# VIRTUALBOX INSTALLATION IN UBUNTU

VirtualBox is a general-purpose virtualization tool for x86 and x86-64 hardware, targeted at server, desktop, and embedded use, that allows users and administrators to easily run multiple guest operating systems on a single host.

Step 1: Open a terminal, and enter the following to update the repository: sudo apt-get update

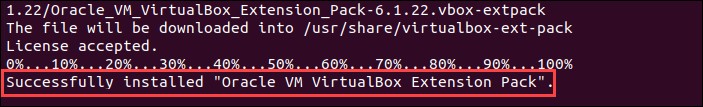
Step 2: Download and install VirtualBox by running:

sudo apt-get install virtualbox

Step 3: Next, install the VirtualBox Extension Pack:

sudo apt-get install virtualbox-ext–pack

**OUTPUT:**



**2. Study of the terminal-based text editor such as Vim or Emacs. Basic Linux commands, familiarity with following commands/operations expected.**

Here discussing some basic Linux commands.

1. **man**

The man stands for manual. The man command displays the user manual of any command that we run on the terminal. It displays the command details such as

NAME, SYNOPSIS, OPTIONS, DESCRIPTION, EXIT STATUS, RETURN VALUES, FIL, ERRORS VERSIONS, AUTHORS, EXAMPLES.

Eg: manls

Display the manual page for the item (program) ls.

1. **ls, echo, read**

**ls**

The ls command is used to view the contents of a directory. By default, this command will display the contents of your current working directory. If you want to see the content of other directories, type ls and then the directory’s path. For example, enter ls /home/username/Documents to view the content of Documents.

**echo**

This command is used to move some data into a file.

Eg: If you want to add the text, “Hello, my name is John” into a file called name.txt, you would type echo Hello, my name is John >> name.txt **read** read command in Linux system is used to read from a file descriptor. Basically, this command read up the total number of bytes from the specified file descriptor into the buffer. If the number or count is zero then this command may detect the errors. But on success, it returns the number of bytes read. Zero indicates the end of the file. If some errors found then it returns -1.

Syntax: read **3. more, less, cat**

**more**

As 'cat' command displays the file content. Same way 'more' command also displays the content of a file. Only difference is that, in case of larger files, 'cat' command output will scroll off your screen while 'more' command displays output one screenful at a time. Syntax: more <filename> **less**

The 'less' command is same as 'more' command but include some more features.

It automatically adjusts with the width and height of the terminal window, while 'more' command cuts the content as the width of the terminal window get shorter.

Syntax: less <filename> **cat** cat (short for concatenate) is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output stdout.

To run this command, type cat followed by the file’s name and its extension. Eg: cat file.txt.

cat -> filename creates a new file. **4. cd, mkdir, pwd, find cd**

To navigate through the Linux files and directories, use the cd. It requires either the full path or the name of the directory, depending on the current working directory that you’re in.

cd .. (With two dots) to move one directory up **mkdir**

Use mkdir command to make a new directory

Eg: If you type mkdir Music it will create a directory called Music. **pwd**

Use the pwd command to find out the path of the current working directory (folder) you’re in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash (/). An example of an absolute path is /home/username.

**find**

Find searches for files and directories. The difference is, you use the find command to locate files within a given directory.

Eg: find /home/ -name notes.txt command will search for a file called notes.txt within the home directory and its subdirectories.

**5. mv, cp, rm, tar**

**mv**

The primary use of the mv command is to move files, although it can also be used to rename files. The arguments in mv are similar to the cp command. You need to type mv, the file’s name, and the destination’s directory. Eg: mv file.txt /home/username/Documents.

To rename files, the Linux is mv oldname.ext newname.ext **cp**

Use the cp command to copy files from the current directory to a different directory. For instance, the command cp scenery.jpg /home/username/Pictures would create a copy of scenery.jpg (from your current directory) into the Pictures directory.

cp –i : will ask for user’s consent in case of a potential file overwrite. **rm**

The rm command is used to delete directories and the contents within them.

Eg: rm Music, will deletes the directory named Music.

**tar**

The tar command is the most used command to archive multiple files into a tarball a common Linux file format that is similar to zip format, with compression being optional.

**6. wc, cut, paste wc**

wc command helps in counting the lines, words, and characters in a file. It displays the number of lines, number of characters, and the number of words in a file. Mostly, it is used with pipes for counting operation.

Syntax: wc [OPTION]... [FILE]...

wc [OPTION]... --files0-from=F **cut** cut command is useful for selecting a specific column of a file. It is used to cut a specific sections by byte position, character, and field and writes them to standard output. It cuts a line and extracts the text data. It is necessary to pass an argument with it; otherwise, it will throw an error message.

Syntax: cut OPTION... [FILE]..

**paste**

Paste command is one of the useful commands in Unix or Linux operating system. It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by **tab** as delimiter, to the standard output. When no file is specified, or put dash (“- “) instead of file name, paste reads from standard input and gives output as it is until an interrupt command **[Ctrl-c]** is given.

Syntax: paste [OPTION]... [FILES]...

**7. head, tail, grep, expr head**

The head command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking.

For example, if you only want to show the first five lines, type head -n 5 filename.ext. **tail**

This one has a similar function to the head command, but instead of showing the first lines, the tail command will display the last ten lines of a text file.

For example, tail -n filename.ext.

**grep**

Another basic Linux command that is undoubtedly helpful for everyday use is grep. It lets you search through all the text in a given file.

Eg: grep blue notepad.txt will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully. Usually output of a previous command is piped into the grep command.

For example: ls -l | grep “kernel” **expr**

The expr command is used to evaluate a given expression and display its standard output. Each separated expression is considered as an argument. These expressions could be integer and string expressions, including regular expressions. If expressions are not passed properly, it will prevent the execution of the command. Syntax: expr expression

**8. chmod, chown chmod**

Linux chmod command is used to change the access permissions of files and directories. It stands for change mode. It cannot change the permission of symbolic links. Even, it ignores the symbolic links come across recursive directory traversal.

Syntax: chmod <options> <permissions> <file name> **chown**

Linux chown command is used to change a file's ownership, directory, or symbolic link for a [user](https://www.javatpoint.com/linux-users) or [group.](https://www.javatpoint.com/linux-groups) The chown stands for change owner. In [Linux,](https://www.javatpoint.com/linux-tutorial) each file is associated with a corresponding owner or group. The Linux system may have multiple users. Every user has a unique name and user ID. If only a user is available in the system, the user will be the owner of each file.

Syntax: chown [OPTION]... [OWNER] [: [GROUP]] FILE...

**9. Redirections & Piping**

**Redirection** is a technique that essentially allows commands to either read data from a text file, or save the output to text files. In other words, it lets you redirect a command’s standard output to a file rather than displaying it on the screen. **Eg:**

file1 $ cat file1 apple 102 cakes drinks bananas 500

301

**After sorting:** file2 $ sort file2 102

301 500 apple bananas cakes drinks

**Piping** is a technique that lets you use Linux commands as building blocks to build your own custom commands.

**Eg:**

$ ls -l total 0

-rw-r--r--. 1 root root 0 Oct 20 19:22 file1

-rw-r--r--. 1 root root 0 Oct 20 19:22 file2 -rw-r--r--. 1 root root 0 Oct 20 19:22 file3 drwxr-xr-x. 2 root root 6 Oct 20 19:22 folder1 drwxr-xr-x. 2 root root 6 Oct 20 19:22 folder2 drwxr-xr-x. 2 root root 6 Oct 20 19:22 folder3 **After piping:**

$ ls -l | grep "^-"

-rw-r--r--. 1 root root 0 Oct 20 19:22 file1

-rw-r--r--. 1 root root 0 Oct 20 19:22 file2

-rw-r--r--. 1 root root 0 Oct 20 19:22 file3 **10. useradd, usermod, userdel, passwd useradd** useradd is used to create a new user, while passwd is adding a password to that user’s account. To add a new person named John type, useradd John and then to add his password type, passwd 12345678

**usermod** usermod command or modify user is a command in Linux that is used to change the properties of a user in Linux through the command line. After creating a user, we have to sometimes change their attributes like password or login directory etc. so in order to do that we use the Usermod command. Eg: sudo usermod -c "This is test user" test\_user **userdel** userdel is to remove a user is very similar to adding a new user. To delete the users account type, Syntax: userdel UserName **passwd** passwdcommand in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.

Syntax:passwd [options] [username]

Eg:Command: passwd

1. **df,top, ps df**

Use df command to get a report on the system’s disk space usage, shown in percentage and KBs. If you want to see the report in megabytes, type df -m.

**top**

As a terminal equivalent to Task Manager in Windows, the top command will display a list of running processes and how much CPU each process uses. It’s very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

**ps**

Ps command will display all current processes along with their process ids (PID). Read manuals for various options.

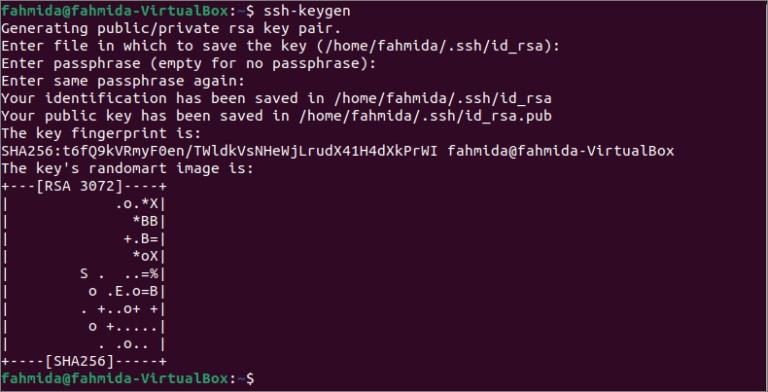
1. **ssh, scp, ssh-keygen, ssh-copy-id ssh**

In Linux, ssh is a protocol, which stands for Secure Shell or Secure SocketShell.

The secure shell is useful for security while connecting to a remote server. The **s**sh command uses a ssh protocol, which is a secure protocol, as the data transfer between the client and the host takes place in encrypted form. It transfers the input through the client to the host and returns the output transferred by the host. It executes through TCP/IP port 22. The encrypted connection is also used to run the commands on a Linux server, port forwarding, tunnelling, and more. Syntax: ssh user\_name@host(IP/Domain\_name) **scp** scp (secure copy) command in Linux system is used to copy file(s) between servers in a secure way. The SCP command or secure copy allows secure transferring of files in between the local host and the remote host or between two remote hosts. It uses the same authentication and security as it is used in the Secure Shell (SSH) protocol. SCP is known for its simplicity, security and preinstalled availability.

Syntax:scp [-346BCpqrTv] [-c cipher] [-F ssh\_config] [-i identity\_file] [-l limit] [-o ssh\_option] [-P port] [-S program] [[user@]host1:]file1 … [[user@]host2:]file2 **ssh-keygen**

SSH or Secure Shell is a useful encrypted protocol to secure connections between the client and the server for different administrative tasks. It supports various types of authentication systems. Public key-based authentication and passwordbased authentication are mostly used. Key-based authentication is more secure than password-based based authentication. Authentication key pairs for the SSH are generated by the ssh-keygen tool that can be used for different purposes such as authenticating the host, automating login, etc.



**ssh-copy-id**

The ssh-copy-id command is a simple tool that allows you to install an SSH key on a remote server’s authorized keys. This command facilitates SSH key login, which removes the need for a password for each login, thus ensuring a passwordless, automatic login process. The ssh-copy-id command is part of OpenSSH, a tool for performing remote system administrations using encrypted SSH connections.

Eg: sudo apt-get update && sudo apt-get install openssh-client

# OUTPUT

$ssh-copy-id

Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity\_file]] [-p port] [[-o <ssh -o options>] ...] [user@]hostname -f: force mode -- copy keys without trying to check if they are already installed -n: dry run -- no keys are actually copied -h|?: print this help

**3. 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.**

# File System Hierarchy

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.
  1. **/ (Root):** Primary hierarchy root and root directory of the entire file system hierarchy.
* Every single file and directory start from the root directory
* The only root user has the right to write under this directory
* /root is the root user’s home directory, which is not the same as /
  1. **/bin:** Essential command binaries that need to be available in single-user mode; for all users, e.g., cat, ls, cp.
* 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 e.g., ps, ls, ping, grep, cp
  1. **/boot :** Boot loader files, e.g., kernels, initrd.
* Kernel initrd, vmlinux, grub files are located under /boot
* Example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic
  1. **/dev :** Essential device files, e.g., /dev/null.

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

* Example: /dev/tty1, /dev/usbmon0
  1. **/etc :** Host-specific system-wide configuration files.
* Contains configuration files required by all programs.
* This also contains startup and shutdown shell scripts used to start/stop individual programs.
* Example: /etc/resolv.conf, /etc/logrotate.conf.
  1. **/home :** Users’ home directories, containing saved files, personal settings, etc.
* Home directories for all users to store their personal files.
* example: /home/kishlay, /home/kv
  1. **/lib :** Libraries essential for the binaries in /bin/ and /sbin/.
* Library filenames are either ld\* or lib\*.so.\*
* Example: ld-2.11.1.so, libncurses.so.5.7
  1. **/media :** Mount points for removable media such as CD-ROMs (appeared in FHS-2.3).
* Temporary mount directory for removable devices.
* Examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives;

/media/cdrecorder for CD writer

* 1. **/mnt :** Temporarily mounted filesystems.
* Temporary mount directory where sysadmins can mount filesystems.
  1. **/opt :** Optional application software packages.
* Contains add-on applications from individual vendors.
* Add-on applications should be installed under either /opt/ or /opt/ sub-directory.
  1. **/sbin :** Essential system binaries, e.g., fsck, init, route.
* Just like /bin, /sbin also contains binary executables.
* The Linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
* Example: iptables, reboot, fdisk, ifconfig, swapon
  1. **/srv :** Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.
* srv stands for service.
* Contains server specific services related data. ● Example, /srv/cvs contains CVS related data.
  1. **/tmp :** Temporary files. Often not preserved between system reboots, and may be severely size restricted.
* Directory that contains temporary files created by system and users.
* Files under this directory are deleted when system is rebooted.
  1. **/usr :** Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.
* 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
  1. **/proc :** Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount. Generally, automatically generated and populated by the system, on the fly.
* 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 **File permissions in Linux**

Every file and directory in Linux have the following three permissions for all the three kinds of owners: **Permissions for files**

* Read – Can view or copy file contents
* Write – Can modify file content
* Execute – Can run the file (if its executable)

**Permissions for directories**

* Read – Can list all files and copy the files from directory
* Write – Can add or delete files into directory (needs execute permission as well) ● Execute – Can enter the directory.

**Each letter denotes a particular permission:**

* r: Read permission
* w: Write permission
* x: Execute permission
* –: No permission set

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

# 1. Creating a shell script

Create a new directory bash\_scripts where you will run the shell scripting exercises.

mkdir bash\_scripts cd bash\_scripts

**Note**: It is recommended to create a new script file for every exercise; make them executable; give them names with extension .sh – it is just a convention rather than a must. The first line of a bash script starts with

**#!/bin/bash**

All non-executable comments in a script are prepended with #, # For example, we list all the files in the current director ls -la ls -l /etc # Comment. Here, we list files in /etc directory.

Script scr1.sh is like a calculator: #!/bin/bash echo "I will work out X\*Y" echo "Enter X" read X echo "Enter Y"

read Y

echo "X\*Y = $X\*$Y = $[X\*Y]" Make the script executable and run chmod 755 scr1.sh

./scr1.sh

# 2. if and case statements

1. #!/bin/bash

X=10 Y=5

if [ "$X" -gt "$Y" ]; then echo "$X is greater than $Y" elif [ "$X" -lt "$Y"]; then echo "$X is less than $Y" else

echo "$X is equal to $Y" fi

1. #!/bin/bash case $1 in

--test|-t)

echo "you used the --test option" exit 0

;;

--help|-h) echo "Usage:" echo " myprog.sh

[--test|--help|--version]" exit 0

;;

--version|-v)

echo "myprog.sh version 0.0.1" exit 0

;; \*)

echo "No such option $1" echo "Usage:" echo " myprog.sh

[--test|--help|--version]"

exit 1

;;

esac echo "You typed \"$1\" on the command-line"

**Note:**always watch for correct syntax of case statement.

**3)** case string in

regex1) commands1

;;

regex2)

commands2

;; ........

esac

Where regex is a regular expression to match the string. To catch all remaining strings, use \*) at the end.

# 3. Looping with while and until statements

**1) Script scr2.sh:**

#!/bin/bash N=1

while [ "$N" -le "10" ] do

echo "Number $N" N=$[N+1] done

**2)Script scr3.sh**

#!/bin/bash N=1

until [ "$N" -gt "10" ] do

echo "Number $N"; N=$[N+1]

done

**Note:** common mistakes in shell scripting are usually due to incorrect syntax.

For example, there should be no spaces before and after operator “=” .

N=1 # correct

N =1 # error

N= 1 # error

N=$[N+1] # correct

N =$[N+1] # error

N= $[N+1] # error

# 4. Looping with for statement

**1)Script scr4.sh**  #!/bin/bash for i in red white blue do

echo "$i is a color" done

**2)Script backup-lots.sh**

#!/bin/bash

for i in 0 1 2 3 4 5 6 7 8 9 ;do

cp $1 $1.BAK-$i

done

Now create a file important\_data with some numbers in it and then run

./backup-lots.sh important\_data which will copy the file 10 times with 10 different extensions. As you can see, the variable $1 has a special meaning – it is the first argument on the command-line.

**Note:** watch for correct syntax:

for i in 0 1 2 3 4 5 6 7 8 9 do .... done

The continue statement is useful for terminating the current iteration of the loop. **3)** #!/bin/bash for i in 0 1 2 3 4 5 6 7 8 9 ; do

NEW\_FILE=$1.BAK-$i

if [ -e $NEW\_FILE ] ; then

echo "backup-lots.sh: \*\*warning\*\* $NEW\_FILE"

echo " already exists - skipping" continue

fi

cp $1 $NEW\_FILE done

# 5. Functions

Function definitions provide a way to group statement blocks into one.

**1)**

#!/bin/bash

function usage ()

{

echo "Usage:"

echo " myprog.sh [--test|--help|--version]"

} case $1 in

--test|-t)

echo "you used the --test option" exit 0

;;

--help|-h) usage

;;

--version|-v)

echo "myprog.sh version 0.0.2"

exit 0

;;

-\*)

echo "Error: no such option $1" usage exit 1

;;

esac echo "You typed \"$1\" on the command-line" **Note:** watch for syntax:

**2)**

function usage ()

{

command1

command2; command3

...... }

The word function in a function is optional.

That is, the following will work as well:

**3)**  usage ()

{

command1

command2; command3

.....

}

# 6. Using quotes

Single forward quotes ' protect the enclosed text from the shell.

**1)** echo 'error $?' echo 'shell name $0'

Double quotes " allow all shell interpretations to take place inside them.

**2)**

echo "error $?" #gives the error code of the last command echo "shell name $0" #gives the current shell name

Command substitution

**3)**

X=`expr 100 + 50 '\*' 3`

echo $X

Assigning command output to a variable:

**4)**

FSIZE=`wc -l /etc/profile`

same as

**5)**

FSIZE=$(wc -l /etc/profile)

# 7. Introduction to awk

The basic function of awk is to search files for lines or other text units containing one or more patterns. When a line matches one of the patterns, special actions are performed on that line. Display user names from /etc/passwd (field 1):

**1)**

awk -F: '{ print $1 }' /etc/passwd

Where F is the field separator in the passwd file. The fields are separated by : Default field separator is a blank space. Awk scans the input file and splits each input line into fields.

**2)**

cat /etc/passwd | awk -F: '{ print $1 }'

Display user names home directories and login shell (fields 1 and 7), and store them in a separate file, users.txt

**3)** awk -F: '{ print $1, $6, $7 }' /etc/passwd > users.txt or

cat /etc/passwd | awk -F: '{ print $1, $6, $7 }' > users.txt `

Default field separator is empty space. To print users (field 1) from just created file users.txt: awk '{ print $1 }' users.txt

# 8.Introduction to sed

String editor, sed, is used for editing lines in a file or a stream; output is going to the standard output and can be re-directed to a new file.

**Syntax:** sed [options] 'command1' [files] sed [options] -e 'command1' [-e command2 ...] [files] sed [options] -f script [files]

**Delete lines from 3 through 5 in file list.txt:** sed '3,5d' list.txt

**Delete lines that contain “O” at the beginning of the line:**

sed '/^O/d' list.txt

**Translate capital C,R,O into small c,r,o:**

sed 'y/CRO/cro/' list.txt **Delete empty lines:** sed '/^$/d' list.txt

**Replace string Oop with Wee for the first occurence on a line** sed 's/Oop/Wee/' lsst.txt

**Remove ss string (replace with empty entry)for the first occurence on a line:** sed 's/ss//' list.txt

**Remove ss string for all occurrences on a line:** sed 's/ss//g' list.txt

**Substitute a single space for any number of spaces wherever they occur on the line:**

sed 's/ \*/ /g' list.txt

**Substitute underscore for any number of spaces wherever they occur on the line:**

sed 's/ \*/\_/g' list.tx

**5. Installation and configuration of LAMP stack. Deploy an open-source application such as phpmyadmin and Wordpress.**

# INSTALLING LAMP

**Procedure:**

**Step 1: Update your system** sudo apt update sudo apt upgrade **Step 2: Install Apache** sudo apt install apache2 -y **To check whether installed or not:**

Open web browser and search localhost in address bar then apache2 default page is displayed .Successfully installed. **Step 3: Firewall setting** sudo ufw status if inactive the sudo ufw enable sudo ufw app list

sudo ufw allow in “Apache Full” clear

**Step 4: Installing MySQL** sudo apt install mysql-server -y or

sudo apt install mariadb-server mariadb-client **Check installation**:

sudo systemctl status mysql sudo mysql\_secure\_installation sudo mysql show databases; create database name;

**Step 5:Installing php** sudo apt install php -y php version : php -v cd /var/www/html sudo gedit filename.php

Open web browser and search localhost/filename.php in address bar **Step 6:Install phpMyAdmin** sudo apt install php-mbstring php-zip php-gd php-curl php-json sudo apt install phpmyadmin -y

**Press space and enter Set password**

sudo systemctl restart apache2

Open browser and type localhost/phpmyadmin

Type username phpadmin and password

# Step 7: Host WORDPRESS

1)Open browser and go to [**https://wordpress.org/latest.zip**](https://wordpress.org/latest.zip)

2)Copy the link address and open terminal

3)Install wget (if not installed) -> sudo apt install wget wget (paste the copied link address)

4)Install unzip (if not installed) -> sudo apt install unzip

5)Go to wordpress directory cd wordpress/

6)Copy all the files in wordpress recursively sudo cp –r \* /var/www/html cd /var/www/html

7)Remove one file name “index.html” sudo rm –rf index.html

8)Change permission(ownership) of the files sudo chmod –R www-data:www-data /var/www/

9)Open browser and type localhost

10)WordPress window will be opened -> successful installation

11)Select English Language and continue 12)Set database Connection details :

13)Open terminal sudo mysql –u root –p (Enter password of database)

⮚ create database [databasename];

⮚ create user “[username]” identified by “[password]”;

⮚ grant all privileges on [databasename].\* to “[username]”;

⮚ exit

Database Name : [databasename]

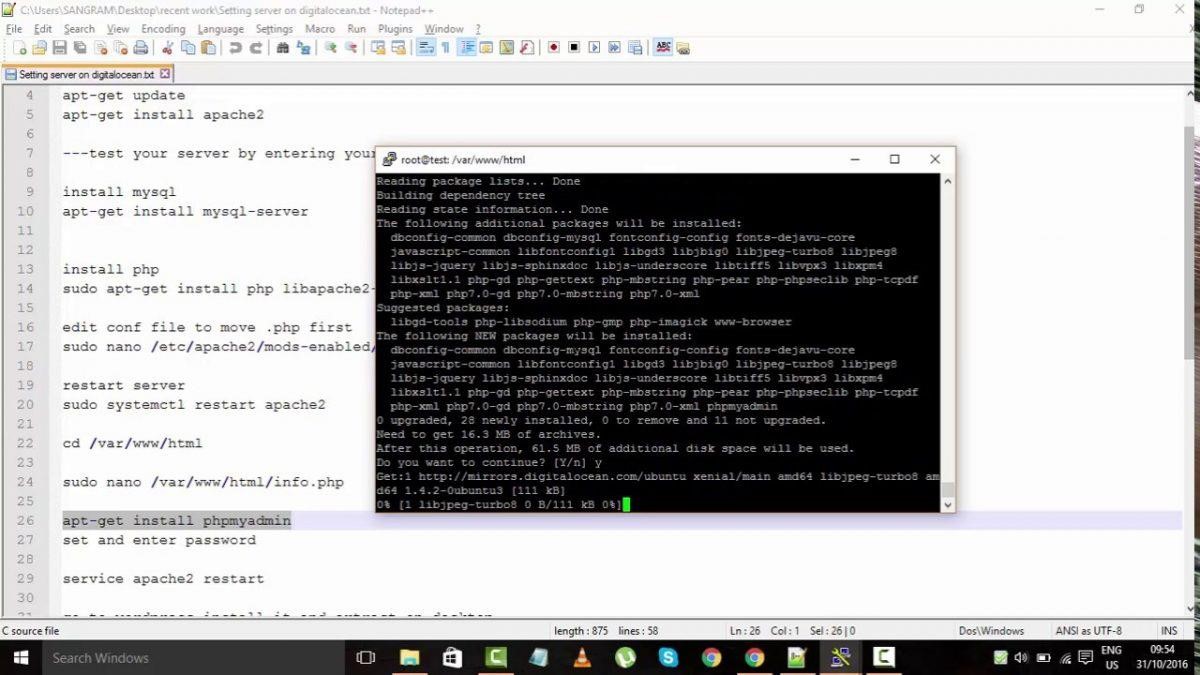
Username : [username]

Password : [password]

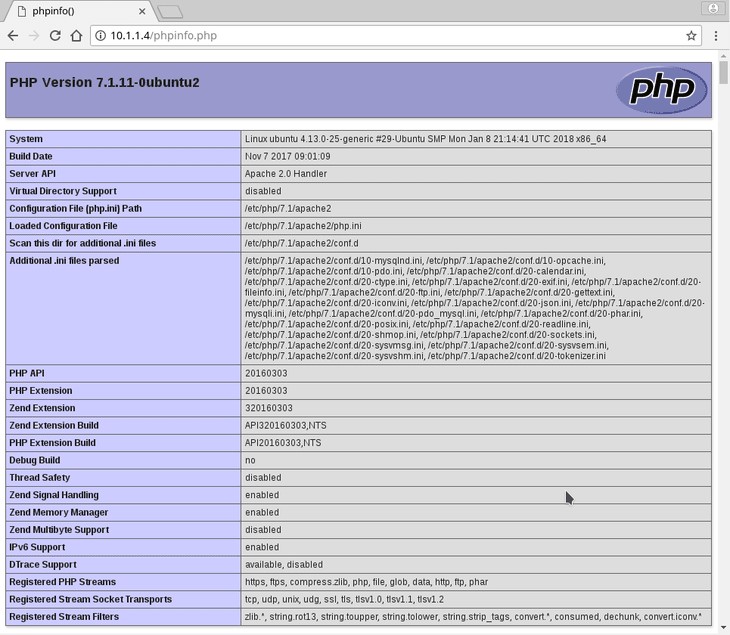
Database Host : localhost

Table Prefix : wp\_

# OUTPUT



# PHPMYADMIN



**6. Installation and configuration of common software frame works such as Laravel. Installing Laravel Procedure:**

# Step 1: Install Apache web server To install apache2, type: sudo apt install apache2

**Once installed, Apache should be running. If it's not, for whatever reason, start it:**

sudo systemctl start apache2

**Then enable it to start on boot time.**

sudo systemctl enable apache2

**To verify the status of Apache, execute:**

sudo systemctl status apache2

**Step 2: Install PHP and additional PHP extensions** Laravel 8 requires PHP 7.3 or above.

PHP 7.4 is available in Ubuntu repositories. So, install PHP and the following PHP extensions.

sudo apt install php libapache2-mod-php php-mbstring php-cli php-bcmath phpjson php-xml php-

**When the installation is complete, verify the PHP version.**  php –v

# Step 3: Create Database for Laravel Application

Next up, we will create a database for the Laravel application. But first, we need to install a database server. Laravel supported database systems are MariaDB, MySQL, SQLite, Postgres, or SQL Server.

**We will go with the MariaDB database engine.**

sudo apt install mariadb-server

**Once the database server is installed, log into the MariaDB prompt:**

sudo mysql -u root -p

**Once logged in create the database, database user, and grant all privileges to the database user.**

CREATE DATABASE laravel\_db;CREATE USER 'laravel\_user'@'localhost'

IDENTIFIED BY 'secretpassword';GRANT ALL ON laravel\_db.\* TO

'laravel\_user'@'localhost';FLUSH PRIVILEGES;QUIT;

# Step 4: Install Composer

Composer is a dependency package manager for PHP. It provides a framework for managing libraries and dependencies and required dependencies. To use Laravel, first install composer.

**To download Composer, invoke the command shown.**

curl -sS https://getcomposer.org/installer | php

**Next, move the composer file to the /usr/local/bin path.**

sudo mv composer.phar /usr/local/bin/composer **Assign execute permission:**

sudo chmod +x /usr/local/bin/composer **Verify the Composer version installed:**

composer -–version

**Composer version 2.1.3 is installed.**

# Step 5: Install Laravel 8 on Ubuntu

With Composer installed, the next course of action is to install Laravel.

**Navigate to the webroot directory, type:**

cd /var/www/html

**Now, install Laravel using the composer command, type:**

sudo composer create-project laravel/laravel laravelapp

The command creates a new directory called laravelapp and installs all the files and directories for Laravel.

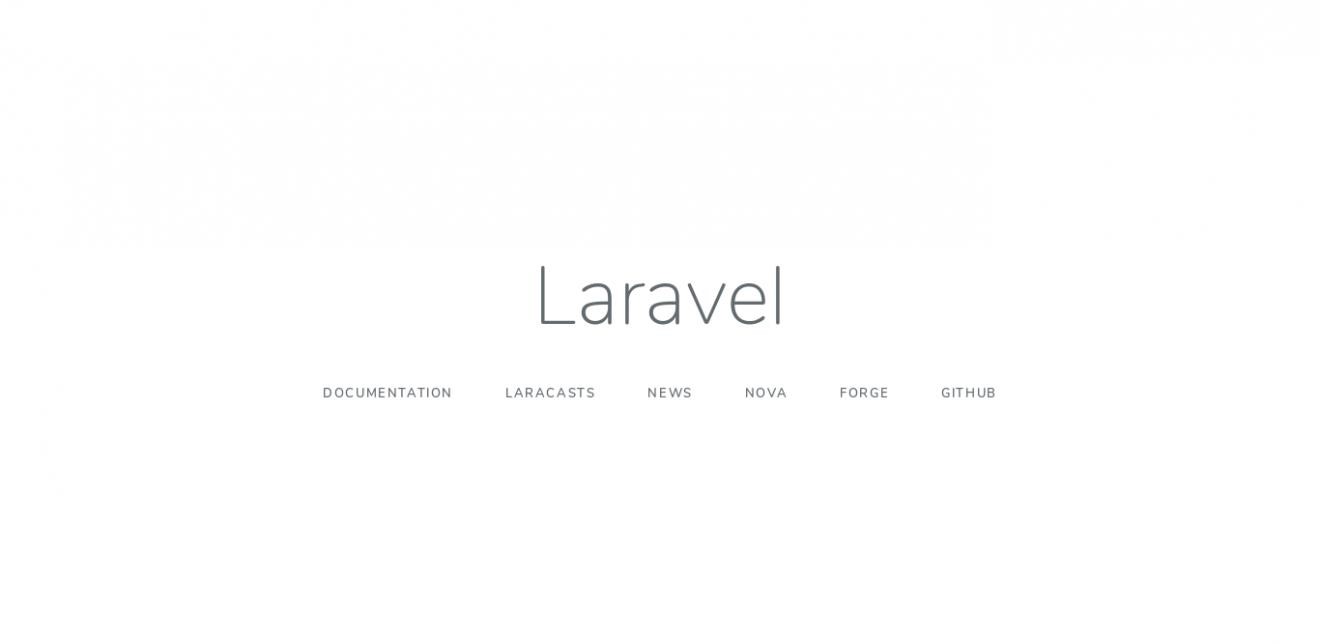
# Step 6: Configure Apache to serve Laravel site

**Lastly, we need to set up the Apache webserver to host the Laravel site. For that to happen, we need to create a virtual host file.**

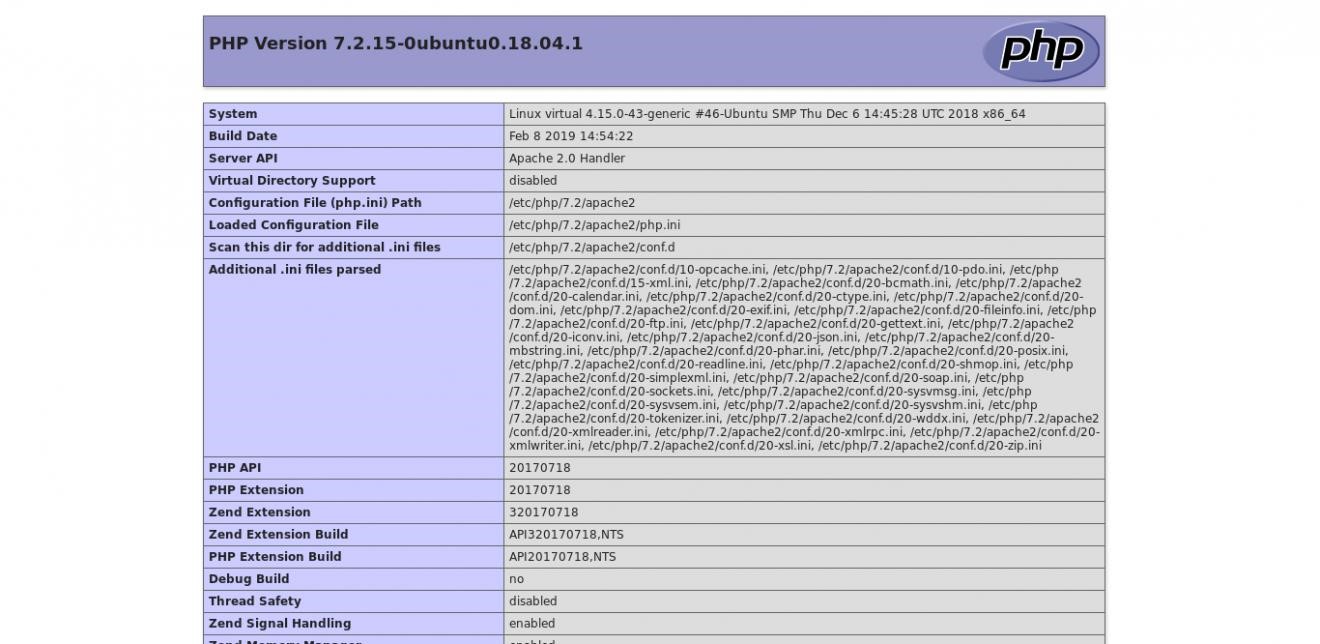
sudo vim /etc/apache2/sites-available/laravel.conf

**Step 7: Access Laravel from a browser** sudo apt Laravel

# OUTPUT



# PHPMYADMIN



**7. Build and install software from source code, familiarity with make and cmake utilities expected.**

**Procedure:**

**Step 1: Set Up the Repository** sudo apt install cmake

**Step 2: Install following packages** sudo apt install \ clang-tools lld llvm-dev libclang-dev liblld-10-dev \ libpng-dev libjpeg-dev libgl-dev \ python3-dev python3-numpy python3-scipy python3-imageio python3-pybind11

\libopenblas-dev libeigen3-dev libatlas-base-dev \ doxygen ninja-build

**Step 3: Building halide with cmake**

Halide$ cmake -G Ninja -DCMAKE\_BUILD\_TYPE=Release -S . -B build dev@host:~/Halide$ cmake --build ./build

**Step 4: CMake Presets**

If you are using CMake 3.19+, we provide several [presets](https://cmake.org/cmake/help/latest/manual/cmake-presets.7.html) to make the above commands more convenient. The following CMake preset commands correspond to the longer ones above.

> cmake --preset=msvc-release # Ninja generator, MSVC compiler, Release build

> cmake --preset=win64 # VS 2019 generator, 64-bit build

> cmake --preset=win32 # VS 2019 generator, 32-bit build

$ cmake --preset=gcc-release # Ninja generator, GCC compiler, Release build $ cmake --list-presets # Get full list of presets.

**Step 5: Installing**

Once built, Halide will need to be installed somewhere before using it in a separate project. On any platform, this means running the [cmake -install](https://cmake.org/cmake/help/latest/manual/cmake.1.html%22%20/l%20%22install-a-project) command in one of two ways. For a single-configuration generator (like Ninja), run either:

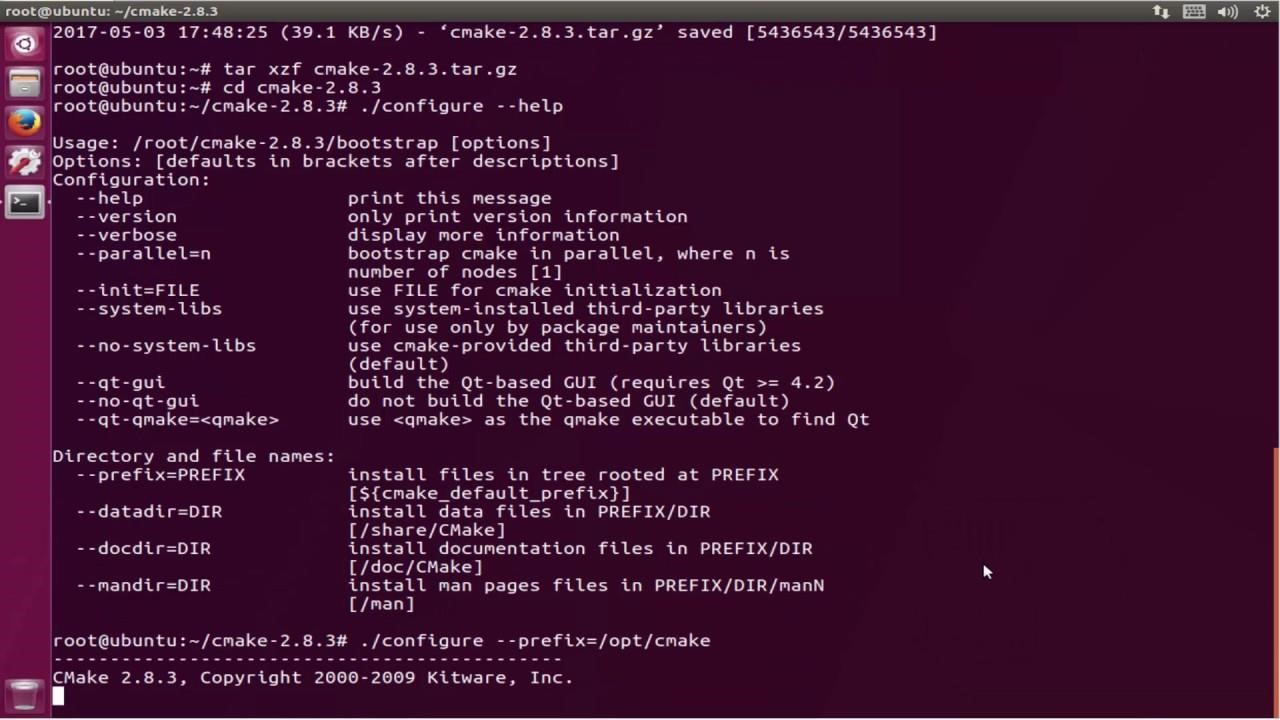
dev@host:~/Halide$ cmake --install ./build --prefix /path/to/Halide-install

> cmake --install .\build --prefix X:\path\to\Halide-install **For a multi-configuration generator (like Visual Studio) run**:

dev@host:~/Halide$ cmake --install ./build --prefix /path/to/Halide-install -config Release

> cmake --install .\build --prefix X:\path\to\Halide-install --config Release

# OUTPUT



**8. 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.**

**Set up repository**

# 1.Ping IP address

Eg: ping 8.8.8.8 **2**.**nslookup**  nslookup facebook.com or nslookup 157.240.23.35 **3.To install traceroute** sudo apt install traceroute

**4. To find how many hopes** traceroute 172.16.13.163 or traceroute 8.8.8 -m 30 **5.To identify ip version** ip -v



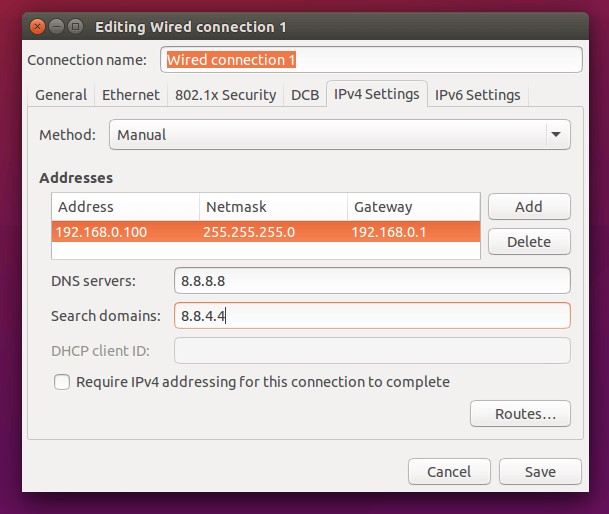
# 6. To get ip address ip addr

**IPv4 networking for Setting up static and dynamic IP addresses.**

❖ **For setting up ip address as static: Step 1: Update the terminal** sudo apt-get update

# Step 2: To show ip address ip address Step 3: To add new ip address

Go to settings-> network->add ip address manually->save Open terminal and type, ifconfig



**Step 4: To see new ip address, refresh network connection and again give command:**

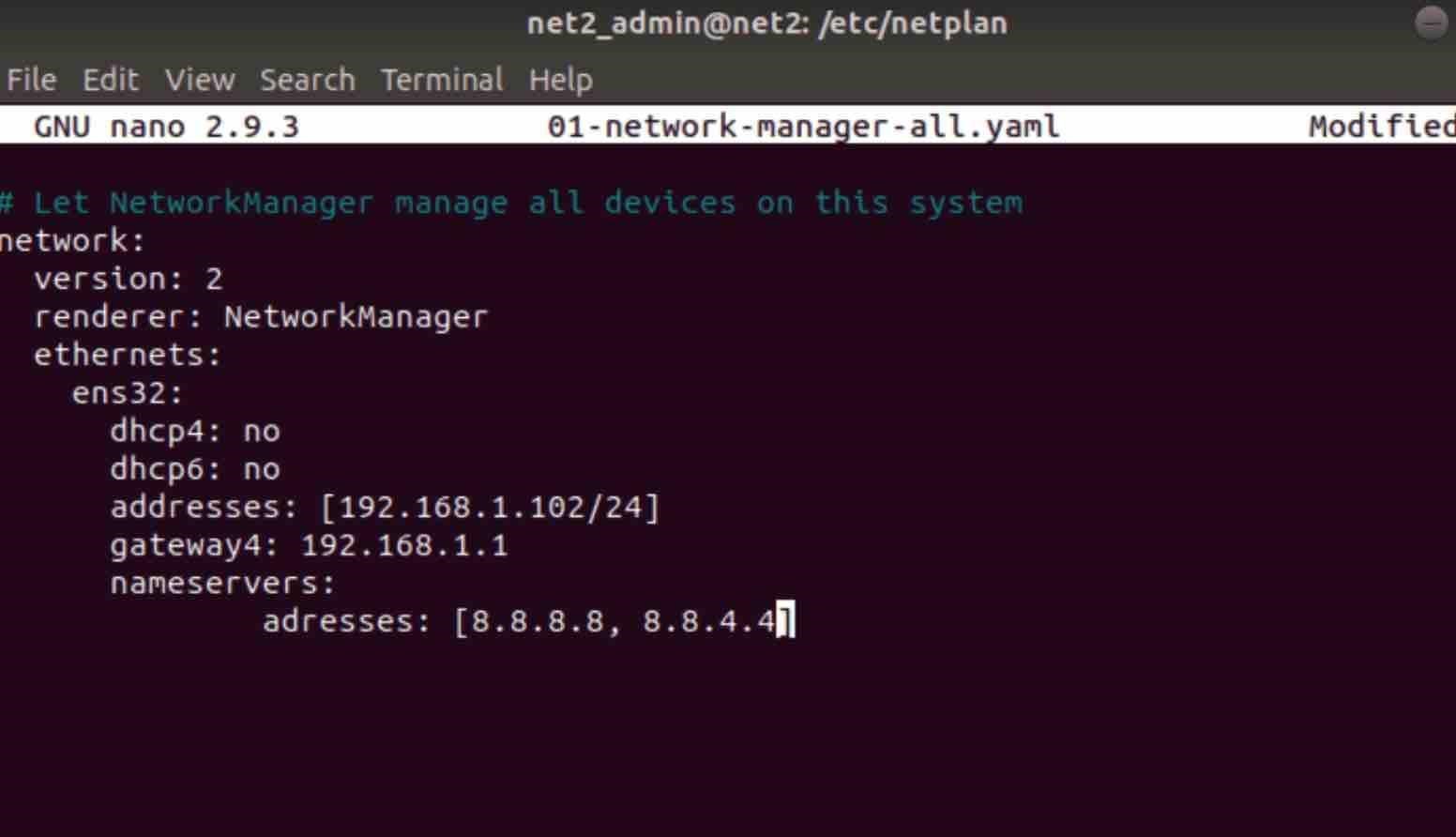
ip address

❖ **For setting up ip address as dynamic: Step 1: To show device name** nmcli connection add con -name “dyn” **Step 2: To show ip address** ip address **Step 3: To connect**

nmcli connection add con-name “dyn” ifname emp4s0 autoconnect yes type ethernet

**Step 4: To show the connection** nmcli connection show **Step 5: To down the connection** nmcli connection down docker 0 **Step 6: To establish the connection** nmcli connection up dyn

# OUTPUT



**9. Analysing network packet stream using tcpdump and wireshark. Perform basic network service tests using nc.**

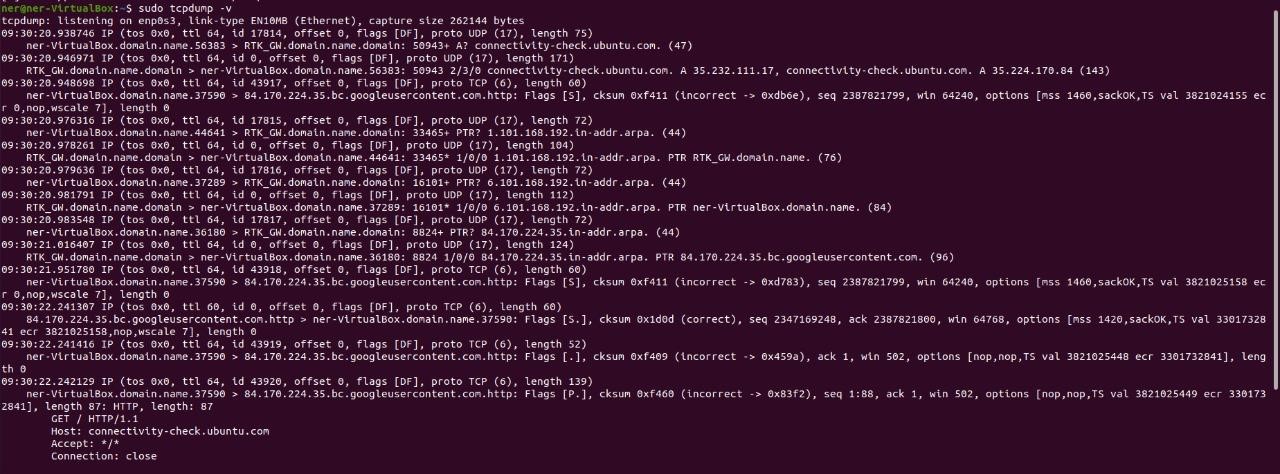
Installing tcpdump and analysing network packet stream **Procedure:**

# Step 1: Update the system sudo apt-get update Step 2: Install tcpdump on the system sudo apt-get install tcpdump Step 3: Check the version tcpdump --version

**To capture packets from a source ip.**

tcpdump –n src host ip-address

# OUTPUT



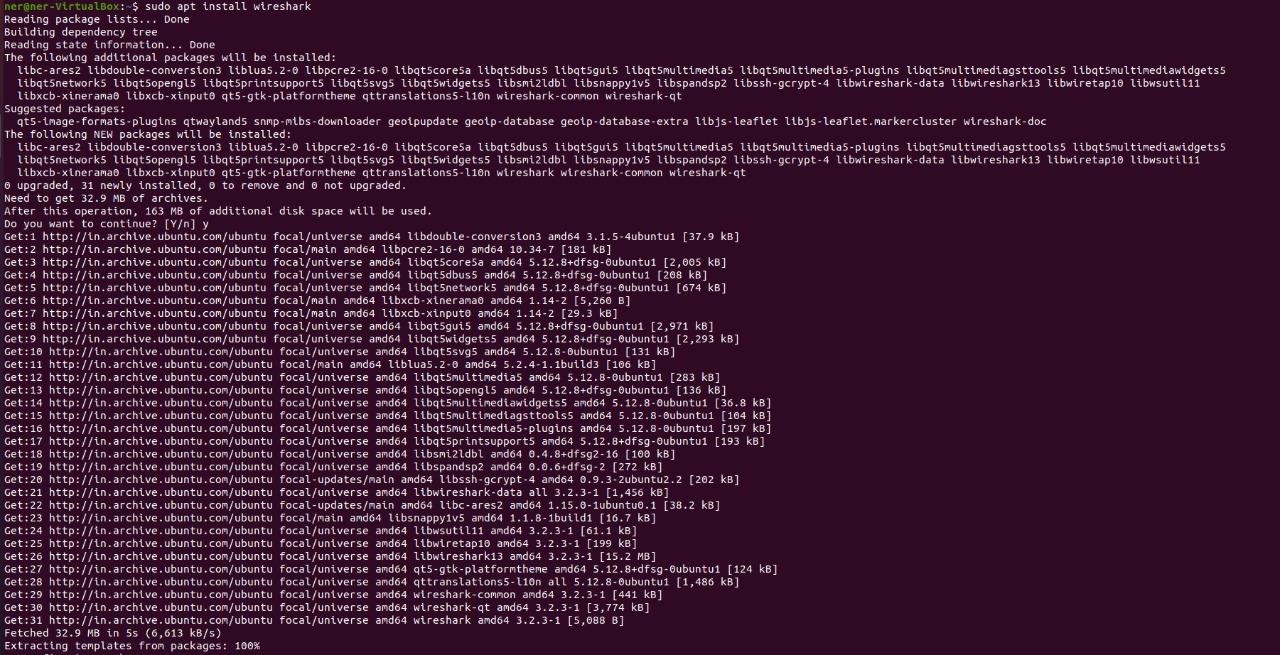
# Installing wireshark and analysing network packet stream

[Wireshark](https://www.wireshark.org/) is a free and open-source network protocol analyser widely used around the globe. With Wireshark, you can capture incoming and outgoing packets of a network in real-time and use it for network troubleshooting, packet analysis, software and communication protocol development, and many more.

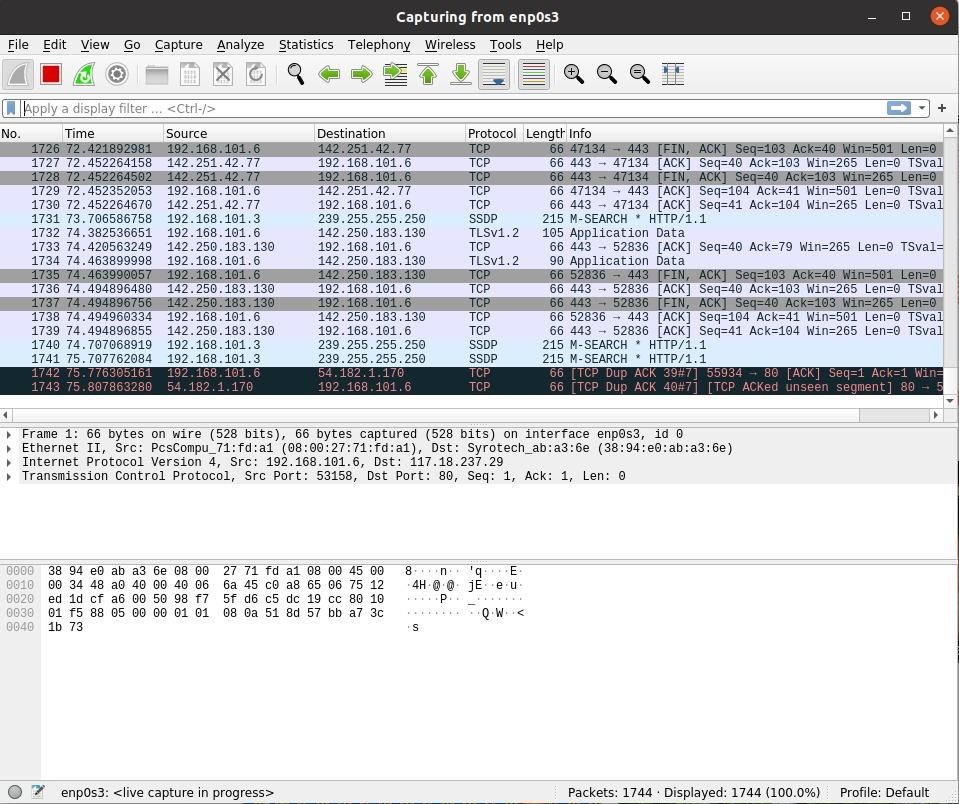
**Procedure:**

**Step 1: Update the system** sudo apt-get update **Step 2: Install wireshark on the system** sudo apt-get install wireshark **Step 3: Check the version** wireshark –version

# OUTPUT



**To capture network packet streams type,** wireshark then copy the ip address



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

# Installation and Deployment of Hypervisor (Type 2)

A [type 2 hypervisor](https://phoenixnap.com/kb/what-is-hypervisor-type-1-2#htoc-type-2-hypervisor) enables users to run isolated instances of other operating systems inside a host system. As a Linux based OS, Ubuntu supports a wide range of virtualization solutions.

Aside from popular third-party apps, such as [VirtualBox](https://phoenixnap.com/kb/install-virtualbox-on-ubuntu) and VMWare, the [Linux kernel](https://phoenixnap.com/kb/linux-kernel-5-7-features) has its own virtualization module called KVM (Kernel-based Virtual Machine).

**Procedure:**

**Step 1: Install KVM Packages**

1. First, update the repositories:

sudo apt update

1. Then, install essential KVM packages with the following command:

sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils

**Step 2: Authorize Users**

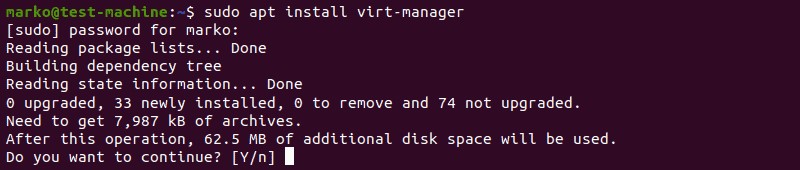
1. Only members of the **libvirt** and **kvm** user groups can run virtual machines. Add a user to the libvirt group by typing: sudo adduser ‘username’ libvirt

Replace username with the actual username.

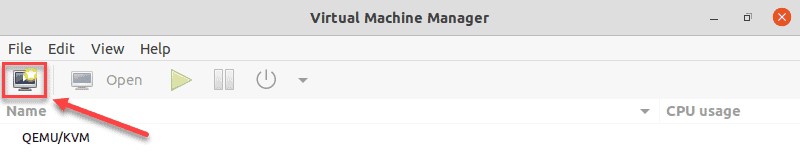
**Step 3: Install Virtual Manager**

1.Type the command in the terminal sudo apt install virt-manager

1. Type Y and press ENTER. Wait for the installation to finish



**Step 4: Check if it is working....**  sudo virt-manager



**Installation and Deployment of Docker Procedure:**

# 1)Set up the repository

**Step 1: Update the apt package index and install packages to allow apt to use a repository over HTTPS:**

sudo apt upgrade

sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \

gnupg \ lsb-release

**Step 2: Add Docker’s official GPG key:** curl -fsSL <https://download.docker.com/linux/ubuntu/gpg>| sudo apt-key add **Step 3: Use the following command to set up the stable repository** echo \

> "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] <https://download.docker.com/linux/ubuntu>\

> $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 2)Install Docker Engine

**Step 1:Update the apt package index, and install the *latest version* of Docker Engine and container, or go to the next step to install a specific version:**

sudo apt-get update **Step 2: Install docker** sudo apt-get install docker-ce docker-ce-cli containerd.io **Step 3: Check that whether it is running** sudo systemctl status docker

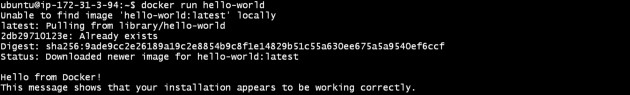
**Step 4: To view different docker commands** docker

**Step 5: Docker information** sudo docker info

**Step 6: Verify that Docker Engine is installed correctly by running the helloworld image.**

sudo docker run hello-world

# OUTPUT



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

Ansible is an open-source automation tool for provisioning, application deployment ([WordPress deployment](https://spinupwp.com/wordpress-deployment-workflow-preparing/) in this case), and configuration management. Gone are the days of SSH’ing into your server to run a command or hacking together bash scripts to semi-automate laborious tasks. Whether you’re managing a single server or an entire fleet, Ansible can not only simplify the process but save you time. So, what makes Ansible so great?

Ansible is completely agent-less, meaning you don’t have to install any software on your managed hosts. All commands are run through Ansible via SSH and if Ansible needs updating you only need to update your single control machine and not any remote machines. The only prerequisite to running Ansible commands is to have Python installed on your control machine.

**Procedure:**

# Installation

**Step 1: First, ensure that pip is installed.**

sudo easy\_install pip **Step 2: Then install Ansible.**

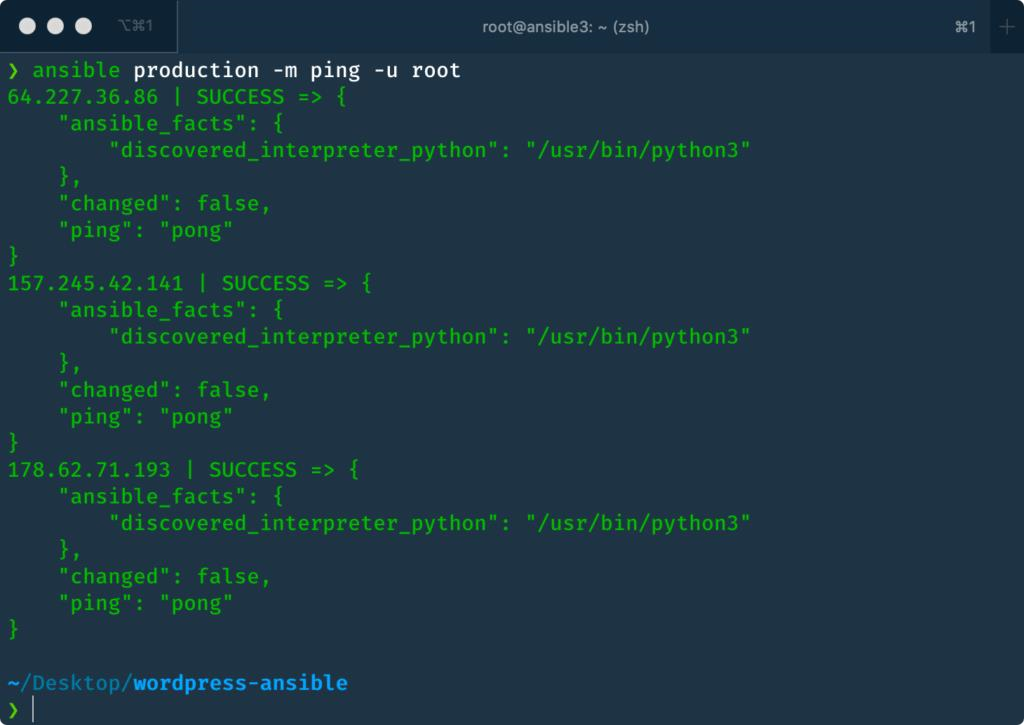
sudo pip install ansible

**Step 3: Once the installation has completed you can verify that everything installed correctly by issuing:** ansible --version

**Step 4: If you were installing Ansible on Ubuntu the commands would be:** sudo apt update sudo apt install software-properties-common sudo apt-add-repository --yes --update ppa:ansible/ansible sudo apt install ansible

# Running Commands

ansible production -m ping -u root



# Playbooks

[Playbooks](https://docs.ansible.com/ansible/latest/user_guide/playbooks.html) allow you to chain commands together, essentially creating a blueprint or set of procedural instructions. Ansible will execute the playbook in sequence and ensure the state of each command is as desired before moving onto the next. This is what makes Ansible idempotent. If you cancel the playbook execution partway through and restart it later, only the commands that haven’t completed previously will execute. The rest will be skipped.

Playbooks allow you to create truly complex instructions, but if you’re not careful they can quickly become unwieldy (think of god classes in OOP), which brings us onto roles.

Roles add organization to playbooks. They allow you to split your complex build instructions into smaller reusable chunks, very much like a function in programming terms. This makes it possible to share your roles across different playbooks, without duplicating code. For example, you may have a role for installing Nginx and configuring sensible defaults, which can be used across multiple hosting environments.

**Organization of Playbook** ansible.cfg hosts provision.yml roles nginx handlers main.yml tasks main.yml

--- - hosts: production user: root vars:

username: ashley password:

$6$rlLdG6wd1CT8v7i$7psP8l26lmaPhT3cigoYYXhjG28CtD1ifILq9KzvA0W0

TH2Hj4.iO43RkPWgJGIi60Mz0CsxWbRVBSQkAY95W0

public\_key: ~/.ssh/id\_rsa.pub roles:

common

* ufw
* user
* nginx
* php
* mariadb
* wp

-cli

* ssh

# OUTPUT

