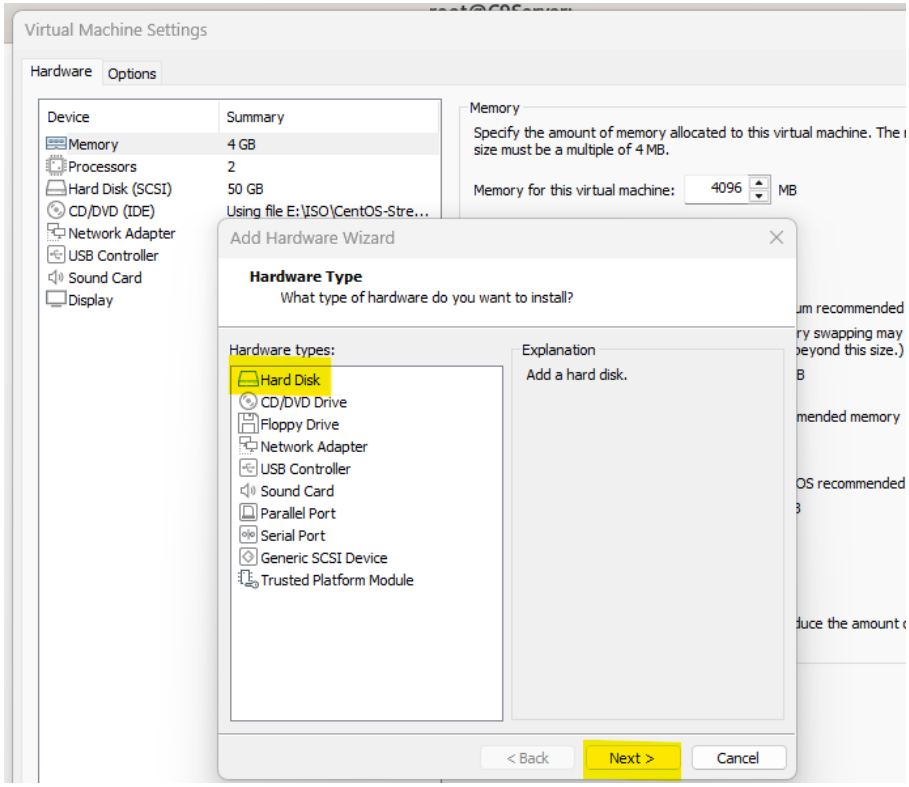
Go to VM settings



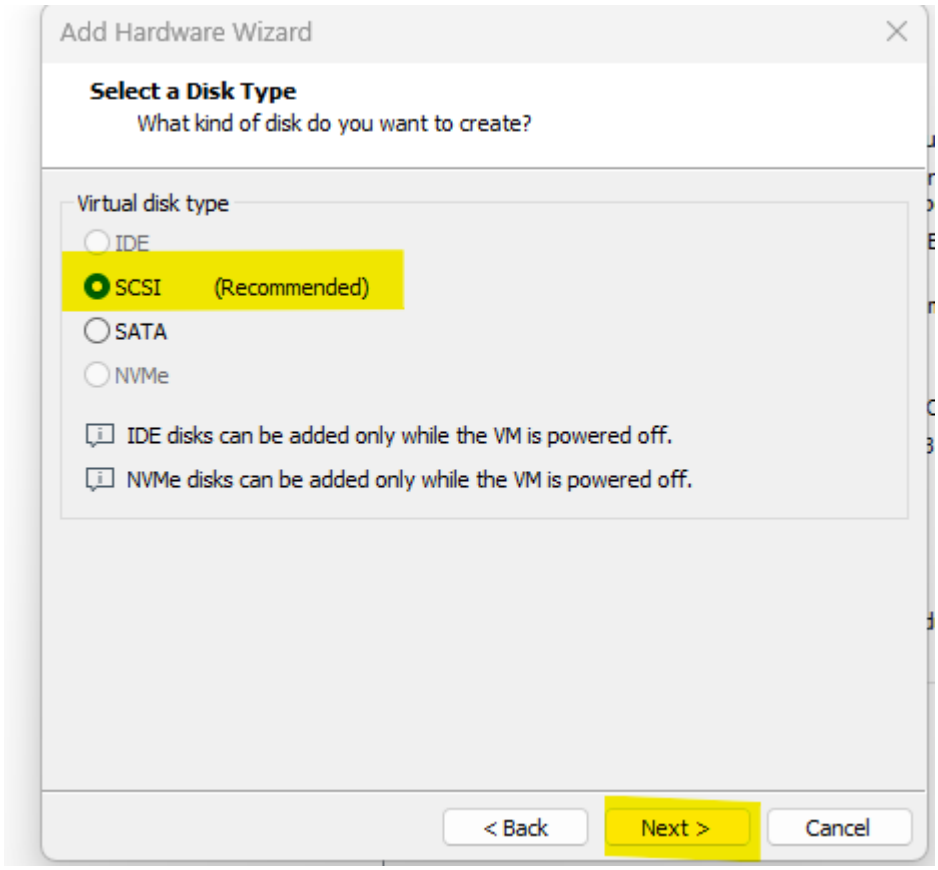
Click on “Add”



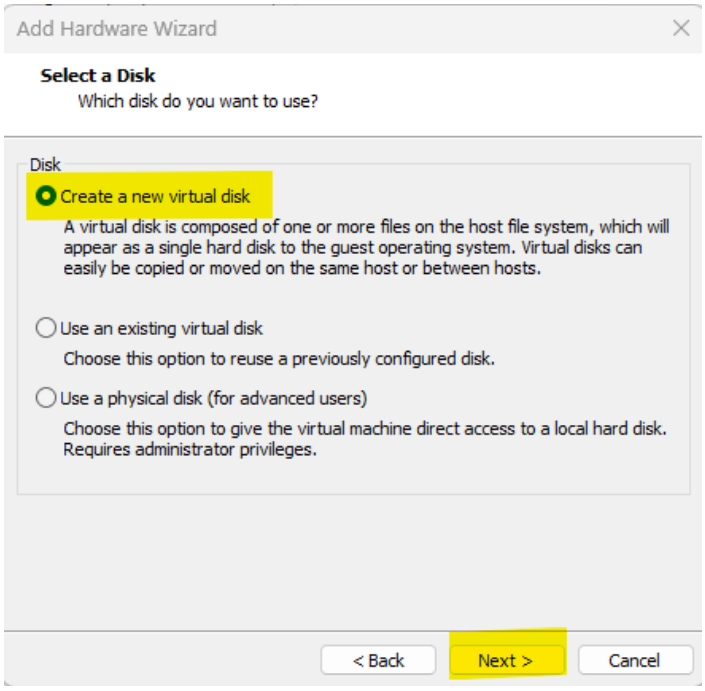
Select “Hard Disk” & click on “Next”



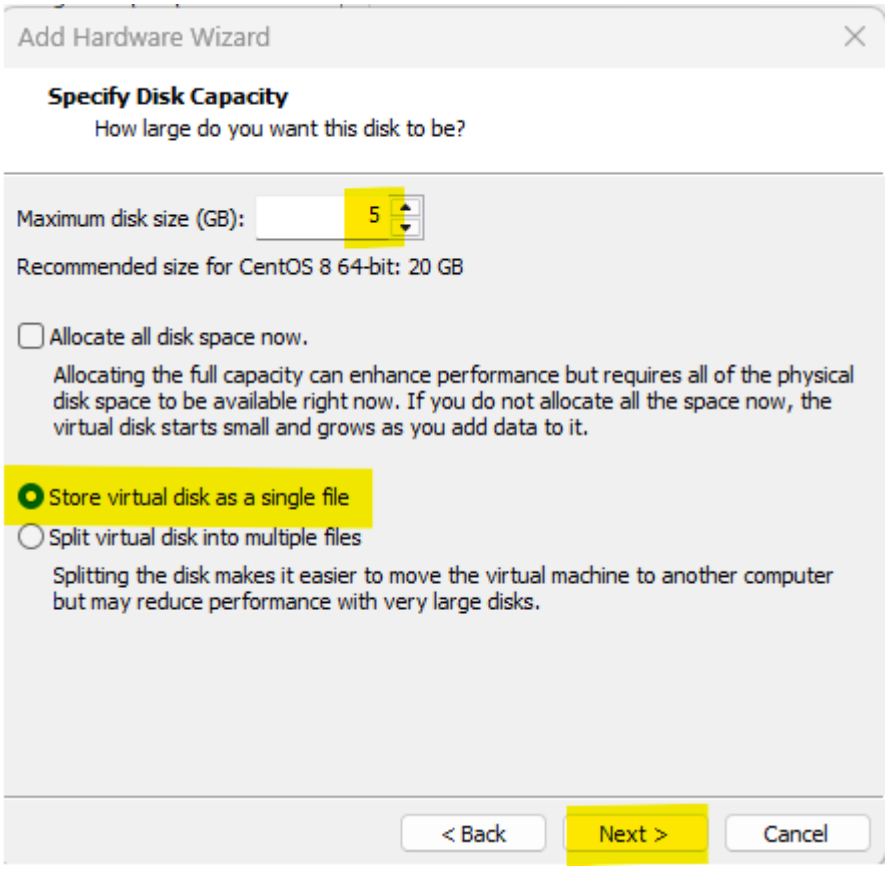
Select SCSI and click Next.



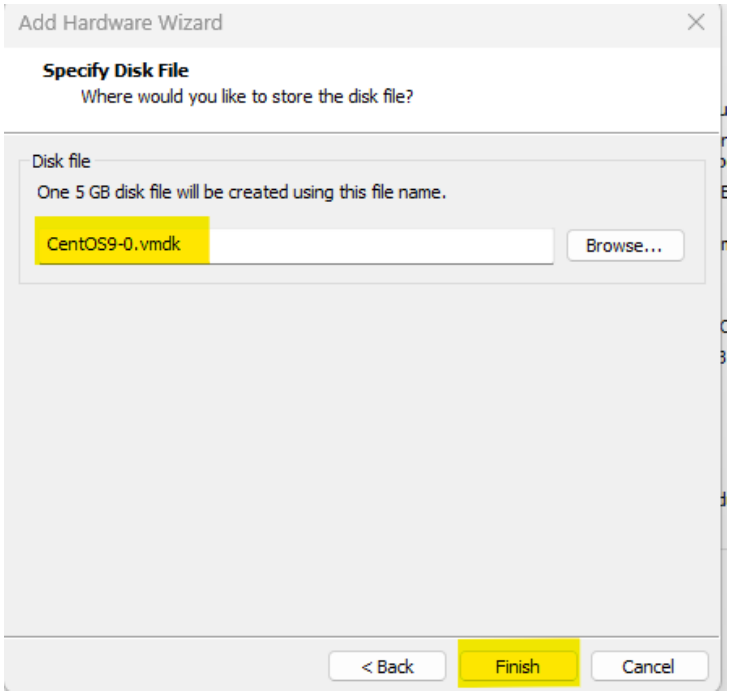
Select “Create a new Virtual Disk”



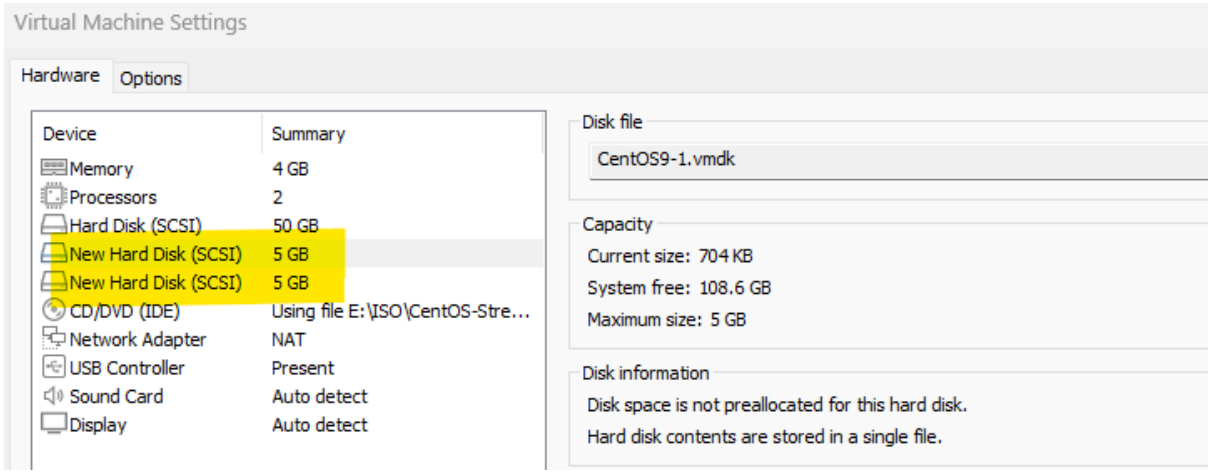
Provide “5GB” and “store virtual disk as a single file”



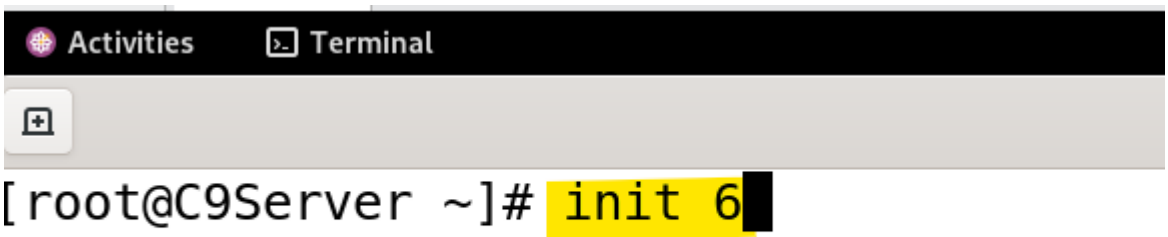
Leave the disk name as default and click on “Finish”.



Similarly, add another 5GB disk.



And then reboot the CentOS machine.



After reboot, login as root user and run command “lsblk”

```
[root@C9Server ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   50G  0 disk
├─sda1        8:1    0    1G  0 part /boot
└─sda2        8:2    0   49G  0 part
  ├─cs-root 253:0    0 45.1G  0 lvm  /
  └─cs-swap 253:1    0  3.9G  0 lvm  [SWAP]
sdb           8:16   0    5G  0 disk
sdc           8:32   0    5G  0 disk
sr0          11:0    1 1024M  0 rom
[root@C9Server ~]#
```

Note - Verify you have sdb and sdc disks (it could be different name in your computer, verify before proceeding further).

Now using /dev/sdb to create a partition.

```
[root@C9Server ~]# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xd3d488f0.

Command (m for help):
```

Press “m” to list all help commands and then “n” to create new partition.

```
Command (m for help): m

Help:

  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag

  Generic
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition

  Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)
```

Provide

“n” → for new partition

“p” → for primary partition

“1” → for 1st partition number

“Press enter” → for 1st sector value (it will take default)

“Press enter” → for last sector value (it will take default)

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-10485759, default 10485759):

Created a new partition 1 of type 'Linux' and of size 5 GiB.
```

Press “w” to save/write data and quit.

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

```
[root@C9Server ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   50G  0 disk
├─sda1        8:1    0    1G  0 part /boot
└─sda2        8:2    0   49G  0 part
  ├─cs-root 253:0    0 45.1G  0 lvm  /
  └─cs-swap 253:1    0  3.9G  0 lvm  [SWAP]
sdb           8:16   0    5G  0 disk
└─sdb1        8:17   0    5G  0 part
sdc           8:32   0    5G  0 disk
sr0          11:0    1 1024M  0 rom
[root@C9Server ~]#
```

Similarly creating another partition on another disk (/dev/sdc)

```
[root@C9Server ~]# fdisk /dev/sdc

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-10485759, default 10485759):

Created a new partition 1 of type 'Linux' and of size 5 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[root@C9Server ~]#
```

Verify

```
[root@C9Server ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   50G  0 disk
├─sda1        8:1    0    1G  0 part /boot
└─sda2        8:2    0   49G  0 part
  ├─cs-root 253:0    0 45.1G  0 lvm  /
  └─cs-swap 253:1    0  3.9G  0 lvm  [SWAP]
sdb           8:16   0    5G  0 disk
└─sdb1        8:17   0    5G  0 part
sdc           8:32   0    5G  0 disk
└─sdc1        8:33   0    5G  0 part
sr0          11:0    1 1024M  0 rom
[root@C9Server ~]#
```

Create physical volume (PV):

Commands to physical volume (PV)

- To create physical volume – pvcreate
- To display physical volume – pvdisplay
- To remove physical volume – pvremove
- To show current physical volumes – pvs

To create physical volume (PV)

```
[root@C9Server ~]# pvs
  PV         VG Fmt  Attr PSize   PFree
  /dev/sda2  cs lvm2 a--  <49.00g    0
[root@C9Server ~]#
[root@C9Server ~]# pvcreate   /dev/sdb1    /dev/sdc1
  Physical volume "/dev/sdb1" successfully created.
  Physical volume "/dev/sdc1" successfully created.
[root@C9Server ~]#
[root@C9Server ~]# pvs
  PV         VG Fmt  Attr PSize   PFree
  /dev/sda2  cs lvm2 a--  <49.00g     0
  /dev/sdb1     lvm2 ---   <5.00g <5.00g
  /dev/sdc1     lvm2 ---   <5.00g <5.00g
[root@C9Server ~]#
```

Create Volume Group (VG):

Commands to volume group (vg)

- To create volume group – vgcreate
- To display volume group – vgdisplay
- To remove volume group – vgremove
- To show current volume group – vgs

To create volume group (VG)

```
[root@C9Server ~]# vgs
  VG #PV #LV #SN Attr   VSize   VFree
  cs   1   2   0 wz--n- <49.00g    0
[root@C9Server ~]#
[root@C9Server ~]# vgcreate  vg   /dev/sdb1    /dev/sdc1
  Volume group "vg" successfully created
[root@C9Server ~]#
[root@C9Server ~]# vgs
  VG #PV #LV #SN Attr   VSize   VFree
  cs   1   2   0 wz--n- <49.00g    0
  vg   2   0   0 wz--n-   9.99g 9.99g
[root@C9Server ~]#
[root@C9Server ~]#
```

Here,

- “vg” = name given to the volume group (it can be any name)

Create Logical Volume (LV):

Commands to logical volume (lv)

- To create logical volume – lvcreate
- To display logical volume – lvdisplay
- To remove logical volume – lvremove
- To show current logical volume – lvs

To create logical volume (LV)

```
[root@C9Server ~]#
[root@C9Server ~]# lvs
  LV   VG Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root cs -wi-ao---- 45.08g
  swap cs -wi-ao----  3.91g
[root@C9Server ~]#
[root@C9Server ~]# lvcreate   -n  lv  -l  100%FREE  vg
  Logical volume "lv" created.
[root@C9Server ~]#
[root@C9Server ~]# lvs
  LV   VG Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root cs -wi-ao---- 45.08g
  swap cs -wi-ao----  3.91g
  lv   vg -wi-a-----  9.99g
[root@C9Server ~]#
```

Here

- "-n" → name of the logical volume
- "-l" → length of the logical volume
- "100%FREE" → size that must be taken from volume group
- "vg" → name of the volume group from which size will be consumed.

To check the actual name (Value of LV path) of the LV:

```
[root@C9Server ~]# lvdisplay /dev/vg/lv
  --- Logical volume ---
  LV Path                /dev/vg/lv
  LV Name                lv
  VG Name                vg
  LV UUID                jO53Qe-9bx6-FYct-cMNk-G4PQ-0Kba-ecaxrq
  LV Write Access        read/write
  LV Creation host, time C9Server, 2025-06-06 10:15:08 +0530
  LV Status              available
  # open                 0
  LV Size                9.99 GiB
  Current LE             2558
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:2

[root@C9Server ~]#
```

The name "**/dev/vg/lv**" will be now used for any operation on logical volume.

Create a file system for this logical volume:

```
[root@C9Server ~]# mkfs.
mkfs.cramfs  mkfs.ext2    mkfs.ext3    mkfs.ext4    mkfs.fat     mkfs.minix   mkfs.msdos   mkfs.vfat    mkfs.xfs
[root@C9Server ~]# mkfs.xfs   /dev/vg/lv
meta-data=/dev/vg/lv             isize=512    agcount=4, agsize=654848 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1 nrext64=0
data     =                       bsize=4096   blocks=2619392, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@C9Server ~]#
[root@C9Server ~]#
```

Command: "**mkfs.xfs /dev/vg/lv**"

Create a mount point

```
[root@C9Server ~]# mkdir   /lvm
[root@C9Server ~]#
[root@C9Server ~]# ls -ld /lvm
drwxr-xr-x. 2 root root 6 Jun  6 10:21 /lvm
[root@C9Server ~]#
[root@C9Server ~]#
```

Mount this logical volume on to /lvm

```
[root@C9Server ~]# mount   /dev/vg/lv    /lvm
[root@C9Server ~]#
[root@C9Server ~]# df -hT
Filesystem           Type      Size  Used Avail Use% Mounted on
devtmpfs             devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs                tmpfs     1.8G     0  1.8G   0% /dev/shm
tmpfs                tmpfs     725M   12M  714M   2% /run
/dev/mapper/cs-root  xfs        46G  9.8G   36G  22% /
/dev/sda1            xfs       960M  373M  588M  39% /boot
tmpfs                tmpfs     363M   96K  363M   1% /run/user/0
/dev/mapper/vg-lv    xfs        10G  104M  9.9G   2% /lvm
[root@C9Server ~]#
```

Here,

- "mount" – helps in mounting the partition
- "/dev/vg/lv" – actual LV path
- "/lvm" – directory where LVM will be mounted.
- "df -hT" – displays all the mounted partitions.

Note – That's all for LVM

What is a Package Manager?

A package manager is a collection of tools that automates:

- Installing
- Upgrading
- Configuring
- Removing software packages

It handles dependencies, verifies integrity, and accesses software from repositories.

Why Use a Package Manager?

Feature	Description
Automation	Installs and configures packages automatically
Dependency Management	Resolves and installs required libraries
Security	Downloads from trusted, signed repositories
Updates	Easily apply system-wide updates
Consistency	Maintains a database of all installed packages

Types of Linux Package Managers

1. DEB-based (Debian, Ubuntu)

Tool	**Description**
dpkg	Low-level tool to manage .deb files
apt (Advanced Package Tool)	Frontend for dpkg, resolves dependencies
apt-get, apt-cache	Older command-line tools for apt

2. RPM-based (RHEL, CentOS, Fedora)

Tool	**Description**
rpm	Low-level tool to manage .rpm files
yum	Legacy dependency manager for RHEL/CentOS
dnf	Modern replacement for yum in Fedora/RHEL 8+

3. Other Distributions

Tool	Used in	Description
zypper	openSUSE	Manages RPM packages
pacman	Arch Linux	Lightweight and powerful
emerge	Gentoo	Source-based package manager

APT (Debian/Ubuntu Systems)

Basic APT Commands

Task	Command
Update package index	sudo apt update
Upgrade installed packages	sudo apt upgrade
Install a package	sudo apt install nginx
Remove a package	sudo apt remove nginx
Search a package	apt search apache2
Get package info	apt show nginx

RPM, YUM, DNF (RHEL/CentOS/Fedora)

Basic RPM Commands

Task	Command
Install a .rpm file	sudo rpm -ivh package.rpm
Remove a package	sudo rpm -e nginx
List installed packages	rpm -qa

YUM (RHEL 7 and below)

Task	Command
Install package	sudo yum install httpd
Remove package	sudo yum remove httpd
Update system	sudo yum update
Search package	yum search git

DNF (RHEL 8+, Fedora)

Task	Command
Install	sudo dnf install httpd
Update	sudo dnf update
Remove	sudo dnf remove httpd
Search	dnf search nginx

GUI Frontends for Package Managers

Tool	Description
Synaptic	GUI frontend for APT
GNOME Software	App store for Fedora, Ubuntu
YaST	Package manager for openSUSE

YUM (Yellowdog Updater, Modified)

What is YUM?

YUM is a high-level package manager for RPM-based distributions that automatically handles package installation, dependency resolution, updates, removal, and repository management.

Features of YUM

Feature	Description
Dependency resolution	Automatically installs/removes package dependencies
Repository support	Downloads packages from online or local repos
Group installation	Allows installing groups of related packages
Rollback support	Limited, but allows viewing transaction history
Command-line interface	Simple syntax, powerful options

YUM Commands

Update Repositories

sudo yum check-update

Install a Package

sudo yum install httpd

Remove a Package

sudo yum remove httpd

Update All Packages

yum search nginx

Get Info About a Package

yum info nginx

Install Package from a .rpm File Using YUM

sudo yum localinstall somefile.rpm

List Installed Packages

yum list installed

Clean Cache

sudo yum clean all

View YUM Transaction History

yum history

yum history list

DNF (Dandified YUM)

What is DNF?

DNF (Dandified YUM) is the next-generation RPM package manager for Red Hat-based systems. It replaces YUM as the default tool for managing packages starting from RHEL 8 and Fedora 22.

It was introduced to address performance and dependency resolution issues in YUM.

Key Features of DNF

Feature	Description
Faster & more efficient	Uses the libsolv dependency resolver
Written in Python 3	Unlike YUM (which used Python 2)
Plugin support	Modular and extensible with plugins
Improved dependency resolution	Uses a SAT solver (libsolv)
Stable API for developers	Easier integration
Better memory management	More scalable
Parallel downloads	Speeds up installations and updates

DNF Package Management Use Cases

Task	Command
List available packages	dnf list available
List packages from a repo	dnf repository-packages baseos list
Download RPM only (no install)	dnf download <package>
Reinstall a package	dnf reinstall <package>
Show dependencies	dnf deplist <package>

DNF vs YUM – Comparison

Feature	YUM	DNF
Default in	RHEL 7	RHEL 8+
Dependency Solver	Basic	Advanced (libsolv)
Python	Python 2	Python 3
Plugin Support	Limited	Better
Speed	Slower	Faster (parallel downloads)
Memory Usage	Higher	Lower
API	Unstable	Stable
Parallel Downloads	❌	☑

DNF Commands list:

1. Check for Updates

sudo dnf check-update

2. Install a Package

sudo dnf install nginx

3. Remove a Package

sudo dnf remove nginx

4. Upgrade All Packages

sudo dnf upgrade

5. Upgrade a Specific Package

sudo dnf upgrade httpd

6. Search for Packages

dnf search python

7. Get Package Information

dnf info vim

8. List Installed Packages

dnf list installed

9. Install from Local .rpm File

sudo dnf install ./somepackage.rpm

10. Install a Software Group

sudo dnf groupinstall "Development Tools"

11. Clean the Cache

sudo dnf clean all

12. View Transaction History

dnf history

Installing, Upgrading, Uninstalling packages- YUM

Firstly, ping to google.com, to check the internet connectivity.

```
[root@C9Server ~]# ping google.com
PING google.com (142.251.42.110) 56(84) bytes of data.
64 bytes from bom07s45-in-f14.1e100.net (142.251.42.110): icmp_seq=1 ttl=128 time=19.0 ms
64 bytes from bom07s45-in-f14.1e100.net (142.251.42.110): icmp_seq=2 ttl=128 time=23.9 ms
64 bytes from bom07s45-in-f14.1e100.net (142.251.42.110): icmp_seq=3 ttl=128 time=21.8 ms
64 bytes from bom07s45-in-f14.1e100.net (142.251.42.110): icmp_seq=4 ttl=128 time=23.4 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3268ms
rtt min/avg/max/mdev = 19.005/22.025/23.912/1.913 ms
[root@C9Server ~]#
```

Also, run the following command on terminal.

```
[root@C9Server ~]# yum clean all
21 files removed
[root@C9Server ~]#
[root@C9Server ~]# yum repolist
repo id                                                     repo name
appstream                                                   CentOS Stream 9 - AppStream
baseos                                                      CentOS Stream 9 - BaseOS
extras-common                                               CentOS Stream 9 - Extras packages
[root@C9Server ~]#
```

To list details of a package (Ex – nginx) using YUM.

```
[root@C9Server ~]# yum info nginx
CentOS Stream 9 - BaseOS
CentOS Stream 9 - AppStream
CentOS Stream 9 - Extras packages
Available Packages
Name         : nginx
Epoch        : 2
Version      : 1.20.1
Release      : 22.el9
Architecture : x86_64
Size         : 37 k
Source       : nginx-1.20.1-22.el9.src.rpm
Repository   : appstream
Summary      : A high performance web server and reverse proxy server
URL          : https://nginx.org
License      : BSD
Description  : Nginx is a web server and a reverse proxy server for HTTP, SMTP, POP3 and
             : IMAP protocols, with a strong focus on high concurrency, performance and low
             : memory usage.

[root@C9Server ~]#
```

To install the package using YUM

```
[root@C9Server ~]# yum install -y nginx
Last metadata expiration check: 0:01:26 ago on Saturday 07 June 2025 06:39:08 AM.
Dependencies resolved.
================================================================================
 Package                          Architecture                Version
================================================================================
Installing:
 nginx                            x86_64                      2:1.20.1-22.el9
Installing dependencies:
 nginx-core                       x86_64                      2:1.20.1-22.el9
 nginx-filesystem                 noarch                      2:1.20.1-22.el9
```

Truncated output…

```
  Verifying        : nginx-2:1.20.1-22.el9.x86_64
  Verifying        : nginx-core-2:1.20.1-22.el9.x86_64
  Verifying        : nginx-filesystem-2:1.20.1-22.el9.noarch

Installed:
  nginx-2:1.20.1-22.el9.x86_64              nginx-core-2:1.20.1-22.el9.x86_64              nginx-filesystem-

Complete!
[root@C9Server ~]#
```

To update packages in linux

```
[root@C9Server ~]# yum update
Last metadata expiration check: 0:04:07 ago on Saturday 07 June 2025 06:39:08 AM.
Dependencies resolved.
================================================================================
 Package                         Arch   Version                     Repo       Size
================================================================================
Installing:
 kernel                          x86_64 5.14.0-587.el9              baseos    297 k
Upgrading:
 NetworkManager                  x86_64 1:1.53.4-1.el9              baseos    2.4 M
 NetworkManager-adsl             x86_64 1:1.53.4-1.el9              baseos     36 k
 NetworkManager-bluetooth        x86_64 1:1.53.4-1.el9              baseos     62 k
 NetworkManager-config-server    noarch 1:1.53.4-1.el9              baseos     21 k
 NetworkManager-libnm            x86_64 1:1.53.4-1.el9              baseos    1.9 M
 NetworkManager-team             x86_64 1:1.53.4-1.el9              baseos     41 k
 NetworkManager-tui              x86_64 1:1.53.4-1.el9              baseos    251 k
 NetworkManager-wifi             x86_64 1:1.53.4-1.el9              baseos     84 k
 NetworkManager-wwan             x86_64 1:1.53.4-1.el9              baseos     69 k
```

Press “y”, if you want to download 1.1GB and update whole OS

```
Installing weak dependencies:
 kernel-devel                    x86_64 5.14.0-587.el9              appstream  21 M

Transaction Summary
================================================================================
Install    7 Packages
Upgrade  302 Packages

Total download size: 1.1 G
Is this ok [y/N]:
```

If you update your OS, a reboot is required (for this, run command “init 6”)

To update specific package:

```
[root@C9Server ~]# yum update httpd -y
Last metadata expiration check: 0:06:52 ago on Saturday 07 June 2025 06:39:08 AM.
Dependencies resolved.
Nothing to do.
Complete!
[root@C9Server ~]#
```

Here, “httpd” is the web server package that needs to be updated.

What is Linux OS Patching?

OS patching means applying updates (security, bug fix, feature) to system software like:

- Kernel
- System libraries (glibc, openssl)
- Services (httpd, sshd, etc.)
- Critical utilities and user-space tools

Why OS Patching Is Important?

Reason	Description
Security	Fix vulnerabilities (CVEs)
Bug Fixes	Resolve system bugs
Performance	Improve system stability and speed
Compliance	Meet regulatory and audit requirements

Types of YUM Updates

Update Type	Command	Description
Regular update	yum update	Updates all installed packages
Security updates only	yum update --security	Security patches only (requires plugin)
Update specific package	yum update <pkg>	Updates only the named package
List updates	yum list updates	Show available package updates

Pre-Patching Checklist

Step	Action
Backup	Backup critical data or take VM snapshot
List current packages	rpm -qa > packages-before.txt
Check available updates	yum check-update
Disable unwanted repos	Temporarily disable 3rd-party repos
Check uptime/usage	Ensure low user/system activity
Test patch in dev	For production systems, always test in staging/dev

Linux OS Patching Using YUM – Step-by-Step

Step 1: Check Available Updates

yum check-update

Step 2: (Optional) Only Show Security Updates

yum updateinfo list security all

Requires:

yum install yum-plugin-security

Step 3: Apply All Updates (OS Patch)

sudo yum update -y

Or for only security patches:

sudo yum update --security -y

Step 4: Reboot (if kernel or glibc is updated)

sudo reboot

Step 5: Verify

uname -r # Check updated kernel version

rpm -qa | grep <package-name> # Check version of specific packages

Useful Patch Management Commands

Task	Command
View update history	yum history
View details of a patch	yum info <package>
List security advisories	yum updateinfo list
Get patch changelog	yum changelog <package> (if plugin installed)

Rollback (Revert) a Patch

- Step 1: Check History
  - ✔ yum history
- Step 2: Undo a Transaction
  - ✔ sudo yum history undo <transaction_id>
    - Note – Rollback doesn't work for all package types or dependencies.

In short,

Task	Command
List available updates	yum check-update
Apply all patches	yum update -y
Apply security patches only	yum update --security -y
Rollback a patch	yum history undo <ID>
View patch history	yum history
View patch log	cat /var/log/yum.log

Linux OS Patching Using DNF – Step-by-Step

Key DNF Patching Concepts

Concept	Description
dnf update	Updates all installed packages to latest available version
dnf upgrade	Alias of dnf update in most cases
--security	Apply only security-related updates
dnf history	Tracks all transactions for rollback or audit
--advisory	Apply based on RHSA, RHBA, etc.

Pre-Patching Steps (Checklist)

Step	Command / Action
Backup system	Snapshot or image-based backup
List current packages	rpm -qa > packages-before.txt
Check available updates	dnf check-update
Test on staging first	Avoid patching production blindly

Patching Linux OS Using DNF – Step-by-Step

1. Check for Available Updates

```
sudo dnf check-update
```

2. Apply All Patches (Recommended)

```
sudo dnf update -y
```

or equivalently:

```
sudo dnf upgrade -y
```

This command updates all system packages, including the kernel and critical libraries.

3. Apply Only Security Updates

```
sudo dnf update --security -y
```

This is useful in environments with strict patch management.

4. Reboot if Kernel or glibc Updated

```
sudo reboot
```

Then confirm:

```
uname -r     # Check if the kernel version has changed
```

5. Verify post-patch

```
dnf list installed
```

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

You can also inspect update history:

```
dnf history
```

What is a Shell in Linux?

A **shell** is a **command-line interface (CLI)** program that allows users to interact with the Linux operating system by entering commands.

It acts as a bridge between the **user** and the **kernel**.

Role of a Shell

Function	Description
🧠 Interpreter	Converts user commands into actions
🛠️ Script Engine	Executes shell scripts (.sh files)
🖥️ Program Launcher	Starts programs and services
📜 Automation Tool	Automates tasks through scripting

Types of Shells in Linux

Shell	Path	Features
Bash (Bourne Again Shell)	/bin/bash	Default shell in most distros, scripting, command history
sh (Bourne Shell)	/bin/sh	Original UNIX shell, simple scripting
zsh	/bin/zsh	Advanced features, autocomplete, themes
ksh (Korn Shell)	/bin/ksh	Compatible with sh, scripting improvements
csh (C Shell)	/bin/csh	C-like syntax, not commonly used now
tcsh	/bin/tcsh	Enhanced csh, supports command line editing

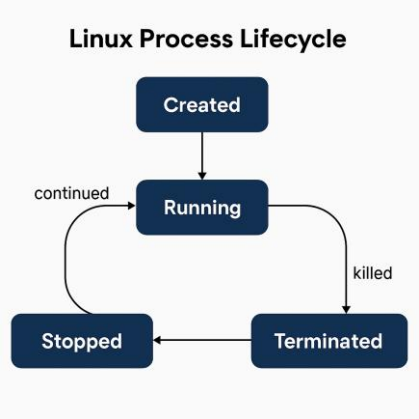
Note - CentOS 9 default shell: ***/bin/bash***

Basic Shell Commands (Bash)

Task	Command
Run a command	ls -l
Command substitution	echo "Today is $(date)"
Redirect output	ls > list.txt
Redirect input	sort < list.txt
Pipe commands	`cat list.txt
Background task	sleep 30 &
Check job	jobs, fg, bg
Exit shell	exit or Ctrl+D

Working with processes

A **process** is an **instance of a running program**. Every time you execute a command, open a service, or run a script — a process is created.



Each process is managed by the Linux kernel and has:

- A unique Process ID (PID)
- A parent process (PPID)
- A set of resources (CPU, memory, files, etc.)

Basic Process Concepts

Term	Meaning
PID	Process ID
PPID	Parent Process ID
UID	User ID (owner of the process)
Foreground Process	Directly attached to a terminal
Background Process	Detached from the terminal (&)
Zombie Process	Process completed, but entry remains in process table
Orphan Process	Parent dies, process gets adopted by init or systemd

Process Signals

Signal	Code	Description
SIGTERM	15	Graceful termination
SIGKILL	9	Force termination
SIGHUP	1	Terminal hangup
SIGSTOP	19	Pause (stop) process
SIGCONT	18	Continue paused process

Managing Processes

Command	Function
kill PID	Sends SIGTERM
kill -9 PID	Sends SIGKILL (force kill)
killall name	Kill by name
nice	Start a process with priority
renice	Change running process priority
nohup	Run process immune to logout
disown	Remove job from shell's job table

Process Management Commands in Linux

Command	Purpose	Example
ps	View snapshot of processes	ps aux
top	Real-time process monitoring	top
htop	Enhanced real-time process monitor (installable)	htop
pidof	Get PID of a process	pidof sshd
pgrep	Search for processes by name	pgrep apache
pkill	Kill processes by name	pkill apache
kill	Send signal to a process by PID	kill -9 1234
killall	Kill all processes by name	killall firefox
nice	Launch process with custom priority	nice -n 10 ./script.sh
renice	Change priority of running process	renice -n 5 -p 1234
jobs	List jobs in current shell	jobs
fg	Bring background job to foreground	fg %1
bg	Resume stopped job in background	bg %1
nohup	Run command immune to hangups	nohup ./script.sh &
disown	Remove job from shell's job list	disown -h %1
watch	Re-run a command periodically	watch -n 2 ps aux
strace	Trace system calls of a process	strace -p 1234
lsof	List open files by processes	lsof -p 1234
pstree	View hierarchical tree of processes	pstree
systemctl	Manage systemd services (processes)	systemctl status sshd
service	Manage older SysV services	service httpd start
uptime	Show load and running processes info	uptime
time	Measure execution time of a command	time ls -l
atop	Advanced process monitoring tool	atop
glances	Cross-platform process monitor	glances
nmon	Performance monitor with process info	nmon
top -u user	Filter processes by user	top -u apache
ps -u user	List processes of a user	ps -u root
ps --forest	Display process hierarchy tree	ps -ef --forest
taskset	Set or retrieve CPU affinity	taskset -cp 1 1234
schedtool	Query or set CPU scheduling policy	schedtool -R -p 1234
timeout	Run a process with a time limit	timeout 30s ./script.sh
cgroup	Limit and isolate resource usage	(via systemd or manual setup)
nice	Adjust priority on new process	nice -n -5 ./prog
renice	Adjust priority on existing PID	renice -n 5 -p 4567

User management

User management in Linux refers to:

- Creating and deleting users
- Managing user groups and permissions
- Setting passwords and account restrictions
- Controlling who can log in and access resources

Important Files for User Management

File	Purpose
/etc/passwd	Stores user account details
/etc/shadow	Stores user passwords (encrypted)
/etc/group	Stores group information
/etc/gshadow	Secure group password file
/home/username/	User's home directory

Creating and Managing Users

Task	Command	Example
Add a user	useradd username	useradd john
Add a user with home directory	useradd -m username	useradd -m david
Add a user with comment	useradd -c "Admin User" username	useradd -c "Backup Admin" raj
Add user with custom shell	useradd -s /bin/bash username	useradd -s /bin/bash alex
Add user with UID	useradd -u 1050 username	useradd -u 1050 dev
Set a password	passwd username	passwd john
Lock user account	passwd -l username	passwd -l testuser
Unlock user account	passwd -u username	passwd -u testuser
Delete user	userdel username	userdel testuser
Delete user with home dir	userdel -r username	userdel -r testuser
Modify user	usermod options username	usermod -s /bin/zsh raj

Group Management

Task	Command	Example
Add group	groupadd groupname	groupadd devs
Delete group	groupdel groupname	groupdel testgrp
Add user to group	usermod -aG group user	usermod -aG devs raj
Remove user from group	Use gpasswd -d	gpasswd -d raj devs
Change user's primary group	usermod -g group user	usermod -g devs raj
View groups for user	groups username	groups raj

View User Information

Command	Usage
id username	Shows UID, GID, and groups
whoami	Shows current user
who	Shows users currently logged in
w	Shows who is logged in and what they are doing
getent passwd	Lists all users
getent group	Lists all groups

Password Aging and Expiry

Task	Command	Example
View aging info	chage -l username	chage -l username
Set password expiry (30 days)	chage -M 30 username	chage -M 30 username
Force password changes at next login	chage -d 0 username	chage -d 0 username

Creating a user in linux (useradd) and verifying:

```
[root@C9Server ~]# useradd -c "Peter Parker" spiderman
[root@C9Server ~]# cat /etc/passwd | grep spiderman
spiderman:x:1001:1001:Peter Parker:/home/spiderman:/bin/bash
[root@C9Server ~]#
[root@C9Server ~]#
```

Modify username (usermod command)

```
[root@C9Server ~]# cat /etc/passwd | grep spiderman
spiderman:x:1001:1001:Peter Parker:/home/spiderman:/bin/bash
[root@C9Server ~]#
[root@C9Server ~]# usermod -c "Tobey Maguire" spiderman
[root@C9Server ~]#
[root@C9Server ~]# cat /etc/passwd | grep spiderman
spiderman:x:1001:1001:Tobey Maguire:/home/spiderman:/bin/bash
[root@C9Server ~]#
```

Deleting user (userdel command)

```
[root@C9Server ~]# cat /etc/passwd | grep spiderman
spiderman:x:1001:1001:Tobey Maguire:/home/spiderman:/bin/bash
[root@C9Server ~]#
[root@C9Server ~]# userdel -r spiderman
[root@C9Server ~]#
[root@C9Server ~]# cat /etc/passwd | grep spiderman
[root@C9Server ~]#
[root@C9Server ~]#
```

Note – No output means, no entry of the user spiderman.

What is sudo in Linux?

- sudo (Superuser Do) is a utility that allows permitted users to execute commands as another user, typically the root user, without switching to that user.
- It is safer than using “su” because it gives limited access with logging and auditing.

Key Configuration Files

File/Command	Purpose
/etc/sudoers	Main sudo configuration file (edit with visudo)
/etc/sudoers.d/	Directory for additional per-user sudo rules
/etc/group	Includes wheel or sudo group for privileged access
visudo	Safe editor for the /etc/sudoers file (syntax-checked)

Format of /etc/sudoers

```
user_or_group  host=(runas_user)  commands  [tags]
```

Commands for Working with Sudo

Command	Description
sudo command	Run a command as root
sudo -i	Start interactive root shell
sudo -u username command	Run as a different user
sudo -k	Invalidate current sudo session (forces password next time)
sudo -l	List permitted commands for current user
sudo su -	Switch to root shell (if allowed)
sudoedit filename	Edit file safely with elevated privilege

Example /etc/sudoers Snippets

```
# Allow user raj to run only user management tools
raj ALL=(ALL) /usr/sbin/useradd, /usr/sbin/usermod, /usr/sbin/userdel

# Allow user dev to run all commands without password
dev ALL=(ALL) NOPASSWD: ALL

# Allow members of admin group to run all commands
%admin ALL=(ALL) ALL
```

What is ACL (Access Control List) in Linux?

An Access Control List (ACL) provides a more granular permission mechanism than traditional owner-group-others file permission (rwx).
ACLs allow you to set file/directory permissions for multiple users or groups, beyond just one owner and one group.

Why Use ACLs?

Traditional Linux file permissions are limited:

- One owner
- One group
- Others

With ACLs, you can:

- Grant different users' different permissions on the same file
- Allow multiple groups to access a file/directory
- Preserve permissions during backups and restores

ACL Terminology

Term	Description
user	Specific user permissions
group	Specific group permissions
mask	Maximum allowed permissions for any named user or group
other	Permissions for users not specified explicitly
default	Default permissions for new files in a directory

Key ACL Commands

Command	Description
setfacl	Set ACL permissions
getfacl	View ACL permissions
setfacl -x	Remove an ACL entry
setfacl -b	Remove all ACLs
setfacl -m	Modify ACLs

Common ACL Commands with Examples

1. Set read permission to user john

setfacl -m u:john:r-- myfile.txt

2. Give user david full access

setfacl -m u:david:rwx myfile.txt

3. Set ACL for a group

setfacl -m g:developers:rw- project.log

4. Set default ACL on a directory (default permissions for new files created inside)

setfacl -d -m u:raj:rwX /mydir

5. View ACLs

getfacl myfile.txt

6. Remove ACL for user

setfacl -x u:david myfile.txt

7. Remove all ACL entries

setfacl -b myfile.txt

ACL summary:

Task	Command
View ACL	getfacl filename
Set user ACL	setfacl -m u:username:permissions file
Set group ACL	setfacl -m g:groupname:permissions file
Remove user/group ACL	setfacl -x u:username file
Remove all ACLs	setfacl -b file
Set default ACL	setfacl -d -m u:user:perm dir

Creating a new file and listing default ACL permissions:

```
[root@C9Server ~]# touch   /aclfile.txt
[root@C9Server ~]# ls -l /aclfile.txt
-rw-r--r--. 1 root root 0 Jun  7 09:43 /aclfile.txt
[root@C9Server ~]#
[root@C9Server ~]# getfacl  /aclfile.txt
getfacl: Removing leading '/' from absolute path names
# file: aclfile.txt
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

Setting ACL and allowing user with read & write access on this file.

```
[root@C9Server ~]# setfacl  -m   u:superuser:rw- /aclfile.txt
[root@C9Server ~]# getfacl  /aclfile.txt
getfacl: Removing leading '/' from absolute path names
# file: aclfile.txt
# owner: root
# group: root
user::rw-
user:superuser:rw-
group::r--
mask::rw-
other::r--

[root@C9Server ~]# ls -l /aclfile.txt
-rw-rw-r--+ 1 root root 0 Jun  7 09:43 /aclfile.txt
[root@C9Server ~]#
[root@C9Server ~]#
```

Note – the "+" sign in "ls -l /aclfile.txt" represents that the file has ACL applied to it.

To remove ACL on a file, use "-b" option.

```
[root@C9Server ~]# getfacl  /aclfile.txt
getfacl: Removing leading '/' from absolute path names
# file: aclfile.txt
# owner: root
# group: root
user::rw-
user:superuser:rw-
group::r--
mask::rw-
other::r--

[root@C9Server ~]#
[root@C9Server ~]# ls -l /aclfile.txt
-rw-rw-r--+ 1 root root 0 Jun  7 09:43 /aclfile.txt
[root@C9Server ~]#
[root@C9Server ~]# setfacl -b  /aclfile.txt
[root@C9Server ~]# ls -l /aclfile.txt
-rw-r--r--. 1 root root 0 Jun  7 09:43 /aclfile.txt
[root@C9Server ~]# getfacl  /aclfile.txt
getfacl: Removing leading '/' from absolute path names
# file: aclfile.txt
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

What is Backup & Restore in Linux?

- Backup is the process of copying data from a source to a secure location for protection against data loss.
- Restore is the act of recovering data from a backup.

Internal Linux Backup Tools (Built-in Commands)

Tool	Purpose
cp	Simple file/directory copy
rsync	Efficient syncing and backup
tar	Archive and compress files
dd	Low-level disk and partition cloning
gzip, bzip2, xz	Compression tools used with tar
find	Used to locate files for backup
cron	Automate backup jobs

Using rsync for Efficient Backup

Cmd: *rsync -avh /etc/ /etc_backup/*

Creating a new directory "/etc_backup"

```
[root@C9Server ~]# mkdir /etc_backup
[root@C9Server ~]#
[root@C9Server ~]# rsync -avh /etc/ /etc_backup/
sending incremental file list
./
.pwd.lock
.updated
DIR_COLORS
DIR_COLORS.lightbgcolor
GREP_COLORS
adjtime
aliases
```

Common Options:

- -a: archive (preserves permissions, timestamps, etc.)
- -v: verbose
- -h: human-readable
- --delete: remove files not in source
- --exclude: skip files/dirs

Backup and Restore using tar

- Full Backup:
  - ✔ tar -cvpzf /backup/full_home.tar.gz /home
  - ✔ Explanation:
    - c: create archive
    - v: verbose
    - p: preserve permissions
    - z: gzip compression
    - f: filename
- Restore:
  - ✔ tar -xvpzf /backup/full_home.tar.gz -C /
- Partial Backup:
  - ✔ tar -cvf /backup/etc_only.tar /etc/passwd /etc/hosts
- Extract a Single File:
  - ✔ tar -xvzf /backup/full_home.tar.gz home/raj/file.txt

Disk/Partition Backup using dd

- Full Disk Clone:
  - ✔ dd if=/dev/sda of=/backup/sda_backup.img bs=4M status=progress
- Restore Disk:
  - ✔ dd if=/backup/sda_backup.img of=/dev/sda bs=4M status=progress
    - ⚠️ Use with caution — it clones the entire disk bit-by-bit.
- Backup MBR (Master Boot Record) Only:
  - ✔ dd if=/dev/sda of=/backup/mbr.img bs=512 count=1

Best Practices

- Always test restore procedures.
- Keep backups offsite or on separate storage.
- Set file permissions (chmod 600) on backup files.
- Monitor disk space during backups.
- Rotate backups to save space (e.g. daily, weekly, monthly).
- Use rsnapshot or scripts to wrap around rsync/tar for incremental backups.

Thank You