MASTER OF COMPUTER APPLICATIONS

PRACTICAL RECORD WORK

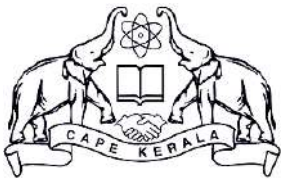
ON

20MCA136 Networking & System Administration Lab

Submitted

By

ATHIRA T
(Reg. No. : VDA20MCA-2022)



DEPARTMENT OF COMPUTER APPLICATIONS
COLLEGE OF ENGINEERING VADAKARA
(CAPE - GOVT. OF KERALA)

APRIL - 2021

DEPARTMENT OF COMPUTER APPLICATIONS
COLLEGE OF ENGINEERING VADAKARA
(CAPE - GOVT. OF KERALA)

CERTIFICATE

Certified that this is a bona fide record of the practical work on the course **20MCA136 NETWORKING & SYSTEM ADMINISTRATION** LAB done by Ms. ATHIRA T (Reg.No.: **VDA20MCA-202)** Second Semester MCA student of Department of Computer Applications at College of Engineering Vadakara in the partial fulfilment for the award of the degree of Master of Computer Applications (MCA) of APJ Abdul Kalam Technological University (KTU)

(Ms. Nidhina)

(Ms. Anagha)

FACULTY-IN-CHARGE **HEAD OF THE DEPARTMENT**

CEV
03/10/2021

EXAMINERS:

SL.NO	EXPERIMENTS	REMARKS
1	INTRODUCTION TO COMPUTER HARDWARE	
2	STUDY OF TERMINAL BASED TEXT EDITOR SUCH AS VIM	
3	FILE SYSTEM HIERARCHY IN A COMMON LINUX DISTRIBUTION	
4	SHELL SCRIPTING	
5	INSTALLATION AND CONFIGURATION OF LAMP STACK	
6	INSTALLATION AND CONFIGURATION OF COMMON SOFTWARE FRAMEWORKS SUCH AS LARAVEL	
7	BUILD AND INSTALL SOFTWARE FROM SOURCE CODE	
8	INTRODUCTION TO COMMAND LINE TOOLS FOR NETWORKING	
9	ANALYZING PACKET STREAM USING TCPDUMP AND WIRESHARK	
10	INTRODUCTION TO HYPERVISORS,VMS ,DOCKERS	
11	AUTOMATION USING ANSIBLE	

EXPERIMENT 1

AIM:-

Introduction to Computer hardware: Physical identification of major components of a computer system such as mother board, RAM modules, daughter cards, bus slots, SMPS, internal storage devices,i nterfacing ports. Specifications of desktop and server class computers. Installation of common operating systems for desktop and server use. (Students may be asked to formulate specification for computer to be used as Desktop, Web server)

ANSWER:-

Computer Hardware is the physical part of a computer, as distinguished from the **computer software** that executes or runs on the hardware. The hardware of a computer is infrequently changed, while software and data are modified frequently. The term soft refers to readily created, modified, or erased. These are unlike the physical components within the computer which are hard.

i. MOTHERBOARD

The motherboard is the body or mainframe of the computer, through which all other components **interface.** It is the central circuit board making up a complex electronic system. A motherboard provides the electrical connections by which the other components of the system communicate. The mother board includes many components such as: central processing unit (CPU), random access memory (RAM), firmware, and internal and external buses.

ii. RANDOM ACCESS MEMORY(RAM)

Random access memory (RAM) is fast-access memory that is cleared when the computer is power-down. RAM attaches directly to the motherboard, and is used to store programs that are currently running. RAM is a set of integrated circuits that allow the stored data to be accessed in any order (why it is called random).

There are many different types of RAM. Distinctions between these different types include: writable vs. read-only, static vs. dynamic, volatile vs. non-volatile, etc.

iii. DAUGHTER CARDS

The daughter board is a computer hardware. It is also known as the piggyback board, riser card, daughter board, daughtercard or daughter card. A daughter board is a printed circuit board which is connected to the motherboard or expansion card. As compared to the motherboard, it is smaller in size. A daughter board does not act as an expansion card. An expansion card adds extra new functions to the computer. But a daughter board that is connected to the motherboard adds or supports the main functions of the motherboard.

Daughter boards are directly connected to the motherboards. You know that expansion cards are connected to the motherboard by using the bus and other serial interfaces. But daughter board is directly connected to the board by soldering. As an update of the motherboard or expansion card, daughter boards are released to extend the features and services of the motherboard or expansion cards.

iv. BUS SLOTS

Alternatively known as a expansion port, an expansion slot is a connection or port inside a computer on the motherboard or riser card. It provides an installation point for a hardware expansion card to be connected. For example, if you wanted to install a new video card in the computer, you'd purchase a video 5 expansion card and install that card into the compatible expansion slot.

An expansion slot is a socket on the motherboard that is used to insert an expansion card (or circuit board), which provides additional features to a computer such as video, sound, advanced graphics, Ethernet or memory.

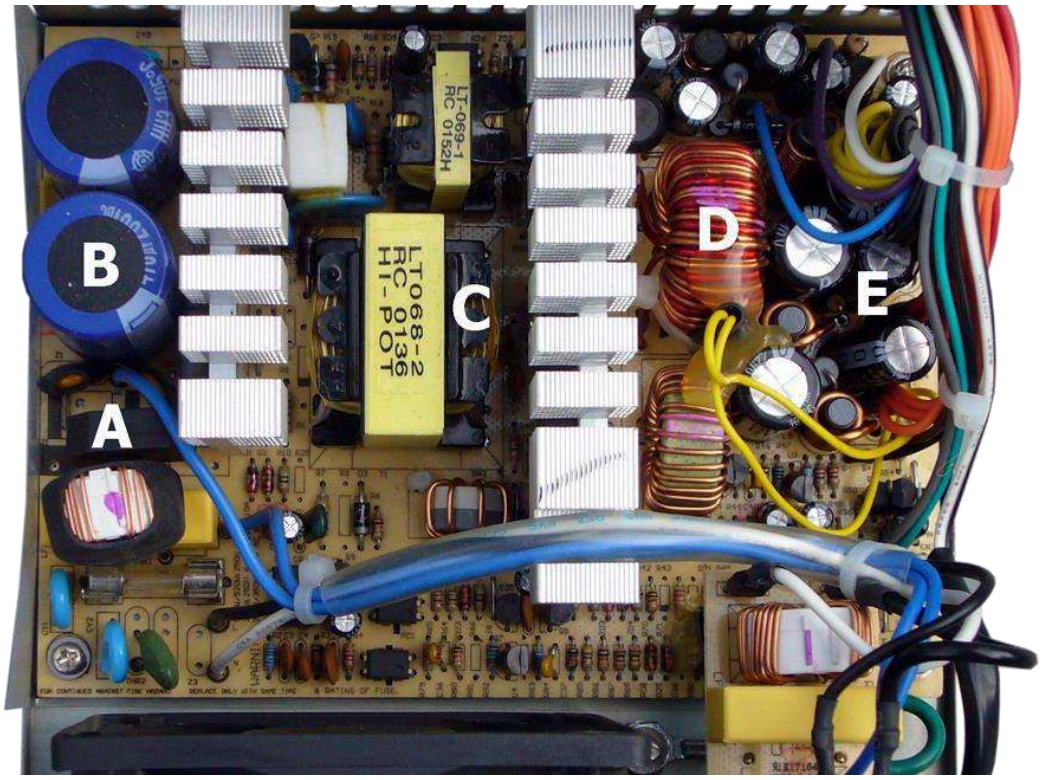
The expansion card has an edge connector that fits precisely into the expansion slot as well as a row of contacts that is designed to establish an electrical connection between the motherboard and the electronics on the card, which are mostly integrated circuits. Depending on the form factor of the case and motherboard, a computer system generally can have anywhere from one to seven expansion slots. With a backplane system, up to 19 expansion cards can be installed.

v. SMPS

A switched-mode power supply (SMPS) is an electronic circuit that converts power using switching devices that are turned on and off at high frequencies, and storage components such as inductors or capacitors to supply power when the switching device is in its non-conduction state.

Switching power supplies have high efficiency and are widely used in a variety of electronic equipment, including computers and other sensitive equipment requiring stable and efficient power supply.

A switched-mode power supply is also known as a switch-mode power supply or switching-mode power supply.

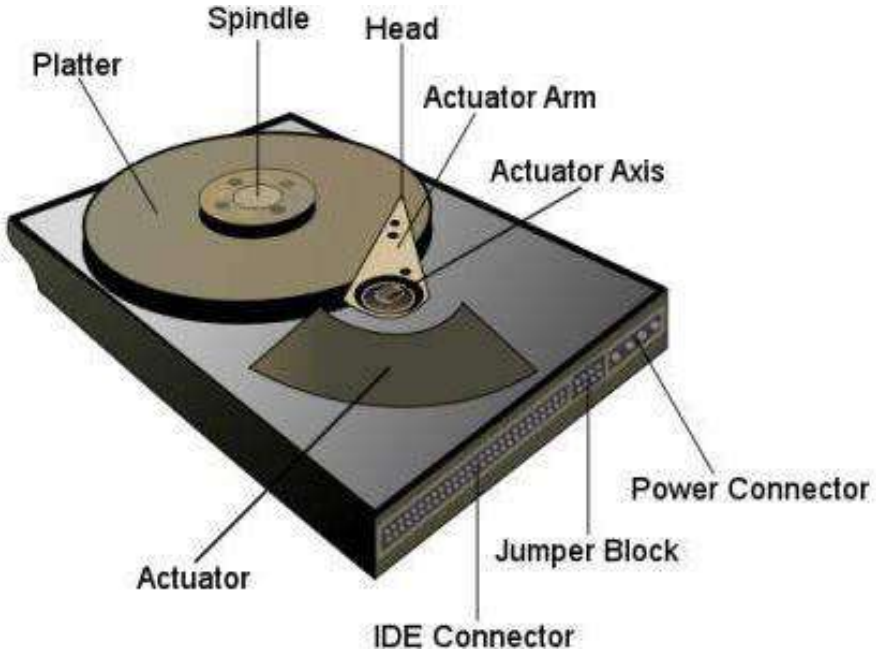


vi. INTERNAL STORAGE

Internal storage is hardware that keeps data inside the computer for later use and remains persistent even when the computer has no power. There are a few different types of internal storage. Hard disks are the most popular type of internal storage. Solid-state drives have grown in popularity slowly. A disk array controller is popular when you need more storage then a single har disk can hold.

i. Hard Disk Drive

A hard disk drive (HDD) is a non-volatile storage device which stores digitally encoded data on rapidly rotating platters with magnetic surfaces. Just about every new computer comes with a hard disk these days unless it comes with a new solid-state drive. Typical desktop hard disk drives store between 120 and 400GB, rotate at 7,200 rpm, and have a madia transfer rate of 1 Gbit/s or higher. Hard disk drives are accessed over one of a number of bus types, including parallel ATA(also called IDE), Serial ATA (SATA), SCSI, Serial Attached SCSI, and Fibre Channel.



ii. Solid-State Drive

A solid-state drive (SSD) is a data storage device that uses solid-state memory to store persistent data. An SSD emulates a hard disk drive, thus easily replacing it in any application. SSDs have begun to appear in laptops because they can be smaller than HDDs. SSDs are currently more expensive per unit of capacity than HDDs which is why they have not caught on so quickly.

iii. Disk Array Controller

A disk array controller is a device which manage the physical disk drives and presents them to the computer as logical units. It almost always implements hardware RAID. RAID (Redundant Array of Independent Drives) is a technology that employs the simultaneous use of two or more hard disk drives to achieve greater levels of performance, reliability, and/or larger data volume sizes. A disk array controller also provides additional disk cache.

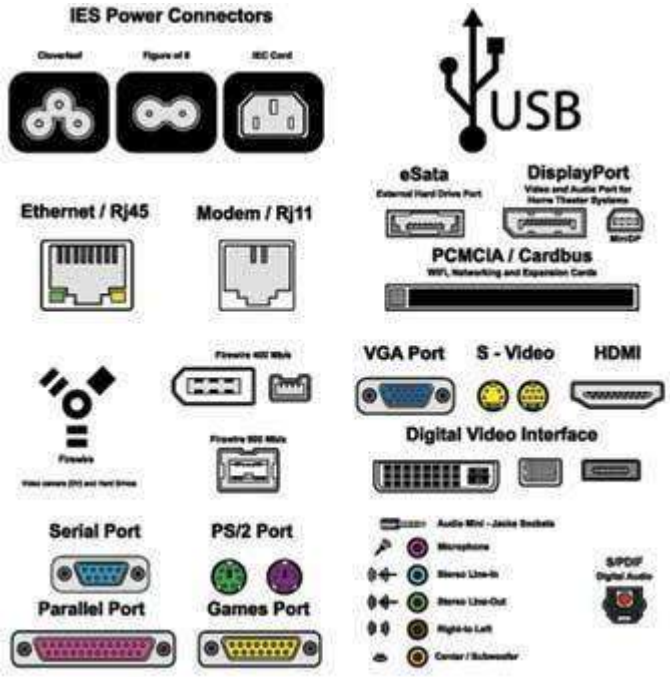
vii. INTERFACING PORTS

A port is a physical docking point using which an external device can be connected to the computer. It can also be programmatic docking point through which information flows from a program to the computer or over the Internet.

A port has the following characteristics −

- External devices are connected to a computer using cables and ports.

- Ports are slots on the motherboard into which a cable of external device is plugged in.
- Examples of external devices attached via ports are the mouse, keyboard, monitor, microphone, speakers, etc.



Let us now discuss a few important types of ports −

Serial Port

- Used for external modems and older computer mouse
- Two versions: 9 pin, 25 pin model
- Data travels at 115 kilobits per second

Parallel Port

- Used for scanners and printers
- Also called printer port

- IEEE 1284-compliant Centronics port

PS/2 Port

- Used for old computer keyboard and mouse
- Also called mouse port
- Most of the old computers provide two PS/2 port, each for the mouse and keyboard
- IEEE 1284-compliant Centronics port

Universal Serial Bus (or USB) Port

- It can connect all kinds of external USB devices such as external hard disk, printer, scanner, mouse, keyboard, etc.
- It was introduced in 1997.
- Most of the computers provide two USB ports as minimum.
- Data travels at 12 megabits per seconds.
- USB compliant devices can get power from a USB port.

VGA Port

- Connects monitor to a computer's video card.
- It has 15 holes.
- Similar to the serial port connector. However, serial port connector has pins, VGA port has holes.

Power Connector

- Three-pronged plug.
- Connects to the computer's power cable that plugs into a power bar or wall socket.

Firewire Port

- Transfers large amount of data at very fast speed.
- Connects camcorders and video equipment to the computer.
- Data travels at 400 to 800 megabits per seconds.
- Invented by Apple.
- It has three variants: 4-Pin FireWire 400 connector, 6-Pin FireWire 400 connector, and 9-Pin FireWire 800 connector.

Modem Port

Ethernet Port

- Connects to a network and high speed Internet.
- Connects the network cable to a computer.
- This port resides on an Ethernet Card.
- Data travels at 10 megabits to 1000 megabits per seconds depending upon the network bandwidth.

Game Port

- Connect a joystick to a PC
- Now replaced by USB

Digital Video Interface, DVI port

- Connects Flat panel LCD monitor to the computer's high-end video graphic cards.
- Very popular among video card manufacturers.

Sockets

- Sockets connect the microphone and speakers to the sound card of the computer.

System Specification

System specifications help to define the operational and performance guidelines for a system.System requirements can be broadly classified as functional requirements, data requirements, quality requirements and constraints. They are often provided to consumers in complete detail. System requirements often indicate the minimum and the recommended configuration. The former is the most basic requirement, enough for a product to install or run, but performance is not guaranteed to be optimal. The latter ensures a smooth operation.

Hardware system requirements often specify the operating system version, processor type, memory size, available disk space and additional peripherals, if any, needed. Software system requirements, in addition to the aforementioned requirements, may also specify additional software dependencies (e.g., libraries, driver version, framework version). Some hardware/software manufacturers provide an upgrade assistant program that users can download and run to determine whether their system meets a product's requirements.

What affects a computers performance?

Overall, the performance of a computer is dependant on how well it works together as a whole. Continually upgrading one part of the computer while leaving outdated parts installed will not improve performance much, if at all. Below, we discuss some of the most important parts of the computer regarding it's speed and computing power. The description of these parts is by no means complete and only serves to give newer users some understanding of what various computer specifications mean. It should also be noted that this web page was last updated January 2003, but the same factors can still be applied in 2006. The processor, memory and videocard are the most important components when determining performance inside a computer. Any specifics about pieces of hardware will be outdated in about six months or so. Gaining an understanding of what each specification means, and what each part does, is the goal of this section.

Reference Chart

Bit (b)	Smallest unit of storage possible. 1 or 0.
Byte (B)	8 bits
KiloByte (KB)	1000* Bytes
MegaByte (MB)	1000 KB
GigaByte (GB)	1000 MB

* Commonly approximated as 1000 for convenience. Actual value is 1024.

1. Processor speed (MHZ, L1 L2 cache, x86 and other chip types)

Average PC Desktop (1.5 - 2.5 Ghz)

Average Laptop or Macintosh (1.0 Ghz)

Clock speed, a.k.a. Processor speed is often played up to be the major factor in a computer's overall performance. In rare cases this is true, but an average user rarely uses 100 percent of his Central Processing Unit's power. (CPU). Things like encoding video or encrypting files, or anything that computes large, complex, numbers requires a lot of processor power. Most

users spend most of their time typing, reading email or viewing web pages. During this time, the computer's CPU is probably hovering around 1 or 2 percent of it's total speed. Startup time is probably the only time the CPU is under stress, and even then it's often limited due to the hard drive speed.

Megahertz (MHZ) or Gigahertz (GHZ or 1000MHZ) is the number of times the CPU can switch back and forth from 1 to 0. It is the driving force in the power the processor has (all other things being equal). Higher MHZ chips use more power and produce more heat.

Level 1 (L1) and Level 2(L2) cache is usually onchip RAM that is extremely fast. SRAM is different than System RAM and only used on processors. It stores data right before and after it is processed. SRAM is extremely expensive; most chips today only have 128 Kilobytes of L1 cache, and 256-512K of L2 cache. (This is what makes a Pentium 3 or 4 chip different than a celeron chip)

x86 is the architecture type of all Windows based computers. All processors sold today for computers running the Windows OS (operating system) are 32 bit, meaning they process 32 bits of information each clock cycle (a 1GHZ chip does 1 billion clock cycles per second). Not all CPU's are x86. For example Apple computers use Motorola's chip design called PowerPC, which comes in both 64 and 128 bit flavors. This is one reason apple computers can outpreform high-end PC's, despite their lower processor speeds. Currently Intel and AMD are developing 64 bit x86 chips. The disadvantage of higher bit architecture is that one needs to make changes to any software that one may want to work with on the new design; this is one reason Mac software will not run without specialized software on PC's, and visa- versa.

2. System RAM speed and size (MHZ and Megabytes)

Average Desktop - 256 megabytes

Average Laptop - 128 megabytes

The amount and speed of the RAM in your computer makes a huge difference in how your computer performs. If you are trying to run Windows XP with 64 MB of RAM it probably won't even work. When the computer uses up all available RAM it has to start using the hard drive to cache data, which is much slower. The constant transfer of data between RAM and virtual memory (hard drive memory) slows a computer down considerably. Especially when trying to load applications or files.

The two types differ in the technology they use to hold data, dynamic RAM being the more common type. Dynamic RAM needs to be refreshed thousands of times per second. Static RAM does not need to be refreshed, which makes it faster; but it is also more expensive than dynamic RAM. Both types of RAM are volatile, meaning that they lose their contents when the power is turned off.Also the speed of your RAM can be influential. The normal speed of RAM in most computers today is pc100 (100mhz). This runs fine for most applications. Gamers or high-end machines probably are using DDR (double data rate) RAM. It's newer and more expensive, but runs considerably faster (266mhz). Note that all computers cannot use DDR RAM.

3. Disk speed and size (RPM's and Gigabytes)

Average Desktop (40 Gigabytes)

Average Laptop (20 Gigabytes)

The biggest factor in your computer's performance is the hard disk speed. How fast the hard drive can find (average seek time), read, write, and transfer data will make a big difference in the way your computer performs. Most hard drives today spin at 7,200 RPMS, older models and laptops still spin at 5,200 RPMS, which is one reason laptops often appear sluggish to a desktop equivalent.

The size of your hard drive plays a very little role in the performance of a computer. As long as your have enough free space for virtual memory and keep the disk defragmented it will perform well no matter what the size.

4. Video card - (onboard video RAM, chip type and speed)

Average Desktop (32 - 64 Megabyte low end AGP card)

Average Laptop (16 Megabyte onboard chip)

Whenever your computer puts an image on the screen something has to render it. If a computer is doing this with software it is often slow and will affect the performance of the rest of the computer. Also, the image will not be rendered as crisp or as smoothly in the case of video. Even a low-end video card will significantly improve the performance of the computer by taking the large task of rendering the images on the screen from the CPU to the graphics card. If you work with large image files, video or play games you will want a higher end video card.Video cards use their own RAM called Video RAM. The more Video RAM a computer has the more textures and images the card can remember at a time. High end graphics cards for desktops now come with up to 64 megabytes of Video RAM, Laptops often only have 8 or 16 megabytes of Video RAM.

SERVER

A server is a computer that serves information to other computers. These computers, called clients, can **connect to a server** through either a local area network or a wide area network, such as the internet. A server is a vital piece of **your IT infrastructure**. A server collects and sends information across a network. That can be a local network, like your business network, or a wider network across multiple locations.

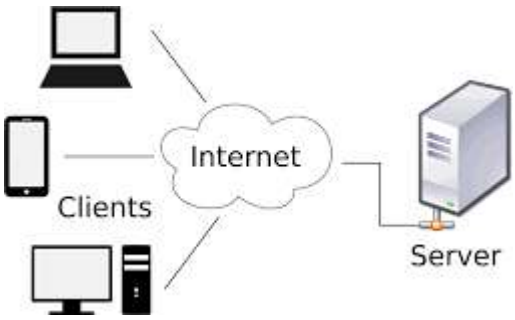
Any computer running the right software can be a server. Although when we hear the word server we think of enormous, high-powered machines that push and pull data across the web.

How does a server works?

Every time you use the internet you are accessing a server. When you enter a URL into a browser your computer communicates with the server hosting that website and pulls the data onto your machine.

This is a simplistic view of how the process works

- You enter a URL and your web browser requests a web page
- The web browser requests a full URL for the site it wants to display
- This information is sent to the server
- The web server finds and builds all the data needed to display the site (this is why some sites load quicker than others)
- Your web browser receives the data and displays the website to you



Types of servers

There are many types of servers, which are as follows:

- Webserver
- Application server
- Blade server
- Cloud server
- Database server

- Print server
- Proxy server
- File server
- Mail server
- Standalone server
- Domain name service

Web Server

A web server offers web pages or other content to the web browser by loading the information from a disc and transfer files by using a network to the user's **web browser.** It is used by a computer or collection of computers to provide content to several users over the internet. This exchange was done with the help of **HTTP** communicating between the browser and the server. There are some examples of web servers given below; you can also download these web servers from given below *download links:*

- Apache: **https://www.apache.org/**
- Tomcat: **https://tomcat.apache.org/**
- Nginx: **https://www.nginx.com/**
- Savant: **http://savant.sourceforge.net/**
- Boa: **http://www.boa.org/**
- FoxServ: **http://www.foxserv.net/**
- IIS: **https://www.iis.net/**
- Lighttpd: **https://www.lighttpd.net/**

Application server

It is an environment where applications are able to run, no matter which types of applications and what operation they perform. It is also known as a type of middleware and can be able to develop and run web-based applications. Generally, it is used to connect database servers and end-user. There are several types of application servers, as well as **.NET Framework**, **Java**, and **PHP** application servers.

Furthermore, it offers users various advantages, such as:

- It allows applications for a more centralized approach to updates and upgrades, which provides data and code integrity.
- It offers security with the help of the authenticating process and centralizing the management of data access.
- For heavy usage applications, it improves performance by limiting network traffic.

Blade server

It is a hardware component, also known as an expansion module, or a high-density server that can be installed into a chassis. It provides advanced functionality, such as allows an expansion card in a computer at a much bigger scale. For example, if more fiber lines are required, additional fiber blades can be added, as a switch or router with the blade server provides complete customization.

Dell PowerEdge 1855

Servers can be reduced to a single thin server by removing hard drives, ongoing miniaturization of computing parts, and eliminating internal cooling, which is known as the blade server. Additionally, it can be stored in racks in server rooms as the blade servers are smaller in size and can be replaced more easily. It can save space and make easy a network of hundreds of servers.

Cloud server

It is a virtual server instead of a physical server that runs in a cloud computing environment. It can be accessed by using remote as it is hosted, built, and delivered via cloud computing platform over the internet. It has similar functionality and capabilities to a traditional physical server but accessed through remotely from a cloud service provider. Today's there are different types of server providers, as well as **IBM** Cloud, Google's Cloud Platform, and Microsoft Azure.

Database server

It is a computer system that allows other systems to access and retrieve data from a database. These servers respond to several requests to the clients and run database applications. Databases can require extraordinary amounts of disk space and can be accessed by multiple clients at any given time. It is also used by many companies for storage purposes. It allows users to access the data with the help of running a query by using a query language specific to the database. For example, SQL is a structured query language, which allows executing a

query to access the data. The most common types of database server software include **DB2, Oracle,** Microsoft **SQL,** and Informix.

Dedicated server

A dedicated server is a single computer, which is hosted by a company and allows only one company to rent and access. It is dedicated to only one client and cannot be shared with any other clients. Some of the networks require one computer to be isolated for managing connections between all other devices. A dedicated server can be a part of a computer that has the capability to manage printer resources.

Remember that all servers cannot be a dedicated server. In some networks, it can be possible for a computer to work as a server and also able to perform other functionalities. The hosting company offers an add-on service for the client, like administration services to freeing the client from having to worry about the server. The hosting company also utilizes hard security plans for providing safeguard their clients' data.

Furthermore, the hosting company keeps all or most of the maintenance on the dedicated server. Such as:

- It maintains all update activities of the operating system and any installed applications.
- It monitors the server and applications and manages security by intrusion detection and prevention.
- It contains data backups, disaster recovery, and firewall maintenance.

Print server

The printer server manages one or more printers over the network. It is responsible for responding to print requests from several clients, rather than attaching a printer to every workstation. Nowadays, some higher-end and larger printers are available with their own built-in print server that eliminates the requirement for an additional computer-based server.

Proxy server

A computer server that acts as an intermediary between a client and a server known as a proxy server. It is a part of another computer or gateway server that isolates a local network from outside networks. It takes requests from the client and passes it to another server for processing. It receives the requested information from the second server. Then, it replies to the original client as if it is giving a reply own self.

A proxy server loads the page faster and reduces the network bandwidth as it caches all pages that accessed through the network. A page that is not in proxy server cache, it accesses this page via its own IP address. Thereafter, it caches that page and sends it to the user.

File server

It is a computer on a network that is used to store and distribute files. It allows multiple users or clients to share files, which is stored on a server. Furthermore, it can improve performance by maximizing readability and writing speeds.

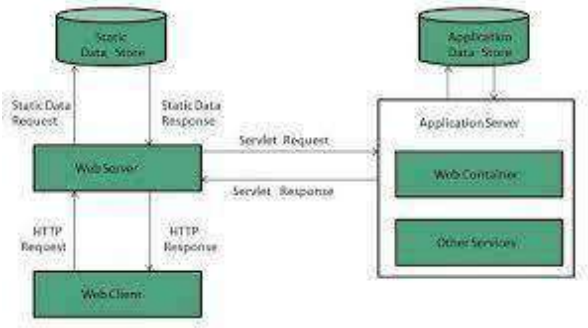
Mail server

A mail server is a central computer that stores electronic emails for clients over the network. It is much like the post office that obtains emails sent to the user and stores them until it is not requested by a user. It uses standard email protocols to send and receive an email like, **simple mail transfer protocol (SMTP)** handles outgoing mail requests and sends messages. The POP3 and IMAP protocols are used to process incoming mail and also receive messages.
These protocols handle all the connections when users log on to a mail server by using email or webmail interface.

Sometimes, mail servers and web servers are merged in a single machine. However, Hotmail and Gmail (public mail services) and large ISPs (Internet service provides) may use dedicated hardware to send and receive an email. A mail server software must be installed on the computer, which gives permission to the administrator of the system to create and manage email accounts for any domains hosted on the server. For instance, if the domain name 'javatpoint.com' is hosted by the server, it has the ability to provide email accounts ending in 'javatpoint.com.

Standalone server

A standalone server is a serial transmission replacement for the parallel SCSI, and it runs alone. It is an improvement of traditional SCSI and does not belong to a **Windows** domain. It supports a maximum of 128 synchronous devices at a transmission speed of 3 Gb in a second. It can also communicate with SATA and SCSI and includes two data ports. It offers local authentication and access control for any resource that is generated from a standalone server. Additionally, users only need to create user account other than it does not need any complex actions, as it does not offer network logon services.

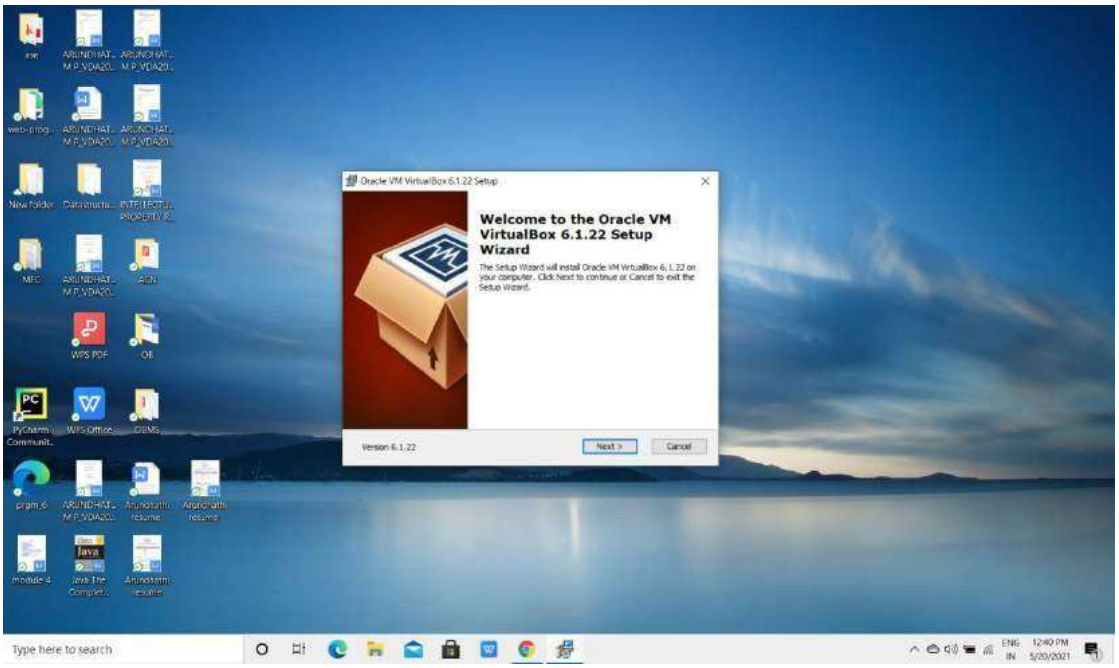


VIRTUAL BOX INSTALLATION

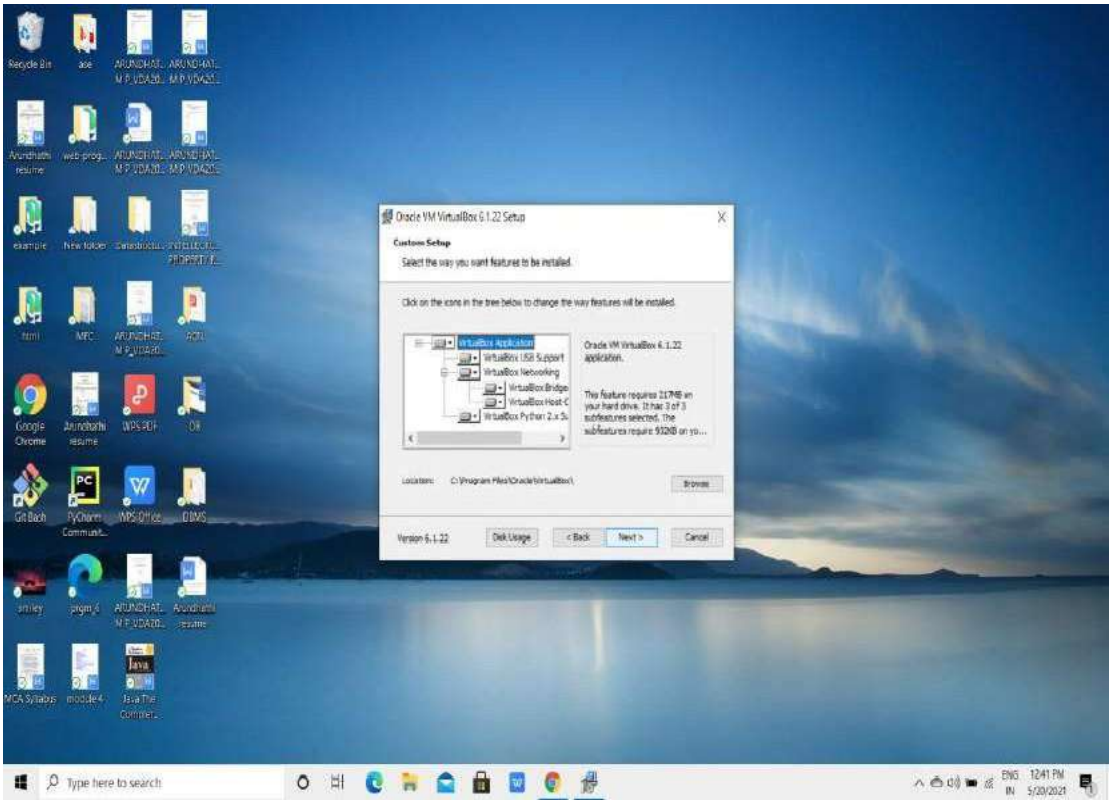
STEP 1:Go and search for virtual box in the browser.

STEP 2:Download the virtualbox for windows.

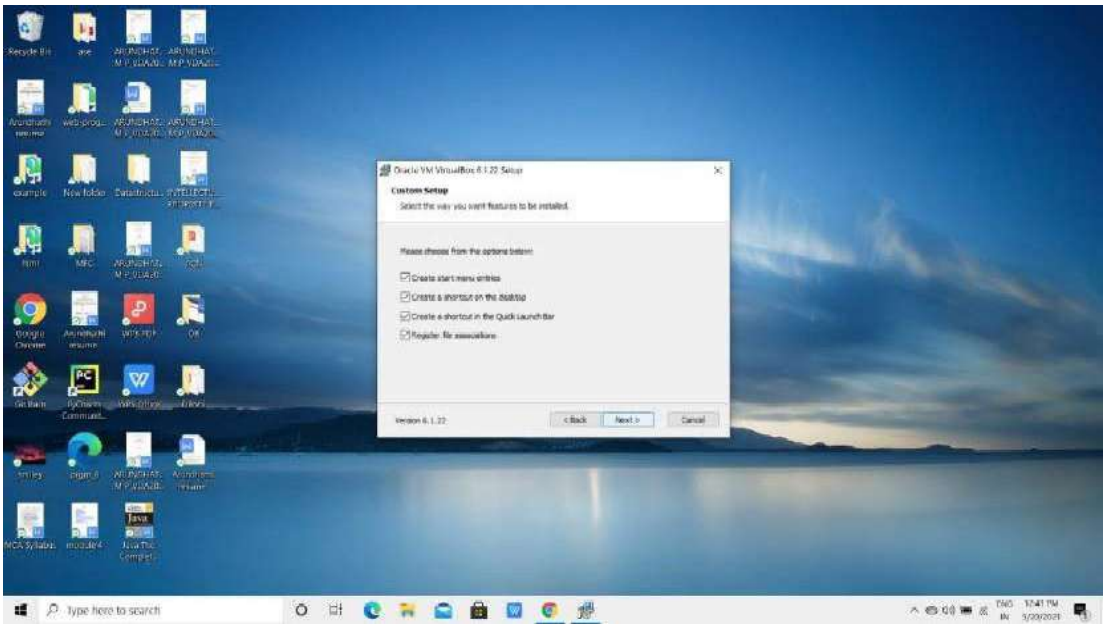
STEP 3:Virtualbox downloaded.



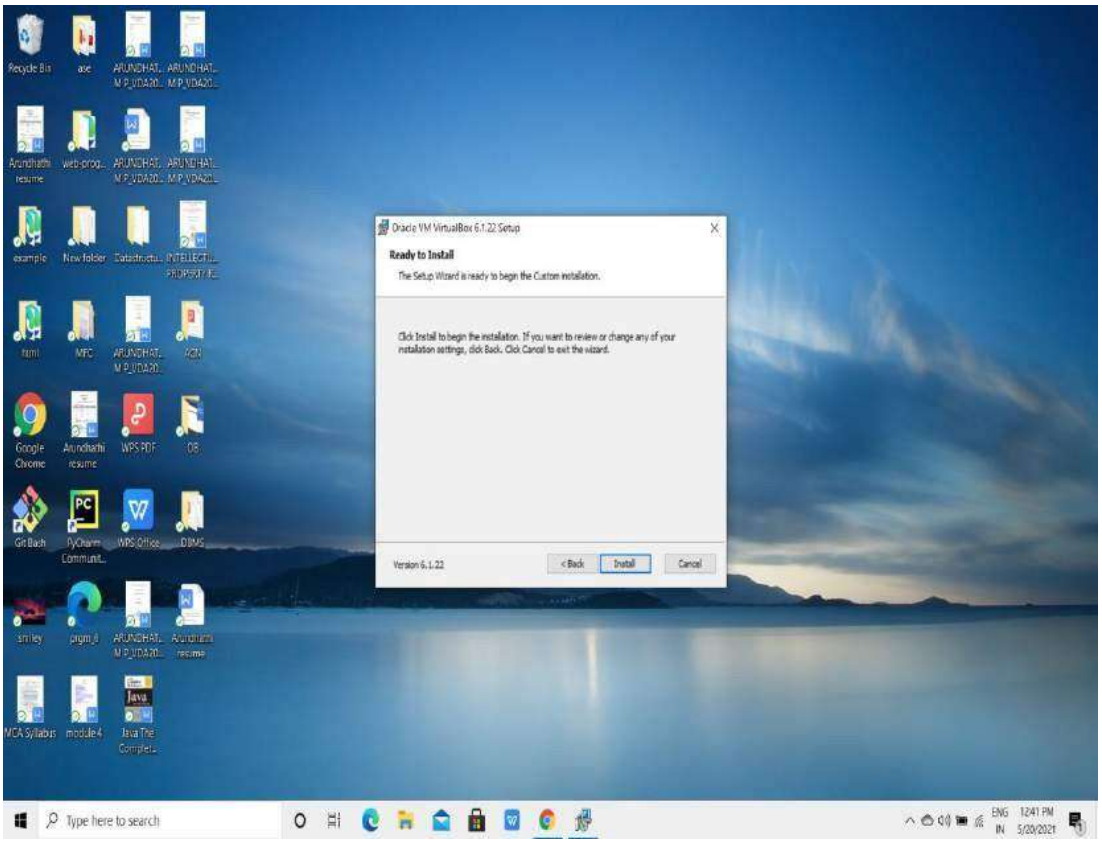
STEP 4:Click Next



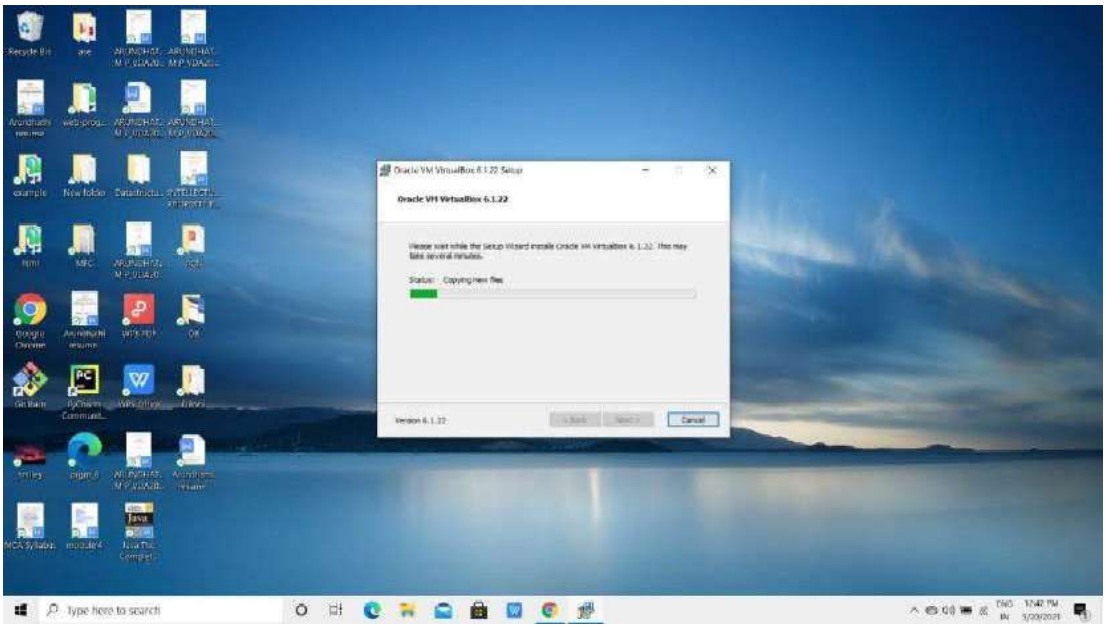
STEP 5:Click Next



STEP 6:Click YES and Click install.



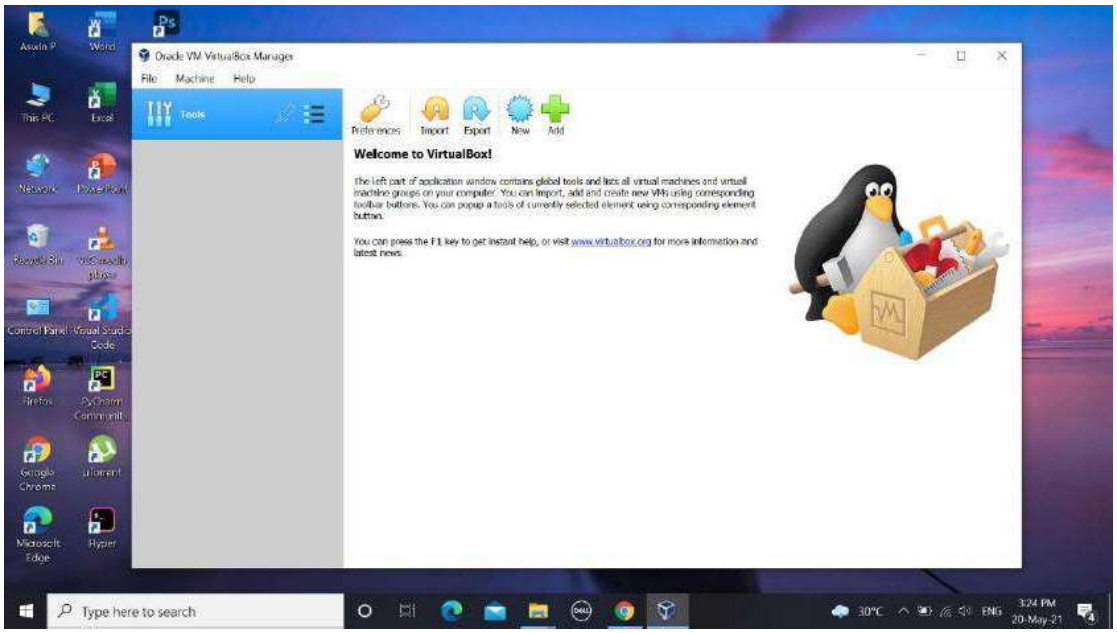
STEP 7:Installing VirtualBox



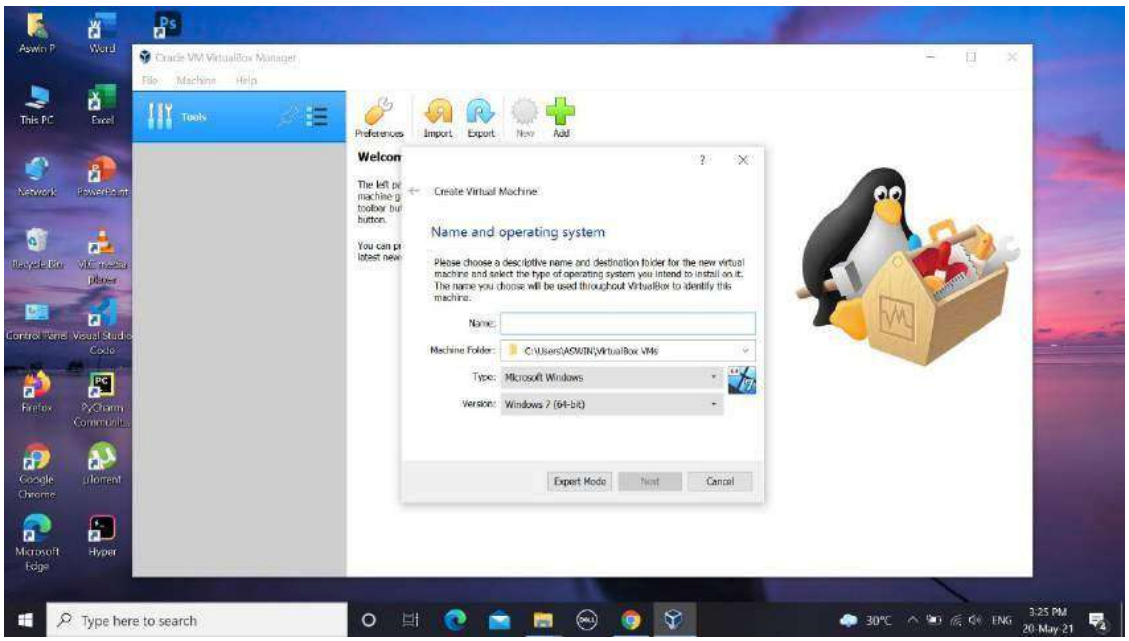
STEP 8: Installation is completed.Then Click on Finish.

VIRTUALBOX SETUP

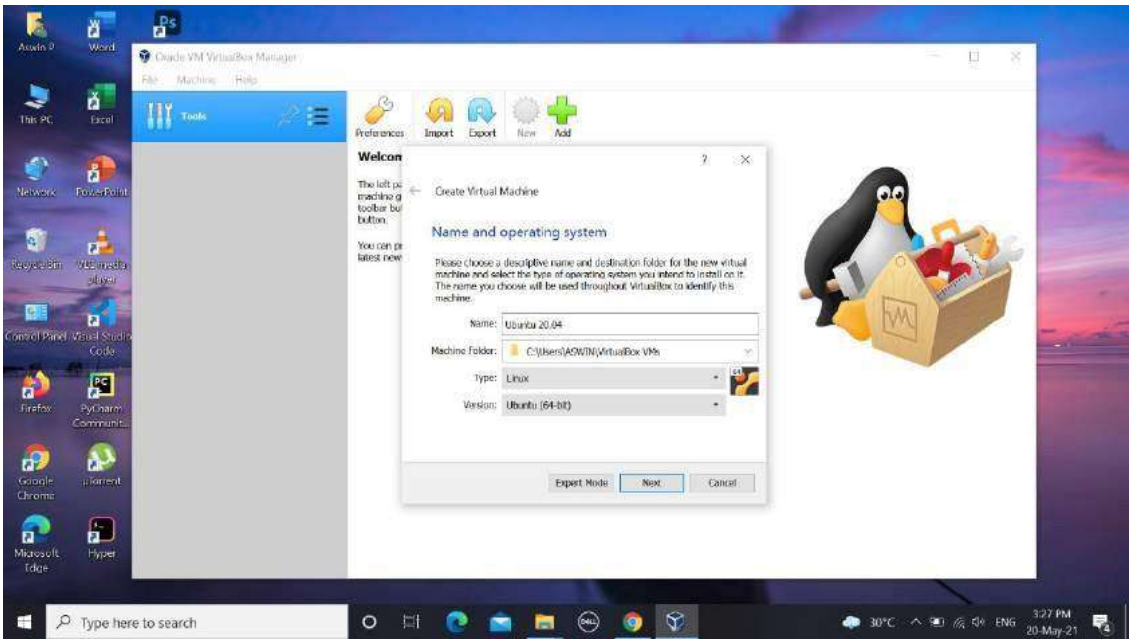
STEP 1: Next step is to Open VirtualBox



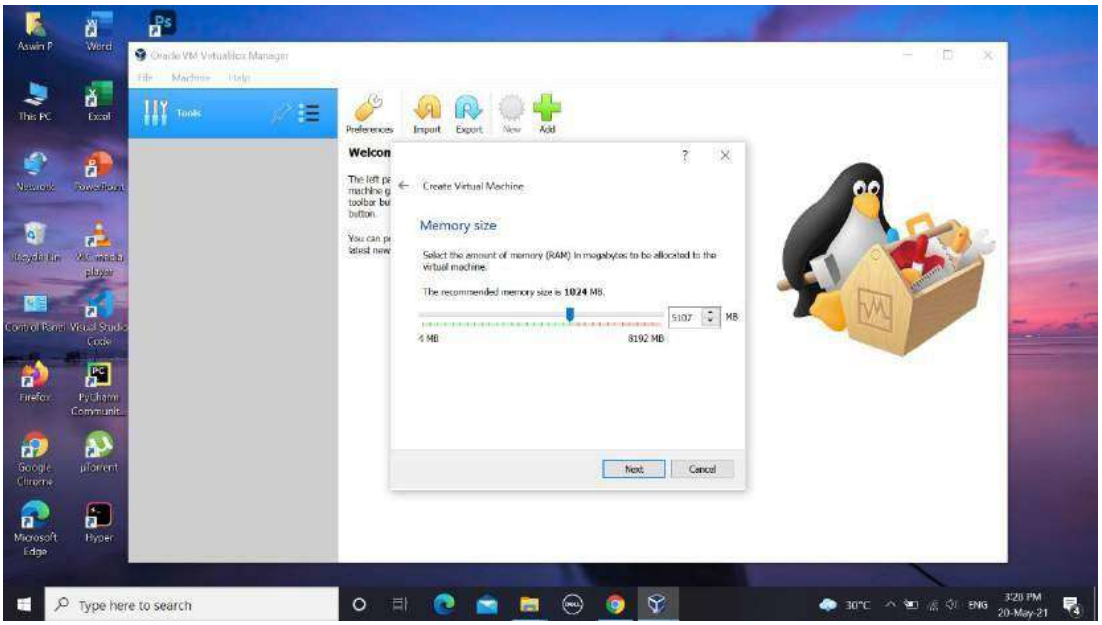
STEP 2: Click 'New' button to open a dialog.



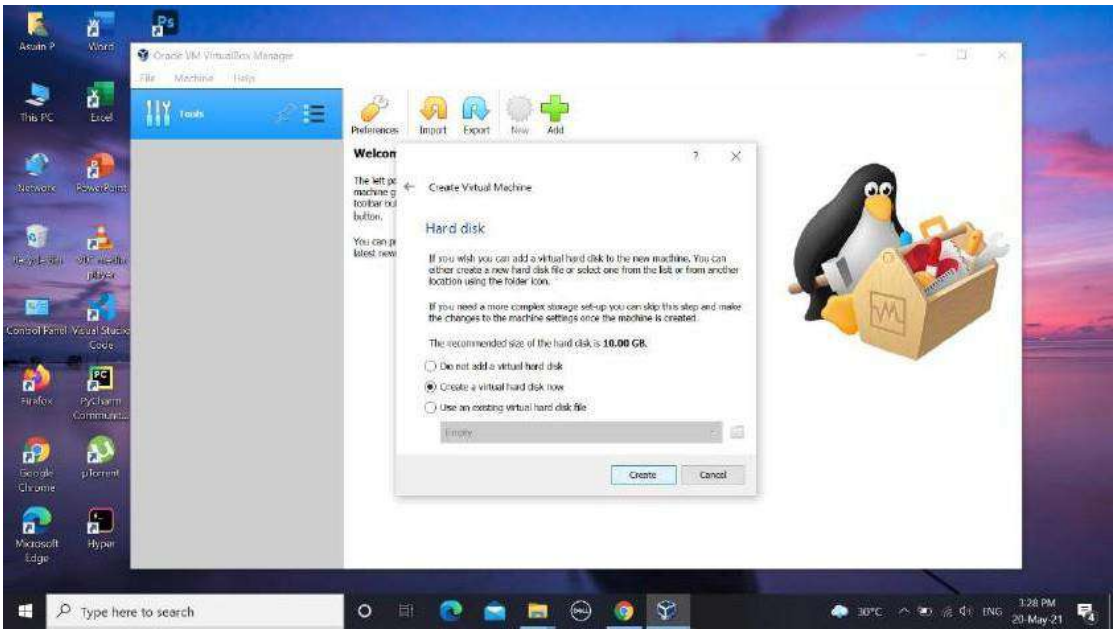
STEP 3 :Type a name for the new virtual machine.
VirtualBox automatically changes 'Type' to Linux and 'Version'Ubuntu .



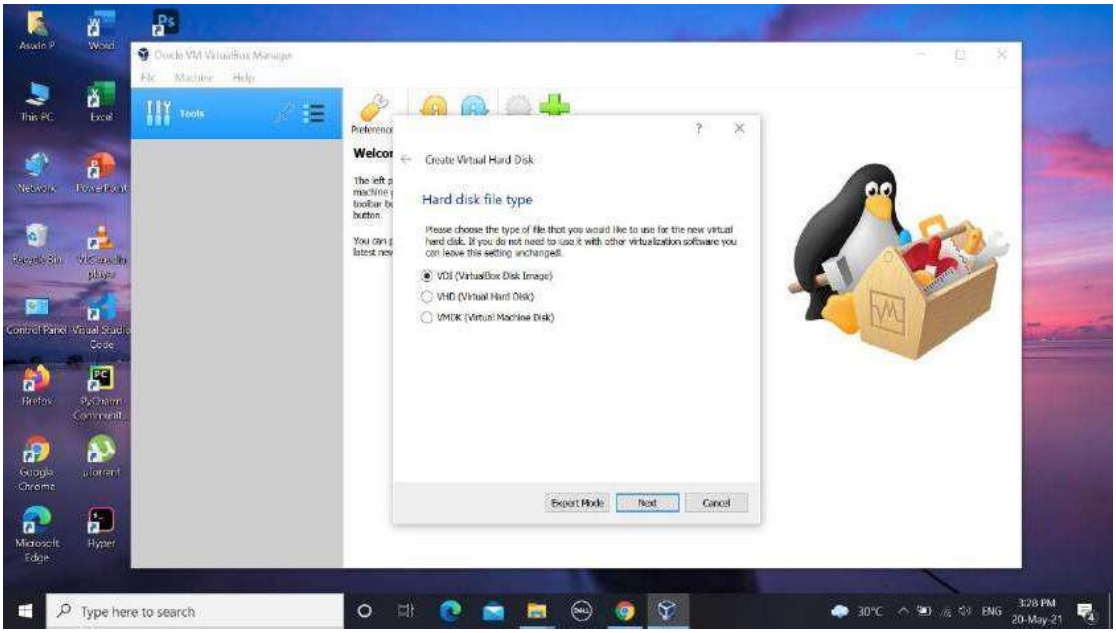
STEP 4:Click Next.Select theRAM to use.
.Click and drag the slider left or right to decrease or increase the amount of RAM .



STEP 5: Accept the default 'Create a virtual hard drive now' and click 'Create' button.

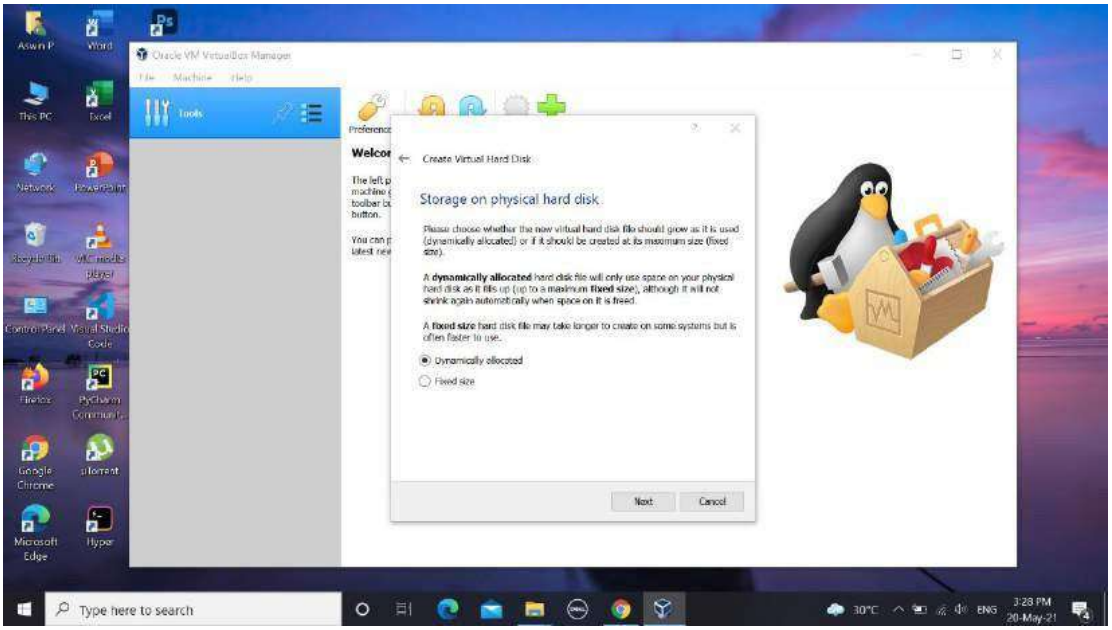


STEP 6:Continue to accept the default 'VDI' drive file type and click 'Next' button



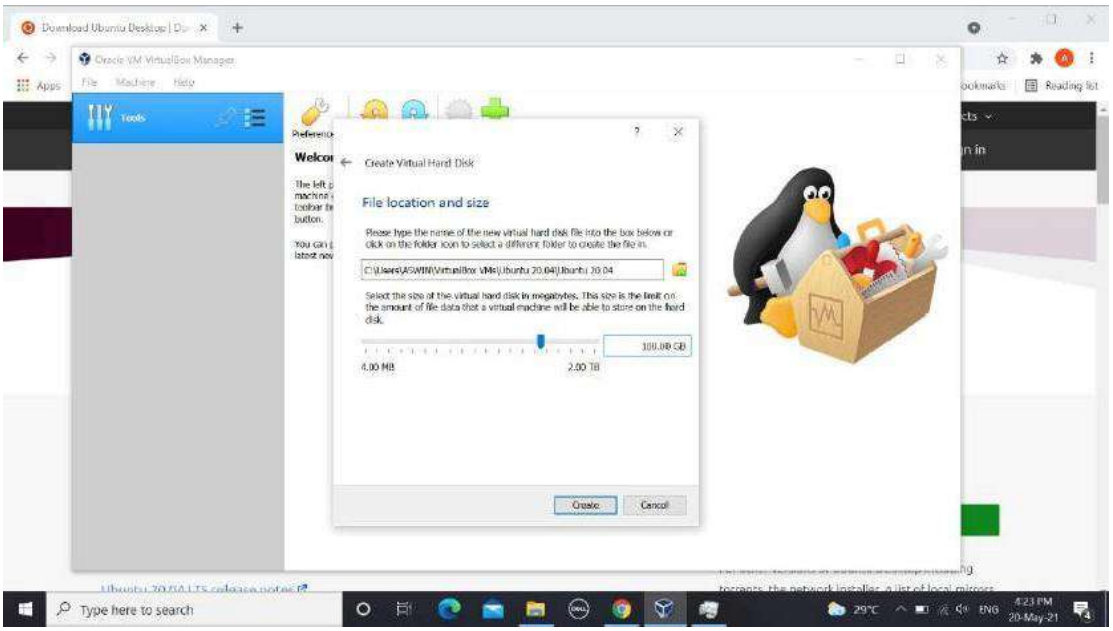
STEP 7: Storage type from the default is 'Dynamically allocated'

Click Next.



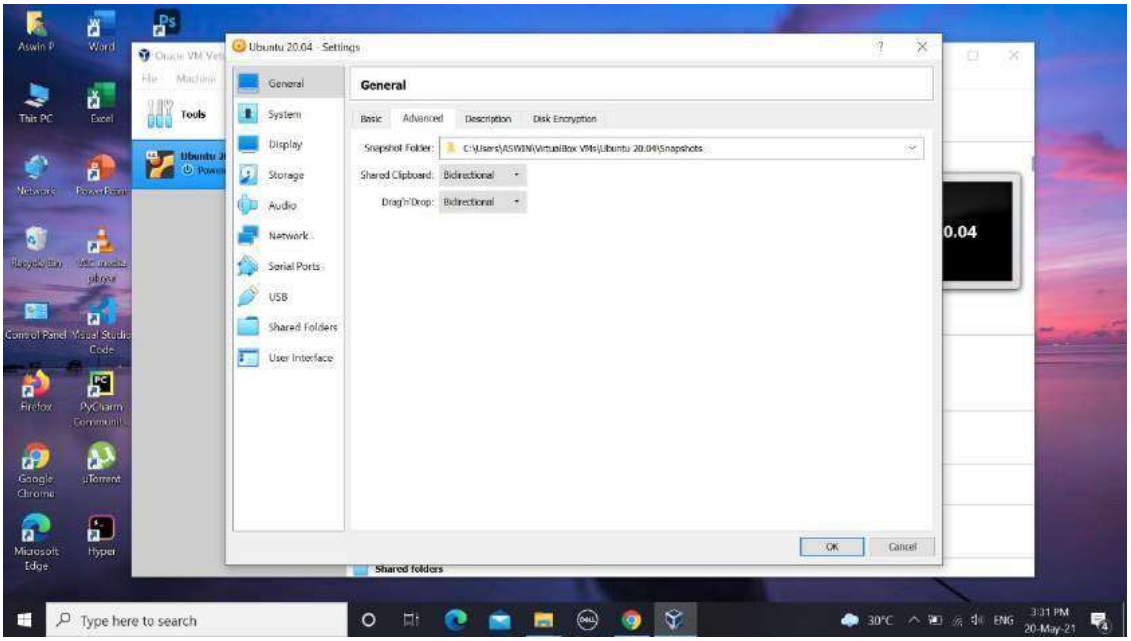
STEP 8: Enter the Size of Virtual Hard Drive.Give the size is equals to 100 GB .

Click Create

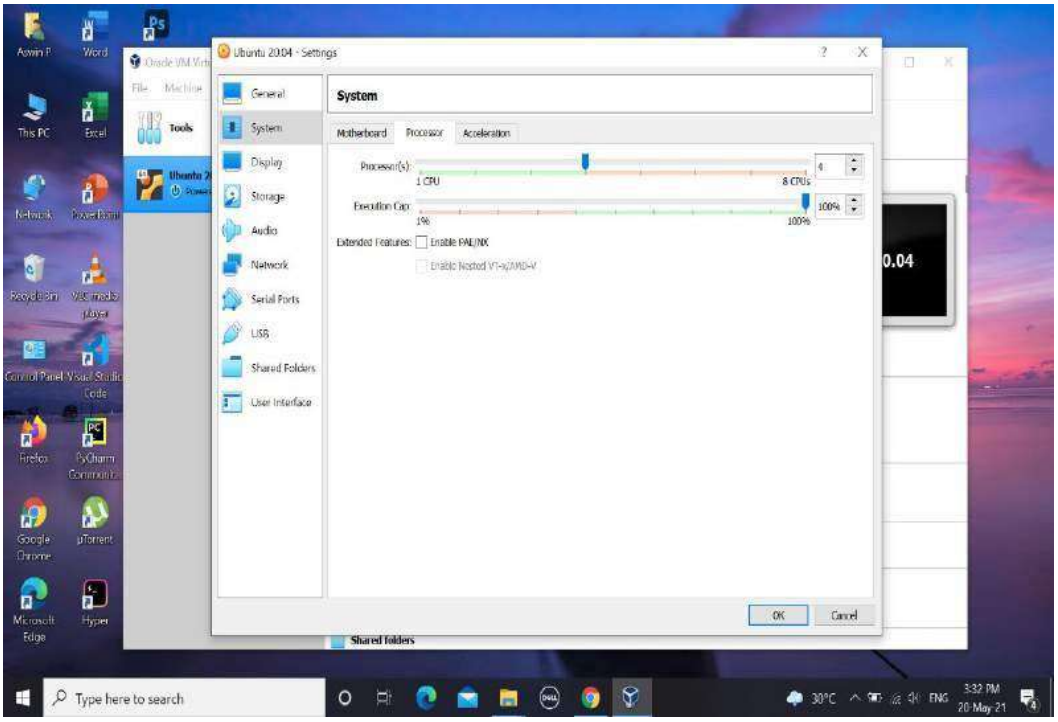


STEP 9: To start the virtual machine click on “Settings”.

Click on “General → Advanced”



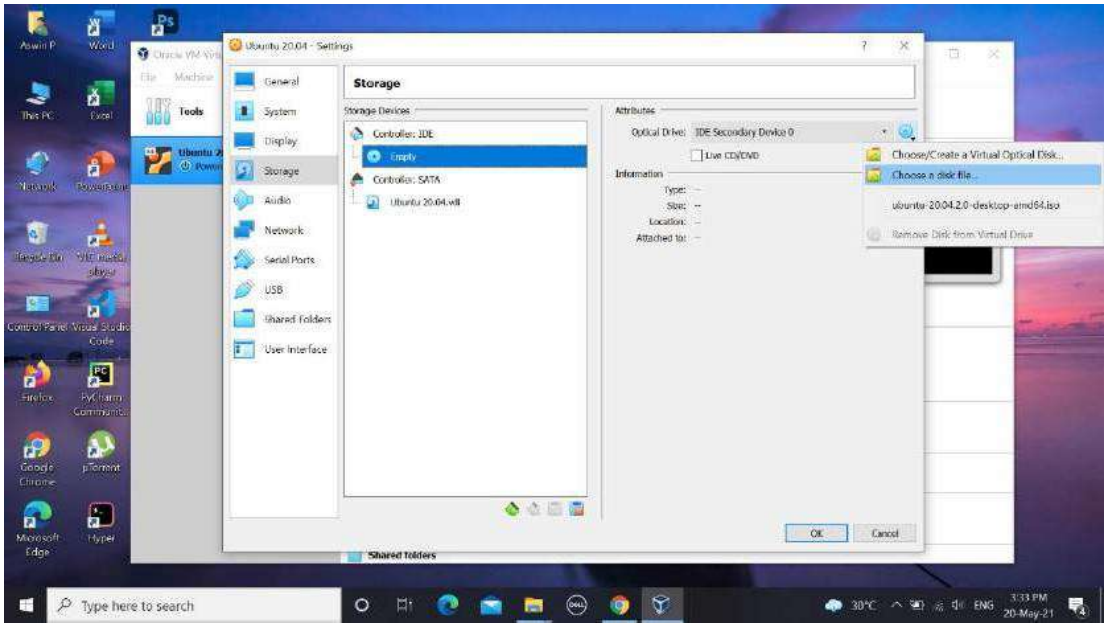
STEP 10: Click “System → Processor”



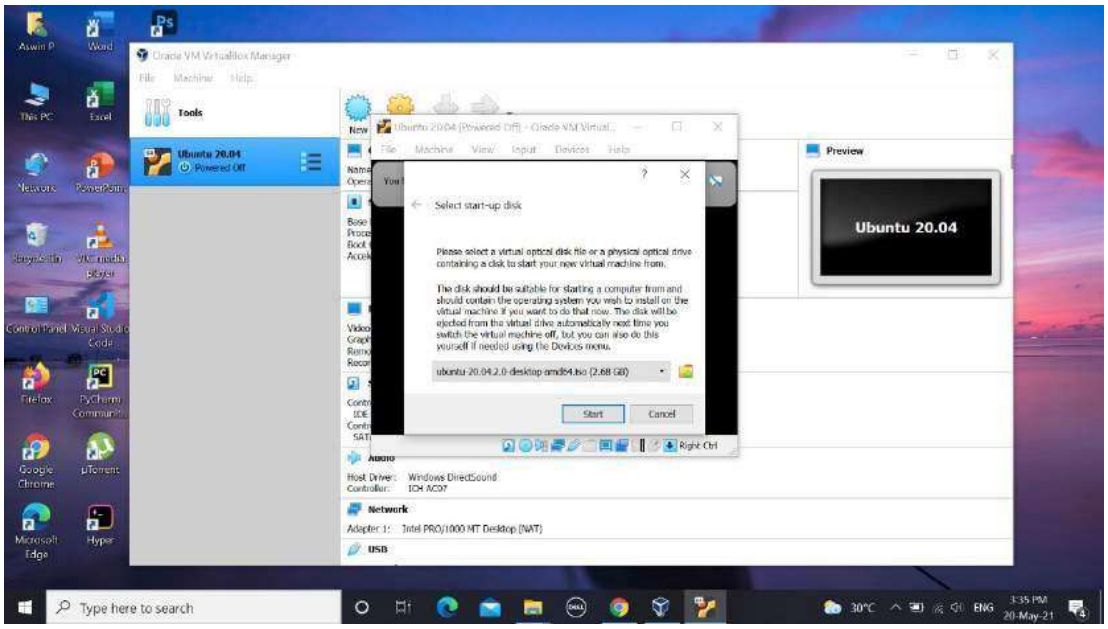
STEP 11: Click on “Storage → Controller IDE”.

Click the disk symbol.

Select disk file source to start an installation of OS.

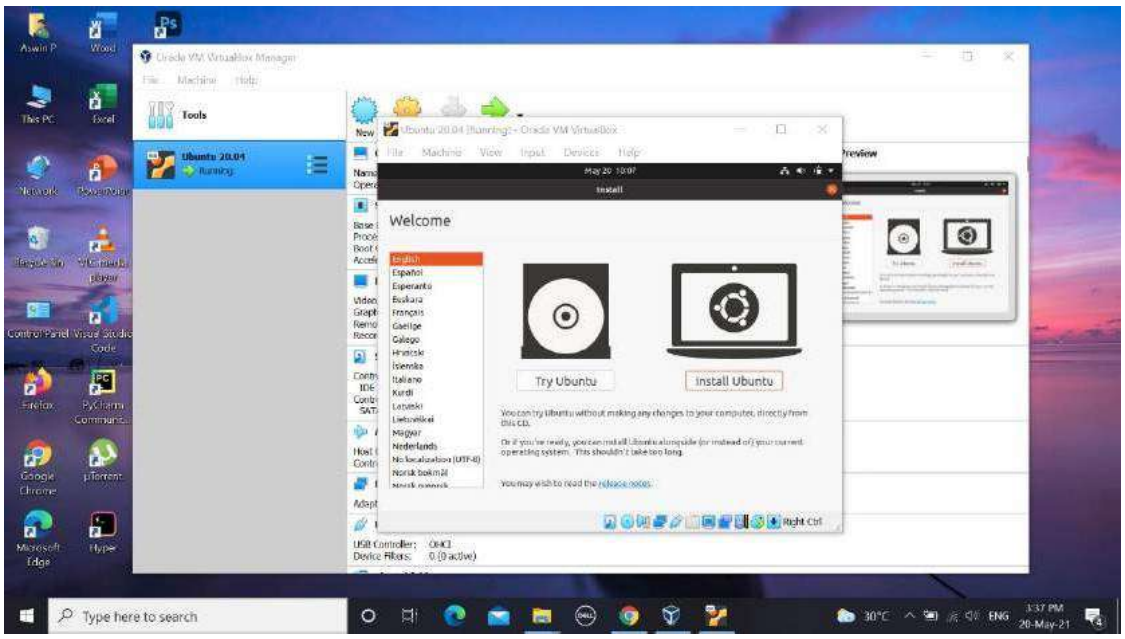


STEP 12: Click “Start”.

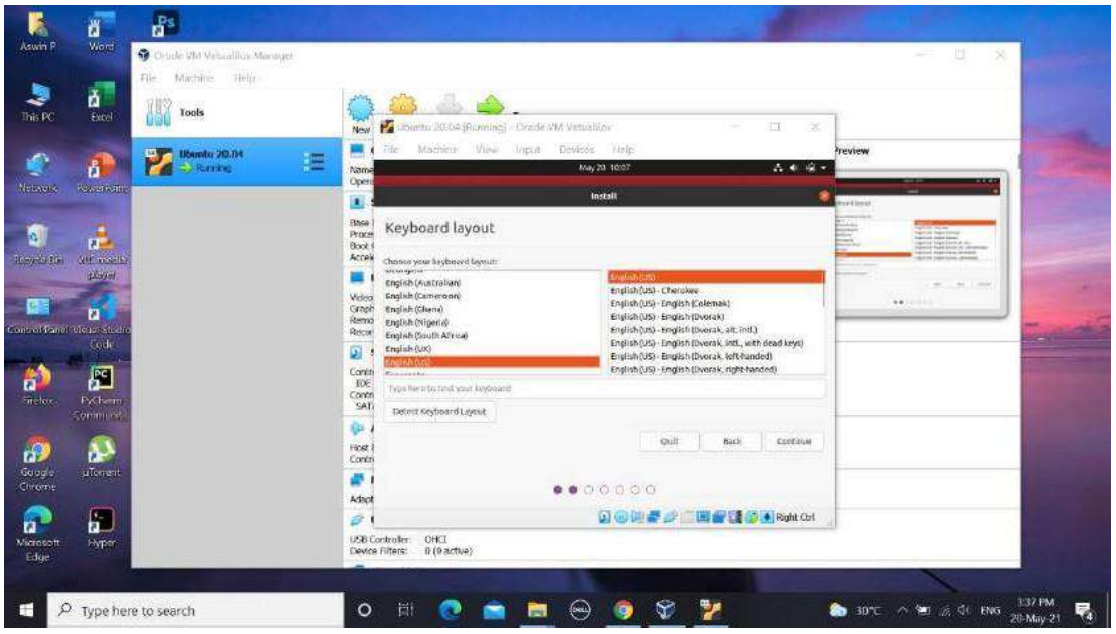


UBUNTU INSTALLATION

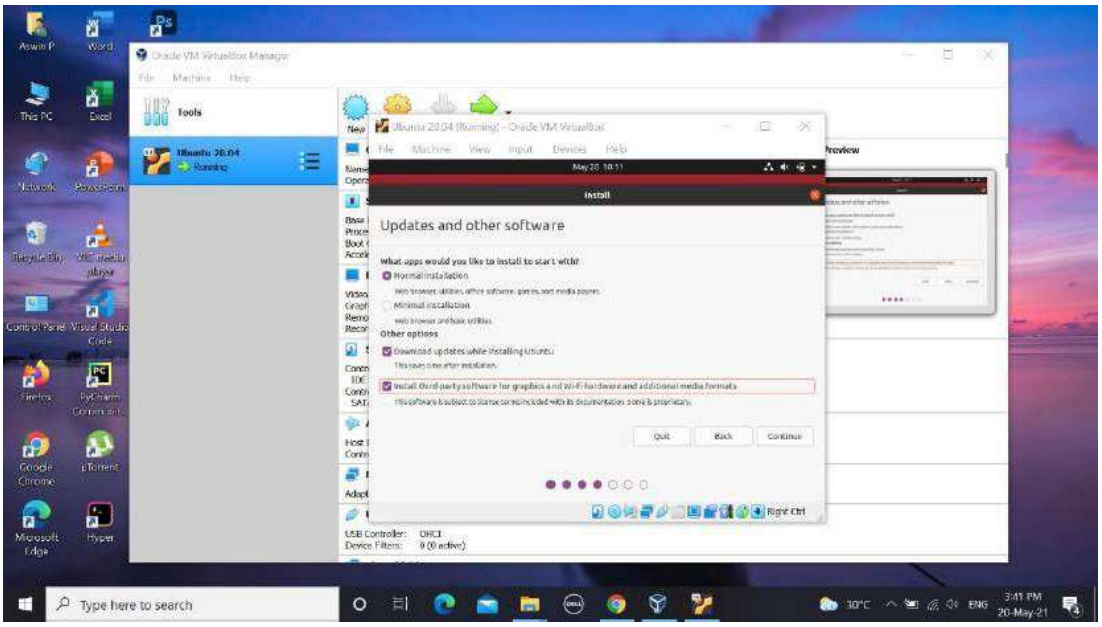
STEP 1: Click “Install Ubuntu”.



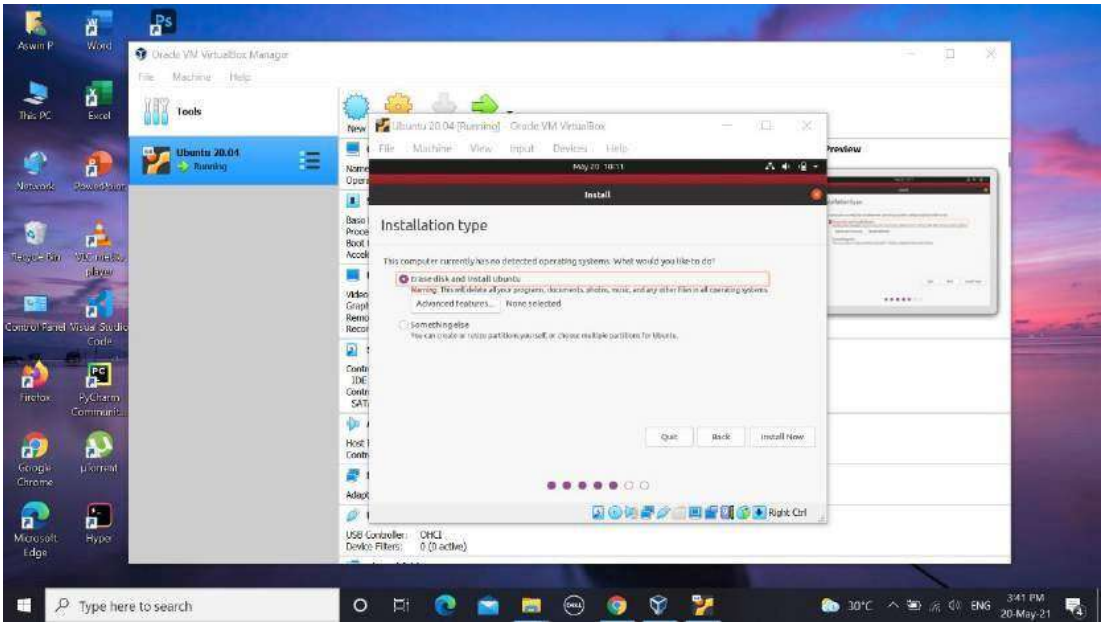
STEP 2: Select Keyboard layout for your machine.



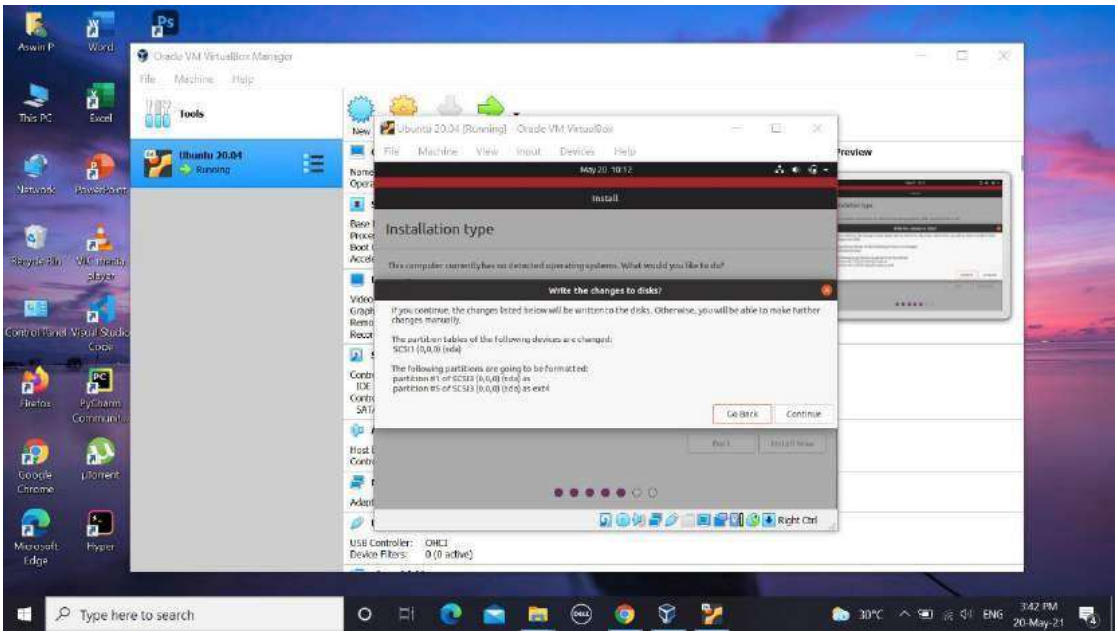
STEP 3:Click continue.



STEP 4: Check Erase disk and install ubuntu.

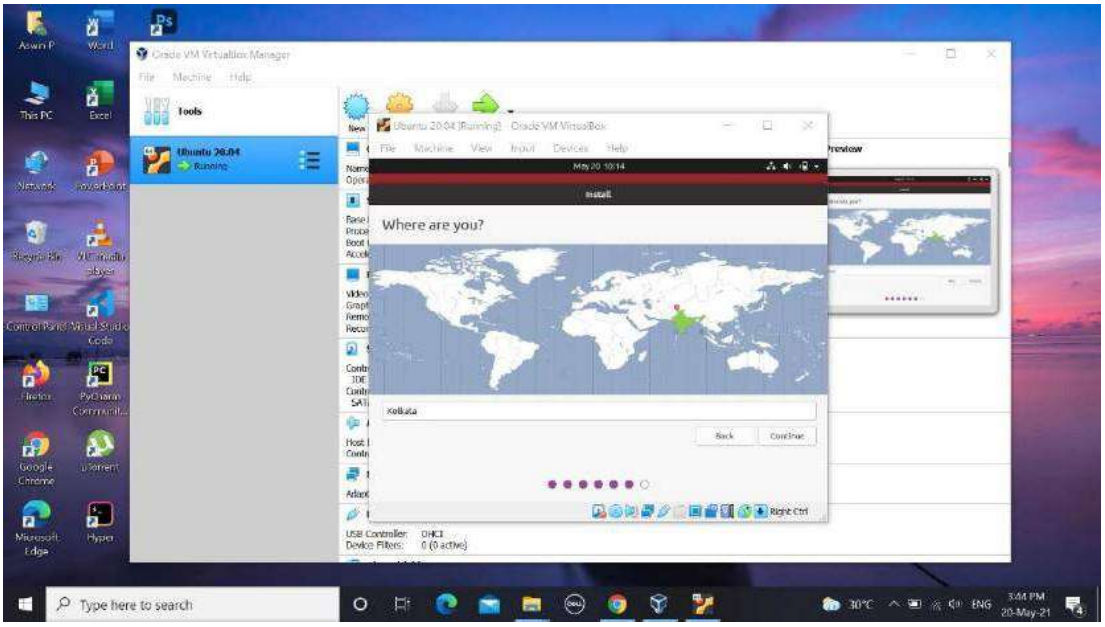


STEP 5:Click Continue.

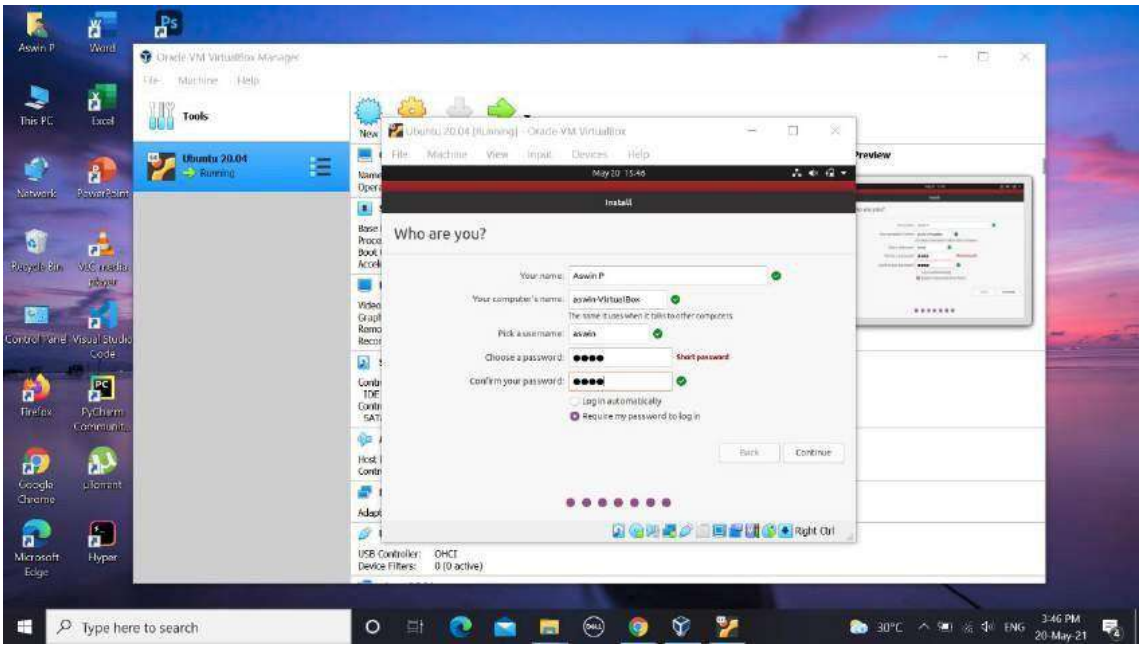


STEP 6: Select location.

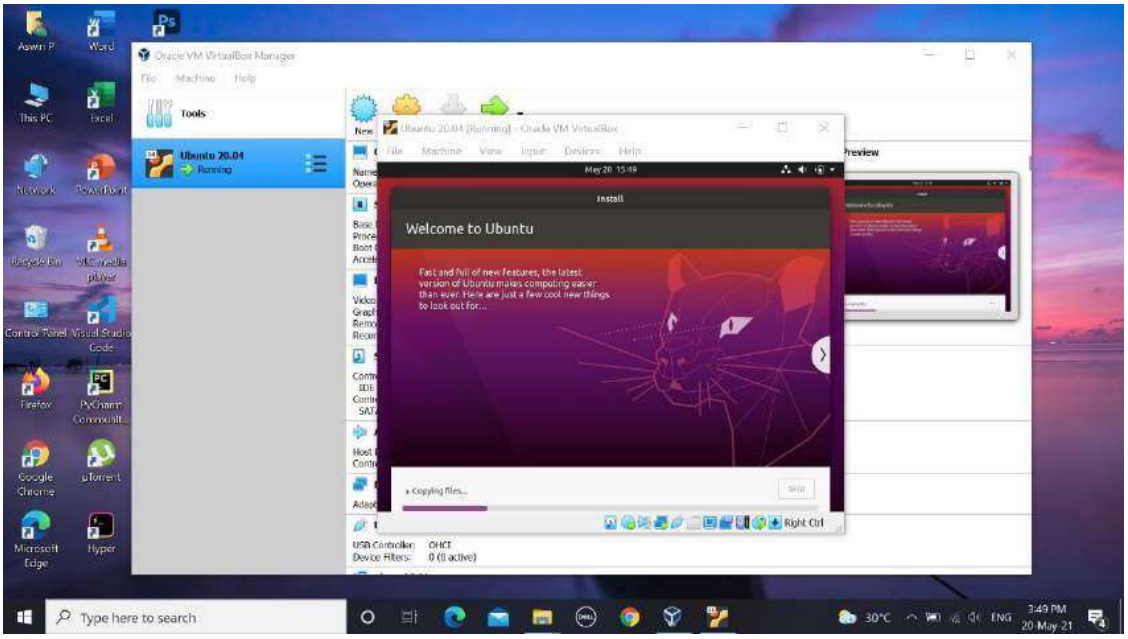
Click continue.



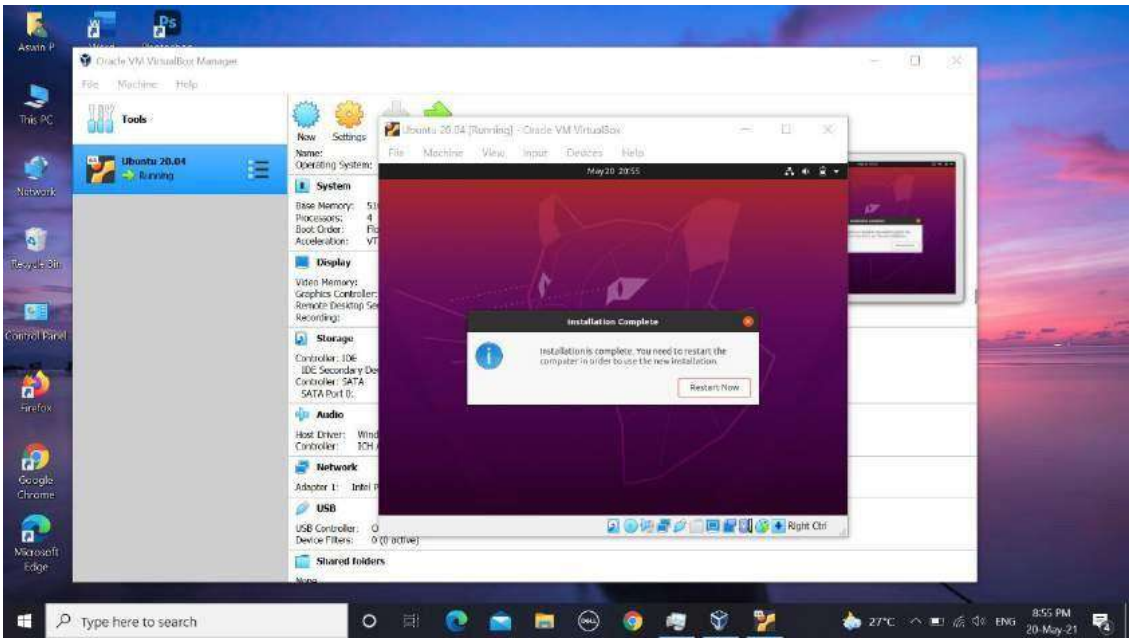
STEP 7: Click “Continue”before that fill the dialogue box.



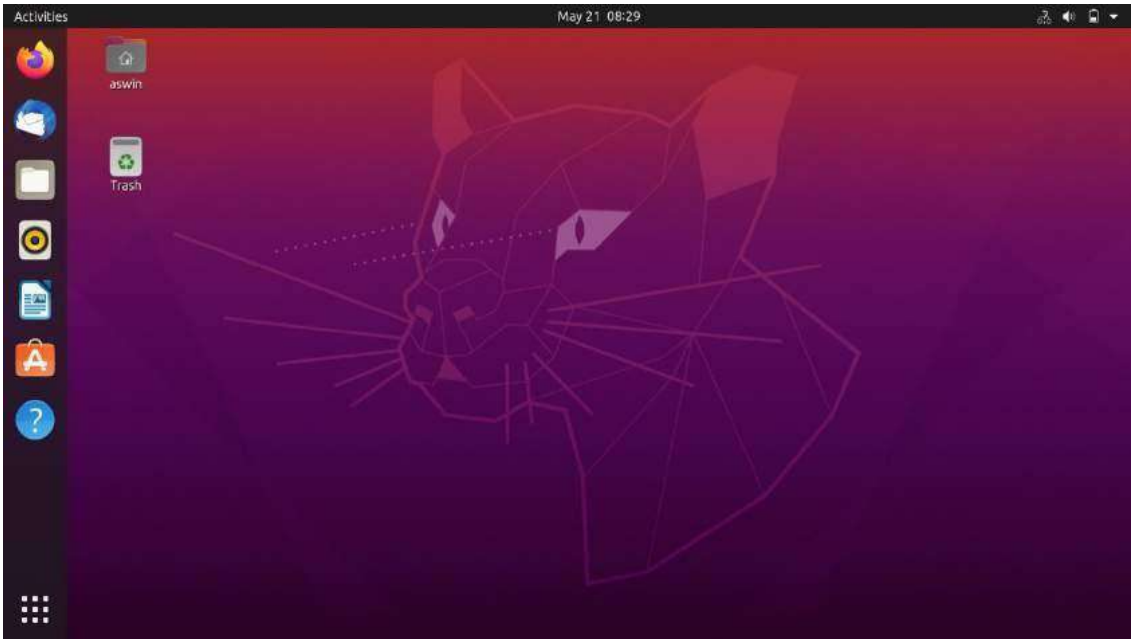
STEP 8: Installation started.



STEP 9: Click “Restart Now”.



STEP 10: Ubuntu is successfully installed.



Congratulations! You have successfully installed the world’s most popular Linux operating system!

It’s now time to start enjoying Ubuntu!!!.....

EXPERIMENT 2:

AIM:- Study of a terminal based text editor such as Vim or Emacs. (By the end of the course, students are expected to acquire following skills in using the editor: cursor operations, manipulate text, search for patterns, global search and replace) Basic Linux commands, familiarity with following commands/operations expected

ANSWER:

The Linux command is a utility of the Linux operating system. All basic and advanced tasks can be done by executing commands. The commands are executed on the Linux terminal. The terminal is a command-line interface to interact with the system, which is similar to the command prompt in the Windows OS. Commands in Linux are case-sensitive.The commands are explained below

1. **pwd** — When you first open the terminal, you are in the home directory of your user. To know which directory you are in, you can use the "pwd" command. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash(/). The user directory is usually something like "/home/username".

```
nayso@Alok-Aspire:~$ pwd
/home/nayso
```

2. **ls** — Use the "ls" command to know what files are in the directory you are in. You can see all the hidden files by using the command "ls -a".

```
nayso@Alok-Aspire:~$ ls
Desktop           itsuserguide.desktop  reset-settings     VCD_Copy
Documents         Music                 School_Resources   Videos
Downloads         Pictures              Students_Works_10
examples.desktop  Public                Templates
GplatesProject    Qgis Projects         TuxPaint-Pictures
```

3. **cd** — Use the "cd" command to go to a directory.

```
nayso@Alok-Aspire:~$ cd Downloads
nayso@Alok-Aspire:~/Downloads$ cd
nayso@Alok-Aspire:~$ cd Raspberry\ Pi
nayso@Alok-Aspire:~/Raspberry Pi$ cd ..
nayso@Alok-Aspire:~$
```

4. **mkdir & rmdir** — Use the mkdir command when you need to create a folder or a directory.

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ mkdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
DIY
nayso@Alok-Aspire:~/Desktop$ rmdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

5. **rm** - Use the rm command to delete files and directories. Use "rm -r" to delete just the directory. It deletes both the folder and the files it contains when using only the rm command.

```
nayso@Alok-Aspire:~/Desktop$ ls
newer.py  New Folder
nayso@Alok-Aspire:~/Desktop$ rm newer.py
nayso@Alok-Aspire:~/Desktop$ ls
New Folder
nayso@Alok-Aspire:~/Desktop$ rm -r New\ Folder
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

6. **touch** — The touch command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example, "touch new.txt".

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ touch new.txt
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
```

7. **man & --help** — To know more about a command and how to use it, use the man command. It shows the manual pages of the command. For example, "man cd" shows the manual pages of the cd command. Typing in the command name and the argument helps it show which ways the command can be used (e.g., cd –help).

```
TOUCH(1)                        User Commands                        TOUCH(1)

NAME
       touch - change file timestamps

SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times of each FILE to the current
       time.

       A FILE argument that does not exist is created empty, unless -c  or  -h
       is supplied.

       A  FILE  argument  string of - is handled specially and causes touch to
       change the times of the file associated with standard output.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a     change only the access time

 Manual page touch(1) line 1 (press h for help or q to quit)
```

8. **cp** — Use the cp command to copy files through the command line. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

```
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ cp new.txt /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
new.txt
```

9. **mv** — Use the mv command to move files through the command line. We can also use the mv command to rename a file. For example, if we want to rename the file
"text" to "new", we can use "mv text new". It takes the two arguments, just like the cp command.

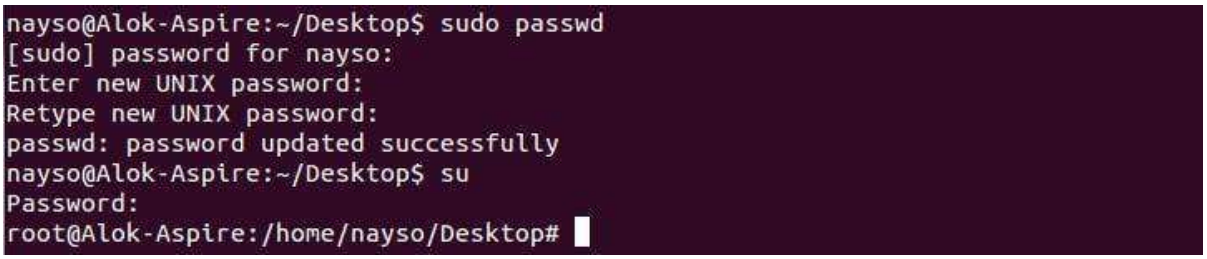
```
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
nayso@Alok-Aspire:~/Desktop$ mv new.txt newer.txt
nayso@Alok-Aspire:~/Desktop$ ls
newer.txt
```

10. **echo** — The "echo" command helps us move some data, usually text into a file. For example, if you want to create a new text file or add to an already made text file, you just need to type in, "echo hello, my name is alok >> new.txt". You do not need to separate the spaces by using the backward slash here, because we put in two triangular brackets when we finish what we need to write.

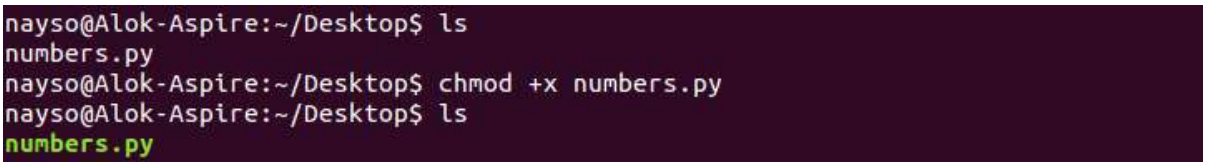
11. **cat** — Use the cat command to display the contents of a file. It is usually used to easily view programs.

```
nayso@Alok-Aspire:~/Desktop$ echo hello, my name is alok >> new.txt
nayso@Alok-Aspire:~/Desktop$ cat new.txt
hello, my name is alok
nayso@Alok-Aspire:~/Desktop$ echo this is another line >> new.txt
nayso@Alok-Aspire:~/Desktop$ cat new.txt
hello, my name is alok
this is another line
```

12. **sudo** — A widely used command in the Linux command line, sudo stands for "SuperUser Do".


nayso@Alok-Aspire:~/Desktop$ sudo passwd
[sudo] password for nayso:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
nayso@Alok-Aspire:~/Desktop$ su
Password:
root@Alok-Aspire:/home/nayso/Desktop#

13. **chmod** — Use chmod to make a file executable and to change the permissions granted to it in Linux. Imagine you have a python code named numbers.py in your computer. You'll need to run "python numbers.py" every time you need to run it. Instead of that, when you make it executable, you'll just need to run "numbers.py" in the terminal to run the file. To make a file executable, you can use the command "chmod +x numbers.py" in this case. You can use "chmod 755 numbers.py" to give it root permissions or "sudo chmod +x numbers.py" for root executable. Here is some more information about the chmod command.



VIM TEXT EDITOR ON LINUX UBUNTU

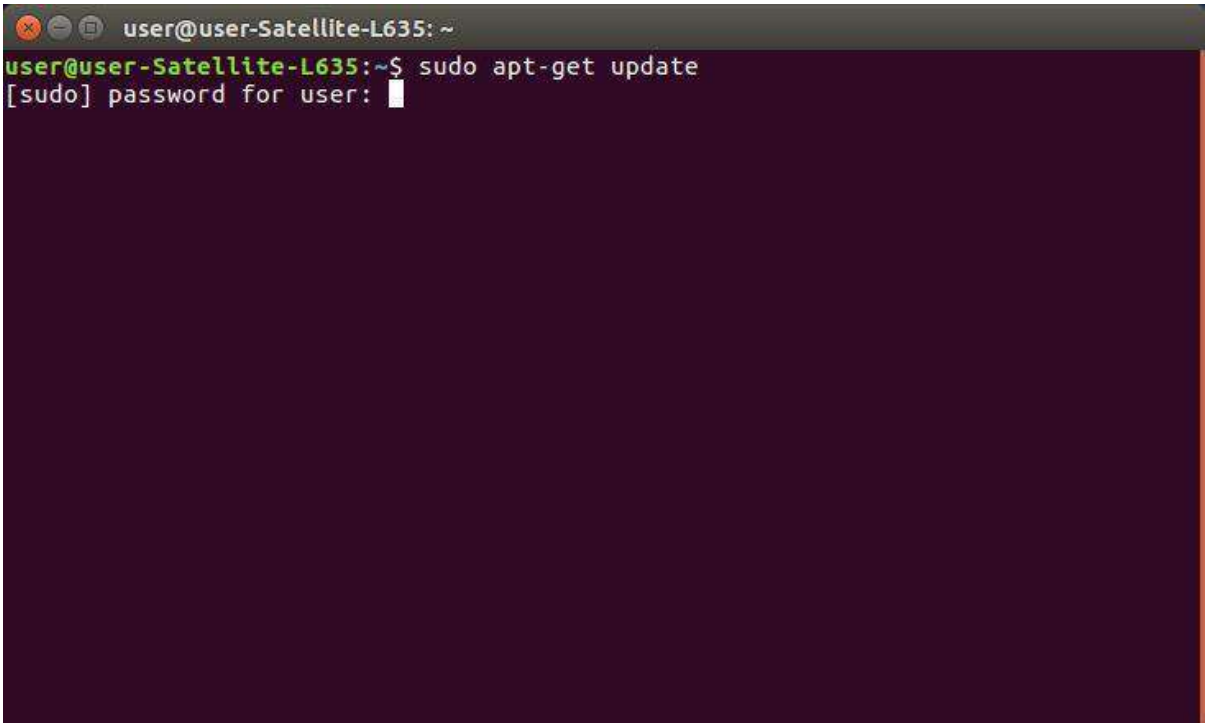
Vim is a very useful and helpful editor for creating and editing different types of files more efficiently. Many new features are added in this editor that makes it a powerful editor. Many plugins are developed by many coders for this editor to increase and configure its core functionalities. Some of them are Pathogen, Syntastic, indent guides, Fugitive, Git Gutter, etc. Git is a distributed version control system (DVCS) that helps the developers to manage the modified source codes over time. It is totally free to use. Using git command, the track changes and the revision history of the source codes can be easily traced. Git command works in the command line interface. The vim plugin named fugitive plugin is developed by Tim pope which is used to work with the git tool without terminating the editor. So, vim and git can work together by using the fugitive plugin.

INSTALLATION OF VIM

Step 1: First open the terminal application and then update package database using the apt command or apt-get command:

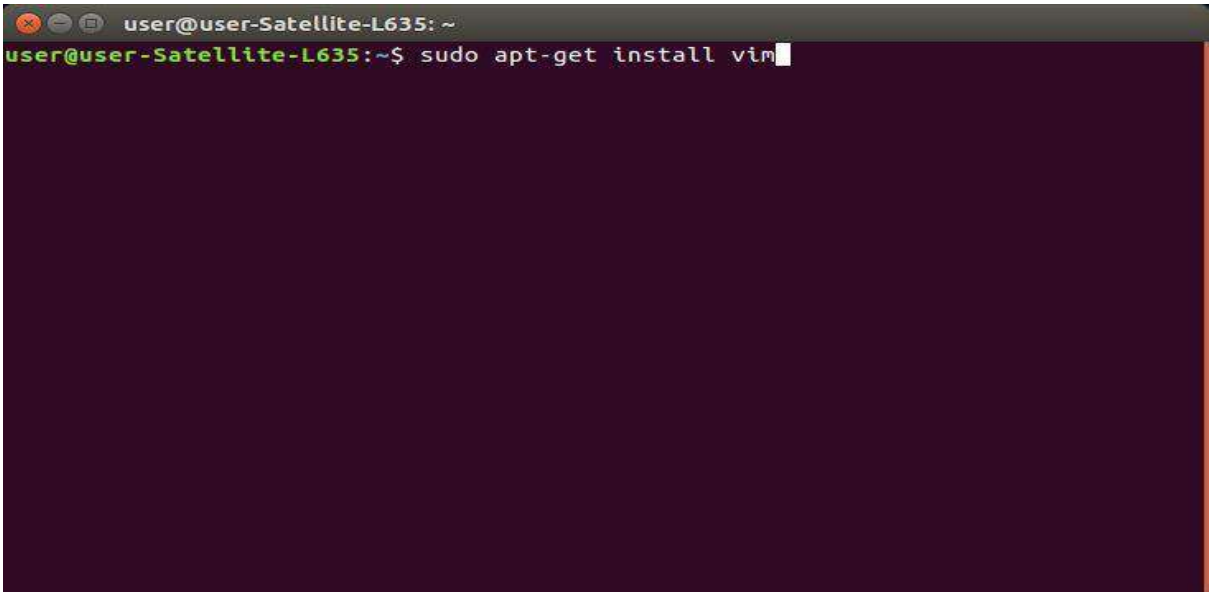
sudo apt-get update

Next enter the password for the linux.

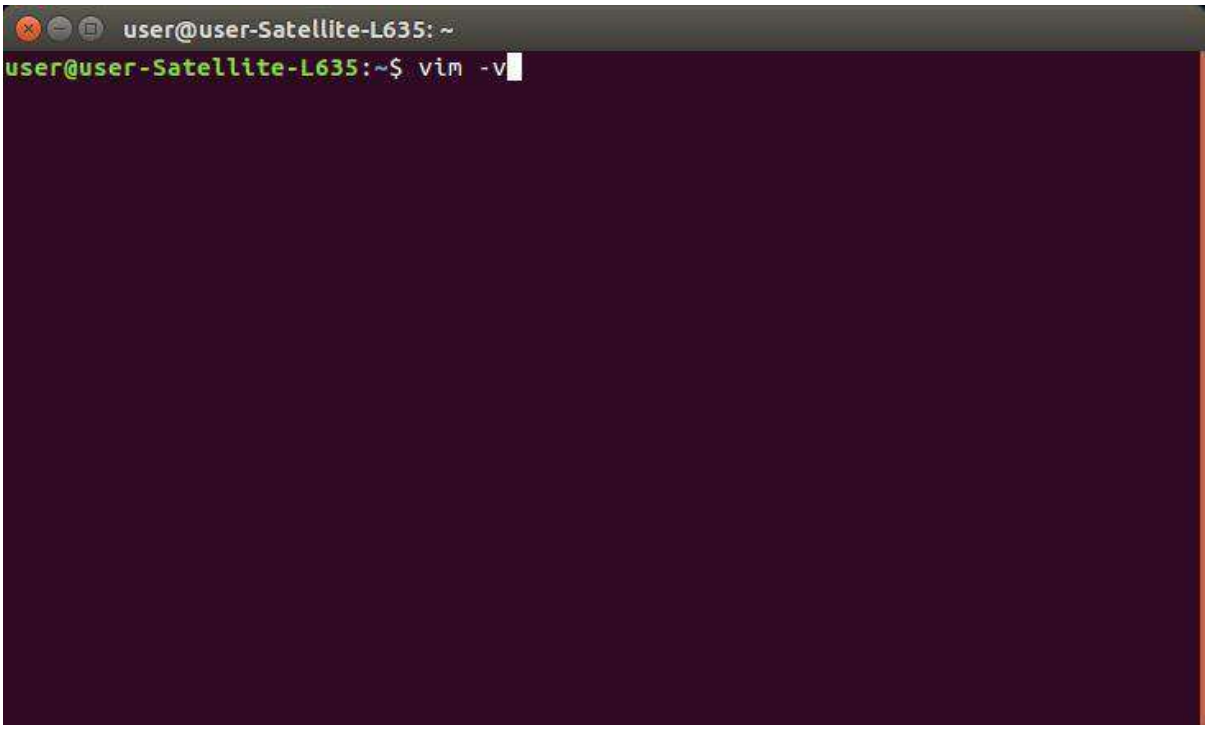


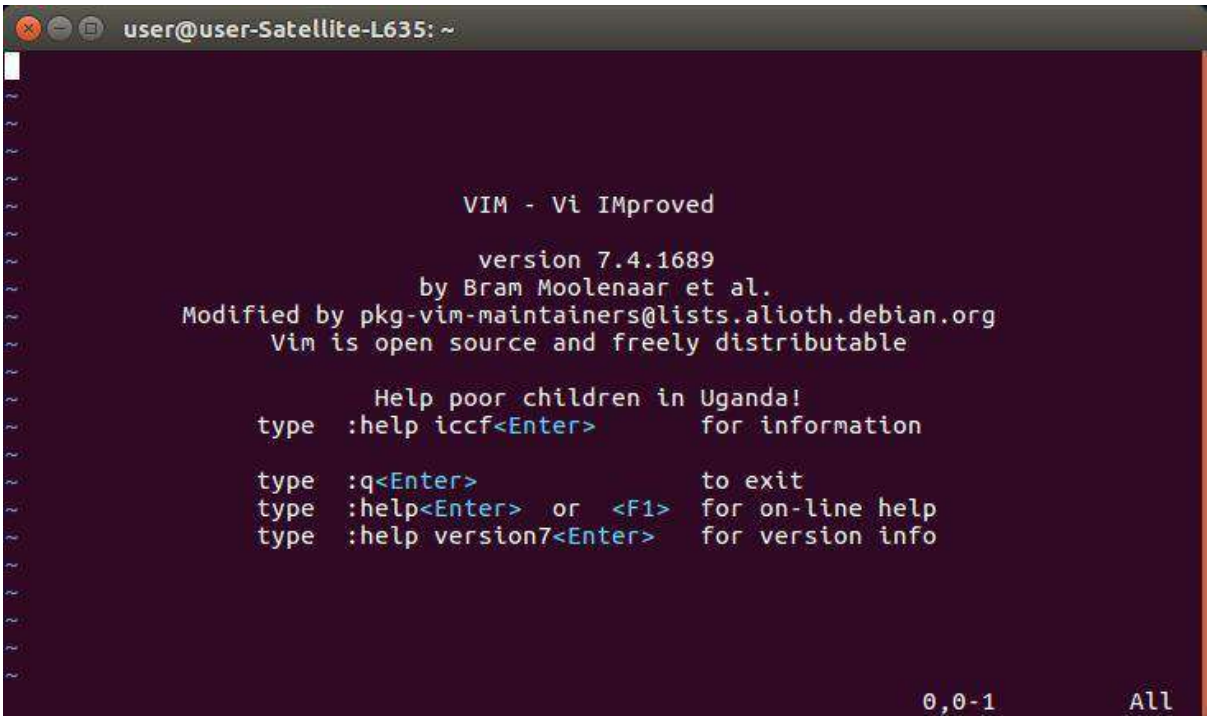
Step 2:Type the following apt-get command to install vim text editor (when promoted type your own password):

sudo apt-get install vim



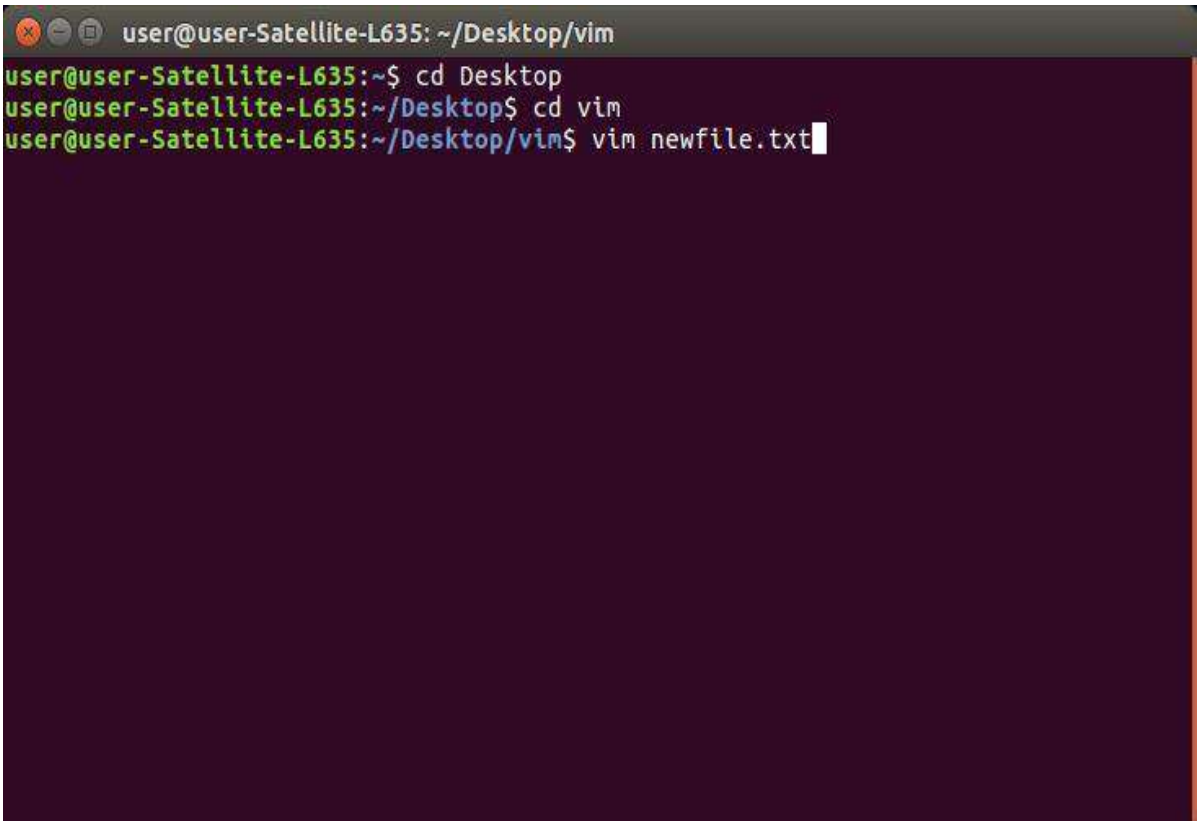
Step 3: Type the command ‘’vim -v’’





Step 4:Create a folder named vim in your desktop.

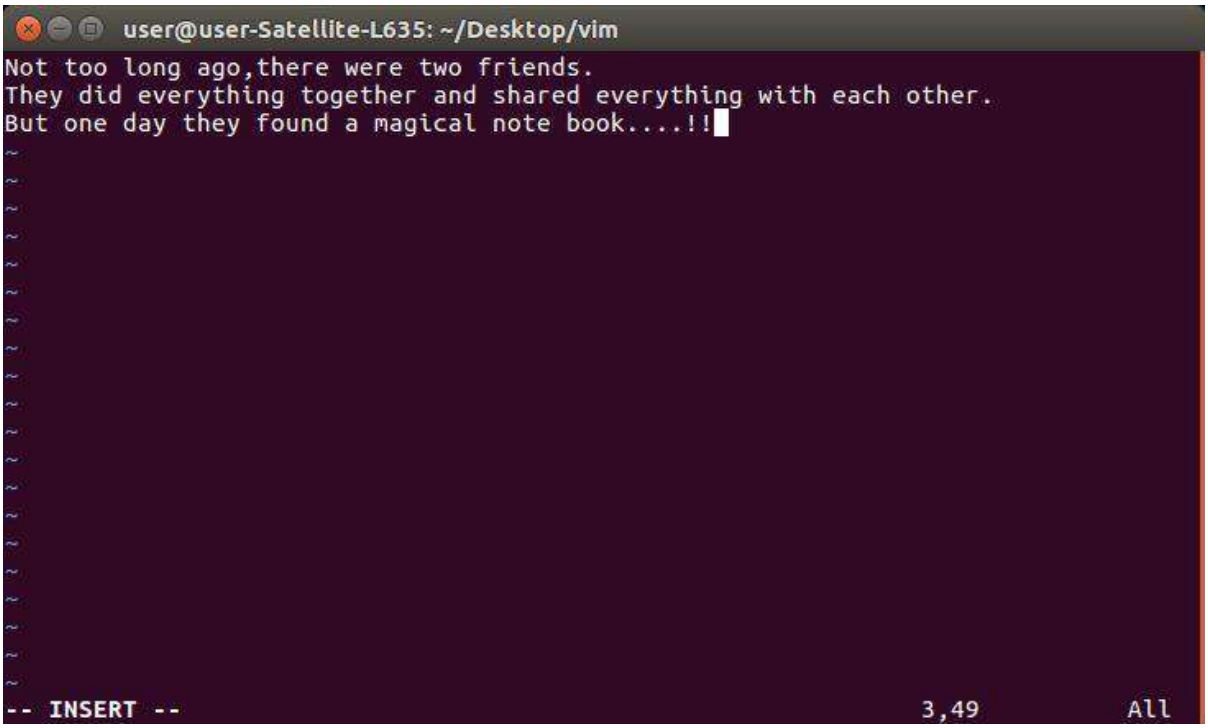
Go to that folder using cd command.



Step 5: Creating a document with Vim.To create a document and start editing it, just run the vim command followed by the file name:

vim my_file

When executing the command above, the “my_file” file will be generated (the extension hasn’t been indicated but can optionally be defined, i for example, by typing: vim newfile.txt).An editor, where writing the content of the document, will immediately open.By using insert mode “i” we can insert text to the editor.



Once you open the document, start writing the content. Once you have finished writing, press "Esc" and then type ": wq". Press "Enter" and your file will be saved in the path where you started the command (the meaning of ": wq" will be explained later).

This is the easiest way to use Vim as a text editor, but this tool has much more potential to show! !

VIM TEXT EDITOR MODES

The main difference between Vim and other text editors is that this tool can be considered as a 'modal' editor: depending on how you are using Vim, the same key combination can take on different functions. There are three modes:

i .Normal ii

.Insert

iii.Command-line.

Exit from any mode by pressing the ESC key, through which you will enter the "command line" mode.

i. Normal mode

With the normal mode it is possible to edit a text or navigate through the lines. In this mode, by pressing the h , j , k , l keys, you move through the text just as if you were pressing the directional arrows on the keyboard. More precisely:

h: move the cursor to the left j:

move the cursor down

k: move the cursor up

l: move the cursor to the right

Each command can be executed by placing a number next to it. For example the 3k command will move the cursor 3 lines up.

ii. Insert mode

The insert mode is the one through which Vim is used as a normal text editor, thus allowing you to add text, delete it, etc. Also, in this mode there are key combinations for inserting text in different points of the document.

To exit the entry mode, simply press the "ESC" key. To enter

this mode, just type the command: i . **iii.Command line**

mode

From this mode more complex commands, such as saving the changes made to the document or even closing Vim, can be executed.

These commands must be preceded by : (colon). Again, macros (combinations of commands) to be executed in series can be created.For example, if you want to save and close a document, use the command : wq where:

"W" (write) represents the request to write the changes made (save) "Q" (quit) is

used to close the document.

These commands can be combined to use the potential of this editor.

VIM COMMANDS

Basic commands

Esc - Exits current mode into the “command mode” i - Exists

current mode into the “insert mode”

:help <keyword> - Searches the Help documentation for your keyword

:w - Saves your file

:wq - Saves and closes your file

:q - Closes your file

ZZ - Saves your file and exits Vim

Navigating commands

h - Moves the cursor to the left

j - Moves the cursor down one line k -

Moves the cursor up one line

l - Moves the cursor to the right

10j - Moves the cursor down 10 lines

H - Moves cursor to the top line on the screen

M - Moves cursor to the middle line on the screen L -

Moves cursor to the bottom line on the screen

w - Moves the cursor to the beginning of the next word

b - Moves the cursor to the beginning of the previous word e - Moves

the cursor to the end of the current word

gg - Moves cursor to the first line of the file G -

Moves cursor to the last line of the file

0 - Moves cursor to the beginning of the current line

- This command takes you to line #, where # is specified by you

Editing commands

i - Insert before the current character a -

Insert after the current character

o - Insert a line below the current line, then enter insert mode O - Insert a line above the current line, then enter insert mode s - Delete character at cursor and insert

S - Delete line at cursor and insert

. - Repeat last command

r - Replace one character and return to command mode u - Undo

Searching commands

/<keyword> - Searches document for where the keyword is

/word - Finds the next instance of ‘word’

* - Finds the next instance of the current word

- Finds the previous instance of the current word

n - Searches your text again in the direction of the last search

Working with multiple files

:bn - Moves to next buffer

:bp - Moves to previous buffer

:bd - Deletes a buffer

:sp <filename> - Opens a file in a new buffer and splits screen horizontally

:vsp <filename> - Opens a new file in a new buffer and splits the screen vertically

ctrl + ws - Split windows

ctrl + ww - Switch between windows ctrl + wq - Quit

a window

ctrl + wv - Split windows vertically

Marking Commands

v - Starts visual mode, marks lines, and runs a command V - Starts linewise

visual mode

o - Moves to the other end of marked area ctrl

+ v - Starts visual block mode

aw - Marks a word

What makes Vim so popular is that it’s incredibly functional without having to use a mouse. As you can see, there are tons of keyboard shortcuts that Vim offers.....

EXPERIMENT :-3

AIM : File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc, familiarizing log files for system events, user activity, network events

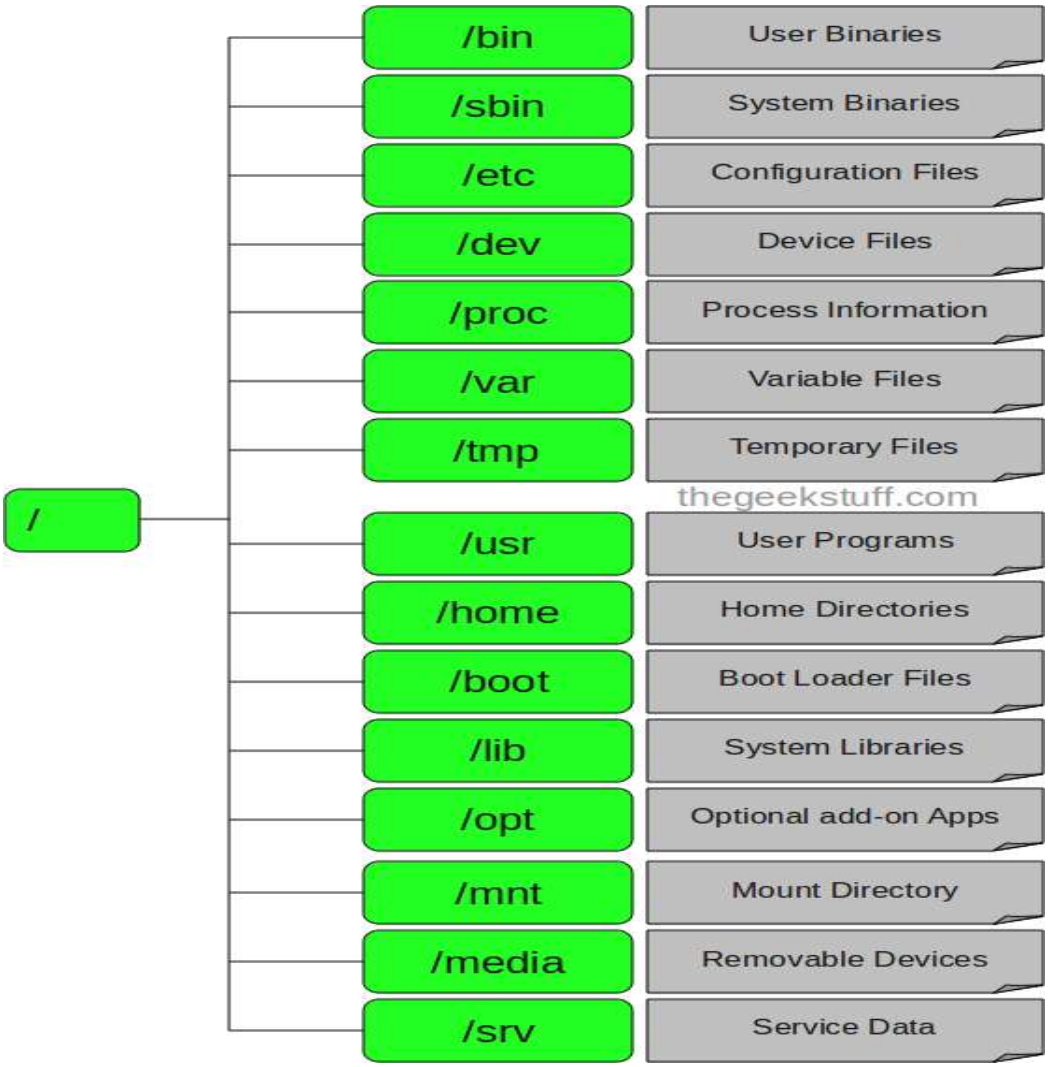
Linux File Hierarchy Structure

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.

Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.


/
/bin
User Binaries
/sbin
System Binaries
/etc
Configuration Files
/dev
Device Files
/proc
Process Information
/var
Variable Files
/tmp
Temporary Files
thegeekstuff.com
/usr
User Programs
/home
Home Directories
/boot
Boot Loader Files
/lib
System Libraries
/opt
Optional add-on Apps
/mnt
Mount Directory
/media
Removable Devices
/srv
Service Data

1. / – Root

Every single file and directory starts from the root

directory. Only root user has write privilege under this

directory.

Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

Contains binary

executables.

Common linux commands you need to use in single-user modes are located under this

directory. Commands used by all the users of the system are located here.

For example: ps, ls, ping, grep, cp.

3. /sbin – System Binaries

Just like /bin, /sbin also contains binary executables.

But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.

For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

Contains configuration files required by all programs.

This also contains startup and shutdown shell scripts used to start/stop individual

programs. For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

Contains device files.

These include terminal devices, usb, or any device attached to the

system. For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

Contains information about system process.

This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.

This is a virtual filesystem with text information about system resources. For example: /proc/uptime

7. /var – Variable Files

var stands for variable files.

Content of the files that are expected to grow can be found under this directory.

This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

Directory that contains temporary files created by system and

users. Files under this directory are deleted when system is

rebooted.

9. /usr – User Programs

Contains binaries, libraries, documentation, and source-code for second level programs.

/usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp

/usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under
/usr/sbin. For example: atd, cron, sshd, useradd, userdel

/usr/lib contains libraries for /usr/bin and /usr/sbin

/usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

Home directories for all users to store their personal

files. For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

Contains boot loader related

files.

Kernel initrd, vmlinux, grub files are located under /boot

For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

Contains library files that supports the binaries located under /bin

and /sbin Library filenames are either ld* or lib*.so.*

For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on

Applications opt stands for optional.

Contains add-on applications from individual vendors.

add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

Temporary mount directory for removable

devices.

For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

16. /srv – Service

Data srv stands for

service.

Contains server specific services related

data. For example, /srv/cvs contains CVS

related data. FILE PERMISSIONS IN

LINUX

Linux file permissions, attributes, and ownership control the access level that the system processes and users have to files. This ensures that only authorized users and processes can access specific files and directories.

The basic Linux permissions model works by associating each system file with an owner and a group and assigning permission access rights for three different classes of users:

- The file owner.
- The group members.
- Others (everybody else)

User/Owner

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same Linux group permissions access to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else.

Hence, when you set the permission for others, it is also referred as set permissions for the world.

Permissions

Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed

Write: The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.

- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.

Changing file/directory permissions with 'chmod' command

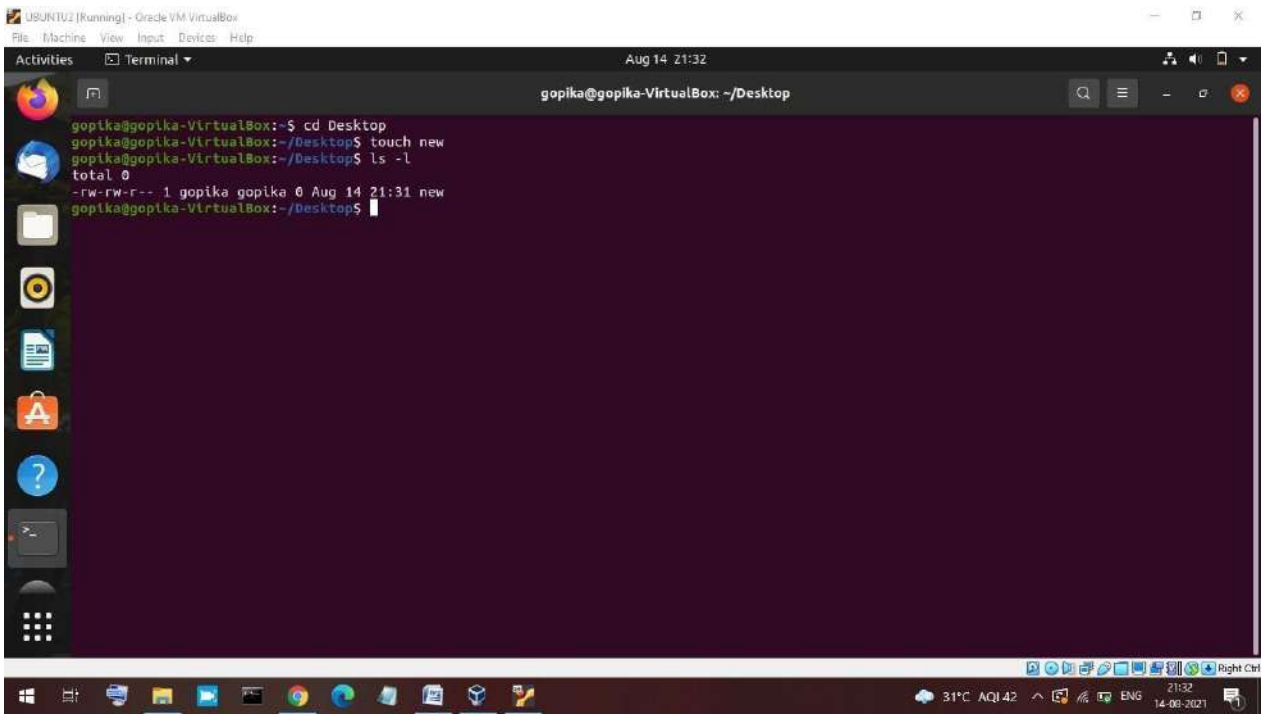
Symbolic Mode

In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the Unix file permissions.

OPERATOR	DESCRIPTION
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permission set earlier

The various owners are represented as –

User Denotations	
u	User /owner
g	group
o	others
a	all



The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –

- The first three characters (2-4) represent the permissions for the file's owner. For example,**-rw-rw-r--** represents that the owner has read (r) and write (w) permission, but no execute permission.

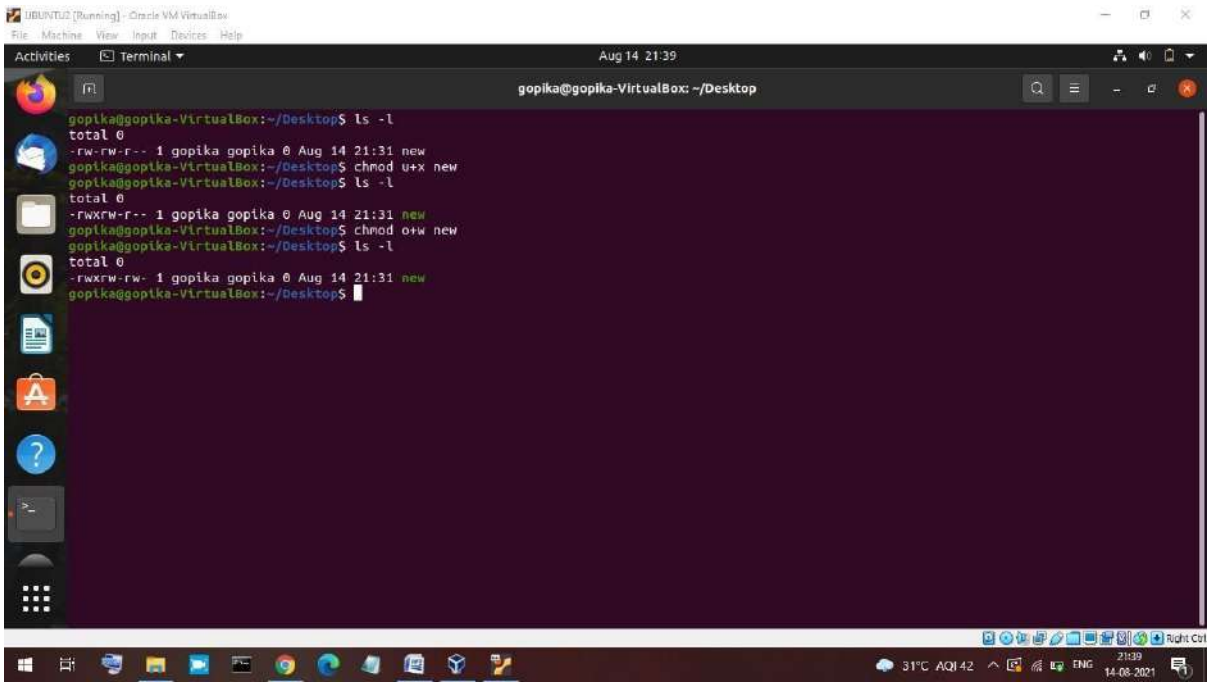
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, **-rw-rw-r--** represents that the group has read (r) and write(w) permission but no execute (x) permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, **-rw-rw-r--** represents that there is **read (r)** only permission.

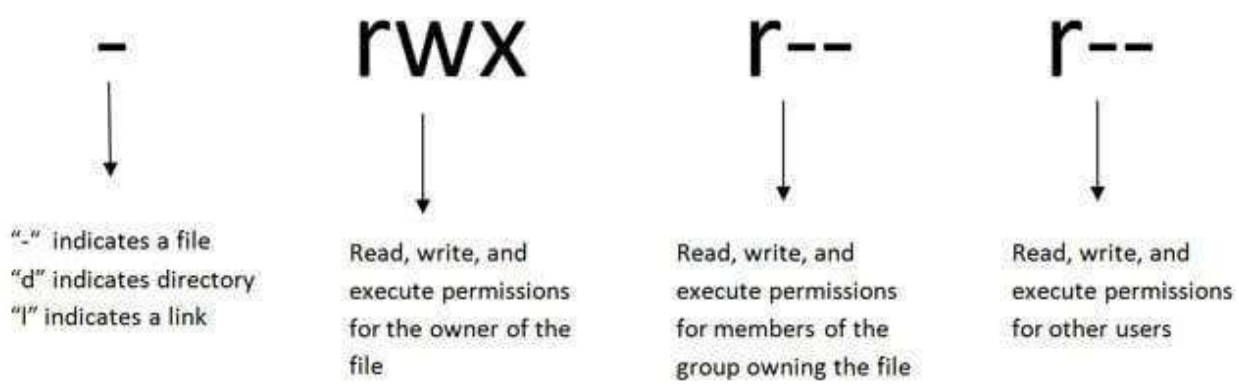
To change directory permissions in Linux, use the following:

- **chmod + rwx filename** to add permissions.
- **chmod - rwx directoryname** to remove permissions.
- **chmod + x filename** to allow executable permissions.
- **chmod - wx filename** to take out write and executable permissions.

Here “r” is for read, “w” is for write, and “x” is for execute. This

only changes the permissions for the owner of the file.

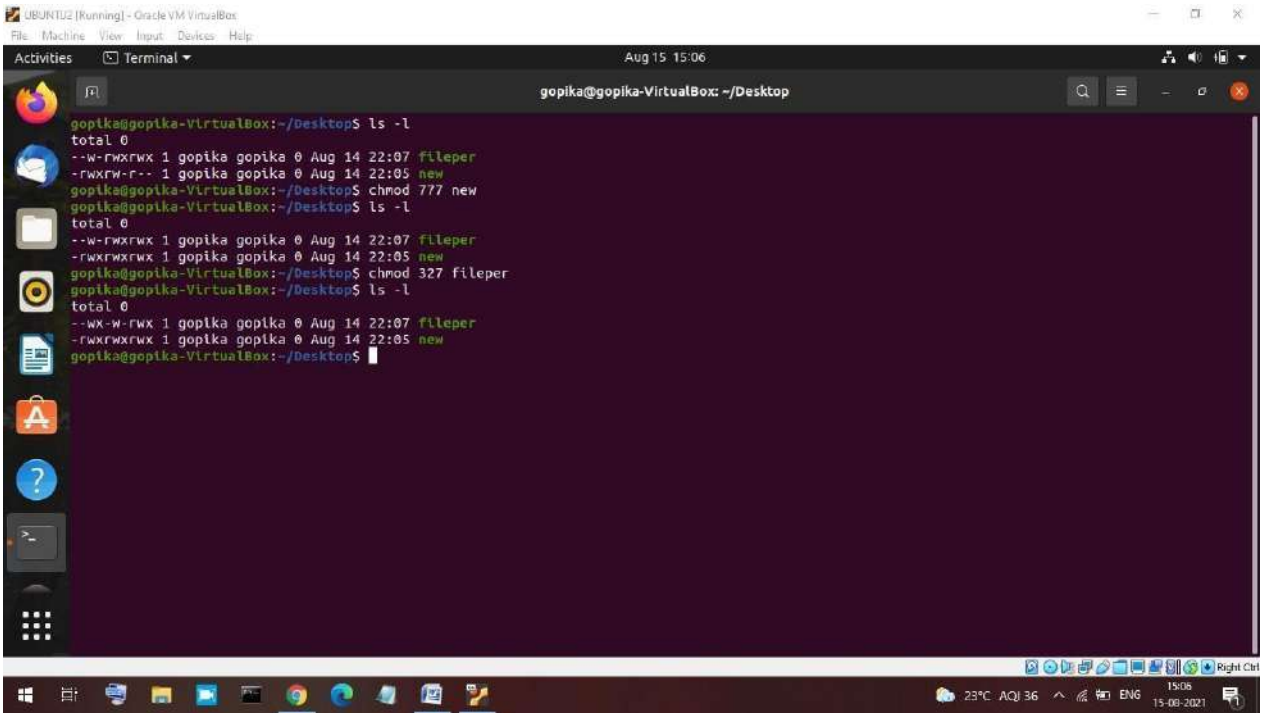




Numeric Mode

Permission numbers are:

- **0 = ---** no permission
- **1 = --x**execute permission only
- **2 = -w-** permission to write only
  - **3** = **-wx** permission for write and execute
- **4 = r-**permission for read
- **5 = r-x** permission for read and execute
- **6 = rw-** permission for read and write
  - **7** = **rwx** permission for read ,write and execute



CHANGING FILE OWNERSHIPS

The chown command allows us to change the user and/or group ownership of a given file, directory, or symbolic link.

In Linux, all files are associated with an owner and a group and assigned with permission access rights for the file owner, the group members, and others.

These commands will give ownership to someone, but all sub files and directories still belong to the original owner.

Chown <name> <file name>

Or

chown [OPTIONS] USER[:GROUP] FILE(s)

USER is the user name or the user ID (UID) of the new owner. GROUP is the name of the new group or the group ID (GID). FILE(s) is the name of one or more files, directories or links. Numeric IDs should be prefixed with the + symbol.

- USER - If only the user is specified, the specified user will become the owner of the given files, the group ownership is not changed.
- USER: - When the username is followed by a colon :, and the group name is not given, the user will become the owner of the files, and the files group ownership is changed to user's login group.
- USER:GROUP - If both the user and the group are specified (with no space betwen them), the user ownership of the files is changed to the given user and the group ownership is changed to the given group.
- :GROUP - If the User is omitted and the group is prefixed with a colon :, only the group ownership of the files is changed to the given group.
- : If only a colon : is given, without specifying the user and the group, no change is made.

By default, on success, chown doesn't produce any output and returns

zero. Superuser permissions are necessary to execute the chown command.

EXPERIMENT4:

AIM : Shell scripting: study bash syntax, environment variables, variables, control constructs such as if, for and while, aliases and functions, accessing command line arguments passed to shell scripts. Study of startup scripts login and logout scripts, familiarity with systemd and system 5 init scripts is expected.

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

The Bourne Shell

The C Shell

The Korn Shell

The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Extended Shell Scripts

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. Of course, most scripts are more complex than the above one.The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.

Common scripting languages include:

Shell scripts - sh, bash, csh, tcsh

Other scripting languages - TCL, Perl, Python

We will look at sh as it is simple, portable, powerful and just like the command line

Repeated tasks can be done much faster

Scripts act as exact records of what commands were run

Scripts avoid small inconsistent errors in processing

Knowing scripting helps with any computer tasks, not just analysis

Examples of useful tasks:

Automatically call BET with several different options

Measure image stats/volumes/PEs for a set of subjects with a single command

Extract timing info from stimulus / behavioural data files

Renaming sets of files

Changing the format of a set of files (e.g. png to tiff)

In these slides there are several accompanying practicals that are extremely useful and can be found by following the links marked with e at the bottom right (to the left of the navigation arrows)

DATA MANAGEMENT

In order to help scripting (and your own sanity) it is a good idea to give files and directories systematic names.

For example:

Con0001 , Con0002, Con0003 , ... , Con0020, Pat0001, Pat0002, ... , Pat0020

Padding the numbers with zeros like this (Con0020 instead of Con20) helps keep the ordering consistent for scripting

In the UK (and other countries) you must not have identifiers (names, dates of birth, even initials) in the filenames

Consider renaming your original images to more meaningful (and consistent) names

Keep track of your disk usage and delete any analyses that were incorrect (many people have res.feat, res+.feat, res++.feat, res+++.feat , etc. which is confusing and a waste of space)

BASIC SHELL SCRIPT

A bourne shell (sh) script is a list of lines in a file that are executed in the bourne shell (a forerunner of bash); simplest is just commands that could be run at the prompt.

The first line in a sh script MUST be #!/bin/sh

Things to remember:

always make sure it has executable status chmod

a+x filename

script runs in the current directory (pwd)

may not inherit the same environment - esp. if used by others

Example:

#!/bin/sh

bet im1 im1_brain -m

mv im1_brain_mask.nii.gz mask1.nii.gz

A script can be stopped at any point by using return: e.g. return 0

USEFUL SHELL SCRIPT TEMPLATE

Before learning things systematically, here is a fairly simple script which is very powerful and useful for modifying for many different tasks.

```
#!/bin/sh

for filename in *.nii.gz ; do

  fname=`$FSLDIR/bin/remove_ext ${filename}`

  fslmaths ${fname} -s 2 ${fname}_smooth2

  mv ${fname}.nii.gz ${fname}_smooth0.nii.gz

done
```

What this does:

For each image (*.nii.gz) it smooths it to make a new one of the same name but ending in _smooth2 and also renames the unsmoothed image to end with _smooth0

How this works:

The variable filename is used in a for loop to go through each name matching *.nii.gz

The variable fname is set to the filename with the ending (e.g. .nii.gz) removed.

Don't worry about how this works for now - the details will be explained later.

${filename} and ${fname} are used to get the values (contents) of the variables

fslmaths is used to do the smoothing.

mv is used to do the renaming (notice that .nii.gz is needed here, but not for the fsl tools, as they work with or without the .nii.gz endings).

SCRIPTING TIPS

Here are some basic, but useful, tips for writing scripts

Put in comments (to jog your memory when you write your paper months/years later)

Put in some echo output commands so that you get some feedback on what your script is doing as it runs

If your script starts doing something bad (or nothing at all) then use control-C to stop it

If your script makes new files, changes files or deletes files then start with a version which uses echo in front of the important commands. When you run this version it will just display the commands to the screen so that you can examine them carefully and make sure they are right. Once you are happy with them then remove the echo from in front of these commands and run this version.

It doesn't hurt to make a backup of key files before running a script, just in case.

BASIC SCRIPTING CONCEPTS

We will now look systematically at the following shell and scripting concepts:

Wildmasks

Echo (printing to the screen/file)

Variables

Braces

Command Line Arguments

Single Quotes and Backslash

Double Quotes

Backquotes

Pipes

File Redirection

Following this some useful utilities and programming constructs (like the for loop) will be covered.

WILDMASKS

Can use wildmasks for matching patterns in filenames; expand into a list of all filename matches. E.g.:

* matches any string

? matches any one character

[abgj] matches any one character in this range/list

$ ls

sub1_t1.nii.gz sub1_t2.nii.gz sub2_t1.nii.gz sub2_t2.nii.gz sub3_pd.nii.gz

$ ls sub*

sub1_t1.nii.gz sub1_t2.nii.gz sub2_t1.nii.gz sub2_t2.nii.gz sub3_pd.nii.gz

$ ls sub1*

sub1_t1.nii.gz sub1_t2.nii.gz

$ ls sub*t1*

sub1_t1.nii.gz sub2_t1.nii.gz

$ ls sub[13]*

sub1_t1.nii.gz sub1_t2.nii.gz sub3_pd.nii.gz

$ ls sub?_t2.nii.gz

sub1_t2.nii.gz sub2_t2.nii.gz

ECHO

echo prints the rest of the line to the screen (standard output).

This is useful for providing output or updates in a script.

Wildmasks (for filenames) and variables (values) are substituted in the argument before echo prints them.

Examples:

$ echo Hello All!

Hello All!

$ echo sub*t1*

sub1_t1.nii.gz sub2_t1.nii.gz

$ echo j*k

j*k

VARIABLES

Like most programming languages, the shell allows items to be stored in variables.

All shell variables store strings.

A variable is set using:

NAME=VALUE

The variable name should start with a letter but can contain numbers and underscores

The value of a variable can be returned/used by adding a prefix $

Examples:

$ var1=im1.nii.gz

$ echo $var1

im1.nii.gz

$ echo var1

var1

$ ls $var1

im1.nii.gz

BRACES

Any name that starts with a letter can be used as a variable name.

For instance: v, v1, v1_1, v_filename_4

To add a string immediately after a variable name can be confusing.

The situation is solved by putting the variable name inside braces.

Examples:

$ v=im1

$ echo $v_new

$ echo ${v}_new

im1_new

NB: all unused variables are blank by default (generate no error)

COMMAND LINE ARGUMENTS

Inside a script the variables $1 $2 $3 etc. store the value of the command line arguments.

e.g. if a script called reg_vol is executed as:

$ reg_vol im1 3 abc

then $1 = im1, $2 = 3, $3 = abc

Other special variables are:

$0 = name of the script (often including the path)

$# = number of command line arguments given

$@ = all the command line arguments

(i.e. $1 $2 $3 ...)

$$ = process ID number (unique to this process)

SINGLE QUOTES AND BACKSLASH

The shell substitutes variable names and wildmasks before executing the command - sometimes this is undesirable.

To avoid substitutions either

prefix the special character (wildmask or $ sign) with a backslash: \

put the desired string in single quotes: '

Examples:

```
$ var1=im1.nii.gz
 $ echo $var1
   im1.nii.gz
$ echo \$var1
   $var1
$ echo '$var1'
   $var1
```

DOUBLE QUOTES

To group several strings together as one argument it is necessary to use double quotes: "

For example:

```
$ v=Hello World
 $ echo $v
   Hello
```

$ v="Hello World"

$ echo $v

Hello World

NB: Variable substitutions are done inside double quotes but wildmasks are not expanded:

e.g. echo "*" just prints a *

but echo "$v" is the same as echo $v

BACKQUOTES

The (text) result of any command can be captured using backquotes: `

This is very useful for setting variables.

Examples:

$ v=`ls sub[13]*`

$ echo $v

sub1_t1.nii.gz sub1_t2.nii.gz sub3_pd.nii.gz

$ echo `fslval sub1_t1 pixdim2`

4.0

NB: the result is always treated as a single string, even if it contains spaces

PIPE

One of the most powerful features of the shell is the ability to chain commands together, each taking its input from the previous command's output.

This is done using the pipe symbol: |

Examples (using the wordcount utility):

$ cat .bashrc | wc

7 83 534

$ echo "Hello World" | wc

1 2 12

Technically this redirects standard output of one command to be the standard input of another.

Error messages that are printed to standard error are not redirected with the pipe.

FILE REDIRECTION

Command input can be taken from a file with: <

Command output can be redirected to a file with: >

Command output can be appended to a file with: >>

Examples:

$ echo "smoothing=10mm" > settings.txt

$ echo "No lowpass" >> settings.txt

$ cat settings.txt

smoothing=10mm

No lowpass

A simple program to add two numbers

#!/bin/bash

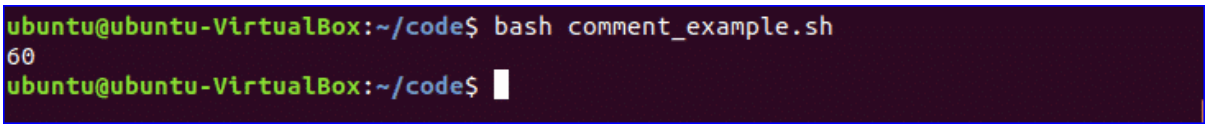
Add two numeric value

((sum=25+35))

#Print the result

echo $sum

Output



variables

A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

A variable is nothing more than a pointer to the actual data. The shell enables you to create, assign, and delete variables.

Variable Names

The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_).

By convention, Unix shell variables will have their names in UPPERCASE.

The following examples are valid variable names −

_ALI

TOKEN_A

VAR_1

VAR_2

Defining Variables

Variables are defined as follows:

variable_name=variable_value

For example:

NAME="Zara Ali"

The above example defines the variable NAME and assigns the value "Zara Ali" to it. Variables of this type are called scalar variables. A scalar variable can hold only one value at a time.

Accessing Values

To access the value stored in a variable, prefix its name with the dollar sign ($).

For example, the following script will access the value of defined variable NAME and print it on STDOUT

```
#!/bin/sh

NAME="Zara Ali"

echo $NAME
```

The above script will produce the following value:

Zara Ali

Read-only Variables

Shell provides a way to mark variables as read-only by using the read-only command. After a variable is marked read-only, its value cannot be changed.

For example, the following script generates an error while trying to change the value of NAME:

```
#!/bin/sh

NAME="Zara Ali"

readonly NAME

NAME="Qadiri"
```

The above script will generate the following result:

```
/bin/sh: NAME: This variable is read only.
```

Unsetting Variables

Unsetting or deleting a variable directs the shell to remove the variable from the list of variables that it tracks. Once you unset a variable, you cannot access the stored value in the variable.

Following is the syntax to unset a defined variable using the unset command −

```
unset variable_name
```

The above command unsets the value of a defined variable. Here is a simple example that demonstrates how the command works:

```
#!/bin/sh

NAME="Zara Ali"

unset NAME

echo $NAME
```

The above example does not print anything. You cannot use the unset command to unset variables that are marked readonly.

Variable Types

When a shell is running, three main types of variables are present:

- Local Variables − A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.

- Environment Variables – An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually, a shell script defines only those environment variables that are needed by the programs that it runs.

- Shell Variables – A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

Control constructs

You can control the execution of Linux commands in a shell script with control structures. Control structures allow you to repeat commands and to select certain commands over others. A control structure consists of two major components: a test and commands. If the test is successful, then the commands are executed. In this way, you can use control structures to make decisions as to whether commands should be executed.

There are two different kinds of control structures: loops and conditions. A loop repeats commands, whereas a condition executes a command when certain conditions are met. The BASH shell has three loop control structures: while, for, and for-in. There are two condition structures: if and case. The control structures have as their test the execution of a Linux command. All Linux commands return an exit status after they have finished executing. If a command is successful, its exit status will be 0. If the command fails for any reason, its exit status will be a positive value referencing the type of failure that occurred. The control structures check to see if the exit status of a Linux command is 0 or some other value. In the case of the if and while structures, if the exit status is a zero value, then the command was successful and the structure continues.

Test Operations

With the test command, you can compare integers, compare strings, and even perform logical operations. The command consists of the keyword test followed by the values being compared, separated by an option that specifies what kind of comparison is taking place. The option can be thought of as the operator, but it is written, like other options, with a minus sign and letter codes. For example, -eq is the option that represents the equality comparison.
However, there are two string operations that actually use an operator instead of an option. When you compare two strings for equality you use the equal sign, =. For inequality you use !=. Here is some of the commonly used options and operators used by test.

The syntax for the test command is shown here:

test value -option

-gt	Greater-than
-lt	Less-than
-ge	Greater-than-or-equal-to
-le	Less-than-or-equal-to
-eq	Equal
-ne	Not-equal

String Comparisons Function

-z	Tests for empty string
=	Equal strings
!=	Not-equal strings

Logical Operations Function

-a	Logical AND
-o	Logical OR
!	Logical NOT

File Tests Function

-f	File exists and is a regular file
-s	File is not empty
-r	File is readable
-w	File can be written to, modified
-x	File is executable
-d	Filename is a directory name

Conditional Control Structures

The BASH shell has a set of conditional control structures that allow you to choose what Linux commands to execute. Many of these are similar to conditional control structures found in programming languages, but there are some differences. The if condition tests the success of a Linux command, not an expression. Furthermore, the end of an if-then command must be indicated with the keyword fi, and the end of a case command is indicated with the keyword esac.

The if structure places a condition on commands. That condition is the exit status of a specific Linux command. If a command is successful, returning an exit status of 0, then the commands within the if structure are executed. If the exit status is anything other than 0, then the command has failed and the commands within the if structure are not executed. The if command begins with the keyword if and is followed by a Linux command whose exit condition will be evaluated. The keyword fi ends the command. The elsels script in the next example executes the ls command to list files with two different possible options, either by size or with all file information. If the user enters an s, files are listed by size; otherwise, all file information is listed.

if

if executes an action if its test command is true.

Syntax:

if command then

command

fi

if then else

if-else executes an action if the exit status of its test command is true; if false, then the else action is executed.

Syntax:

```
 if command then
     command
else
     command
fi
```

elif

elif allows you to nest if structures, enabling selection among several alternatives; at the first true if structure, its commands are executed and control leaves the entire elif structure.

Syntax:

```
 if command then
    command
 elif command then
    command
else
    command
fi
```

case

case matches the string value to any of several patterns; if a pattern is matched, its associated commands are executed.

Syntax:

```
case string in
pattern)
command;;
esac
```

Logical AND

The logical AND condition returns a true 0 value if both commands return a true 0 value; if one returns a non-zero value, then the AND condition is false and also returns a non-zero value.

Syntax:

```
command && command
```

Logical OR

The logical OR condition returns a true 0 value if one or the other command returns a true 0 value; if both commands return a non-zero value, then the OR condition is false and also returns a non-zero value.

Syntax:

command || command

Logical NOT

The logical NOT condition inverts the return value of the command.

Syntax:

! command

Loop Control Structures

The while loop repeats commands. A while loop begins with the keyword while and is followed by a Linux command. The keyword do follows on the next line. The end of the loop is specified by the keyword done. The Linux command used in while structures is often a test command indicated by enclosing brackets.

The for-in structure is designed to reference a list of values sequentially. It takes two operands—a variable and a list of values. The values in the list are assigned one by one to the variable in the for-in structure. Like the while command, the for-in structure is a loop. Each time through the loop, the next value in the list is assigned to the variable. When the end of the list is reached, the loop stops. Like the while loop, the body of a for-in loop begins with the keyword do and ends with the keyword done.

while

while executes an action as long as its test command is true.

Syntax:

while command

do

    command

done

until

until executes an action as long as its test command is false.

Syntax:

until command

do

command

done

for-in

for-in is designed for use with lists of values; the variable operand is consecutively assigned the values in the list.

Syntax:

for variable in list-values

do

command

done

for

for is designed for reference script arguments; the variable operand is consecutively assigned each argument value.

Syntax:

for variable

do

command

done

select

select creates a menu based on the items in the item-list; then it executes the command; the command is usually a case.

Syntax:

select string in item-list

do

command

done

Scope of an environment variable

Scope of any variable is the region from which it can be accessed or over which it is defined. An environment variable in Linux can have global or local scope.

Global

A globally scoped ENV that is defined in a terminal can be accessed from anywhere in that particular environment which exists in the terminal. That means it can be used in all kind of scripts, programs or processes running in the environment bound by that terminal.

Local

A locally scoped ENV that is defined in a terminal cannot be accessed by any program or process running in the terminal. It can only be accessed by the terminal (in which it was defined) itself.

Accessing ENVs

Syntax:

$NAME

NOTE: Both local and global environment variables are accessed in the same way.

Displaying ENVs

To display any ENV

Syntax:

$ echo $NAME

To display all the Linux ENVs

Syntax:

$ printenv //displays all the global ENVs

or

$ set //display all the ENVs(global as well as local)

or

$ env //display all the global ENVs

Setting environment variables

To set a global ENV

$ export NAME=Value

or

$ set NAME=Value

To set a local ENV

Syntax:

$ NAME=Value

Some commonly used ENVs in Linux:

$USER : Gives current user's name.

$PATH : Gives search path for commands.

$PWD : Gives the path of present working directory.

$HOME : Gives path of home directory.

$HOSTNAME : Gives name of the host.

$LANG : Gives the default system language.

$EDITOR : Gives default file editor.

$UID : Gives user ID of current user.

$SHELL : Gives location of current user's shell program.

Shell functions

Functions enable you to break down the overall functionality of a script into smaller, logical subsections, which can then be called upon to perform their individual tasks when needed.

Using functions to perform repetitive tasks is an excellent way to create code reuse. This is an important part of modern object-oriented programming principles.

Shell functions are similar to subroutines, procedures, and functions in other programming languages.

Creating Functions

To declare a function, simply use the following syntax:

```
function_name () {
    list of commands
}
```

The name of your function is function_name, and that's what you will use to call it from elsewhere in your scripts. The function name must be followed by parentheses, followed by a list of commands enclosed within braces.

Example

Following example shows the use of function:

```
#!/bin/sh
# Define your function here
Hello () {
    echo "Hello World"
}
# Invoke your function
Hello
```

Upon execution, you will receive the following output:

```
$./test.sh
Hello World
```

Pass Parameters to a Function

You can define a function that will accept parameters while calling the function. These parameters would be represented by $1, $2 and so on.

Following is an example where we pass two parameters Zara and Ali and then we capture and print these parameters in the function.

```
#!/bin/sh

# Define your function here

Hello () {

    echo "Hello World $1 $2"

}

# Invoke your function

Hello Zara Ali
```

Upon execution, you will receive the following result:

```
$./test.sh

Hello World Zara Ali
```

Returning Values from Functions

If you execute an exit command from inside a function, its effect is not only to terminate execution of the function but also of the shell program that called the function.

If you instead want to just terminate execution of the function, then there is way to come out of a defined function.

Based on the situation you can return any value from your function using the return command whose syntax is as follows:

```
return code
```

Here code can be anything you choose here, but obviously you should choose something that is meaningful or useful in the context of your script as a whole.

Example

Following function returns a value 10:

```
#!/bin/sh

# Define your function here

Hello () {
```

```
    echo "Hello World $1 $2"
    return 10
}
# Invoke your function
Hello Zara Ali
# Capture value returnd by last command
ret=$?
echo "Return value is $ret"
```

Upon execution, you will receive the following result:

```
$./test.sh
 Hello World Zara Ali
 Return value is 10
```

Nested Functions

One of the more interesting features of functions is that they can call themselves and also other functions. A function that calls itself is known as a recursive function.

Following example demonstrates nesting of two functions:

```
#!/bin/sh
# Calling one function from another
number_one () {
    echo "This is the first function speaking..."
    number_two
}
number_two () {
    echo "This is now the second function speaking..."
}
# Calling function one.
number_one
```

Upon execution, you will receive the following result:

This is the first function speaking...

This is now the second function speaking...

Function Call from Prompt

You can put definitions for commonly used functions inside your.profile. These definitions will be available whenever you log in and you can use them at the command prompt.

Alternatively, you can group the definitions in a file, say test.sh, and then execute the file in the current shell by typing:

$. test.sh

This has the effect of causing functions defined inside test.sh to be read and defined to the current shell as follows:

$ number_one

This is the first function speaking...

This is now the second function speaking...

$

To remove the definition of a function from the shell, use the unset command with the .f option. This command is also used to remove the definition of a variable to the shell.

$ unset -f function_name

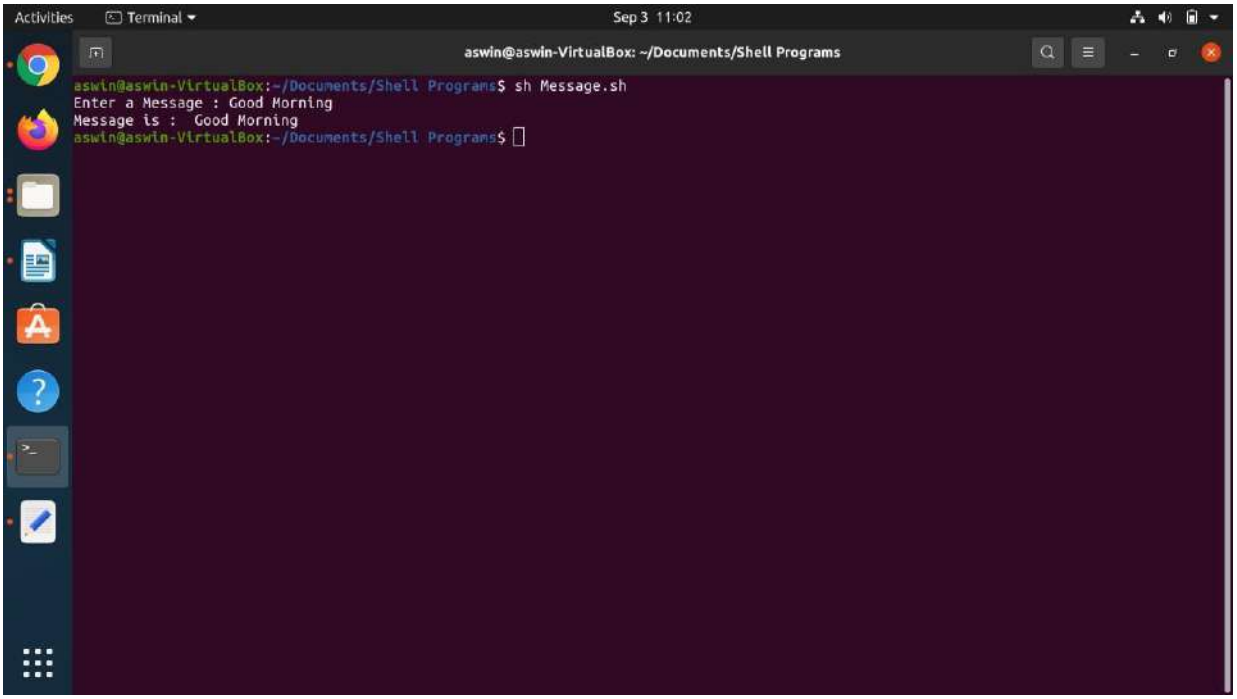
Shell Programming

1. Write a shell script program to display a given message

Source Code

```
#!/bin/bash
read -p "Enter a Message : " m
echo "Message is : " $m
```

Output

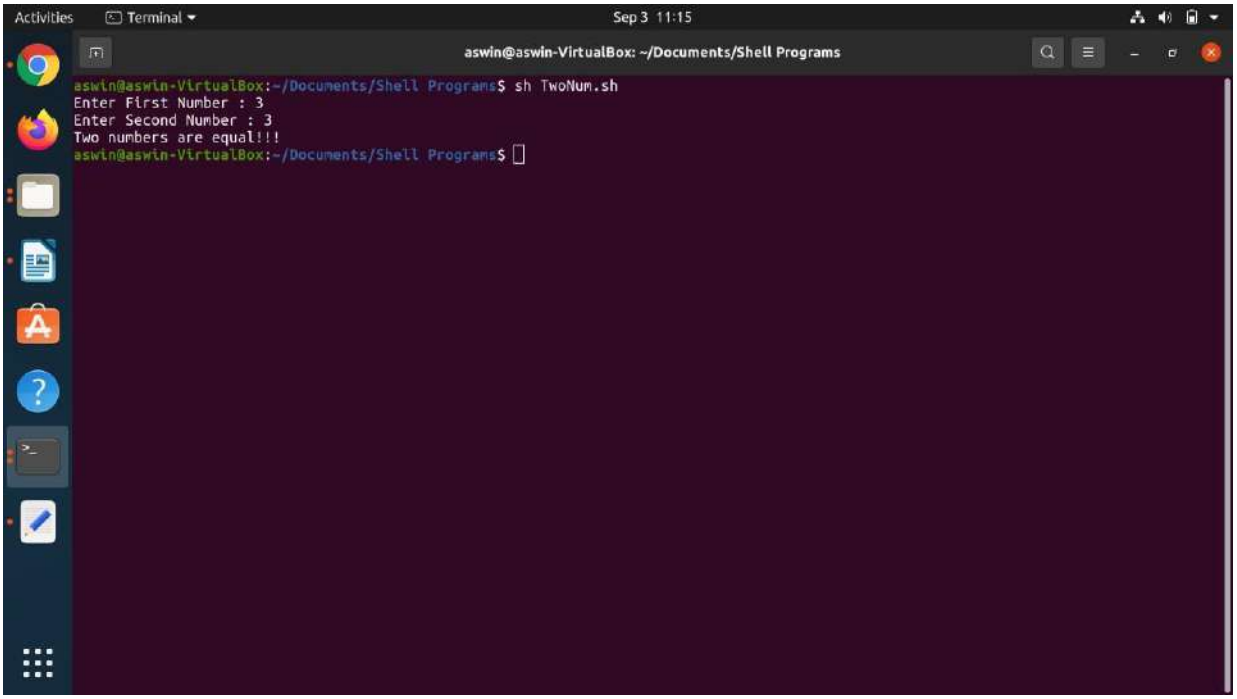


2. Write a shell script to print whether two numbers are equal or not

Source Code

```
#!/bin/bash
read -p "Enter First Number : " n1
read -p "Enter Second Number : " n2
if [ $n1 -eq $n2 ]
then
echo "Two numbers are equal!!!"
else
echo "Two numbers are not equal!!!"
fi
```

Output



3. Write a Shell Program to find the roots of the quadratic equation.

Source Code

```
#!/bin/bash
echo Enter the coefficient of x^2:
read a
echo Enter the coefficient of x:
read b
echo Enter the constant term:
read c
f=`echo "-($b)" |bc`
p=`expr 2 \* $a`
if [ $a -ne 0 ]
then
d=`echo \( \( $b \* $b \) - \( 4 \* $a \* $c \) \) | bc`
if [ $d -lt 0 ]
then
x=`echo "-($d)" | bc`
s=`echo "scale=2; sqrt ( $x )" | bc`
echo The first root is:
echo "($f + $s i) / $p" echo
The second root is:
echo "($f - $s i) / $p"
elif [ $d -eq 0 ]
then
res=`expr $f / $p` echo
The root is: $res else
s=`echo "scale=2; sqrt( $d )" | bc`
res1=`echo "scale=2; ( $f + $s) / ( $p )"|bc`
res2=`echo "scale=2; ( $f - $s) / ( $p )"|bc`
```

echo The first root is: $res1 echo

The second root is: $res2 fi

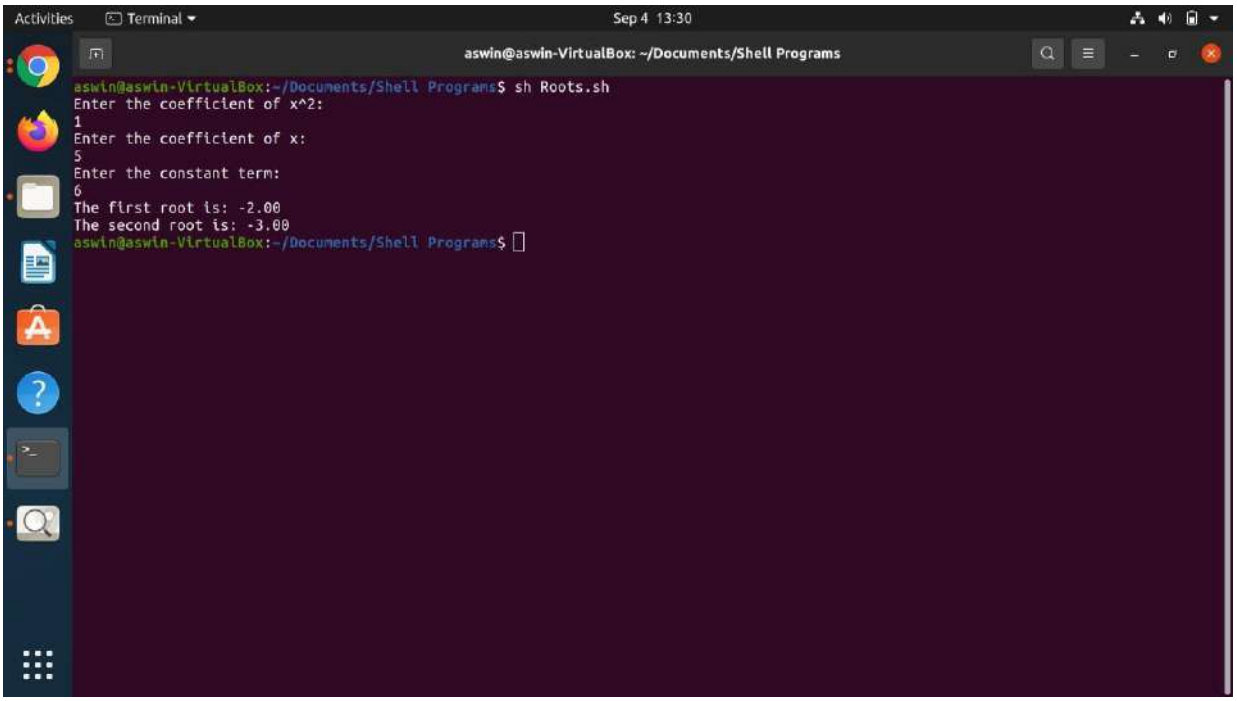
else

echo Coefficient of x^2 can not be 0.

fi"Two numbers are not equal!!!"

fi

Output

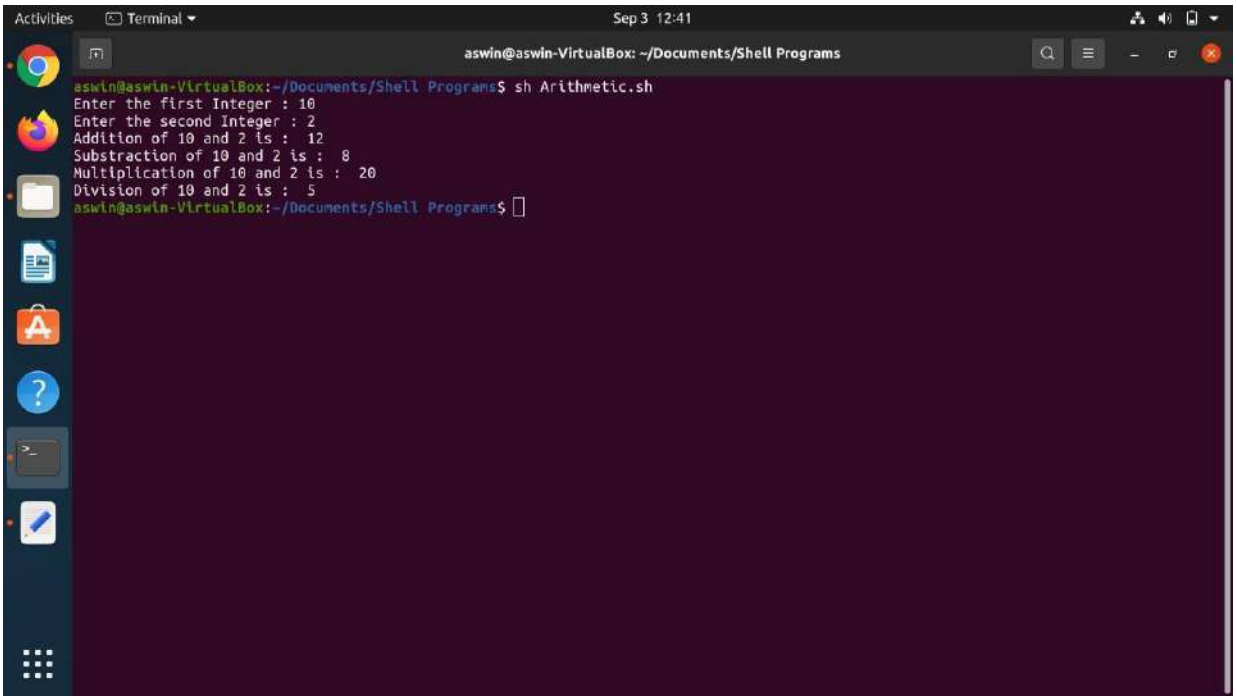


4. Write a shell script to perform integer arithmetic operations.

Source Code

```
!/bin/bash
read -p "Enter the first Integer : " a
read -p "Enter the second Integer : " b
add=$(( $a+$b ))
sub=$(( $a-$b ))
mul=$(( $a*$b ))
div=$(( $a/$b ))
echo "Addition of $a and $b is : " $add
echo "Substraction of $a and $b is : " $sub
echo "Multiplication of $a and $b is : " $mul
echo "Division of $a and $b is : " $div
```

Output

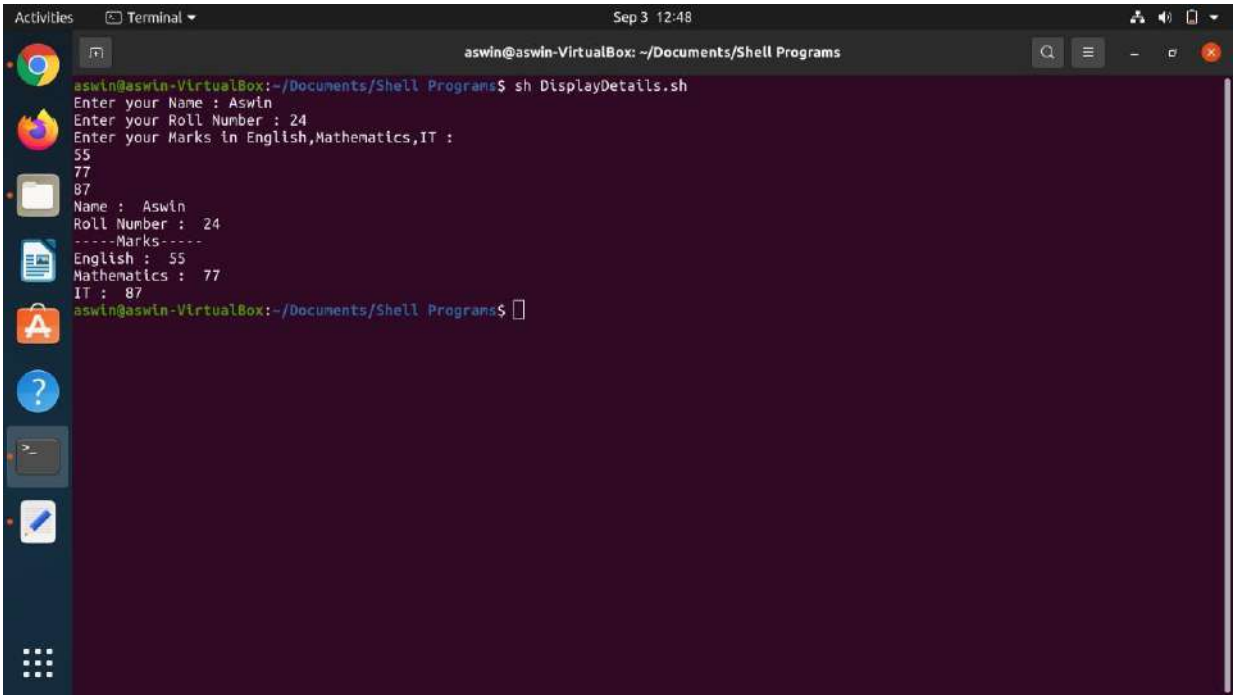


5. Write a shell script to getting input details like name, roll number and marks and print them.

Source Code

```
#!/bin/bash

read -p "Enter your Name : " name

read -p "Enter your Roll Number : " roll

echo "Enter your Marks in English,Mathematics,IT : "

read english

read maths

read it

echo "Name : " $name echo

"Roll Number : " $roll echo "-

----Marks ---------- "

echo "English : " $english echo

"Mathematics : " $maths echo

"IT : " $it
```

Output

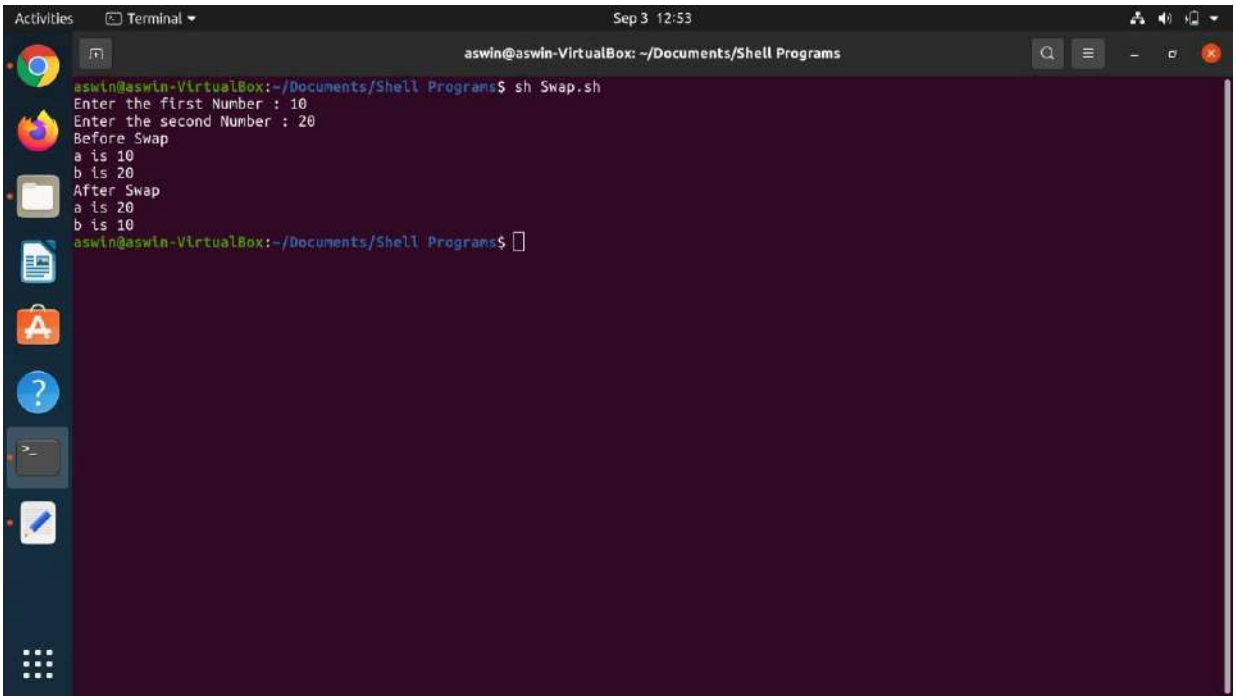


6. Write a Shell program to swap two values

Source Code

```
#!/bin/bash
read -p "Enter the first Number : " a
read -p "Enter the second Number : " b
echo "Before Swap"
echo "a is $a"
echo "b is $b"
a=$(($a+$b))
b=$(($a-$b))
a=$(($a-$b))
echo "After Swap"
echo "a is $a" echo "b
is $b"
```

Output

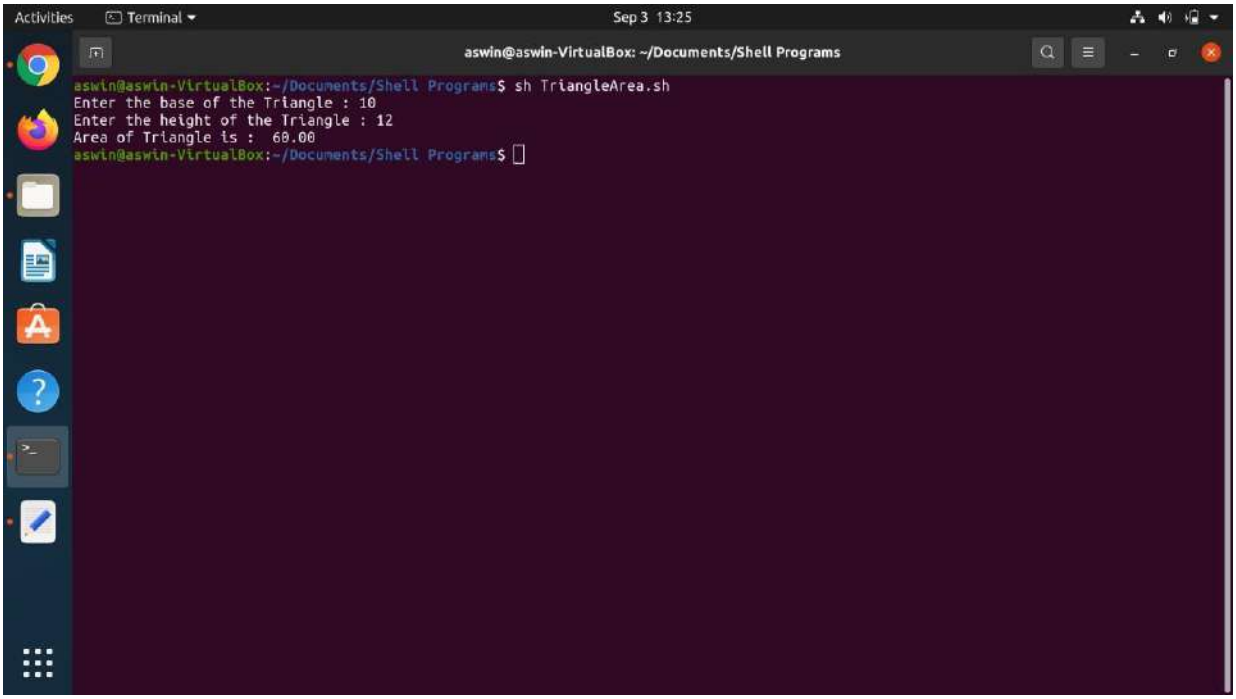


7. Write a shell program to find the area of a triangle.

Source Code

```
#!/bin/bash
read -p "Enter the base of the Triangle : " b
read -p "Enter the height of the Triangle : " h
area=`expr "scale=2; 1/2*$b*$h"|bc`
echo "Area of Triangle is : " $area
```

Output

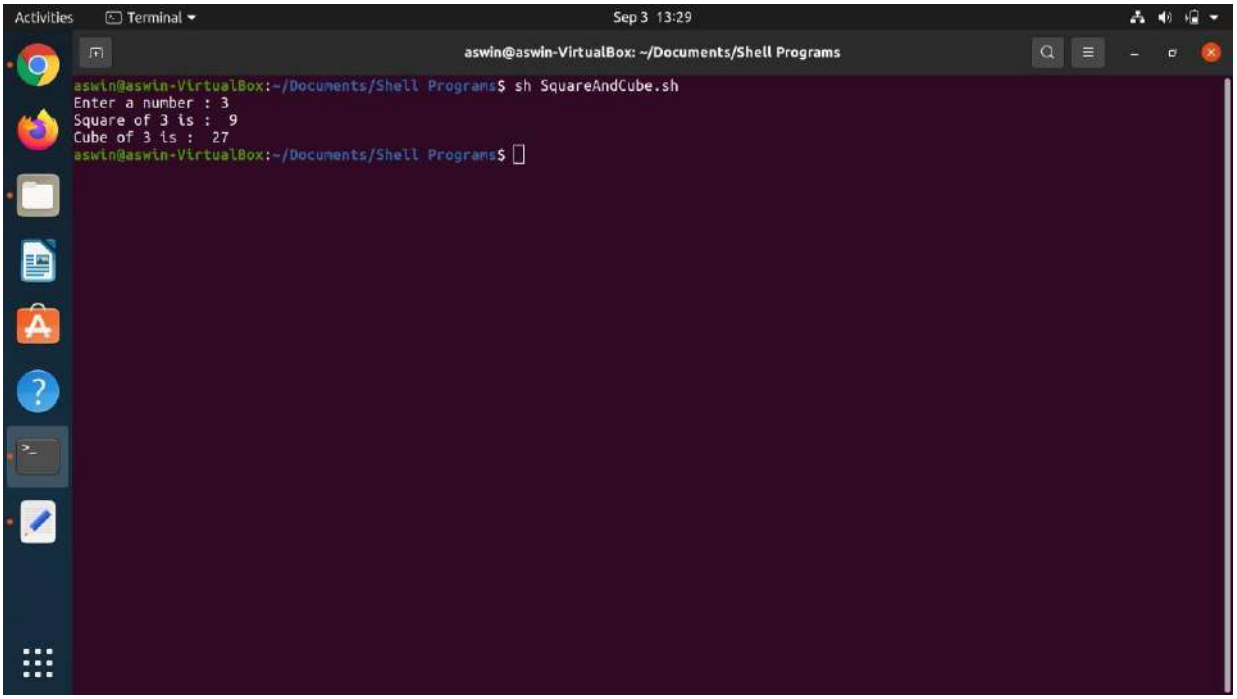


8. Write a shell program to find the square and cube of a number

Source Code

```
#!/bin/bash
read -p "Enter a number : " a
square=$(( $a*$a )) cube=$((
$a*$a*$a ))
echo "Square of $a is : " $square
echo "Cube of $a is : " $cube
```

Output

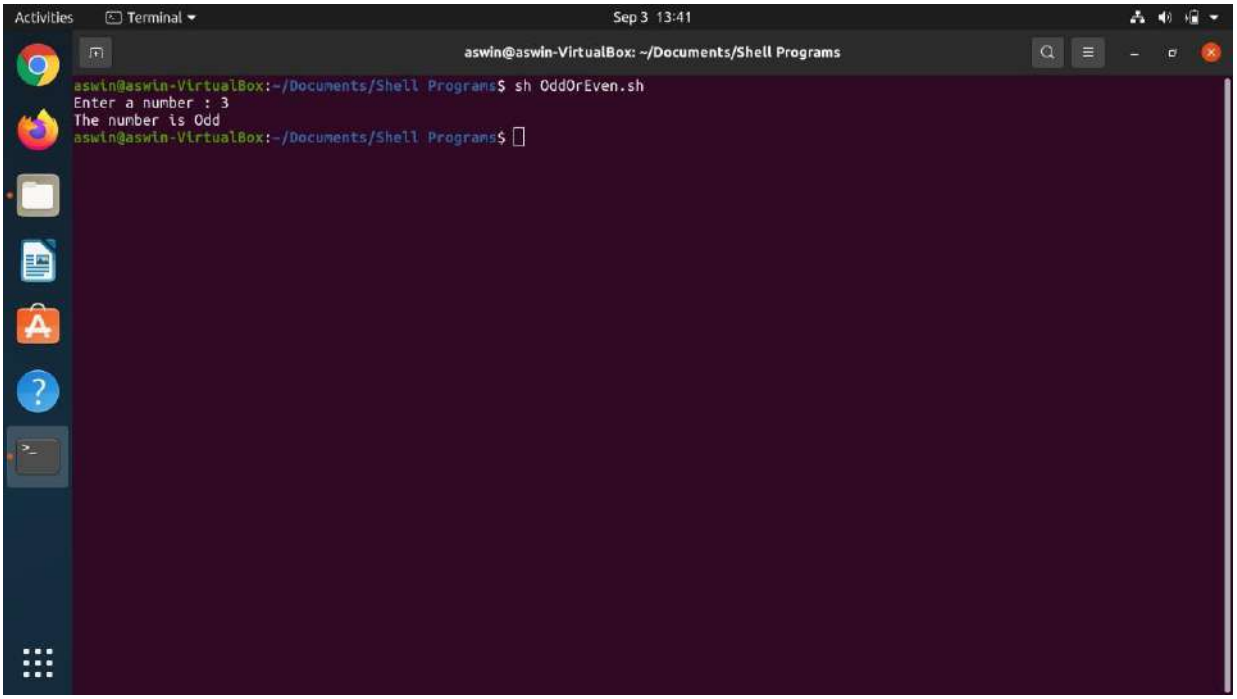


9. Write a shell program to check whether the given number is odd or even.

Source Code

```
#!/bin/bash
read -p "Enter a number : " a if
[ $(( a%2 )) -eq 0 ]
then
echo "The number is Even" else
echo "The number is Odd" fi
```

Output

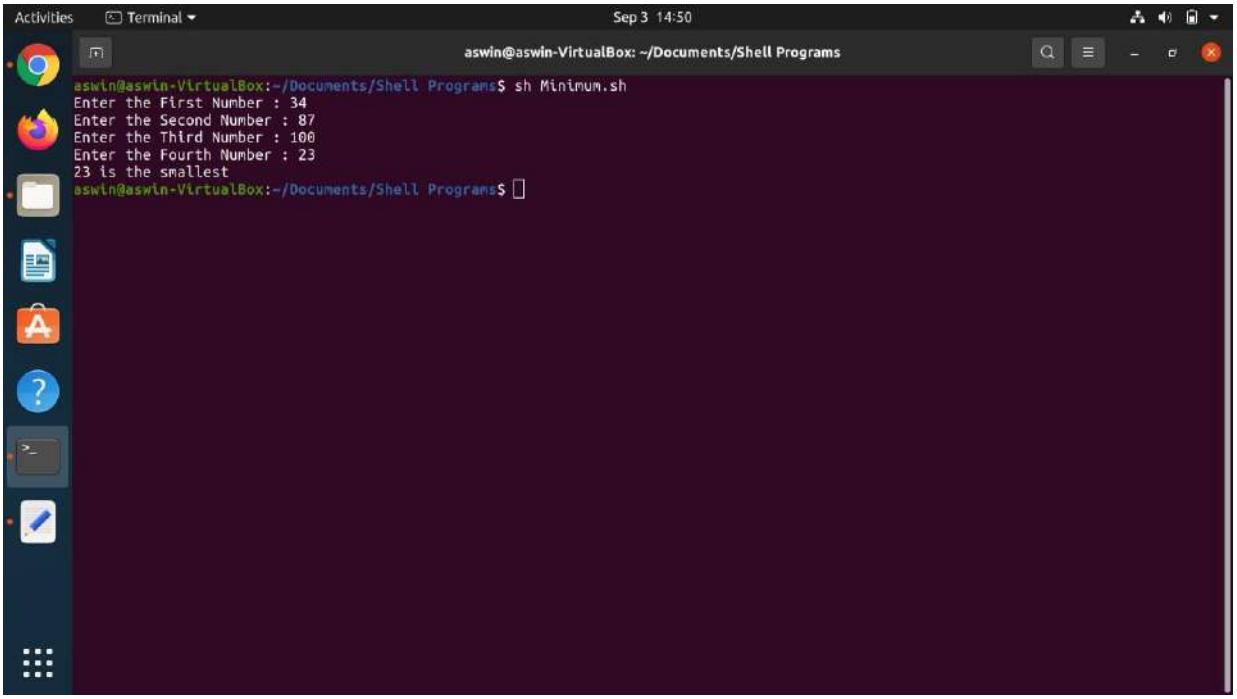


10. Write a shell program to find the minimum among four values.

Source Code

```
#!/bin/bash
read -p "Enter a number : " a if
[ $(( a%2 )) -eq 0 ]
then
echo "The number is Even" else
echo "The number is Odd" fi
```

Output

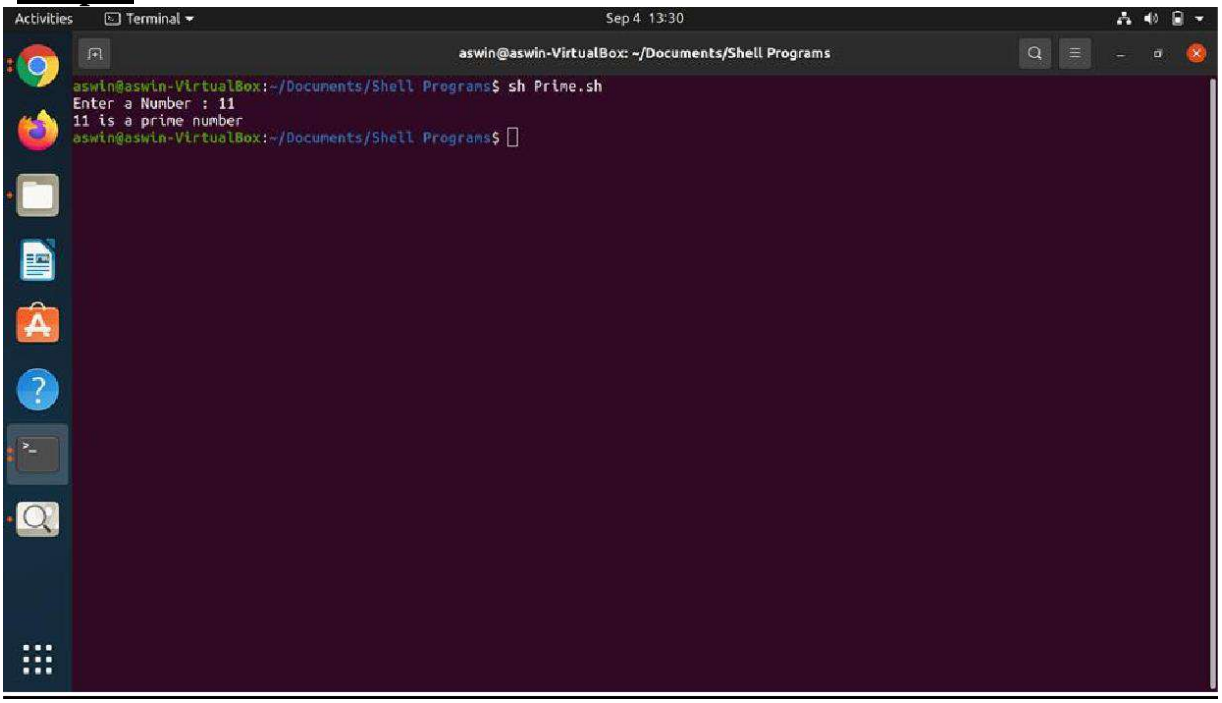


11. Write a shell program to check whether the input number is prime or not.

Source Code

```
#!/bin/bash
read -p "Enter a Number : " a
flag=0
half=$(( $a/2 ))
for i in $(seq 2 $half) do
if [ $(( a % i )) -eq 0 ]
then
echo "$a is not a prime number"
flag=1
break fi
done
if [ $flag -eq 0 ] then
echo "$a is a prime number" fi
```

Output



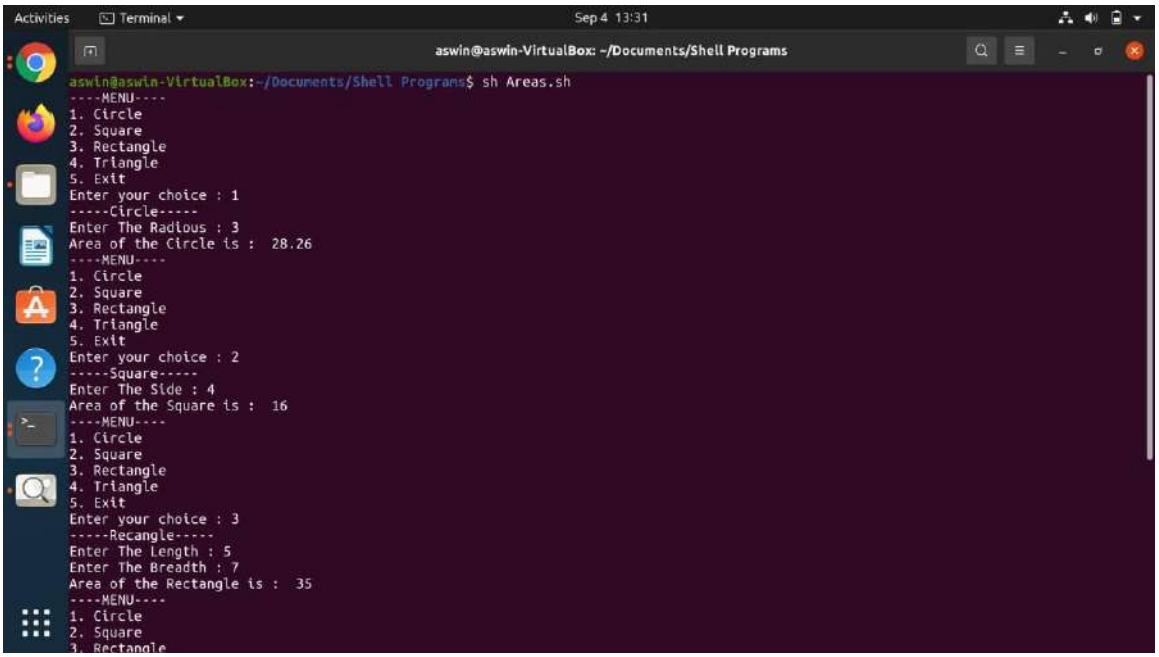
12. Write a shell program to find the area of circle, square, rectangle and triangle using case statements.

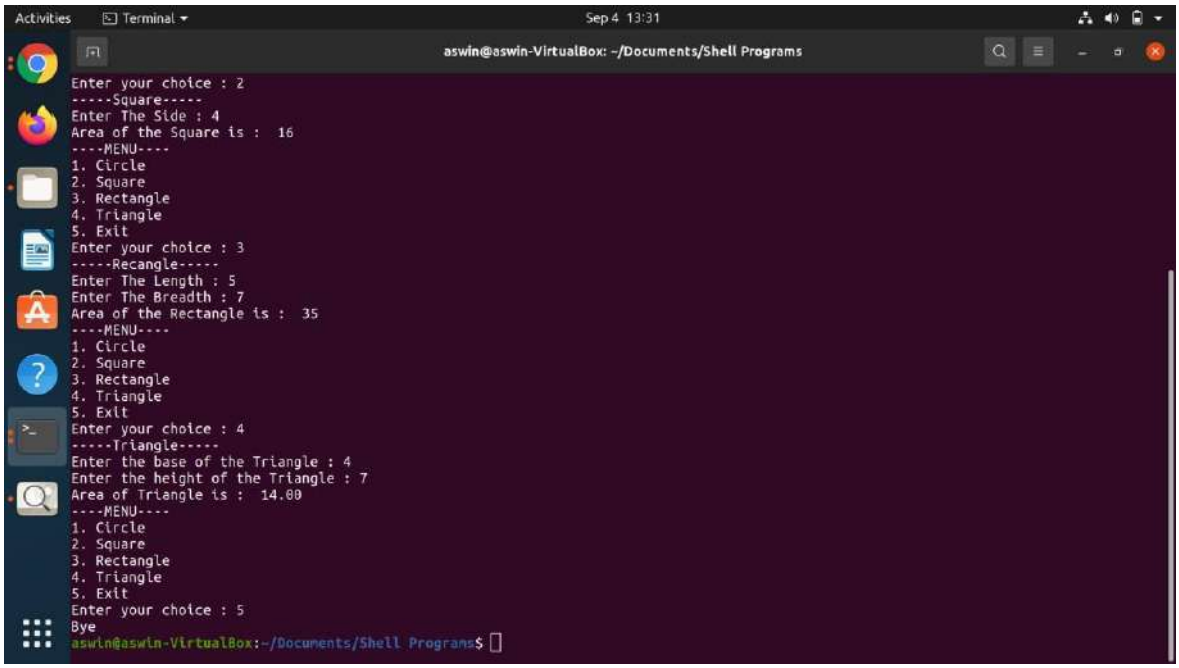
Source Code

```
#!/bin/bash
val=1
while [ $val = 1 ] do
echo "----MENU------- "
echo "1. Circle"
echo "2. Square"
echo "3. Rectangle"
echo "4. Triangle"
echo "5. Exit"
read -p "Enter your choice : " ch
case "$ch" in
1) echo "-----Circle--- "
read -p "Enter The Radious : " r
area=$(echo "scale=2; 3.14*$r*$r" | bc)
echo "Area of the Circle is : " $area;;
2) echo "-----Square -- "
read -p "Enter The Side : " s
area=$(( $s * $s ))
echo "Area of the Square is : " $area;;
3) echo "-----Recangle--- "
read -p "Enter The Length : " l
read -p "Enter The Breadth : " b
area=$(( $l * $b ))
echo "Area of the Rectangle is : " $area;;
4) echo "-----Triangle--- "
read -p "Enter the base of the Triangle : " b
read -p "Enter the height of the Triangle : " h
area=`expr "scale=2; 1/2*$b*$h"|bc`
```

```
echo "Area of Triangle is : " $area;;
        5) echo "Bye"
        val=0;;
*)echo "Invalid Input"
esac
done
```

Output



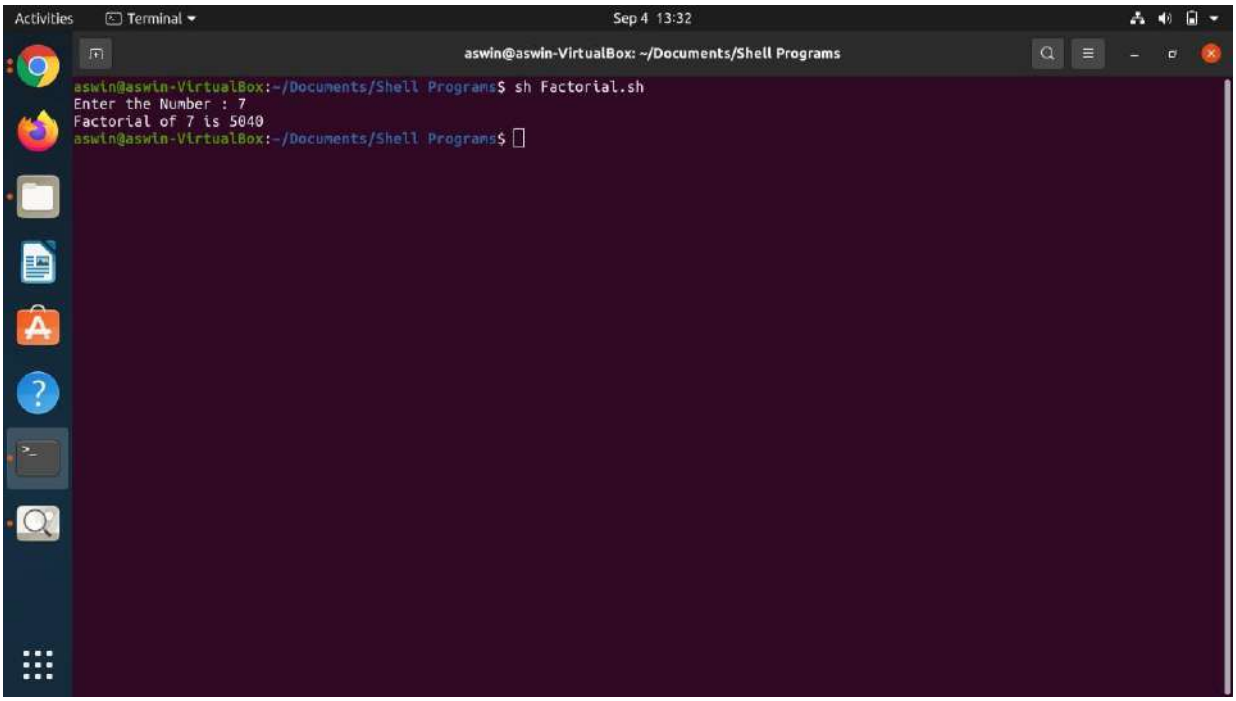


13. Write a shell program to find the factorial of a given number

Source Code

```
#!/bin/bash
read -p "Enter the Number : " n
fact=1
for i in $(seq 2 $n) do
fact=$(( fact*i ))
done
echo "Factorial of $n is $fact"
```

Output

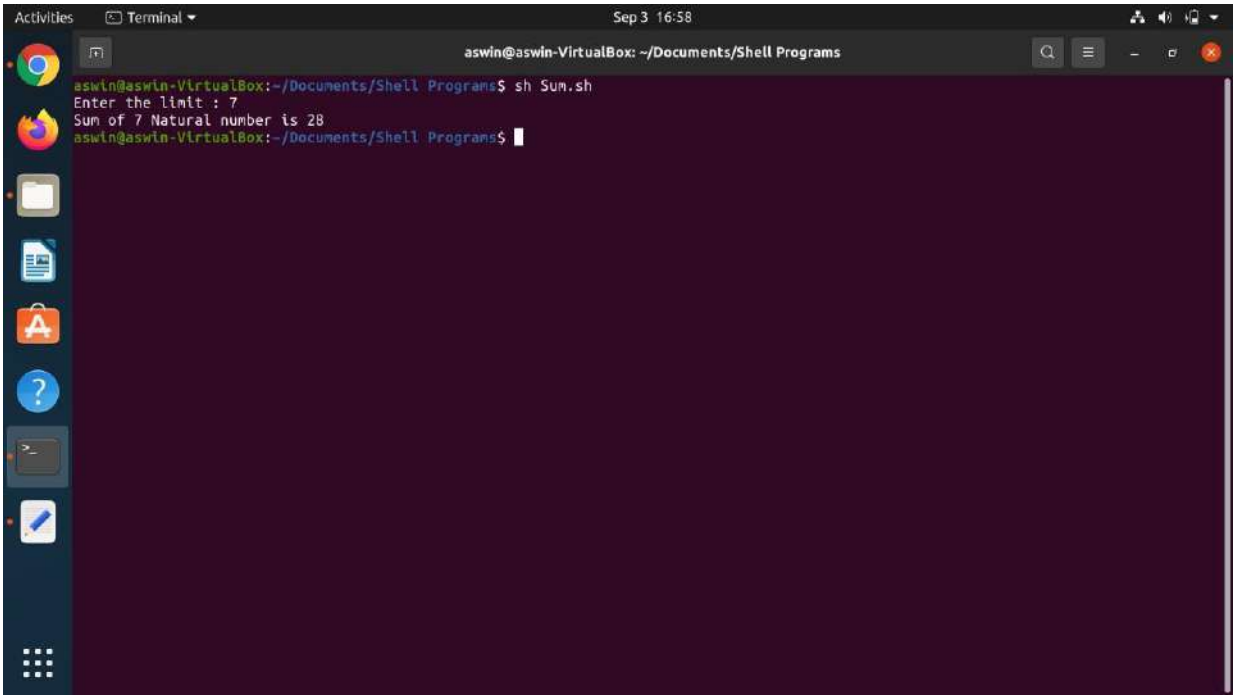


14. Write a Simple Shell script to print the sum of n natural numbers

Source Code

```
#!/bin/bash
read -p "Enter the limit : " n
sum1=0
for i in $(seq 1 $n) do
sum1=$(( sum1+i ))
done
echo "Sum of $n Natural number is $sum1"
```

Output

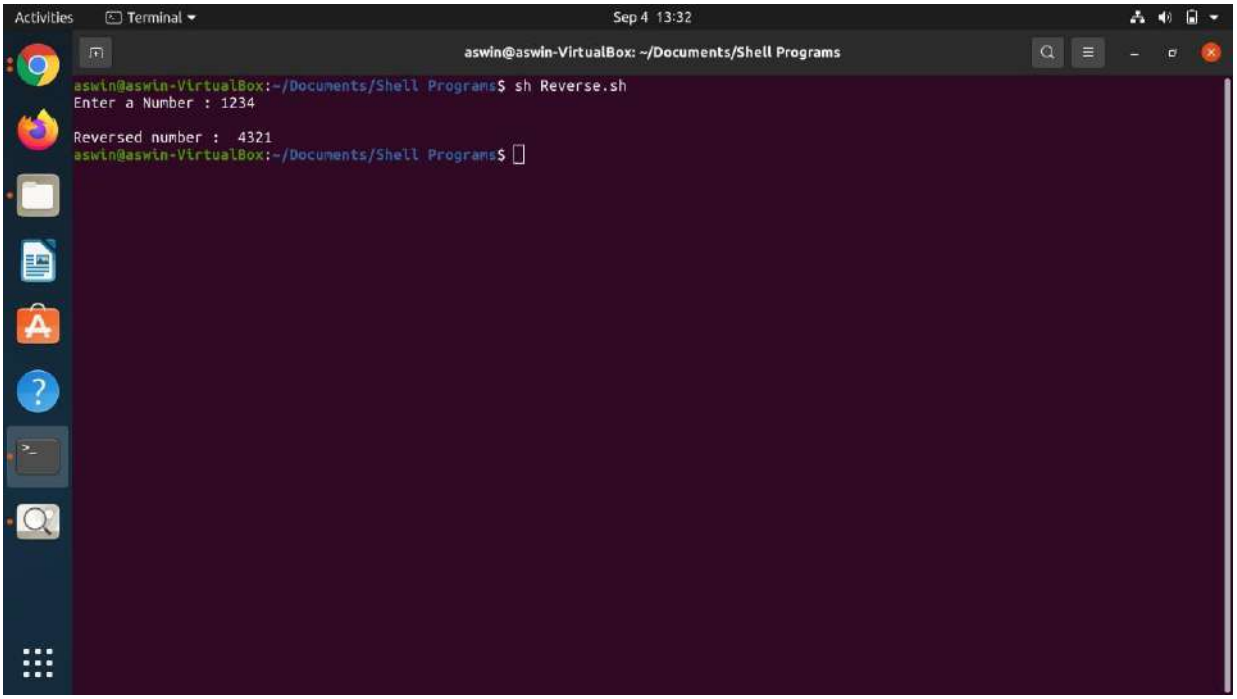


15. Write a shell program to reverse a number.

Source Code

```
#!/bin/bash
read -p "Enter a Number
: " n while [ $n -ne 0 ]
do
rem=$(( $n%10 ))
rev=$((
rev*10+rem ))
n=$(( n/10 ))
do
ne
ec
ho
echo "Reversed number : " $rev
```

Output



EXPIRIMENT-5:

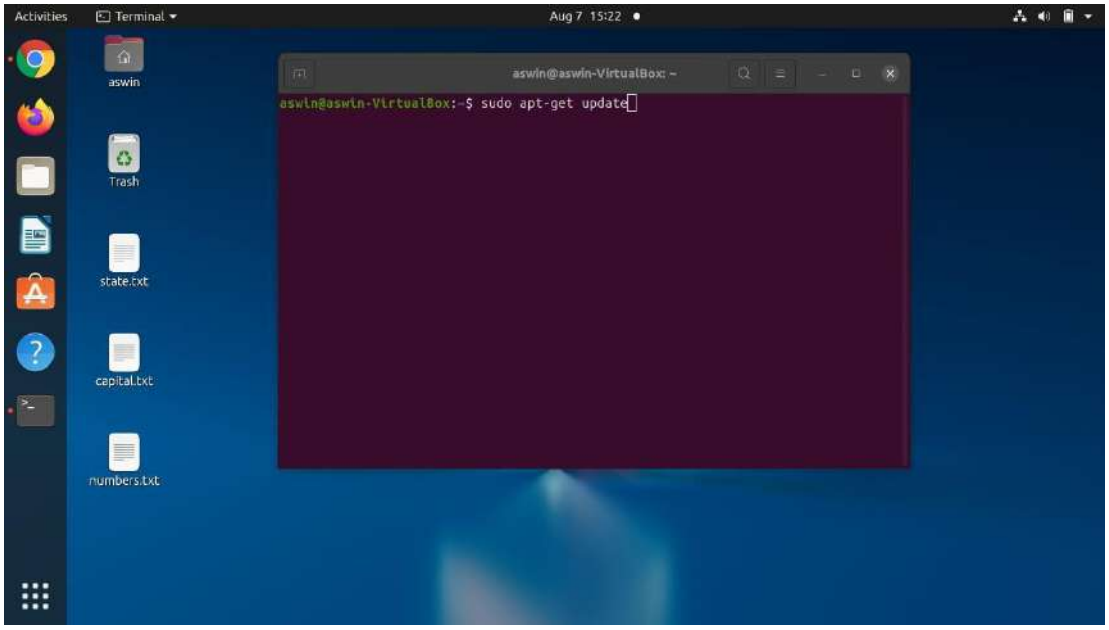
Aim: Installation and configuration of LAMP stack. Deploy an open source application such asphpmyadmin and Wordpress.

ANSWER :-

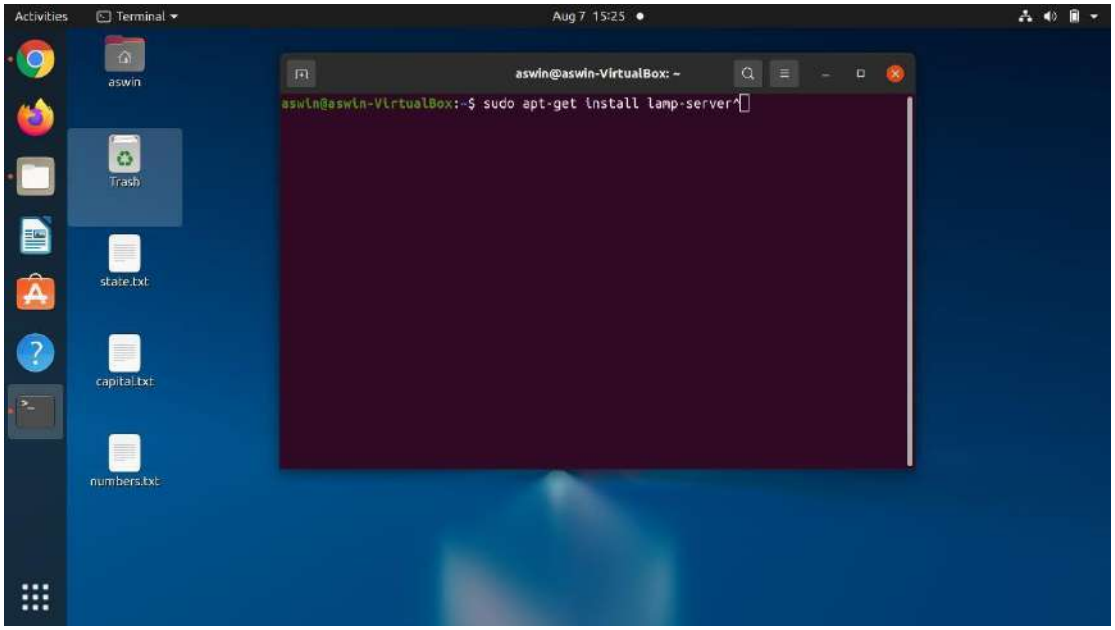
WordPress is an extremely popular open-source technology for making websites and blogs on the internet today. we can use a LAMP (**L**inux, **A**pache, **M**ySQL, and **P**HP) stack, which is one option for a server architecture that supports WordPress by providing the Linux operating system, Apache web server, MySQL database, and PHP programming language. We'll install and set up WordPress via LAMP on a Linux Ubuntu server.

STEP1 :

When setting up our LAMP stack, we only required a very minimal set of extensions in order to get PHP to communicate with MySQL. WordPress and many of its plugins leverage additional PHP extensions.We can download and install some of the most popular PHP extensions for use with WordPress by typing:

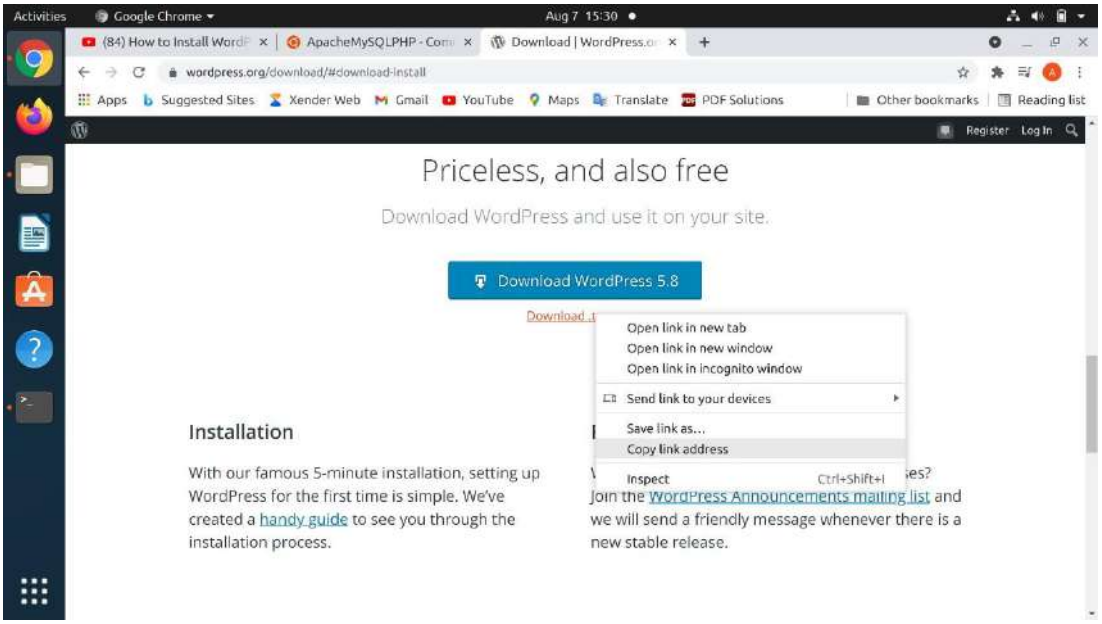


Install the LAMP stack which is one option for a server architecture that supports WordPress by providing the Linux operating system, Apache web server, MySQL database, and PHP programming language.

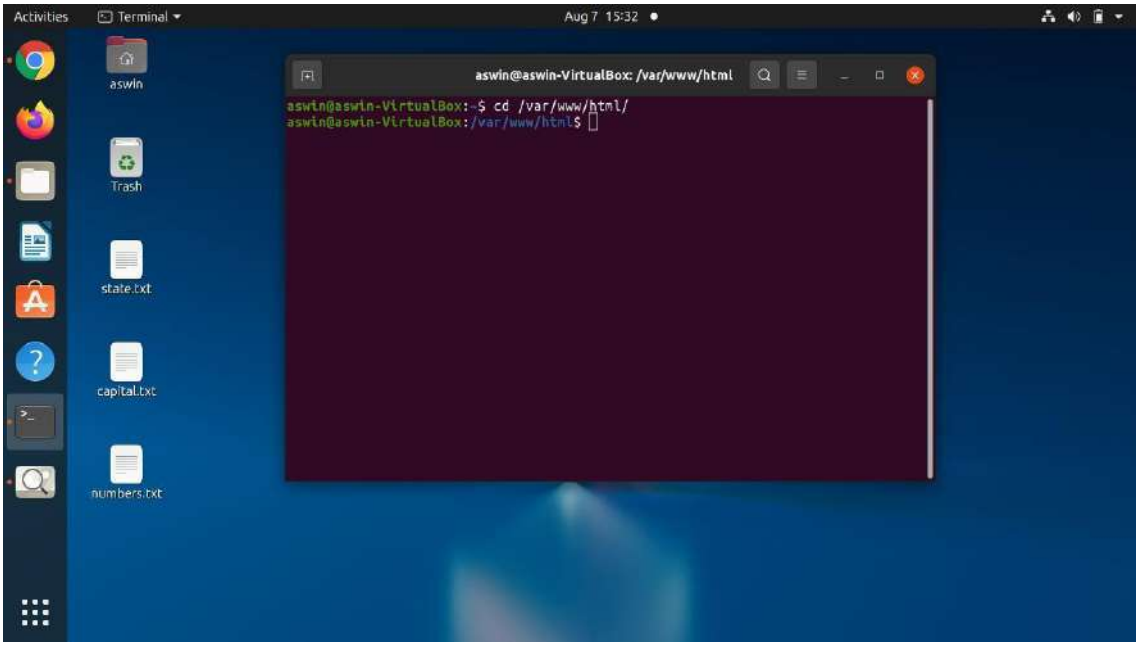


STEP 3:

WordPress link address is coppied from browser

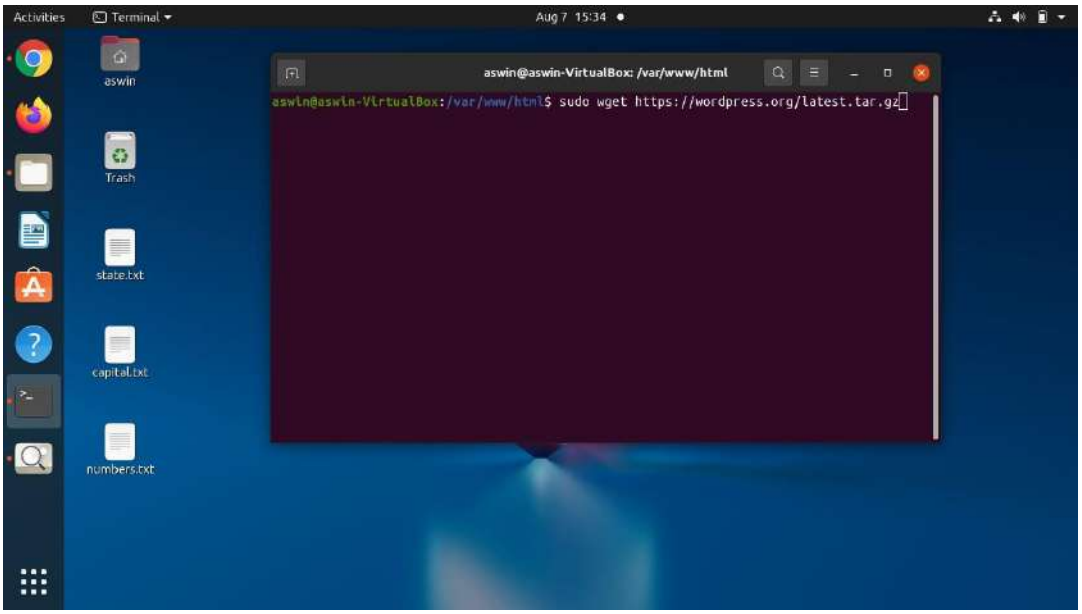


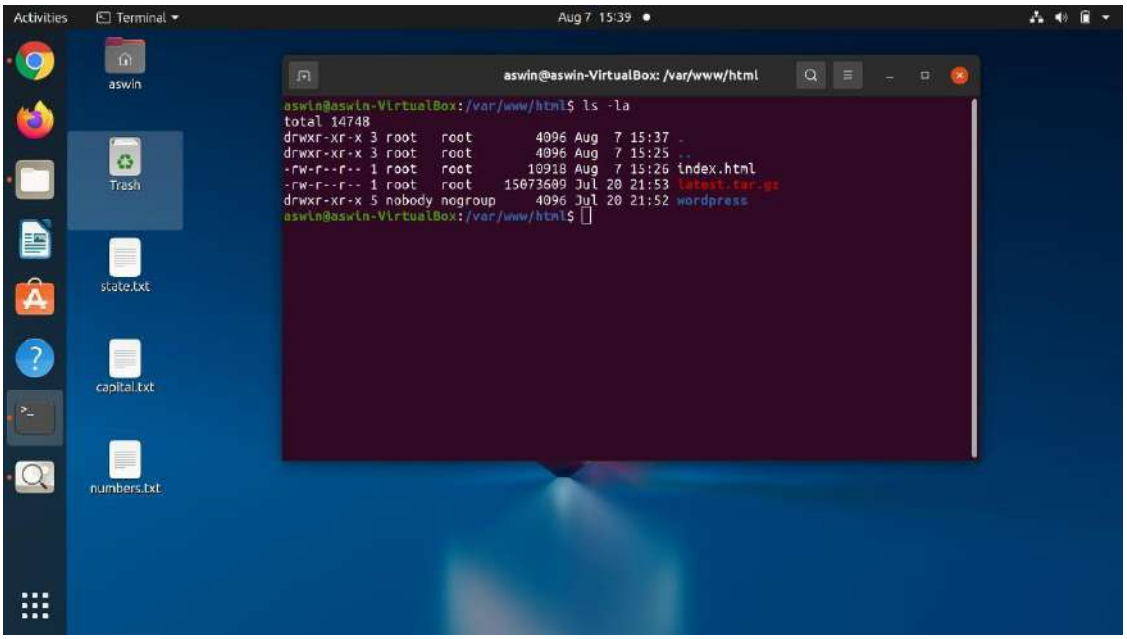
Change the directory to root directory of Apache.



STEP 5:

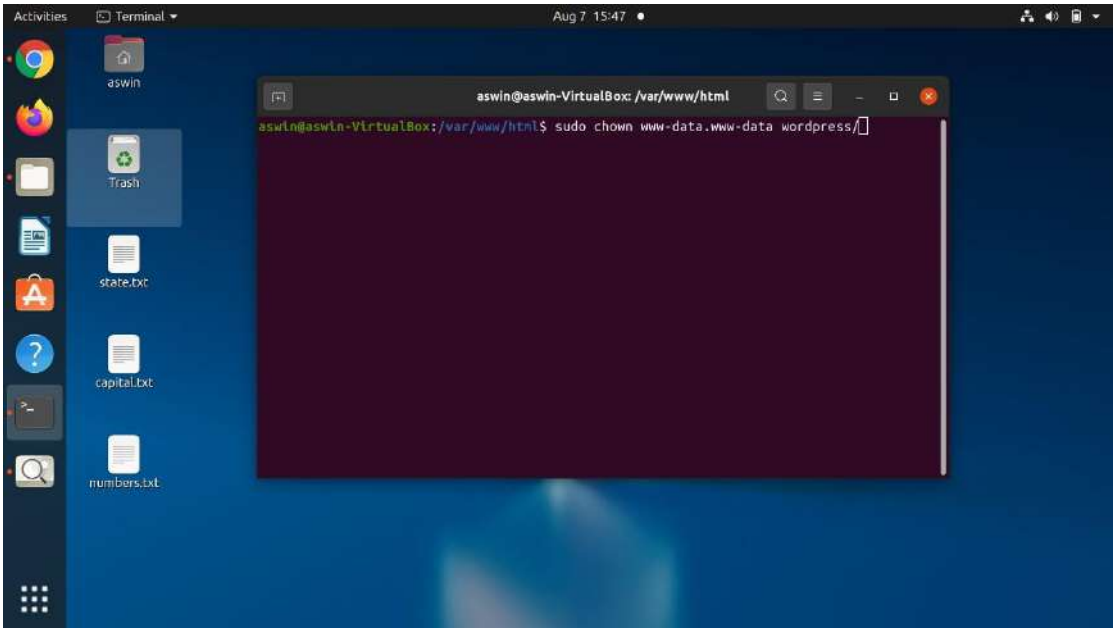
Now that our server software is configured, we can download and set up WordPress. For security reasons in particular, it is always recommended to get the latest version of WordPress from their site.Download the compressed release.

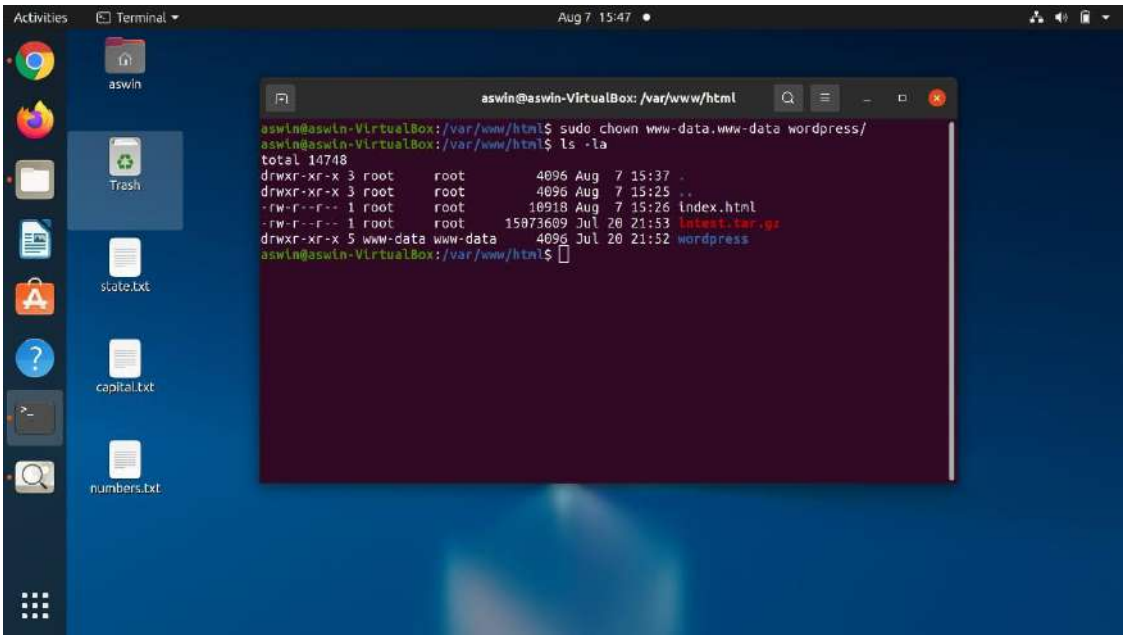




STEP 7

Change owner of the WordPress file.





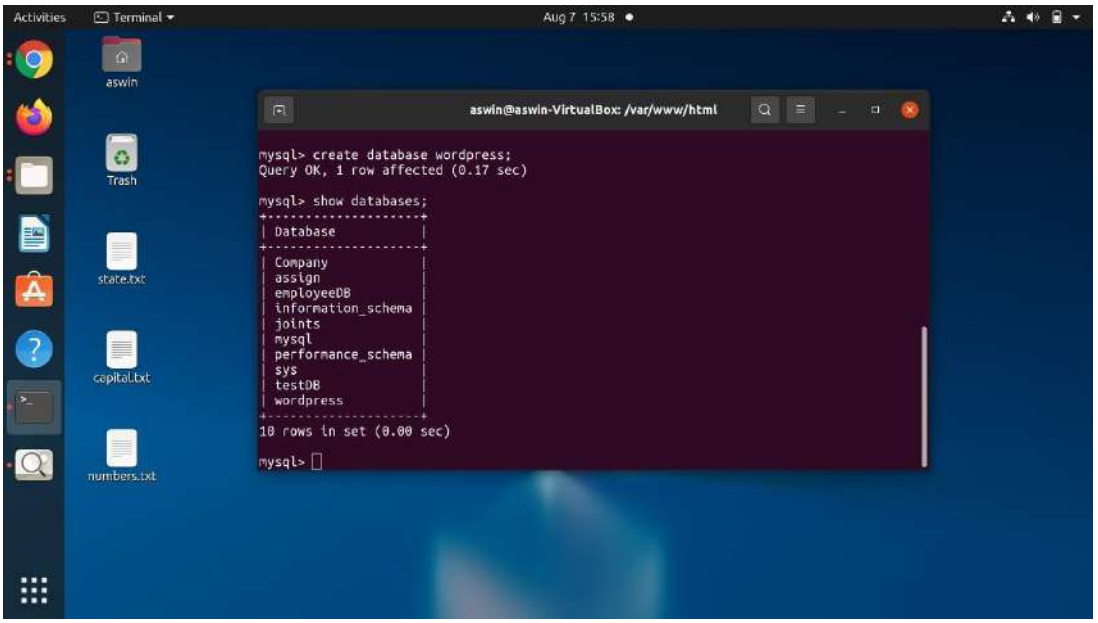
STEP 9

Creating a WordPress Database for Website.Open the terminal and type command

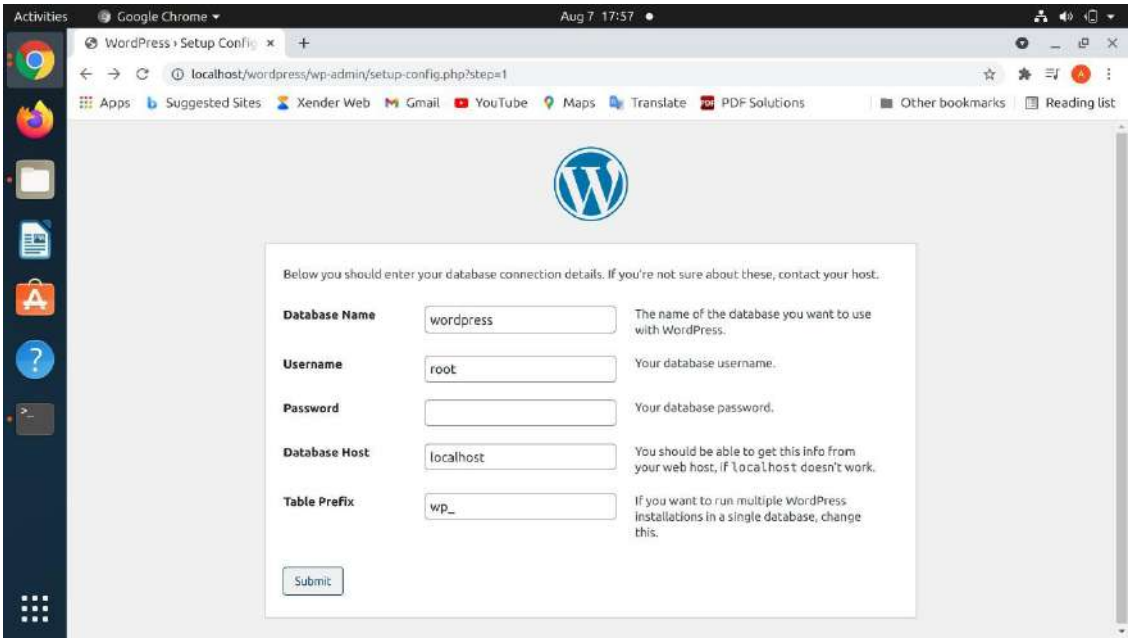
sudo mysql -u root –p

After login, run the following commands to create your site's database

CREATE DATABASE WORDPRESS

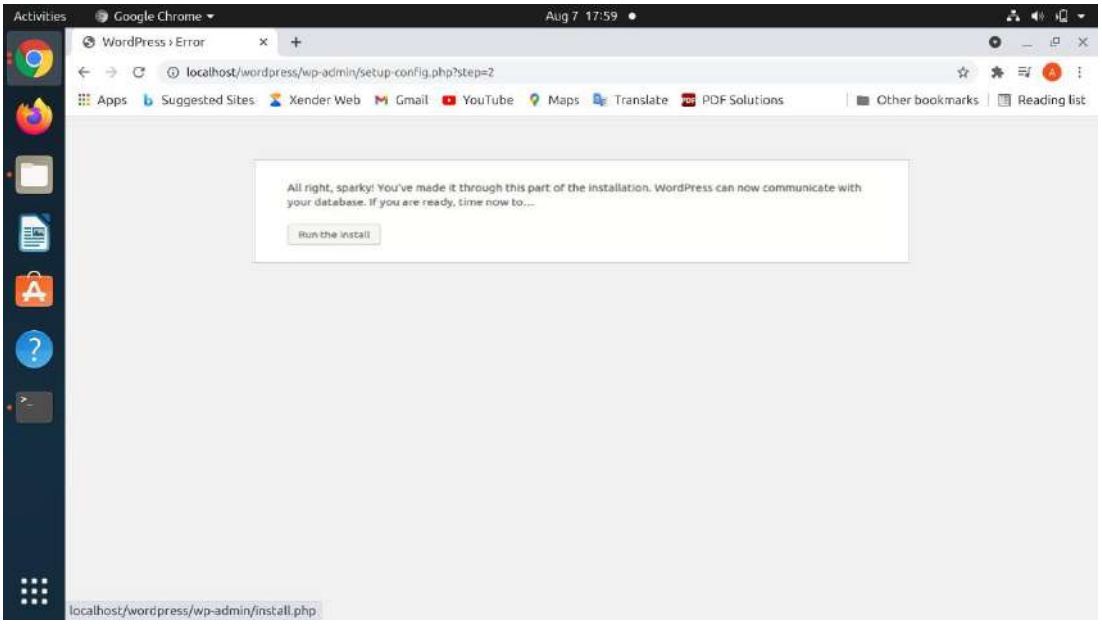


and click submit.

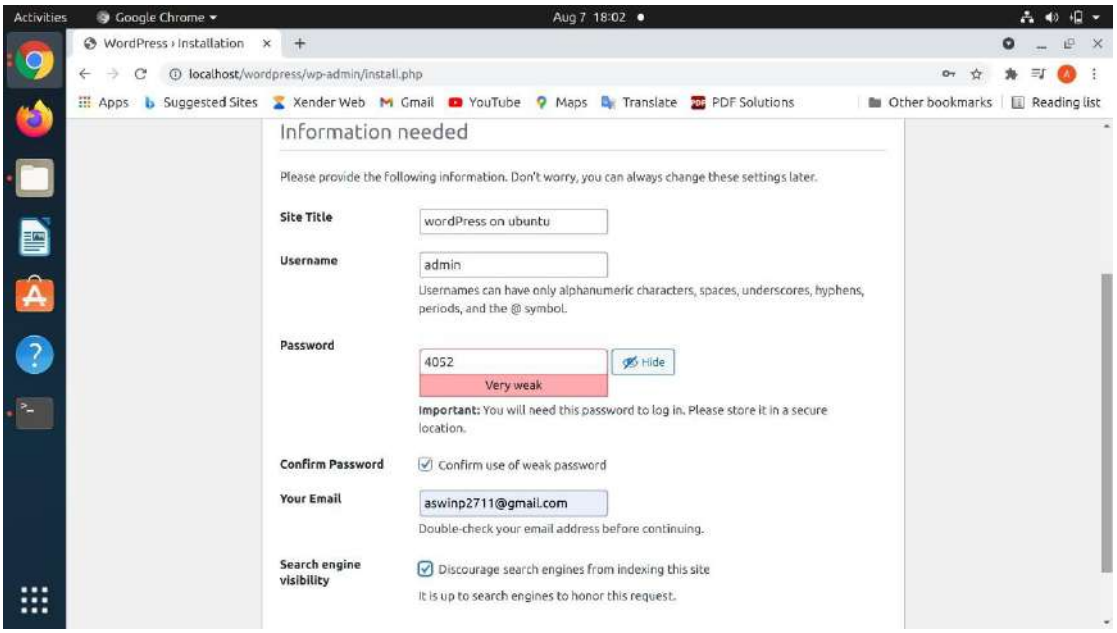


STEP 11:

Click Run the install.

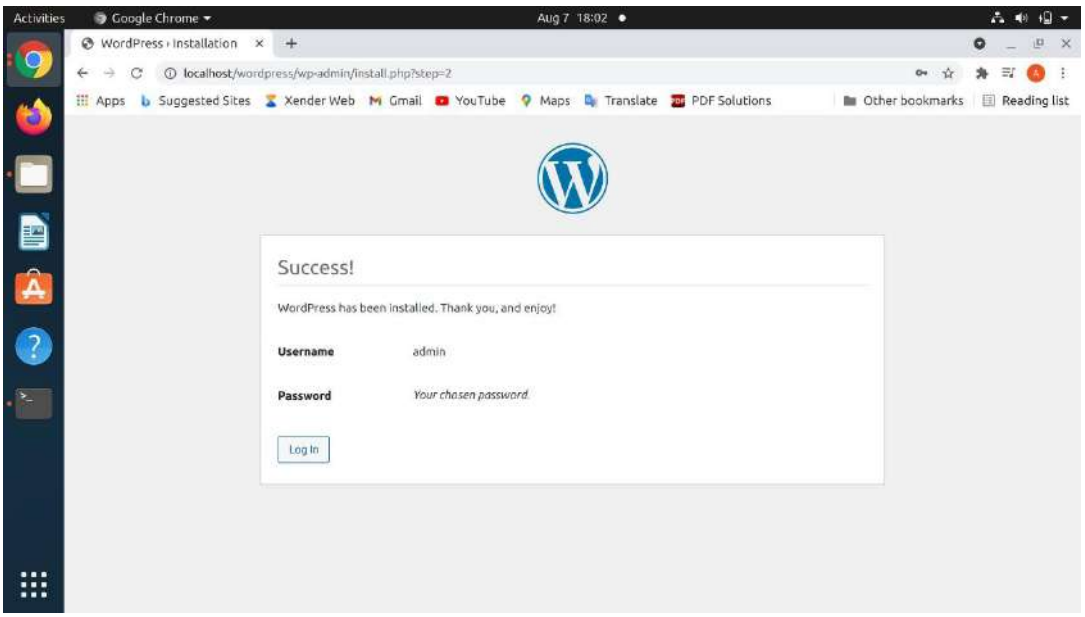


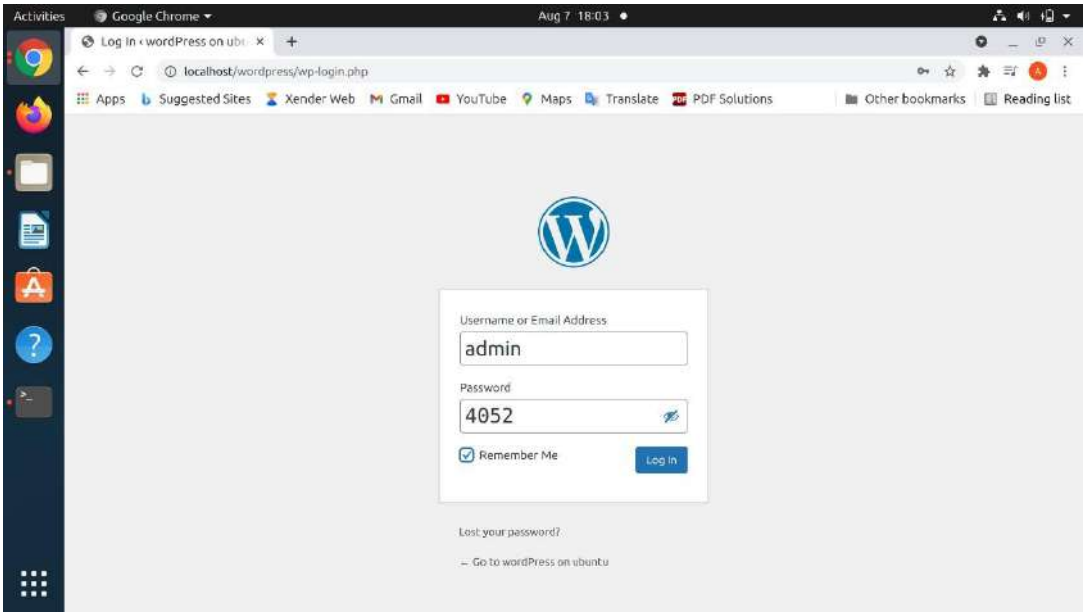
choose a username (it is recommended not to choose something like “admin” for security purposes). A strong password is generated automatically. Save this password or select an alternative strong password.Enter your email address and select whether you want to discourage search engines from indexing your site:



STEP 13

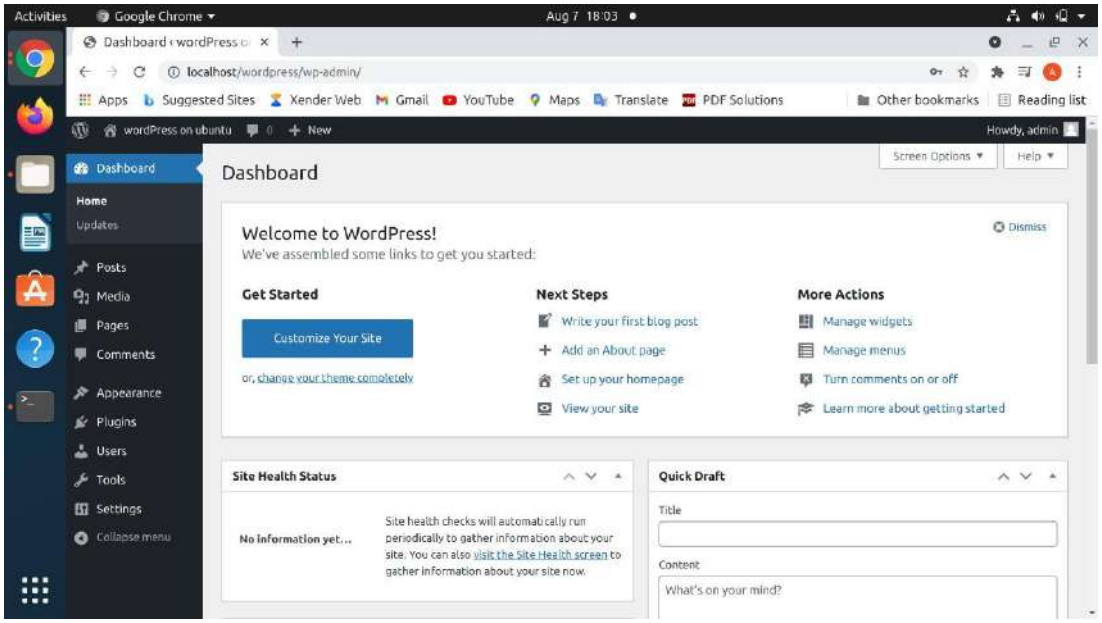
Finally successfully installed! Click login.





STEP 15

Once you log in, you will be taken to the WordPress administration dashboard: We entered into the WordPress dashboard.



We completed our installation successfully and we are now in wordpress. ..... !!!

EXPIRIMENT-6:

Aim: Installation and configuration of common software frame works such as Laravel. (Student should acquire the capability to install and configure a modern framework)

Solution :-

Laravel

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development.

If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks.

Composer

Composer is a tool which includes all the dependencies and libraries. It allows a user to create a project with respect to the mentioned framework (for example, those used in Laravel installation). Third party libraries can be installed easily with help of composer.

All the dependencies are noted in composer.json file which is placed in the source folder.

PHP

PHP is a server-side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

PHP scripts can only be interpreted on a server that has PHP installed.

The client computers accessing the PHP scripts require a web browser only.

A PHP file contains PHP tags and ends with the extension ".php".

Apache

Apache HTTP Server is a free and open-source web server that delivers web content through the internet. It is commonly referred to as Apache and after development, it quickly became the most popular HTTP client on the web. It's widely thought that Apache gets its name from

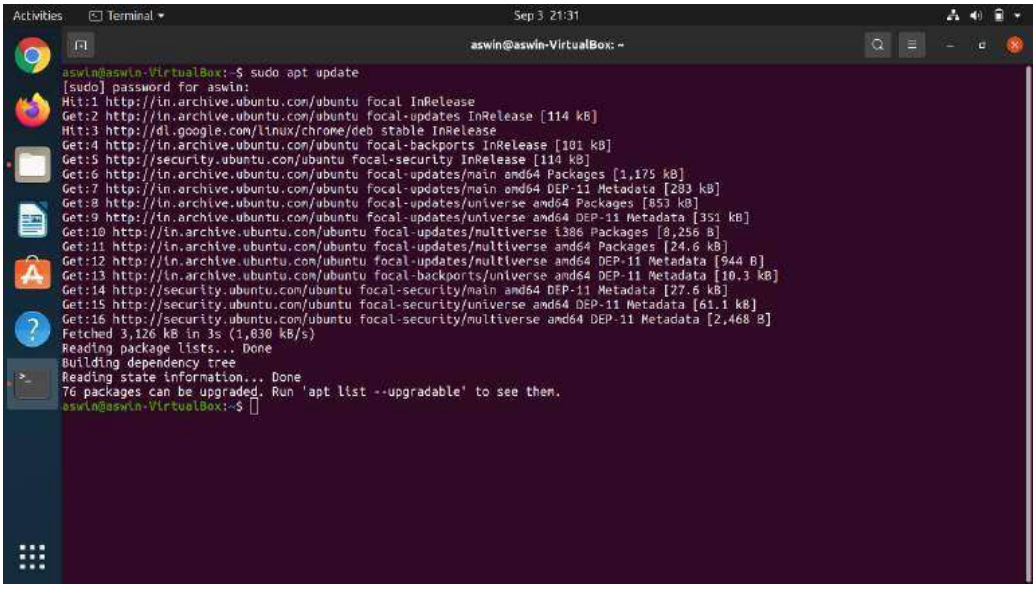
its development history and process of improvement through applied patches and modules but that was corrected back in 2000. It was revealed that the name originated from the respect of the Native American tribe for its resiliency and durability.

Now, before we get too in depth on Apache, we should first go over what a web application is and the standard architecture usually found in web apps.

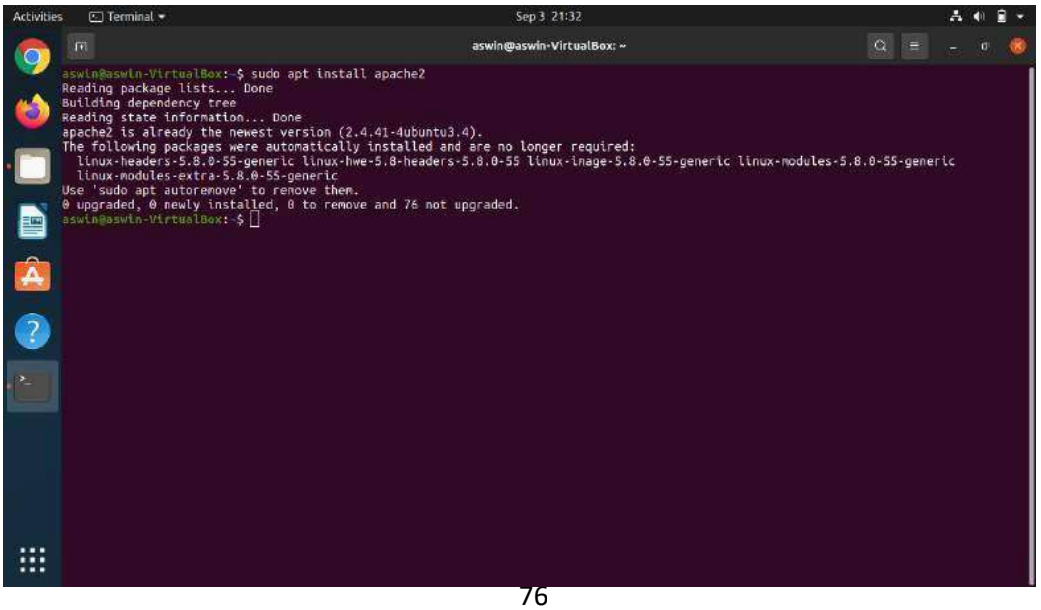
SETUP LARAVEL ON UBUNTU WITH APACHE

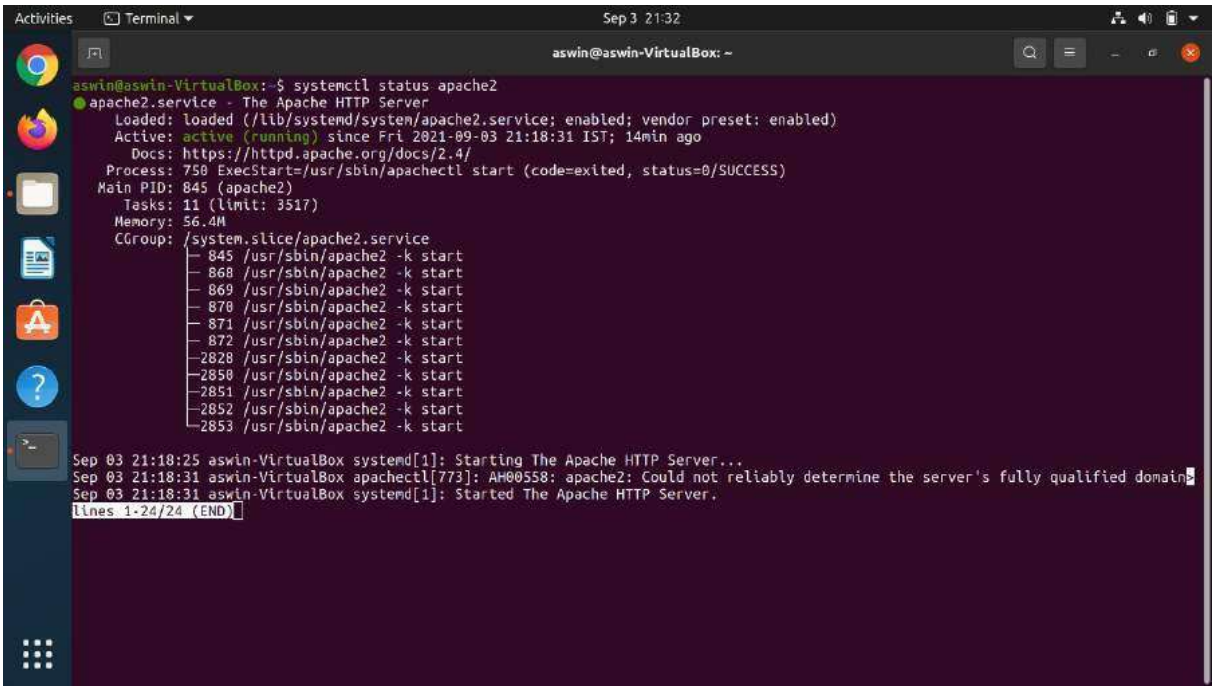
Step-1: Install Apache Web Server

- Let's open up a Terminal and do first thing first update your package list using Sudo apt update command.

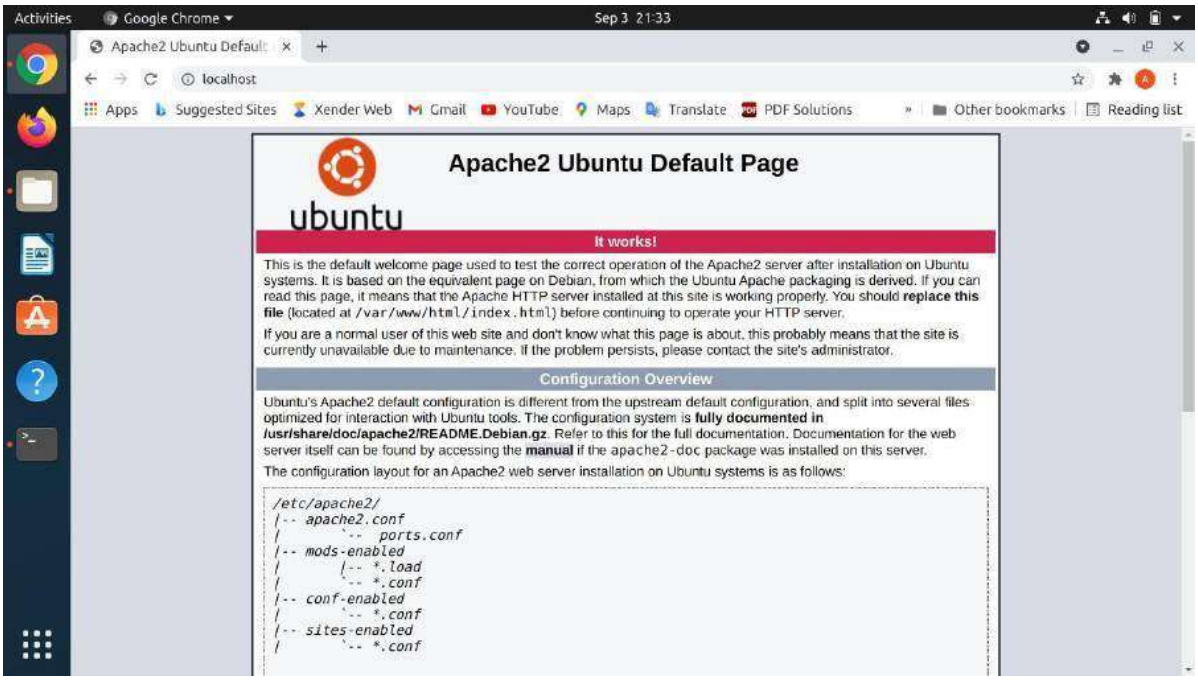


- After updating your package list install apache webserver. So, go ahead and type sudo apt install apache2 then hit the enter key. Press y key to proceed. You can also setup laravel with Nginx instead of the apache web server.



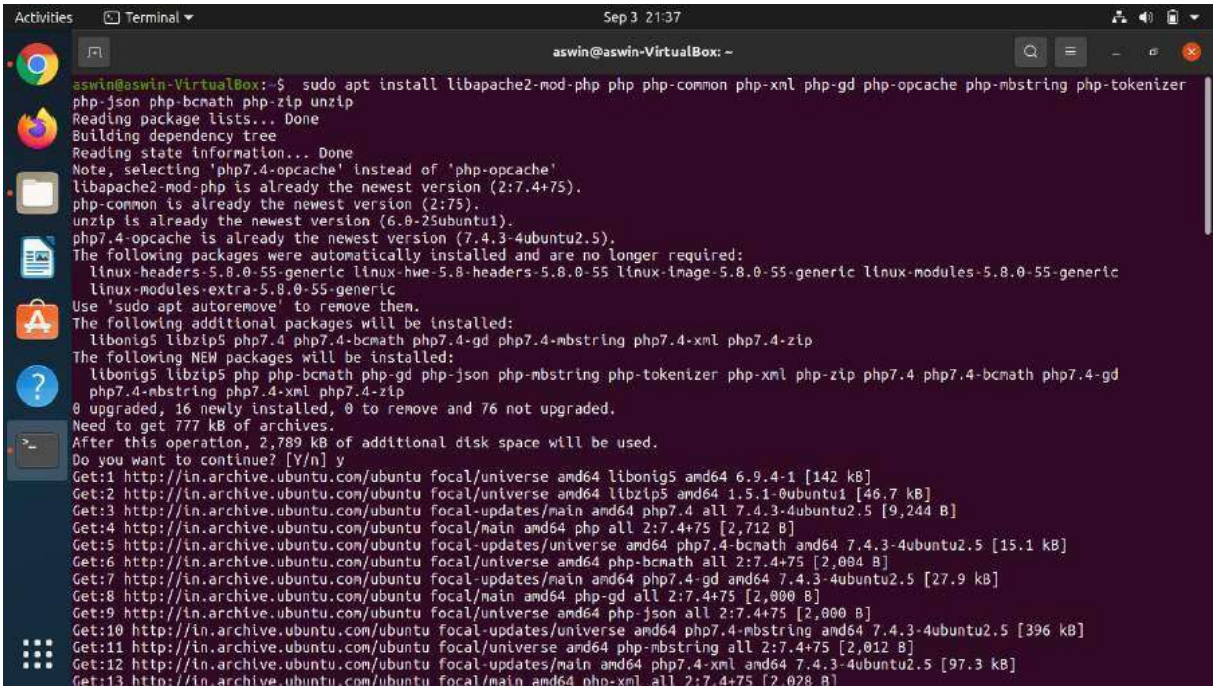


- Check Apache server is working

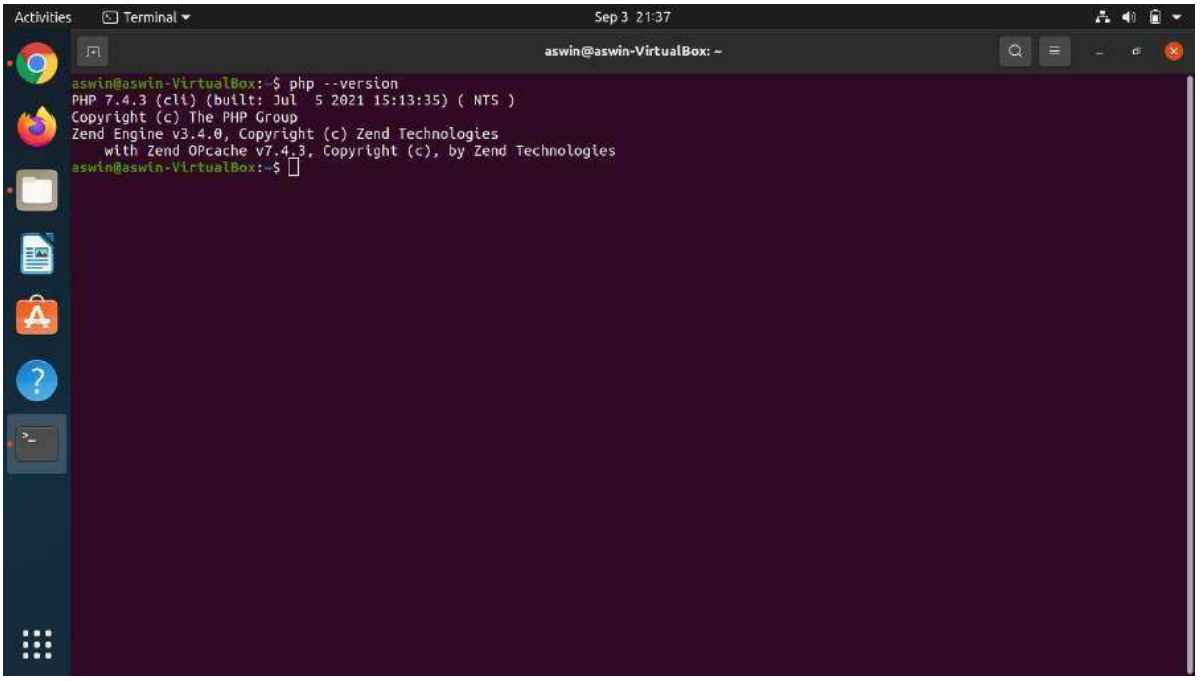


Step-2: Install and Configure PHP 7.4

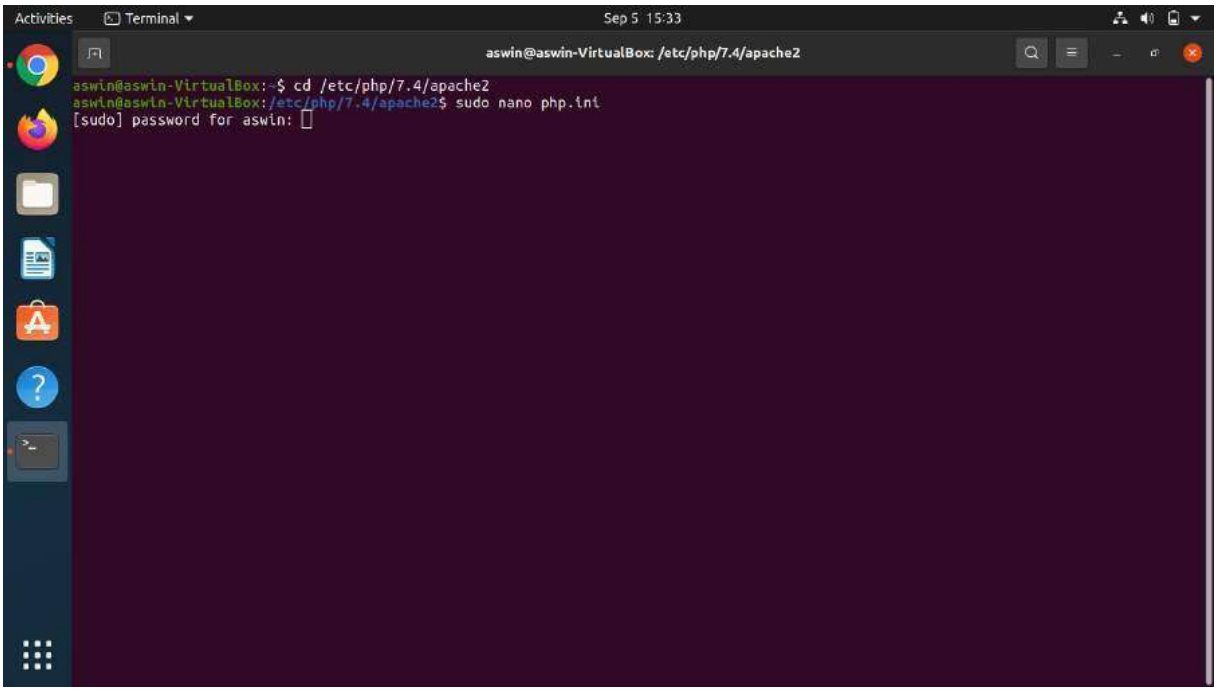
- To install Laravel 8.x, at least you must have PHP >= 7.3 on your system. And by default, the official Ubuntu 20.04 repository provides PHP 7.4 packages. Install PHP
  7.4 packages using the apt command

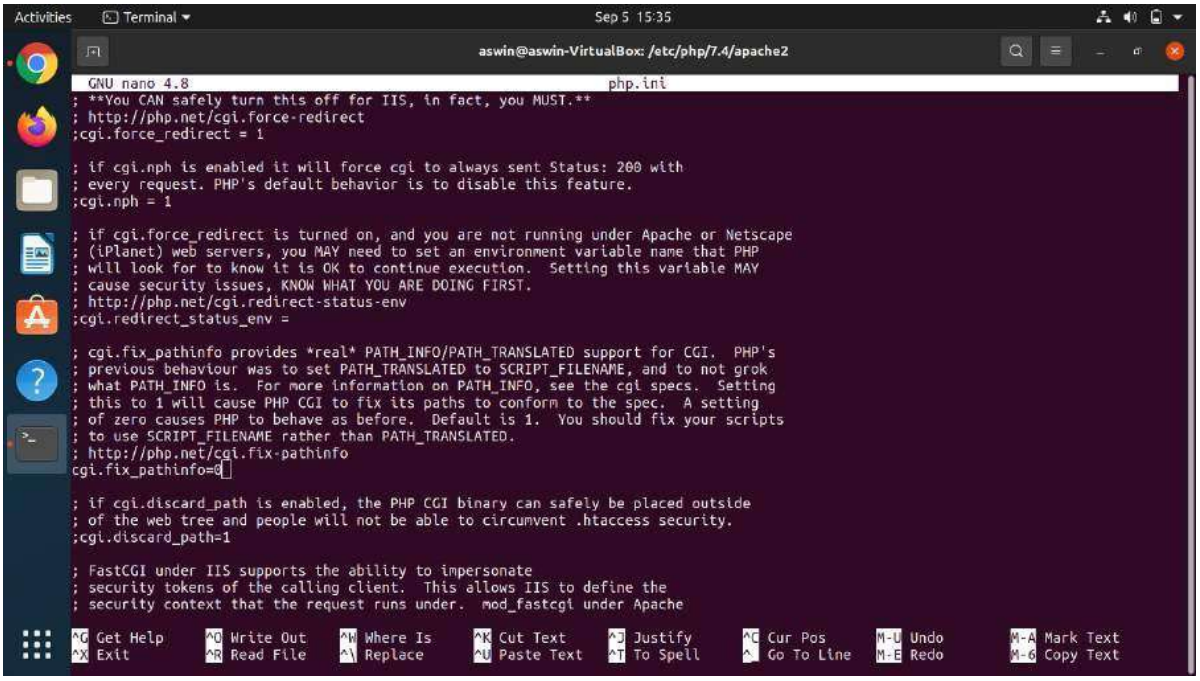


- You can check your PHP version using it.

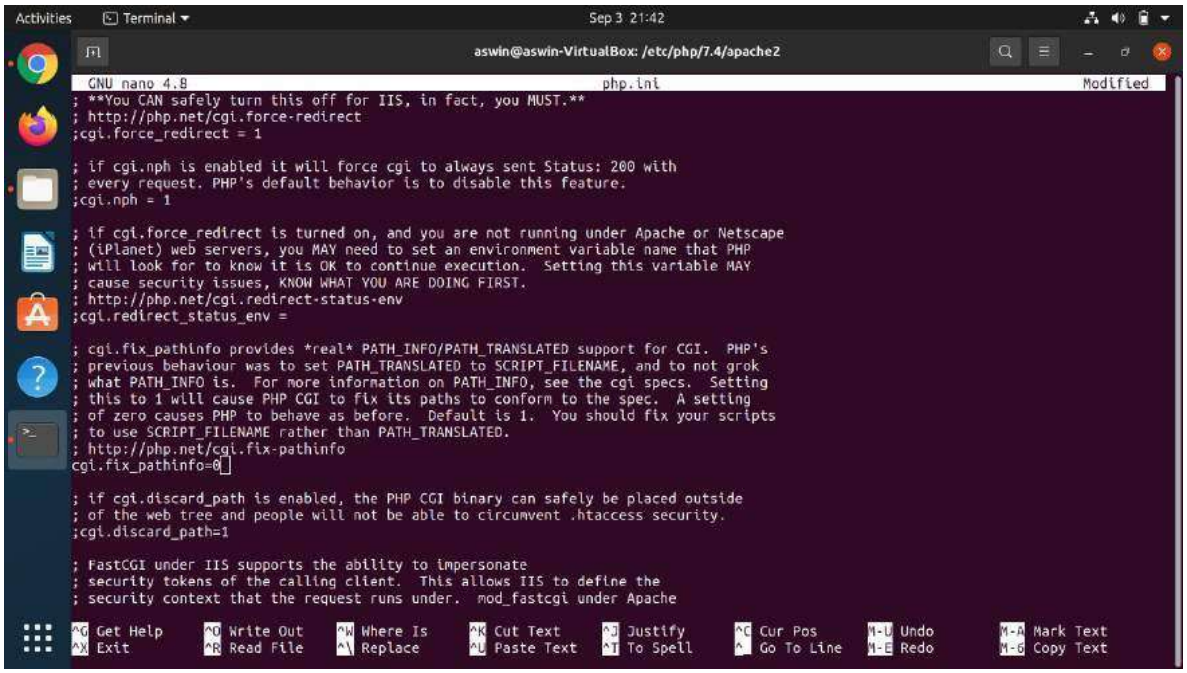


- Now go ahead and make tweak changes in PHP ini file and set cgi.fix_pathinfo set to be 0. If this number is kept as a 1, the php interpreter will do its best to process the file that is as near to the requested file as possible. This is a possible security risk. If this number is set to 0, conversely, the interpreter will only process the exact file path—a much safer alternative.



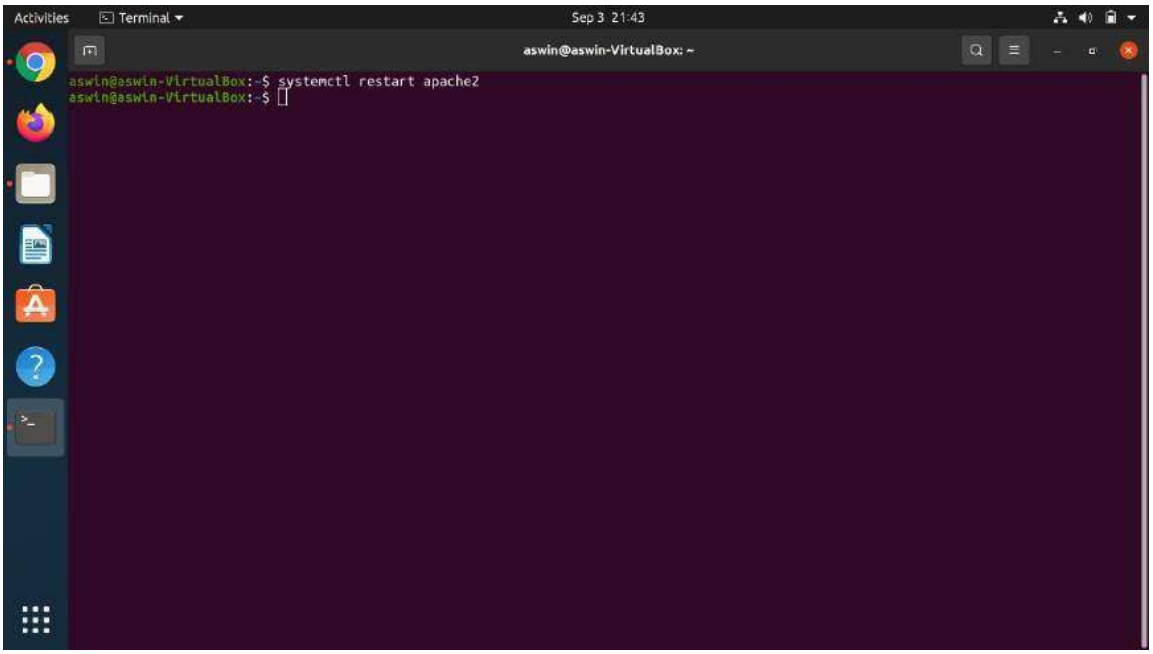


- Press ctrl+w and search for the word “cgi.fix” the uncomment the line and set it to 0.



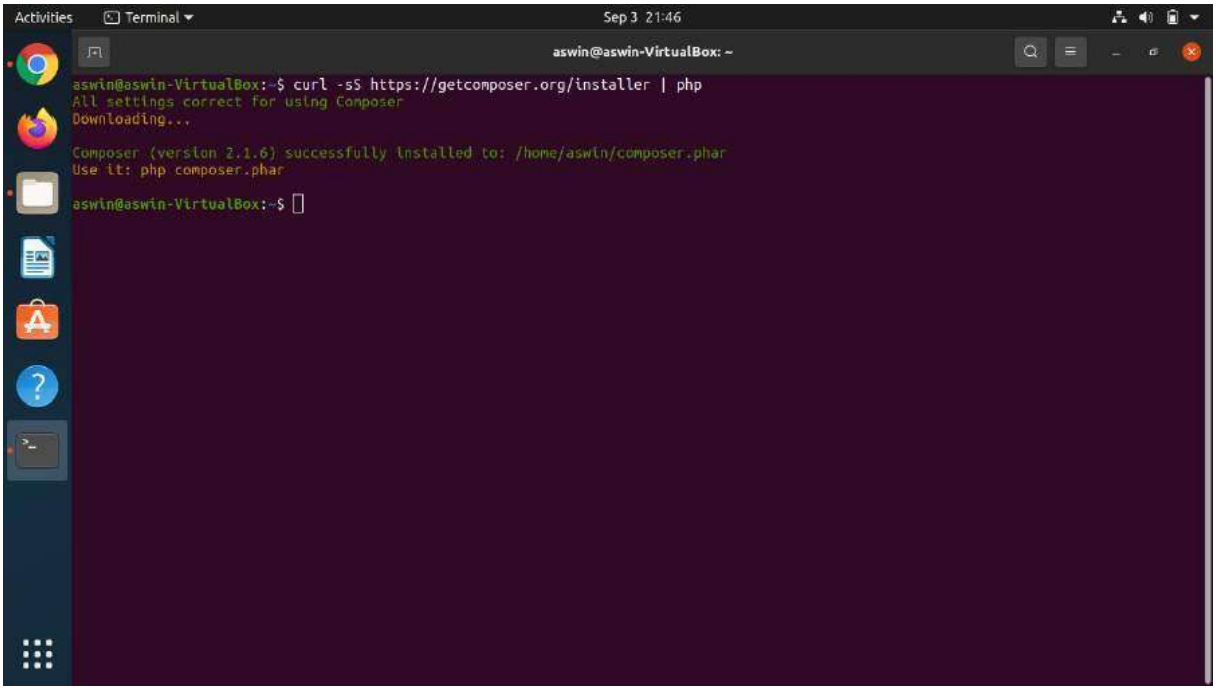
Press Ctrl + x then y to Save and Exit.

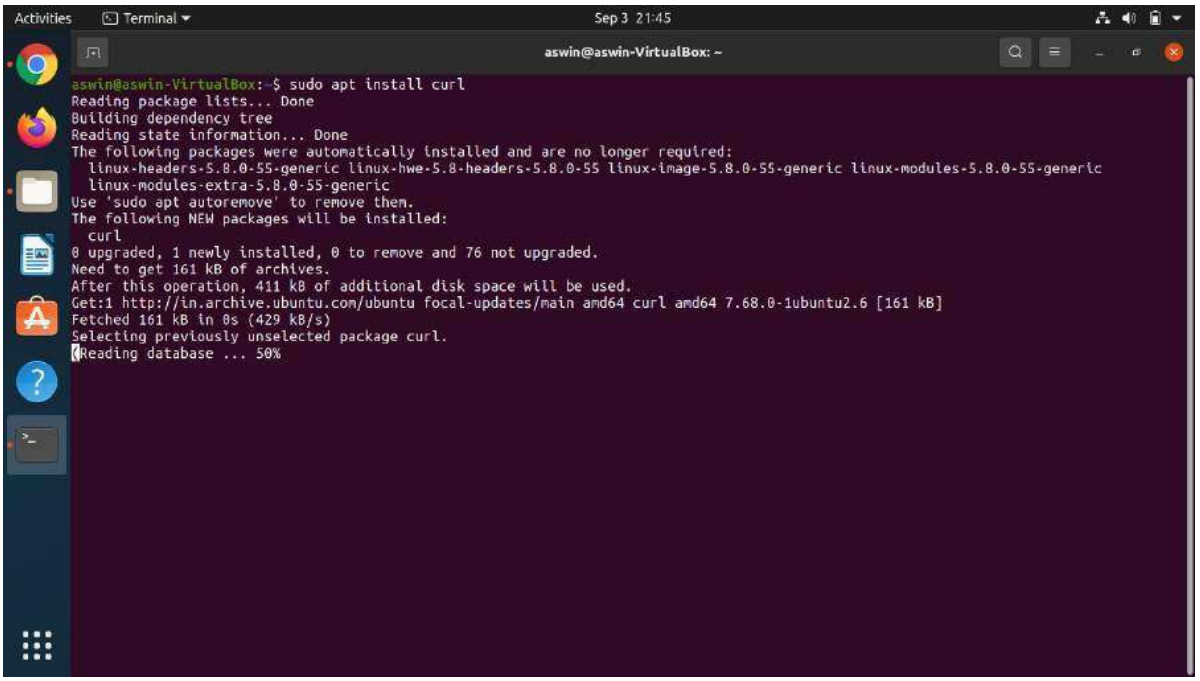
- Now Restart The apache service.

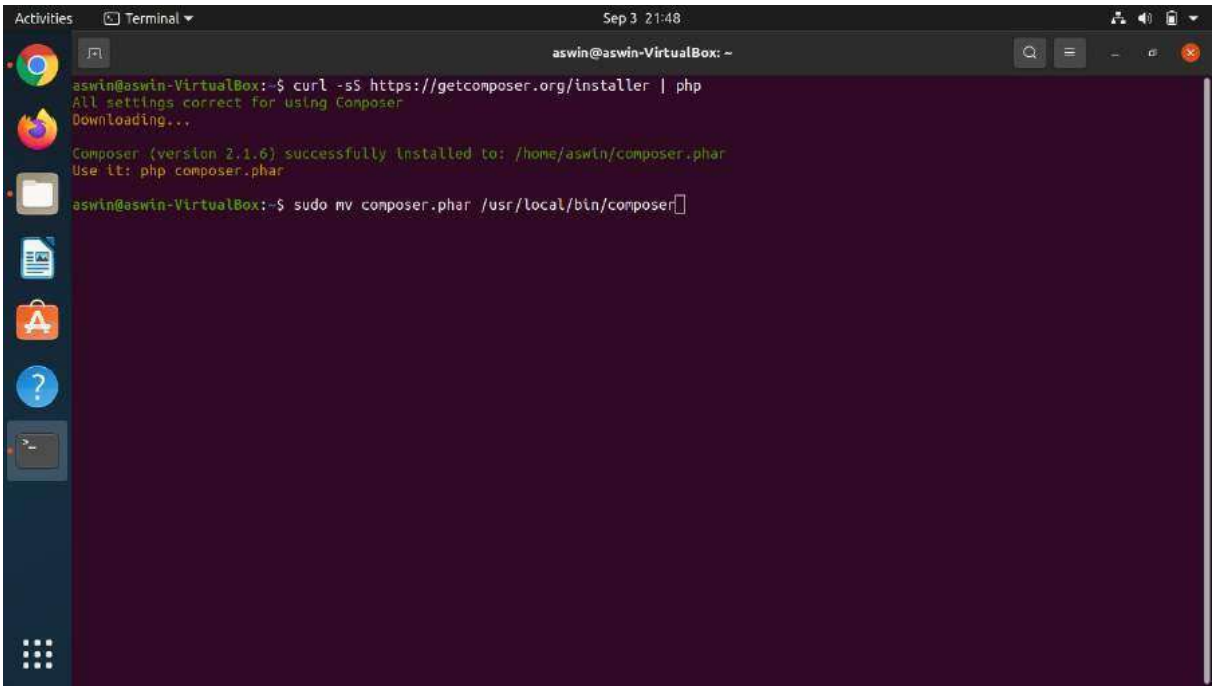


Step-3: Install Composer PHP Packages Management

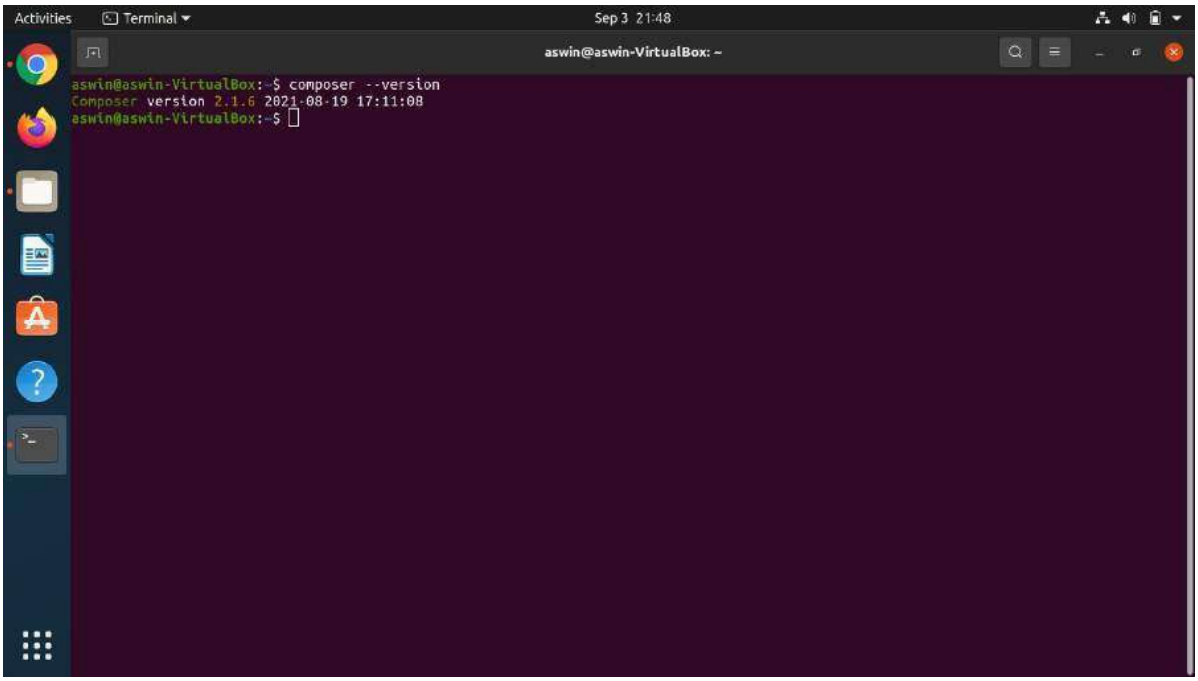
- install the composer package manager go ahead and download and install Composer. and move the composer .phar file to usr/local/bin/composer directory.





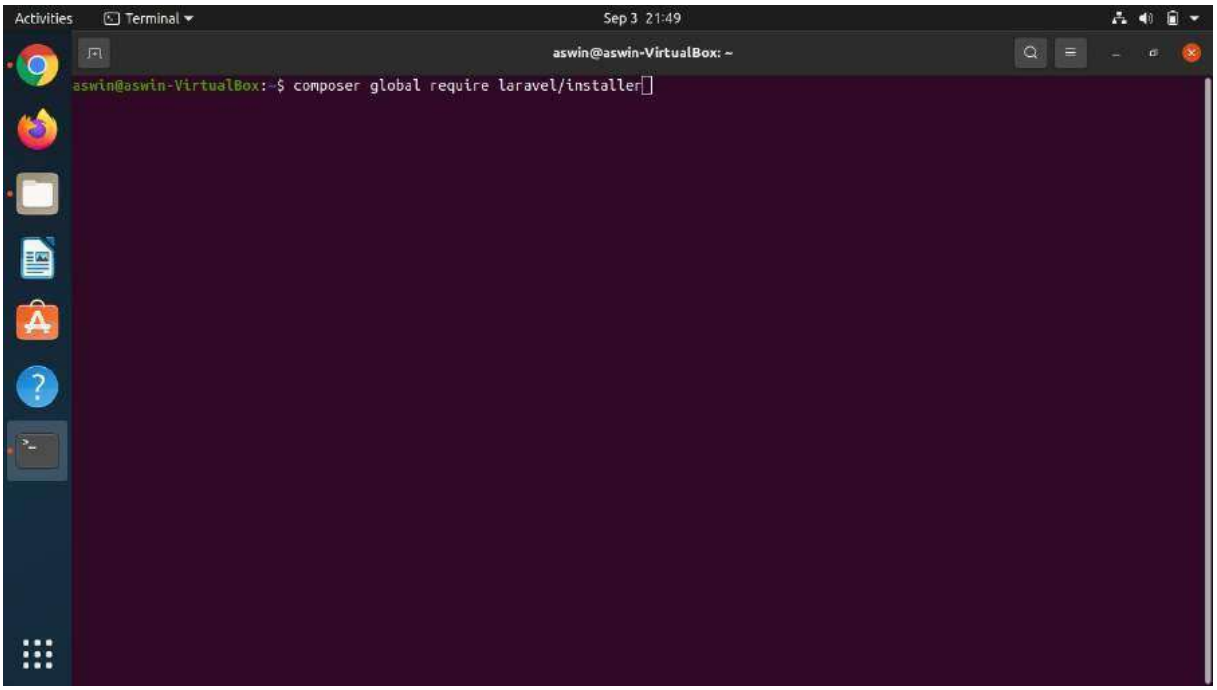


- You can check your installed composer version by typing the composer – version.

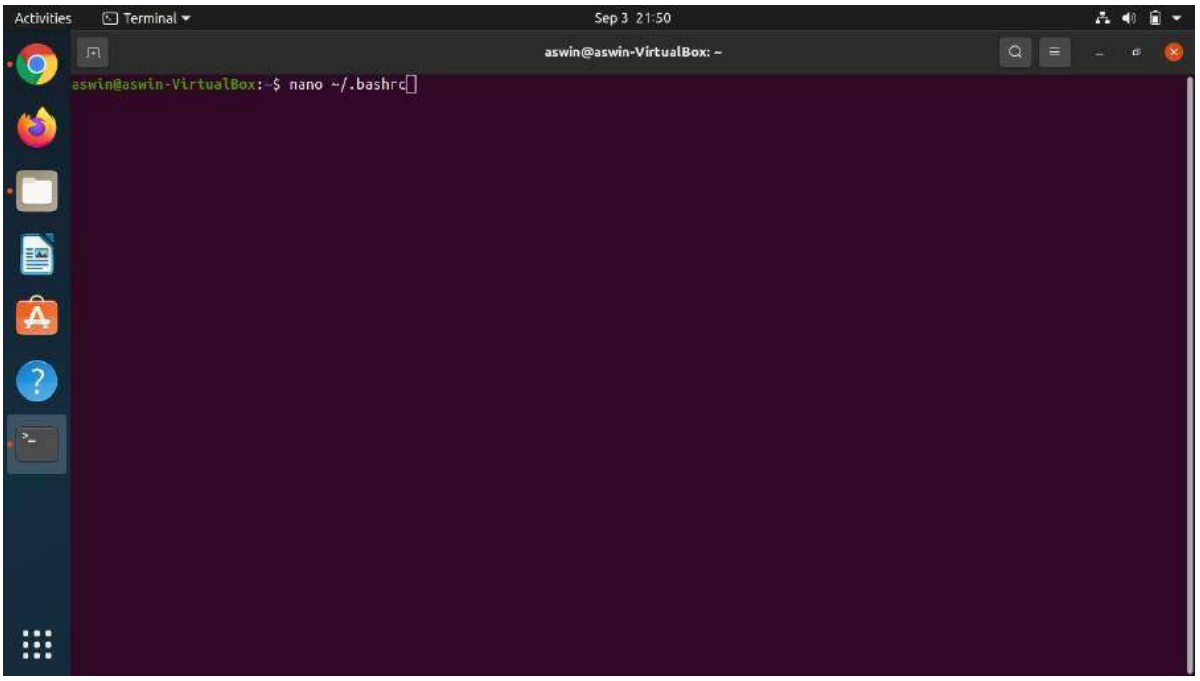


Step-4: Install Laravel 8.x on Ubuntu 20.04

- Now install Laravel Framework using composer, just type composer global require Laravel/installer It will take a while to complete download its dependencies.

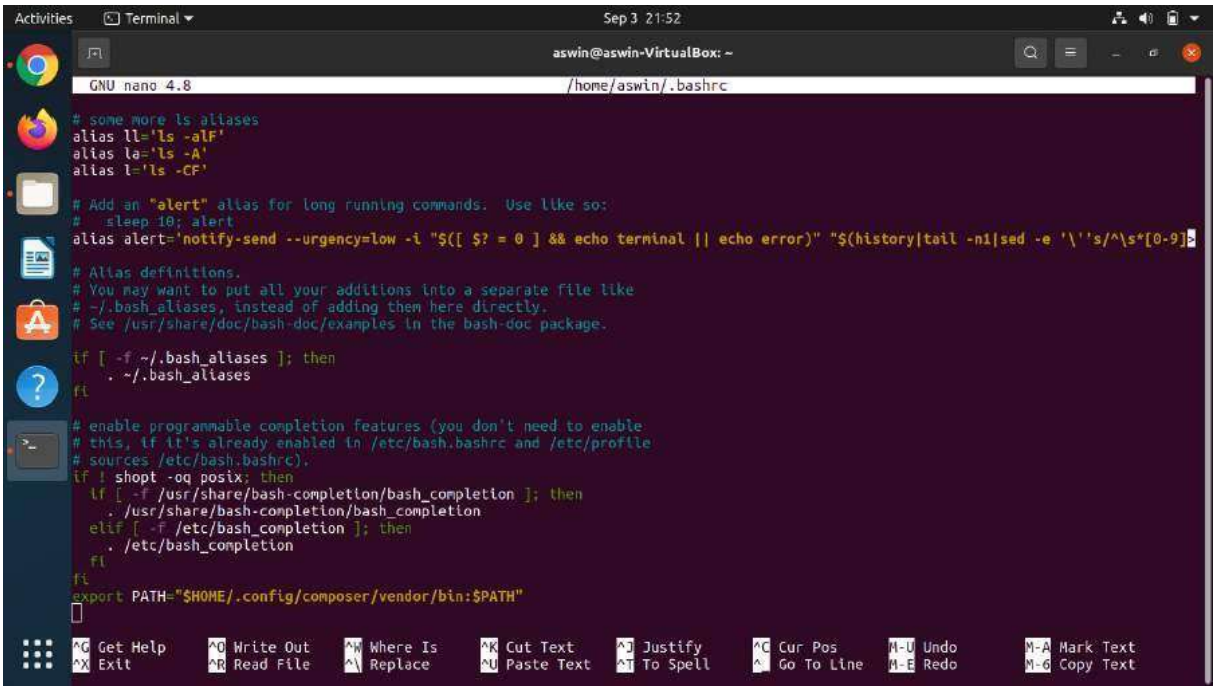


- As you had seen above image, all packages have been installed on the '~/.config/composer' directory. Next, we need to add the 'bin' directory to the PATH environment through the ~/.bashrc configuration. So Now Edit the ~/.bashrc configuration using nano command.

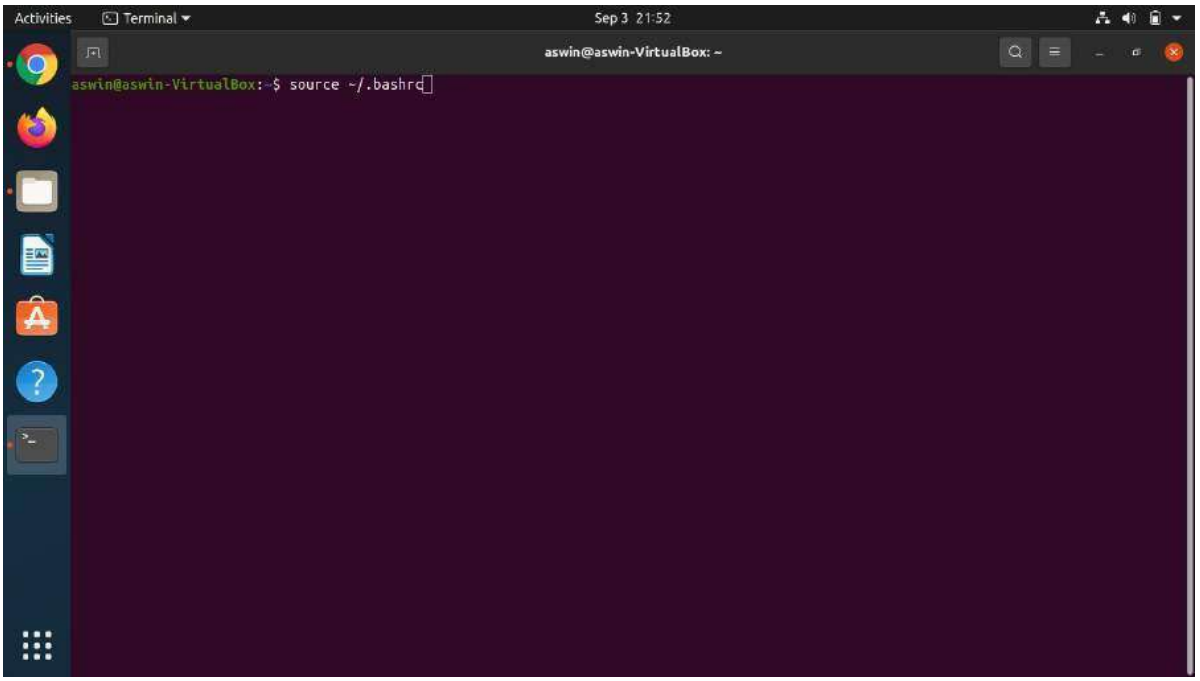


And add the following line at the end of the file.

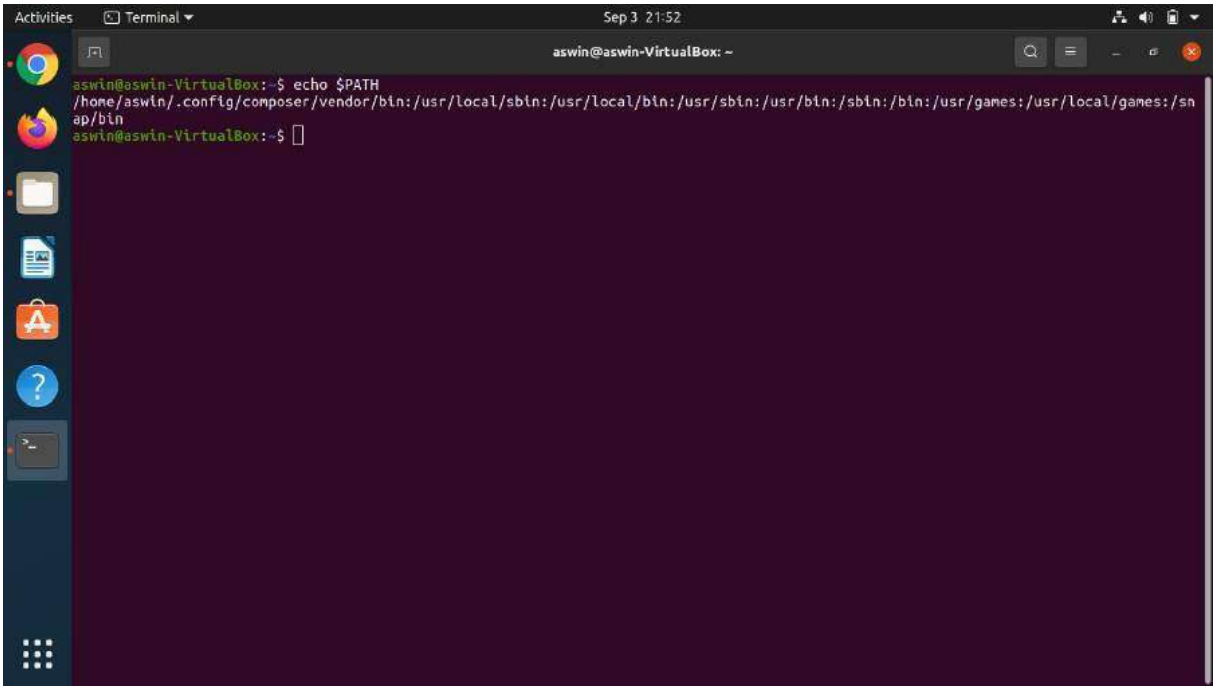
```
export PATH="$HOME/.config/composer/vendor/bin:$PATH"
```



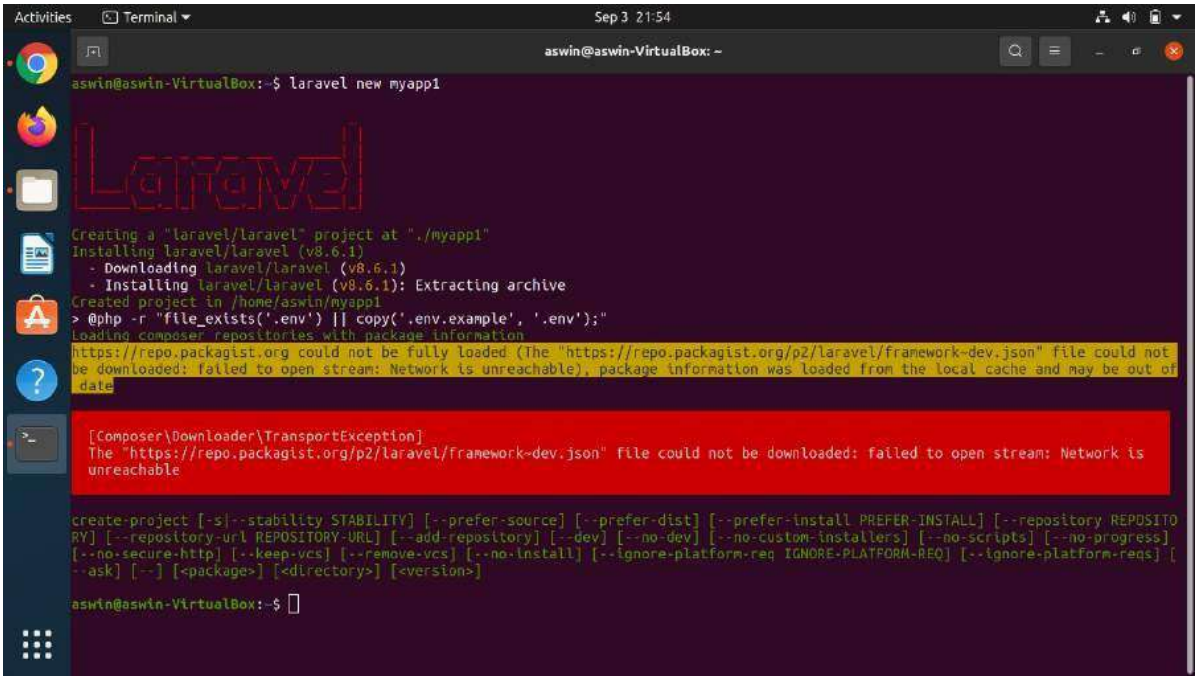
- Now reload your bashrc configuration using the source command.



- Now echo $PATH. It will return your “Bin” directory path for the Composer package.

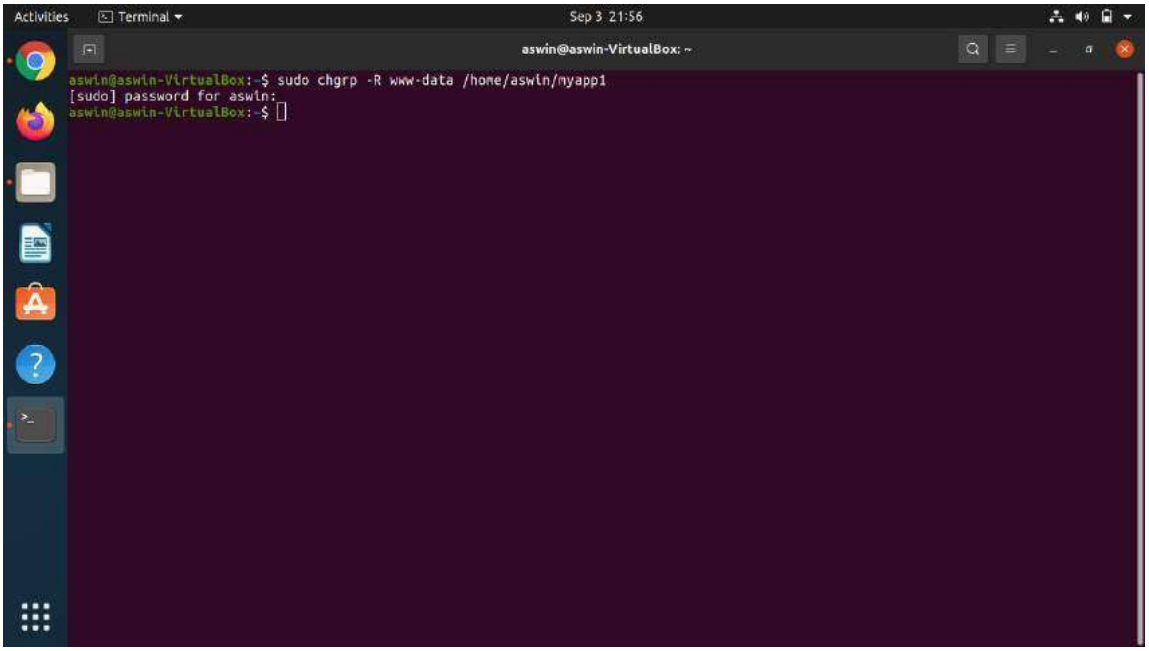


- The ‘bin’ directory for the composer packages has been added to the $PATH environment variable. And as a result, you can use the command ‘laravel’ to start and create a new project. Now go ahead and type Laravel new then your project name to start a new Laravel project.



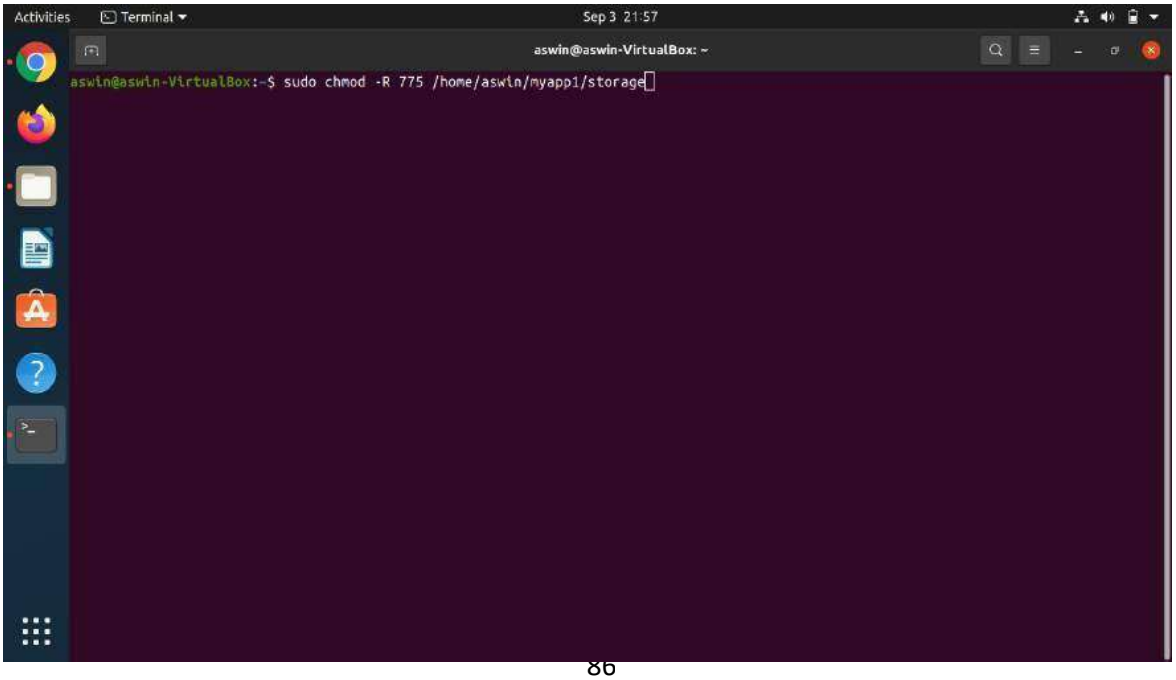
Step-5: Finally Configure Apache for Laravel and test it

- Add your project directory to www-data group use the following command

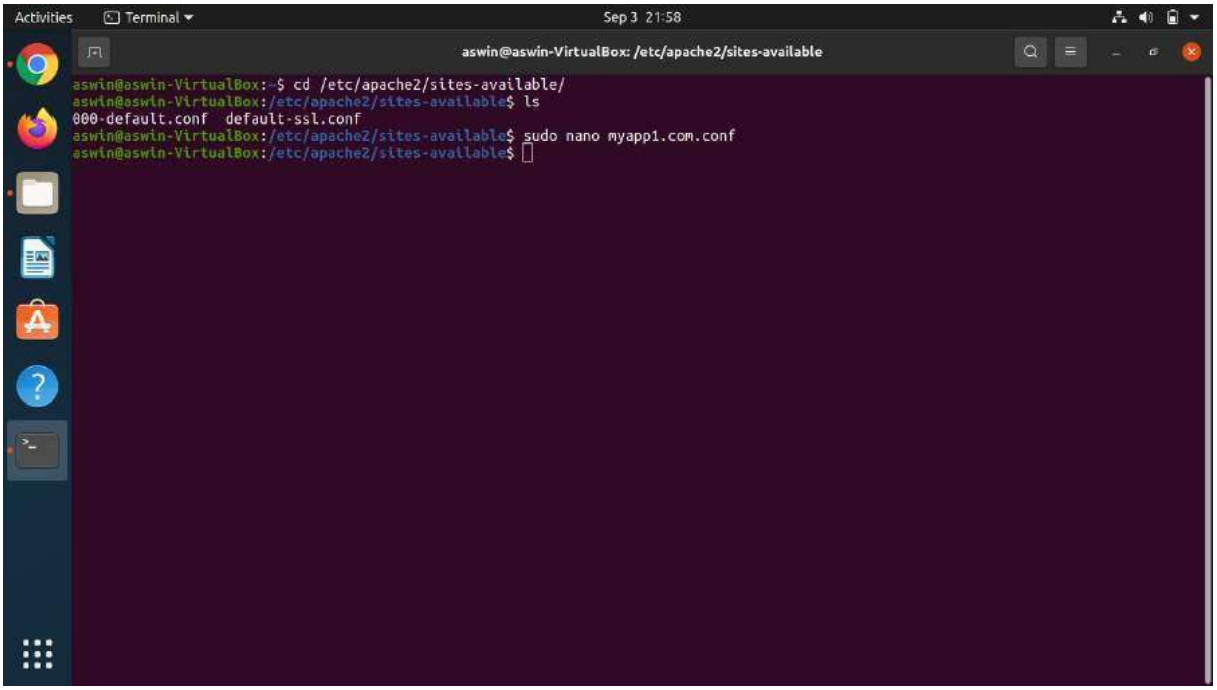


-R flag is recursive, Recursive means all subdirectory and files under your project directory become changed to the “www-data” group.

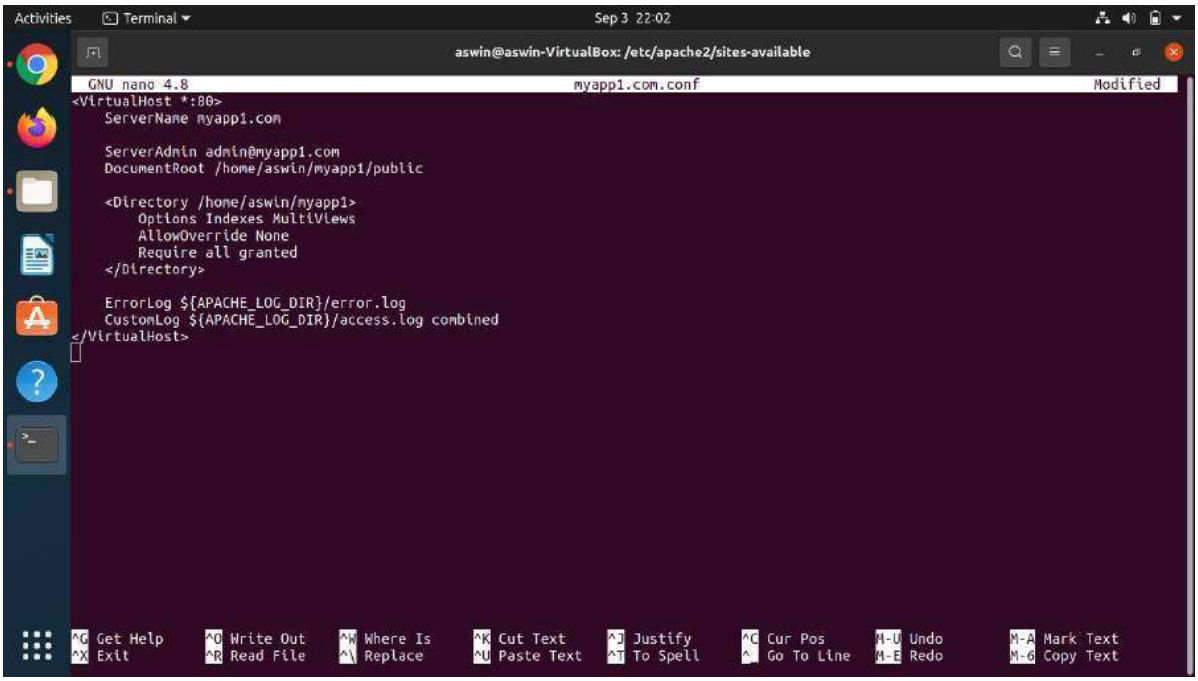
- Also, you need to change access permission 775 of the storage directory under your project. So, go ahead and use the following command.



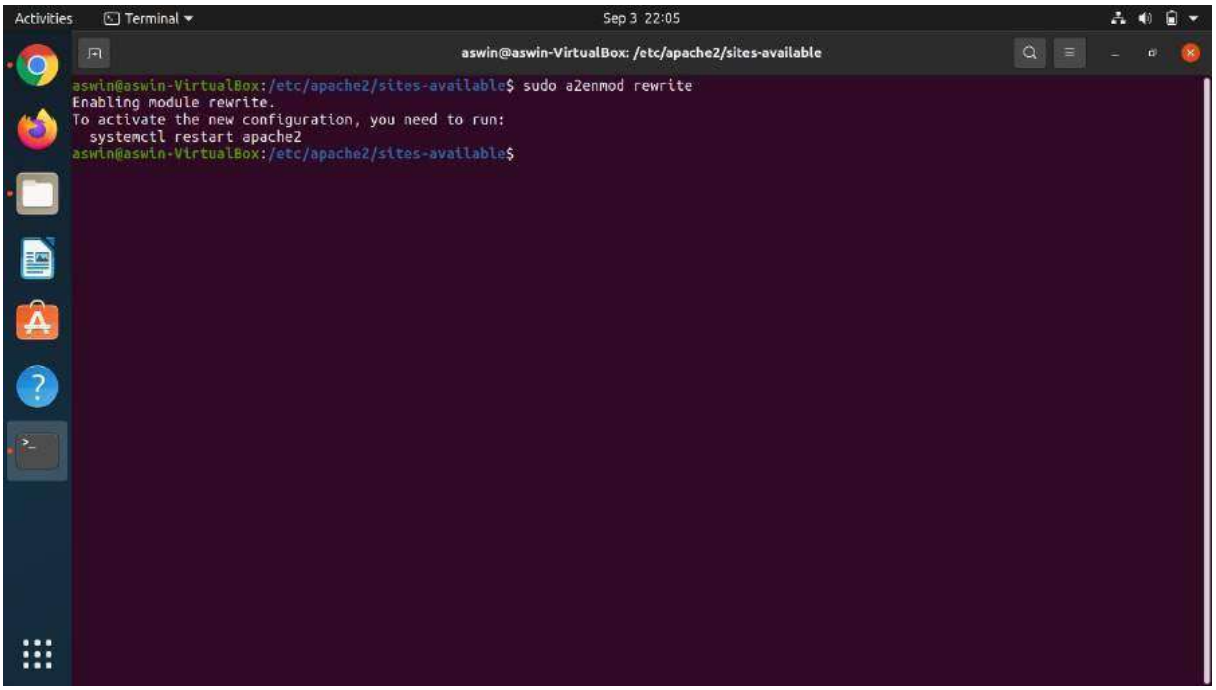
- Now create an apache vhost configuration go to the following directory and create a vhost config file using nano file editor.



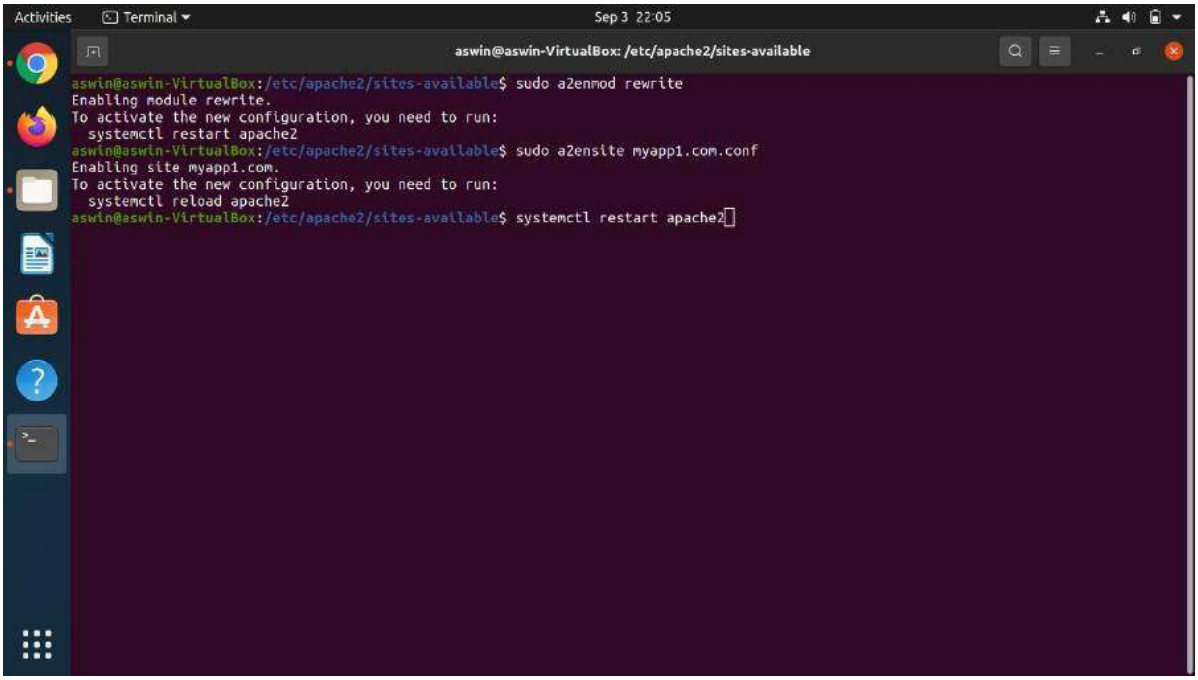
- And type the following line inside the file.



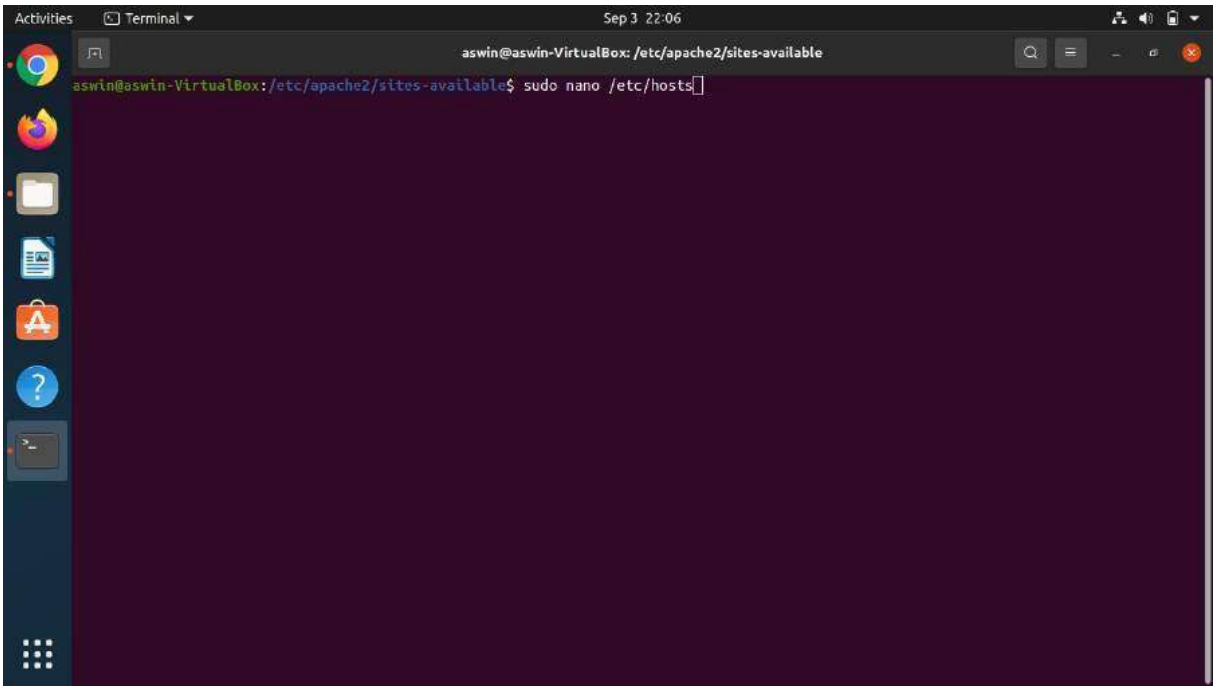
- Now enable mod rewrite for apache2 just type

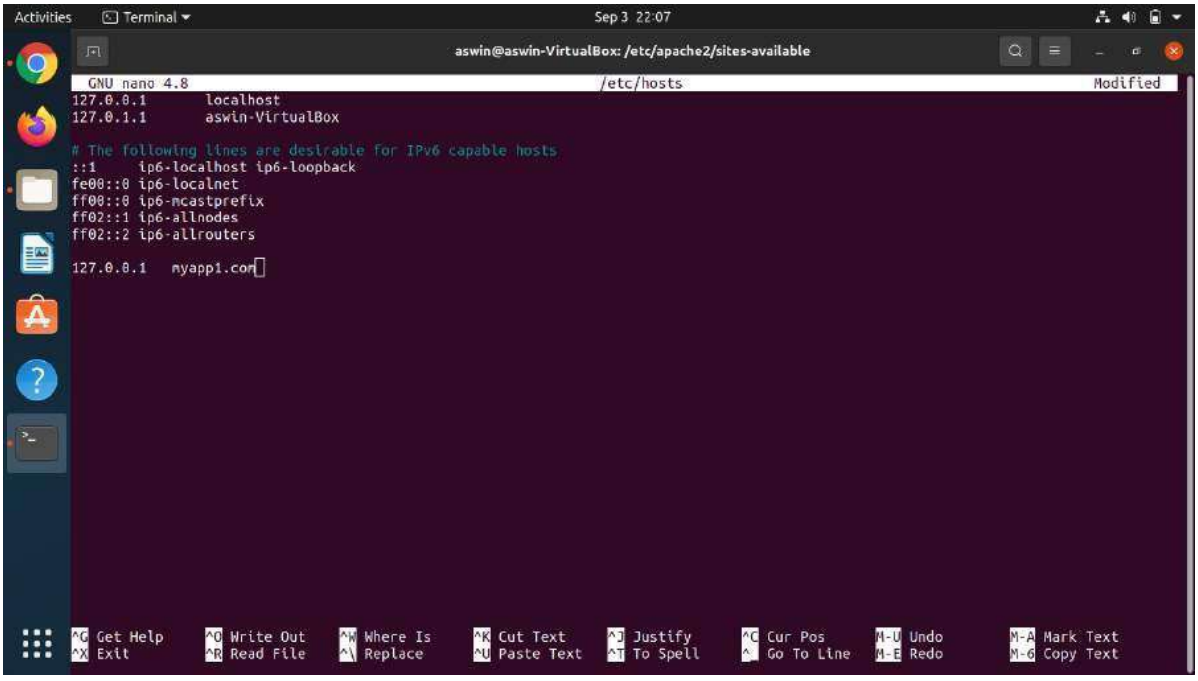


- Now enable your site, just type
  Finally, Restart the apache service, type

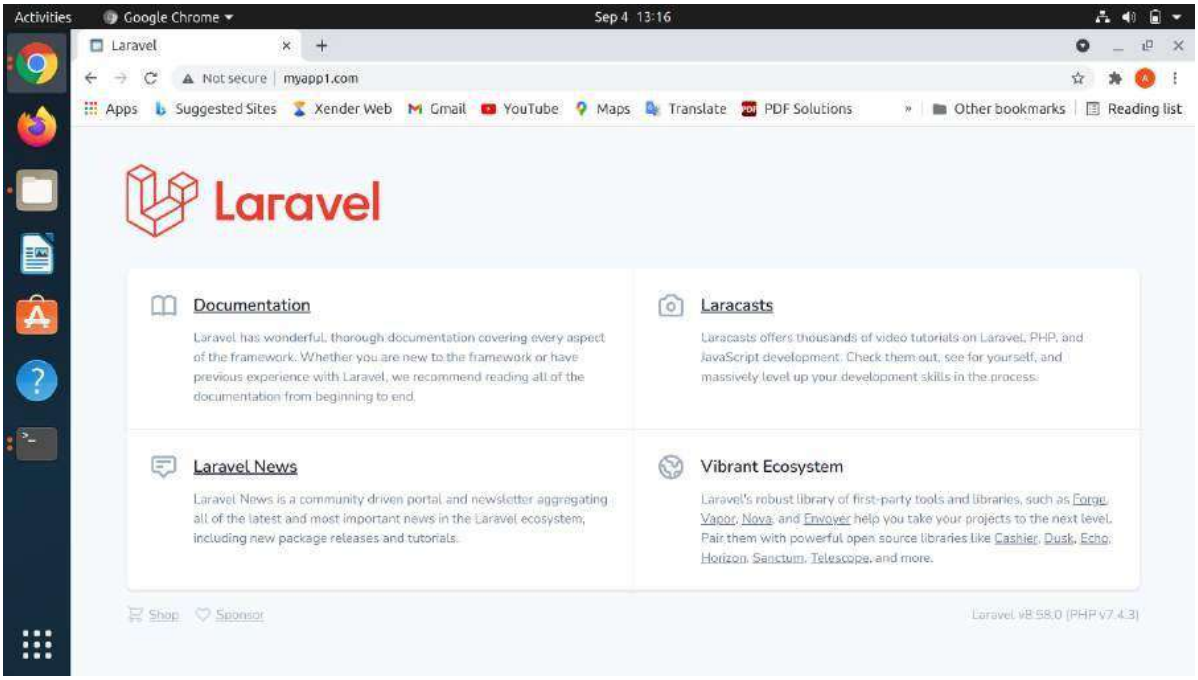


- As you are in a local environment you need a local dns resolver for your site. Go ahead and edit /etc/hosts file, add a dns record for your site then save the file.





- Now get back to the web browser and open a tab then type your project hostname.



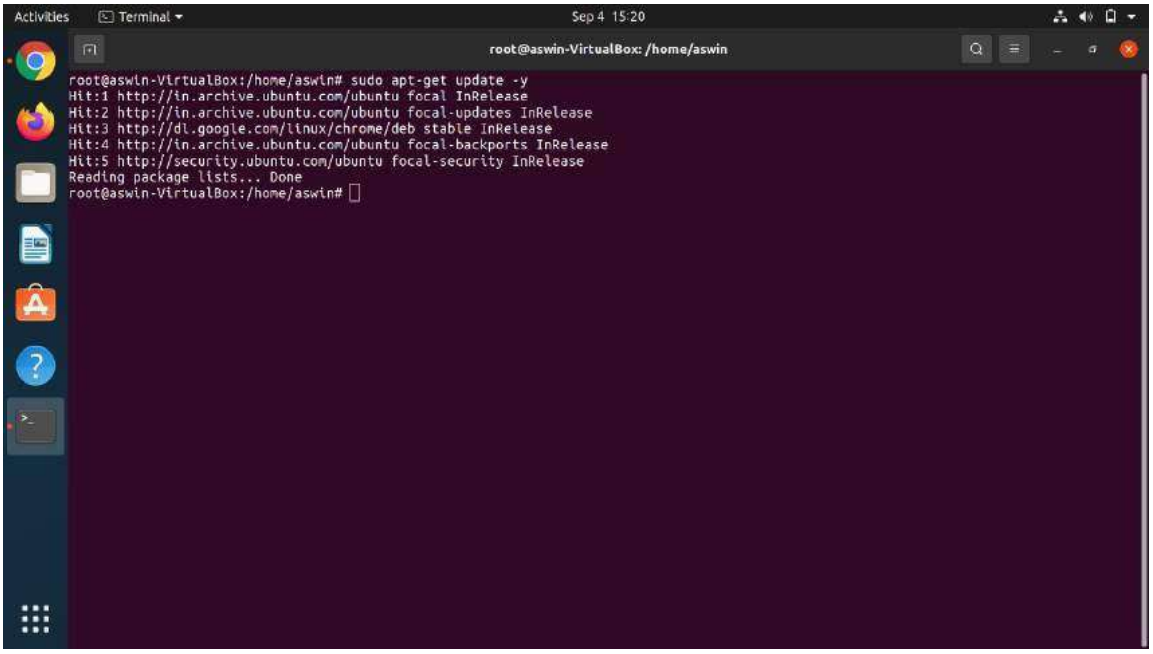
EXPIRIMENT-7:

Aim: Build and install software from source code, familiarity with make and cmake utilities expected

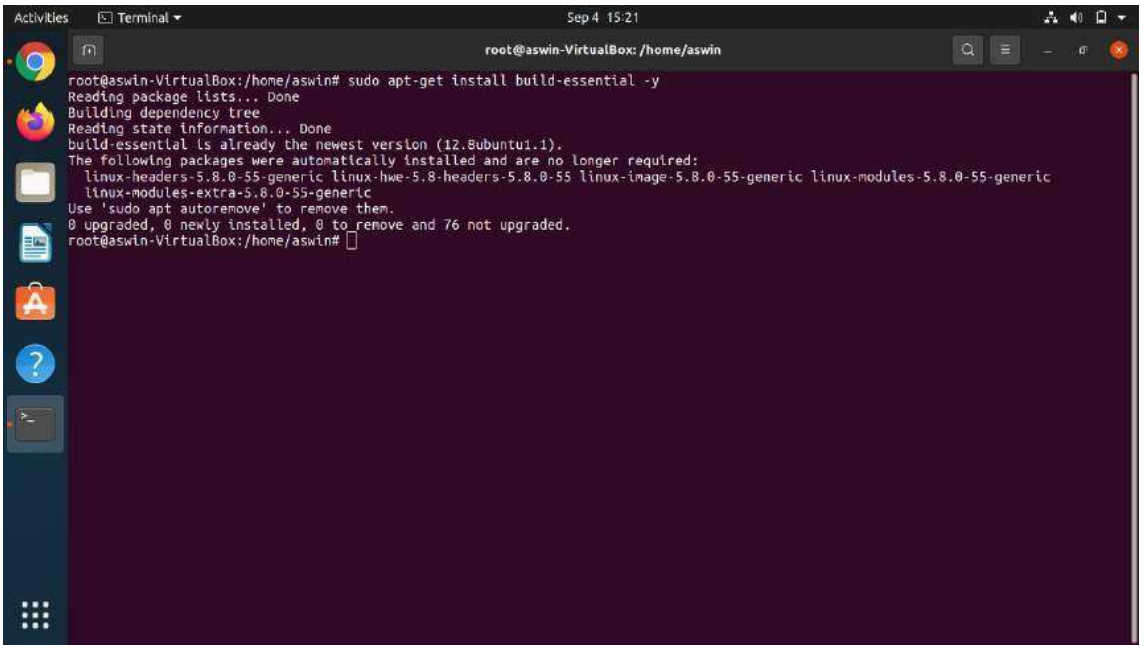
Solution :-

Step 1: Get the Server Ready

- As a best practice, make sure your packages are up to date:

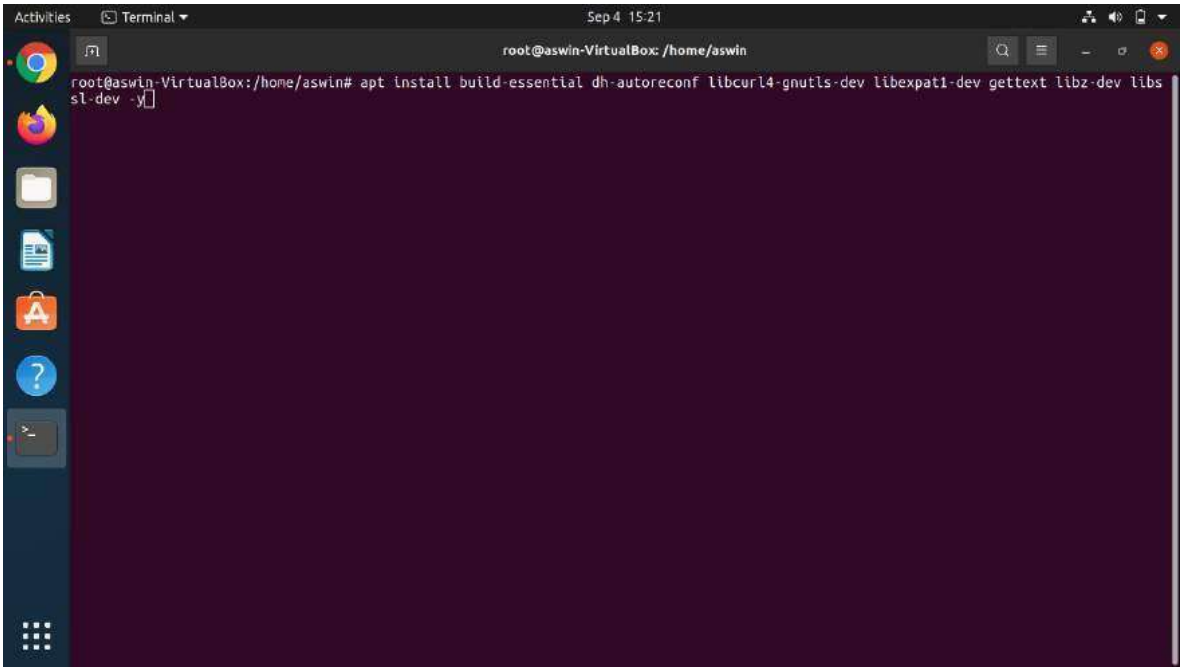


- Next, you'll need to make sure you have a compiler available. Run this command to install build-essential:



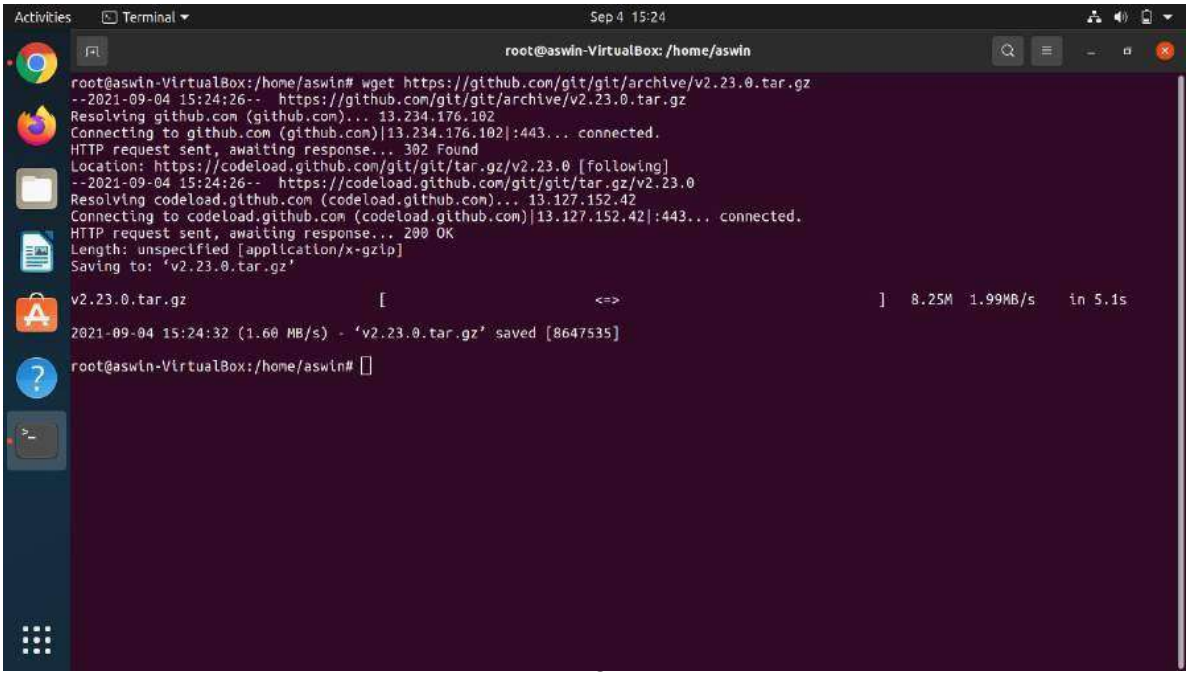
Step 2: Download Dependencies

- When installing a package from source code, you'll need to manage the installation of the package dependencies. We'll use apt-get to install git's dependencies:

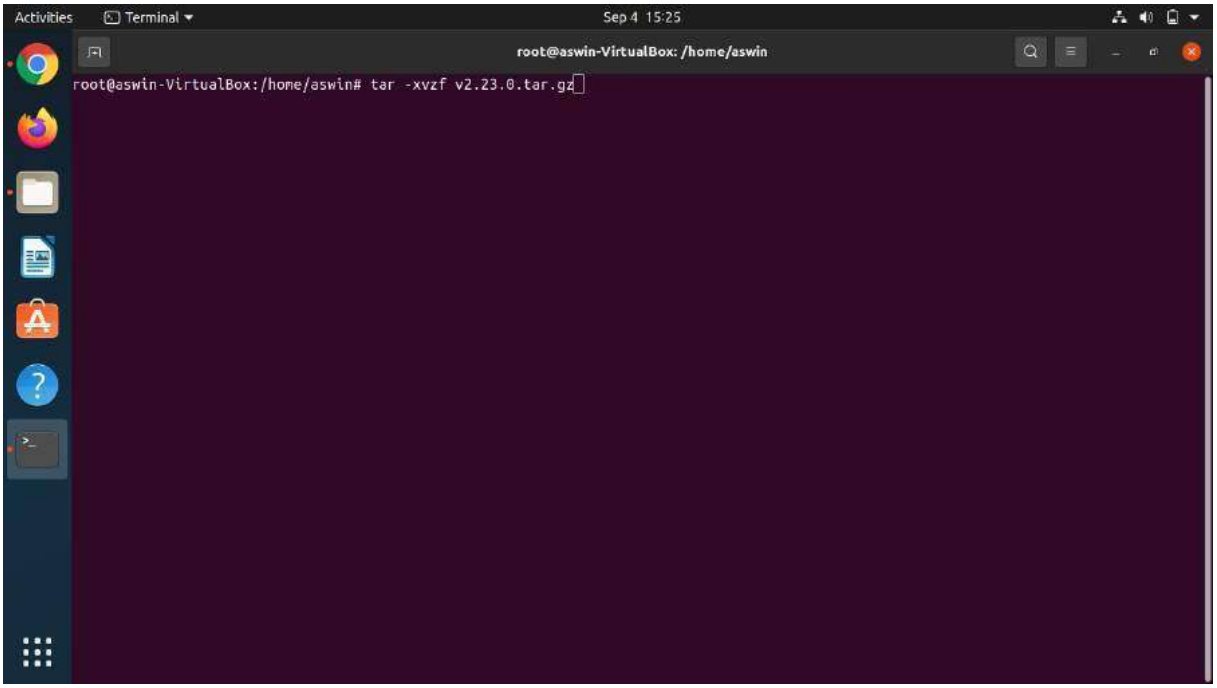


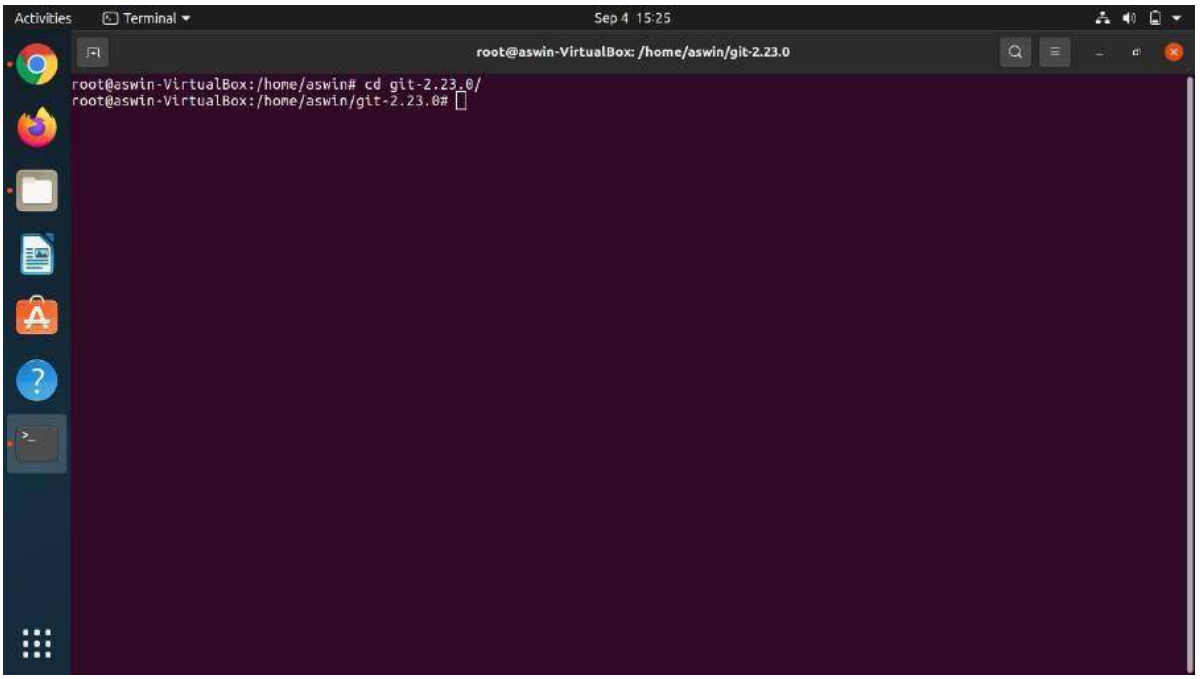
Step 3: Download the Source Package

- Once the package dependencies are in place, it's time to download the package with
  wget:



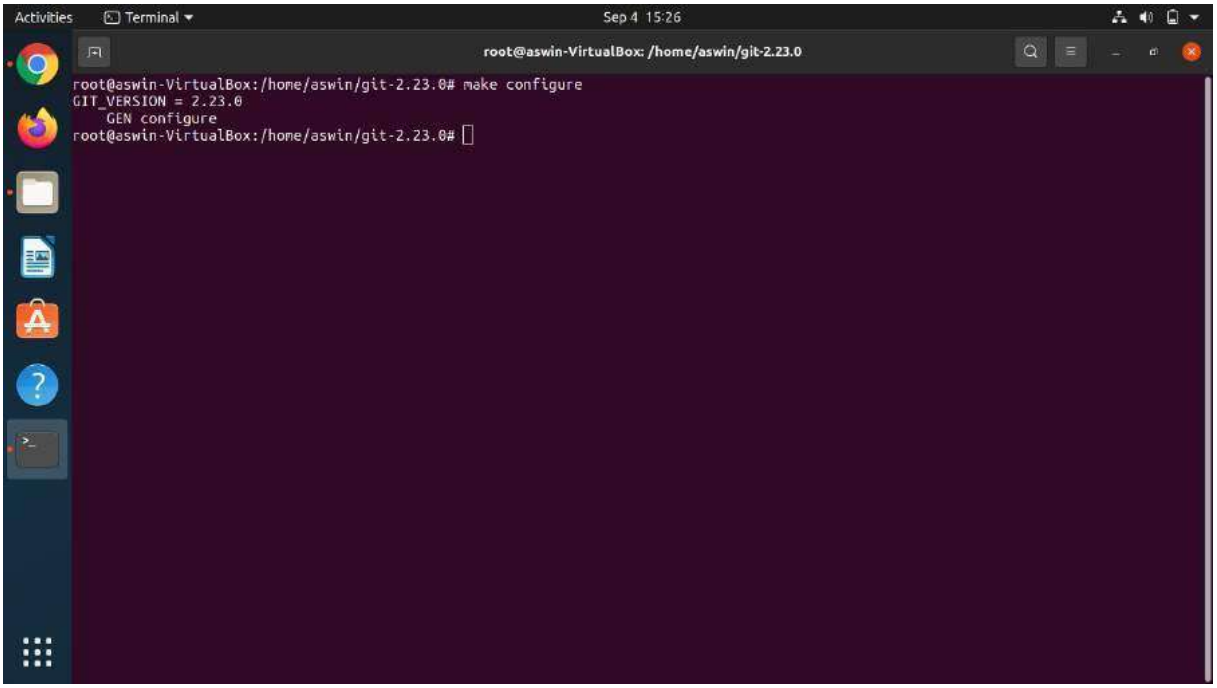
- Next, we need to extract the archive and cd (change directories) into the new git directory:



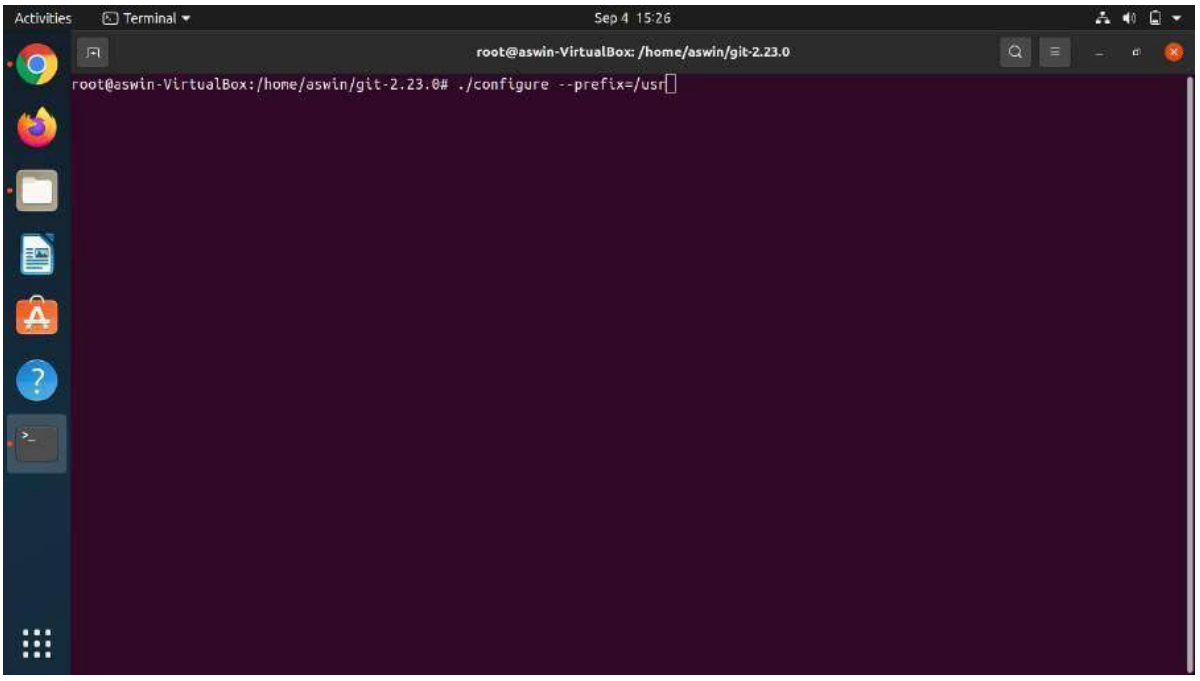


Step 4: Install Git

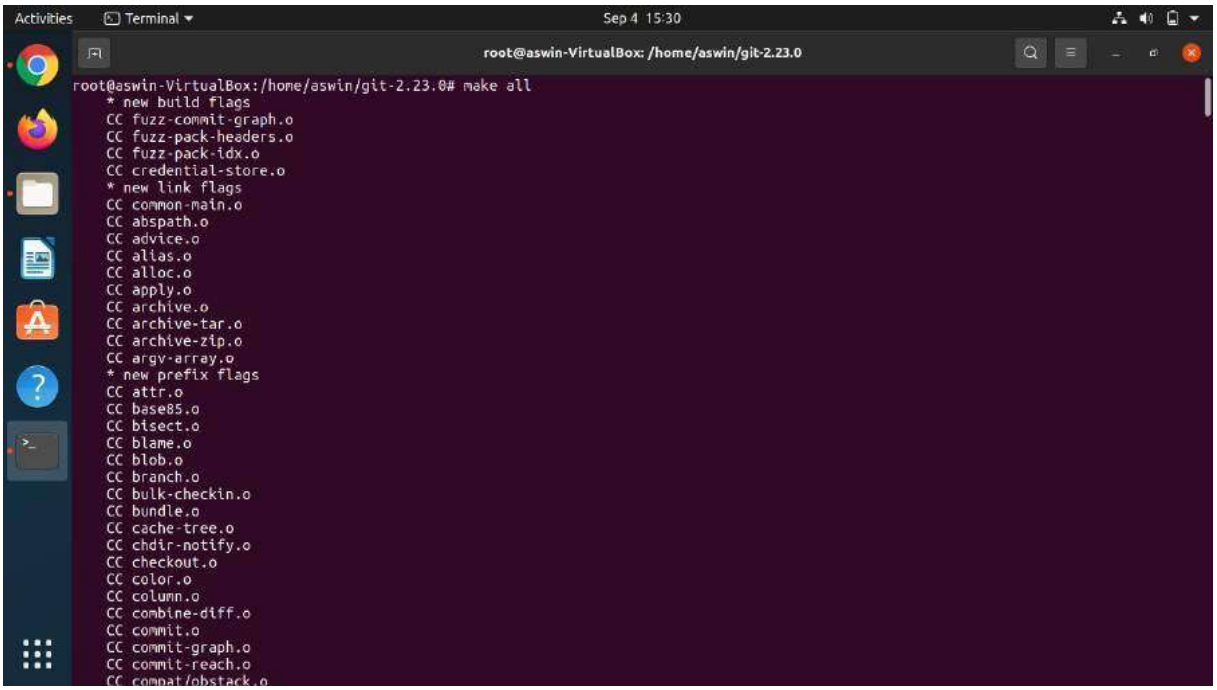
- Now that we have our package extracted and ready to go, we need to configure it:



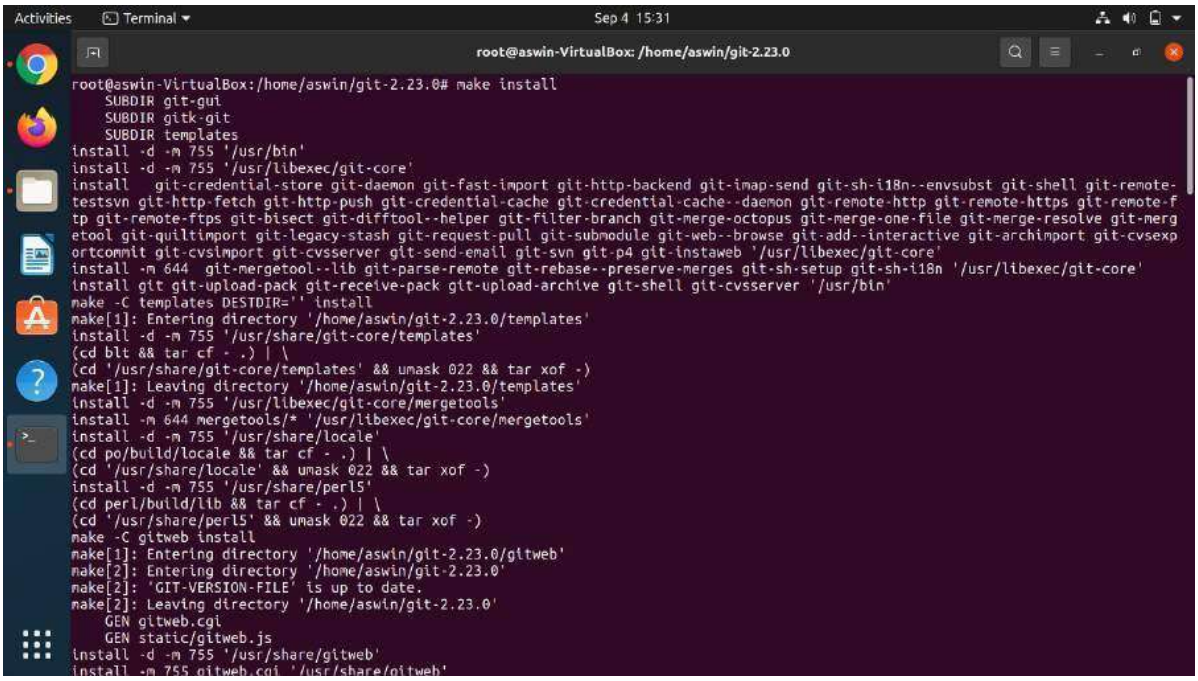
- Next, let's verify that all of the dependencies necessary to build the package are available by running this command:



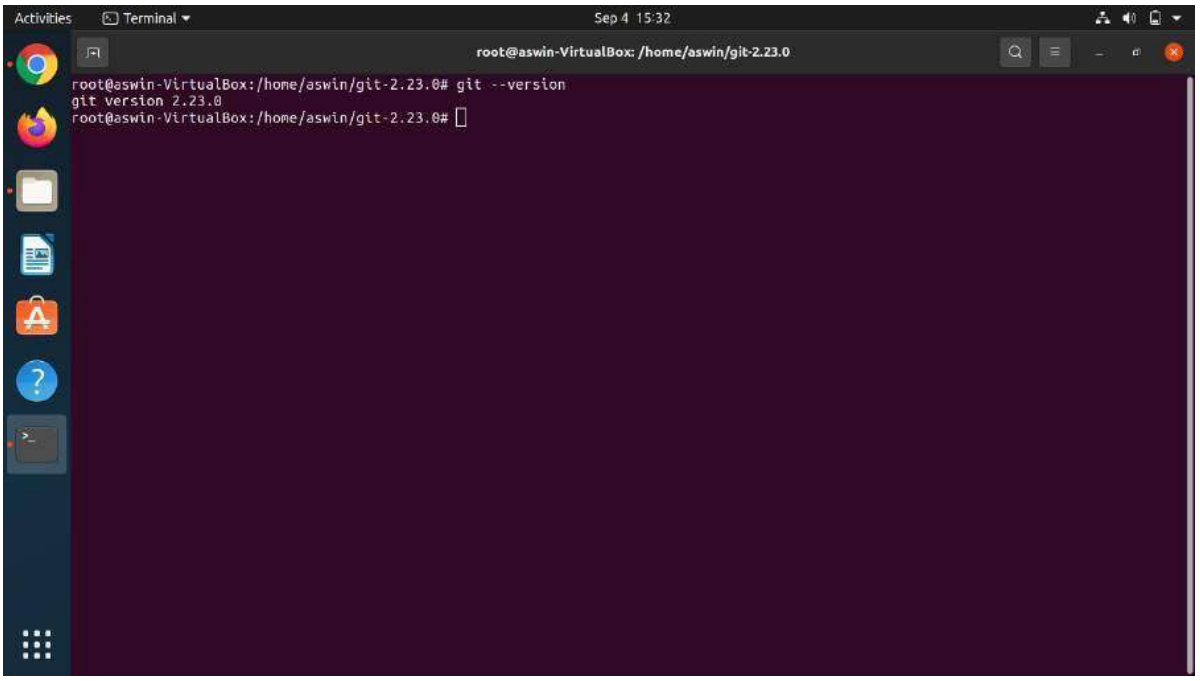
- After that, we'll build the source code:



- Now that the binaries are all built, its time to install git:



- That’s it! The last thing to do is to verify that git is working:



EXPIRIMENT 8:

Aim: Introduction to command line tools for networking IPv4 networking, network commands: ping route traceroute, nslookup, ip. Setting up static and dynamic IP addresses. Concept of Subnets, CIDR address schemes, Subnet masks, iptables, setting up a firewall for LAN, Application layer (L7) proxies.

Solution :-

The network infrastructure is a very complex structure of cables, routers, access points, data packets and a million other small components that together make the entire network work seamlessly. Any issue in any of these smaller components may lead to an overall collapse of the network infrastructure. This may lead to disruption of WiFi, cellular and wired (ethernet) infrastructure. This is the reason why it is very important to have an access to how the network is performing and know troubleshooting techniques.

The operating system acts as an intermediate platform between the user and the underlying network infrastructure. To use the below commands in Windows operating system, one needs to click on Start, go to Run and type cmd. This will open up the command prompt. In Mac OS, you can use the terminal application.

Ipv4 Networking:

The operating system consists of various built-in, command-line networking utilities that are used for network troubleshooting.

IP is part of an internet protocol suite, which also includes the transmission control protocol. Together, these two are known as TCP/IP. The internet protocol suite governs rules for packetizing, addressing, transmitting, routing, and receiving data over networks.

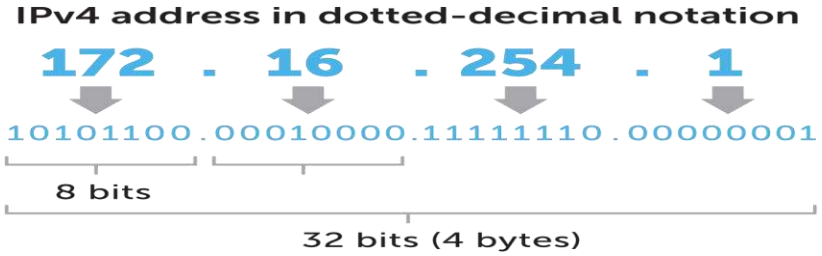
IP addressing is a logical means of assigning addresses to devices on a network. Each device connected to the internet requires a unique IP address.

Most networks that handle internet traffic are packet-switched. Small units of data, called packets, are routed through a network. A source host, like your computer, delivers these IP packets to a destination host, such as a server, based on IP addresses in packet headers. Packet-switching allows many users on a network to share the same data path.

An IP address has two parts—-one part identifies the host, such as a computer or other device. And the other part identifies the network it belongs to. TCP/IP uses a subnet mask to separate them.

IP (version 4) addresses are 32-bit integers that can be expressed in hexadecimal notation. The more common format, known as dotted quad or dotted decimal, is x.x.x.x, where each x can be any value between 0 and 255. For example, 192.0.2.146 is a valid IPv4 address.

IPv4 still routes most of today's internet traffic. A 32-bit address space limits the number of unique hosts to 232, which is nearly 4.3 billion IPv4 addresses for the world to use (4,294,967,296, to be exact).

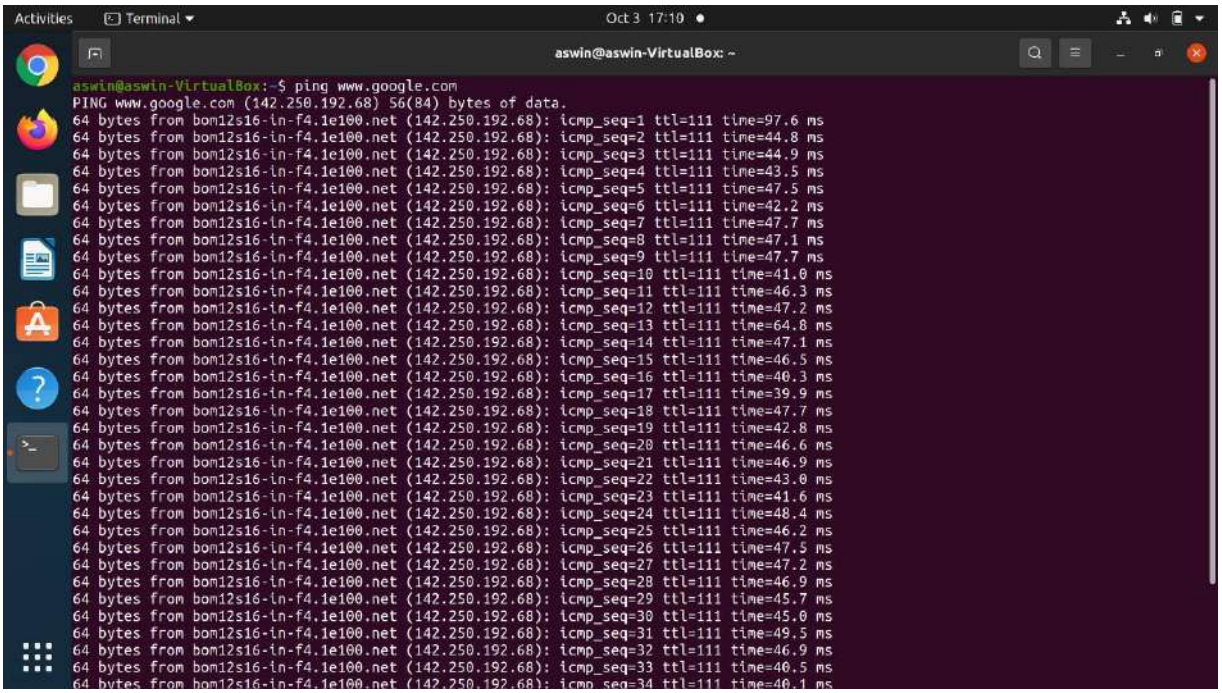


Network Commands:

ping: Ping command is typically used for checking the network connectivity from your system to an end device like a server or a printer and also of a website. This command is used while troubleshooting the entire network. So, when you enter a URL in your web browser, what you are actually doing is instructing your machine to connect to the website name. The website name is actually an alias for the IP address. So this command can be used in two ways:

1. It can be used to ping a network IP address.

2. It can used to ping a website or hostname directly.



Route: Using the route command displays or modifies the computer's routing table. For a typical computer that has a single network interface and is connected to a local area network (LAN) that has a router, the routing table is pretty simple and isn't often the source of network problems. Still, if you're having trouble accessing other computers or other networks, you can use the route command to make sure that a bad entry in the computer's routing table isn't the culprit.

For a computer with more than one interface and that's configured to work as a router, the routing table is often a major source of trouble. Setting up the routing table properly is a key part of configuring a router to work.

Syntax:

route [-f] [-p] [command [destination] [mask subnetmask] [gateway] [metric costmetric]]

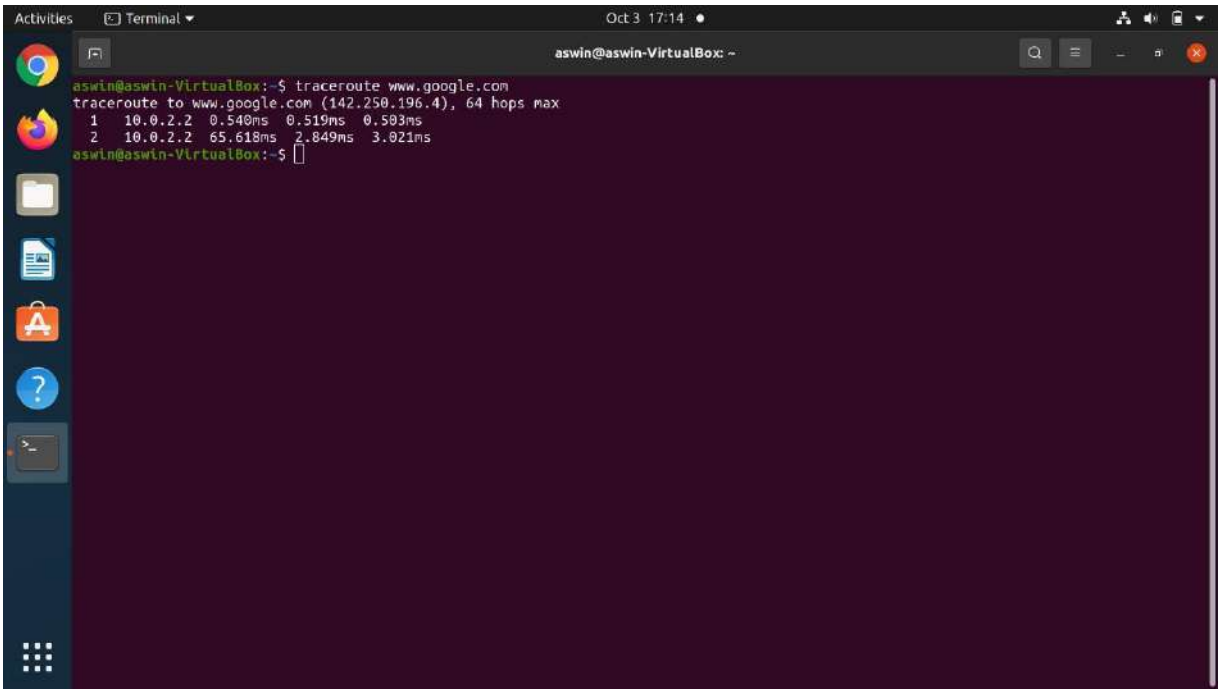
This section explains each of the options that you can use with the route command.

- ✓ The -f option clears the routing tables of all gateway entries. If you use the -f option in conjunction with one of the commands, the tables are cleared before you run the command.

- ✓ By default, routes are not preserved when you restart the system. Use the -p option with the add command to make a route persistent. Use the -p option with the print command to view the list of registered persistent routes.

Traceroute

traceroute command in Linux prints the route that a packet takes to reach the host. This command is useful when you want to know about the route and about all the hops that a packet takes.



The first column corresponds to the hop count. The second column represents the address of that hop and after that, you see three space-separated time in milliseconds. *traceroute* command sends three packets to the hop and each of the time refers to the time taken by the packet to reach the hop.

Syntax:

traceroute [options] host_Address [pathlength]

Options:

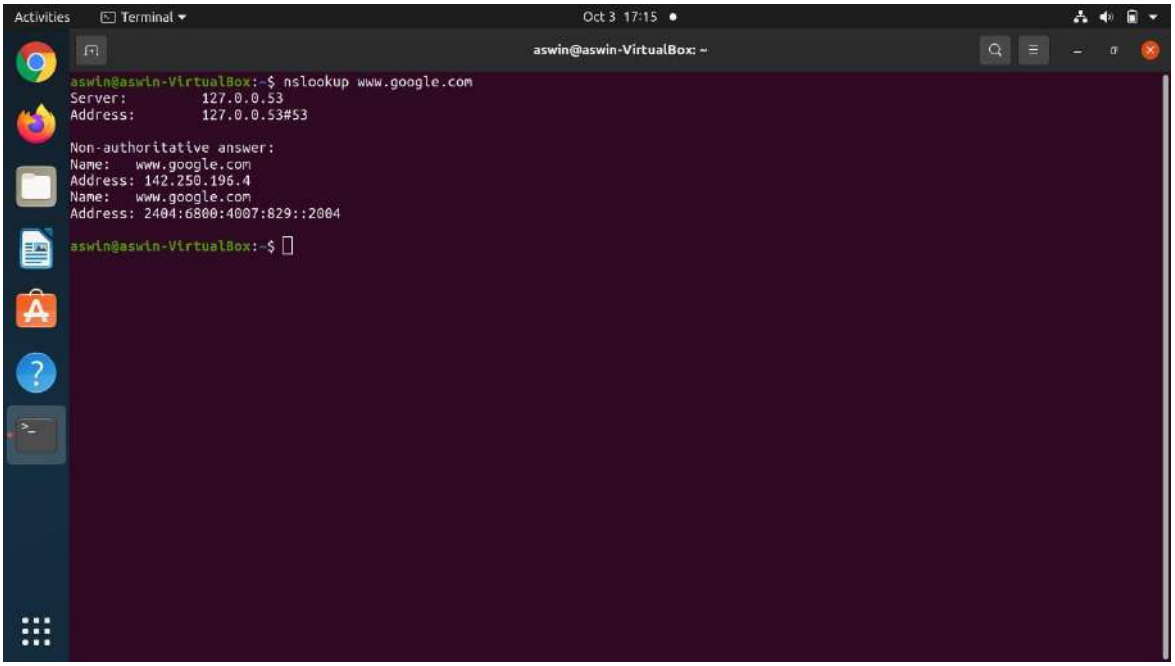
- -4 Option: Use ip version 4 i.e. use Ipv4
- -6 Option: Use ip version 6 i.e. use Ipv6
- -F Option: Do not fragment packet.

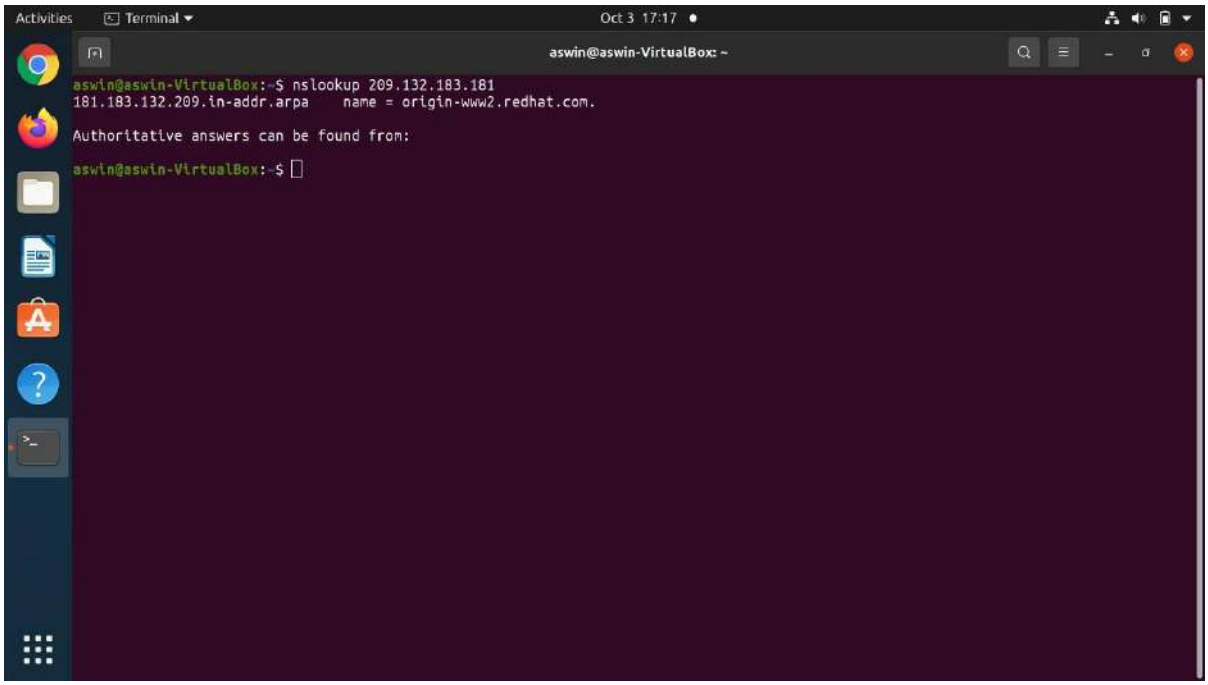
Nslookup:

nslookup(stands for "Name Server Lookup") is a useful command for getting information from DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS related problems.

Syntax:

nslookup [option]





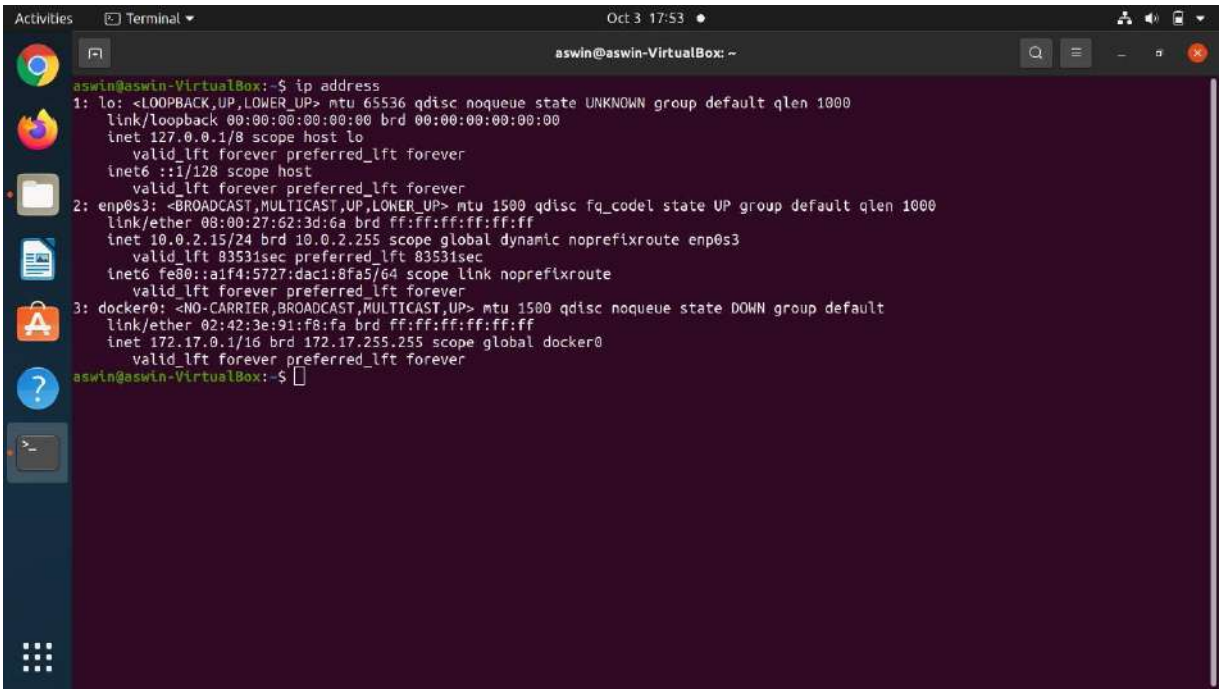
ip:

ipcommandinLinuxispresentinthenet-toolswhichisusedforperformingseveral network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It is similar to *ifconfig* command but it is much more powerful with more functions and facilities attached to it. *ifconfig* is one of the deprecated commands in the net-tools of Linux that has not been maintained for many years. ip command is used to perform several tasks like assigning an address to a network interface or configuring network interface parameters.

It can perform several other tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and setting up IP addresses and routes.

Syntax:

ip [OPTIONS] OBJECT { COMMAND | help }



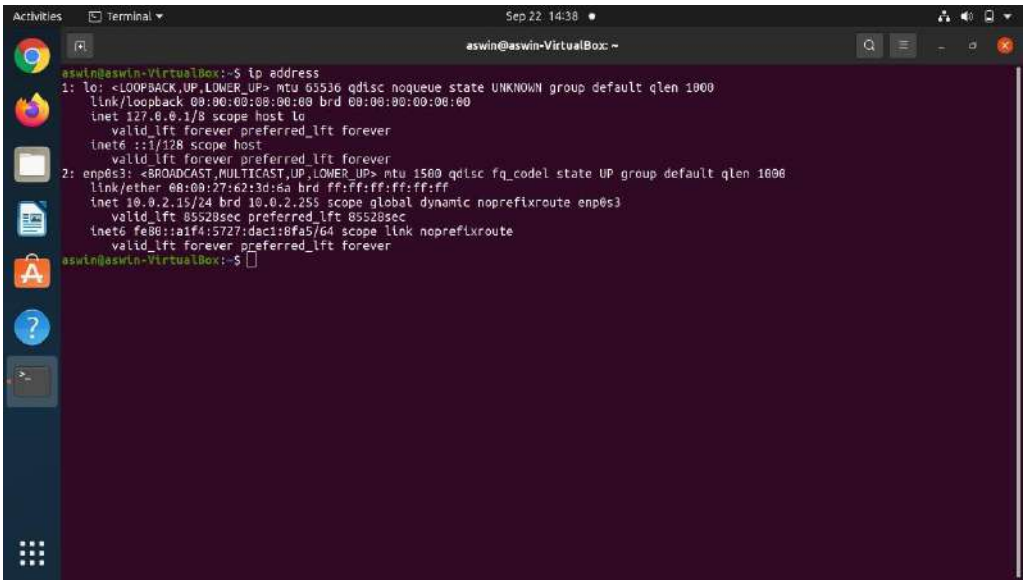
This will show the information related to all interfaces available on our system, but if we want to view the information of any particular interface, add the options show followed by the name of the particular network interface.

Options:

- -address: This option is used to show all IP addresses associated on all network devices.
- -link: It is used to display link layer information, it will fetch characteristics of the link layer devices currently available. Any networking device which has a driver loaded can be classified as an available device.

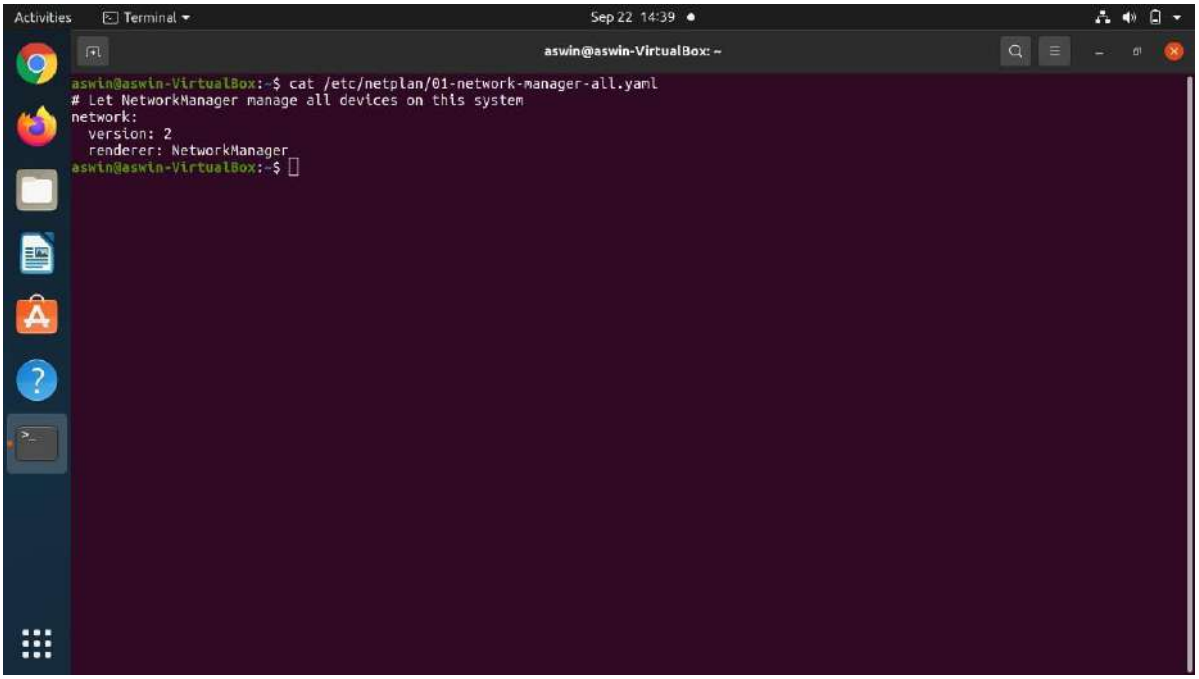
Setting up static IP addresses

Step1: List all the interfaces in the system. Use the ip address command to define a static IP address on an interface.

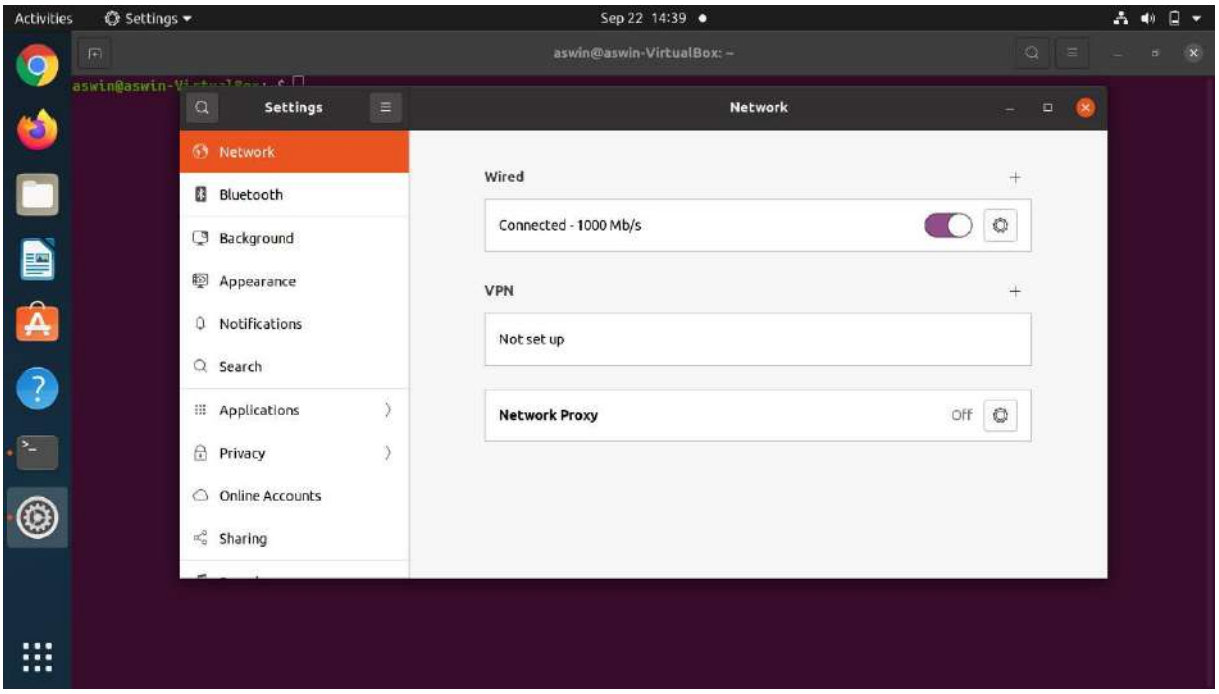


Step2:ToviewthecontentofNetplannetworkconfigurationfile,runthefollowing command:

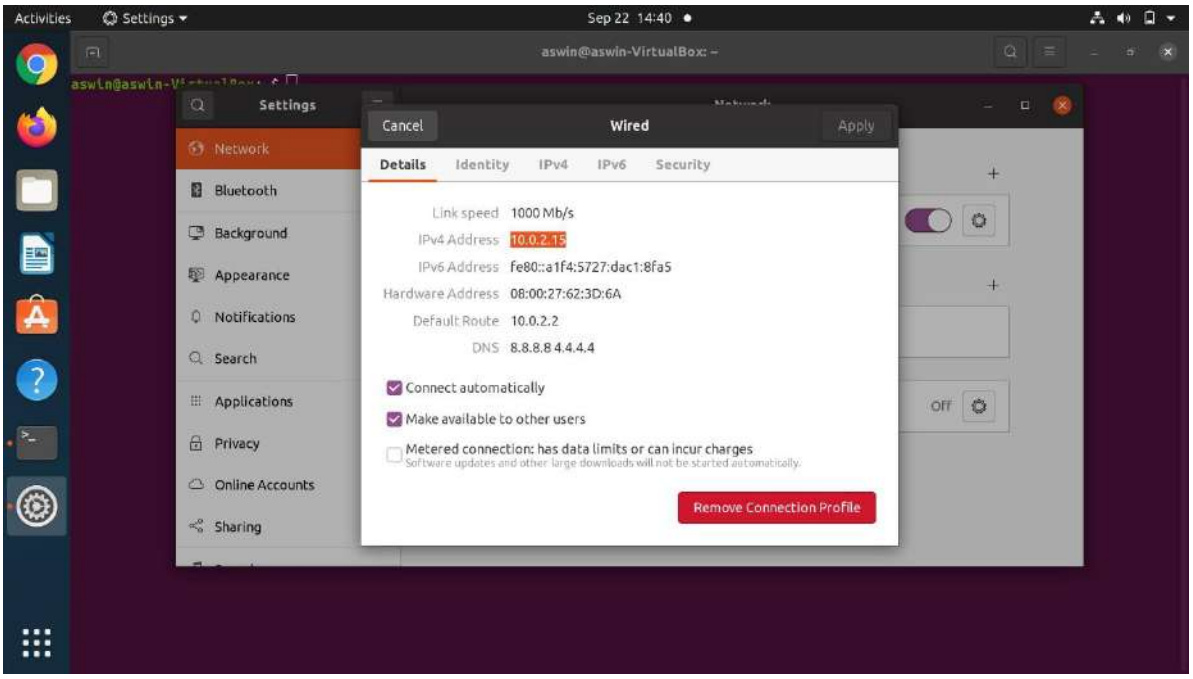
cat /etc/netplan/01-network-manager-all.yaml

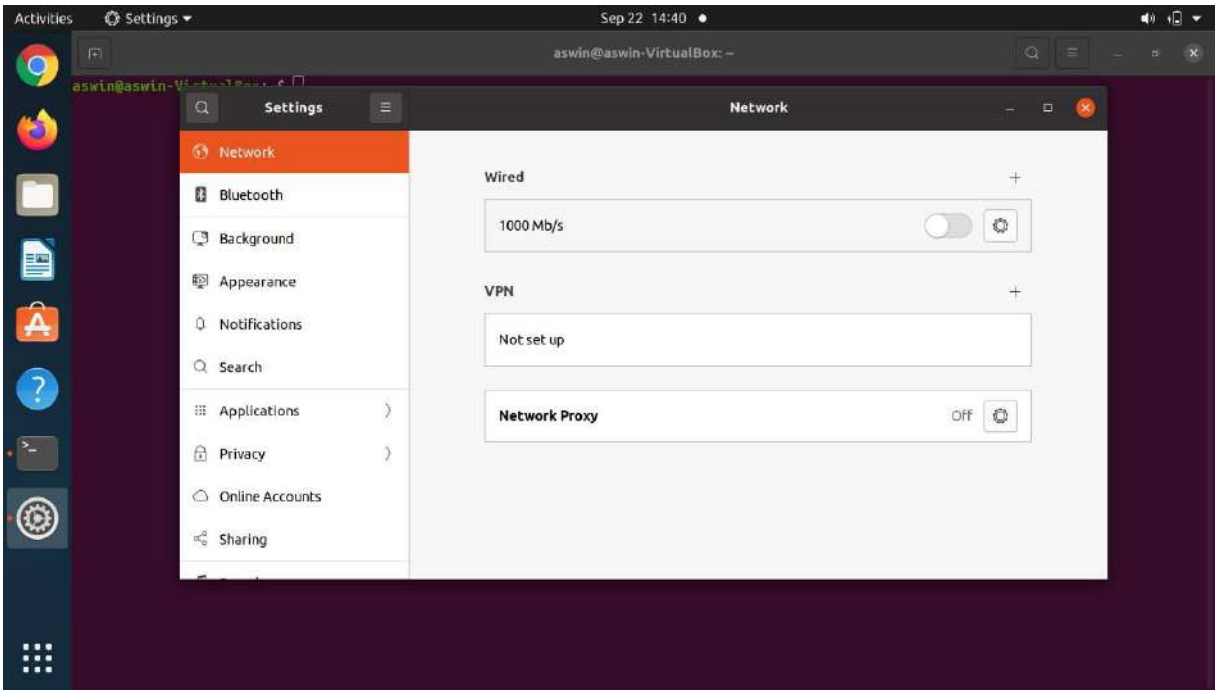


Step3:Clickonthetoprightnetworkiconandselectsettingsofthenetwork interfaceyouwish toconfiguretouseastaticIPaddressonUbuntu.

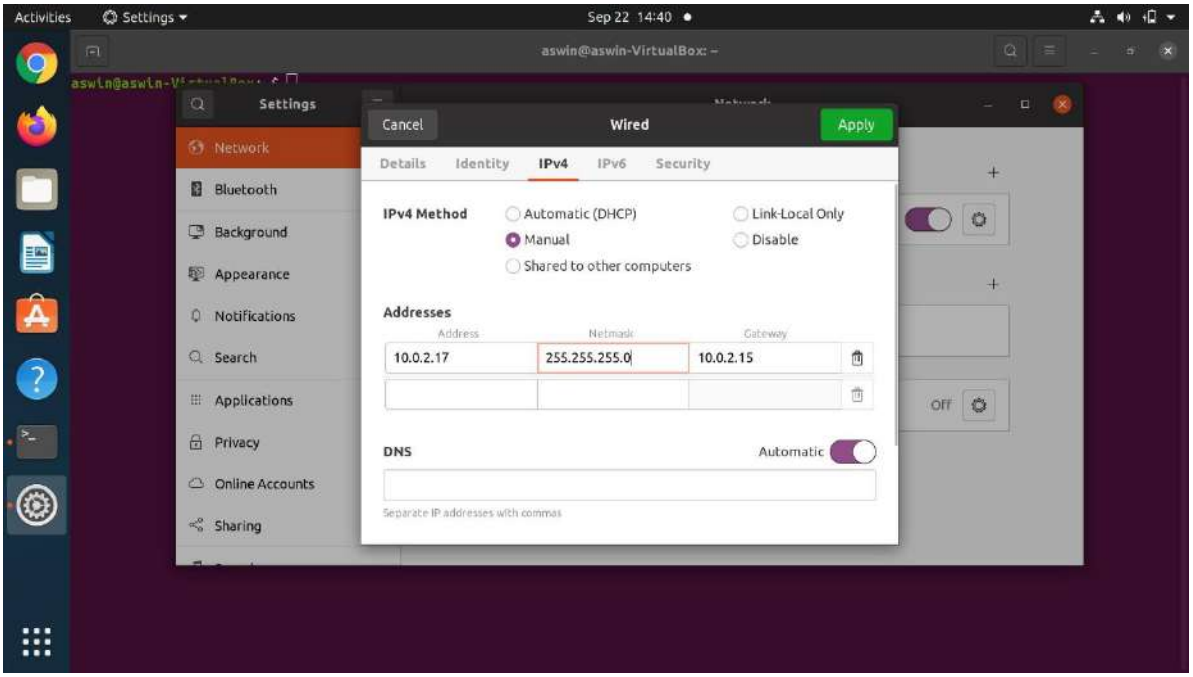


Click on the settings icon to start IP address configuration.

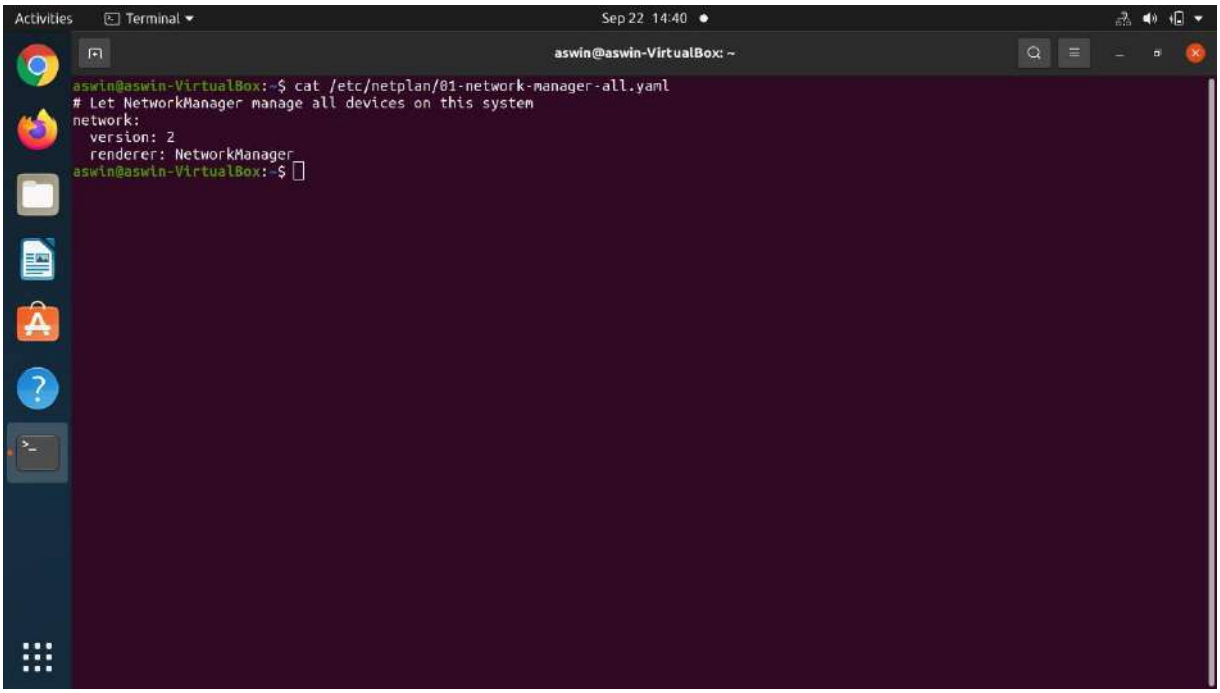


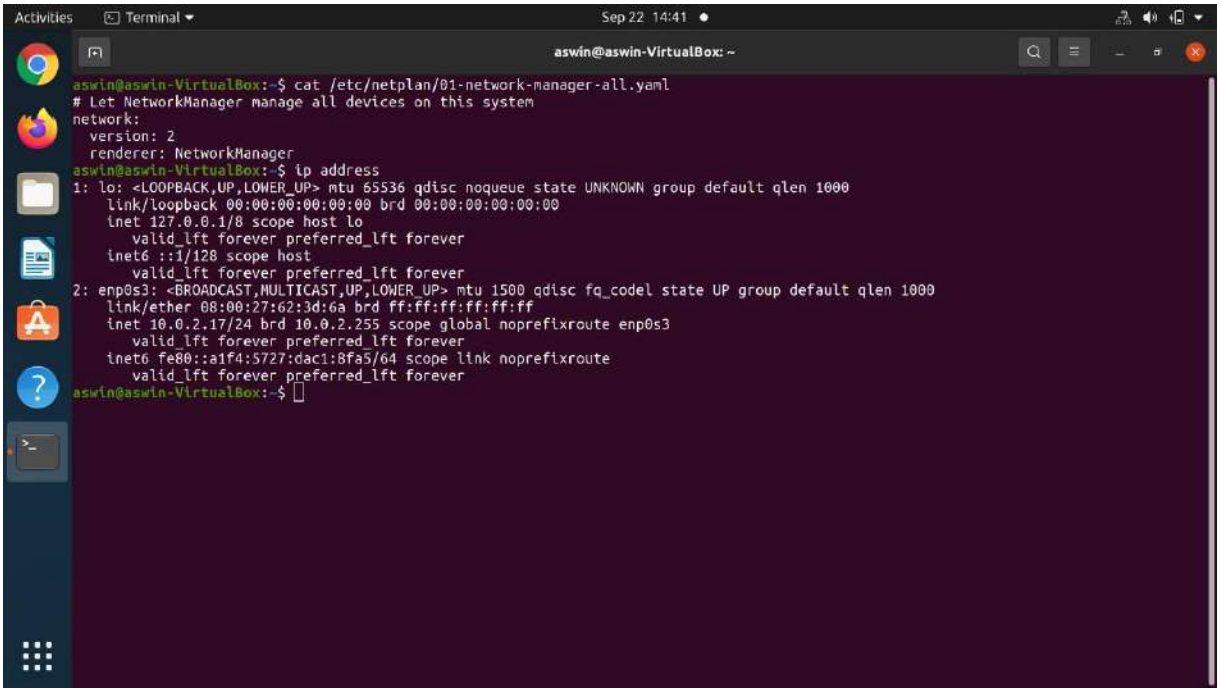


TurnOFFandONswitchtoapplyyournewnetworkstaticIPconfigurationsettings.



Step5:Runthecommandipaddressandclickonthenetworksettingsicononce againto confirm your new static IP address settings.





Configure and Set Up a Firewall on Ubuntu

UFWstandsforUncomplicatedFirewallwhichactsasaninterfacetoIPTABLESthat simplifiesthe processoftheconfigurationoffirewallsitwillbeaveryhardfora

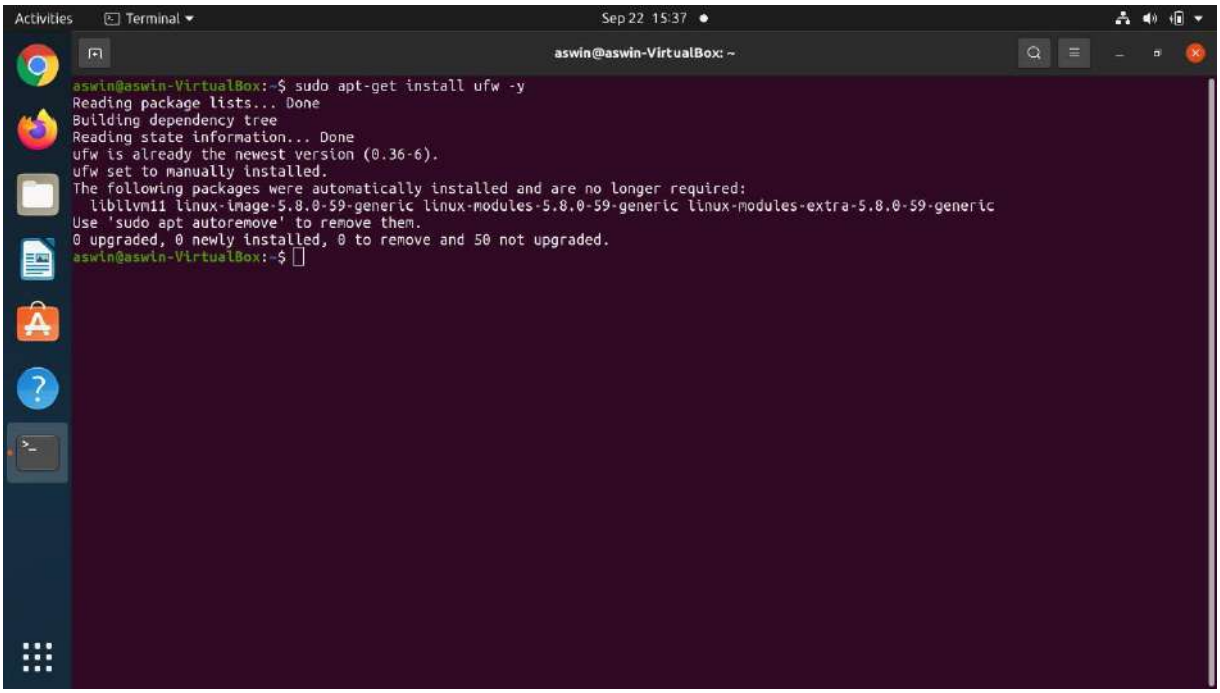
beginnertolearnsandconfigurethefirewallruleswherewewillsecurethenetwork fromunknownusers aremachines.UFWworksonthepoliciesweconfigureasrules.

- For this, we needed a non-root user with root permission on the machine.

Installing the UFW (Firewall)

UFWisinstalledbydefaultwithUbuntu,ifnotinstalledthenwewillinstallthemusing the below command:

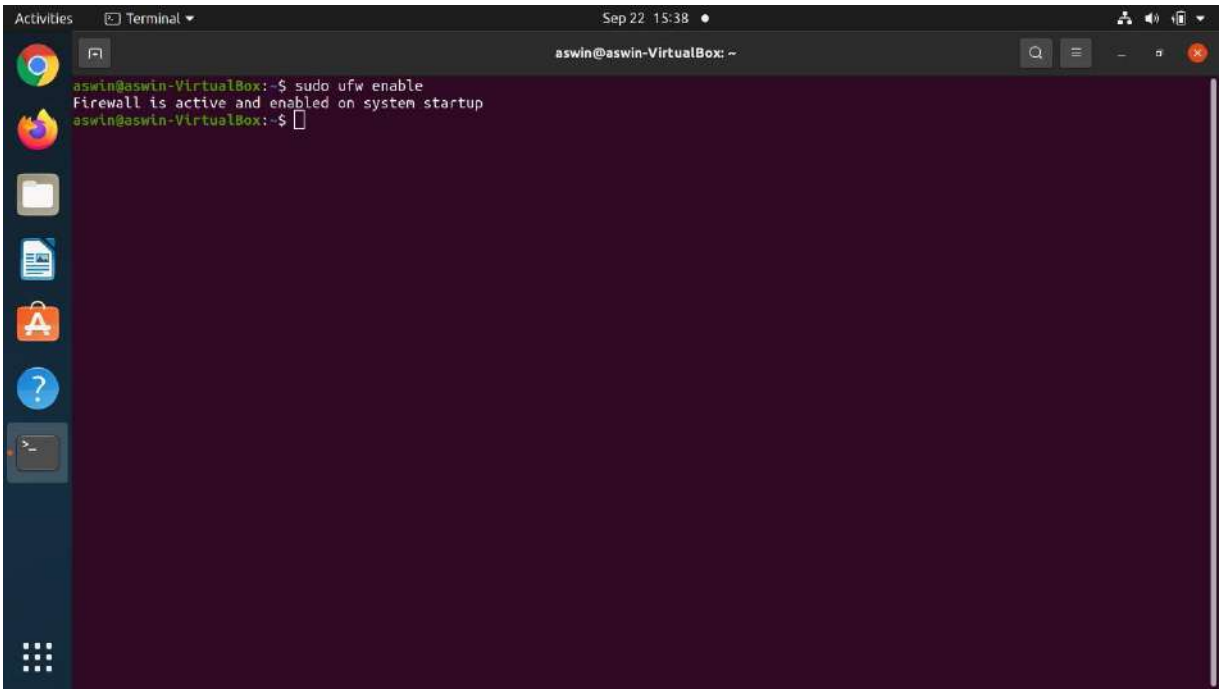
sudo apt-get install ufw –y



Enabling the UFW (Firewall)

Below is the command to enable the UFW –

sudo ufw enable

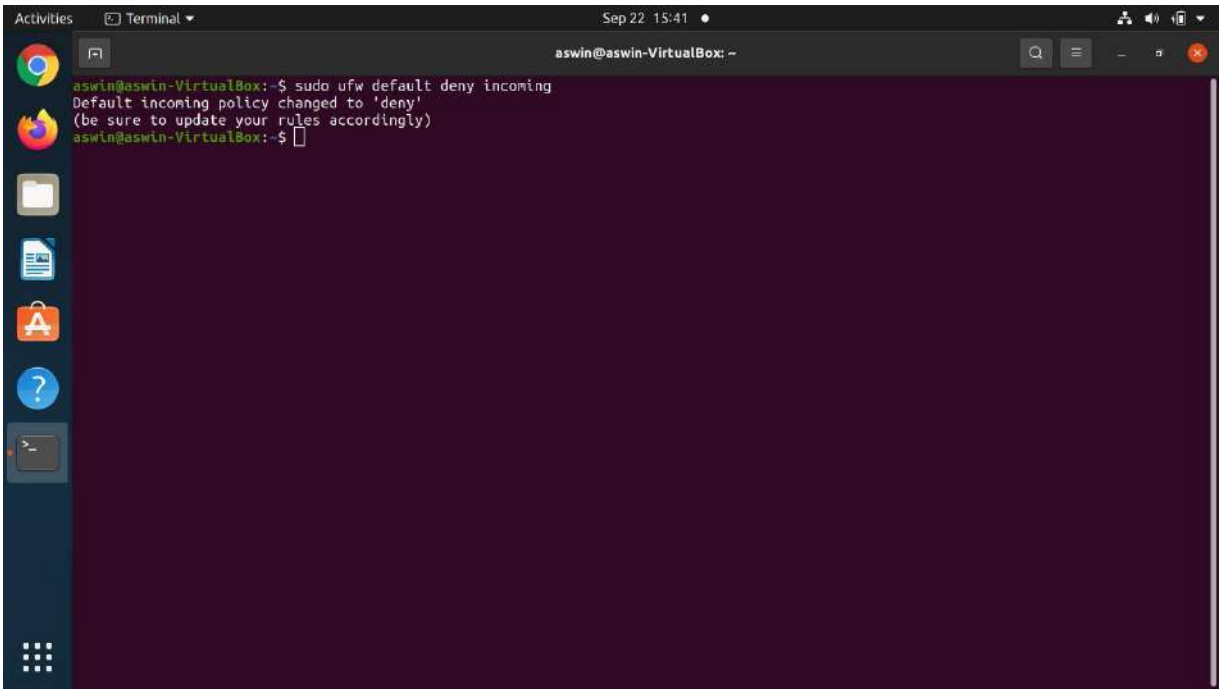


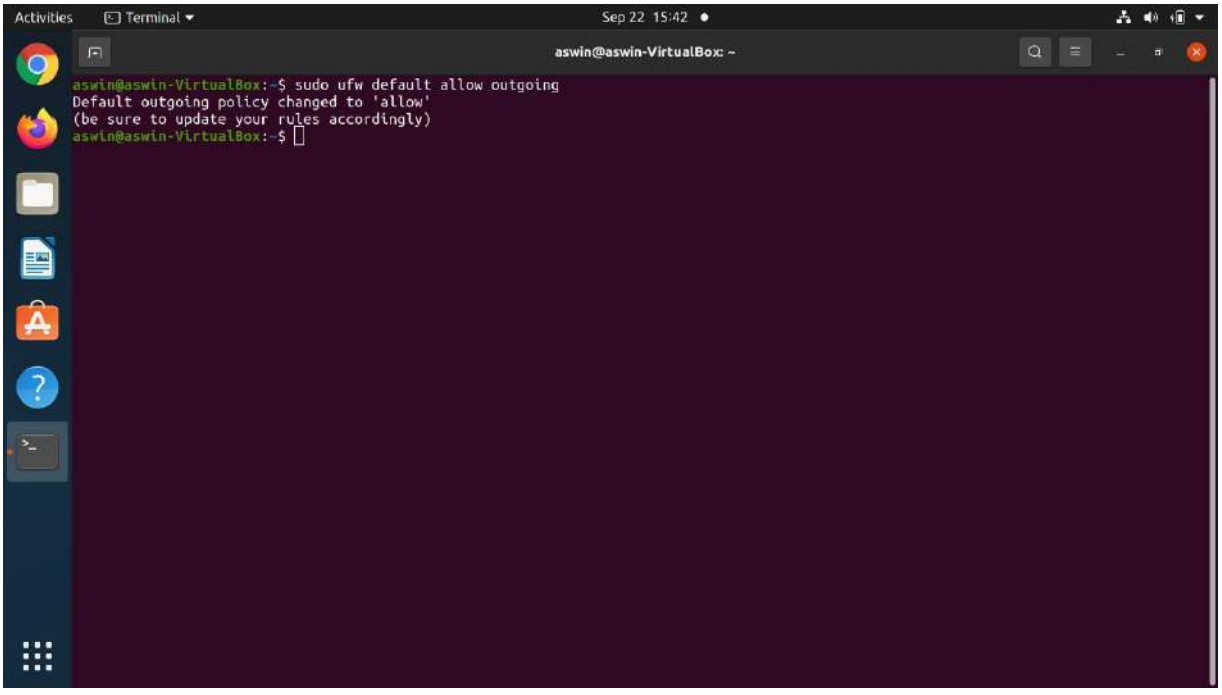
Enabling the Default Policies

As the beginner, we will first configure default policies, which control and handles the traffic which will not match the other rules. By default, the rules will deny all incoming connections and allow all outgoing connections will be allowed which stops someone trying to reach the machine from the internet world.

sudo ufw default deny incoming sudo

ufw default allow outgoing

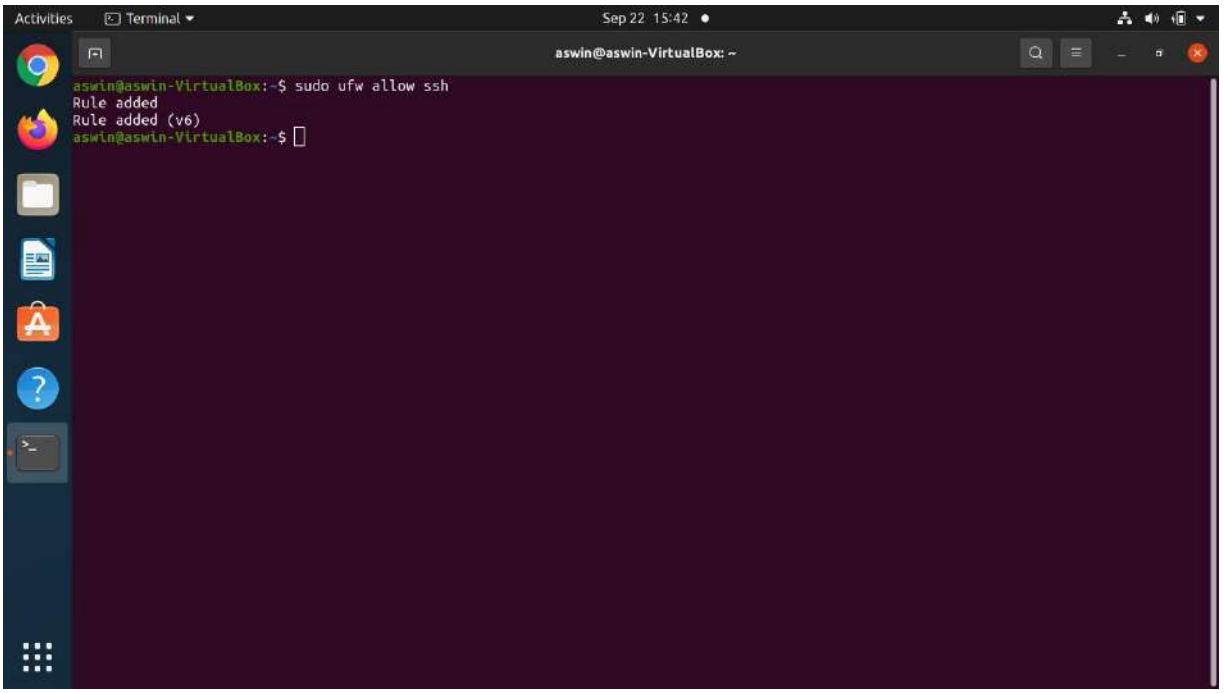




Enabling SSH Connections

Using the above commands, we have disabled all the incoming connections, it will deny all the incoming connections, we needed to create a rule which will explicitly allow the SSH incoming connection. Below is the command to enable the incoming connection for SSH.

sudo ufw allow ssh

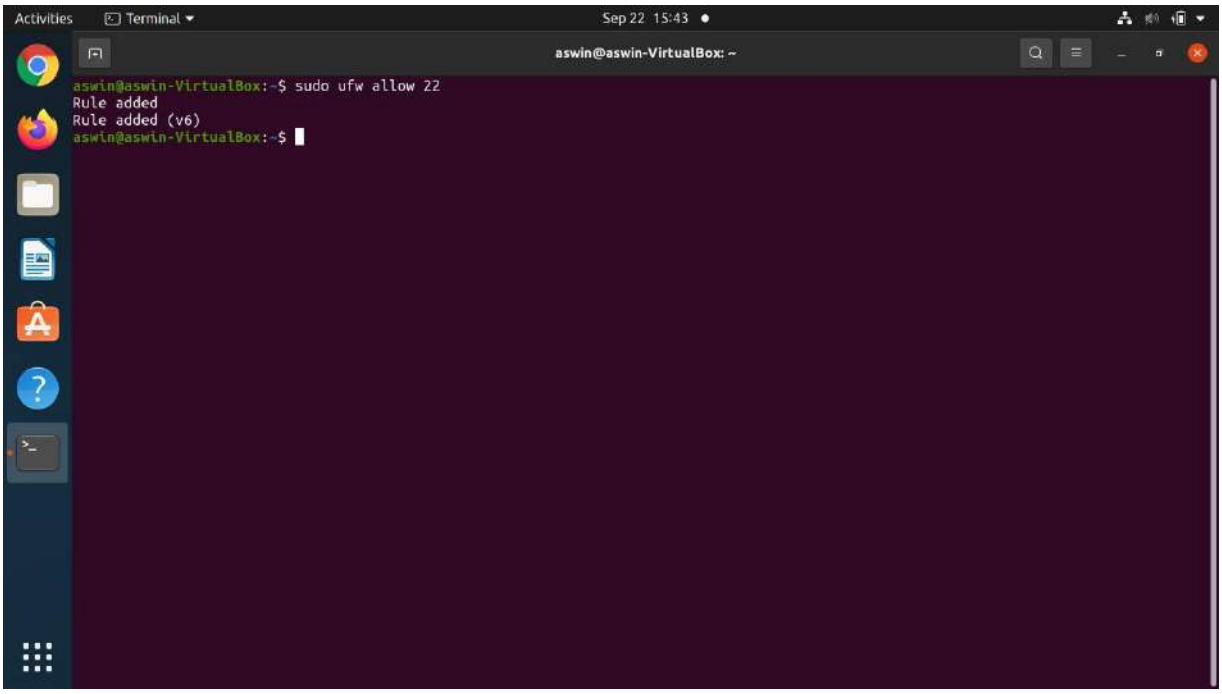


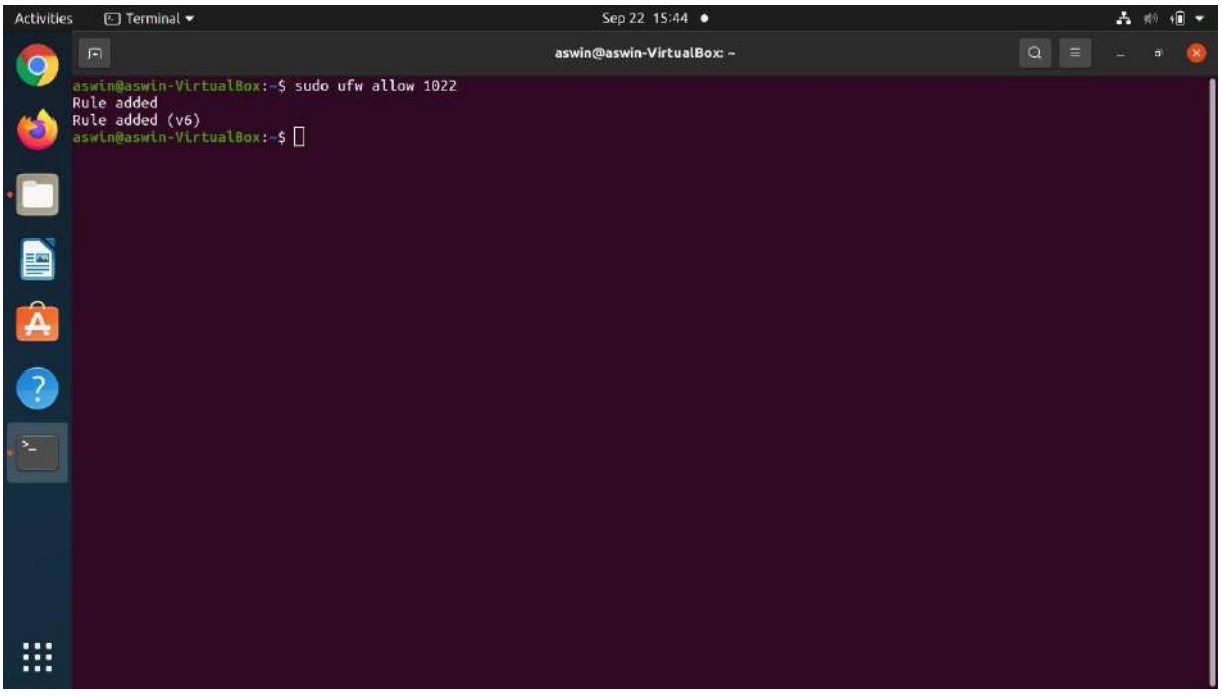
With the above command, the port 22 will be allowed for incoming connections. We can use the below command directly using the port no 22 to allow the SSH connections.

sudo ufw allow 22

However, if we have configured the SSH daemon to use a different port like 2022 or 1022, then we can use the below command

sudo ufw allow 1022

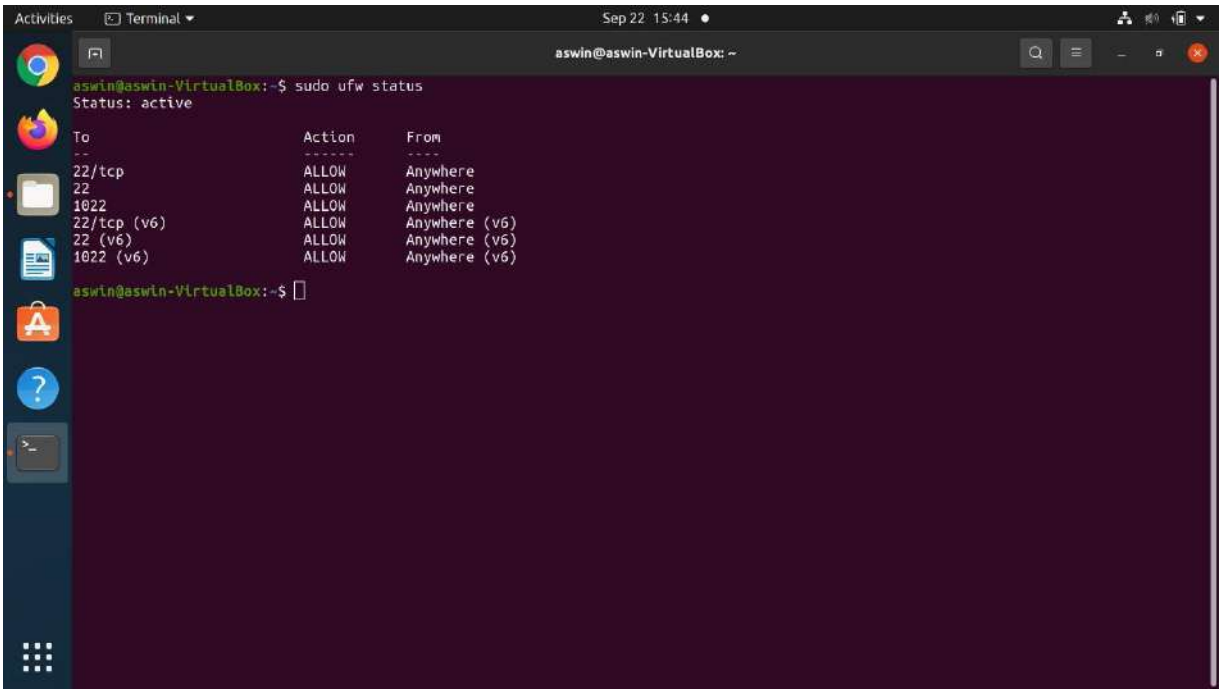




Checking the UFW (Firewall) Status

Below is the command to check the current status of the firewall rules.

sudo ufw status



Enabling the UFW for regular port like (HTTP, HTTPS & FTP)

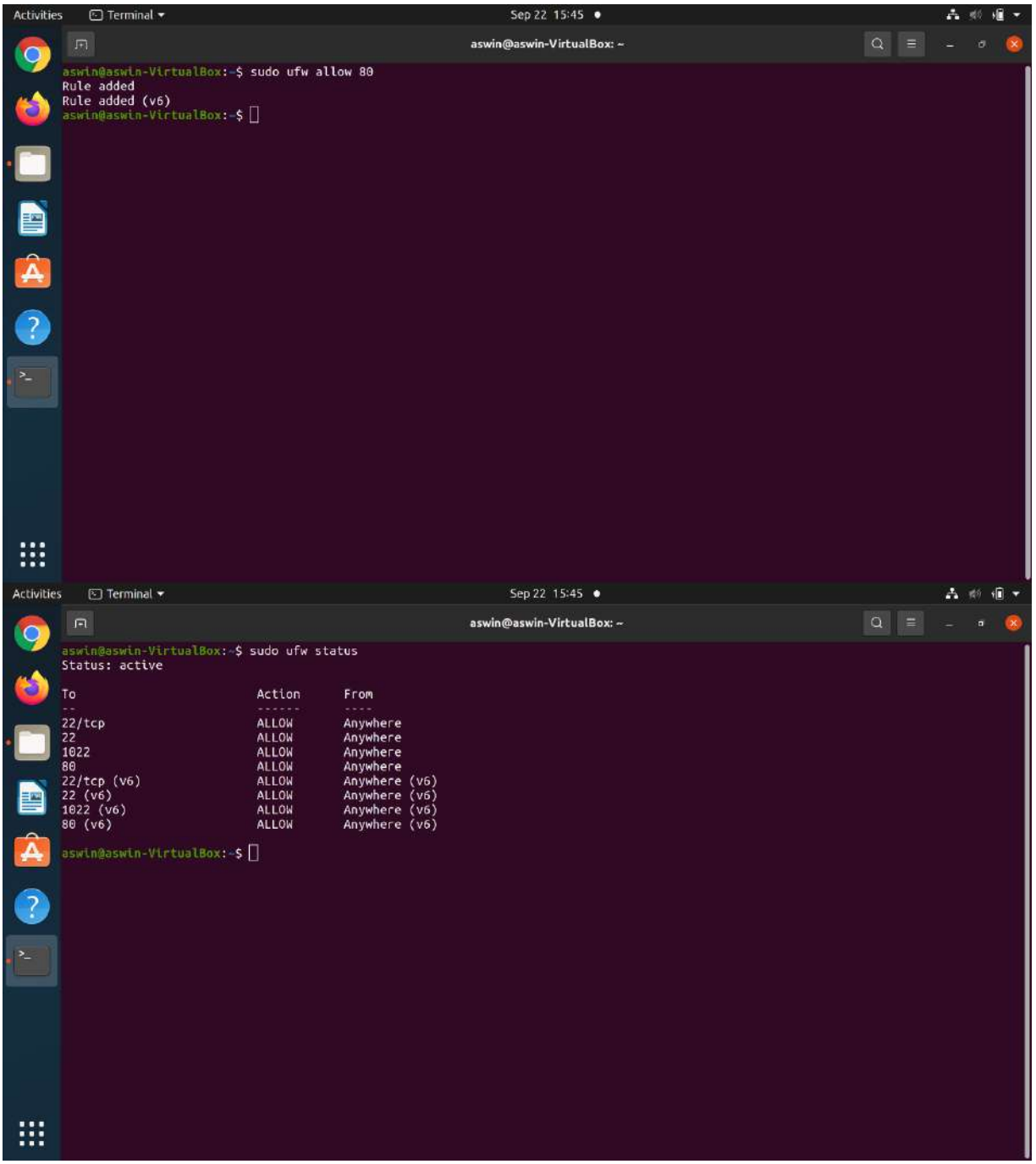
At this point, we will allow others to connect to the server for the regular ports like HTPP, HTTPS, and FTP ports respectively.

HTTP port 80

sudo ufw allow 80

We can check the UFW (Firewall) status using the below command

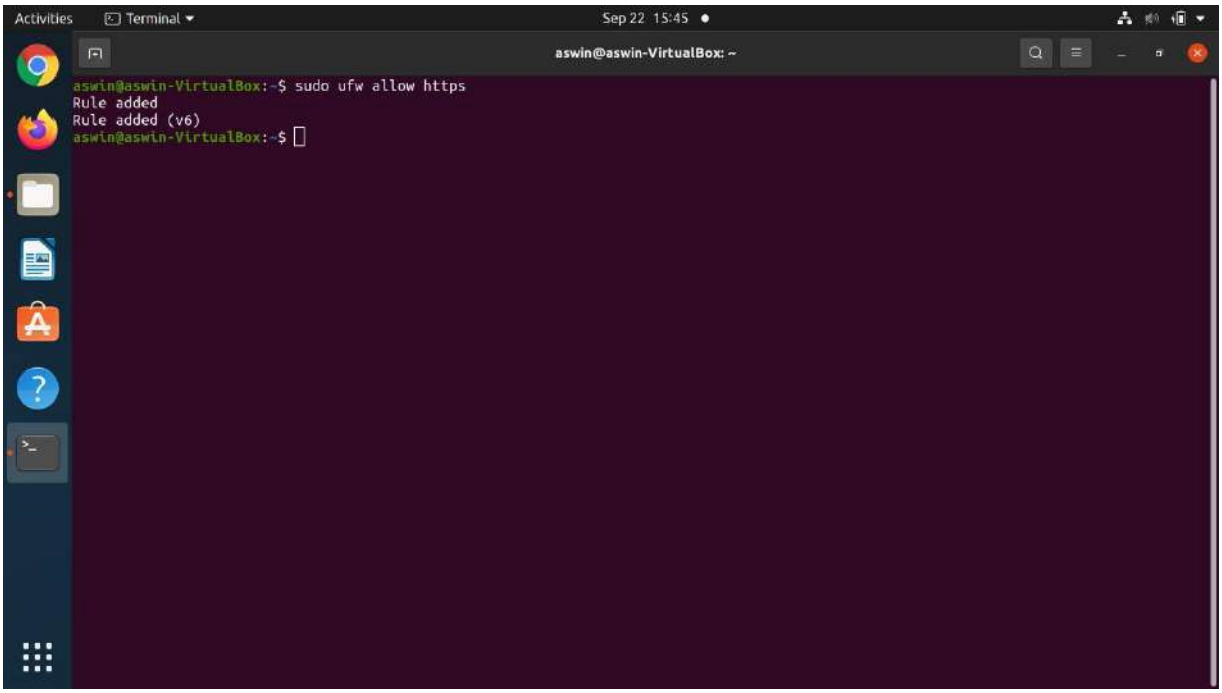
sudo ufw status

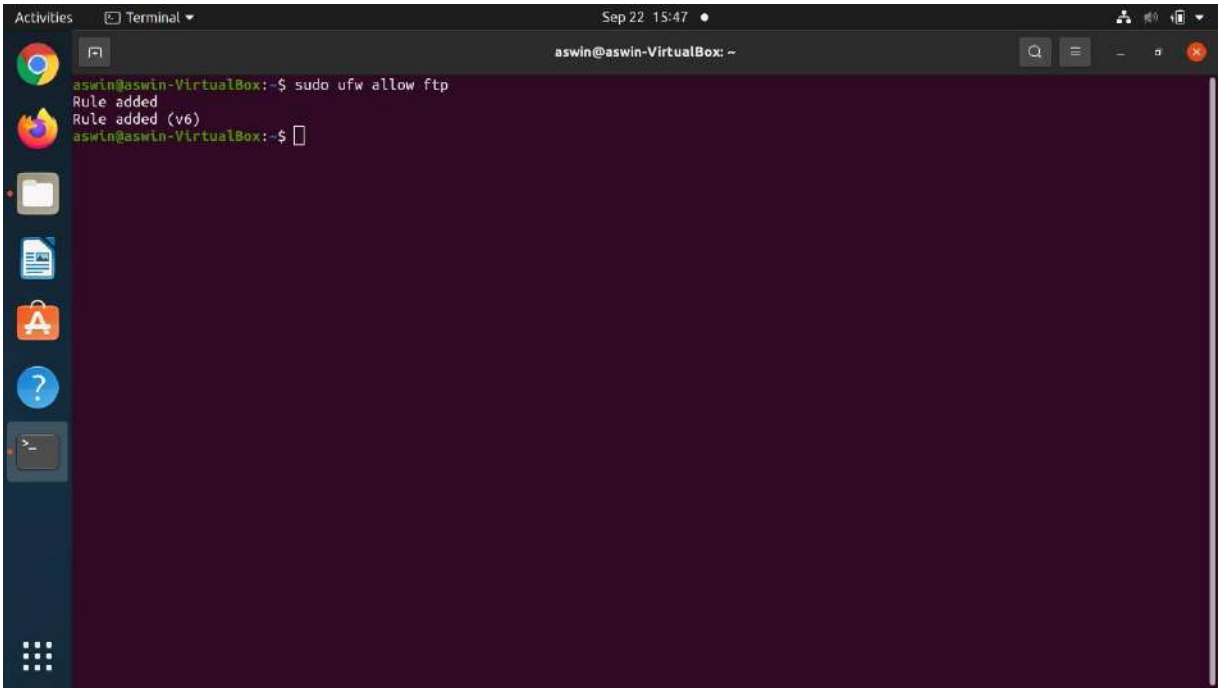


Like that will use the below command to enable HTTPs and FTP ports (443 and 21) respectively.

sudo ufw allow https

sudo ufw allow ftp

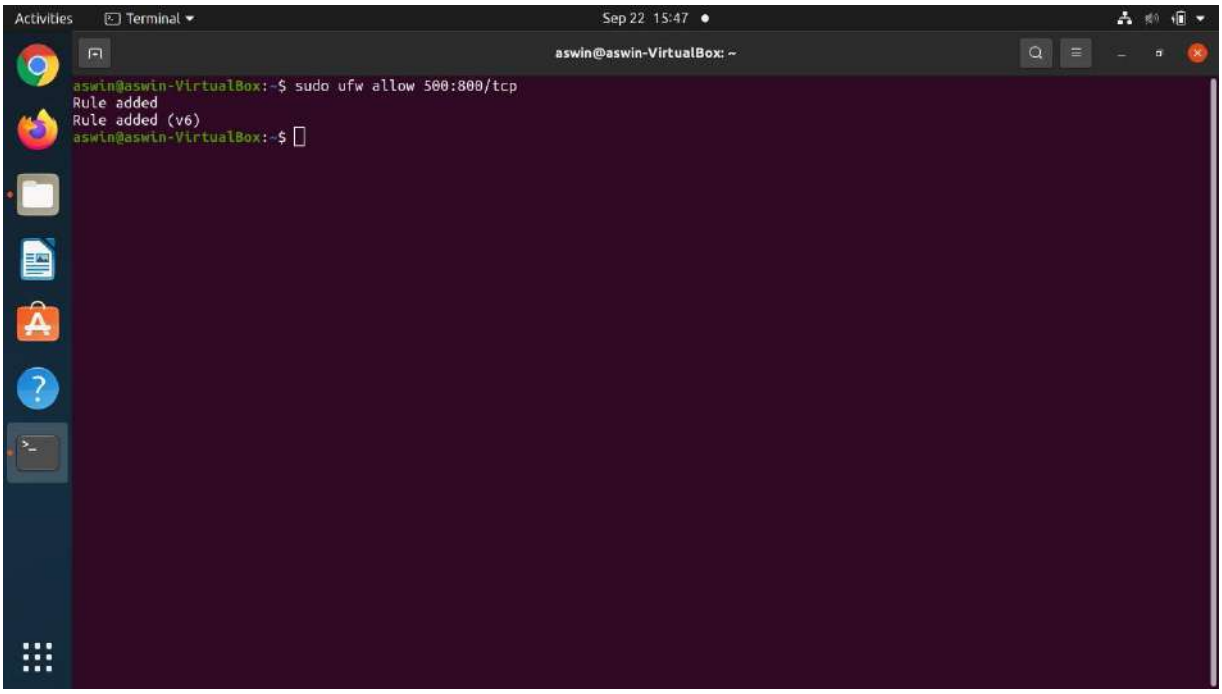




Enabling to Allow Specific Range of Ports

We can also allow or deny particular ranges of ports with UFW to allow the multiple ports instead of allowing single ports. Below is the command to enable a specific range of ports.

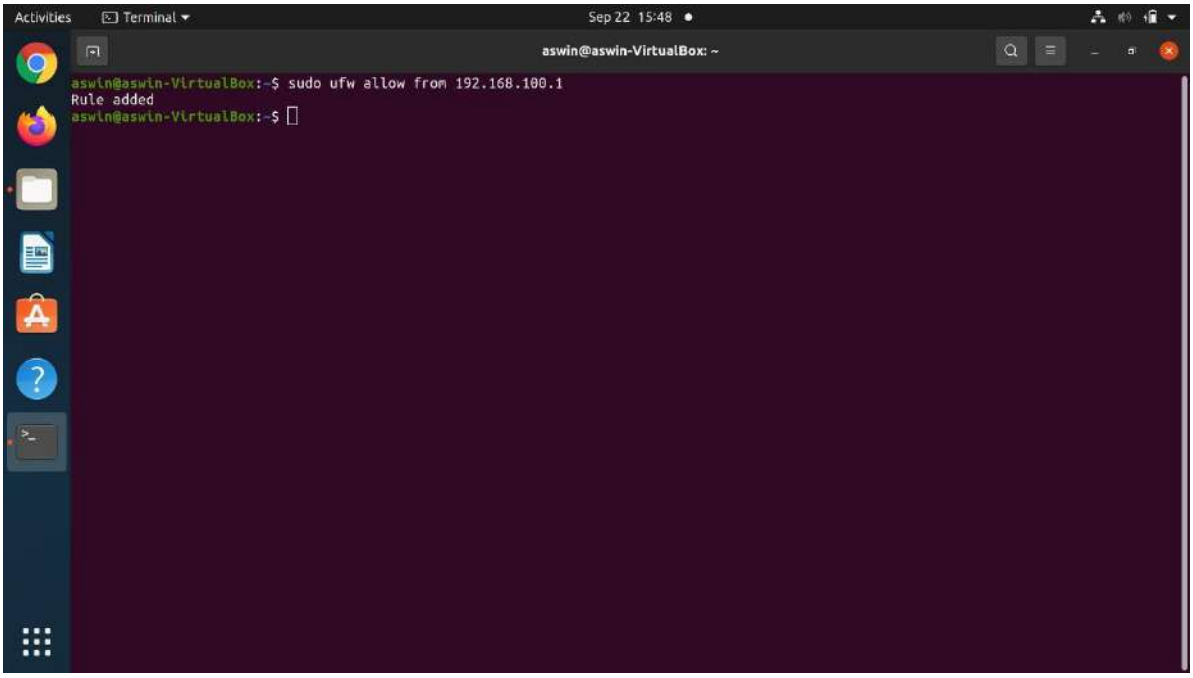
sudo ufw allow 500:800/tcp



Enable to Allow specific IP Addresses

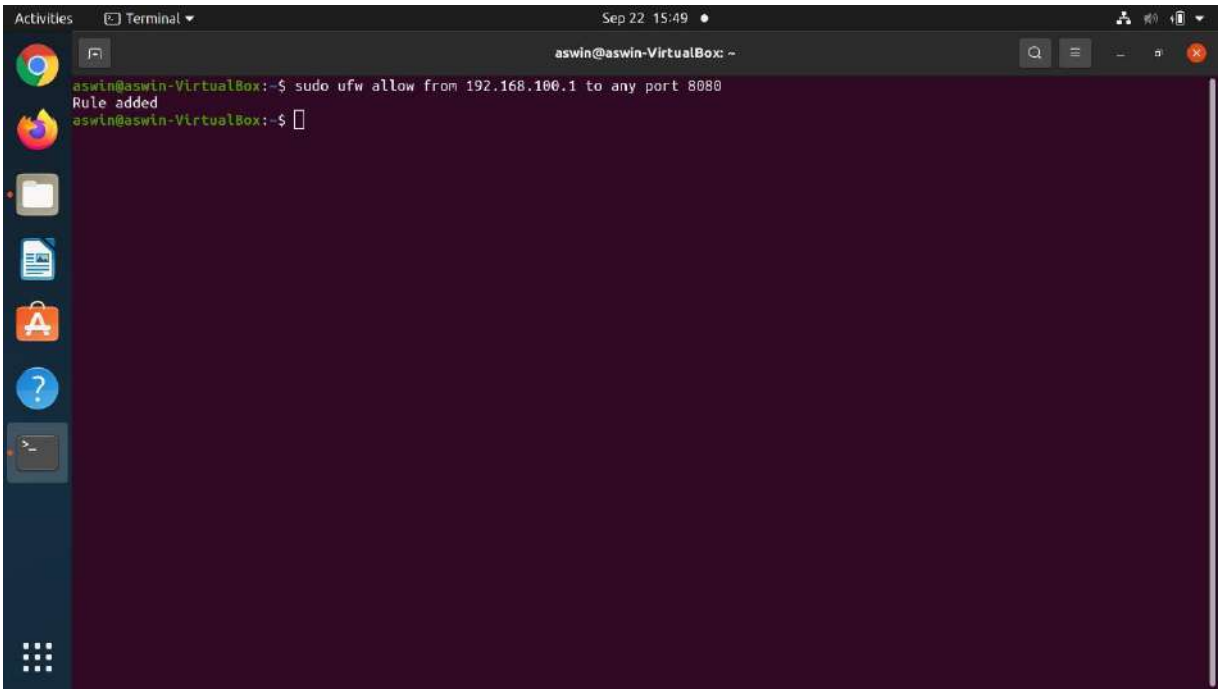
Ifwewanttoallowa particularmachinetoallowforalltheports.Wecanusethe below command.

sudo ufw allow from 192.168.100.1



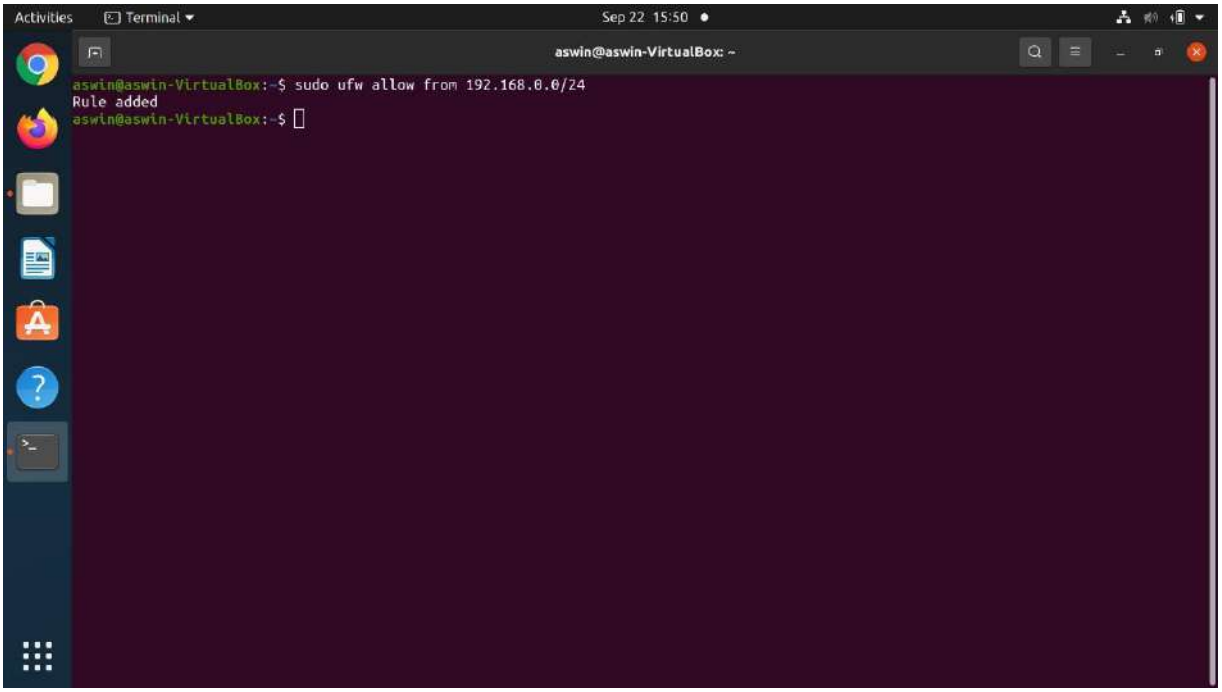
If we want to allow for only specific port we can use the below command.

sudo ufw allow from 192.168.100.1 to any port 8080



If we want to enable the specific subnets like we want to enable for office networks we can use the below command.

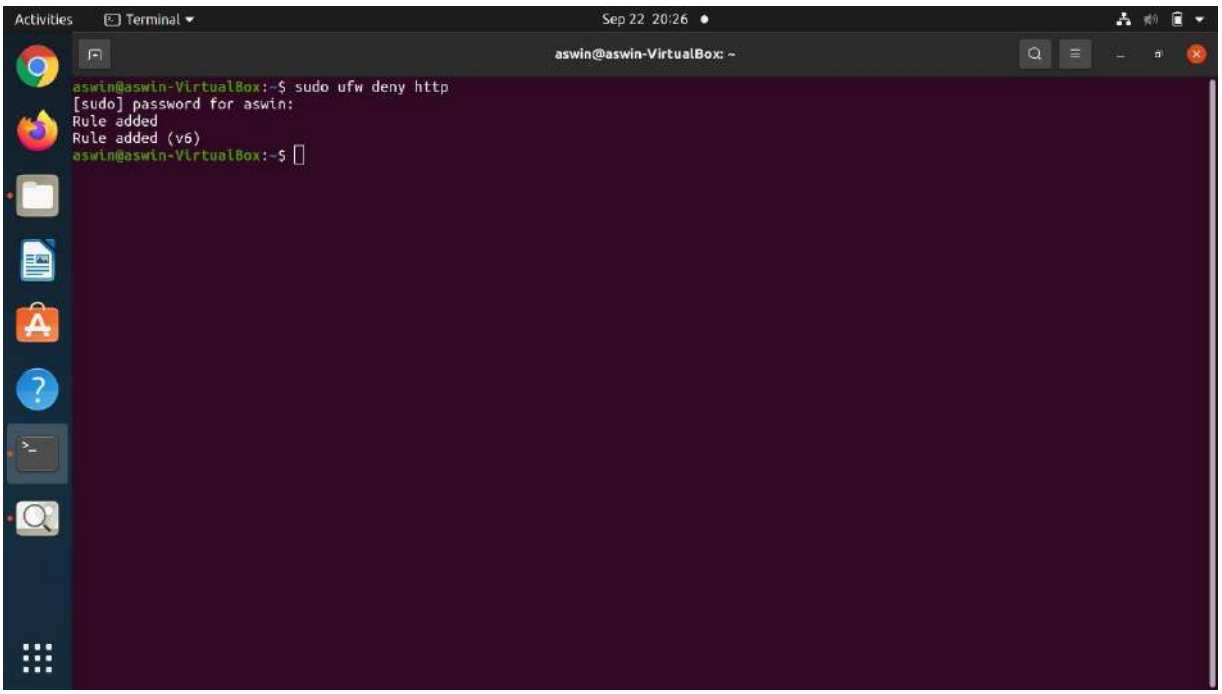
sudo ufw allow from 192.168.0.0/24



Deny the Connections or Rules

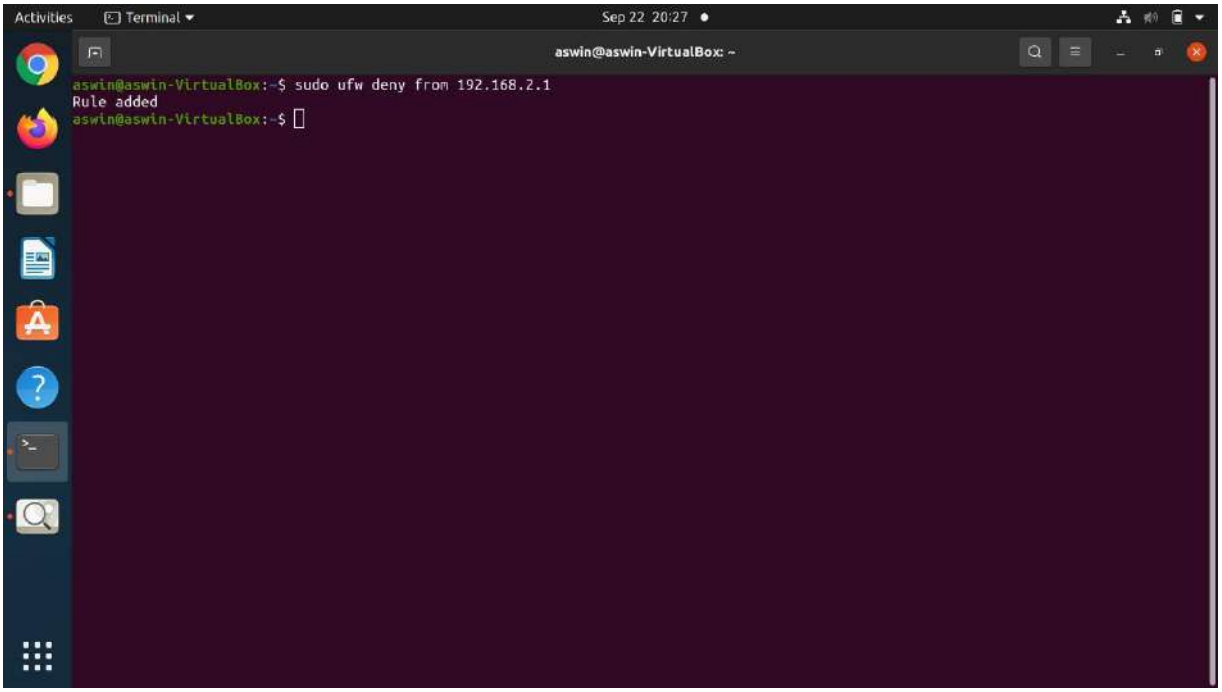
Ifwewanttodenyanyportsornetworkwecanusethebelowcommandstodenythe connections.

sudo ufw denyhttp



Ifwewanttodenyalltheconnectsfromaspecificnetworkwecanusethebelow command.

sudo ufw deny from 192.168.2.1



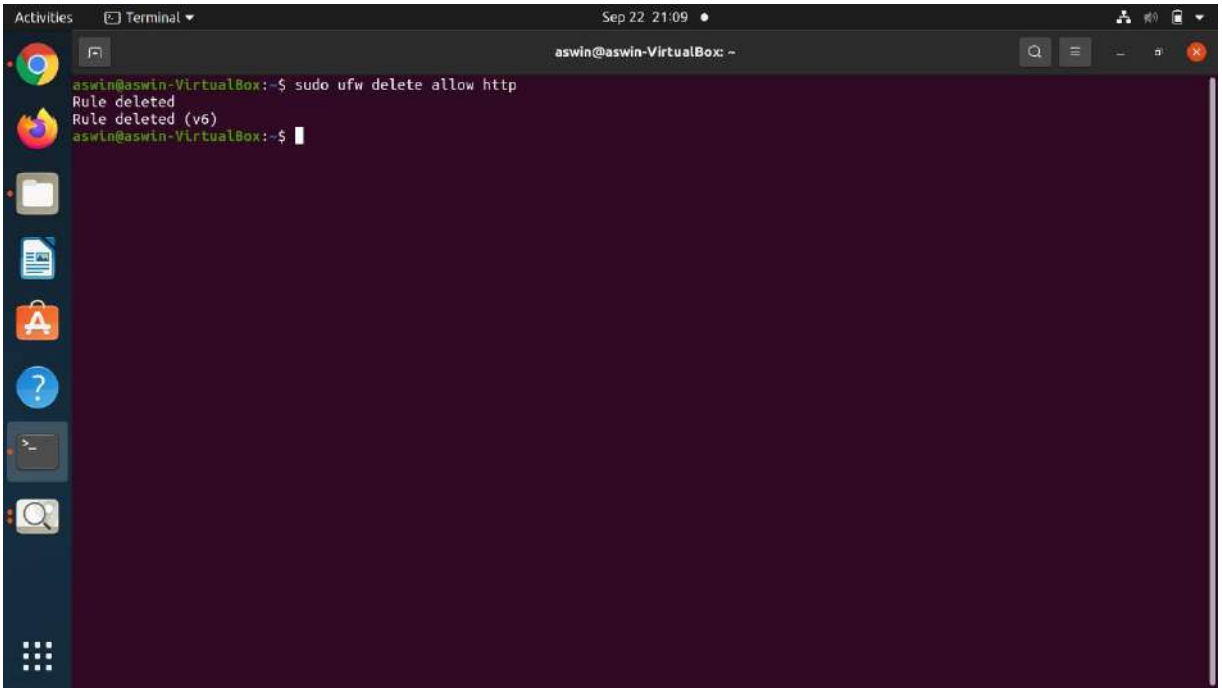
Deleting the Rules

We can delete the rules in two ways one with the actual rules and other with the rules numbers.

Actual Rules

The rules can be deleted using the actual rule which we allowed using the allow command. Below is the command to delete the HTTP rules from UFW.

sudo ufw allow http

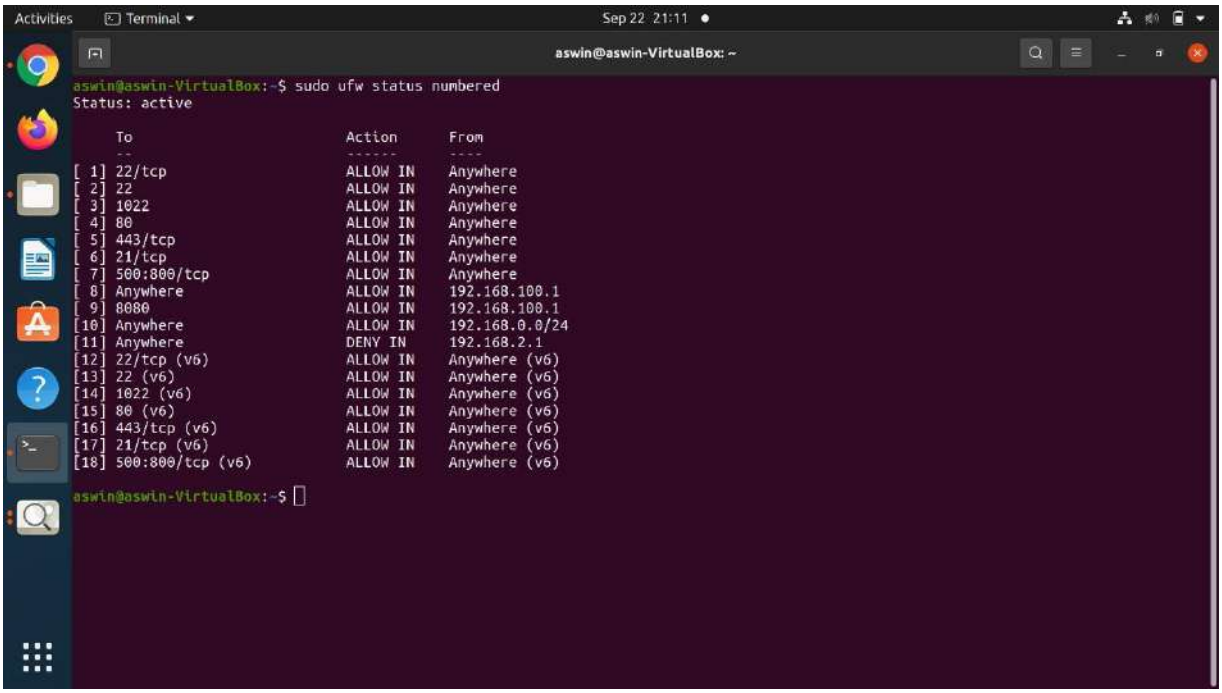


sudo ufw delete allow http

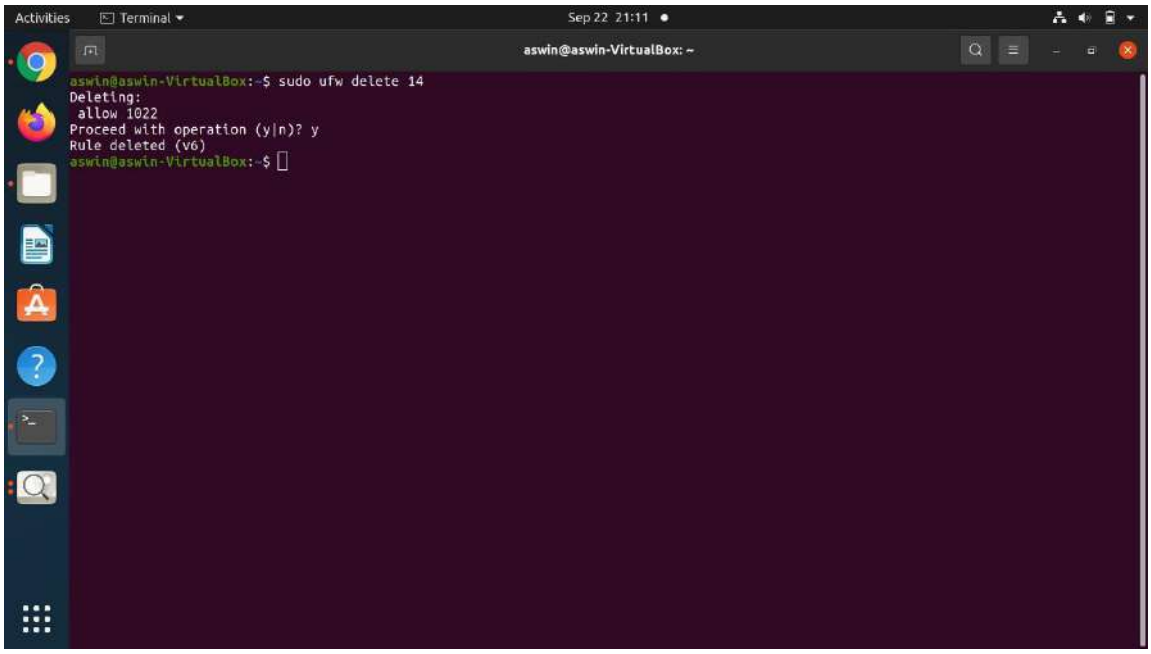
Rules Number

We can use the Rules numbers to delete the firewall rules, we can get the list of firewall rules with the below command.

sudo ufw status numbered



If we want to delete the rule 14, then we can use the below command to delete the rules with the below command.



sudo ufw delete

EXPIRIMENT 9:

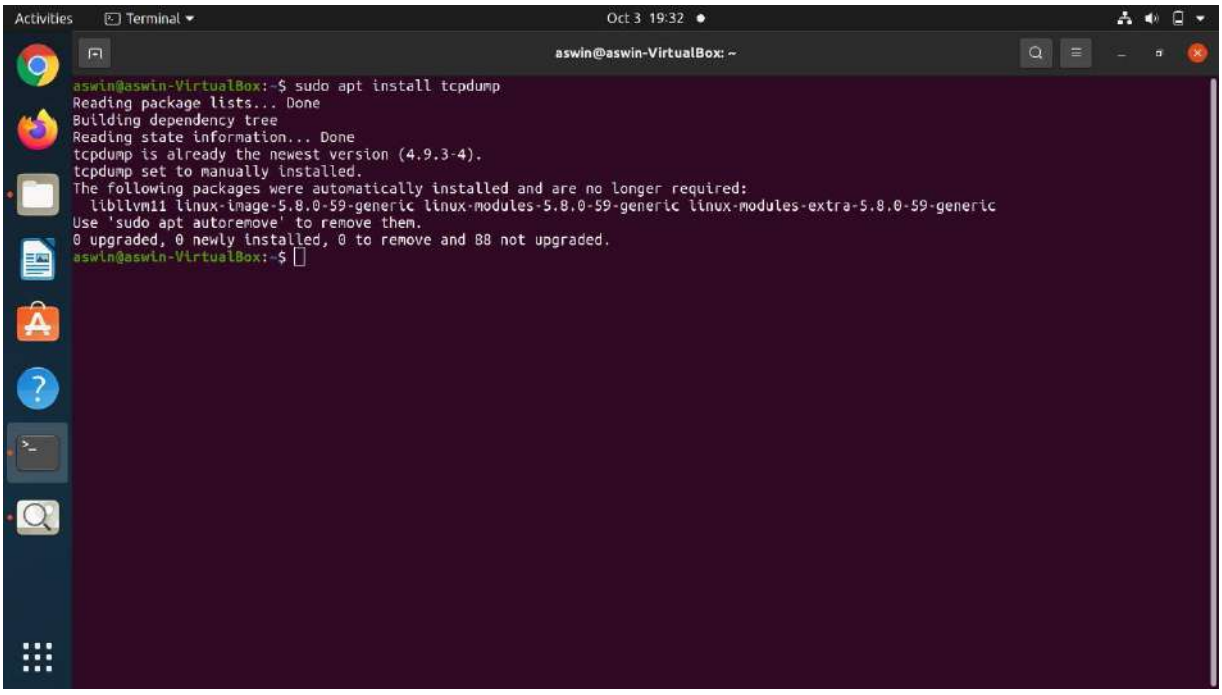
Aim: Analysing network packet stream using tcpdump and wireshark. Perform basic network service tests using nc.

Solution :-

tcpdump:

tcpdump is a packet sniffing and packet analyzing tool for a System Administrator to troubleshoot connectivity issues in Linux. It is used to capture, filter, and analyze network traffic such as TCP/IP packets going through your system. It is many times used as a security tool as well. It saves the captured information in a pcap file, these pcap files can then be opened through Wireshark or through the command tool itself.

Installing tcpdump tool in Linux

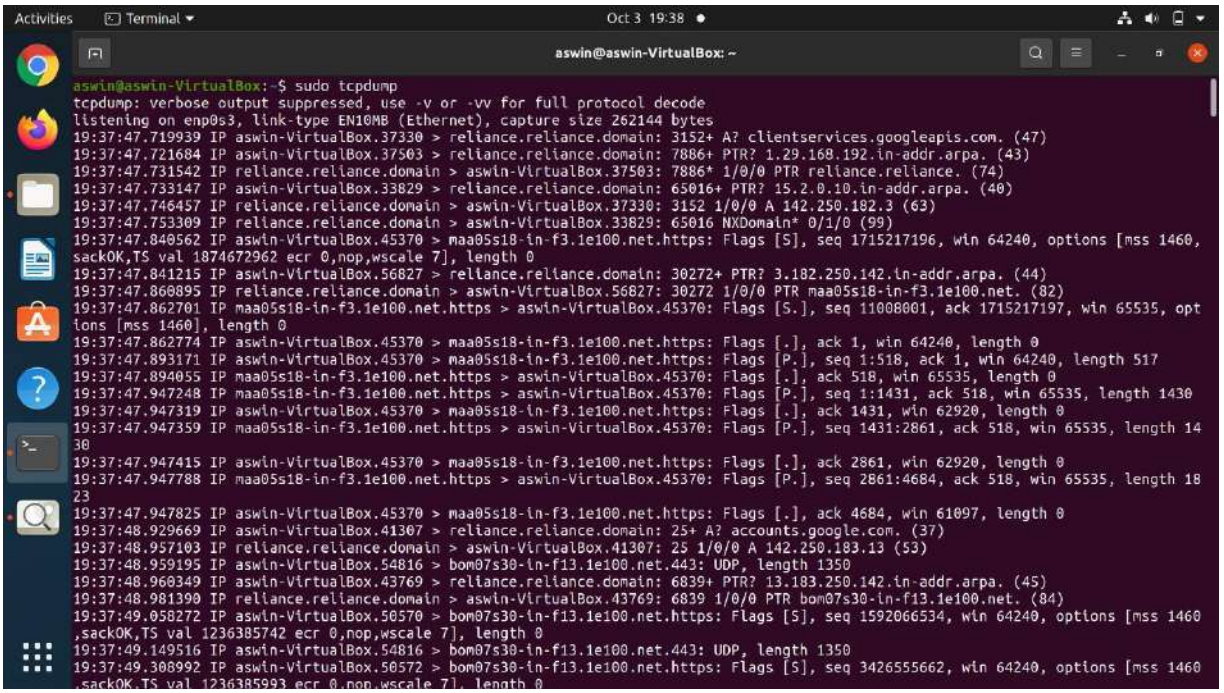


Working with tcpdump command

1. To capture the packets of current network interface

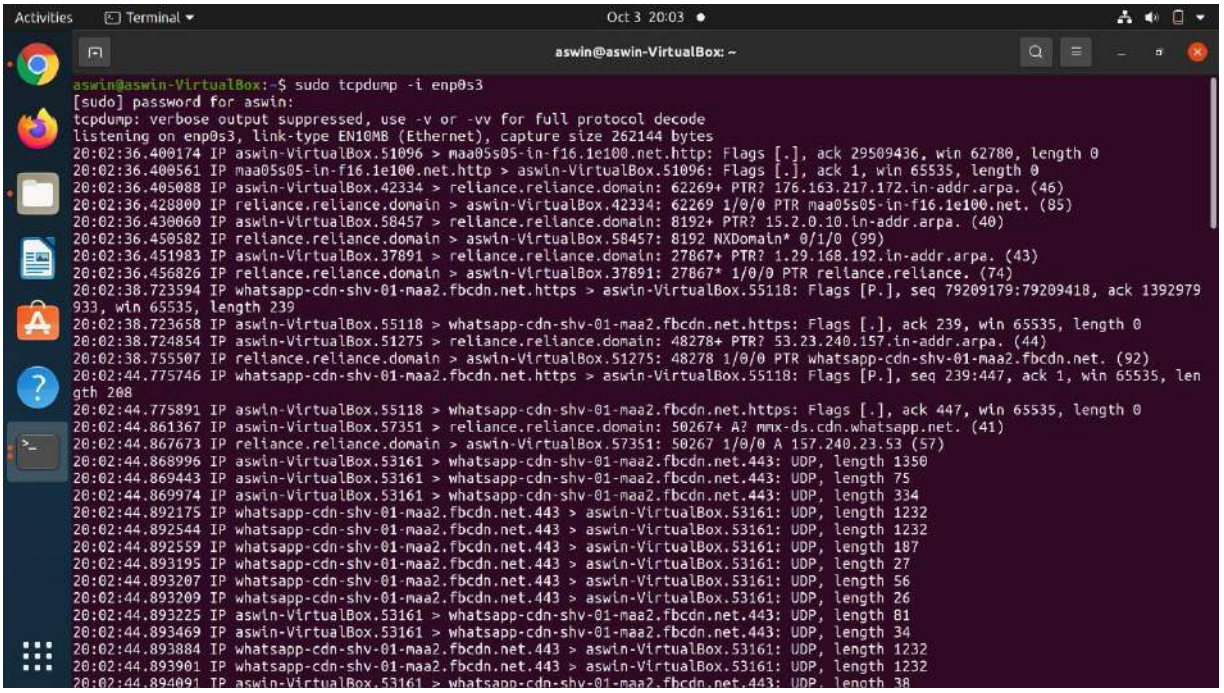
sudo tcpdump

This will capture the packets from the current interface of the network through which the system is connected to the internet.



2. To capture packets from a specific network interface

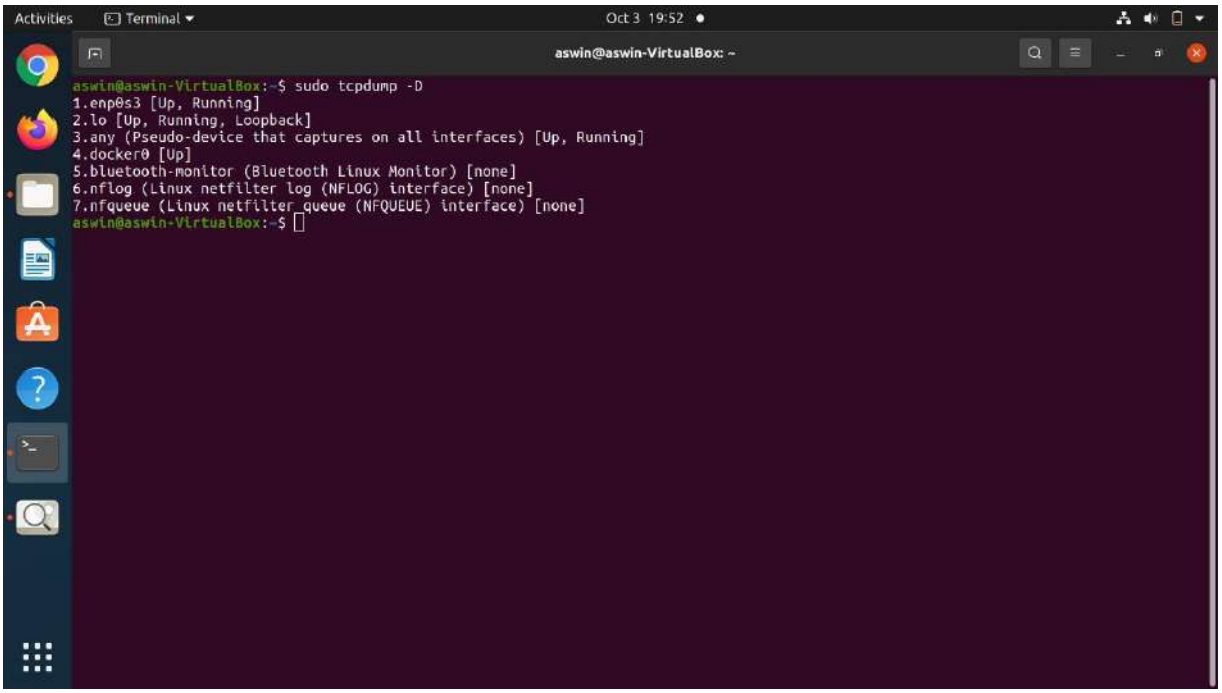
sudo tcpdump -i enp0s3



This command will now capture the packets fromwlp2s0 network interface.

3) To display all available interfaces

sudo tcpdump -D



wireshark:

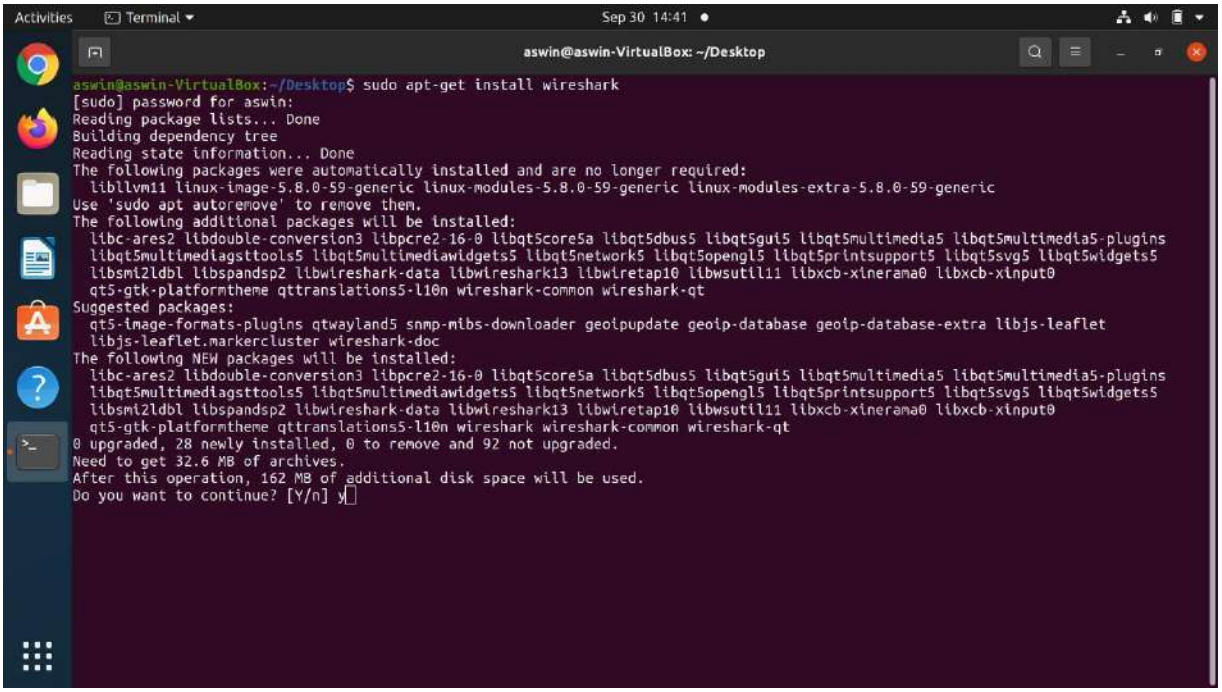
Wireshark is a software tool used to monitor the network traffic through a network interface. It is the most widely used network monitoring tool today. Wireshark is loved equally by system administrators, network engineers, network enthusiasts, network security professionals and black hat hackers. The extent of its popularity is such, that experience with Wireshark is considered as a valuable/essential trait in a computer networking related professional.

There are many reasons why Wireshark is so popular :

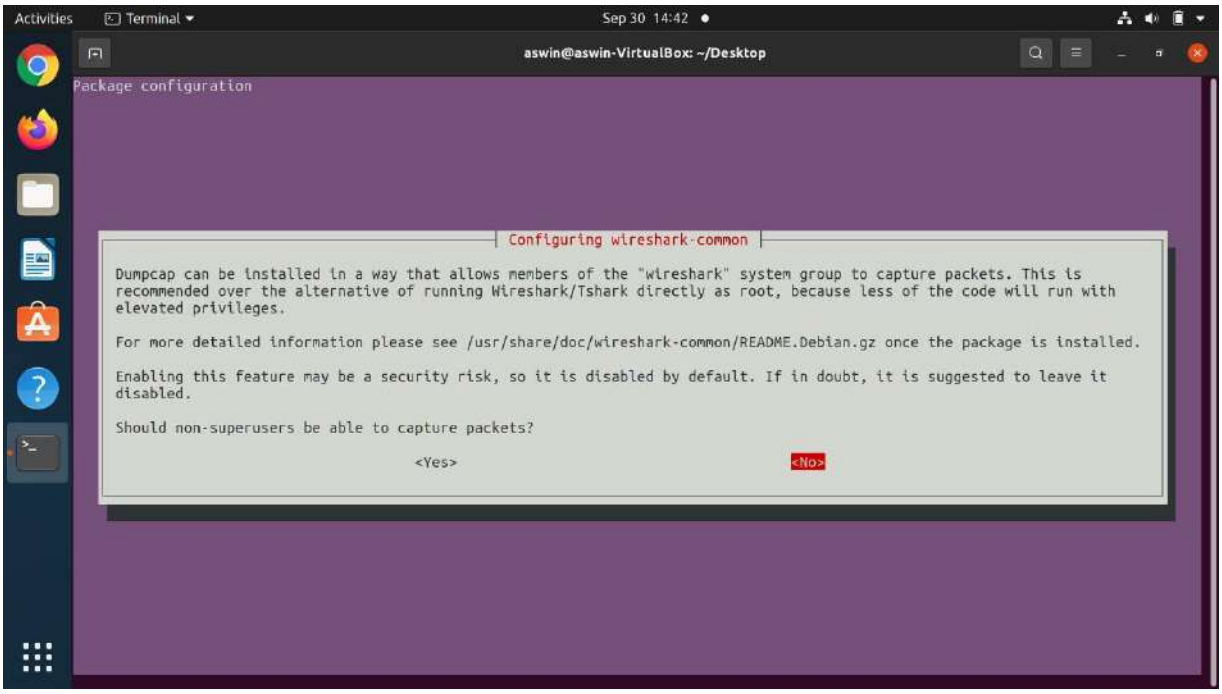
- It has a great GUI as well as a conventional CLI(T Shark).
- It offers network monitoring on almost all types of network standards (ethernet, wlan, Bluetooth etc)
- It is open source with a large community of backers and developers.
- All the necessary components for monitoring, analysing and documenting the network traffic are present. It is free to use.

Wireshark installation:

sudo apt install wireshark command is used to install wireshark in linux

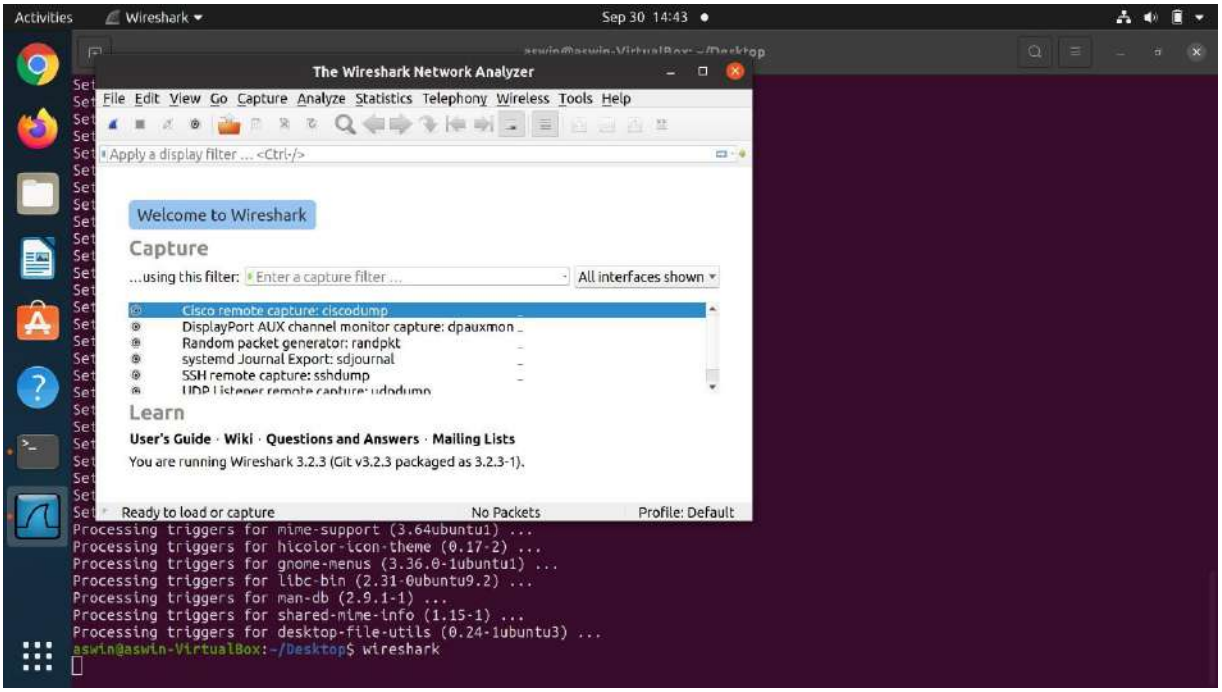


Enter Y to continue



You can give appropriate option according to your need. After this the wireshark will be downloaded to the system

On launching Wireshark, you will see a screen like this:



The basic features of Wireshark are:

1) Packet Monitor: This segment visually shows the packets flowing inside the network. There are colour codes for each type of packets. The packets are shown with following

information:

1. Source address
2. Destination address
3. Packet type
4. Hex dump of the packet
5. Contents of the packet in text
6. Source port(if applicable)
7. Destination port(if applicable)

2) Import from a capture file:

This feature lets you import packets dump from a capture file to analyse further. There are many formats supported by Wireshark, some of them are:

- pcapng
- libpcap
- Oracle snoop and atmsnoop
- Finisar (previously Shomiti) Surveyor captures
- Microsoft Network Monitor captures
- Novell LANalyzer captures
- AIX iptrace captures
- Cinco Networks NetXray captures
- Network Associates Windows-based Sniffer and Sniffer Pro captures
- Network General/Network Associates DOS-based Sniffer (compressed or uncompressed) captures
- AG Group/WildPackets/Savvius EtherPeek/TokenPeek/AiroPeek/EtherHelp/PacketGrabber captures
- RADCOM's WAN/LAN Analyzer captures
- Network Instruments Observer version 9 captures
- Lucent/Ascend router debug output
- HP-UX's nettl

- Toshiba's ISDN routers dump output
- ISDN4BSD i4btrace utility
- Traces from the EyeSDN USB S0
- IPLog format from the Cisco Secure Intrusion Detection System

pppd logs (pppdump format)

- the output from VMS's TCPIPtrace/TCPtrace/UCX$TRACE utilities
- the text output from the DBS Etherwatch VMS utility
- Visual Networks' Visual UpTime traffic capture
- the output from CoSine L2 debug
- the output from Accellent's 5Views LAN agents
- Endace Measurement Systems' ERF format captures
- Linux Bluez Bluetooth stack hcidump -w traces
- Catapult DCT2000 .out files
- Gammu generated text output from Nokia DCT3 phones in Netmonitor mode
- IBM Series (OS/400) Comm traces (ASCII & UNICODE)
- Juniper Netscreen snoop captures
- Symbian OS btsnoop captures
- Tamosoft CommView captures
- Textronix K12xx 32bit .rf5 format captures
- Textronix K12 text file format captures
- Apple PacketLogger captures
- Captures from Aethra Telecommunications' PC108 software

3) Export to a capture file: Wireshark lets you save the results as a capture file to continue working on them at later point of time. The supported formats are:

- pcapng (*.pcapng)
- libpcap, tcpdump and various other tools using tcpdump's capture format (*.pcap, *.cap, *.dmp)

- Accellent 5Views (*.5vw)
- HP-UX's nettl (*.TRC0, *.TRC1)
- Microsoft Network Monitor – NetMon (*.cap)
- Network Associates Sniffer – DOS (*.cap, *.enc, *.trc, *fdc, *.syc)
- Network Associates Sniffer – Windows (*.cap)
- Network Instruments Observer version 9 (*.bfr)
- Novell LANalyzer (*.tr1)
- Oracle (previously Sun) snoop (*.snoop, *.cap)
- Visual Networks Visual UpTime traffic (*.*).

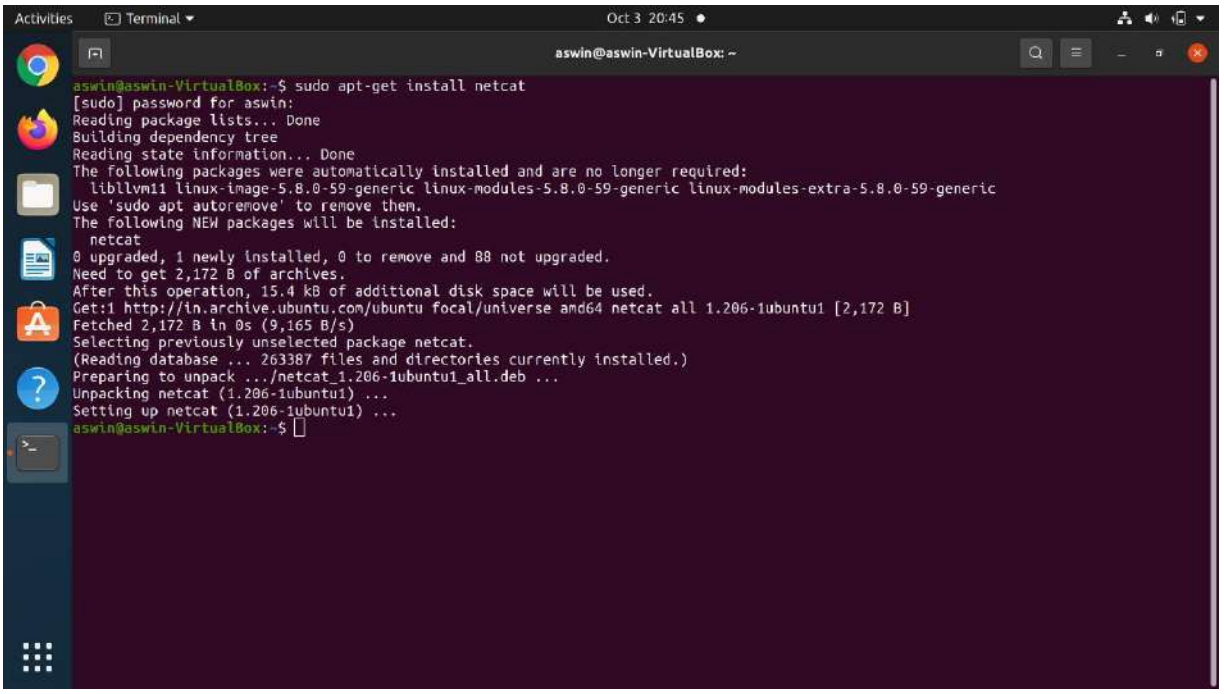
Netcat:

Netcat (or nc in short) is a simple yet powerful networking command-line tool used for performing any operation in Linux related to TCP, UDP, or UNIX-domain sockets.

Netcat can be used for port scanning, port redirection, as a port listener (for incoming connections); it can also be used to open remote connections and so many other things. Besides, you can use it as a backdoor to gain access to a target server.

Installing netcat on linux:

sudo apt-get install netcat



Port scanning:

Netcat can be used for port scanning: to know which ports are open and running services on a target machine. It can scan a single or multiple or a range of open ports.

The -z option sets nc to simply scan for listening daemons, without actually sending any data to them. The -v option enables verbose mode and -w specifies a timeout for connection that cannot be established.

Syntax:

nc -vz IP_address port

Connection timed out:

A connection timed out response indicates that your connection is not working, which could mean your firewall is blocking the port. Test the connection status by adding a rule that accepts connections on the required port.

Connection succeeded

If the initial connection succeeds, Netcat can connect to the service. Look at the connection in more detail.

Syntax:

nc -vt IP Address Port

Closing the connection

You can terminate the connection by either pressing Ctrl-C or type the service-specific quit command.

EXPIRIMENT 10:

Aim: Introduction to Hypervisors and VMs, Xen or KVM , Introduction to Containers: Docker, installation and deployment.

Solution :-

Hypervisor:

A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

Hypervisors make it possible to use more of a system's available resources and provide greater IT mobility since the guest VMs are independent of the host hardware. This means they can be easily moved between different servers. Because multiple virtual machines can run off of one physical server with a hypervisor, a hypervisor reduces:

- Space
- Energy
- Maintenance requirements

There are two main hypervisor types, referred to as "Type 1" (or "bare metal") and "Type 2" (or "hosted"). A type 1 hypervisor acts like a lightweight operating system and runs directly on the host's hardware, while a type 2 hypervisor runs as a software layer on an operating system, like other computer programs.

The most commonly deployed type of hypervisor is the type 1 or bare-metal hypervisor, where virtualization software is installed directly on the hardware where the operating system is normally installed. Because bare-metal hypervisors are isolated from the attack-prone operating system, they are extremely secure. In addition, they generally perform better and more efficiently than hosted hypervisors. For these reasons, most enterprise companies choose bare-metal hypervisors for data centre computing needs.

Benefits of hypervisors

There are several benefits to using a hypervisor that hosts multiple virtual machines:

- Speed: Hypervisors allow virtual machines to be created instantly, unlike bare-metal servers. This makes it easier to provision resources as needed for dynamic workloads.

- Efficiency: Hypervisors that run several virtual machines on one physical machine's resources also allow for more efficient utilization of one physical server. It is more cost- and energy-efficient to run several virtual machines on one physical machine than to run multiple underutilized physical machines for the same task.
- Flexibility: Bare-metal hypervisors allow operating systems and their associated applications to run on a variety of hardware types because the hypervisor separates the OS from the underlying hardware, so the software no longer relies on specific hardware devices or drivers.
- Portability: Hypervisors allow multiple operating systems to reside on the same physical server (host machine). Because the virtual machines that the hypervisor runs are independent from the physical machine, they are portable. IT teams can shift workloads and allocate networking, memory, storage and processing resources across multiple servers as needed, moving from machine to machine or platform to platform. When an application needs more processing power, the virtualization software allows it to seamlessly access additional machines.

Virtual machine:

A Virtual Machine (VM) is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine. Each virtual machine runs its own operating system and functions separately from the other VMs, even when they are all running on the same host. This means that, for example, a virtual MacOS virtual machine can run on a physical PC.

Virtual machine technology is used for many use cases across on-premises and cloud environments. More recently, public cloud services are using virtual machines to provide virtual application resources to multiple users at once, for even more cost efficient and flexible compute.

What are virtual machines used for?

Virtual machines (VMs) allow a business to run an operating system that behaves like a completely separate computer in an app window on a desktop. VMs may be deployed to accommodate different levels of processing power needs, to run software that requires a different operating system, or to test applications in a safe, sandboxed environment.
Virtual machines have historically been used for server virtualization, which enables IT teams to consolidate their computing resources and improve efficiency. Additionally, virtual machines can perform specific tasks considered too risky to carry out in a host environment, such as accessing virus-infected data or testing operating systems. Since the

virtual machine is separated from the rest of the system, the software inside the virtual machine cannot tamper with the host computer.

Advantages of virtual machines

Virtual machines are easy to manage and maintain, and they offer several advantages over physical machines:

- VMs can run multiple operating system environments on a single physical computer, saving physical space, time and management costs.
- Virtual machines support legacy applications, reducing the cost of migrating to a new operating system. For example, a Linux virtual machine running a distribution of Linux as the guest operating system can exist on a host server that is running a non-Linux operating system, such as Windows.
- VMs can also provide integrated disaster recovery and application provisioning options.

KVM:

KVM stands for **Kernel-based Virtual Machine**. It allows the kernel to operate as a hypervisor. Moreover, it requires a processor with hardware virtualization extensions such as Intel VT or AMD-V. KVM was initially designed for x86 processors. Later, it was ported to processors such as ARM, PowerPC etc. Operating systems such as FreeBSD and illumos contain KVM as loadable kernel modules. Also, KVM provides hardware-assisted virtualization for many guest operating systems such as Linux, Solaris, Windows, Haiku and OS X. Furthermore, Android 2.2, Darwin 8.-.1 etc. work with some limitations.

Furthermore, some graphical management tools having KVM are as follows.

Kimchi is KVM's web-based virtualization management tool

Virtual Machine Manager allows creating, editing, starting and stopping KVM based virtual machine

OpenQRM allows managing and controlling various data centre infrastructures.

GNOME Boxes – Gnome interface for handling libvirt guests on Linux.

oVirt is an open-source virtualization management tool for KVM

Proxmox Virtual Environment is an open-source virtualization management package with KVM and LXC.

Xen:

Xen or Xen Project is a type 1 hypervisor. It provides services to allow multiple computer operating systems to execute on the same computer hardware simultaneously.

In brief, KVM and Xen are two hypervisors written in C language. The main difference between KVM and Xen is that KVM is a virtualization module in Linux kernel that works similar to a hypervisor while Xen is a type 1 hypervisor that allows multiple operating systems to execute on the same computer hardware simultaneously.

Docker Container:

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Docker containers that run on Docker Engine:

- **Standard:** Docker created the industry standard for containers, so they could be portable anywhere
- **Lightweight:** Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs
- **Secure:** Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry

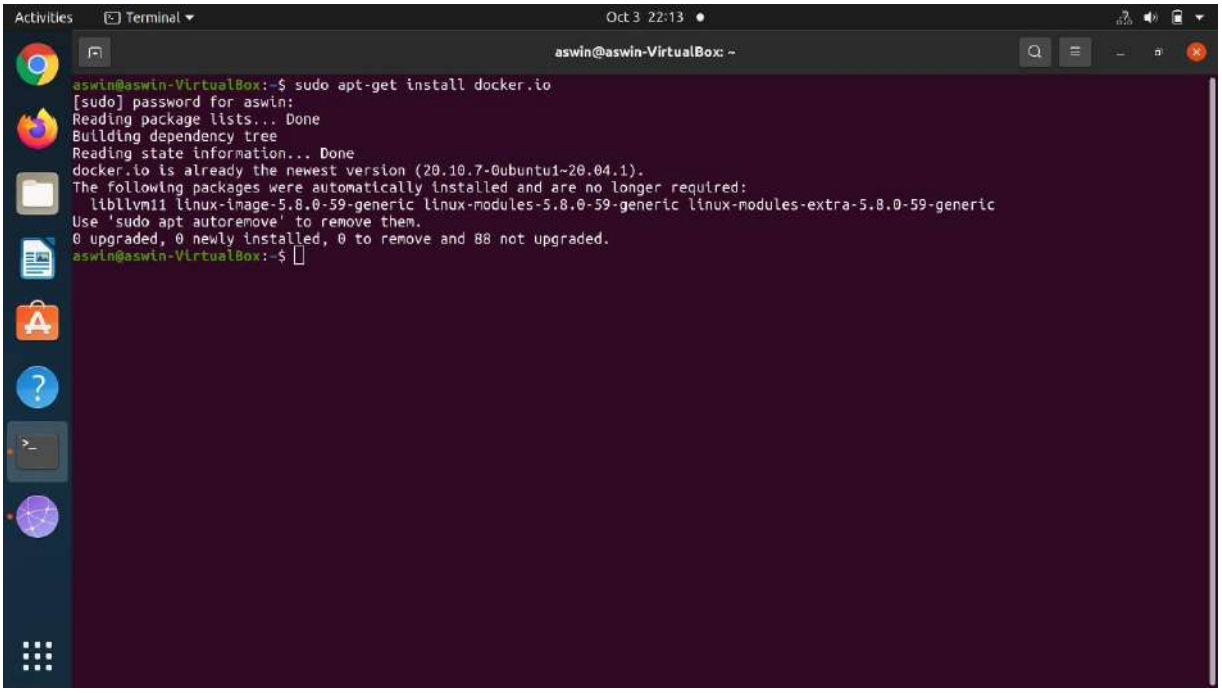
Installing Docker:

1) Updating the local repository

sudo apt update

2) Installing Docker

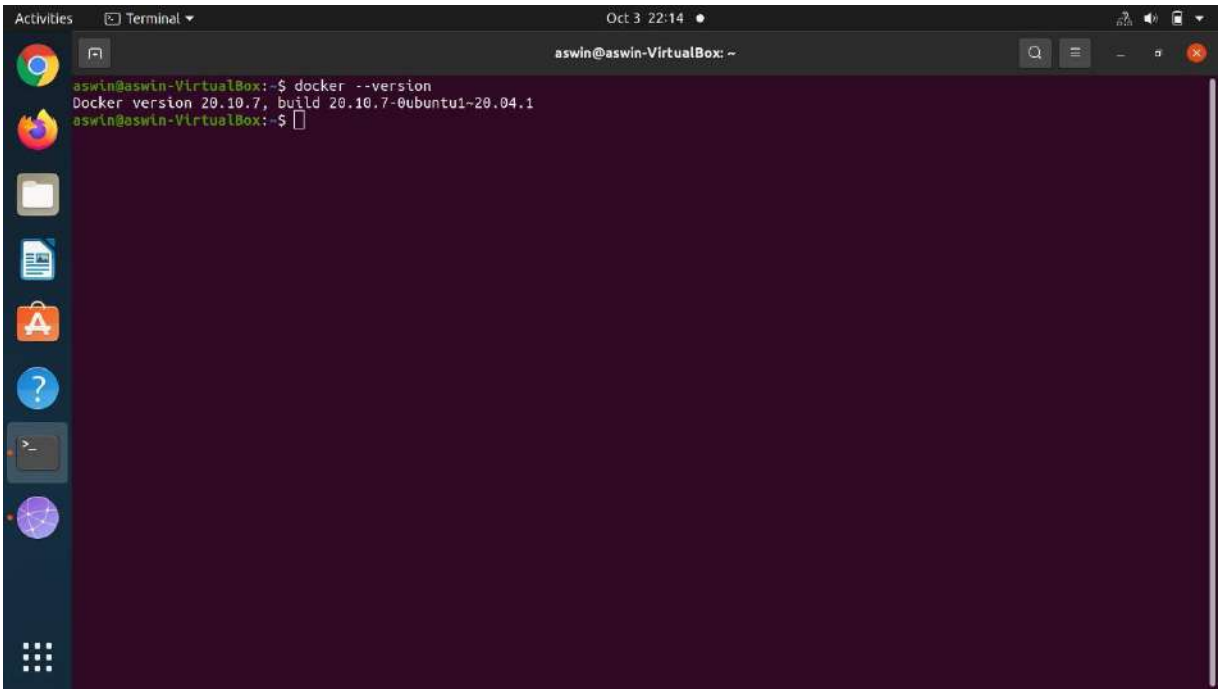
sudo apt install docker.io



Type `y` and hit **Enter** to confirm the installation. Once the install is completed, the output notifies you Docker has been installed.

3) Checking docker installation

docker –version



4) Starting docker service

* Start the Docker service by running:

sudo systemctl start docker

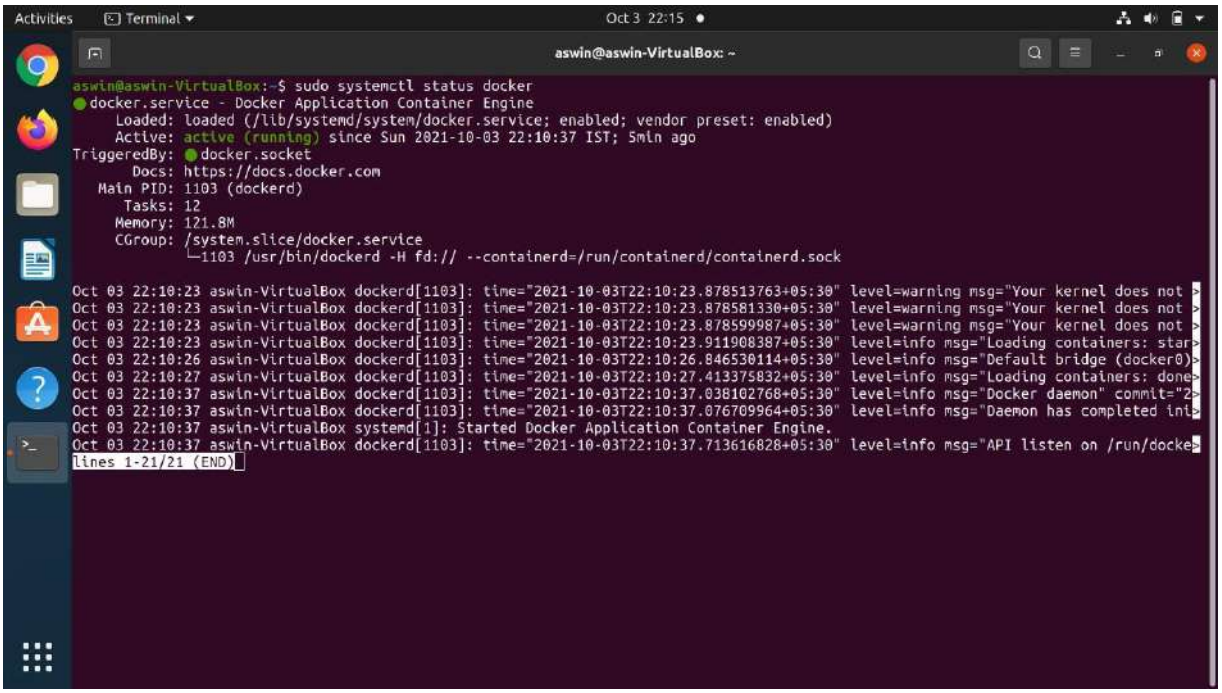
* Then, enable it to run at startup:

sudo systemctl enable docker

*To check the status of the service, run:

sudo systemctl status docker

The output should verify Docker is active (running).



Use Docker on Ubuntu 20.04

The basic syntax for docker commands is:

sudo docker [option] [command] [argument]

Working With Docker Images

Docker images are files that contain the source code, libraries, dependencies, tools, and other files a container need. You can create Docker images with Dockerfiles or use existing ones available on Docker Hub.

To download a new Docker image, use the command:

```
docker pull [image_name]
```

If you don’t know the exact name of the image, search for it in Docker’s repository with:

```
docker search ubuntu
```

After working with Docker for some time, you will collect a local registry of images. Display a list of all Docker images on the system with:

```
docker images
```

Working With Docker Containers

Docker containers are isolated virtual environments that run based on the Docker image assigned to them.
To run a container based on an existing Docker image, use the command:

```
docker run [image_name]
```

Using the command above runs a container but doesn’t move you inside of it. To run a container in interactive mode and change to the container command prompt, run:

```
docker run -it [image_name]
```

Another useful docker command is listing all the containers on the system. To list all active containers, type:

```
docker container ps
```

To view all containers (active and inactive), run:

```
docker container ps -a
```

EXPIRIMENT 11:

Aim: Automation using Ansible: Spin up a new Linux VM using Ansible playbook.

Solution :-

Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools.

It is an automation and orchestration tool popular for the following reasons:

- Simple to Install.
- Free and open source.
- Lightweight and consistent.
- OpenSSH security features make it very secure.

Ansible Concepts

1. Control node:

Commands and Playbooks can run by invoking /usr/bin/ansible or /usr/bin/ansible-playbook, from any control node. You can use any computer that has Python installed on it as a control node. However, one can not use a computer with Windows OS as a control node. One can have multiple control nodes.

2. Managed nodes:

Also sometimes called "hosts", Managed nodes are the network devices (and/or servers) you manage with Ansible.

3. Inventory:

Also sometimes called "hostfile", Inventory is the list of Managed nodes use to organize them. It is also used for creating and nesting groups for easier scaling.

4. Modules:

These are the units of code executed by Ansible. Each module can be used for a specific purpose. One can invoke a single module with a task, or invoke several different modules in a playbook.

5. Tasks:

The units of action in Ansible. One can execute a single task once with an ad- hoc command.

6. Playbooks3:

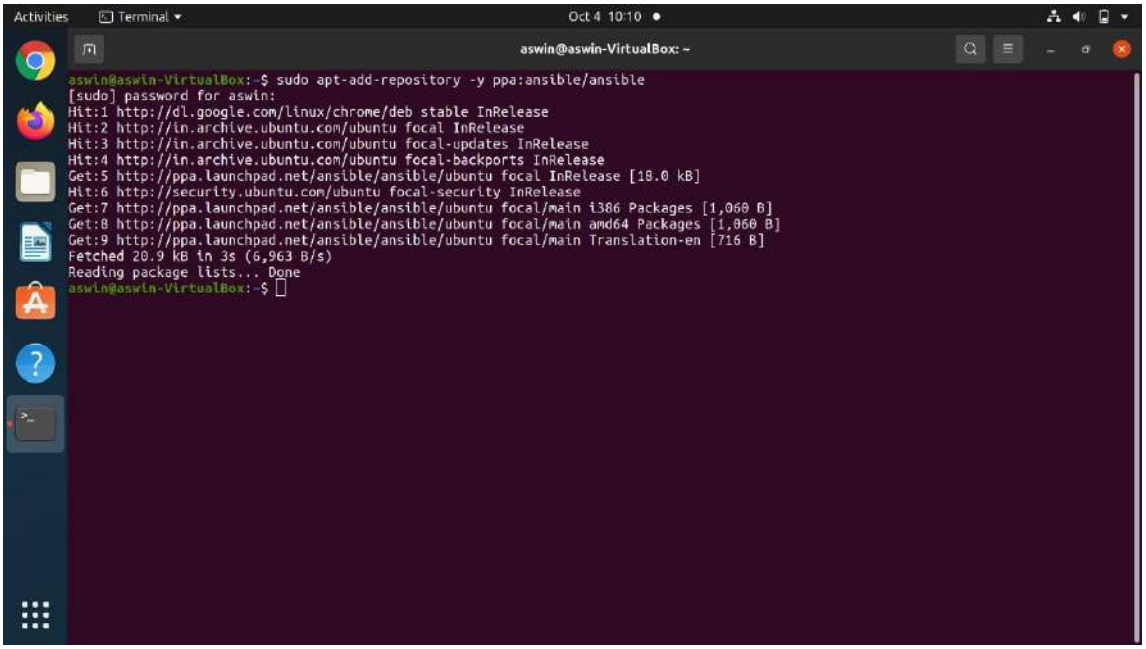
These are the ordered list of tasks that are saved so you can run those tasks

inthatorderrepeatedly.PlaybooksarewritteninYAMLandareeasytoread, write, share and understand.

Installation

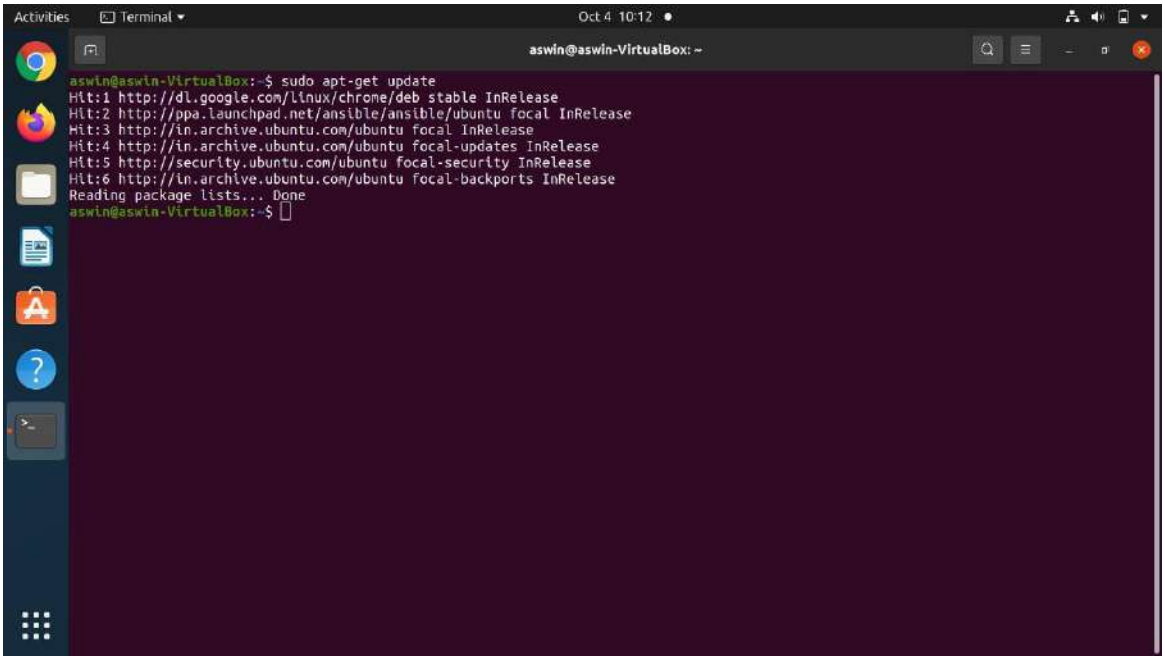
Step 1: Add the Ansible Repository.

sudo apt-add-repository -y ppa:ansible/ansible



Step 2: Update the system repository listings.

sudo apt-get update



Step 3: Install the ansible packages.

sudo apt-get install -y ansible