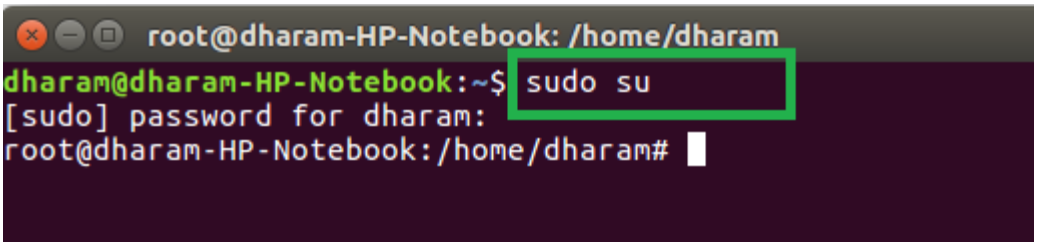
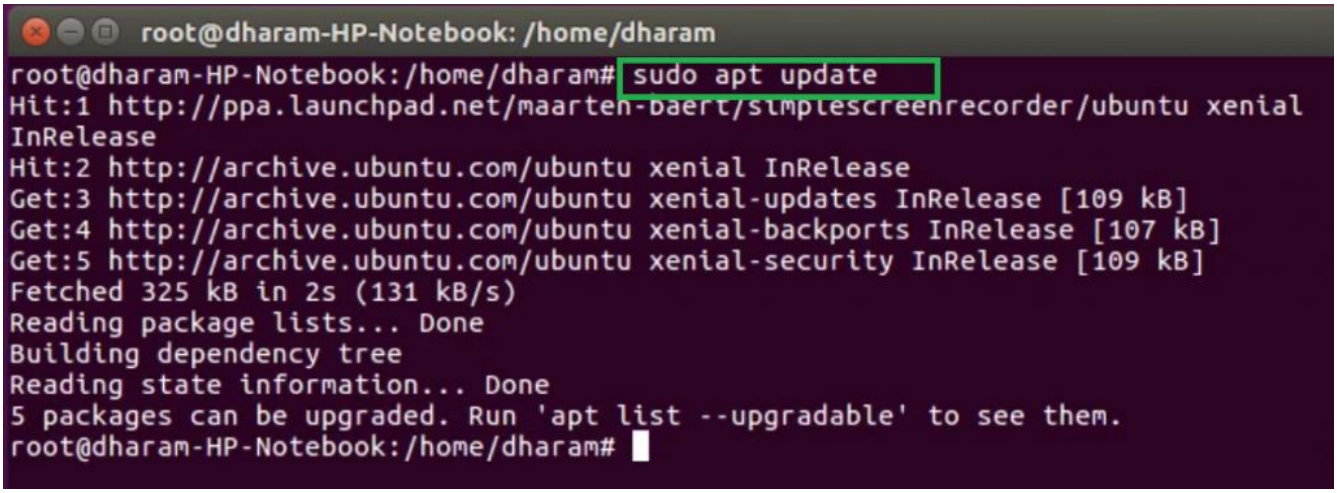
Steps to install the Apache server:

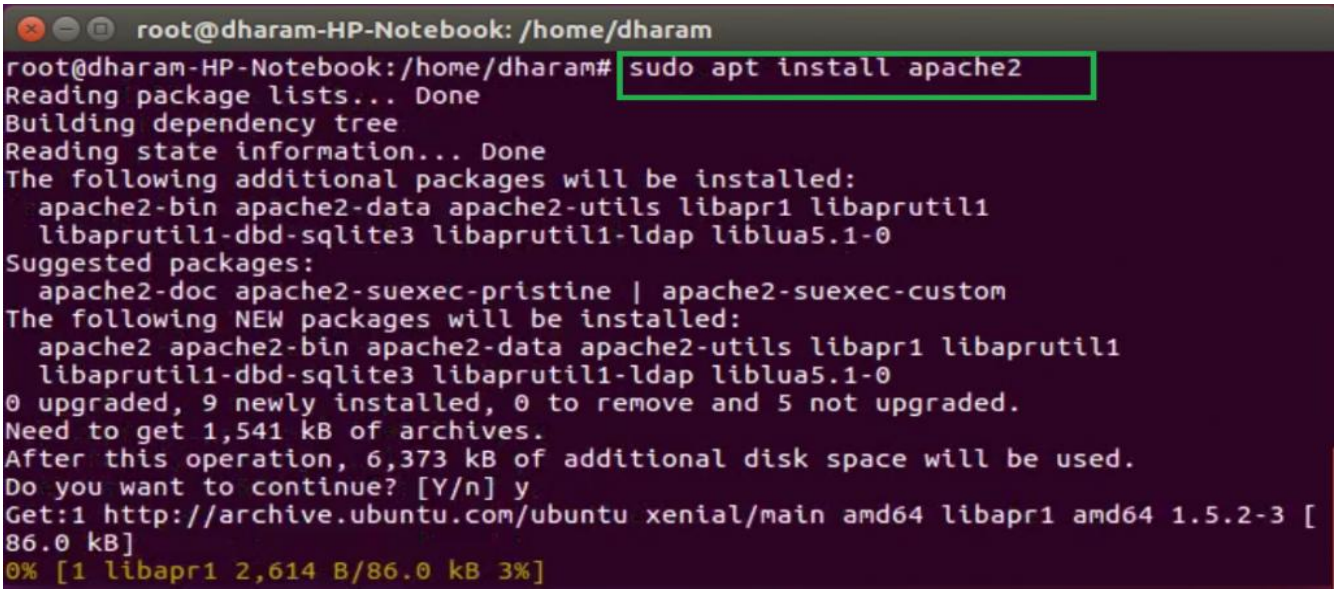
- **Make Superuser:** Open terminal and use following command to make yourself superuser.
  `$Sudo su`



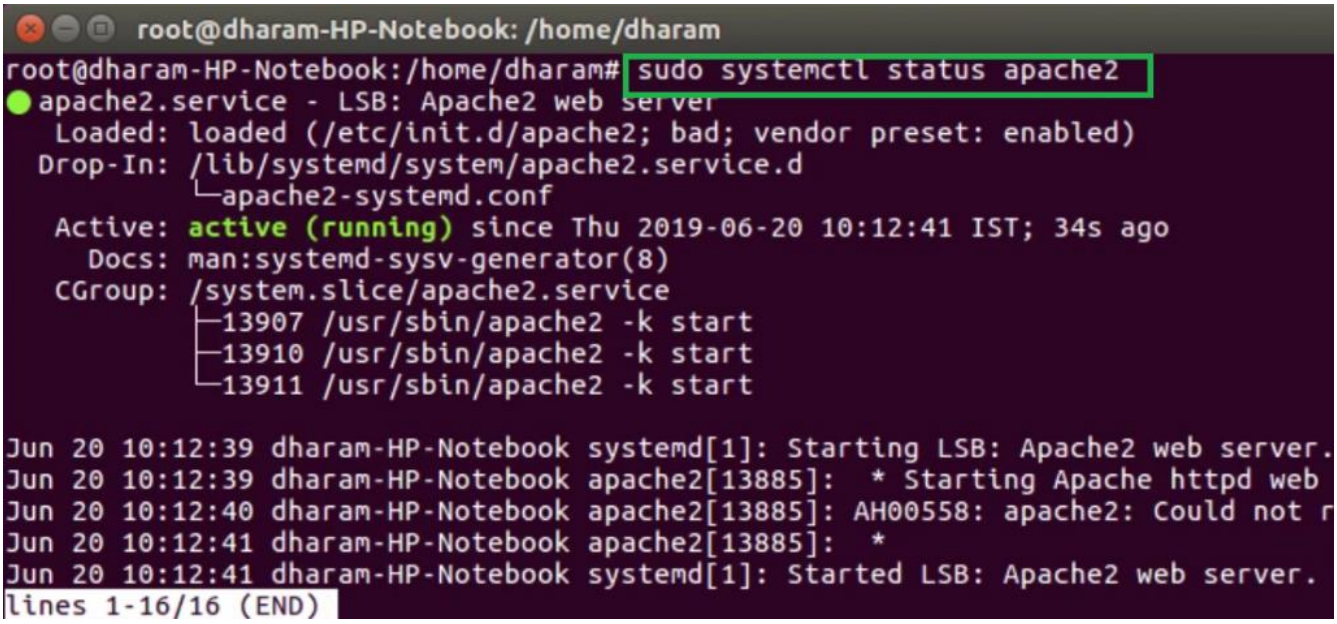
- **Update Ubuntu package:** Use the following command to update the Ubuntu package list.
  `sudo apt update`



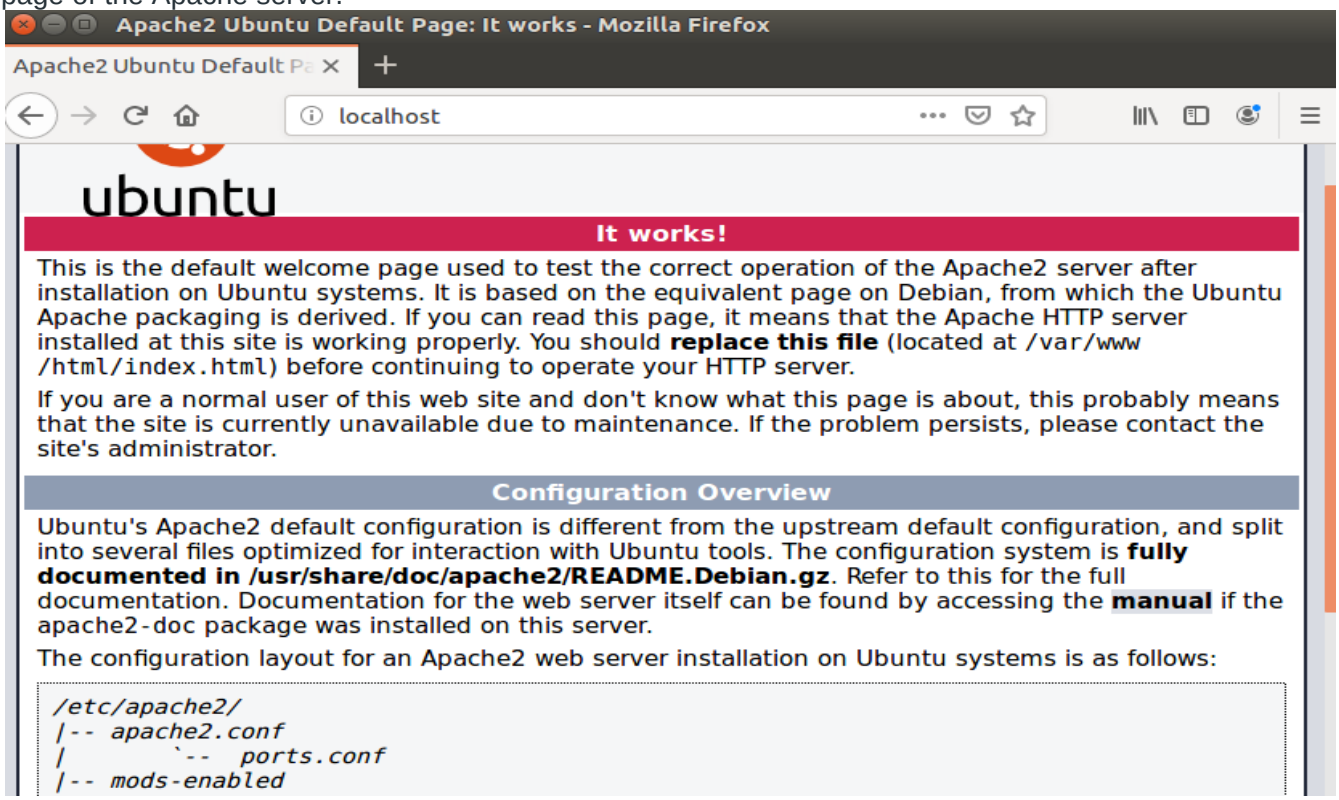
- **Install Apache:** After installing the Ubuntu package list, use the following command to install apache server.
  `sudo apt install apache2`



- After completion of the installation process, the Apache server automatically start.
- The status of the Apache server can be checked by using the following command.
  `Sudo systemctl status apache2`



- Open the browser and type **localhost** or **127.0.0.1** on the address bar. It will display the default page of the Apache server.




```
#Adjusting the Firewall
$sudo ufw app list
$sudo ufw allow 'Apache'
$sudo ufw status


#Managing the Apache Process
$sudo systemctl stop apache2
$sudo systemctl start apache2
```

```
$sudo systemctl restart apache2
$sudo systemctl reload apache2
$sudo systemctl enable apache2
$sudo systemctl disable apache2
```

Managing Apache Server

¬ You must understand the main configuration files of Apache which can be found in the "/etc/apache2" folder.

¬ You can explore this folder by following command:

$cd /etc/apache2

$ls

you will see file listing. Following configuration files are necessary to understand:

apache2.conf : This is the main configuration file for the server. You can do almost all configurations within this file.

ports.conf : You can specify the ports from where virtual hosts can listen.

conf.d/ (Directory) : This directory is used for controlling specific features of the Apache configuration. For example, It is frequently used to define SSL configuration.

sites-available/ (Directory) : This directory contains all of the virtual hosts files that define different web sites.

sites-enabled/ (Directory) : This directory defines which virtual host definitions are actually being used.

mods-[enabled,available]/ (Directory) : These directories are similar in function to sites directories, but they define modules that can be loaded optionally.

#how to create user in linux

$sudo adduser mital

#Verification

$cat /etc/passwd

#You can also use the getent command to verify that user account added to the system:

$getent passwd mital

#How to delete a user account

$sudo userdel mital

#How to change Linux user password

$sudo passwd mital

#Install and Configure Samba

A Samba file server enables file sharing across different operating systems over a network. It lets you access your desktop files from a laptop and share files with Windows and macOS users.

#Installing Samba

$sudo apt update

$sudo apt install samba

$where is samba

#Setting up Samba

Now that Samba is installed, we need to create a directory for it to share:

$mkdir /home/<username>/sambashare/

#he command above creates a new folder sambashare in our home directory which we will share later.

#The configuration file for Samba is located at /etc/samba/smb.conf. To add the new directory as a share, we edit the file by running:

$sudo nano /etc/samba/smb.conf

#At the bottom of the file, add the following lines:

[sambashare]

comment = Samba on Ubuntu

path = /home/username/sambashare

read only = no

browsable = yes

#Then press Ctrl-O to save and Ctrl-X to exit from the nano text editor.

#What we've just added

comment: A brief description of the share.

path: The directory of our share.

read only: Permission to modify the contents of the share folder is only granted when the value of this directive is no.

browsable: When set to yes, file managers such as Ubuntu's default file manager will list this share under "Network" (it could also appear as browseable).

#Now that we have our new share configured, save it and restart Samba for it to take effect:

$sudo service smbd restart

#Update the firewall rules to allow Samba traffic:

$sudo ufw allow samba

#Setting up User Accounts and Connecting to Share

$sudo smbpasswd -a mital

How to install Firewall in Ubuntu

Ubuntu comes with **UFW (Uncomplicated Firewall)** pre-installed. However, if it's missing or you want to reinstall it, follow these steps:

1. Update Package Lists

Before installing, update your system's package list:

```
sudo apt update
```

2. Install UFW

Run the following command:

```
sudo apt install ufw -y
```

3. Verify Installation

Check if UFW is installed:

```
sudo ufw status
```

If it's inactive, enable it with:

```
sudo ufw enable
```

4. (Optional) Install `iptables` or `firewalld`

- If you prefer **iptables**:

  ```
  sudo apt install iptables -y
  ```

- If you want to use **firewalld** (alternative to UFW):

  ```
  sudo apt install firewalld -y
  sudo systemctl enable --now firewalld
  ```

Configure Ubuntu's Built-In Firewall

1. Check Firewall Status

```
sudo ufw status
```

If it’s **inactive**, you’ll need to enable it.

---

2. Enable UFW

```
sudo ufw enable
```

This starts the firewall and ensures it runs on startup.

---

3. Allow or Deny Specific Ports

- Allow SSH (default port 22):

  ```
  sudo ufw allow ssh
  ```

- Allow HTTP (port 80):

  ```
  sudo ufw allow 80/tcp
  ```

- Allow HTTPS (port 443):

  ```
  sudo ufw allow 443/tcp
  ```

- Deny a specific port (e.g., 23 for Telnet):

  ```
  sudo ufw deny 23
  ```

---

4. Allow/Deny Specific IPs

- Allow only a specific IP (e.g., `192.168.1.100`) on SSH:

  ```
  sudo ufw allow from 192.168.1.100 to any port 22
  ```

- Deny access from an IP:

  ```
  sudo ufw deny from 192.168.1.200
  ```

---

5. Allow/Deny Specific Network Ranges

- Allow a subnet (e.g., `192.168.1.0/24`) on SSH:

  ```
  sudo ufw allow from 192.168.1.0/24 to any port 22
  ```

---

6. Enable Logging

```
sudo ufw logging on
```

Logs are stored in `/var/log/ufw.log`.

---

7. Disable UFW

If needed, you can disable UFW with:

```
sudo ufw disable
```

---

8. Reset UFW (Clear All Rules)

```
sudo ufw reset
```

This will disable UFW and remove all rules.

---

9. Reload Firewall

```
sudo ufw reload
```

Use this after making changes to apply the new rules.

---

10. Advanced: Allow Specific Services

Instead of using port numbers, you can allow services:

```
sudo ufw allow OpenSSH
```

Run `sudo ufw app list` to see available services.

Final Check

After configuring, verify rules:

```
sudo ufw status verbose
```

Working with Wine:-

WINE (Wine Is Not an Emulator) allows you to run Windows applications on Linux. Here's a step-by-step guide to installing and using WINE on Ubuntu.

How to install Wine on Ubuntu 24.04

1. Update and upgrade the Ubuntu package list

```
sudo apt update
```

Next, upgrade all packages using the following command.

```
sudo apt upgrade -y
```

2. Enable the 32-bit architecture to support 32-bit Windows applications

Unlike many software programs, Wine uses the 32-bit architecture. This is because most Windows programs still use the 32-bit architecture. To verify that Wine can execute 32-bit Windows apps, run the command below on a 64-bit Ubuntu.

```
sudo dpkg --add-architecture i386
```

3. Create a keyring and download the Wine repository key

Wine has a repository key that verifies the integrity of the packages retrieved from the Wine repository. This repository key has to be stored in a keyring. In Linux, keyrings are used to store sensitive data safely. Use the following command to create a keyring for the Wine repository key.

```
sudo mkdir -pm755 /etc/apt/keyrings
```

Next, download the repository key and store it in the previous keyring you created.

```
sudo wget -O /etc/apt/keyrings/winehq-archive.key https://dl.winehq.org/wine-builds/winehq.key
```

4. Add the Wine repository source file to Ubuntu

Use the following command to add the Wine repository source file to Ubuntu. By adding the official WineHQ repository, you will receive access to the most recent stable, development, or staging versions of Wine, which include new improvements, and bug fixes not found in the usual Linux repositories.

```
sudo wget -NP /etc/apt/sources.list.d/ https://dl.winehq.org/wine-builds/ubuntu/dists/jammy/winehq-jammy.sources
```

5. Synchronize the Ubuntu system to apply changes

Next, synchronize your Ubuntu system using the following command to apply the previous changes you made.

```
sudo apt update
```

6. Install the stable version of Wine

Finally, use the following command to install Wine:

```
sudo apt install --install-recommends winehq-stable -y
```

7. Confirm if the Wine installation was a success by checking the version

To confirm that Wine has been installed successfully, run the following command:

```
wine --version
```

Copy

You will get the following similar output that shows the installed Wine version:

```
boemo@Boemo-Wame-Mmopelwa:~$ wine --version
wine-9.0
boemo@Boemo-Wame-Mmopelwa:~$
```

GNU/GPL

The **GNU General Public License (GNU GPL)** is a widely used free software license that guarantees end users the freedom to run, study, share, and modify software. It was created by **Richard Stallman** for the **GNU Project** as part of the Free Software Foundation (FSF).

Key Concepts of GNU GPL:

1. **Freedom to Use** – You can use the software for any purpose.
2. **Freedom to Study and Modify** – The source code must be available, allowing users to study and change it.
3. **Freedom to Share** – You can distribute copies of the software to others.
4. **Freedom to Share Modifications** – If you modify the software and distribute it, you must also share the source code under the same GPL license.

Copyleft Principle:

- GPL is a **copyleft** license, meaning that any derivative work must also be licensed under GPL.
- This ensures that software based on GPL-licensed code remains **free and open-source**.

Different Versions:

- **GPLv1 (1989)** – The first version.
- **GPLv2 (1991)** – Added better protection against license circumvention.
- **GPLv3 (2007)** – Updated for modern concerns, including software patents and Tivoization (hardware restrictions).

GPL vs. Other Open-Source Licenses:

- **GPL requires derivatives to be GPL** (copyleft).
- **MIT, Apache, BSD licenses** allow proprietary redistribution (permissive).

Common Uses:

- Many popular software projects use GPL, including **Linux Kernel, GNU Utilities, and MySQL**.

Startup, Shutdown, and Boot Loaders in Linux

Linux follows a structured process for **starting up** (booting) and **shutting down**. The boot process involves multiple stages, from powering on the

system to loading the operating system. Below is an overview of **startup, shutdown, and bootloaders** in Linux.

---

1. Startup Process in Linux

The Linux startup process follows these main steps:

Step 1: BIOS / UEFI Initialization

- When the system is powered on, the **BIOS (Basic Input/Output System)** or **UEFI (Unified Extensible Firmware Interface)** initializes hardware components.
- It searches for a **bootable disk** and loads the bootloader.

Step 2: Bootloader Execution

- The bootloader (e.g., **GRUB**, **LILO**, **Syslinux**, **systemd-boot**) is responsible for loading the Linux kernel.
- It presents a menu to select the kernel or OS (in case of dual-boot systems).
- The bootloader loads the selected kernel into memory.

Step 3: Kernel Initialization

- The kernel initializes system hardware, mounts the root filesystem, and starts the first system process (init or systemd).
- Kernel loads drivers and sets up hardware interfaces.

Step 4: Init/Systemd Process Execution

- The init **(older systems) or** systemd **(modern Linux distros)** is the first user-space process (PID 1).
- It starts essential system services and background daemons.
- The system enters the default **runlevel/target** (multi-user, graphical mode, etc.).

Step 5: User Login & Desktop Environment

- A login prompt (CLI or GUI) is presented.
- The user logs in, and the desktop environment (GNOME, KDE, etc.) starts.

---

2. Bootloaders in Linux

A bootloader is software that loads the Linux kernel into memory and starts the operating system.

Common Bootloaders:

1. **GRUB (GRand Unified Bootloader)**
    - The most commonly used Linux bootloader.
    - Supports multiple kernels and operating systems.
    - Allows kernel parameter modifications at boot time.
2. **LILO (Linux Loader)**
    - An older bootloader, replaced by GRUB.
    - Lacks a graphical interface and dynamic configuration.
3. **Syslinux / PXELINUX**
    - Lightweight bootloaders for embedded systems and network booting.
4. **systemd-boot**
    - A simple UEFI boot manager integrated with systemd.

---

3. Shutdown Process in Linux

Shutting down Linux involves terminating processes and unmounting filesystems properly.

Shutdown Commands:

1. shutdown -h now → Immediately shuts down the system.
2. shutdown -r now → Restarts the system.
3. reboot → Simple reboot command.
4. poweroff → Powers down the machine.

During shutdown:

- **systemd/init sends termination signals to processes.**
- **Unmounts filesystems safely.**
- **Kernel halts the sys**tem **or reboots.**

- **List the Special three section of the smb.conf file.**

The `smb.conf` file in Samba is divided into different sections, including three **special sections** that define global and share-specific settings. These three special sections are:

1. `[global]` **Section**
   - This section contains settings that apply globally to the Samba server, such as authentication, networking, logging, and security configurations.
2. `[homes]` **Section**
   - This special share allows users to automatically access their home directories when they log in. Samba dynamically maps this to the respective user's home directory.
3. `[printers]` **Section**
   - This section is used for printer sharing. It provides automatic sharing of all printers configured on the Samba server.

These sections help in configuring Samba for file and printer sharing efficiently.