Rashtriya Raksha University

School of Information Technology, Artificial Intelligence & Cyber Security (SITAICS)

At- Lavad, Dahegam, Gandhinagar, Gujarat-382305

LSS Practical File

(Linux and Shell Scripting)

Name:	Sarthak Sanay
Enrollment No:	230031101611051
Subject Name:	Linux and Shell Scripting (G4A20LSS)
Program:	B.Tech CSE (with specialization in Cyber Security)
Year:	2nd year (Semester-IV)

This is to certify that **Mr. Sarthak Sanay** has satisfactorily completed **25** out of **25** practical work prescribed by SITAICS (School of Information Technology, Artificial Intelligence, & Cyber Security) at the **GFL** laboratory.

Dr. Vivek Joshi
SUBJECT INCHARGE

INDEX

Enrollment No: 230031101611051

Name: Sarthak Sanay

Subject: Linux and Shell Scripting

Subject (Code): G4A20LSS

SR.NO.	**TITLE**
10	Write a shell script to display the CPU, Memory, & Disk Space Usage.
11	Write a script to monitor a running process (eg: Apache, MySQL), and notify if it stops.
12	Write a shell script to log system uptime and the number of logged-in users every 5 minutes.
13	Write a script that checks the top 5 memory-consuming processes.
14	Provide an example of a script that extracts error messages from /var/log/syslog.
15	Write a script to find and list all files larger than 100 MB in the home directory.
16	Write a shell script to check the status of network connectivity. (eg: ping to a server).
17	Write a shell script to back up a specified directory daily.
18	Write a cron expression to schedule a shell script that cleans temporary files every night at midnight.
19	Write a shell script to monitor disk I/O using the iostat command.
20	Write a script that counts the number of failed login attempts from the /var/log/auth.log file.

SR.NO.	**TITLE**
21	Write a shell script to monitor the temperature of the CPU (if supported by sensors).
22	Write a shell script to generate a report of system resource usage (CPU, memory, disk) and email it to the admin.
23	Explain the importance of logging in shell scripts. Modify one of your earlier scripts to write logs with timestamps to a file.
24	Write an Awk script to count the number of lines in a given file
25	Write an Awk script to display lines that contain a specific keyword, such as "fail" or "error"

PRACTICAL - 1

AIM: To make your own custom Ubuntu OS

Step 1: Set Up the Required Directories and Tools :-

1. Install required packages:

```
➢ sudo apt update && sudo apt install -y squashfs-tools
  xorriso genisoimage isolinux syslinux
```

2. Create the working directories:

```
➢ mkdir ~/custom-iso
➢ cd ~/custom-iso
➢ mkdir mnt work iso
```

3. Mount the Ubuntu ISO:

```
➢ sudo mount -o loop /path/to/ubuntu.iso mnt
```

4. Copy the contents of the mounted ISO:

```
➢ cp -a mnt/. iso/
```

5. Unmount the ISO:

```
➢ sudo unmount mnt
```

Step 2: Extract and Modify the SquashFS Filesystem:-

1. Mount the SquashFS filesystem:

```
➢ sudo mount -o loop iso/casper/filesystem.squashfs work/
```

2. Copy the filesystem to make it writable:

```
➢ sudo cp -a work/. work_copy/
➢ sudo umount work
```

3. Chroot into the mounted filesystem:

```
➢ sudo chroot work_copy/
```

4. Update the system repositories:

```
➢ sudo apt update && apt upgrade -y
```

Step 3: Add Custom RRU Wallpaper and Logo:-

1. Add the RRU wallpaper to the default user's folder:

```
➢ cp ~/Documents/rru-wallpaper.jpg
  /usr/share/backgrounds/rru-wallpaper.jpg
```

2. Set the wallpaper as default:

```
➢ gsettings set org.gnome.desktop.background picture-uri
  file:///usr/share/backgrounds/rru-wallpaper.jpg
```

3. Change Ubuntu logo to RRU logo

➢ `cp ~/Documents/rru-logo.png /usr/share/plymouth/themes/ubuntu-logo/ubuntu-logo.png`

Step 4: Add BTech Syllabus :-

1. Add B.Tech Syllabus

➢ `cp /path/to/syllabus.pdf /home/user/`

2. Ensure that the file is accessible after installation

➢ `chown user:user /home/user/syllabus.pdf`

Step 5: Configure iptables rule :-

1. Configure iptables rules (Edit /etc/iptables/rules.v4):

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables-save > /etc/iptables/rules.v4
```

2. Ensure iptables is loaded at boot:

➢ `apt install iptables-persistent`

3. Save iptables rules:

➢ `iptables-save > /etc/iptables/rules.v4`

Step 6: Remove Pre-installed Applications :-

1. To see all the installations, and then choose which to uninstall.

```
➢ dpkg --get-selections
```

2. Remove unnecessary pre-installed applications:
   a. Remove Thunderbird (email client)
   ```
   ➢ sudo apt remove --purge thunderbird -y
   ```
   b. Remove Rhythmbox (default music player)
   ```
   ➢ sudo apt remove --purge rhythmbox -y
   ```

3. Clean up unused dependencies:

```
➢ sudo apt autoremove --purge -y
➢ sudo apt clean
```

Step 7: Install New Applications :-

1. **Install Google Chrome** (It is not there in the apt repository by default, so we need to add the repository before installing it.)
   a. Download the Chrome .deb package (using wget)
   ```
   ➢ wget
     https://dl.google.com/linux/direct/google-chrom
     e-stable_current_amd64.deb
   ```

b. Install Google Chrome

➢ `sudo dpkg -i google-chrome-stable_current_amd64.deb`

➢ `sudo apt-get install -f -y`

2. **Install Spotify**:

a. Setup the prerequisites:

➢ `sudo apt install curl -y`

➢ `curl -sS https://download.spotify.com/debian/pubkey.gpg | sudo tee /etc/apt/trusted.gpg.d/spotify.asc`

b. Add the Spotify repository:

➢ `echo "deb http://repository.spotify.com stable non-free" | sudo tee /etc/apt/sources.list.d/spotify.list`

c. Update the apt package list and install Spotify:

➢ `sudo apt update`

➢ `sudo apt install spotify-client -y`

3. **Install VLC** (lightweight media player)

➢ `sudo apt install vlc -y`

4. **Install GIMP** (FOSS image editor)

➢ `sudo apt install gimp -y`

5. Update apt list (To add the latest versions of the applications in the repository list)

```
➢ sudo apt update
```

Step 8: Unmount and Rebuild the SquashFS Filesystem:-

1. Unmount the filesystem:

```
➢ exit
➢ sudo umount work_copy
```

2. Rebuild the SquashFS with the new changes:

```
➢ sudo mksquashfs work_copy iso/casper/filesystem.squashfs
  -comp xz -e boot
```

Step 9: Rebuild the ISO:-

1. Create the new bootable ISO:

```
➢ sudo mkisofs -D -r -V "Custom Ubuntu" -cache-inodes -J
  -l -b isolinux/isolinux.bin -c isolinux/boot.cat
  -no-emul-boot -boot-load-size 4 -boot-info-table -o
  /path/to/custom-ubuntu.iso iso/
```

2. Make the ISO bootable:

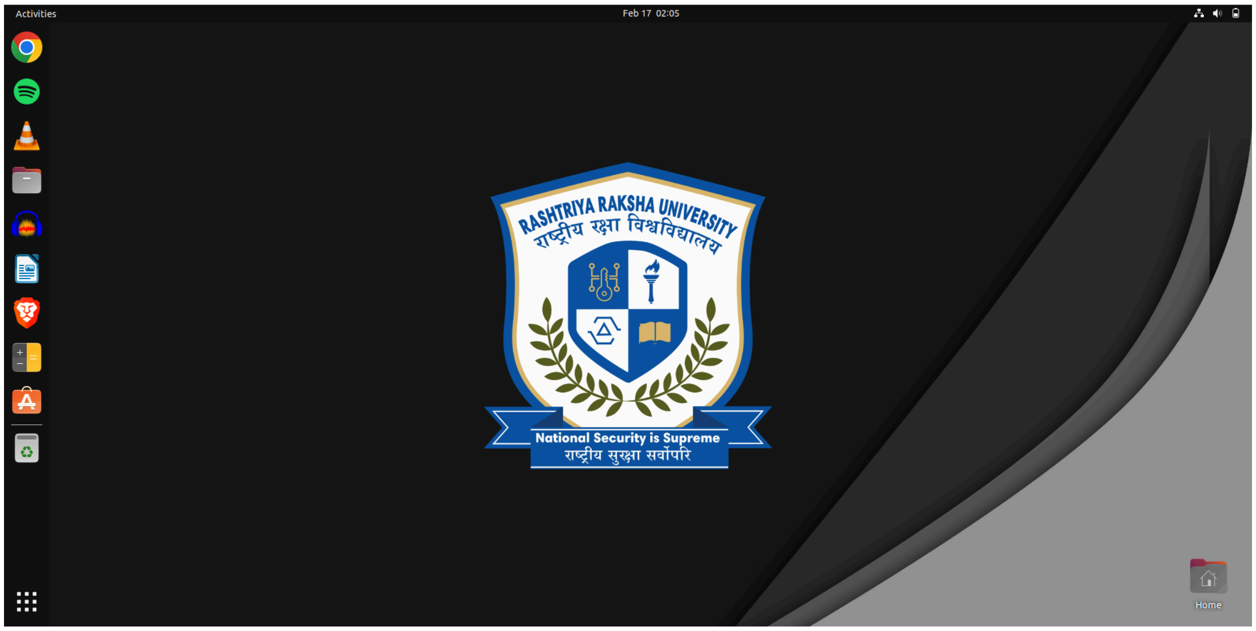
```
➢ sudo isohybrid /path/to/custom-ubuntu.iso
```

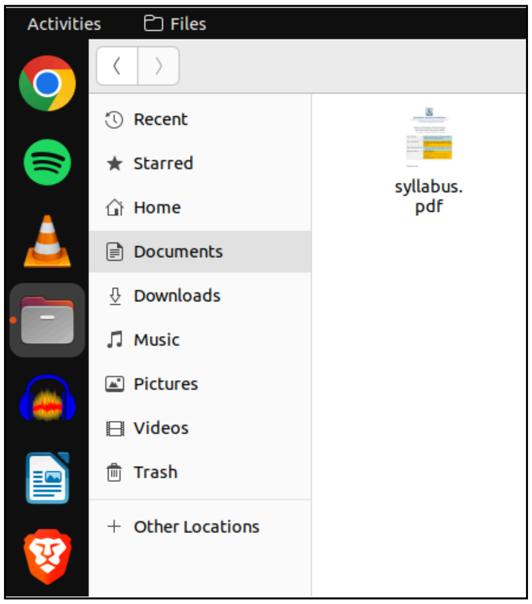
Step 10: Running the ISO in VirtualBox :-

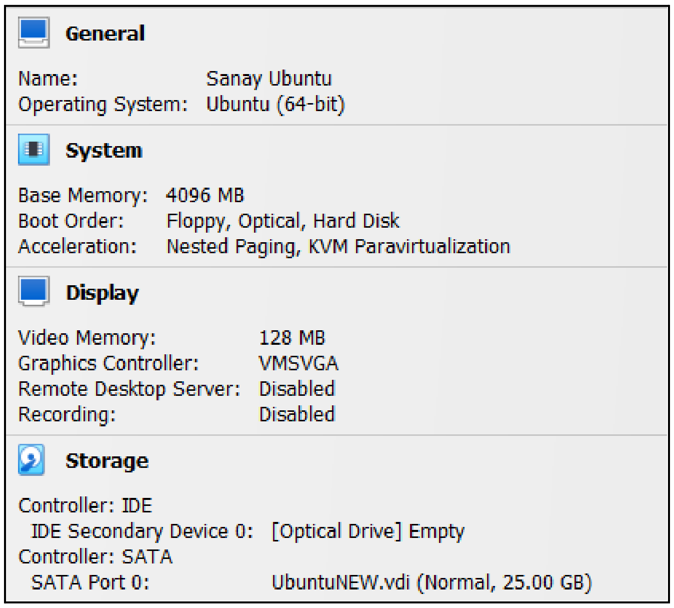
Note: To run the ISO using a bootable USB, make sure to burn it beforehand. Use the following command:

➢
```
sudo dd if=/path/to/custom-ubuntu.iso of=/dev/sdX bs=4M
status=progress && sync
```

Over here, the /dev/sdX is the name of the pendrive.


Activities
Feb 17 02:05
RASHTRIYA RAKSHA UNIVERSITY
राष्ट्रीय रक्षा विश्वविद्यालय
National Security is Supreme
राष्ट्रीय सुरक्षा सर्वोपरि
Home





General

Name: Sanay Ubuntu
Operating System: Ubuntu (64-bit)

System

Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: Nested Paging, KVM Paravirtualization

Display

Video Memory: 128 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: UbuntuNEW.vdi (Normal, 25.00 GB)

PRACTICAL - 2

AIM: Setup Docker Engine (Terminal) in Linux

About Docker :-

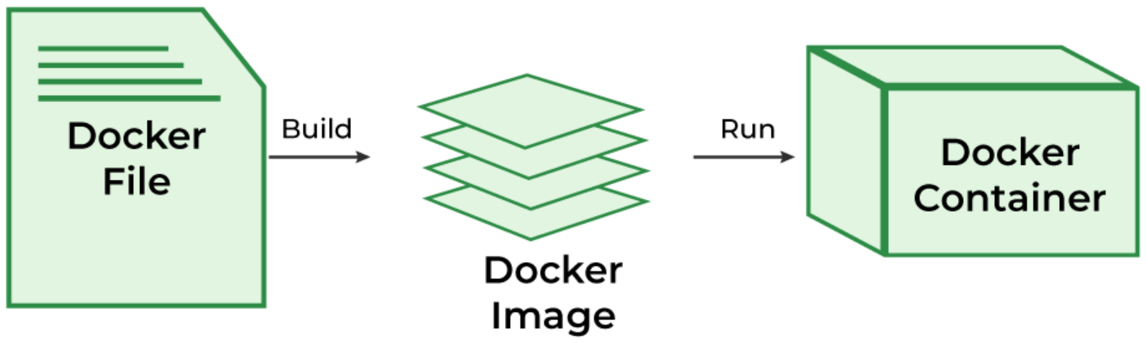
Docker is an open-source containerization platform by which you can pack your application and all its dependencies into a standardized unit called a container. Containers are light in weight which makes them portable and they are isolated from the underlying infrastructure and from each other container. You can run the docker image as a docker container in any machine where docker is installed without depending on the operating system.

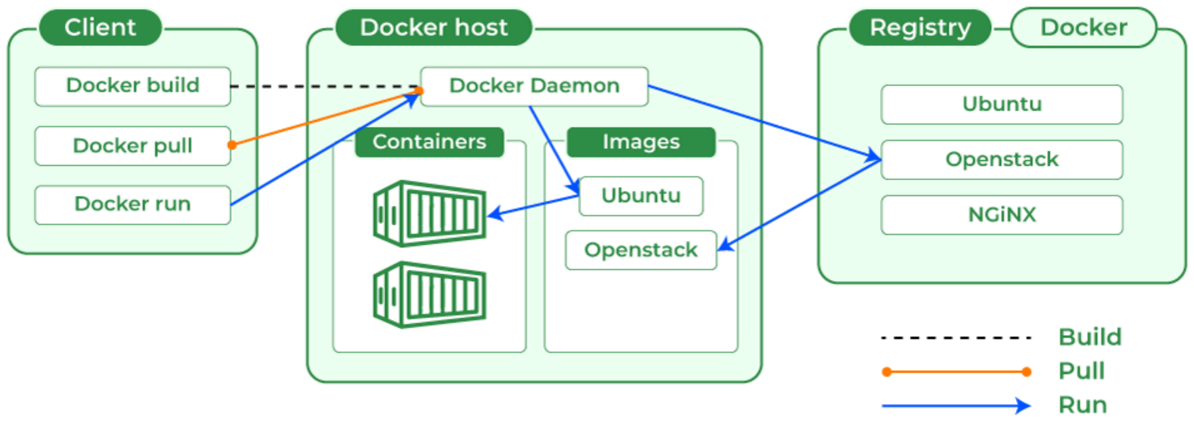
Features of Docker :-

- ➢ **Portability**: Docker facilitates the developers in packaging their applications with all dependencies into a single lightweight containers. It facilities in ensuring the consistent performance across the different computing environments.
- ➢ **Reproducibility**: Through encapsulating the applications with their dependencies within a container it ensures in software setups remaining consistent across the development, testing and production environments.
- ➢ **Efficiency**: Docker through its container based architecture it optimizes the resource utilization. It allows the developers to run the multiple isolated applications on a single host system.
- ➢ **Scalability**: Docker's scalability features facilitated the developers in making easier of their applications handling at time of workloads increment.

Key Components of Docker :-

- **Docker Engine**: It is a core part of docker, that handles the creation and management of containers.
- **Docker Image**: It is a read-only template that is used for creating containers, containing the application code and dependencies.
- **Docker Hub**: It is a cloud based repository that is used for finding and sharing the container images.
- **Dockerfile**: It is a script that containing instructions to build a docker image.
- **Docker Registry**: It is a storage distribution system for docker images, where you can store the images in both public and private modes.





Note: The above image showcases the Docker Architecture and how it works.

Difference Between Docker Containers and Virtual Machines :-

Docker Containers	**Virtual Machines**
Docker Containers contain binaries, libraries, and configuration files along with the application itself.	Virtual Machines (VMs) run on Hypervisors, which allow multiple Virtual Machines to run on a single machine along with its own operating system.
They don't contain a guest OS for each container and rely on the underlying OS kernel, which makes the containers lightweight.	Each VM has its own copy of an operating system along with the application and necessary binaries, which makes it significantly larger and it requires more resources.
Containers share resources with other containers in the same host OS and provide OS-level process isolation.	They provide Hardware-level process isolation and are slow to boot.

Steps to Install Docker Engine on Ubuntu :-

➢ Open your browser and visit the official website to install Docker Engine for Ubuntu (Linux), and follow the instructions :-

https://docs.docker.com/engine/install/ubuntu/

➢ Run the following command to uninstall all conflicting packages:

```
for pkg in docker.io docker-doc docker-compose
docker-compose-v2 podman-docker containerd runc; do sudo
apt-get remove $pkg; done
```

Now, we are going to install it using the **apt** repository.

1. Set up Docker's **apt** repository.

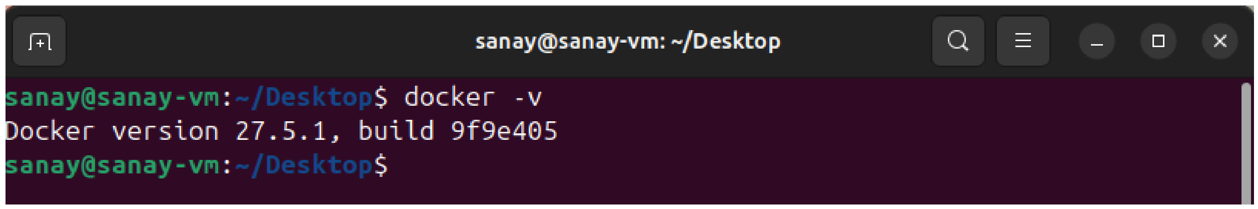
```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL
https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo
"${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
sudo apt-get update
```

2. Install the latest Docker packages.

   ```
   sudo apt-get install docker-ce docker-ce-cli
   containerd.io docker-buildx-plugin docker-compose-plugin
   ```

3. Verify that the installation is successful by running the hello-world image:

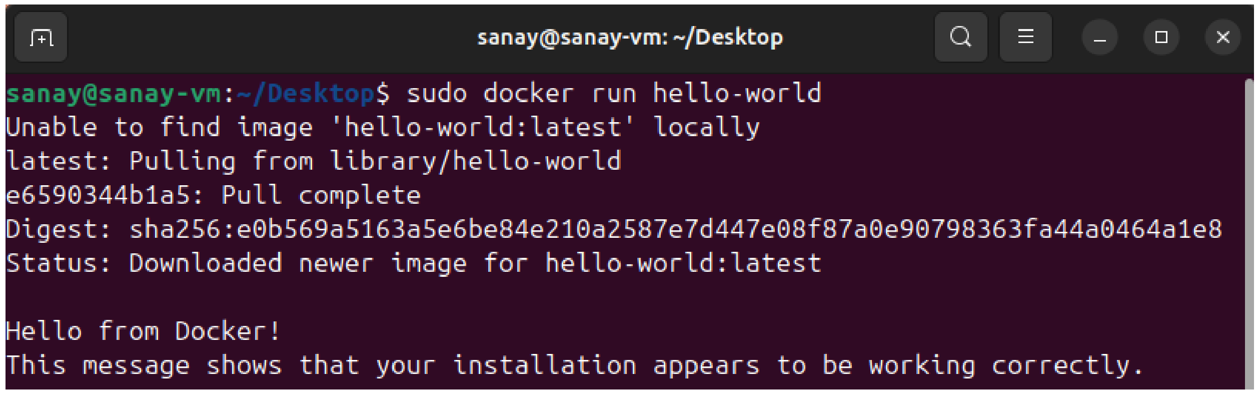
   ```
   docker -v
   ```



```
sanay@sanay-vm:~/Desktop$ docker -v
Docker version 27.5.1, build 9f9e405
sanay@sanay-vm:~/Desktop$
```

   or,

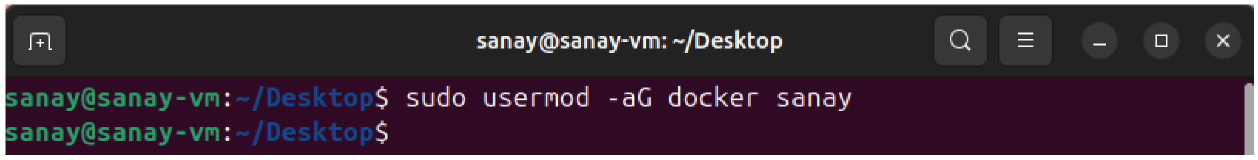
   ```
   sudo docker run hello-world
   ```



```
sanay@sanay-vm:~/Desktop$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:e0b569a5163a5e6be84e210a2587e7d447e08f87a0e90798363fa44a0464a1e8
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Additional enhancement: To run Docker without using sudo everytime, add your user to the Docker sudoers list using the following command:

➢ `sudo usermod -aG docker $USER`



```
sanay@sanay-vm:~/Desktop$ sudo usermod -aG docker sanay
sanay@sanay-vm:~/Desktop$
```

PRACTICAL - 3

AIM: Setup and Host website using NGINX Server

1. *INSTALLATION :-*

```
➢ sudo apt update
➢ sudo apt install nginx
```

2. *CREATING YOUR WEB APPLICATION :-*

By default, the page is located at the path `/var/www/html/` location. You can place your static pages here, or customise the configuration file and place it at some other location.

Let's say, our project name is "tutorial".

So, simply create a directory named "tutorial", and then make a new HTML page in the path `/var/www/tutorial/`. Create `index.html` file in this location. Use the following commands :-

```
➢ cd /var/www
➢ sudo mkdir tutorial
➢ cd tutorial
➢ touch index.html
```

Open the index.html file using nano or any text editor of your choice.

```
➢ sudo nano index.html
```

Paste the following boilerplate code for a basic HTML site.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Website!</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>We have just configured our Nginx web server
on Ubuntu!</p>
</body>
</html>
```

- ➔ Press `Ctrl + S` to save!
- ➔ Press `Ctrl + X` to exit the nano text-editor!

In the next step we are going to make changes to the configuration file to make Nginx use pages from this location.

3. ***CREATING A NEW CONFIGURATION*** **:-**

To set up a virtual host, we need to create a file in `/etc/nginx/sites-enabled/` directory.
This is the path where all the config files are located, for the sites hosted using Nginx.

Follow the next commands to move to the sites-enabled directory and create a new configuration file.

```
➢ cd /etc/nginx/sites-enabled
➢ touch tutorial
➢ sudo nano tutorial
```

Paste the following server block into the *tutorial* config file.

```
server {
        listen 81;
        listen [::]:81;

        server_name _;

        root /var/www/tutorial;
        index index.html;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

- ➔ Press `Ctrl + S` to save!
- ➔ Press `Ctrl + X` to exit the nano text-editor!

<u>Note</u>: *For this tutorial, we will make our site available on 81 port, not the standard 80 port. You can change it if you would like to.*
You can use any port number. There are a total of 65,535 TCP/UDP ports available.
If you want to host your website using port 80 (HTTP), then make changes to the “default” file, by replacing the parameters in that file with our needed configuration.

4. ***<u>ACTIVATING THE SERVER AND TESTING RESULTS</u>*** **:-**

```
➢ sudo systemctl enable nginx
➢ sudo systemctl start nginx
➢ Sudo systemctl status nginx
```

Make sure to always restart the server after making any changes to the code of the website. Use the following command for it :-

```
➢ sudo systemctl restart nginx
```

Now, to find your IP Address, use either of the following commands :-

- ➢ `ip a`

  or use,

- ➢ `ifconfig`

Your website is now live! 🎉
Open up your browser, and type in the following.

To access it anywhere in the LAN, use the following :-

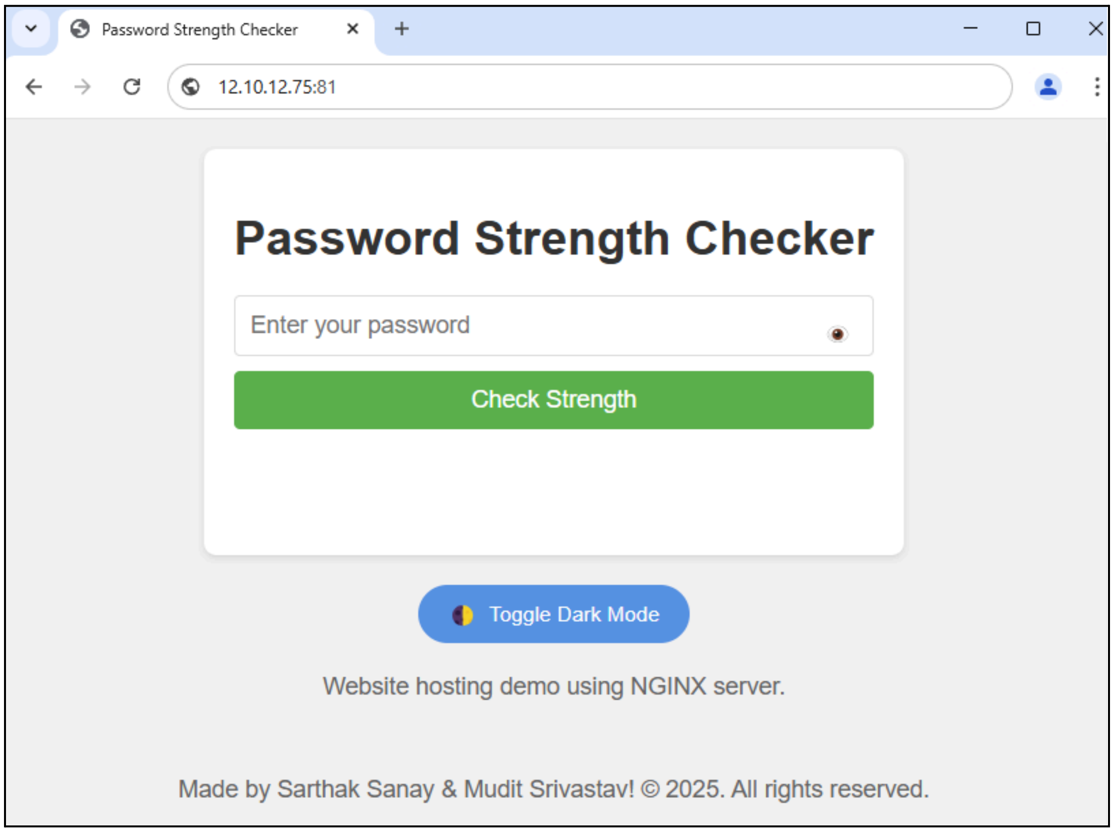
`http://your_ip:81`

For example, if my IP address is 12.10.12.75, then I'll use :-

`http://12.10.12.75:81`

To access it on your system, you can also use :-

`http://localhost:81`



Website hosting demo using NGINX server.

Made by Sarthak Sanay & Mudit Srivastav! © 2025. All rights reserved.

PRACTICAL - 4

AIM: Run NGINX Server within Docker container

Why to run NGINX within a Docker container ?

Running NGINX in a Docker container eliminates administration overhead. You don't need to manage NGINX through a package manager or build it from source. Docker allows you to replace the entire container when a new NGINX version is released, with no extra administrative tasks. This means you only need to maintain the configuration file and your content, simplifying the management process.

Prerequisites :-

- ➢ Ubuntu 22 or later (latest version is 24)
- ➢ Docker Engine or Docker Desktop installed on your Ubuntu machine.

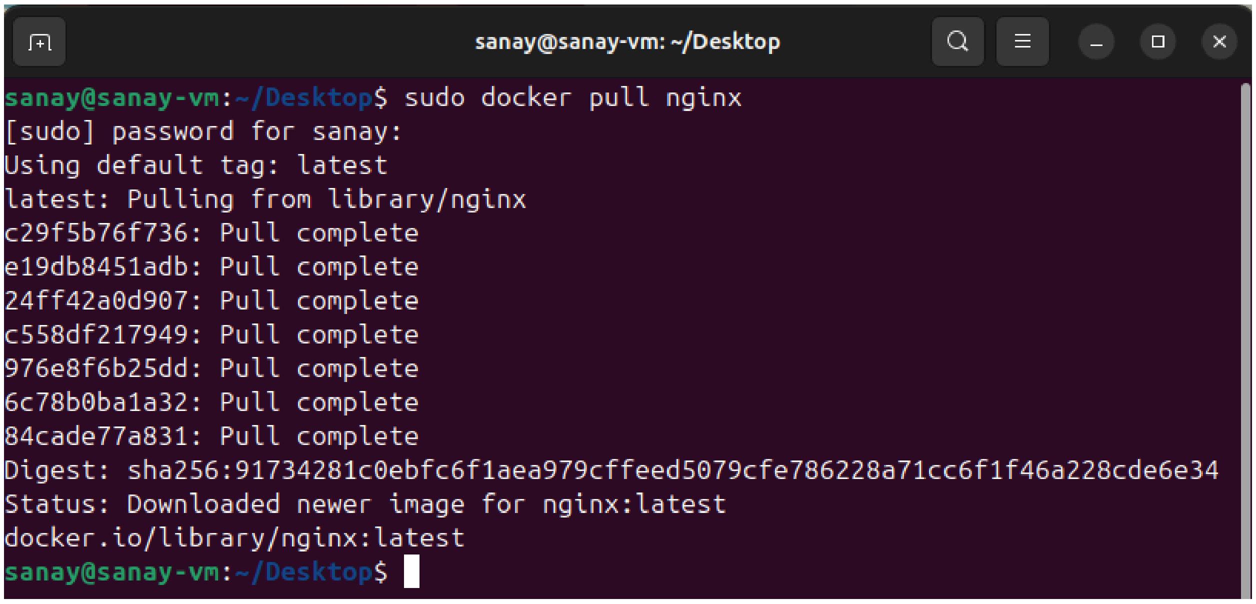
Basic Setup :-

Run the following commands in your terminal :-

```
➢ sudo docker pull nginx
➢ sudo docker run --name docker-nginx -p 80:80 nginx
```

Explanation to breakdown the working of the commands:

- pull nginx downloads a pre-built Docker image for nginx with default configuration.
- run is the command to create a new container
- The --name flag is how you specify the name of the container. If left blank, a generated name like nostalgic_hopper will be assigned.
- -p specifies the port you are exposing in the format of -p local-machine-port:internal-container-port. In this case, you are mapping port :80 in the container to port :80 on the server.
- nginx is the name of the image on Docker Hub.



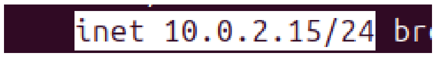
```
sanay@sanay-vm:~/Desktop$ sudo docker pull nginx
[sudo] password for sanay:
Using default tag: latest
latest: Pulling from library/nginx
c29f5b76f736: Pull complete
e19db8451adb: Pull complete
24ff42a0d907: Pull complete
c558df217949: Pull complete
976e8f6b25dd: Pull complete
6c78b0ba1a32: Pull complete
84cade77a831: Pull complete
Digest: sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e34
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
sanay@sanay-vm:~/Desktop$
```

Use the following command to get your system’s IP address:

➢ `ip a`

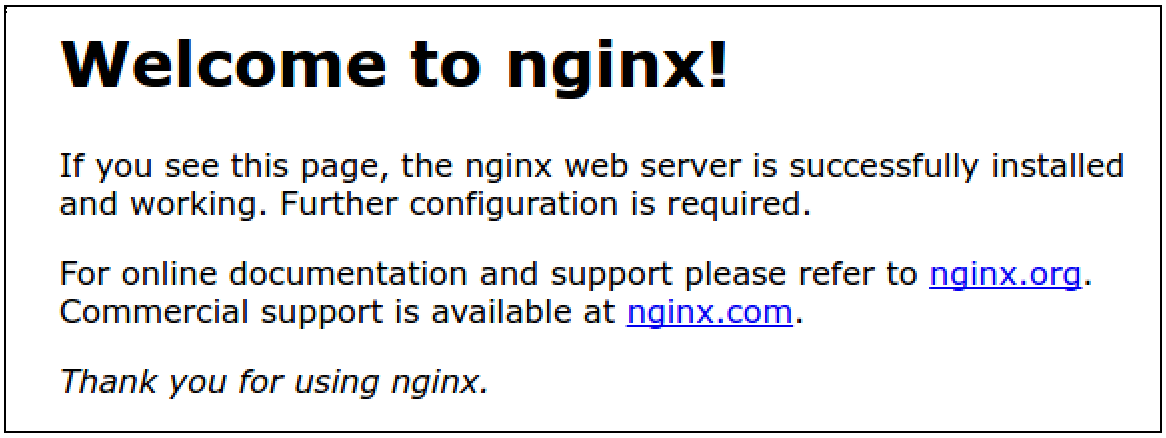
or,

➢ `ifconfig`



```
    inet 10.0.2.15/24 br
```

In the web browser, enter your server's IP address to reveal Nginx's default landing page:



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

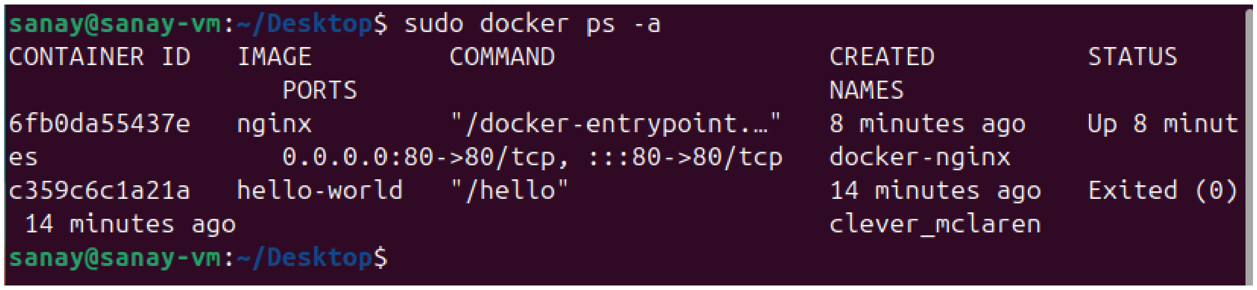
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Docker container status & removal :-

Run the following command in your terminal to view the Docker containers' status :-

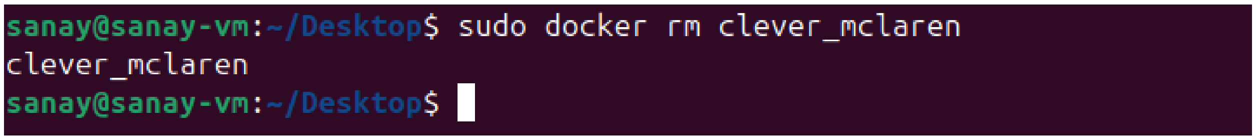
➢ `sudo docker ps -a`



```
sanay@sanay-vm:~/Desktop$ sudo docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS
                 PORTS                                NAMES
6fb0da55437e   nginx         "/docker-entrypoint.…"   8 minutes ago    Up 8 minut
es               0.0.0.0:80->80/tcp, :::80->80/tcp    docker-nginx
c359c6c1a21a   hello-world   "/hello"                 14 minutes ago   Exited (0)
 14 minutes ago                                        clever_mclaren
sanay@sanay-vm:~/Desktop$
```

Run the following command in your terminal to remove any Docker containers :-

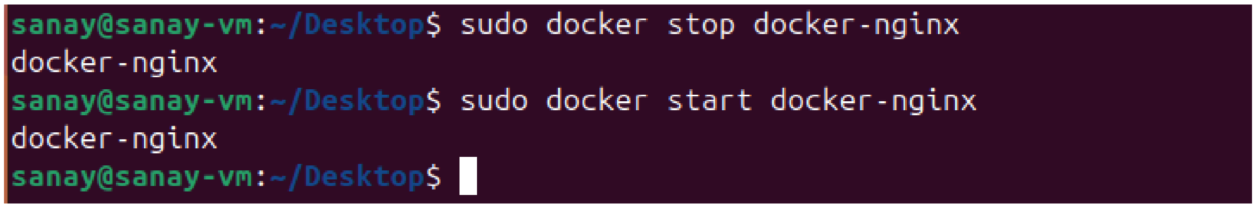
➢ `sudo docker rm docker-nginx`



```
sanay@sanay-vm:~/Desktop$ sudo docker rm clever_mclaren
clever_mclaren
sanay@sanay-vm:~/Desktop$
```

Apart from this, we also have the *start* and *stop* commands, which can be performed on any container :-

- ➢ `sudo docker start container-name`
- ➢ `sudo docker stop container-name`

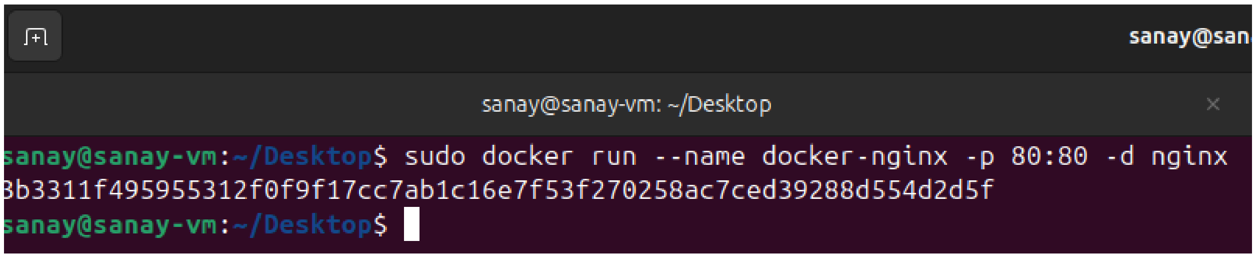


```
sanay@sanay-vm:~/Desktop$ sudo docker stop docker-nginx
docker-nginx
sanay@sanay-vm:~/Desktop$ sudo docker start docker-nginx
docker-nginx
sanay@sanay-vm:~/Desktop$
```

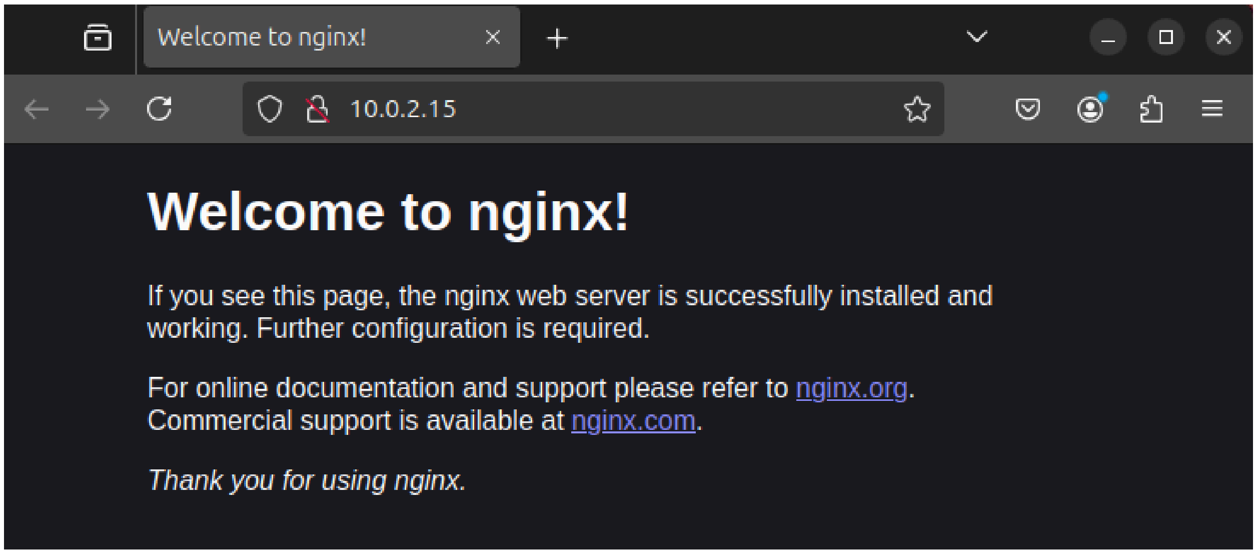
<u>Run Container in the background</u> :-

To run the Docker container in the background, we can use the *-d* flag.

- ➢ `sudo docker run --name docker-nginx -p 80:80 -d nginx`



```
anay@sanay-vm:~/Desktop$ sudo docker run --name docker-nginx -p 80:80 -d nginx
3b3311f495955312f0f9f17cc7ab1c16e7f53f270258ac7ced39288d554d2d5f
anay@sanay-vm:~/Desktop$
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

<u>Building a custom webpage to serve it on NGINX (linking the path)</u> :-

Create a new directory for your website content within the home directory:

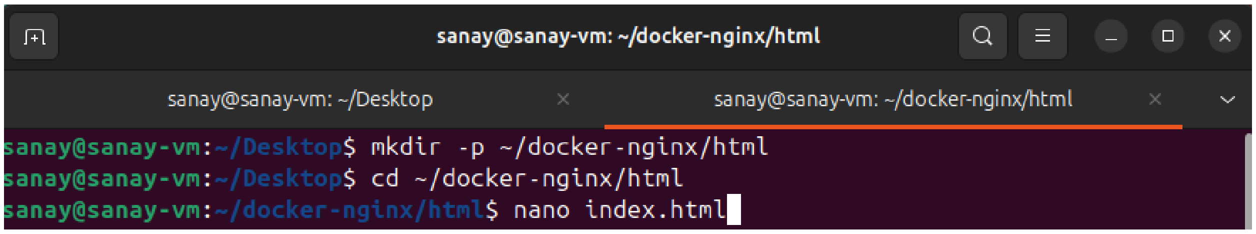
➢ `mkdir -p ~/docker-nginx/html`

Move inside the directory:

➢ `cd ~/docker-nginx/html`

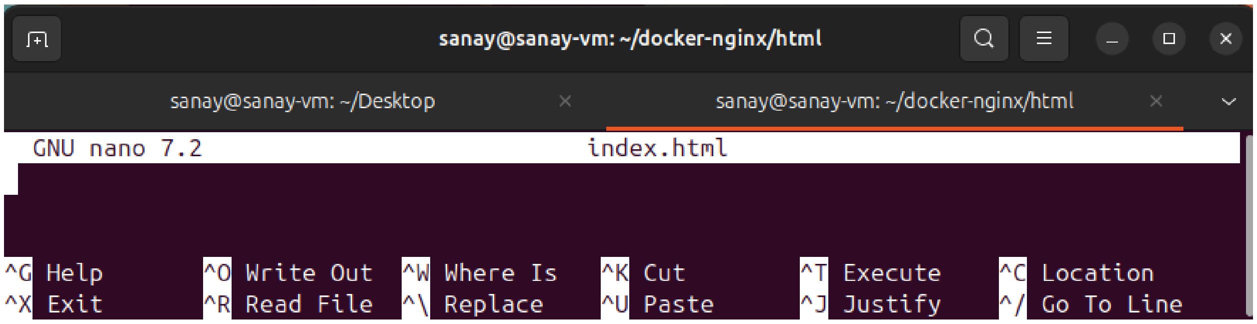
Create an HTML file within your serving directory. We are using *nano* text-editor to create it:

➢ `nano index.html`


sanay@sanay-vm: ~/docker-nginx/html
sanay@sanay-vm: ~/Desktop
sanay@sanay-vm: ~/docker-nginx/html

```
sanay@sanay-vm:~/Desktop$ mkdir -p ~/docker-nginx/html
sanay@sanay-vm:~/Desktop$ cd ~/docker-nginx/html
sanay@sanay-vm:~/docker-nginx/html$ nano index.html
```

Now, paste your HTML code over here:


sanay@sanay-vm: ~/docker-nginx/html
sanay@sanay-vm: ~/Desktop
sanay@sanay-vm: ~/docker-nginx/html

```
 GNU nano 7.2                            index.html




^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line
```

Here’s a sample HTML page:

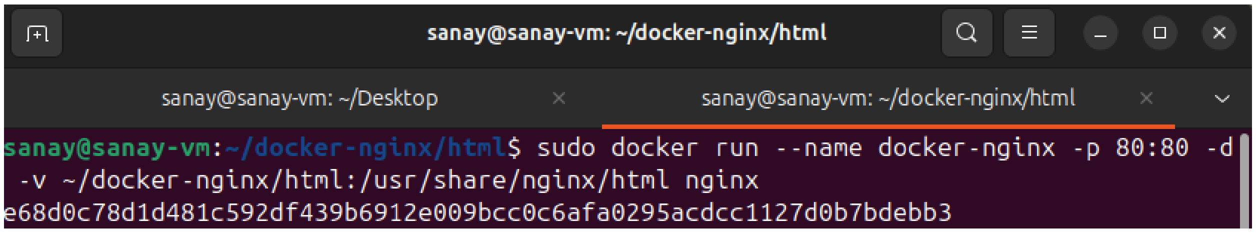
```
<html>
  <head>
    <title>Docker NGINX Tutorial</title>
  </head>
  <body>
    <div class="container">
      <h1>Website made by Sarthak Sanay</h1>
      <p>This Nginx service is running within a Docker container!</p>
      <p>This website was created for the tutorial of Docker
integration with NGINX!</p>
    </div>
  </body>
</html>
```

Linking the Container to the Local Filesystem:

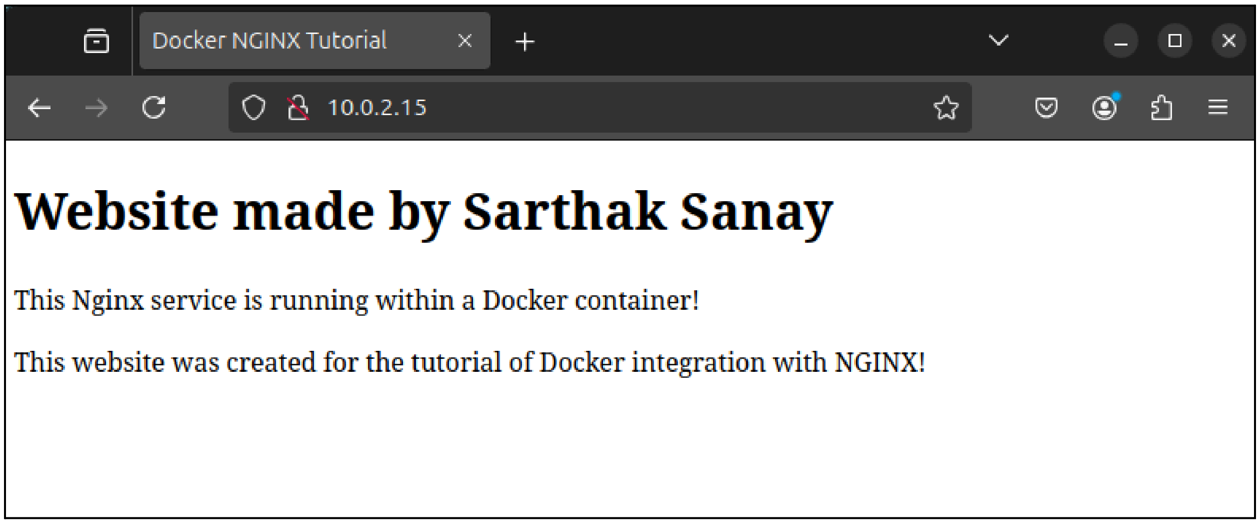
➢
```
sudo docker run --name docker-nginx -p 80:80 -d -v
~/docker-nginx/html:/usr/share/nginx/html nginx
```

Here’s a brief explanation of the command:-

- Use the -v flag to map the ~/docker-nginx/html folder from your server to a relative path in the container/usr/share/nginx/html
- -v flag specifies that you’re linking a volume.
- To the left of the : is the location of your directory on your server, ~/docker-nginx/html.
- To the right of the : is the location that you are symbolically linking to your container /usr/share/nginx/html.


sanay@sanay-vm: ~/docker-nginx/html
sanay@sanay-vm: ~/Desktop
sanay@sanay-vm: ~/docker-nginx/html

```
sanay@sanay-vm:~/docker-nginx/html$ sudo docker run --name docker-nginx -p 80:80 -d
 -v ~/docker-nginx/html:/usr/share/nginx/html nginx
e68d0c78d1d481c592df439b6912e009bcc0c6afa0295acdcc1127d0b7bdebb3
```



Website made by Sarthak Sanay

This Nginx service is running within a Docker container!

This website was created for the tutorial of Docker integration with NGINX!

<u>PRACTICAL</u> - <u>5</u>

<u>AIM</u>: Analyze Log Files in Linux using custom Shell Script

<u>About Log Files</u> :-

Log files are records of events, activities, and errors generated by systems, applications, or devices. They help monitor operations, diagnose issues, and ensure security. Log files typically include timestamps, system messages, error codes, and user actions. They are crucial for troubleshooting, performance analysis, and auditing. Stored in various formats, log data can be accessed using specialized tools to track system health, detect security breaches, and maintain operational efficiency. Proper log management is essential for system administration.

<u>Common Log File Formats in Linux</u> :-

1. **Plain Text Logs:** Most log files are stored in plain text format, which is easy to read and process.
2. **JSON Logs:** Some modern applications use JSON (JavaScript Object Notation) format for structured logging, making it easier to parse and analyze.
3. **Binary Logs:** Some logs are stored in binary format for efficiency and are not human-readable without specific tools.

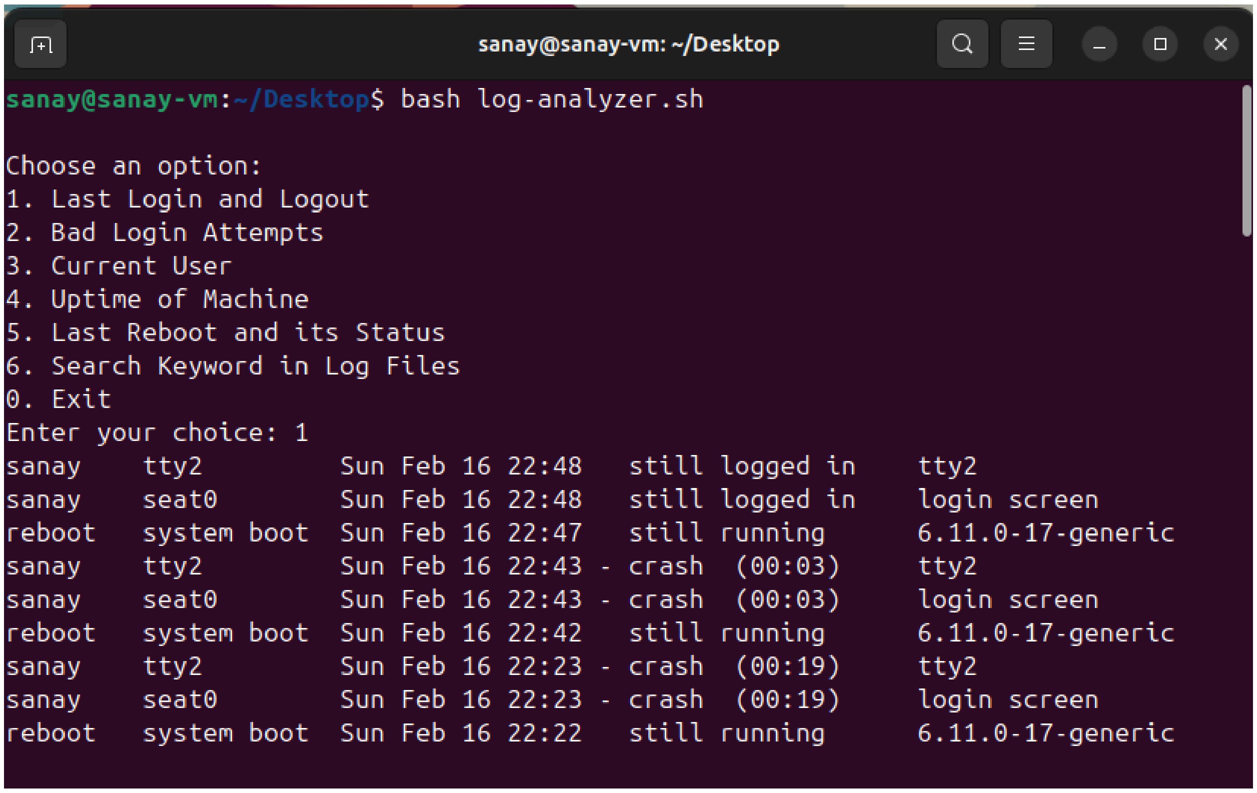
These formats can be analyzed using *cat*, *grep*, *journalctl*, or *less*.

Shell Script to Analyze Log Files in Linux :-

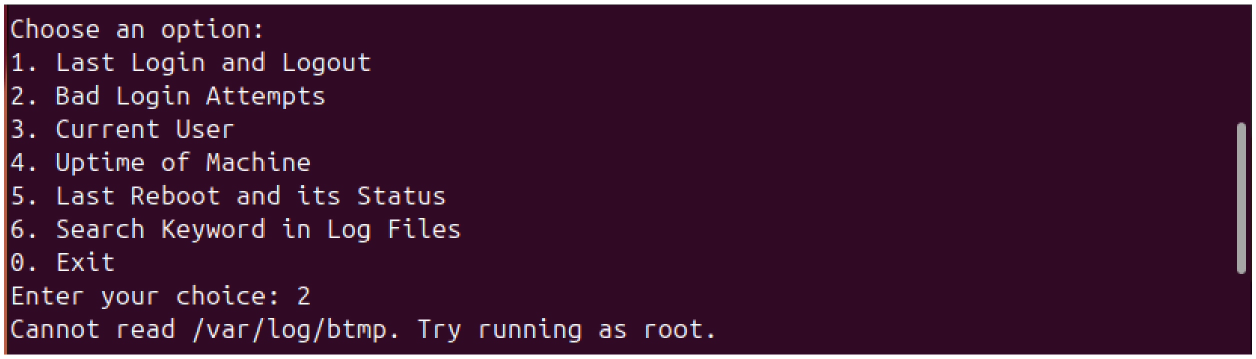
```
#!/bin/bash
log_file="/var/log/auth.log"  # Default log file
btmp_file="/var/log/btmp"  # Bad login attempts log file

display_last_login_logout() {
    last -a | head -10
}
display_bad_logins() {
    if [[ -r "$btmp_file" ]]; then
        lastb | head -10
    else
        echo "Cannot read $btmp_file. Try running as root."
    fi
}
display_current_user() {
    whoami
}
display_uptime() {
    uptime -p
}
display_last_reboot() {
    last reboot | head -1
}
# Function to search keyword within log files
process_log_file() {
    echo "Filtering logs for: $1"
    grep "$1" "$log_file" | less
}
```

```
menu() {
    while true; do
        echo -e "\nChoose an option:"
        echo "1. Last Login and Logout"
        echo "2. Bad Login Attempts"
        echo "3. Current User"
        echo "4. Uptime of Machine"
        echo "5. Last Reboot and its Status"
        echo "6. Search Keyword in Log Files"
        echo "0. Exit"
        read -p "Enter your choice: " choice
        case $choice in
            1) display_last_login_logout ;;
            2) display_bad_logins ;;
            3) display_current_user ;;
            4) display_uptime ;;
            5) display_last_reboot ;;
            6) read -p "Enter search keyword: " keyword
               process_log_file "$keyword" ;;
            0) echo -e "\nExited the program successfully!"
               exit 0 ;;
            *) echo "Invalid choice, please try again." ;;
        esac
    done
}

# Running the main function
menu
```

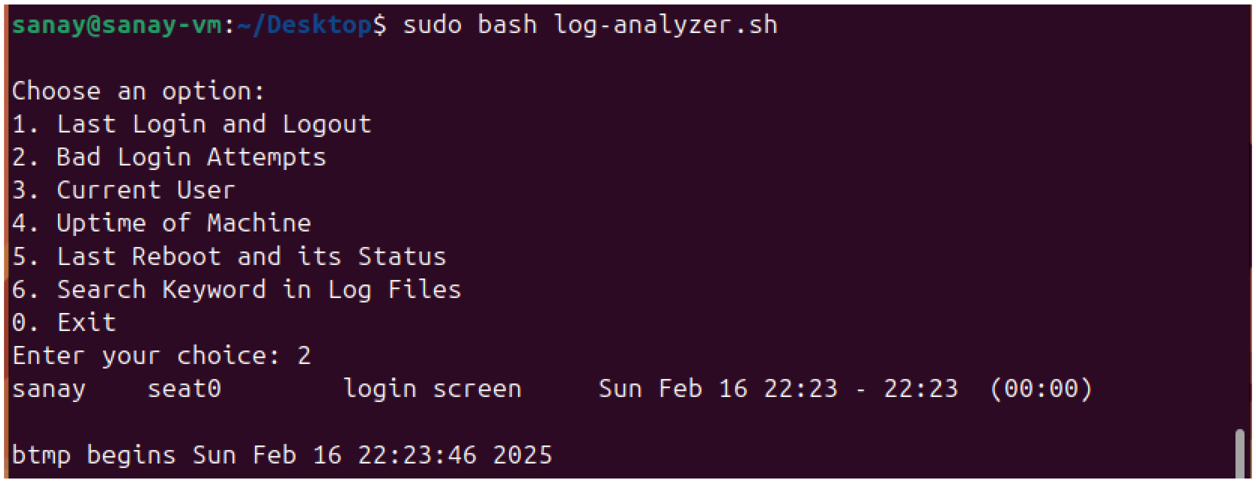
Output :-


sanay@sanay-vm: ~/Desktop

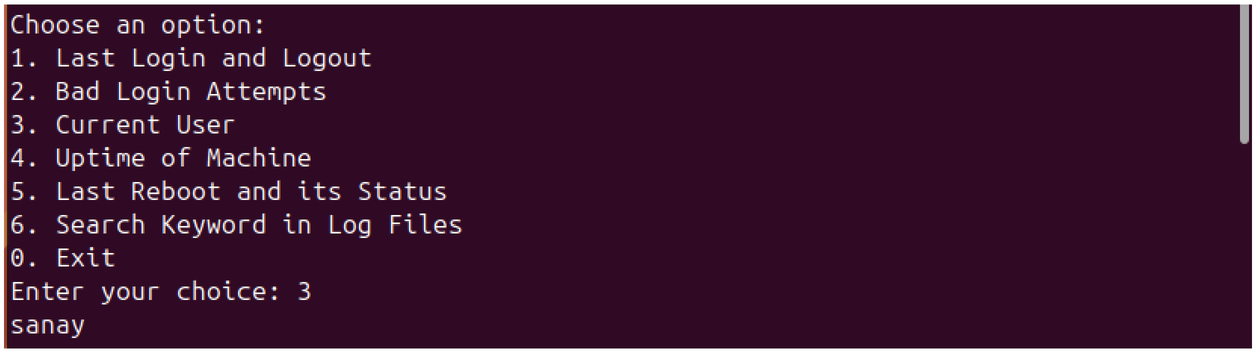
```
sanay@sanay-vm:~/Desktop$ bash log-analyzer.sh

Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 1
sanay    tty2         Sun Feb 16 22:48   still logged in    tty2
sanay    seat0        Sun Feb 16 22:48   still logged in    login screen
reboot   system boot  Sun Feb 16 22:47   still running      6.11.0-17-generic
sanay    tty2         Sun Feb 16 22:43 - crash  (00:03)     tty2
sanay    seat0        Sun Feb 16 22:43 - crash  (00:03)     login screen
reboot   system boot  Sun Feb 16 22:42   still running      6.11.0-17-generic
sanay    tty2         Sun Feb 16 22:23 - crash  (00:19)     tty2
sanay    seat0        Sun Feb 16 22:23 - crash  (00:19)     login screen
reboot   system boot  Sun Feb 16 22:22   still running      6.11.0-17-generic
```



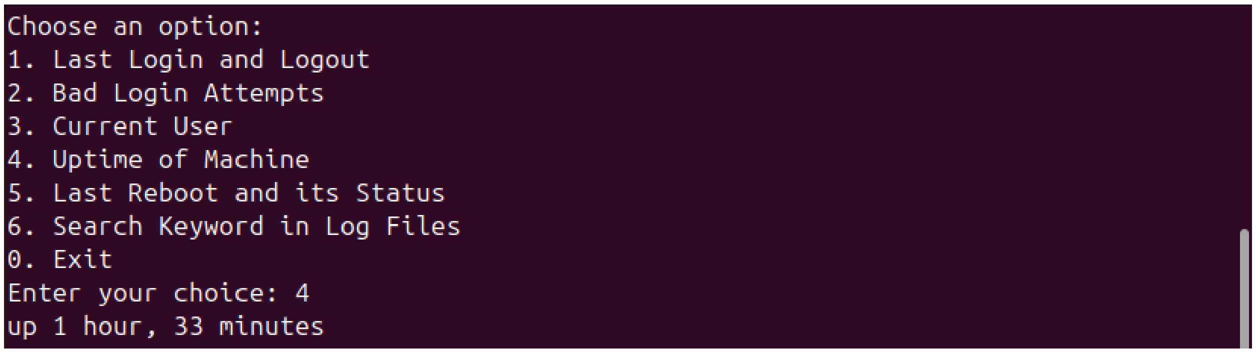
```
Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 2
Cannot read /var/log/btmp. Try running as root.
```



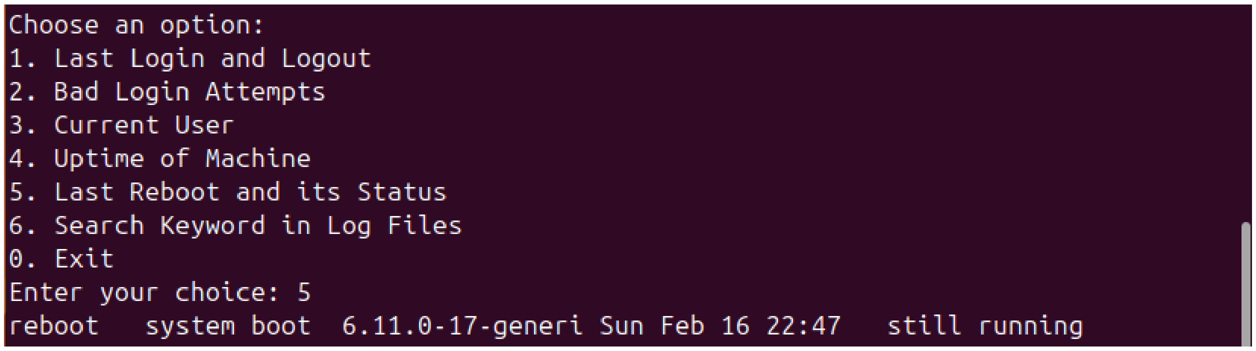
```
sanay@sanay-vm:~/Desktop$ sudo bash log-analyzer.sh

Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 2
sanay    seat0        login screen     Sun Feb 16 22:23 - 22:23  (00:00)

btmp begins Sun Feb 16 22:23:46 2025
```



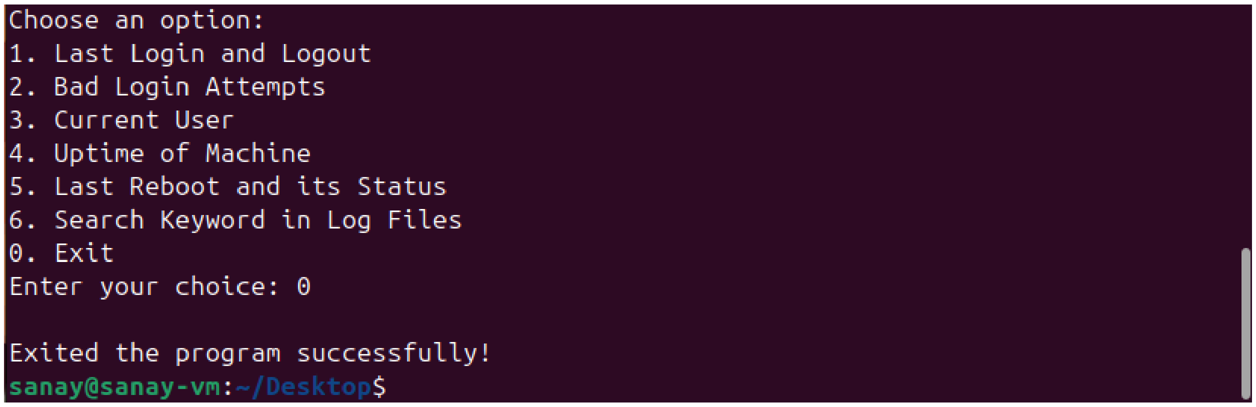
```
Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 3
sanay
```



```
Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 4
up 1 hour, 33 minutes
```



```
Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 5
reboot   system boot  6.11.0-17-generi Sun Feb 16 22:47   still running
```



```
Choose an option:
1. Last Login and Logout
2. Bad Login Attempts
3. Current User
4. Uptime of Machine
5. Last Reboot and its Status
6. Search Keyword in Log Files
0. Exit
Enter your choice: 0

Exited the program successfully!
sanay@sanay-vm:~/Desktop$
```

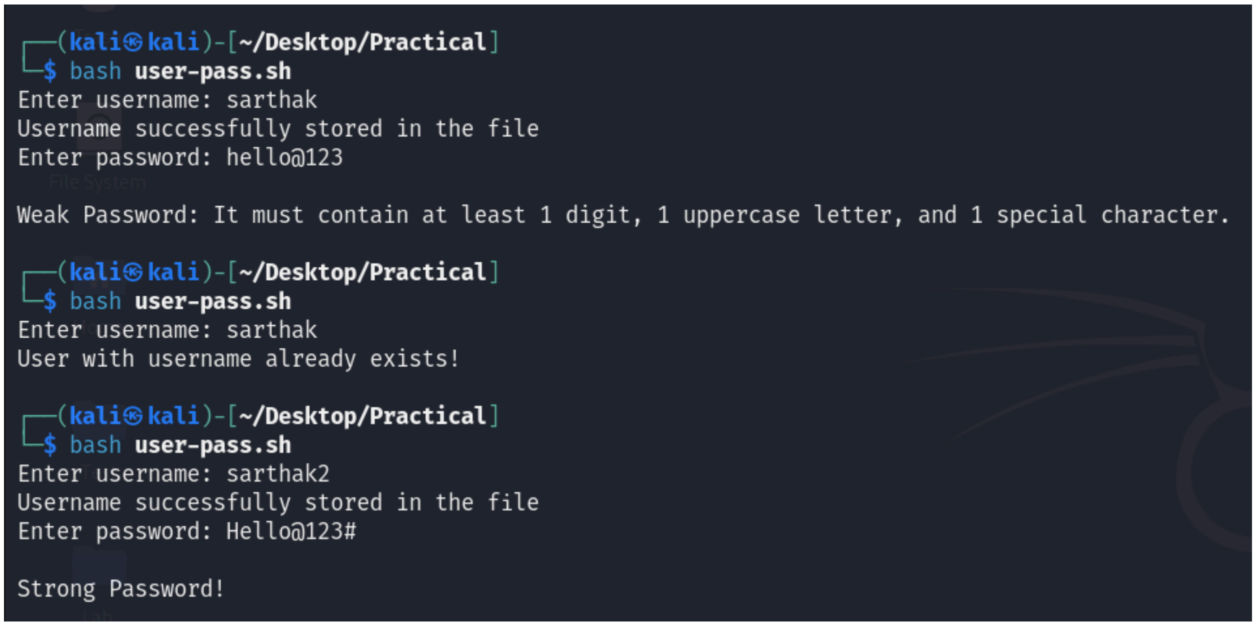
PRACTICAL - 6

AIM: Write a script to check the strength of a password entered by the user.

```
#!/bin/bash
output="users.txt"
read -p "Enter username: " username

if [[ "$username" =~ ^[^a-zA-Z0-9] ]]; then
    echo "Invalid! Username cannot start with a special character"
    exit 1
elif grep -q "^$username$" "$output"; then
    echo "User with username already exists!"
    exit 1
fi

echo "$username" >> "$output"
echo "Username successfully stored in the file"
read -p "Enter password: " password
echo

if [[ ${#password} -lt 8 ]]; then
    echo "Weak Password: It must be at least 8 characters long."
    exit 1
fi
```

```
if ! [[ "$password" =~ [0-9] && "$password" =~ [A-Z] && "$password"
=~ [[:punct:]] ]]; then

    echo "Weak Password: It must contain at least 1 digit, 1
uppercase letter, and 1 special character."

    exit 1

fi


echo "Strong Password!"
```

<u>Output</u> :-



```
┌──(kali㉿kali)-[~/Desktop/Practical]
└─$ bash user-pass.sh
Enter username: sarthak
Username successfully stored in the file
Enter password: hello@123

Weak Password: It must contain at least 1 digit, 1 uppercase letter, and 1 special character.

┌──(kali㉿kali)-[~/Desktop/Practical]
└─$ bash user-pass.sh
Enter username: sarthak
User with username already exists!

┌──(kali㉿kali)-[~/Desktop/Practical]
└─$ bash user-pass.sh
Enter username: sarthak2
Username successfully stored in the file
Enter password: Hello@123#

Strong Password!
```

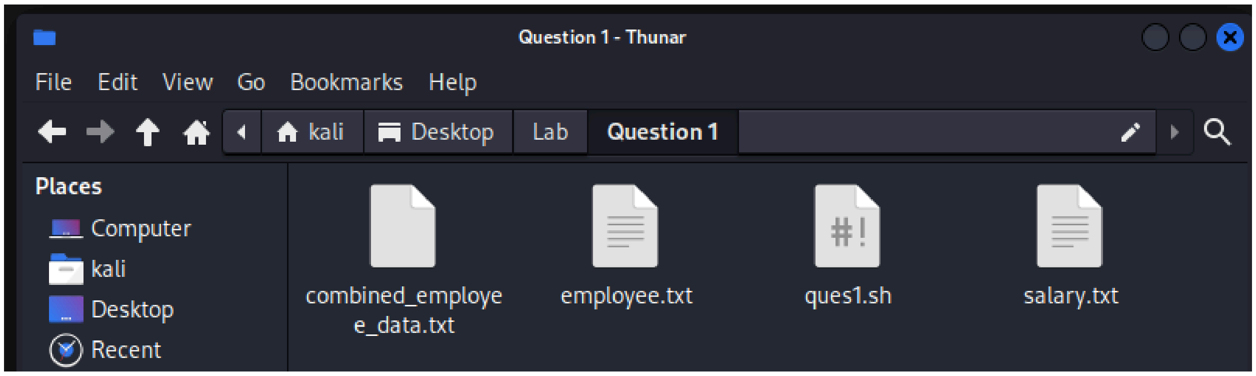
PRACTICAL - 7

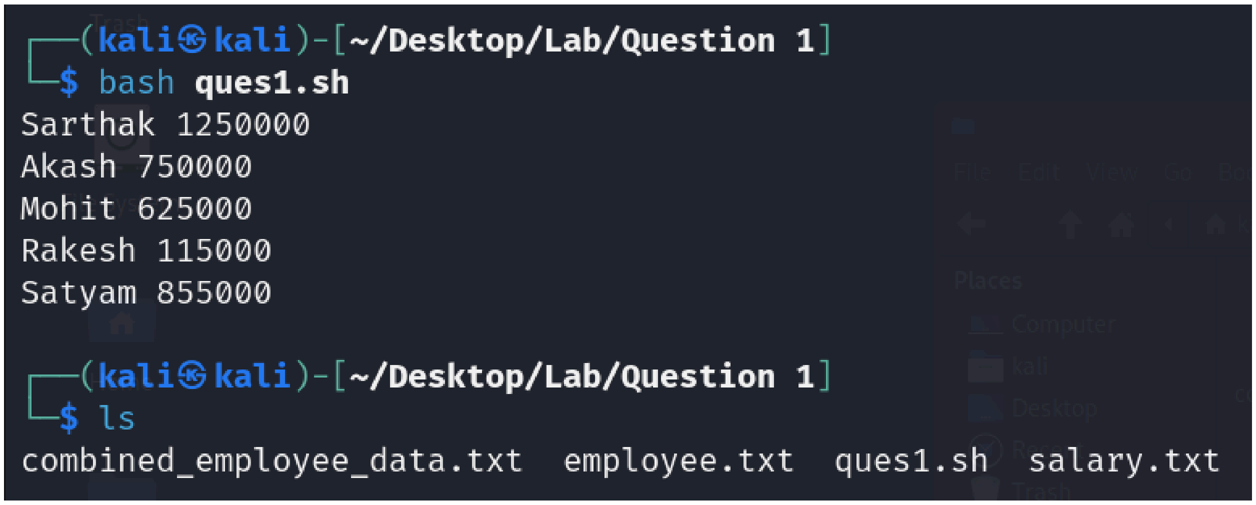
AIM: Create two files. (One file having the name of the employee as content, and the second file having the salary of the employee). Write a script to combine the data of both the files into a single file.

Bash Script :-

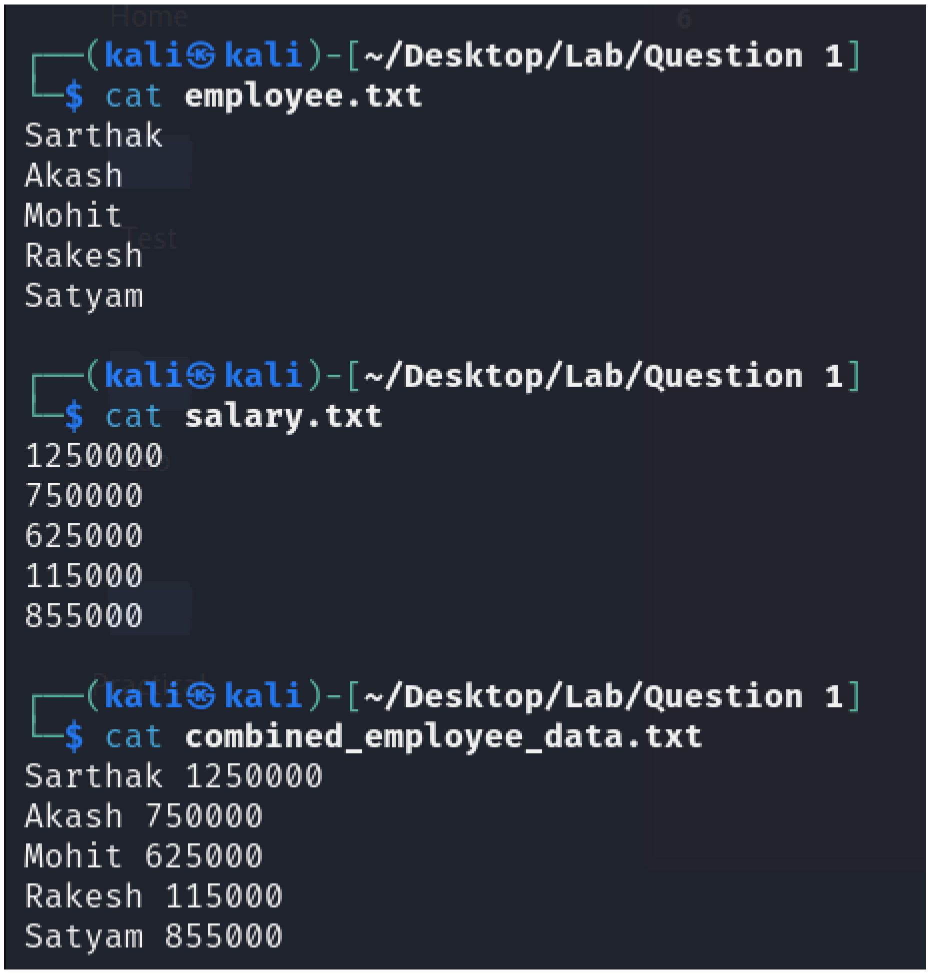
```
#!/bin/bash
paste -d ' ' employee.txt salary.txt > combined_employee_data.txt
# join employee.txt salary.txt > combined_employee_data.txt
cat combined_employee_data.txt
echo "Combined employee and salary data stored in $output_file"
```

Output :-





```
┌──(kali㉿kali)-[~/Desktop/Lab/Question 1]
└─$ bash ques1.sh
Sarthak 1250000
Akash 750000
Mohit 625000
Rakesh 115000
Satyam 855000

┌──(kali㉿kali)-[~/Desktop/Lab/Question 1]
└─$ ls
combined_employee_data.txt  employee.txt  ques1.sh  salary.txt
```



```
┌──(kali㉿kali)-[~/Desktop/Lab/Question 1]
└─$ cat employee.txt
Sarthak
Akash
Mohit
Rakesh
Satyam

┌──(kali㉿kali)-[~/Desktop/Lab/Question 1]
└─$ cat salary.txt
1250000
750000
625000
115000
855000

┌──(kali㉿kali)-[~/Desktop/Lab/Question 1]
└─$ cat combined_employee_data.txt
Sarthak 1250000
Akash 750000
Mohit 625000
Rakesh 115000
Satyam 855000
```

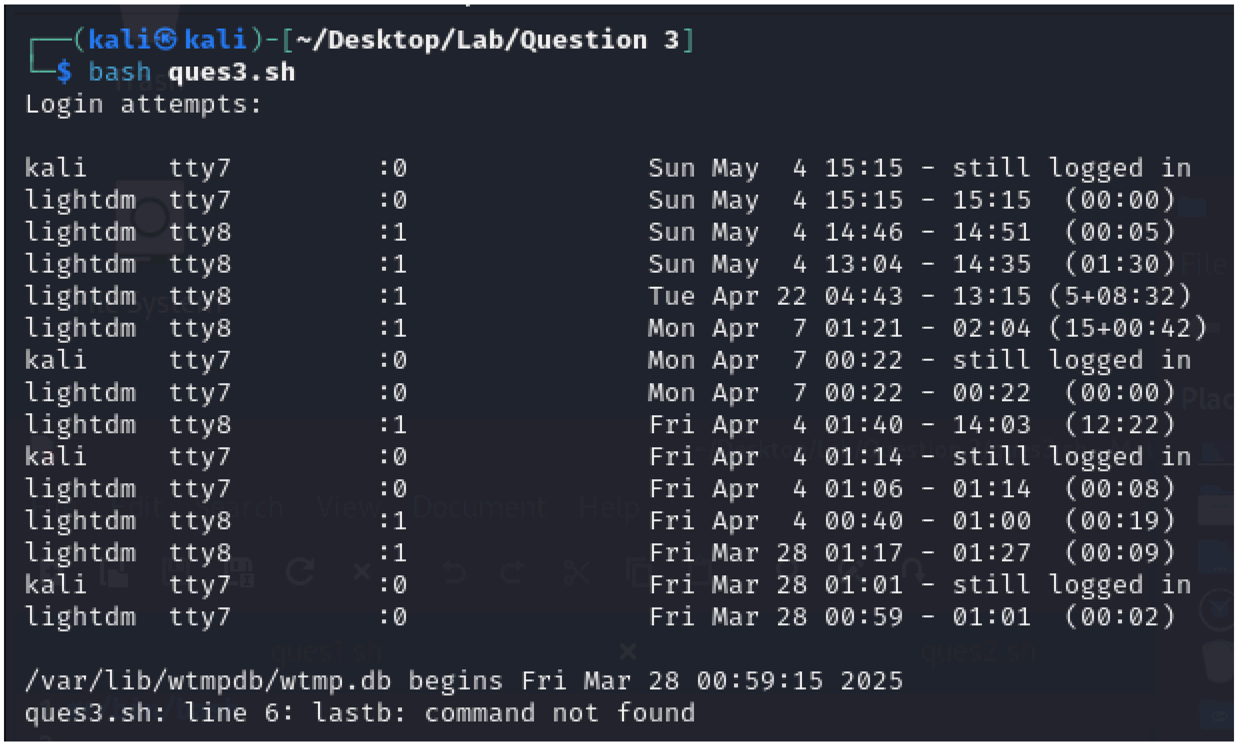
<u>PRACTICAL</u> - <u>8</u>

<u>AIM</u>: Write a script to identify the logs of the system, and search password file, login details and no. of users from the logs.

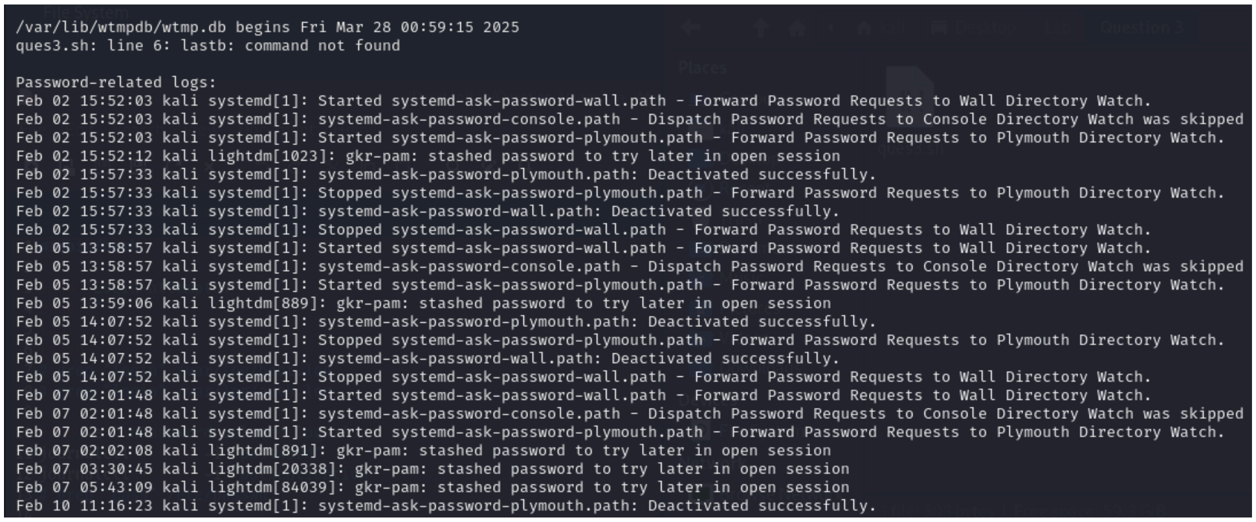
<u>Bash Script</u> :-

```
#!/bin/bash

echo "Login attempts:"
echo ""
last
lastb
# grep "login" /var/log/boot.log
# grep "login" /var/log/auth.log

echo -e "\nPassword-related logs:"
journalctl | grep -i "password"
journalctl | grep -i "authentication"
# grep "pass" /etc/passwd

echo -e "\nTotal users:"
cat /etc/passwd | wc -l
```

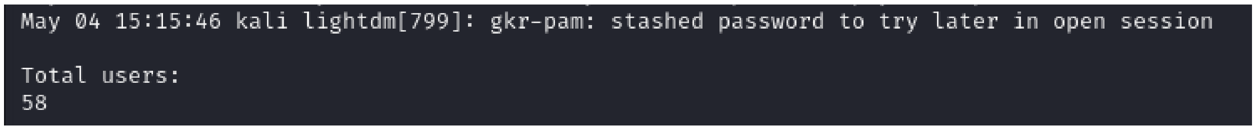
<u>Output</u> :-



```
┌──(kali㉿kali)-[~/Desktop/Lab/Question 3]
└─$ bash ques3.sh
Login attempts:

kali     tty7         :0               Sun May  4 15:15 - still logged in
lightdm  tty7         :0               Sun May  4 15:15 - 15:15  (00:00)
lightdm  tty8         :1               Sun May  4 14:46 - 14:51  (00:05)
lightdm  tty8         :1               Sun May  4 13:04 - 14:35  (01:30)
lightdm  tty8         :1               Tue Apr 22 04:43 - 13:15 (5+08:32)
lightdm  tty8         :1               Mon Apr  7 01:21 - 02:04 (15+00:42)
kali     tty7         :0               Mon Apr  7 00:22 - still logged in
lightdm  tty7         :0               Mon Apr  7 00:22 - 00:22  (00:00)
lightdm  tty8         :1               Fri Apr  4 01:40 - 14:03  (12:22)
kali     tty7         :0               Fri Apr  4 01:14 - still logged in
lightdm  tty7         :0               Fri Apr  4 01:06 - 01:14  (00:08)
lightdm  tty8         :1               Fri Apr  4 00:40 - 01:00  (00:19)
lightdm  tty8         :1               Fri Mar 28 01:17 - 01:27  (00:09)
kali     tty7         :0               Fri Mar 28 01:01 - still logged in
lightdm  tty7         :0               Fri Mar 28 00:59 - 01:01  (00:02)

/var/lib/wtmpdb/wtmp.db begins Fri Mar 28 00:59:15 2025
ques3.sh: line 6: lastb: command not found
```



```
/var/lib/wtmpdb/wtmp.db begins Fri Mar 28 00:59:15 2025
ques3.sh: line 6: lastb: command not found

Password-related logs:
Feb 02 15:52:03 kali systemd[1]: Started systemd-ask-password-wall.path - Forward Password Requests to Wall Directory Watch.
Feb 02 15:52:03 kali systemd[1]: systemd-ask-password-console.path - Dispatch Password Requests to Console Directory Watch was skipped
Feb 02 15:52:03 kali systemd[1]: Started systemd-ask-password-plymouth.path - Forward Password Requests to Plymouth Directory Watch.
Feb 02 15:52:12 kali lightdm[1023]: gkr-pam: stashed password to try later in open session
Feb 02 15:57:33 kali systemd[1]: systemd-ask-password-plymouth.path: Deactivated successfully.
Feb 02 15:57:33 kali systemd[1]: Stopped systemd-ask-password-plymouth.path - Forward Password Requests to Plymouth Directory Watch.
Feb 02 15:57:33 kali systemd[1]: systemd-ask-password-wall.path: Deactivated successfully.
Feb 02 15:57:33 kali systemd[1]: Stopped systemd-ask-password-wall.path - Forward Password Requests to Wall Directory Watch.
Feb 05 13:58:57 kali systemd[1]: Started systemd-ask-password-wall.path - Forward Password Requests to Wall Directory Watch.
Feb 05 13:58:57 kali systemd[1]: systemd-ask-password-console.path - Dispatch Password Requests to Console Directory Watch was skipped
Feb 05 13:58:57 kali systemd[1]: Started systemd-ask-password-plymouth.path - Forward Password Requests to Plymouth Directory Watch.
Feb 05 13:59:06 kali lightdm[889]: gkr-pam: stashed password to try later in open session
Feb 05 14:07:52 kali systemd[1]: systemd-ask-password-plymouth.path: Deactivated successfully.
Feb 05 14:07:52 kali systemd[1]: Stopped systemd-ask-password-plymouth.path - Forward Password Requests to Plymouth Directory Watch.
Feb 05 14:07:52 kali systemd[1]: systemd-ask-password-wall.path: Deactivated successfully.
Feb 05 14:07:52 kali systemd[1]: Stopped systemd-ask-password-wall.path - Forward Password Requests to Wall Directory Watch.
Feb 07 02:01:48 kali systemd[1]: Started systemd-ask-password-wall.path - Forward Password Requests to Wall Directory Watch.
Feb 07 02:01:48 kali systemd[1]: systemd-ask-password-console.path - Dispatch Password Requests to Console Directory Watch was skipped
Feb 07 02:01:48 kali systemd[1]: Started systemd-ask-password-plymouth.path - Forward Password Requests to Plymouth Directory Watch.
Feb 07 02:02:08 kali lightdm[891]: gkr-pam: stashed password to try later in open session
Feb 07 03:30:45 kali lightdm[20338]: gkr-pam: stashed password to try later in open session
Feb 07 05:43:09 kali lightdm[84039]: gkr-pam: stashed password to try later in open session
Feb 10 11:16:23 kali systemd[1]: systemd-ask-password-plymouth.path: Deactivated successfully.
```



```
May 04 15:15:46 kali lightdm[799]: gkr-pam: stashed password to try later in open session

Total users:
58
```

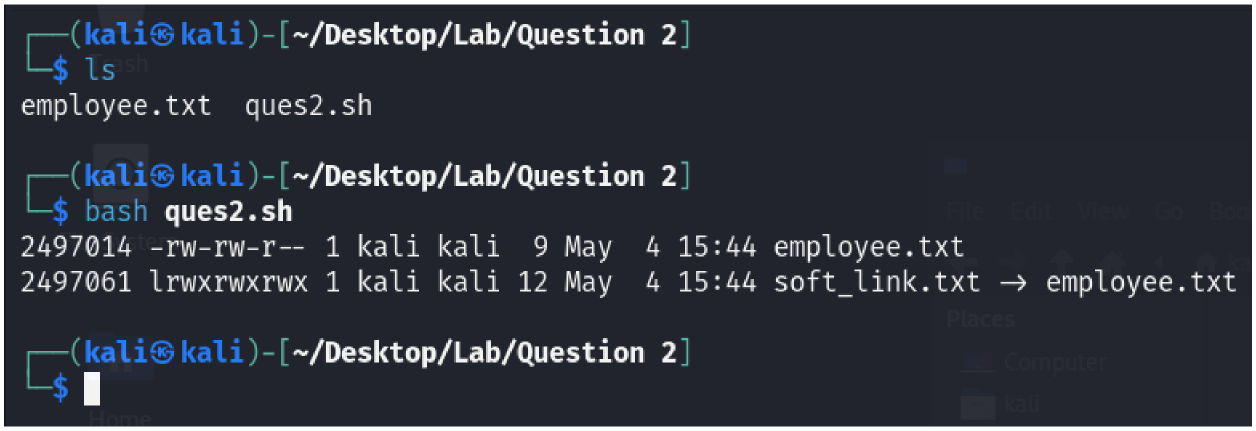
PRACTICAL - 9

AIM: Write a script to create a soft and hard link of the existing file. Then delete the soft link of the file.

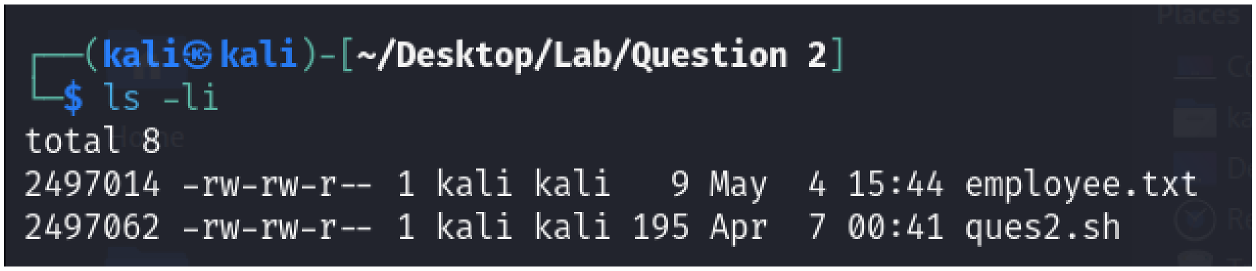
Bash Script :-

```
#!/bin/bash
echo "Employee" > employee.txt
ln -s employee.txt soft_link.txt
# ln employee.txt hard_link.txt
ls -li employee.txt soft_link.txt
rm soft_link.txt
# ls -li employee.txt hard_link.txt
```

Output :-



```
┌──(kali㉿kali)-[~/Desktop/Lab/Question 2]
└─$ ls
employee.txt  ques2.sh

┌──(kali㉿kali)-[~/Desktop/Lab/Question 2]
└─$ bash ques2.sh
2497014 -rw-rw-r-- 1 kali kali  9 May  4 15:44 employee.txt
2497061 lrwxrwxrwx 1 kali kali 12 May  4 15:44 soft_link.txt → employee.txt

┌──(kali㉿kali)-[~/Desktop/Lab/Question 2]
└─$ 
```



```
┌──(kali㉿kali)-[~/Desktop/Lab/Question 2]
└─$ ls -li
total 8
2497014 -rw-rw-r-- 1 kali kali   9 May  4 15:44 employee.txt
2497062 -rw-rw-r-- 1 kali kali 195 Apr  7 00:41 ques2.sh
```

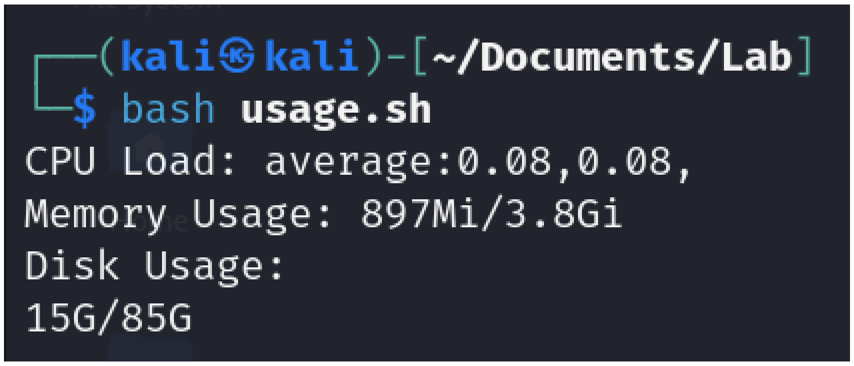
PRACTICAL - 10

AIM: Write a shell script to display CPU usage, memory usage, and disk space usage.

Bash Script :-

```
#!/bin/bash

echo "CPU Load:" $(top -bn1 | awk '/load average/ {print $10
$11 $12}')

echo "Memory Usage:" $(free -h | awk '/Mem:/ {print $3 "/"
$2}')

echo "Disk Usage:" && df -h --total | awk '$1=="total" {print
$3 "/" $2}'
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash usage.sh
CPU Load: average:0.08,0.08,
Memory Usage: 897Mi/3.8Gi
Disk Usage:
15G/85G
```

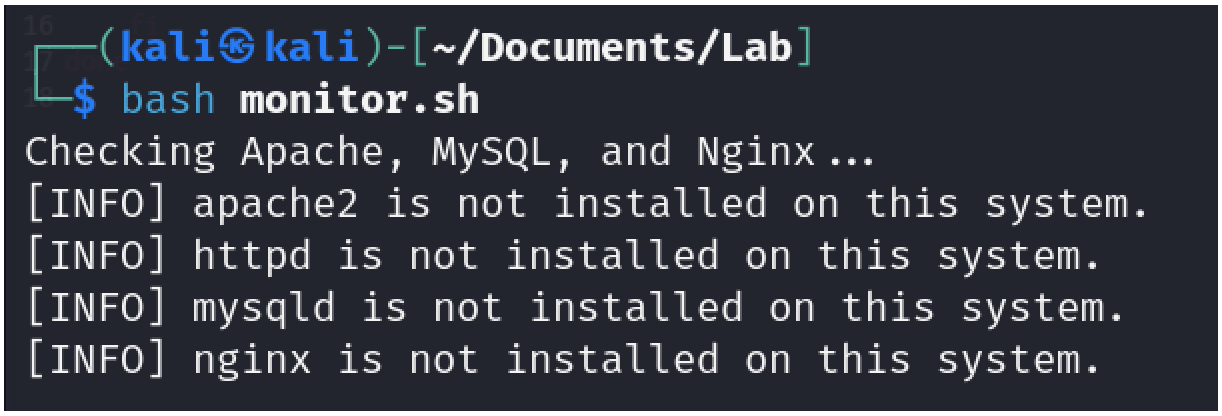
PRACTICAL - 11

AIM: Create a script to monitor a running process (e.g., Apache, MySQL) and notify if it stops.

Bash Script :-

```
#!/bin/bash

services=("apache2" "httpd" "mysqld" "nginx")

echo "Checking Apache, MySQL, and Nginx."


for service in "${services[@]}"; do

    if systemctl list-units --type=service --all | grep -q
"$service"; then

        if systemctl is-active --quiet "$service"; then

            echo "[OK] $service is running."

        else

            echo "[ALERT] $service is NOT running!"

        fi

    else

        echo "[INFO] $service is not installed on this
system."

    fi

done
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash monitor.sh
Checking Apache, MySQL, and Nginx...
[INFO] apache2 is not installed on this system.
[INFO] httpd is not installed on this system.
[INFO] mysqld is not installed on this system.
[INFO] nginx is not installed on this system.
```

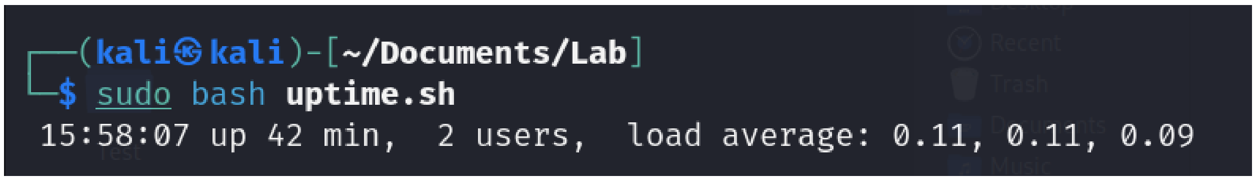
PRACTICAL - 12

AIM: Write a shell script to log system uptime and the number of logged-in users every 5 minutes.

Bash Script :-

```
#!/bin/bash
uptime | tee -a /var/log/sys_stats.log
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo bash uptime.sh
 15:58:07 up 42 min,  2 users,  load average: 0.11, 0.11, 0.09
```

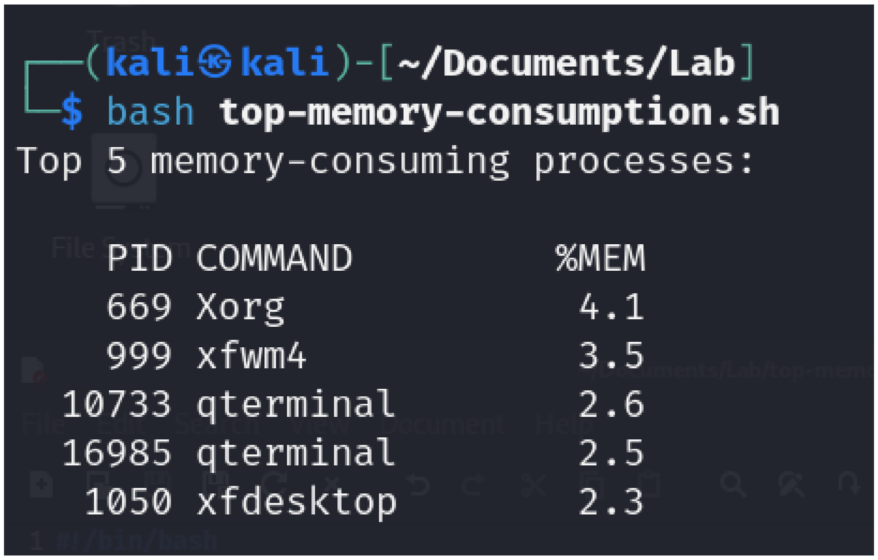
PRACTICAL - 13

AIM: Write a script that checks the top 5 memory-consuming processes.

Bash Script :-

```
#!/bin/bash
echo "Top 5 memory-consuming processes:"
echo
ps -eo pid,comm,%mem --sort=-%mem | head -n 6
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash top-memory-consumption.sh
Top 5 memory-consuming processes:

    PID COMMAND         %MEM
    669 Xorg             4.1
    999 xfwm4            3.5
  10733 qterminal        2.6
  16985 qterminal        2.5
   1050 xfdesktop        2.3
```

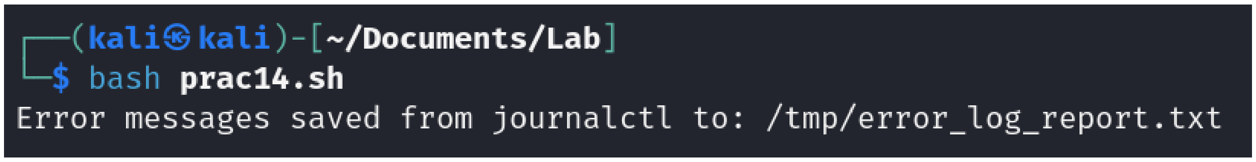
PRACTICAL - 14

AIM: Provide an example of a script that extracts error messages from /var/log/syslog.

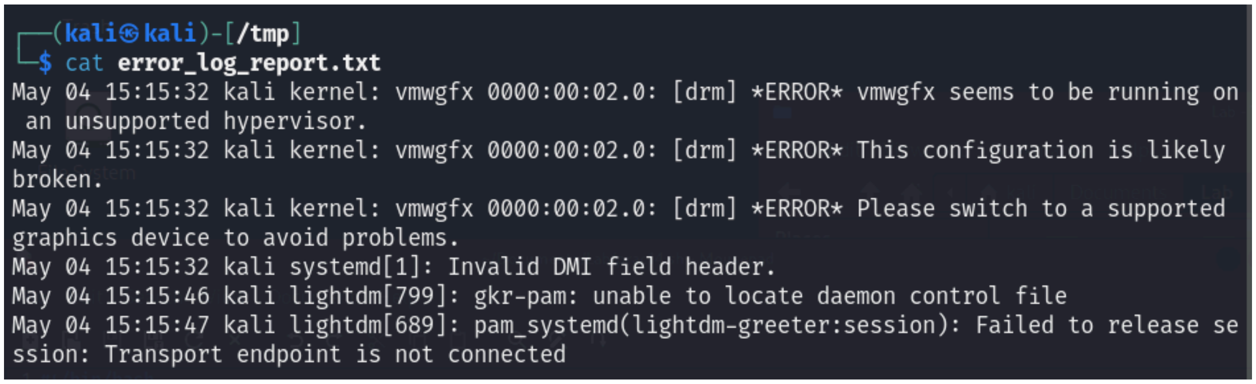
Bash Script :-

```
#!/bin/bash
# Script to Extract ERROR messages using journalctl
OUTPUT="/tmp/error_log_report.txt"
journalctl -p err -b > "$OUTPUT"
echo "Error messages saved from journalctl to: $OUTPUT"
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash prac14.sh
Error messages saved from journalctl to: /tmp/error_log_report.txt
```



```
┌──(kali㉿kali)-[/tmp]
└─$ cat error_log_report.txt
May 04 15:15:32 kali kernel: vmwgfx 0000:00:02.0: [drm] *ERROR* vmwgfx seems to be running on
 an unsupported hypervisor.
May 04 15:15:32 kali kernel: vmwgfx 0000:00:02.0: [drm] *ERROR* This configuration is likely
broken.
May 04 15:15:32 kali kernel: vmwgfx 0000:00:02.0: [drm] *ERROR* Please switch to a supported
graphics device to avoid problems.
May 04 15:15:32 kali systemd[1]: Invalid DMI field header.
May 04 15:15:46 kali lightdm[799]: gkr-pam: unable to locate daemon control file
May 04 15:15:47 kali lightdm[689]: pam_systemd(lightdm-greeter:session): Failed to release se
ssion: Transport endpoint is not connected
```

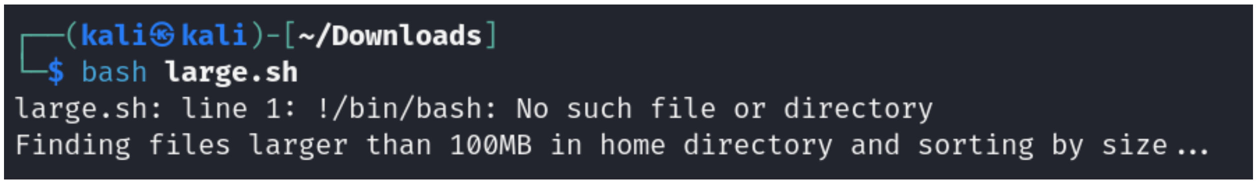
PRACTICAL - 15

AIM: Write a script to find and list all files larger than 100MB in the home directory.

Bash Script :-

```
#!/bin/bash
# Script to Extract ERROR messages using journalctl
OUTPUT="/tmp/error_log_report.txt"
journalctl -p err -b > "$OUTPUT"
echo "Error messages saved from journalctl to: $OUTPUT"
```

Output :-



```
┌──(kali㉿kali)-[~/Downloads]
└─$ bash large.sh
large.sh: line 1: !/bin/bash: No such file or directory
Finding files larger than 100MB in home directory and sorting by size...
```

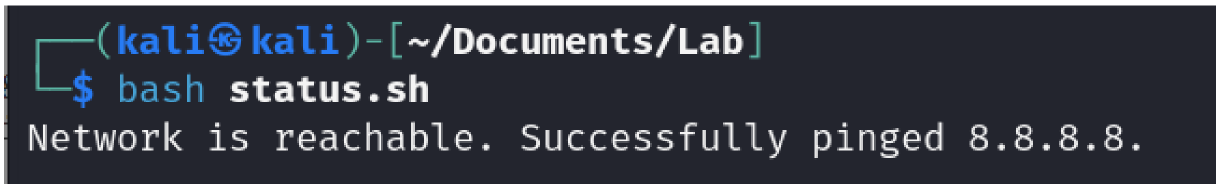
PRACTICAL - 16

AIM: Create a shell script to check the status of network connectivity (e.g., ping to a server).

Bash Script :-

```
#!/bin/bash
HOST="8.8.8.8"  # Google's public DNS server
ping -c 4 $HOST > /dev/null 2>&1
if [ $? -eq 0 ]; then
  echo "Network is reachable. Successfully pinged $HOST."
else
  echo "Network is unreachable. Failed to ping $HOST."
fi
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash status.sh
Network is reachable. Successfully pinged 8.8.8.8.
```

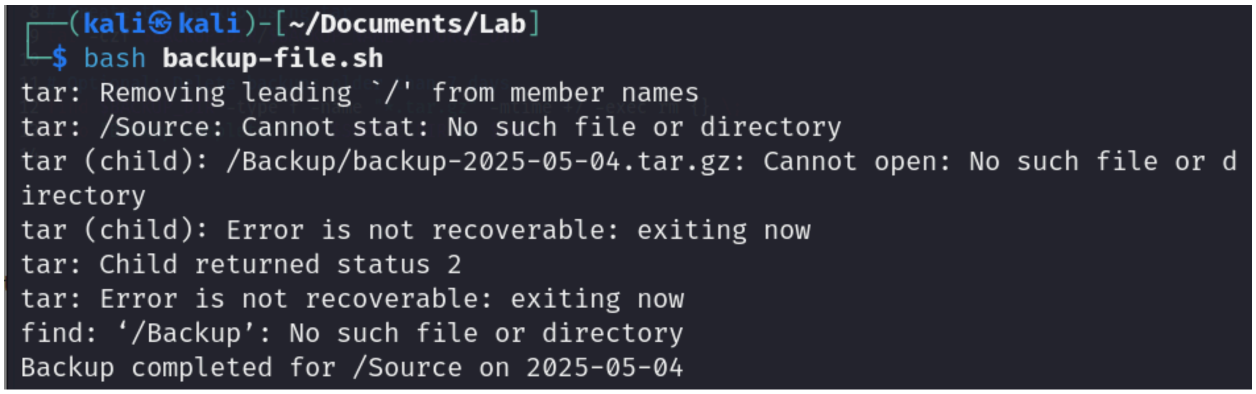
PRACTICAL - 17

AIM: Write a shell script to back up a specified directory daily.

Bash Script :-

```
#!/bin/bash
# Set the directory to back up and the backup location
SOURCE_DIR="/path/to/source"
BACKUP_DIR="/path/to/backup"
DATE=$(date +%F)  # e.g., 2025-04-22
BACKUP_NAME="backup-$DATE.tar.gz"

# Create the backup using tar
tar -czf $BACKUP_DIR/$BACKUP_NAME $SOURCE_DIR

# Optional: Delete backups older than 7 days
find $BACKUP_DIR -type f -name "*.tar.gz" -mtime +7 -exec rm {}
\;
echo "Backup completed for $SOURCE_DIR on $DATE"
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash backup-file.sh
tar: Removing leading `/' from member names
tar: /Source: Cannot stat: No such file or directory
tar (child): /Backup/backup-2025-05-04.tar.gz: Cannot open: No such file or d
irectory
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error is not recoverable: exiting now
find: '/Backup': No such file or directory
Backup completed for /Source on 2025-05-04
```

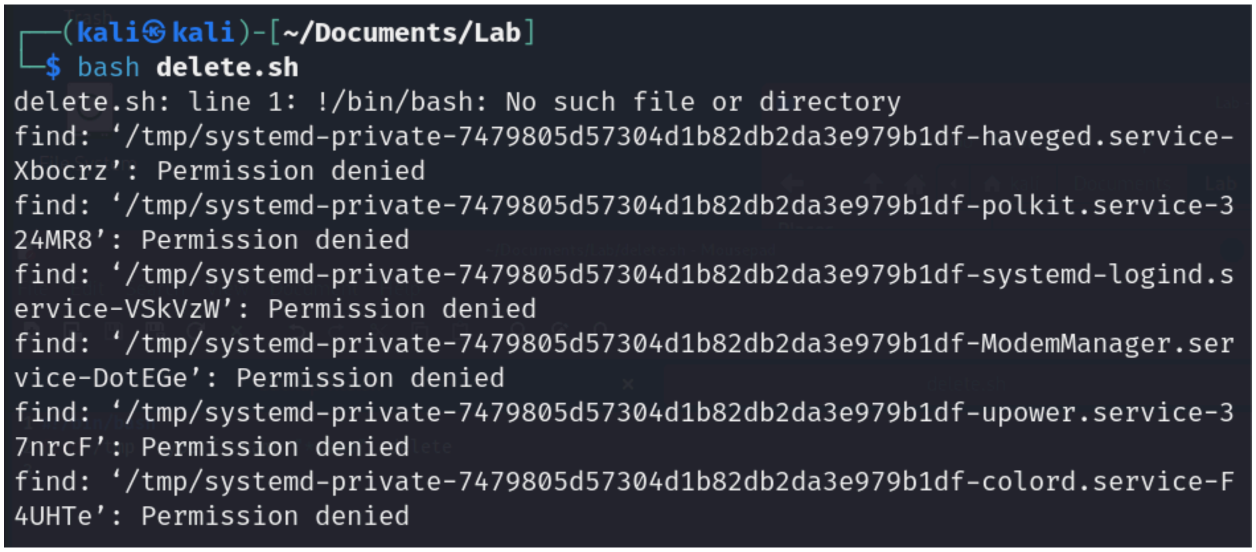
PRACTICAL - 18

AIM: Write a cron expression to schedule a shell script that cleans temporary files every night at midnight.

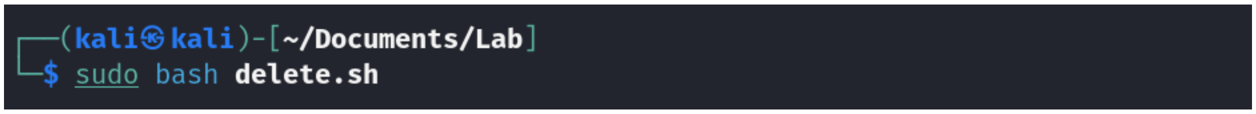
Bash Script :-

```
!/bin/bash
find /tmp -type f -name "*.tmp" -delete
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ bash delete.sh
delete.sh: line 1: !/bin/bash: No such file or directory
find: '/tmp/systemd-private-7479805d57304d1b82db2da3e979b1df-haveged.service-Xbocrz': Permission denied
find: '/tmp/systemd-private-7479805d57304d1b82db2da3e979b1df-polkit.service-324MR8': Permission denied
find: '/tmp/systemd-private-7479805d57304d1b82db2da3e979b1df-systemd-logind.service-VSkVzW': Permission denied
find: '/tmp/systemd-private-7479805d57304d1b82db2da3e979b1df-ModemManager.service-DotEGe': Permission denied
find: '/tmp/systemd-private-7479805d57304d1b82db2da3e979b1df-upower.service-37nrcF': Permission denied
find: '/tmp/systemd-private-7479805d57304d1b82db2da3e979b1df-colord.service-F4UHTe': Permission denied
```



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo bash delete.sh
```

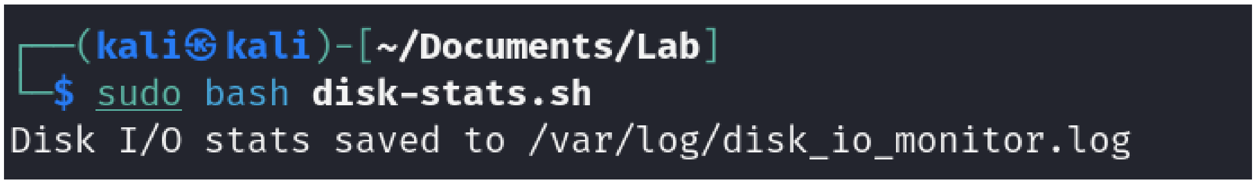
PRACTICAL - 19

AIM: Write a shell script to monitor disk I/O using the iostat command.

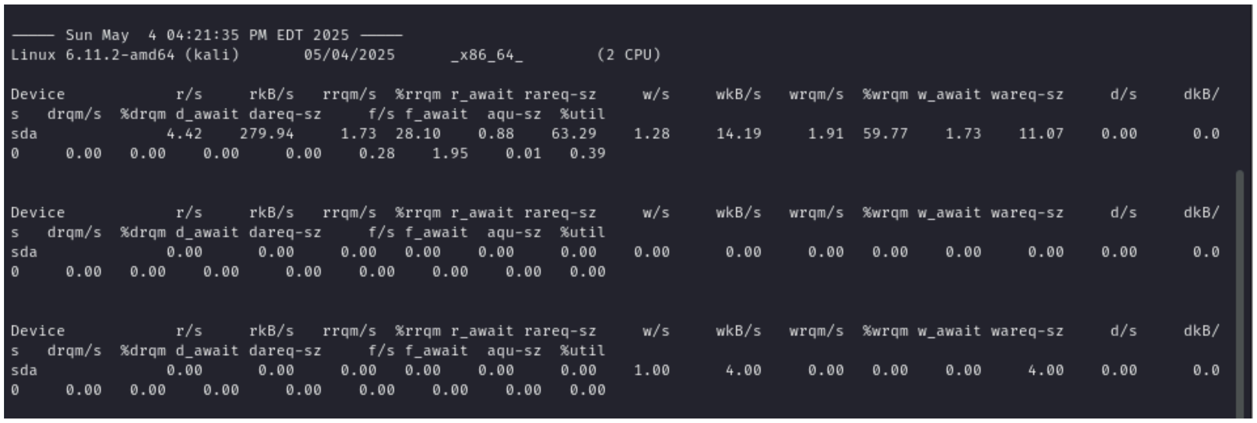
Bash Script :-

```
#!/bin/bash

if ! command -v iostat &> /dev/null; then

    echo "iostat not found. Please install the sysstat package."

    exit 1

fi

LOG_FILE="/var/log/disk_io_monitor.log"

echo "----- $(date) -----" >> $LOG_FILE

iostat -dx 1 3 >> $LOG_FILE

echo "Disk I/O stats saved to $LOG_FILE"
```

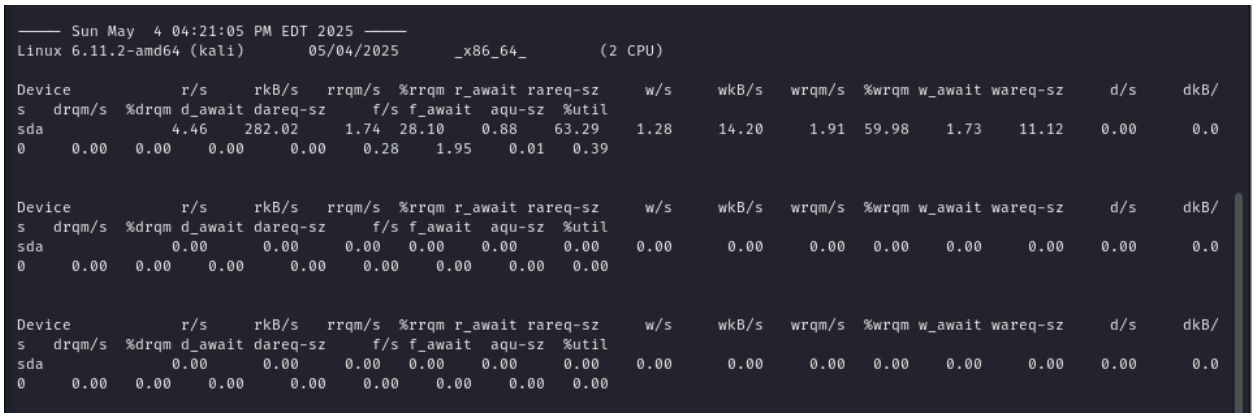
Output :-



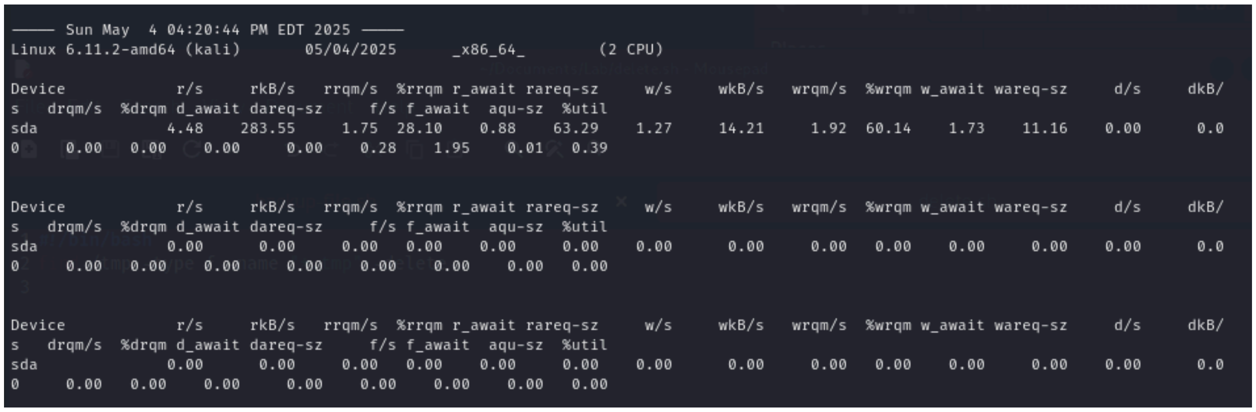
```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo bash disk-stats.sh
Disk I/O stats saved to /var/log/disk_io_monitor.log
```



```
───── Sun May  4 04:21:35 PM EDT 2025 ─────
Linux 6.11.2-amd64 (kali)       05/04/2025      _x86_64_        (2 CPU)

Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              4.42    279.94     1.73  28.10    0.88    63.29    1.28     14.19     1.91  59.77    1.73    11.07    0.00      0.0
0     0.00   0.00    0.00     0.00    0.28    1.95    0.01   0.39


Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.0
0     0.00   0.00    0.00     0.00    0.00    0.00    0.00   0.00


Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              0.00      0.00     0.00   0.00    0.00     0.00    1.00      4.00     0.00   0.00    0.00     4.00    0.00      0.0
0     0.00   0.00    0.00     0.00    0.00    0.00    0.00   0.00
```



```
───── Sun May  4 04:21:05 PM EDT 2025 ─────
Linux 6.11.2-amd64 (kali)       05/04/2025      _x86_64_        (2 CPU)

Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              4.46    282.02     1.74  28.10    0.88    63.29    1.28     14.20     1.91  59.98    1.73    11.12    0.00      0.0
0     0.00   0.00    0.00     0.00    0.28    1.95    0.01   0.39


Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.0
0     0.00   0.00    0.00     0.00    0.00    0.00    0.00   0.00


Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.0
0     0.00   0.00    0.00     0.00    0.00    0.00    0.00   0.00
```



```
───── Sun May  4 04:20:44 PM EDT 2025 ─────
Linux 6.11.2-amd64 (kali)       05/04/2025      _x86_64_        (2 CPU)

Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              4.48    283.55     1.75  28.10    0.88    63.29    1.27     14.21     1.92  60.14    1.73    11.16    0.00      0.0
0     0.00   0.00    0.00     0.00    0.28    1.95    0.01   0.39


Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.0
0     0.00   0.00    0.00     0.00    0.00    0.00    0.00   0.00


Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/
s   drqm/s  %drqm d_await dareq-sz     f/s f_await  aqu-sz  %util
sda              0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.0
0     0.00   0.00    0.00     0.00    0.00    0.00    0.00   0.00
```

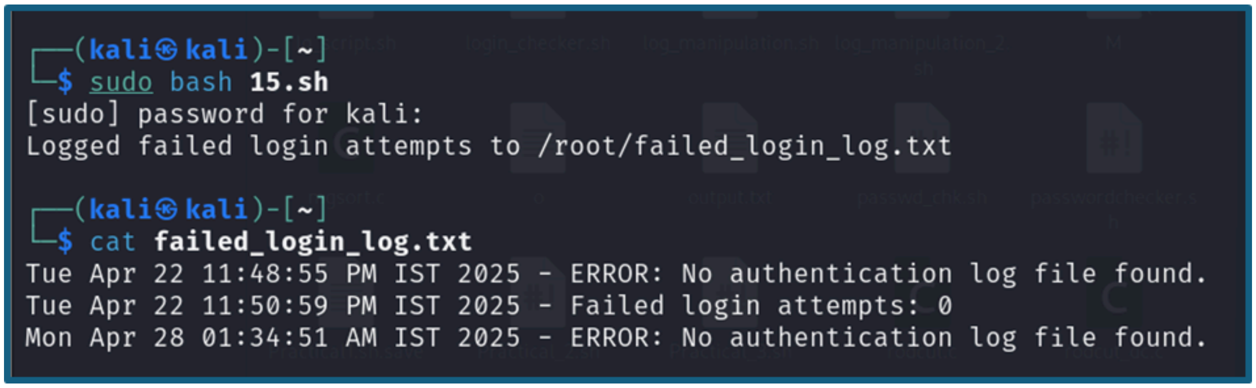
PRACTICAL - 20

AIM: Create a script that counts the number of failed login attempts from the /var/log/auth.log file.

Bash Script :-

```
#!/bin/bash
LOG_FILE="$HOME/failed_login_log.txt"

if [[ -f /var/log/auth.log ]]; then
    AUTH_LOG="/var/log/auth.log"
elif [[ -f /var/log/secure ]]; then
    AUTH_LOG="/var/log/secure"
else
    echo "$(date) - ERROR: No authentication log file found."
>> "$LOG_FILE"
    exit 1
fi

FAILED_COUNT=$(grep "Failed password" "$AUTH_LOG" | wc -l)

echo "$(date) - Failed login attempts: $FAILED_COUNT" >>
"$LOG_FILE"
echo "Logged failed login attempts to $LOG_FILE"
```

Output :-



```
┌──(kali㉿kali)-[~]
└─$ sudo bash 15.sh
[sudo] password for kali:
Logged failed login attempts to /root/failed_login_log.txt

┌──(kali㉿kali)-[~]
└─$ cat failed_login_log.txt
Tue Apr 22 11:48:55 PM IST 2025 - ERROR: No authentication log file found.
Tue Apr 22 11:50:59 PM IST 2025 - Failed login attempts: 0
Mon Apr 28 01:34:51 AM IST 2025 - ERROR: No authentication log file found.
```

PRACTICAL - 21

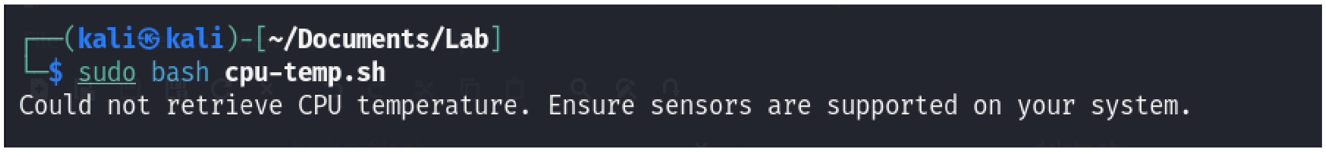
AIM: Write a script to monitor the temperature of the CPU (if supported by sensors).

Bash Script :-

```
#!/bin/bash

threshold=80

logfile="/var/log/cpu_temp.log"

if ! command -v sensors &> /dev/null

then

    echo "sensors command could not be found. Please install
lm-sensors."

    exit 1

fi

cpu_temp=$(sensors | grep 'Core 0' | awk '{print $3}' | tr -d
'+C')  # Remove "+" and "C”

if [ -z "$cpu_temp" ]; then

    echo "Could not retrieve CPU temperature. Ensure sensors
are supported on your system."

    exit 1

fi

timestamp=$(date +"%Y-%m-%d %H:%M:%S")

echo "$timestamp - CPU Temperature: $cpu_temp C" >>
"$logfile"

if [ "$cpu_temp" -ge "$threshold" ]; then
```

```
    echo "Warning: CPU temperature has exceeded the threshold
of $threshold C. Current temperature: $cpu_temp C"
fi
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo bash cpu-temp.sh
Could not retrieve CPU temperature. Ensure sensors are supported on your system.
```

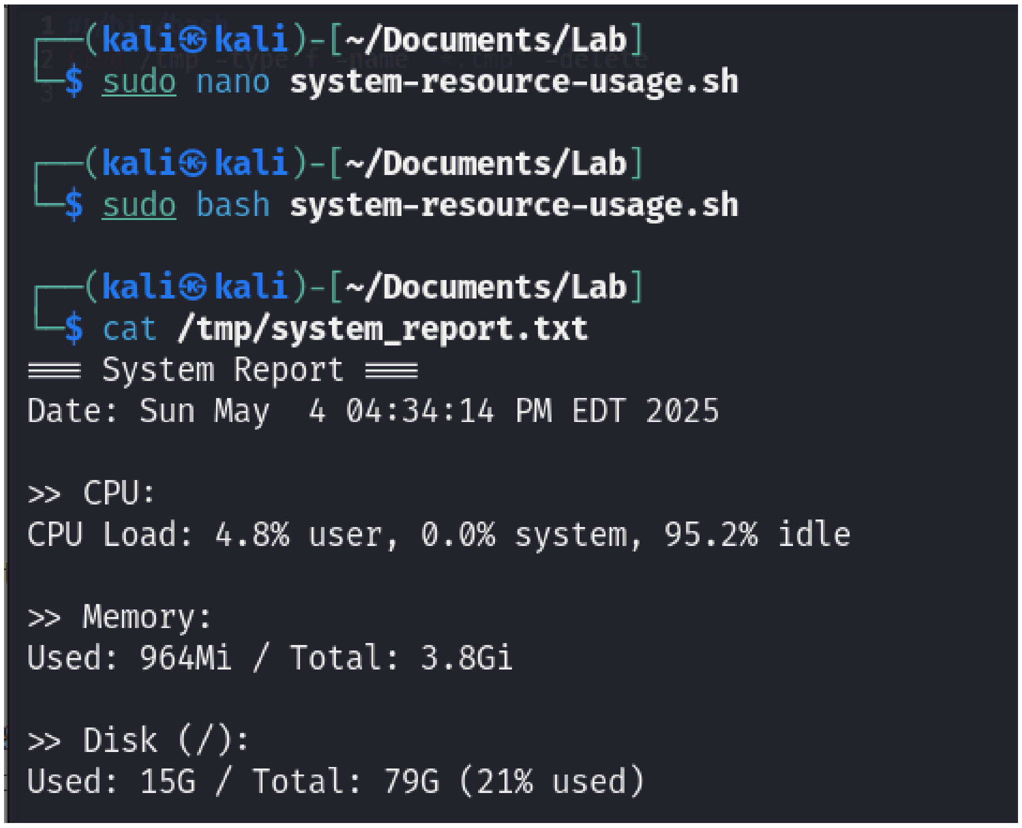
PRACTICAL - 22

AIM: Develop a script to generate a report of system resource usage (CPU, memory, disk) and email it to the admin.

Bash Script :-

```
#!/bin/bash
REPORT="/tmp/system_report.txt"
ADMIN_EMAIL="testmail@gmail.com"
{
    echo "=== System Report ==="
    echo "Date: $(date)"
    echo
    echo ">> CPU:"
    top -bn1 | grep "Cpu(s)" | awk '{print "CPU Load: " $2 "%
user, " $4 "% system, " $8 "% idle"}'
    echo
    echo ">> Memory:"
    free -h | awk '/^Mem:/ {print "Used: "$3" / Total: "$2}'
    echo
    echo ">> Disk (/):"
    df -h / | awk 'NR==2 {print "Used: "$3" / Total: "$2"
("$5" used)"}'
    echo
} > "$REPORT"

mail -s "System Report" "$ADMIN_EMAIL" < "$REPORT"
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo nano system-resource-usage.sh

┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo bash system-resource-usage.sh

┌──(kali㉿kali)-[~/Documents/Lab]
└─$ cat /tmp/system_report.txt
=== System Report ===
Date: Sun May  4 04:34:14 PM EDT 2025

>> CPU:
CPU Load: 4.8% user, 0.0% system, 95.2% idle

>> Memory:
Used: 964Mi / Total: 3.8Gi

>> Disk (/):
Used: 15G / Total: 79G (21% used)
```

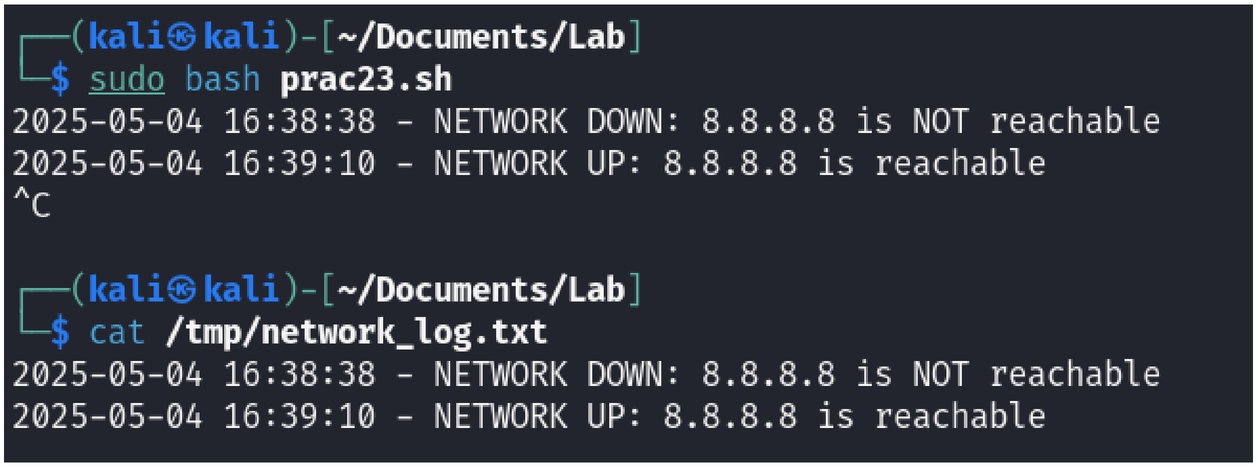
PRACTICAL - 23

AIM: Explain the importance of logging in shell scripts. Modify one of your earlier scripts to write logs with timestamps to a file.

Bash Script :-

```
#!/bin/bash

HOST=8.8.8.8

LOGFILE="/tmp/network_log.txt"

while true; do

    TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

    if ping -c1 -W2 "$HOST" &>/dev/null; then

        MSG="$TIMESTAMP - NETWORK UP: $HOST is reachable"

    else

        MSG="$TIMESTAMP - NETWORK DOWN: $HOST is NOT
reachable"

    fi

    echo "$MSG" | tee -a "$LOGFILE"

    sleep 30

done
```

Output :-



```
┌──(kali㉿kali)-[~/Documents/Lab]
└─$ sudo bash prac23.sh
2025-05-04 16:38:38 - NETWORK DOWN: 8.8.8.8 is NOT reachable
2025-05-04 16:39:10 - NETWORK UP: 8.8.8.8 is reachable
^C

┌──(kali㉿kali)-[~/Documents/Lab]
└─$ cat /tmp/network_log.txt
2025-05-04 16:38:38 - NETWORK DOWN: 8.8.8.8 is NOT reachable
2025-05-04 16:39:10 - NETWORK UP: 8.8.8.8 is reachable
```

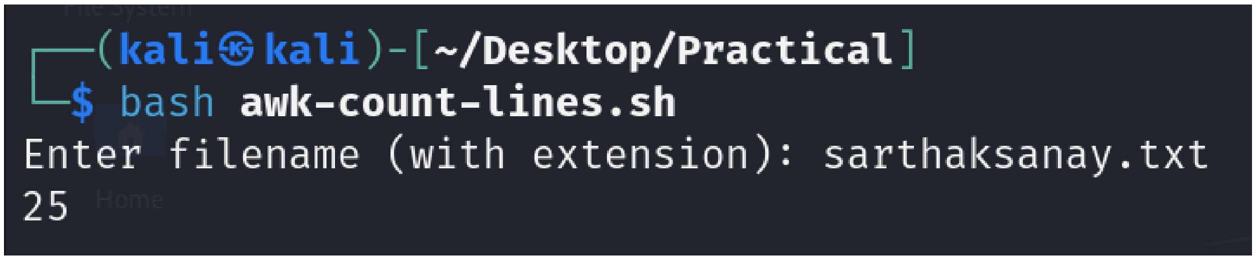
PRACTICAL - 24

AIM: Write an Awk script to count the number of lines in a given file.

Bash Script :-

```
#!/bin/bash
count_lines() {
    file="$1"
    if [[ -f "$file" ]]; then
        awk 'END { print NR }' "$file"
    else
        echo "File not found!"
    fi
}
read -p "Enter filename (with extension): " file
count_lines "$file"
```

Output :-



```
┌──(kali㉿kali)-[~/Desktop/Practical]
└─$ bash awk-count-lines.sh
Enter filename (with extension): sarthaksanay.txt
25
```

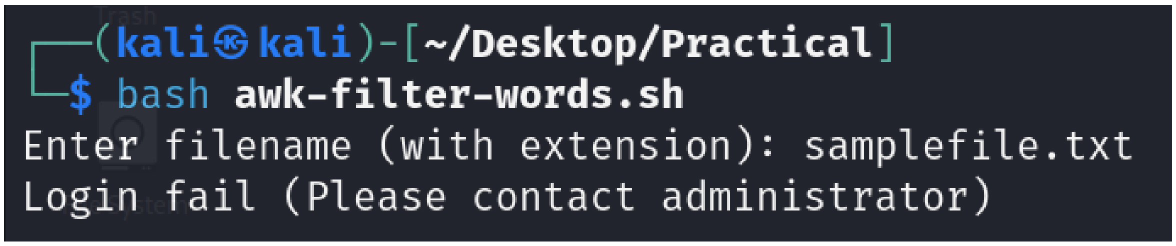
PRACTICAL - 25

AIM: Write an Awk script to display lines that contain a specific keyword, such as "fail" or "error".

Bash Script :-

```
#!/usr/bin/bash
read -p "Enter filename (with extension): " file
if [[ ! -r "$file" ]]; then
  echo "Error: '$file' not found or not readable." >&2
  exit 1
fi
awk '/fail|error/' "$file"
```

Output :-



```
┌──(kali㉿kali)-[~/Desktop/Practical]
└─$ bash awk-filter-words.sh
Enter filename (with extension): samplefile.txt
Login fail (Please contact administrator)
```