**Linux**

<https://discord.gg/p5QXaCW4> Discord channel link :

**What is linux : open source os**

**Linux** is a family of [open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [Unix-like](https://en.wikipedia.org/wiki/Unix-like) [operating systems](https://en.wikipedia.org/wiki/Operating_system) based on the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel),an [operating system kernel](https://en.wikipedia.org/wiki/Kernel_(operating_system)) first released on September 17, 1991, by [**Linus Torvalds**](https://en.wikipedia.org/wiki/Linus_Torvalds)

The [Unix](https://en.wikipedia.org/wiki/Unix) operating system was conceived and implemented in 1969, at [AT&T](https://en.wikipedia.org/wiki/AT%26T_Corporation)'s [Bell Labs](https://en.wikipedia.org/wiki/Bell_Labs),in assembly language , C language



Linux training resources :

1 Youtube

\*\*\*Freecodecamp

\*\*\*The linux foundation

\*\*\*Train with Shubham

\*\*\*Programming knowledge

2.EDX

3. Class central

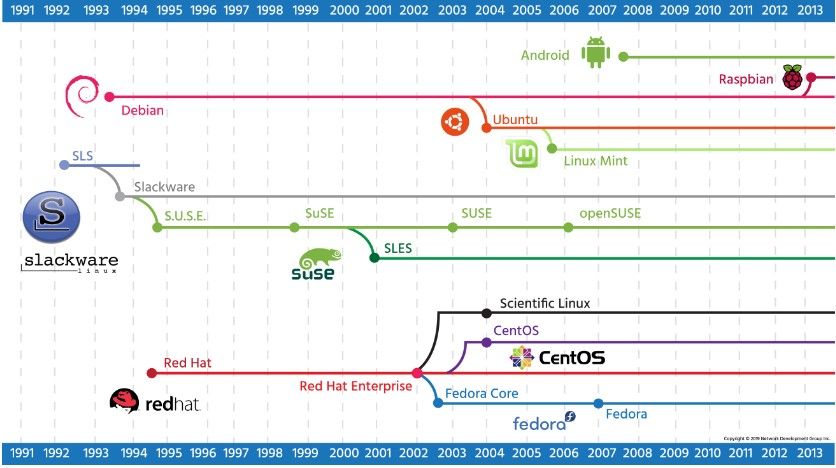
4. CISCO

5. Linux Journe

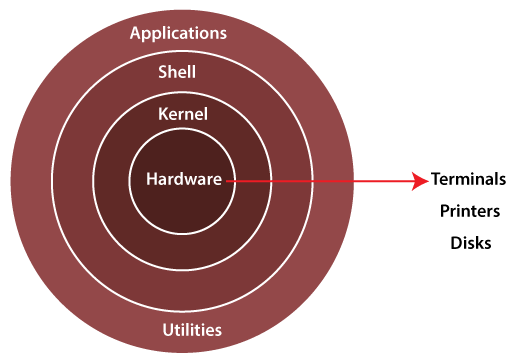
**What is linux distribution :**

Linux is typically [packaged](https://en.wikipedia.org/wiki/Package_manager) as a [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution), which includes the kernel and supporting [system software](https://en.wikipedia.org/wiki/System_software) and [libraries](https://en.wikipedia.org/wiki/Library_(computing)),

***Popular linux distribution :***



**Linux architecture : explain using echo command**



<https://www.tutorialspoint.com/operating_system/os_linux.htm>

he architecture of a Linux System consists of the following layers −

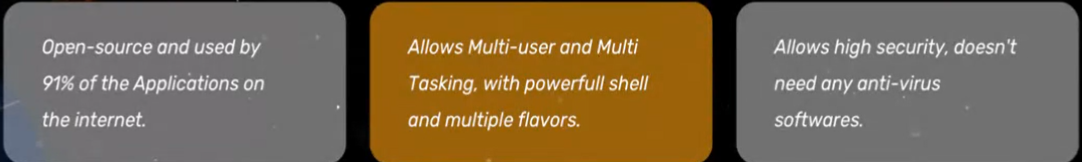
Hardware layer − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).

Kernel − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

Shell − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.

Application layer /Utilities − Utility programs that provide the user most of the functionalities of an operating system.

Commands are programs written in C language …when a user gives command in a shell then the shell translates it into a program that needs to be executed and then the kernel will execute it ..linux command shell command , shell scripting .



Linux is open source , release two time a year (april 04 and oct 10 )

There are 2 types of Ubuntu releases: Interim and LTS. Each Ubuntu LTS is maintained for 10 years total: 5 years of standard support + 5 years of ESM.

There are 2 types of Ubuntu releases: Interim and LTS. Each Ubuntu LTS is maintained for 10 years total: 5 years of standard support + 5 years of ESM.

<https://ubuntu.com/about/release-cycle>

Linux is highly user , no antivirus

Linux is multi user , multitasking os

Linux me har application k liye ek user hota hai

Linux version : flavors - ubuntu , centos mint , fedora ,debian , suse , solarus , kali

—-------------------------------------------------------------------------------------------------------------

**Set up linux :**

1. Set aws free tier account :

<https://www.youtube.com/watch?v=oUoJBayrJT4>

Aws ec2 :

New way:

<https://www.youtube.com/watch?v=OwdIhnpl4BQ>

Old way :

<https://www.youtube.com/watch?v=YSjM_Z_NWpg>

Everything about ec2 :

<https://www.youtube.com/watch?v=8TlukLu11Yo>

1. Set up linux Using vm

<https://www.youtube.com/watch?v=rJ9ysibH768>

1. Set up Virtual Box and Vagrant on Windows

<https://www.youtube.com/watch?v=mPBWWu7sZU4&t=4s>

—-----------------------------------------------------------------------------------------------------------

Ssh -

Through terminal - use .pem key pair

Through putty -use .ppk key pair

—----------------------------------------------------------------------------------------------------------

**Linux commands :**

1. / - root directory
2. Pwd
3. Cd - cd .. – change dir
4. cd / - into root —--ls — os ko chalane k liye required chize all dirs —cd bin (this folder has ll commands and their binaries ) (/bin me sare command hite hai )
5. ls - list files
6. ls -la - list all files and hidden also
7. Clear - clear terminal
8. cd - for going to home —-ls ubuntu user and all user —-cd ~ (/home/ubuntu )
9. whoami
10. Mkdir - make dir (folder )
11. Type file/folder name first letter and tab (autocomplete )
12. touch <file.extension > —-ls —- touch text{1..5}.txt —--(any number of file using wildcard)
13. touch -p A/B/C/D/E —(nested dir ) (-p parent)
14. man (command name )------ for all doc about command
15. cat - concatenate file —------- cat <file > — content of file
16. > redirect - echo “hello ” > first.txt
17. history - all commands history
18. rm <file> - remove file
19. rm -r <dir> - remove dir recursively
20. Text editor - nano , vim , gedit , vi
21. vim <filename> - open file in vim text editor -
22. i (insert mode ) –
23. Esc + :wq - bahar ana and write and quit enter
24. head -3 (first 3 lines of file print )
25. tail -5 (niche se 5 lines print )
26. User add karna —--- sudo useradd -m <username> (-m make dir for user )
27. Create two dir first (create a first\_file.txt )and second - be at parent dir and do cp command

cp first/first.txt second (cp source destination )

1. move : mv first/first\_file.txt second (cut paste )
2. user show in system : cat /etc/passwd
3. Har ek service ek user create karti hai .
4. Switch user : su user
5. Set password : sudo passwd user

Enter passwd :

1. whami — see user (you can do anything in this user also )
2. exit : get out of user
3. Sudo su (superuser )- whoami -(root )-------- ls -ubuntu other users
4. cd user/ (you can switch to any user from root ) - exit
5. System update - packages ke version index update hote hai download hote hai ..not installed

* Sudo apt update

1. System upgrade - packages upgrade hote hi install kar deta hi

* Sudo apt upgrade

1. Install any tool -

* sudo apt install docker.io

1. Uninstall -

* Sudo apt purge docker.io

1. date - today's date
2. cal - Would give you the calendar for this month.

cal 9 1757 - Would give you the calendar for September of 1757.

—------------------------------------------------------------------------------------------------------------------

Jyotsana   
  
File and Directory Permission and Special Permission:

**User, Group and Other Permission Over Files and Directory**:

r (read)   
=File contents can be read.   
=Contents of the directory (the file names) can

be listed.

w (write)   
= File contents can be Changed.  
= Any file in the directory can be created or deleted.

x (execute)

= Files can be executed as commands.

= The directory can become the current working directory. (You can cd into it, but also require read permission to list files found there.)

Q: How to check Permission?  
Ls -l file1.txt = files details

Ls -ld dir1 = directory details

-rw-rw-r--. = user, group, other  
-: regular file, d :directory , l: soft link

ls -ld /home

drwxr-xr-x. 5 root root 4096 Jan 31 22:00 /home

- : Regular file

D : Directory

L - Soft Link

**Changing the Permission with Symbolic Method**

Chmod Who:What:Which file;directory

Who : u,g,o,a

What : +, -, =  
Which: r,w,x

Chmod ug+rw file1

**Changing Permission with Numeric Method**

Chmod 777 file;directory

4 - read

2 - write

1 - execute

Chmod 644 file

Chmod rw-r—r– file

**Change File or Directory - User or Group Ownership**

Chown user1 file321.txt (File Ownership is changed)

Chown :group11 file321.txt (Group ownership can also be changed)

Chgrp group01 file321.txt (Only for Group Ownership change concern)

Chown -R test dir01 (recursively: dir and all its contents user ownership changed)

Chgrp -R test dir2 (recursively: dir and all its contents group ownership changed)

Chown owner:groupowner file321.txt = Changes owner and group owner at the same time

**Special Permission of Files and Directories**:

u+s (suid)   
= File executes as the user that owns the file, not the user that ran the file.   
= No effect.

g+s (sgid)   
= File executes as the group that owns the file.   
= Files newly created in the directory have their group owner set to match the group owner of the directory.

o+t (sticky)   
= No effect.   
= Users with write access to the directory can only remove files that they own; they cannot remove or force saves to files owned by other users.

Example of setuid: ls -l /usr/bin/passwd

Example of setgid: ls -ld /run/log/journal

If the group doesn’t have execute permission, then s in group section is replaced by S

If the others doesn’t have execute permission, then t is represented by T

Example of sticky bit: ls -l /tmp/

**Setting special permission:**

By Symbolic Way: setuid = u+s; setgid = g+s; sticky = o+t

Numerically (fourth preceding digit): setuid = 4; setgid = 2; sticky = 1

Example:  
Chmod g+s directory1

Chmod 2770 dir1 (setgid and r/w/x)

Saswat : 9/8/2023

**Process Management :**

**Process:** Any Running program or command given to a linux system.

A process could run in foreground or background.

-> Run any Process or Command

->Then press **Ctrl+Z**

-> Then **bg –** to send that process to background

-> If we run, **fg** – the process will come to foreground

**top :**

[PID - process id, PR - priority, NI - Nice value ,VIRT - virtual memory used, RES - physical memory, SHR - shared memory used in kb,

S - status(D-uninteruptible sleep,R-Running,S-sleep,T-Traced or stopped,Z-zombie),

%CPU- % of CPU used,%MEM- % physical memory is used,TIME+ - total cpu time,COMMAND- command used ]

**ps:** Process Status

**ps** : show processes

**ps -A** – list of all running processes

**ps -ef** – list of all running process with details

**pidof [process\_name]** – to show the pid of a process

**kill PID** – kill or terminate the process

Niceness : Linux can run multiple processes at a time,It can slow down the speed of some high priority processes and resulting poor performances.

To avoid this it can tell machine to prioritize processes as per requirement.This priority is called Niceness.

Nice value ranges from -20 to 19.

The lower the niceness index, higher would be the priority.

Default value is ‘0’

**df** -- reports the free disk space(Hard Disk) on all the file systems.

**df -h** -- above info in a readable format

**free** -- shows the free and used memory (RAM) on the Linux system.

**netstat -tulpn** – list of listening port

**nslookup google.com**  – tools used to query DNS servers and retrieve info about a specific domain or IP.

Usman : 8/8/2023

**Disk Partitioning and Management in Linux**

**Partitioning Systems**

There are two types of partitioning systems: MBR (Master Boot Record) and GPT (GUID Partition Table).

1. **MBR**:

- Supports up to 4 primary partitions or 3 primary and 1 extended partition.

- Extended partitions can hold 15 to 56 logical partitions.

- Limited to 32-bit systems.

- Max 2TB capacity per partition with BIOS.

- Total usable capacity limited to 2TB on larger disks (e.g., 5TB HDD).

2. **GPT**:

- Supports up to 128 partitions.

- Compatible with UEFI boot mode.

- Utilizes a 64-bit architecture.

- Max partition size up to 9.4 ZB (zettabytes).

**Common Commands**

1. **List Devices and Partitions**: Use `ls /dev` to view all disks and partitions (e.g., sda, sda1, sda2).

2. **Disk Utilization**: `df` displays disk usage, while `df -h` provides human-readable sizes.

3. **Directory Size**: `du -h` shows directory information and size, and `du -sh` displays the size of the current directory.

4. **Swap Partition**: Use `free -h` to view swap partition information.

5. **Partition Creation (MBR)**:

- Use `fdisk /dev/sdb`.

- Enter `n` for new partition, `p` for primary.

- Specify size (e.g., 3GB) for each partition.

- Use `w` to save changes.

- Inform the kernel about changes: `partprobe /dev/sdb`.

6. **Partition Creation (GPT)**:

- Use `gdisk /dev/sdb`.

- Enter `n` for new partition, specify size (e.g., +1G).

- Use partition code (e.g., 8e00 for LVM).

- Save changes with `w`.

7. **Formatting Partitions**:

- Use `mkfs.ext4 /dev/sdb1` for ext4 file system.

- Use `mkfs.xfs /dev/sdb2` for XFS file system.

8. **Mounting Drives**:

- Create a directory (e.g., `mkdir /mnt/mymntdrive`).

- Mount a partition: `mount /dev/sdb2 /mnt/mymntdrive`.

9. **View Mounted Drives**: Use `mount` to display all mounted devices.

10. **Unmounting Drives**: Use `umount /mnt/mymntdrive` to unmount a drive.

[**https://drive.google.com/file/d/1-dKfbof6\_G6zZMvvGX6bOaiseAFANZOx/view?usp=drivesdk**](https://drive.google.com/file/d/1-dKfbof6_G6zZMvvGX6bOaiseAFANZOx/view?usp=drivesdk)

**Let’s Play** 😀:

Login to the system -> do fdisk -l -> din din din our all disks and partition are showing here  -> cd into /dev/ then do ls -> owkhey actually you all guys are here yay i found you  -> okay let's attach one more drive -> now I wanna see the partitions under the disk how to find it? let's check with lsblk ->>>>> woowwww all partitions are showing here -> but our poor recently attached drive is unpartitioned -> let's go into the disk and create partition -> fdisk /dev/sdb -> type n -> then p -> default will be 1 -> go with default sector -> +3G -> press p -> so 1 partition has created -> create more 3 like this -> then create 1 extended to create logical partitions dint it -> press d to delete created partition -> after creating extended partition create logical partitions by pressing n again -> press w to write -> fdisk -l -> now all partitions are showing here but now we can't save data here because we need to format it and need to allocate the filesystem here -> if your kernel is not aware of this activity then inform to the kernal by doing "partprobe /dev/sdb" without restarting the machine -> do blkid next to some partitions uuid is not there means we have not allocated any filesystem  -> let's format the partition -> mkfs.ext2 /dev/sda1 -> blkid woooo filesystem has allocated here -> ok don't by happy  because even after formating the partition you can't store any data here -> so you need to mount it -> so let's do it -> mkdir newdrive1 -> do df -h -> hey our newly added partitions are not here asofnow, how? because we didn't mount them, right? then let's mount it -> mount /dev/sdb1 newdrive1 -> do df -h yahoo this is added -> dan dan -> cd newdrive1 store anything here all data will be storing in sdb1 -> u don't believe? ok let's unmount sdb1 from newdrive1 "umount /dev/sdb1 /newdrive1" -> do df -h -> now go into newdrive1-> do ls -> nothing is there -> create one more directory then mount this partition there then go inside that particular directory you will be seeing the data there now -> how cool na  -> no not cool because this is mounted temporarily-> we need to write it in fstab for permanently mounting -> vi /etc/fstab -> go all the way down -> write -> /dev/sdb1(the partition name, to be mounted) then press tab and go to field number 2 write mount point here /newdrive1 -> then tab in field 3 write filesystem(blkid to see the filesystem) -> then tab and it is field number 4 for Permission we can go with defaults here -> tab -> 5th field to define we should take backup of the drive or not automatically (if want to take backup then setup backup system with dumping command then say 1 else 0) -> if want to check filesystem in booting time then say 1 else 0 (if u set priority 1 then it will check filesystem of every drive so go with 0 if you want this to check in background then set priority as 2) -> now save and exit then reboot the system -> else use mount -a to remount all filesystem listed in fstab -> Great -> now if u want to make antry of other drives in the fstab then do it similarly -> if want delete this partitions agin gdisk into it then d here -> oh o is it saying not possible? yeah because this is not a good practice so first unmount it using umount command to Temporarily unmount but we want these to be permanently unmounted so go to fstab erase all the entries then wq then mount -a -> then fdisk into the drive then d on all the partitions then w so you are out now but if your kernel still unaware about your change then partprobe /dev/sdb now it can know that your partitions are no longer with your system -> then we can remove that hard drive ->

now if you don't want to partition then you can also use your whole drive for that just format the drive -> mkfs.xfs /dev/sdb -f -> lsblk then blkid to check -> then mount it -> or make entry in fstab for permanent mounting -> we can also make entry with uuid for security reasons ->

GPT based Partitioning:

add a hdd/ssd -> to create mbr based partition enter in hdd through fdisk and in gpt based enter through gdisk -> gdisk /dev/sdb -> n -> +200m -> give partition type by hex code caps L to check hex code of partitions-> enter to bydefault partition type (8300- linux partition type) ->n to new so on -> w -> y (to proceed) -> partprobe /dev/sdb->format -> mkfs.xfs /dev/sdb1 -> mkdir /india -> then mount permanently -> mount -a -> df -h

1. find . -type f -mmin +15 - it is used to find files older than 15 minutes and for deleting we can give -delete at the end
2. find . -type f -mtime +5 - it is used to find files older than 10 day and for delete -delete
3. zip -r myfile.zip filename - it is used to compress the files
4. unzip myfile.zip -d /path - it is used to unzip the file
5. chmod 775 filename - it is used to change the file permissions
6. chown -R newuser:newgroup filename - it is used to change the file ownership recursively
7. ifconfig - display network interfaces and IP addresses

Ruby : 9/8/2023

**Shell scripting examples :**

A shell script is a series of commands and instructions written in a scripting language that is interpreted by a shell, which is a command-line interface provided by an operating system. Shell scripting allows users to automate tasks, perform system administration tasks, manipulate files and data, and execute a sequence of commands as if they were entered directly into the command line.

Shell scripts can be used for a variety of purposes, including:

**Automation**: Automate repetitive tasks, such as backups, file processing, and system maintenance.

**Configuration:** Configure system settings, software installations, and environment variables.

**Data Manipulation:** Process and manipulate text, files, and data streams.

**System Administration**: Perform administrative tasks like user management, log rotation, and system monitoring.

**Batch Processing:** Run a sequence of commands in a specific order, often used for scheduled jobs.

**Conditional Execution:** Execute commands conditionally based on specific criteria.

**Control Structures:** Use loops and conditional statements to control program flow.

Shell scripts are typically written in languages such as Bash (Bourne-Again Shell), which is the default shell on most Unix-like operating systems, including Linux. Other popular shells include sh, csh, ksh, and zsh. Shell scripts are usually saved in plain text files with a .sh extension.

Hello World Script:

#!/bin/bash

echo "Hello, world!"

—-----

User Input:

#!/bin/bash

echo "Enter your name:"

read name

echo "Hello, $name!"

—--------

Variables:

#!/bin/bash

message="Hello, Linux!"

echo $message

—--------------

Command Line Arguments:

#!/bin/bash

echo "Script name: $0"

echo "First argument: $1"

echo "Second argument: $2"

—-------------

Arithmetic Operations:

#!/bin/bash

num1=10

num2=5

sum=$((num1 + num2))

echo "Sum: $sum"

—--------------

If-Else Statement:

#!/bin/bash

if [ "$1" -eq 10 ]; then

echo "Argument is 10"

else

echo "Argument is not 10"

fi

—--------

For Loop:

#!/bin/bash

for i in {1..5}; d

echo "Number: $i"

done

—------------

While Loop:

#!/bin/bash

count=1

while [ $count -le 5 ]; do

echo "Count: $count"

((count++))

done

—------------

Functions:

#!/bin/bash

greet() {

echo "Hello, $1!"

}

greet "Alice"

—---------

File Existence Check:

#!/bin/bash

if [ -e "myfile.txt" ]; then

echo "File exists"

else

echo "File does not exist"

fi

—---------------

String Manipulation:

#!/bin/bash

string="Hello, Linux Scripting"

echo "Length: ${#string}"

echo "Substring: ${string:7:5}"

echo "Uppercase: ${string^^}"

—------------

Read Lines from a File:

#!/bin/bash

while IFS= read -r line; do

echo "Line: $line"

done < myfile.txt

—----------

Array:

#!/bin/bash

fruits=("Apple" "Banana" "Orange")

echo "First fruit: ${fruits[0]}"

echo "All fruits: ${fruits[@]}"

—------------------

Case Statement:

#!/bin/bash

case "$1" in

"start")

echo "Starting service"

;;

"stop")

echo "Stopping service"

;;

\*)

echo "Unknown command"

;;

esac

—---------------

Command Substitution:

#!/bin/bash

files=$(ls)

echo "Files: $files"

—---------------

File Operations:

#!/bin/bash

touch newfile.txt

echo "New file created"

rm newfile.txt

echo "File deleted"

—-----------

Exit Codes:

#!/bin/bash

if [ $# -ne 2 ]; then

echo "Usage: $0 <arg1> <arg2>"

exit 1

fi

—--------------

Environment Variables:

#!/bin/bash

echo "User: $USER"

echo "Home directory: $HOME"

—----------------

Redirecting Output:

#!/bin/bash

ls > filelist.txt

echo "Output redirected"

—----------

Creating Directories:

#!/bin/bash

if [ ! -d "mydir" ]; then

mkdir mydir

echo "Directory created"

else

echo "Directory already exists"

fi

—-----------------

a backup of files from one folder to another:

#!/bin/bash

# Source and destination directories

source\_dir="/path/to/source"

backup\_dir="/path/to/backup"

# Create the backup directory if it doesn't exist

mkdir -p "$backup\_dir"

# Copy files from source to backup

cp -r "$source\_dir"/\* "$backup\_dir"

echo "Backup completed from $source\_dir to $backup\_dir"

—------------

Replace /path/to/source with the path to the folder you want to backup, and /path/to/backup with the destination folder where you want to store the backup.

Save this script in a file, give it execute permissions (chmod +x scriptname.sh), and then run it (./scriptname.sh). This script uses the cp command to recursively copy files from the source directory to the backup directory.

—-------

Here are some use cases of shell scripts with examples:

1. **Automating Backups:**

Use Case: Automate the backup of important files to a designated backup directory.

Example:

# backup.sh

# Backup important files to a backup directory

source\_dir="/path/to/source"

backup\_dir="/path/to/backup"

mkdir -p "$backup\_dir"

cp -r "$source\_dir"/\* "$backup\_dir"

echo "Backup completed from $source\_dir to $backup\_dir"

—---------------------------------

1. **System Monitoring and Reporting:**

Use Case: Monitor system resources and generate periodic reports.

Example:

# monitor\_resources.sh

# Monitor system resources and generate a report

cpu\_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')

memory\_usage=$(free -h | awk 'NR==2 {print $3}')

echo "CPU Usage: $cpu\_usage%"

echo "Memory Usage: $memory\_usage"

—-----------------

1. **Software Deployment:**

Use Case: Deploy software on multiple servers using a consistent script.

Example:

# deploy\_app.sh

# Deploy a web application to multiple servers

servers=("server1" "server2" "server3")

app\_dir="/path/to/app"

for server in "${servers[@]}"; do

scp -r "$app\_dir" "$server:/app"

ssh "$server" "systemctl restart app-service"

done

echo "Application deployed to all servers"

—----------------

1. **Log Rotation:**

Use Case: Rotate and manage log files to prevent disk space issues.

Example:

# rotate\_logs.sh

# Rotate and compress log files older than a certain date

log\_dir="/var/log"

old\_logs\_dir="/var/log/old\_logs"

cutoff\_date=$(date -d "7 days ago" "+%Y%m%d")

mkdir -p "$old\_logs\_dir"

find "$log\_dir" -type f -name "\*.log" -mtime +7 -exec mv {} "$old\_logs\_dir" \;

echo "Log files rotated and moved to $old\_logs\_dir"

—----------------------

1. **Automated Testing:**

Use Case: Automate testing tasks, such as running unit tests for a software project.

Example:

# run\_tests.sh

# Run unit tests for a software project

tests\_dir="/path/to/tests"

test\_report="test\_report.txt"

python -m unittest discover "$tests\_dir" > "$test\_report"

echo "Test report generated: $test\_report"

—------------------------

1. **File Processing:**

Use Case: Process files in a folder, perform transformations, and generate reports.

Example:

# process\_files.sh

# Process text files, convert to uppercase, and generate a report

input\_dir="/path/to/input"

output\_dir="/path/to/output"

mkdir -p "$output\_dir"

for file in "$input\_dir"/\*.txt; do

output\_file="$output\_dir/$(basename "$file")"

tr '[:lower:]' '[:upper:]' < "$file" > "$output\_file"

done

echo "Files processed and saved to $output\_dir"