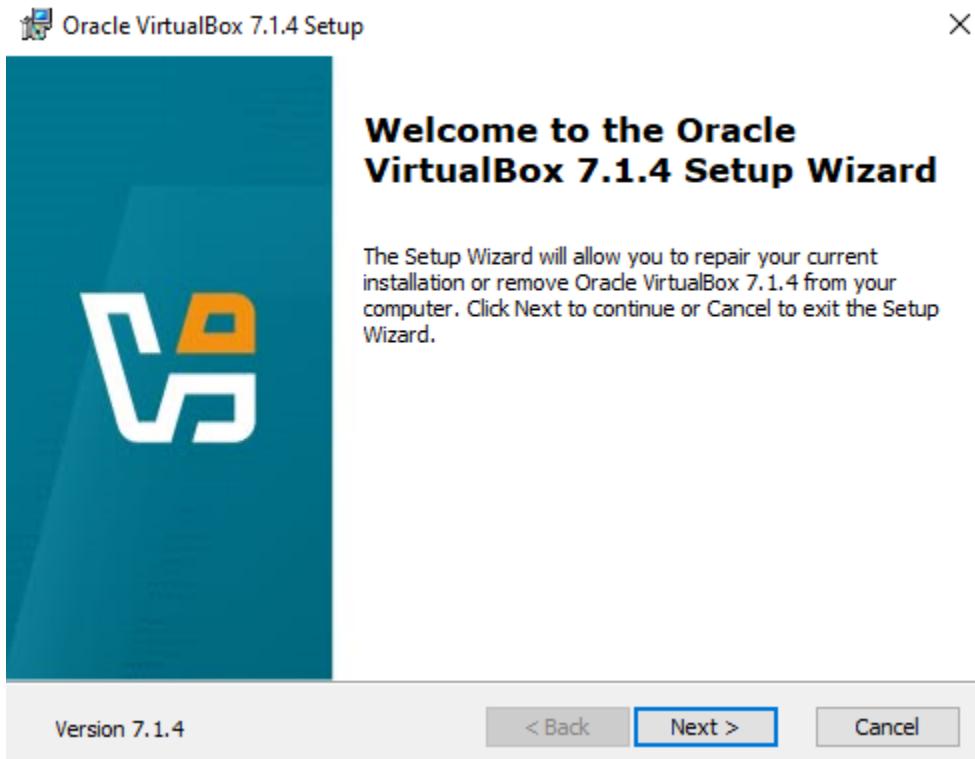


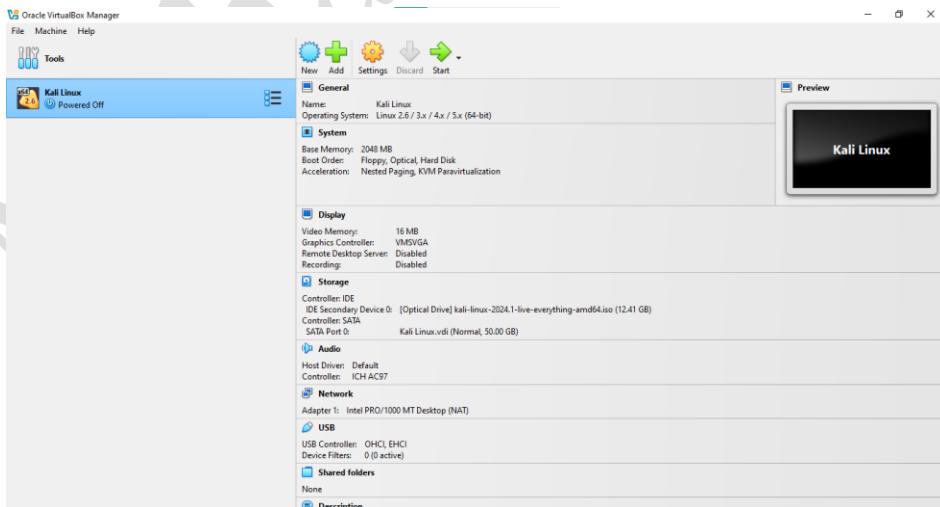
# CYBER SECURITY

## Practical Work

### Installation of Virtual Box:

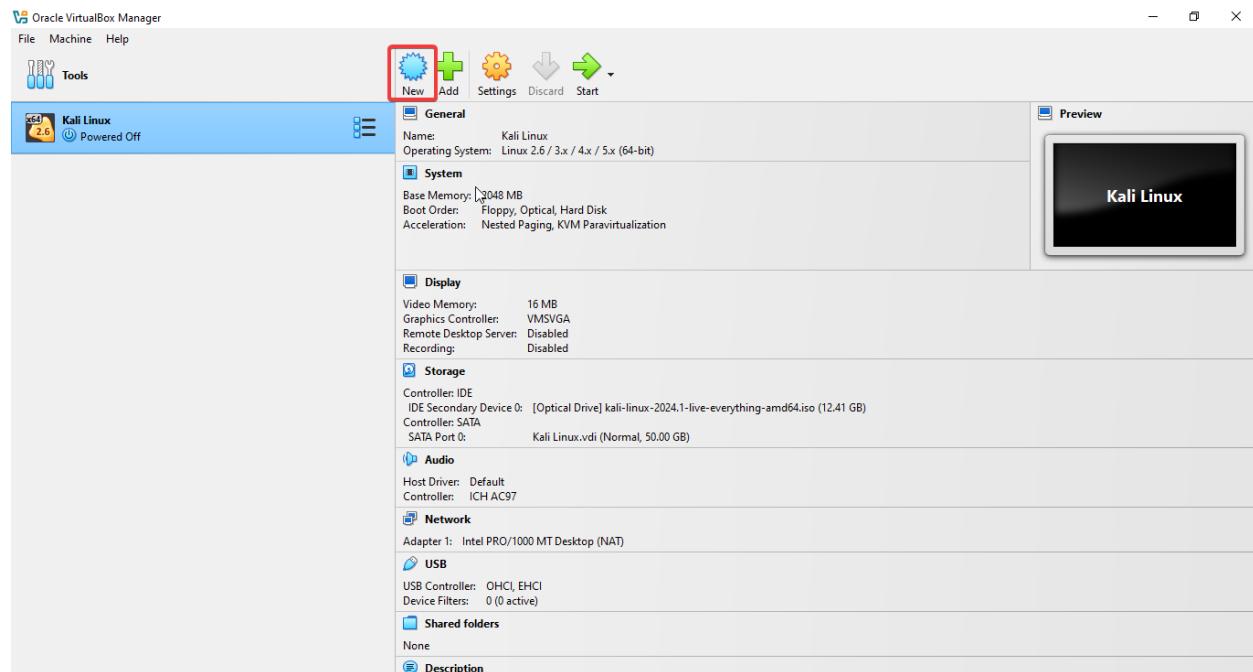


### After Installation:

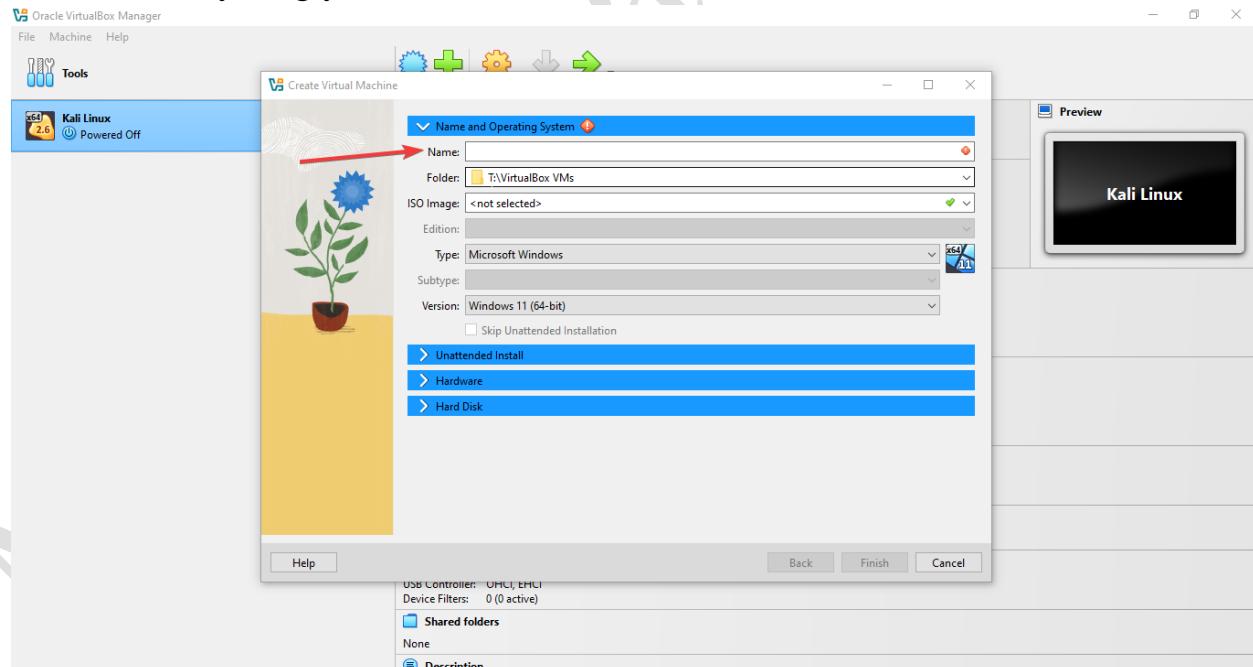


# Installing Kali Linux:

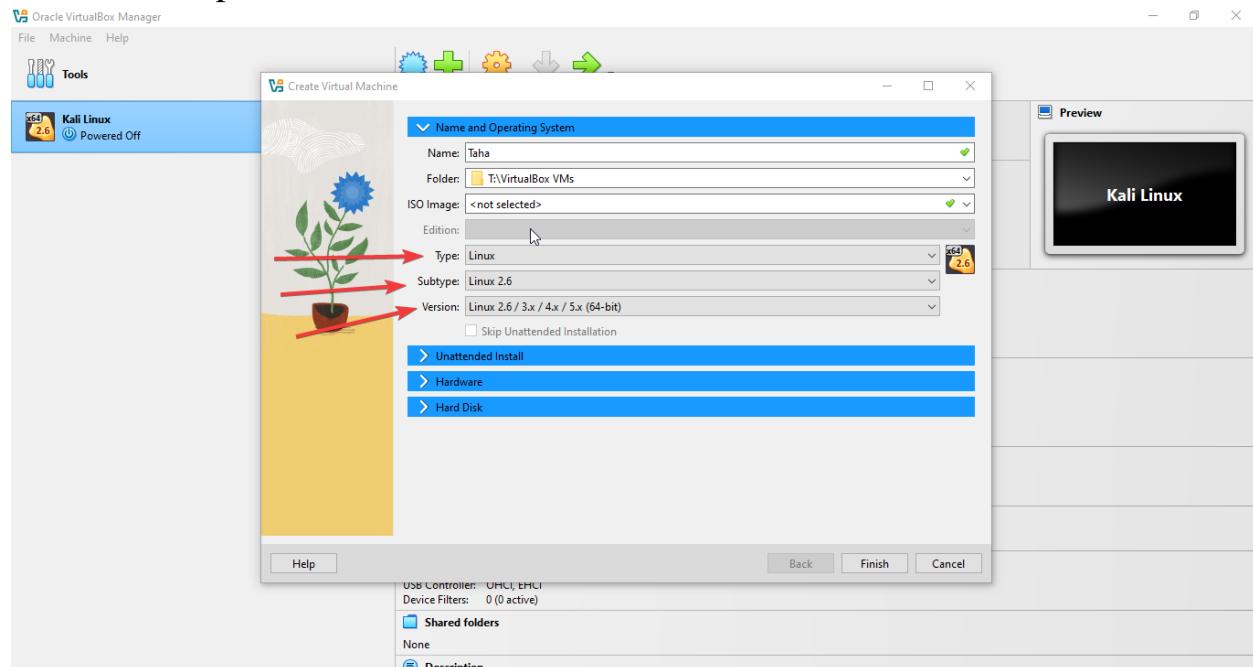
Click the new Button



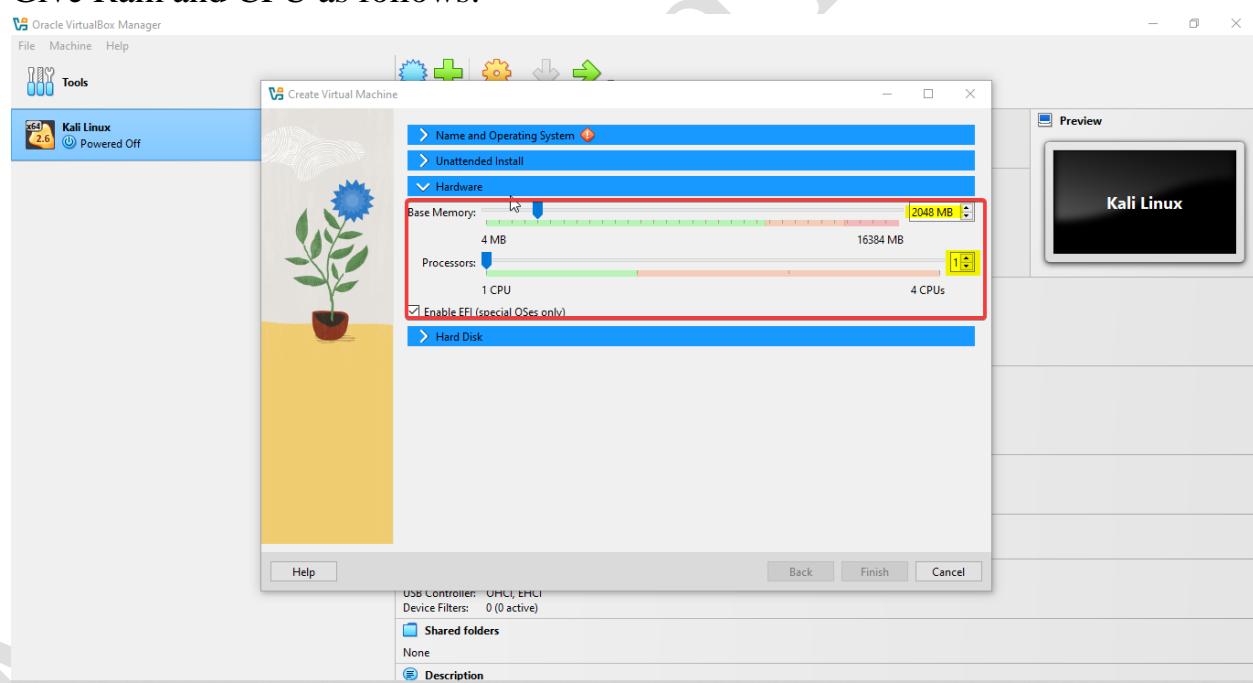
Give name anything you want:



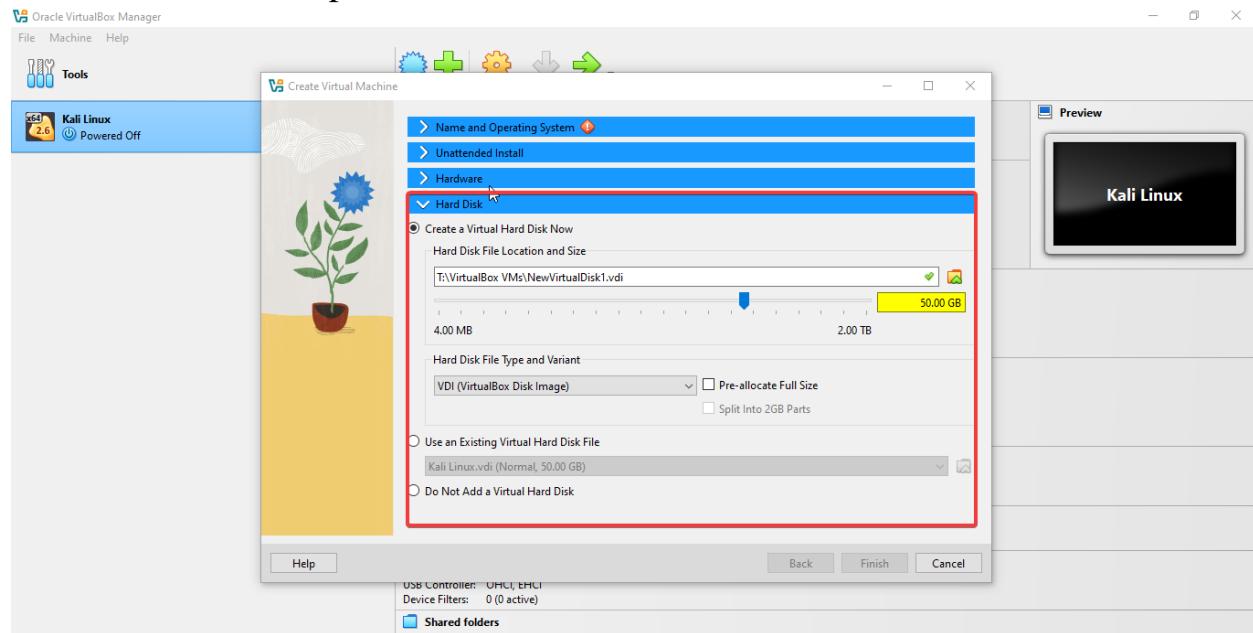
Select these options:



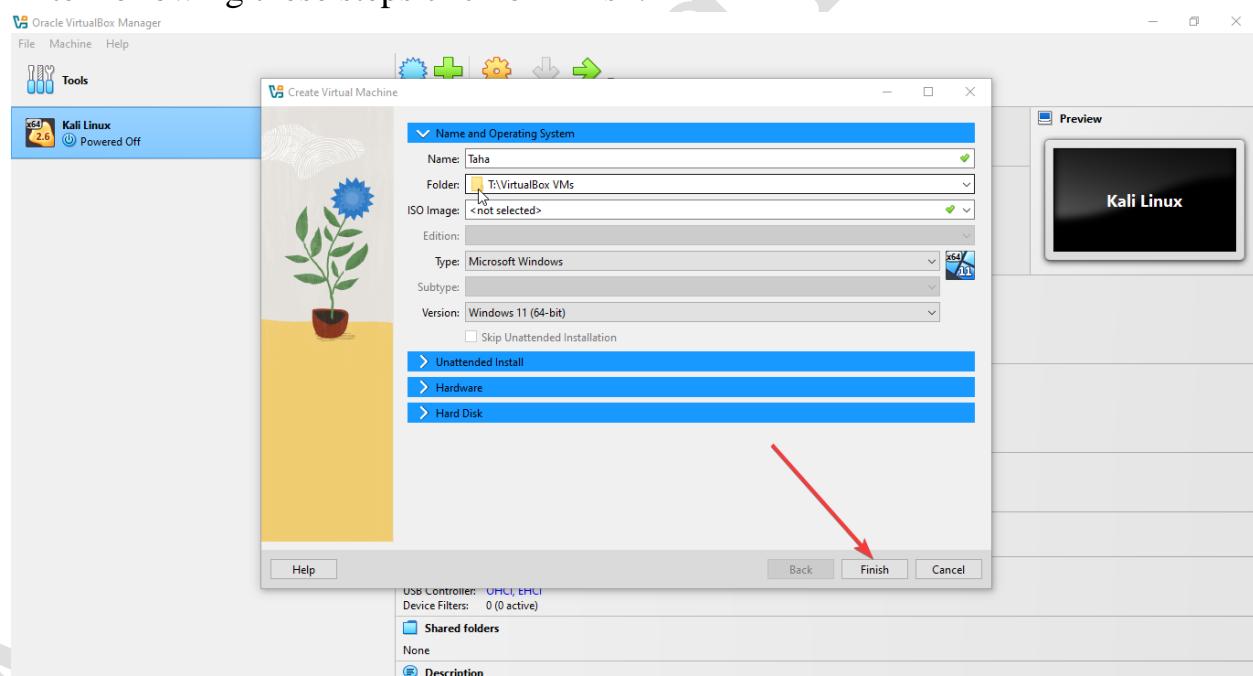
Give Ram and CPU as follows:



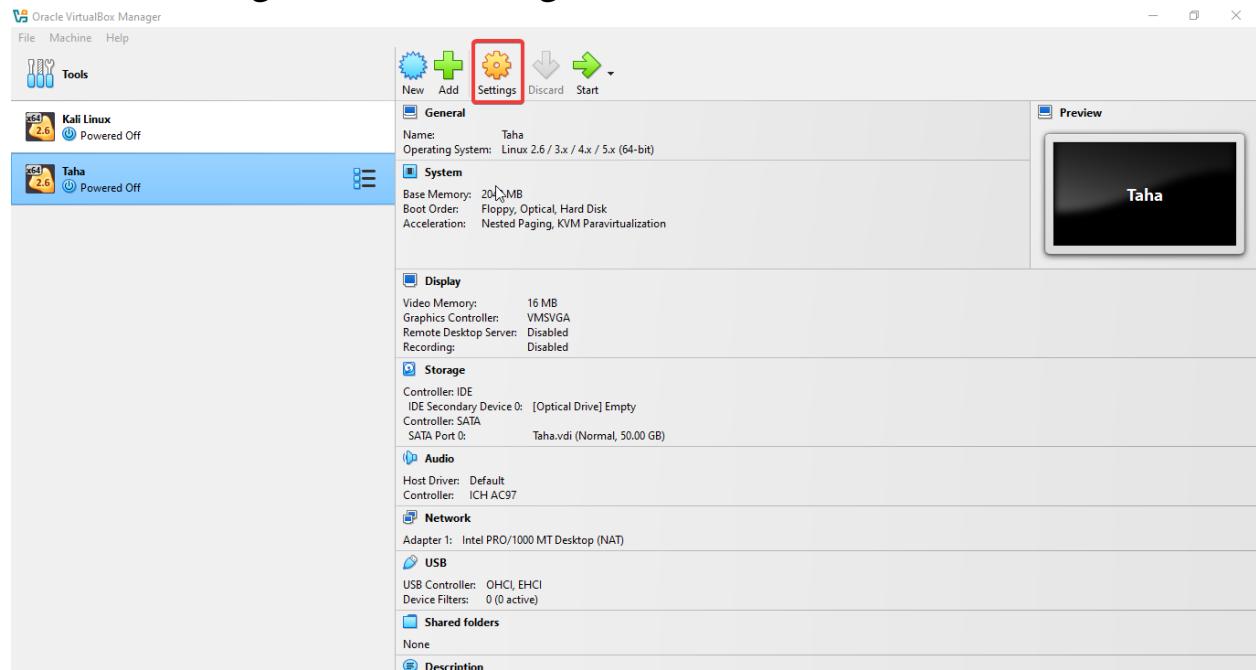
Allocate This much space:



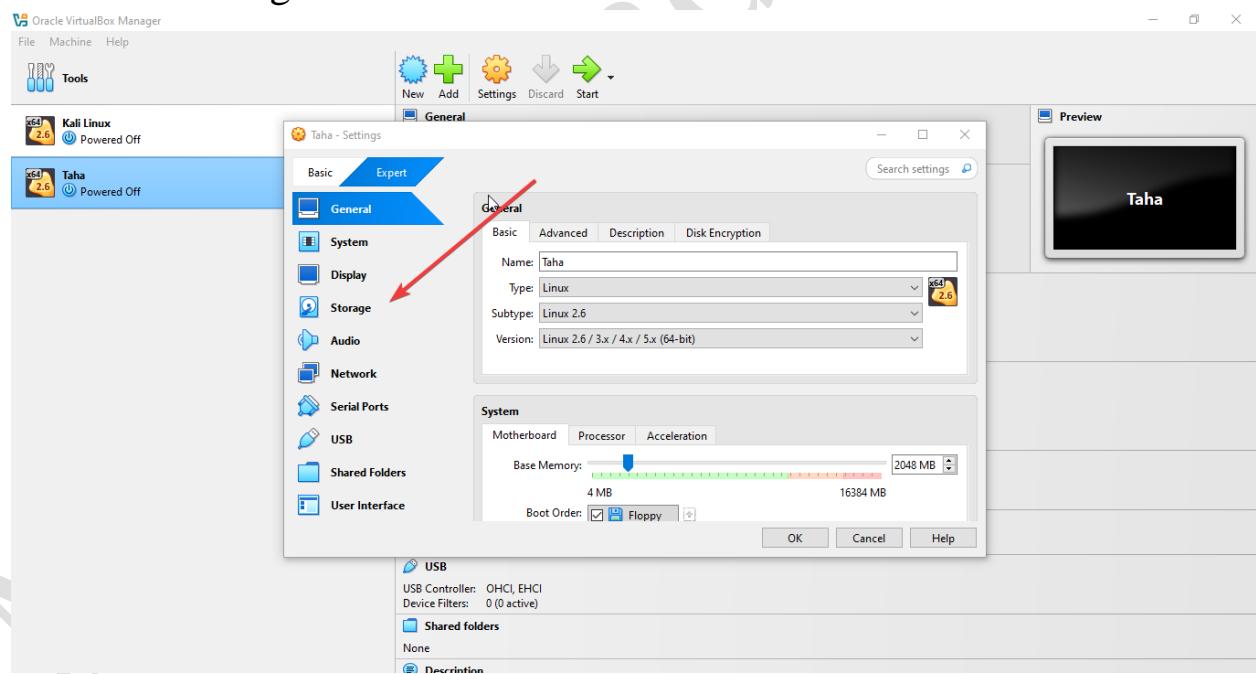
After following these steps click on finish:



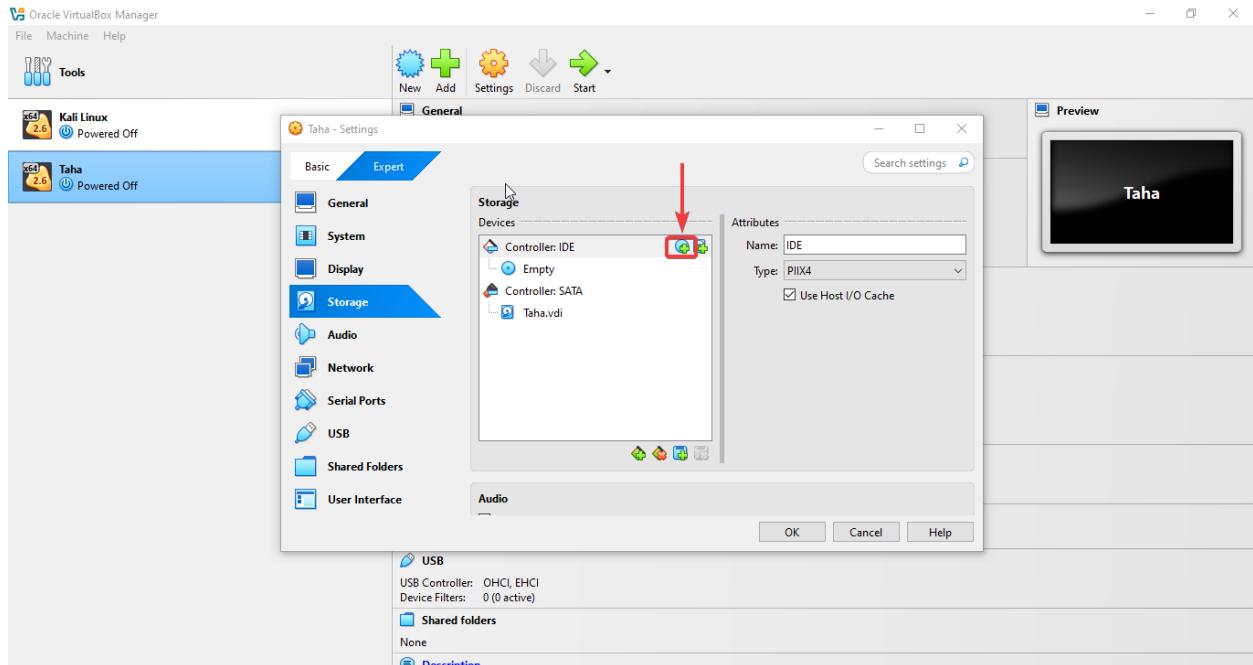
After Finishing click the Settings:



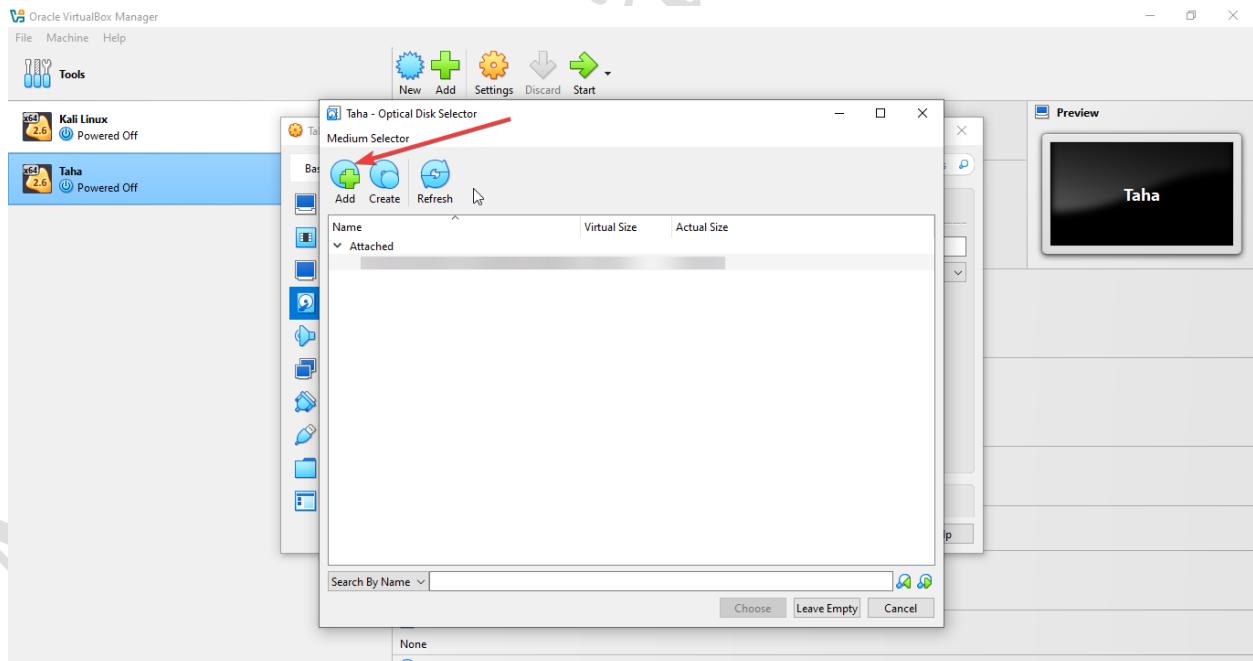
Go to the Storage Tab:



Select This Controller IDE and click this:

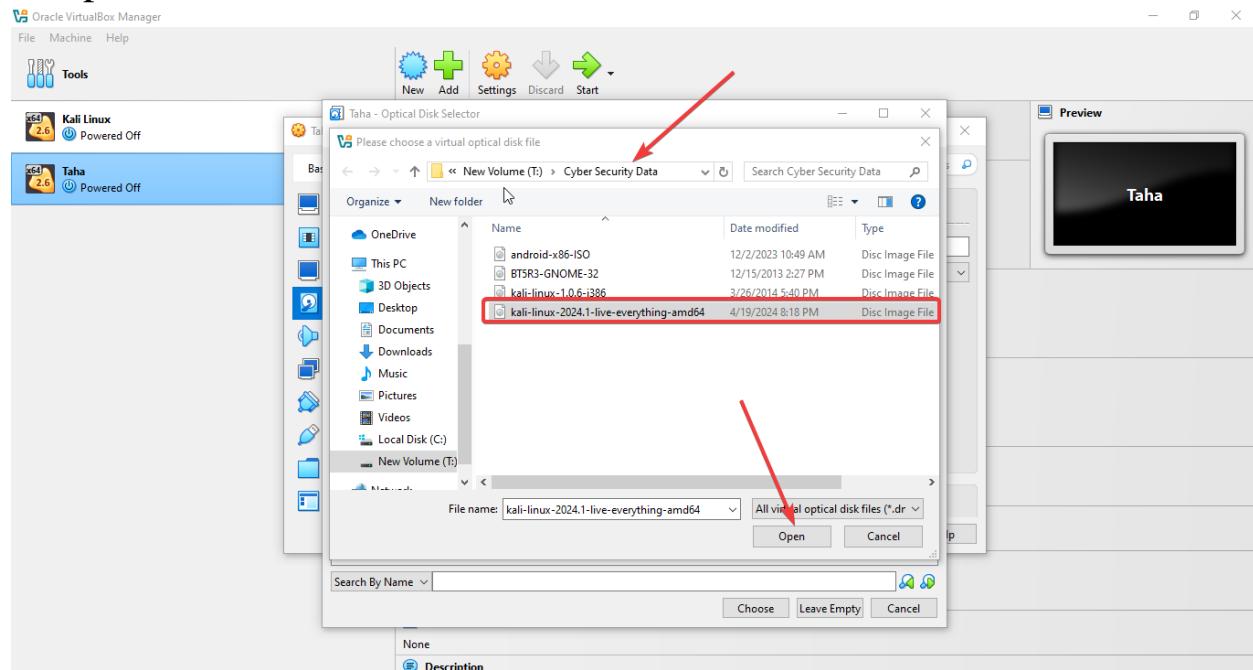


Click on this Add Button:

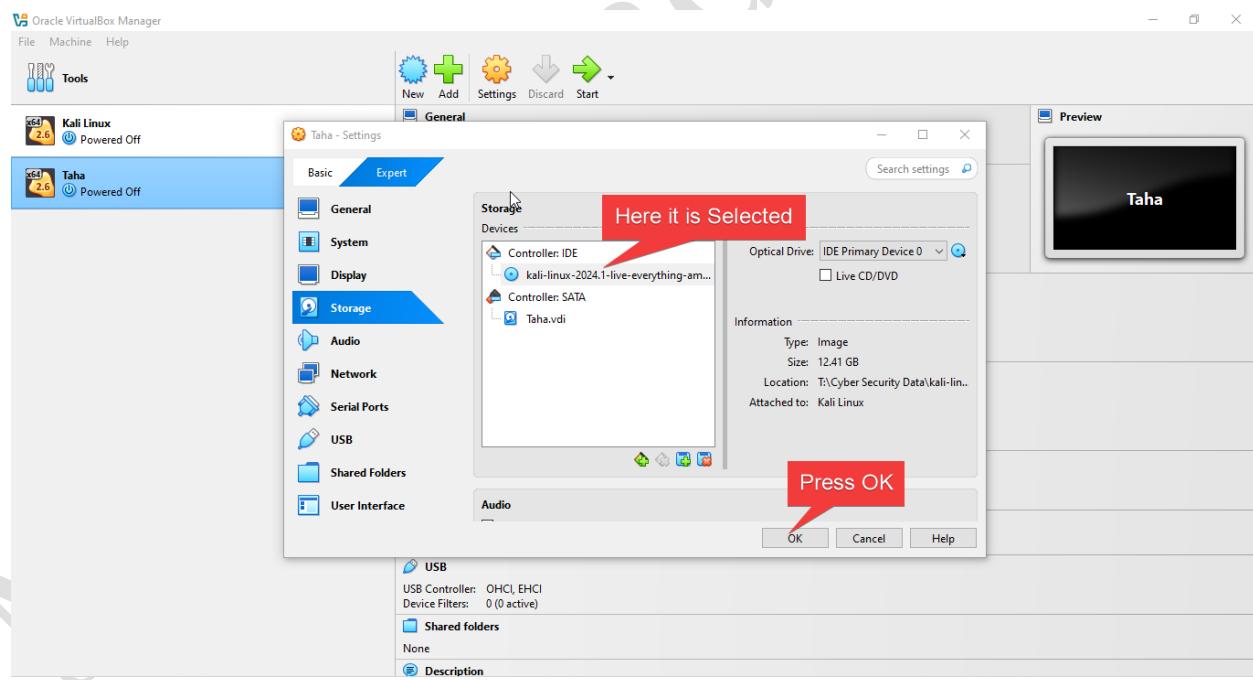


Go to where you put the cyber security data and select kali linux 2024

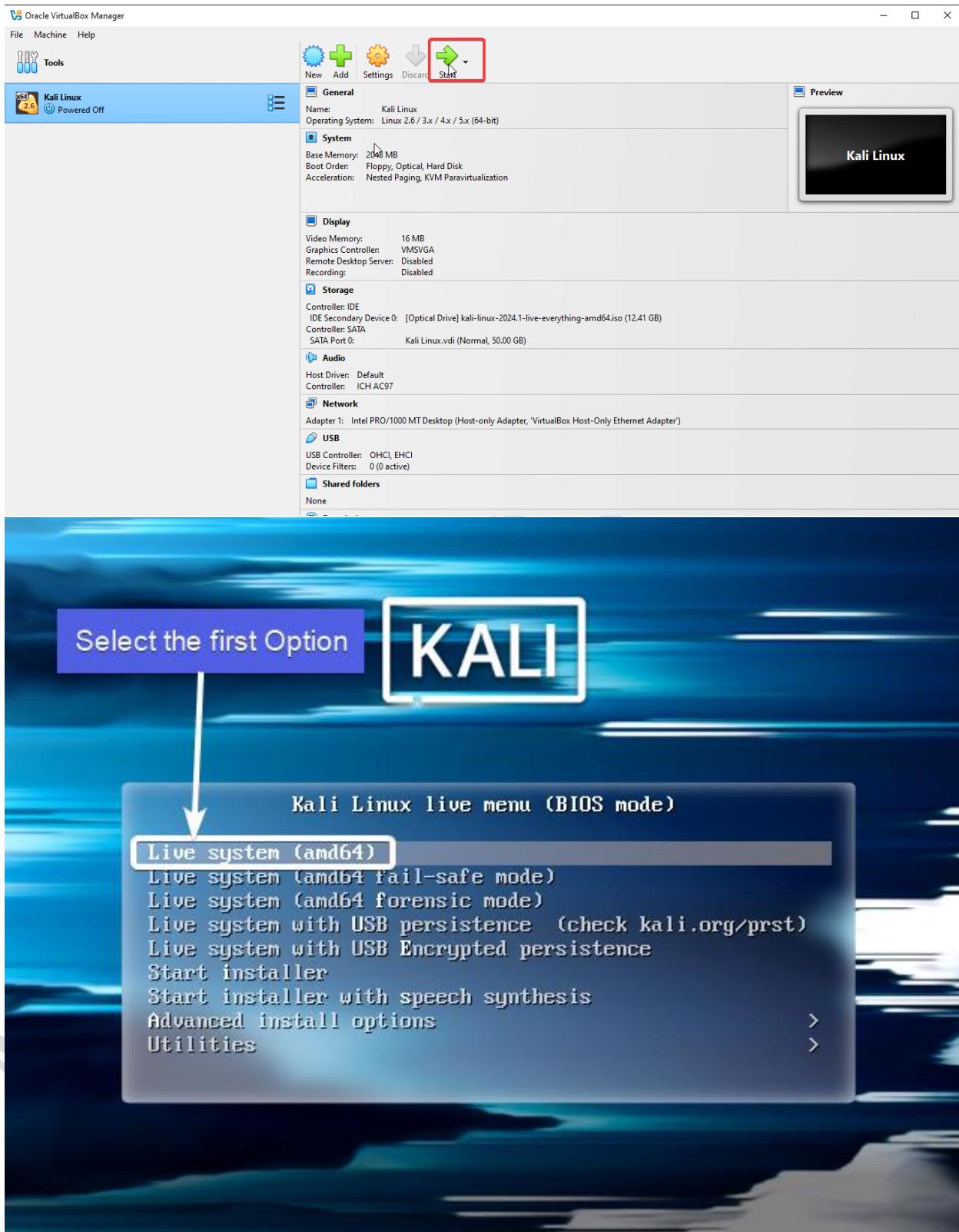
and press choose:



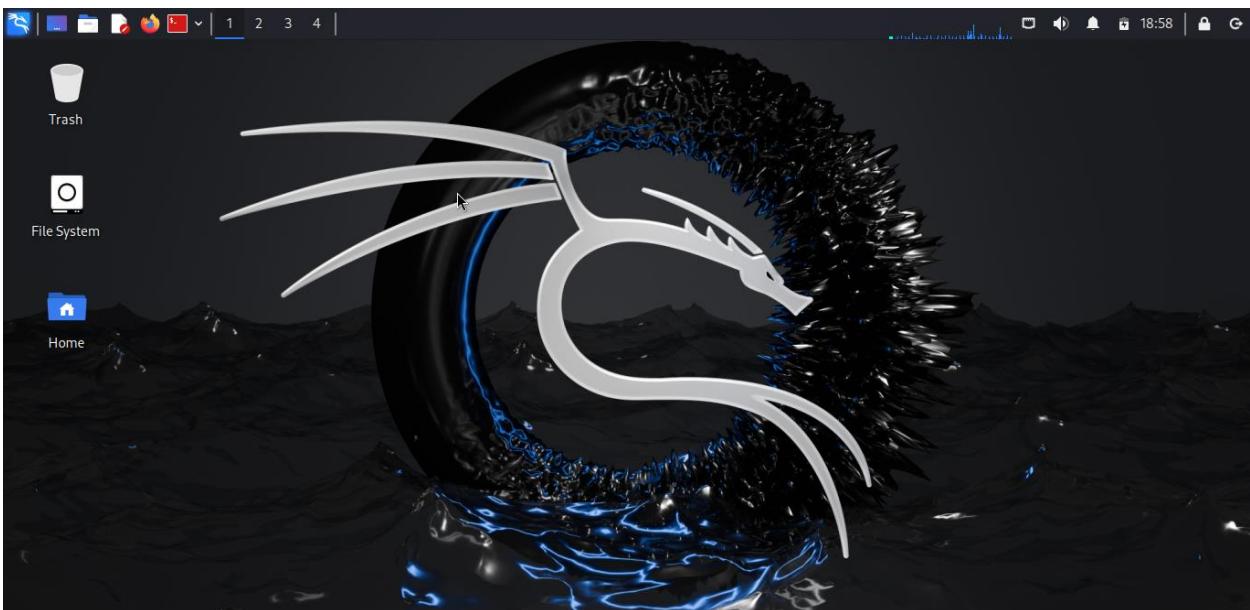
Now it will be installed:



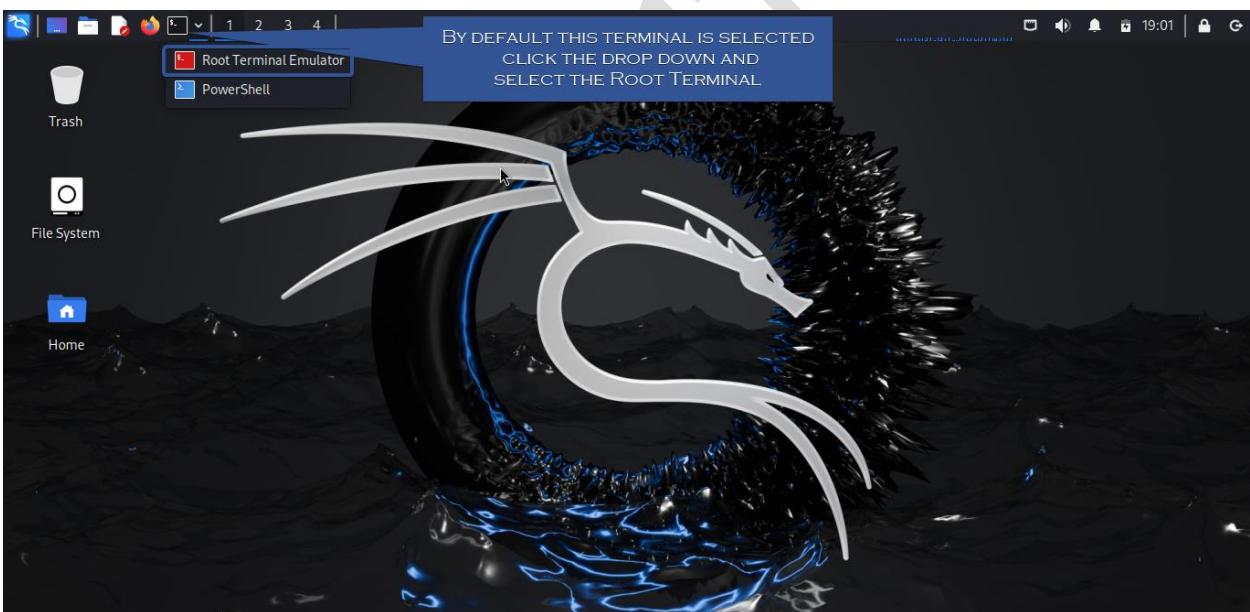
After That click on this start button to start kali linux:



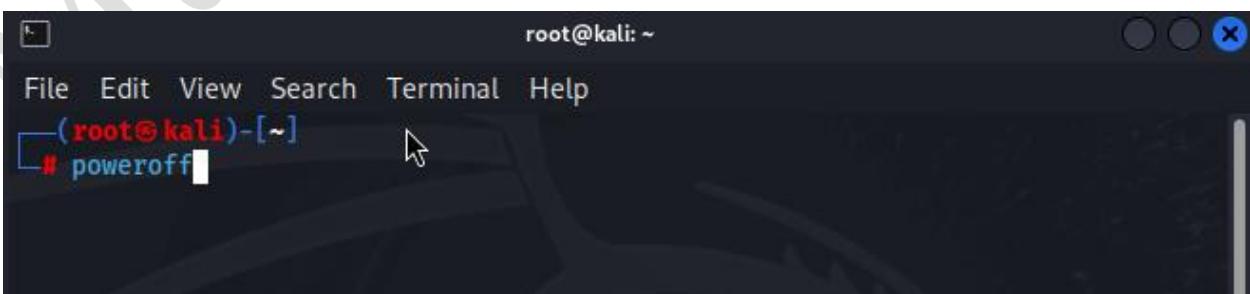
This is the Home Screen of the Kali Linux:



Select the Root Terminal Emulator



Run this command to Power off the VM:



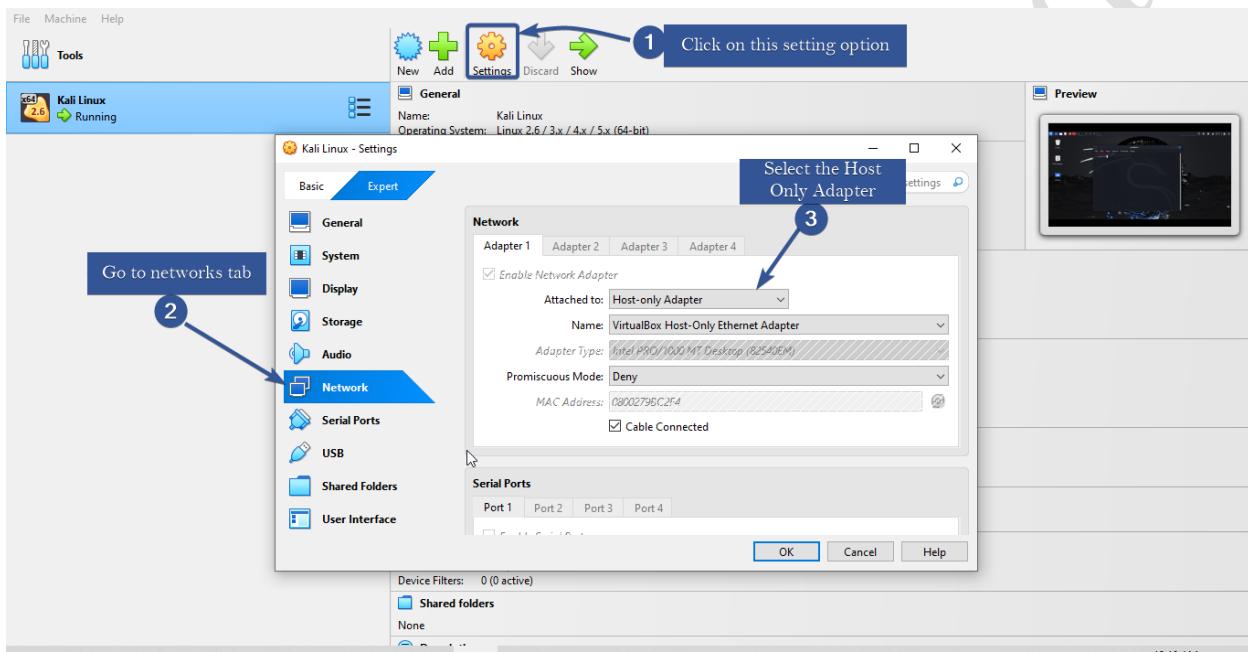
A screenshot of a terminal window titled "root@kali: ~". The window shows a menu bar with File, Edit, View, Search, Terminal, and Help. The terminal prompt is "(root@kali)-[~]". A command "# poweroff" is typed into the terminal, and the cursor is positioned after it. The background of the terminal window is a dark version of the Kali Linux desktop background.

## ➤ Host – only Adapter:-

If we select **Host-only Adapter** for networking in VirtualBox/VMware, then all the machines (VMs) that have chosen **Host-only Adapter** will be able to communicate with each other.

- **Key Point:-**

These machines **can only talk to each other** but will **not have internet access** through this adapter.

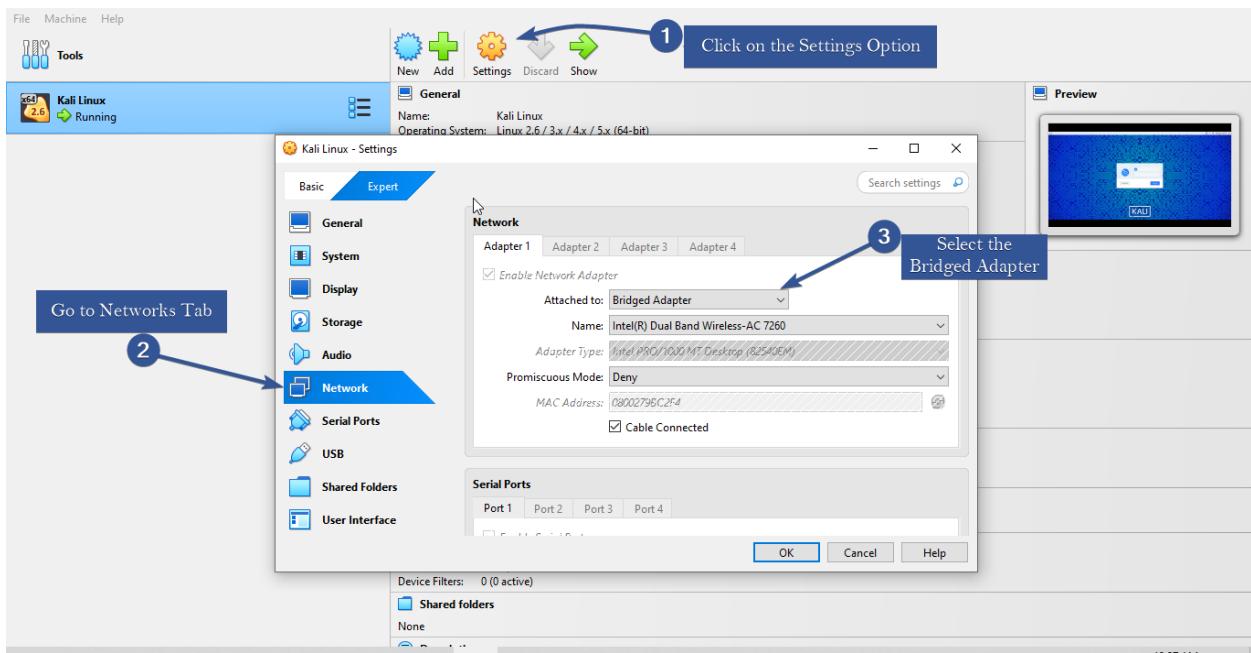


## ➤ Bridged Adapter:

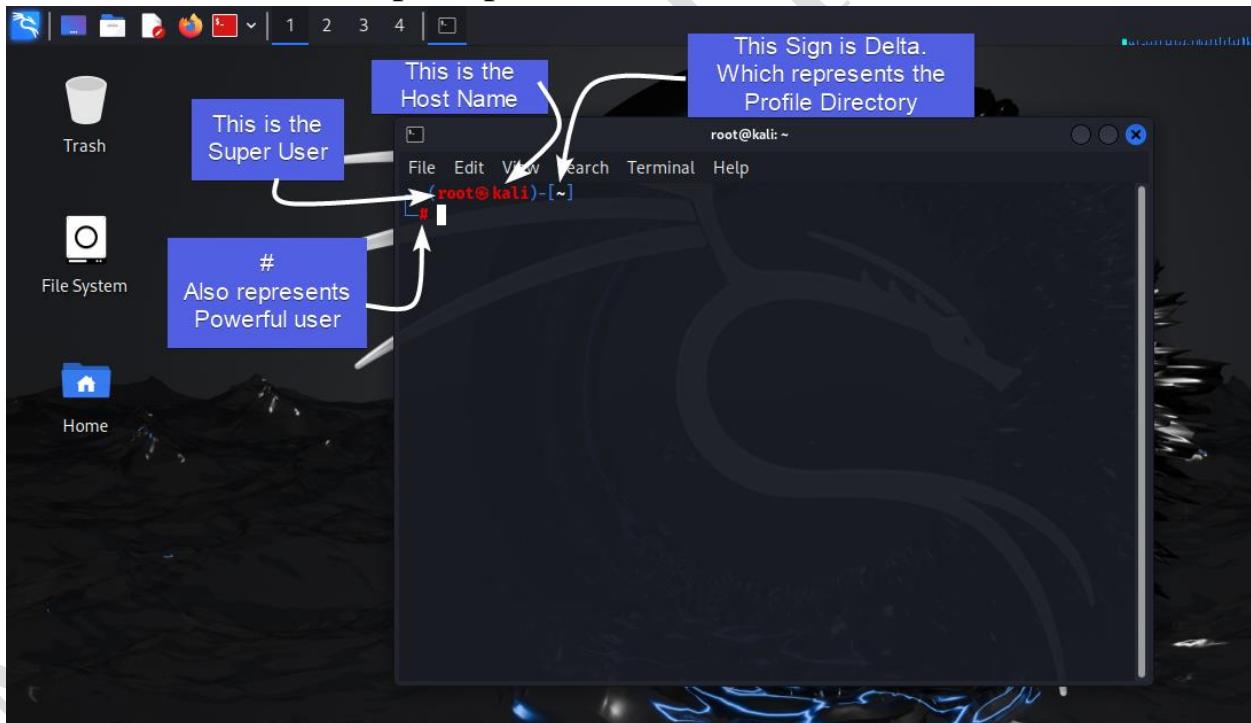
If we select **Bridged Adapter**, then all the machines (VMs) that have chosen **Bridged Adapter** will be connected to each other **and** they will also connect to the **same physical network as the host machine**.

- **Key Points:-**

- If the **physical machine has internet access**, then all the connected VMs will also have internet access.
- If the **physical machine loses internet**, then the VMs will also lose internet.



## Kali Linux Shell Prompt Explained:



## ➤ Commands Learned:-

1. whoami → Shows the currently logged-in user (Linux is **case sensitive**).
2. hostname → Shows the host name of the system

## Navigating in root Terminal:-

3. cd /etc
4. cd /tmp
5. cd /opt

## ➤ When in the root Terminal:-

- ~ changes to the folder name when moving, e.g., cd /etc replaces ~ with /etc.
- pwd → Shows the **present working directory**.



The screenshot shows a terminal window titled "root@kali: /opt". The window has a dark theme with light-colored text. The terminal menu bar includes File, Edit, View, Search, Terminal, and Help. The command line shows the following session:

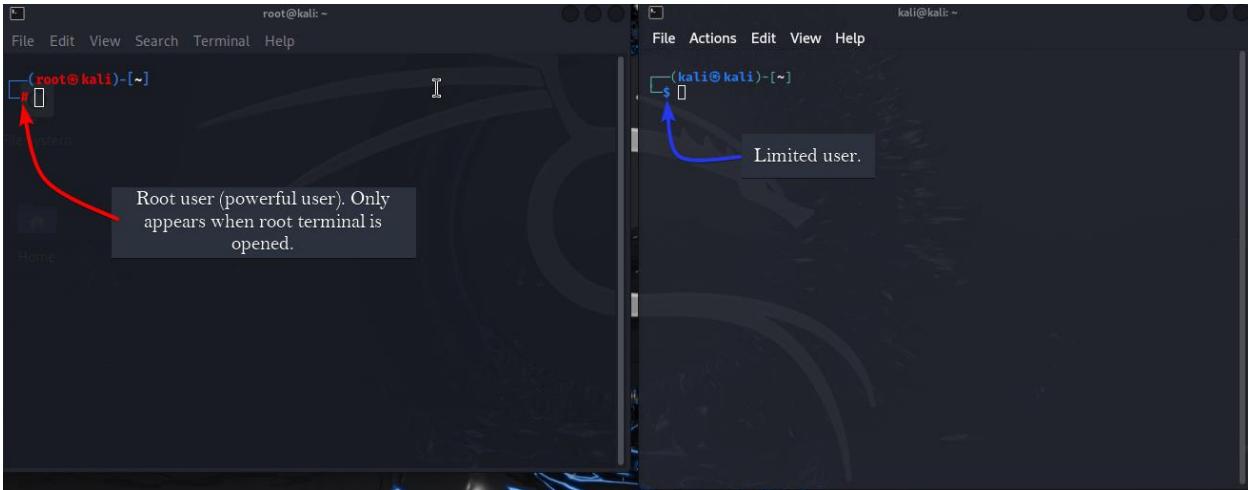
```
root@kali: /opt
File Edit View Search Terminal Help
[(root@kali)-[~]
# whoami
root
[(root@kali)-[~]
# hostname
kali
[(root@kali)-[~]
# cd /etc
[(root@kali)-[/etc]
# cd /tmp
[(root@kali)-[/tmp]
# cd /opt
[(root@kali)-[/opt]
# ]
```

## ➤ Symbols in Linux:-

# → Root user (powerful user). Only appears when root terminal is opened.

\$ → Limited User

- In Windows: you can create any user and give **administrator rights**.
- In Linux: **only root** has administrator rights.



## ➤ User Privileges:-

1. **Root User:** Can perform administrative tasks, no limitations.
2. **Normal User:** Can only perform basic tasks, not everything is possible.

## ➤ File & Directory Commands

- ls → Lists contents of the current directory.
- ls /tmp → Lists contents of /tmp without entering it.
- touch [filename] → Creates a file.
- mkdir [foldername] → Creates a folder (e.g., mkdir M\_Taha).
- cd ~/M\_Taha → Access the created folder.
- rm [filename] → Removes a file.
- rm -r [foldername] → Removes a folder.

```

root@kali: ~
File Edit View Search Terminal Help
[root@kali: ~]
# ls
firmware-mod-kit microsoft Teeth xplico
[root@kali: ~]
# touch Taha
[root@kali: ~]
# mkdir M_Taha
[root@kali: ~]
# ls
Taha
[root@kali: ~]
# cd ~M_Taha
[root@kali: ~/M_Taha]
# cd
[root@kali: ~]
# rm Taha
[root@kali: ~]
# rm -r M_Taha
[root@kali: ~]
# ls
[root@kali: ~]
# 

```

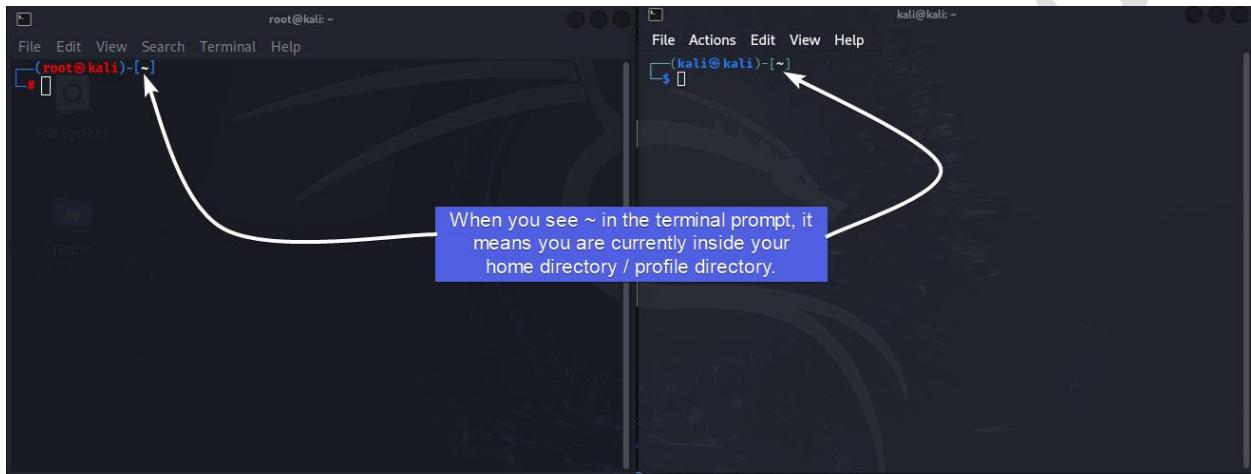
- Files appear in white and folders appear in blue

## ➤ History:-

history → Shows the list of previously used commands.

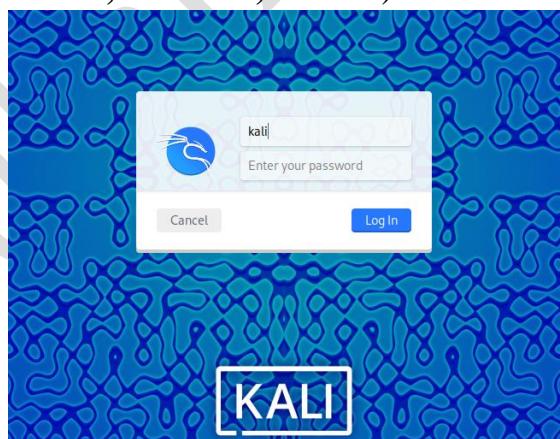
## ➤ Home Directory (~):

- In Linux, the **tilde symbol (~)** represents the **home directory / profile directory** of the currently logged-in user.
- Example: If you're logged in as user kali, then ~ points to /home/kali. If you're logged in as root, ~ points to /root.
- When you see ~ in the terminal prompt, it means you are currently inside your **home directory / profile directory**.



## ➤ Automatic Folders:

- When a user logs in graphically for the first time, Linux automatically creates some default folders (like **Desktop**, **Downloads**, **Documents**, **Music**, **Pictures**, **Videos**) in their **home directory**.



- **Example: Run:**

```
root@kali: /home/kali
File Edit View Search Terminal Help
[root@kali ~]
# cd /home/kali
[root@kali /home/kali]
# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[root@kali /home/kali]
#
```

➤ **Hidden Files:**

- To view **hidden files**, use:

```
root@kali: ~
File Edit View Search Terminal Help
[root@kali ~]
# ls -a
.           .config   .profile
..          .dbus      .ssh
.bashrc     .face     .vboxclient-display-svga-x11-tty1-control.pid
.bashrc.original .face.icon .zshrc
.cache      .gvfs

[root@kali ~]
#
```

- The **-a** option means **all**, including hidden files.
- InLinux (and UNIX-like systems), hidden files and folders start with a **dot (.)**.

Example: `.bashrc`, `.profile`, `.config`/

➤ **In Windows**, hidden files are controlled by system attributes, not by starting the name with a dot.

## ➤ File Extensions:

- In **Windows**, file extensions are important (.txt, .zip, .exe, etc.).
- In **Linux**, file extensions are **optional**. Files are recognized by their **content**, not extension.
- Example: A shell script may not have .sh extension but can still be executed if it has executable permissions.
- In **Linux** the color **White** represents a **text file** and the color **Blue** represents a **directory**.

## ➤ Viewing File Contents:

- To view contents of a file use:

```
cat filename
```

```
(root㉿kali)-[~]
└─# cat .profile
# ~/.profile: executed by Bourne-compatible login shells.

if [ "$BASH" ]; then
    if [ -f ~/.bashrc ]; then
        . ~/.bashrc
    fi
fi

mesg n 2> /dev/null || true

[root@kali ~]#
└─#
```

- **Only use cat for small files** (otherwise the output will overflow).
- To view large files use page by page using this command:

```
more filename
```

```
(root㉿kali)-[~]
└─# more .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
--More--(11%)
```

- Press **Enter** to move line by line.

- Press **Spacebar** to move page by page.
- To view first 10 lines of a file use this command:  
**head filename**
- To view the last 10 lines of a file use this command:  
**tail filename**
- Custom Number of lines let's say first 15 lines:  
**head -n 15 filename**
- Custom number of lines let's say last 15 lines:  
**tail -n 15 filename**
- Get the first line of the file only:  
**head -n 1 filename**
- Get the last line of the file only:  
**tail -n 1 filename**

## ➤ System Information:

- To check CPU details:  
**cat /proc/cpuinfo**
- To check ram details:  
**cat /proc/meminfo**

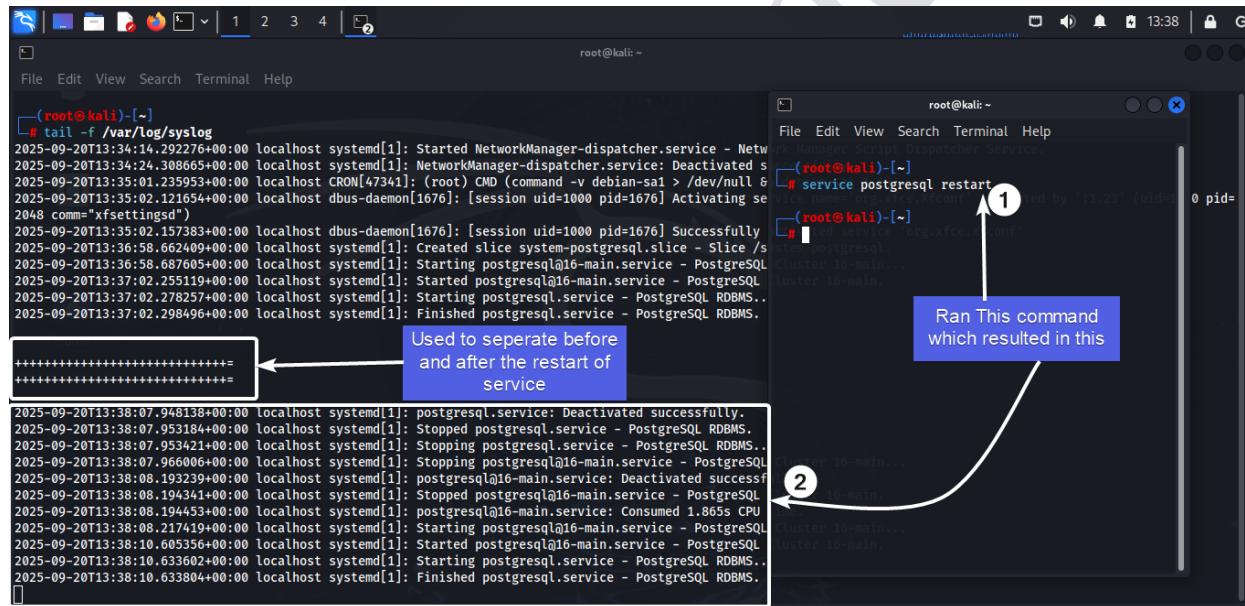
## ➤ Logs

- Linux System logs are stored in **/var/log** To see them:  
**ls /var/log**

```
(root@kali:~]
# ls /var/log
alternatives.log  eaphammer    live          speech-dispatcher
apache           exim4        macchanger.log  stunnel4
apt              faillog      mosquito      syslog
auth.log         fontconfig.log nginx        sysstat
blue_hydra       freeradius   notus-scanner tiger
boot.log         freeradius-wpe openvpn     tor
bootstrap.log   gvm          postgresql   user.log
bttmp            hostapd-wpe  private     wtmp
chkrootkit      inetsim      README      Xorg.0.log
clamav           journal     redis       Xorg.1.log
cron.log         kern.log    redsnarf   Xorg.1.log.old
defectdojo      lastlog     runit      xrdp.log
dpkg.log         lightdm     samba     xrdp-sesman.log
dradis           lighttpd    snort
```

(root@kali:~]

- To monitor a log file continuously (live updates)  
**tail -f /var/log/syslog**
- Example:
  - Run in one Terminal:  
**tail -f /var/log/syslog** (This will keep new logs as they are added)
  - In another terminal restart a service like PostgreSQL  
**service postgresql restart**
  - You will immediately see the related logs in the first terminal



The screenshot shows two terminal windows on a Kali Linux desktop environment. The left terminal window, titled 'root@kali: ~', displays the command `tail -f /var/log/syslog` running. The output shows various system logs, including the start of NetworkManager and the PostgreSQL service. A blue box highlights the PostgreSQL logs, and an annotation with arrow 1 points to the command `# service postgresql restart` in the right terminal. The right terminal window, also titled 'root@kali: ~', shows the execution of this command. An annotation with arrow 2 points to the resulting log output in the left terminal, which includes messages like 'postgresql.service: Deactivated successfully.' and 'postgresql.service: Stopped'.

```

root@kali: ~
# tail -f /var/log/syslog
2025-09-20T13:34:14.292276+00:00 localhost systemd[1]: Started NetworkManager-dispatcher.service - Netw
2025-09-20T13:34:24.308865+00:00 localhost systemd[1]: NetworkManager-dispatcher.service: Deactivated s
2025-09-20T13:35:01.235953+00:00 localhost CRON[47341]: (root) CMD (command -v debian-sai > /dev/null &
2025-09-20T13:35:04.121654+00:00 localhost dbus-daemon[1676]: [session uid=1000 pid=1676] Activating se
2048 comm="xfsettingsd")
2025-09-20T13:35:02.157383+00:00 localhost dbus-daemon[1676]: [session uid=1000 pid=1676] Successfully
2025-09-20T13:36:58.662409+00:00 localhost systemd[1]: Created slice system-persistent.slice - Slice /s
2025-09-20T13:36:58.687605+00:00 localhost systemd[1]: Starting postgresql@16-main.service - PostgreSQL
2025-09-20T13:37:02.255119+00:00 localhost systemd[1]: Started postgresql@16-main.service - PostgreSQL
2025-09-20T13:37:02.278257+00:00 localhost systemd[1]: Starting postgresql.service - PostgreSQL RDBMS..
2025-09-20T13:37:02.298496+00:00 localhost systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.

root@kali: ~
# service postgresql restart
[...]
root@kali: ~
# [1] Ran This command which resulted in this
root@kali: ~
# [2]

```

- You can check the history of the commands you've run using:  
**history**

```
(root@kali)-[~]
# history
 1  pwd
 2  cd /home/kali
 3  cd
 4  cd /home/kali
 5  ls
 6  cd ..
 7  cd /home/kali
 8  mkdir M_Taha
 9  rm -r M_Taha
10  ls
11  cd
12  ls -a
13  clear
14  ls
15  touch .Taha
16  mkdir M_Taha
17  ls -a
18  rm -r M_Taha
19  mkdir .M_Taha
20  ls
21  ls -a
22  rm -r M_Taha
23  rm .Taha
24  ls
25  ls -a
26  rm -r .M_Taha
27  ls -a
28  clear

29  ls -a
30  cat .bashrc
31  clear
32  touch .Taha
33  cat .Taha
34  ls -a
35  cat .vboxclient-display-svga-x11-tty1-control.pid
36  cat .face
37  clear
38  ls -a
39  cat /proc/cpuinfo
40  cat /proc/meminfo
41  more /proc/cpuinfo
42  clear
43  history
44  clear
45  head /proc/cpuinfo
46  tail /proc/meminfo
47  head -n 1 /proc/meminfo
48  ls
49  ls -a
50  head -n 1 .face
51  tail -n 1 .face
52  ls -a
53  head -n 1 .bashrc
54  clear
55  ls /var/log
56  head -n 1 /var/log/syslog
57  tail -f /var/log/syslog
58  clear
```

## ➤ Getting Help in Linux:

- If you need technical help, you can use **Google** or **ChatGPT** but **Linux** itself provide multiple ways to get help about commands.

## ➤ Command Line vs GUI:

- **Command line** is the **professional approach**.
- A **Cybersecurity professional** mainly works on the command line, since most powerful tools and administrative tasks are handled there.

## ➤ Tab Completion:

- If you type the **first few letters** of a command and then press **Tab**, Linux will try to auto-complete the command or list all the possible commands related to that specific command.

The screenshot shows a terminal window with the following details:

- Terminal Title:** root@kali: ~
- Text at the top:** File Edit View Search Terminal Help
- Text in the terminal:** # ls
- Completion Information:** Completing external command, Completing alias, Completing parameter, LS\_COLORS
- Autocompletion List:** ls, lsar, lsassy, lsattr, lsblk, lsb\_release, lscpu, Completing alias ls, Completing parameter LS\_COLORS, lsfd, lsinitramfs, lsipc, lsirq, lslocks, lslogins, lsmod, lsmtd, lsns, lsof, lspci, lsppot, lspower, lspst, lstopo, lstopo-no-graphics, lsusb
- Annotations:**
  - A blue box labeled "1 I typed ls" has an arrow pointing to the "ls" in the command line.
  - A blue box labeled "2 Then i pressed Tab" has an arrow pointing to the completion list.

## ➤ Where Commands Are Stored:

- Linux Commands are stored in specific directories:
  - ls /bin
  - ls /usr/bin
  - ls /usr/local/bin
  - ls /sbin
  - ls /usr/sbin
  - ls /usr/local/sbin

## ➤ bin vs sbin:

bin	sbin
Contains <b>basic commands</b> that all users can run.	Contains <b>administrative commands</b> , usually only run by the root user

Even if you use sudo, not every command in sbin will be accessible unless your account has the required privillages.

**Administrative commands affect the whole system.**

## ➤ Finding Where a Command Lives:

- To check whether a command is basic or administrative use:
  - **which [command name]**
  - **Example:**

The screenshot shows a terminal window titled 'root@kali: ~'. The terminal displays the following commands and output:

```
(root@kali)-[~]
# which cat
/usr/bin/cat

(root@kali)-[~]
# which ifconfig
/usr/sbin/ifconfig

(root@kali)-[~]
#
```

The terminal window has a dark background with a blue and black dragon logo watermark. The text is white and clearly legible.

## ➤ Sudo & Root:

- **Sudo** is used to run commands with administrative permissions.
- Root can run all commands in /sbin.
- A non-root user can run them only if root has granted permission.

## ➤ Manual Pages (man):

- Every command in linux has a **manual page** (documentation).
- **Example: man ls**

The screenshot shows a terminal window titled 'root@kali: ~'. The window title bar also displays 'User Commands' and 'LS(1)'. The terminal content is the manual page for 'ls'. It includes sections for NAME, SYNOPSIS, and DESCRIPTION. The SYNOPSIS section shows the command 'ls [OPTION]... [FILE]...'. The DESCRIPTION section provides details about listing directory contents and handling options like -a, -A, and --author. At the bottom of the page, there is a message: 'Manual page ls(1) line 1 (press h for help or q to quit)'.

```
root@kali: ~
File Edit View Search Terminal Help
LS(1) User Commands LS(1)

NAME
ls - list directory contents

SYNOPSIS
ls [OPTION]... [FILE]...

DESCRIPTION
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

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

-a, --all
      do not ignore entries starting with .

-A, --almost-all
      do not list implied . and ..

--author
Manual page ls(1) line 1 (press h for help or q to quit)
```

- In the Synopsis section of the manual, the **bold text means mandatory arguments**.

## ➤ File Properties:

- To see file and directories properties **ls -l**:

The screenshot shows a terminal window with the command '# ls -l' entered and its output 'total 0' displayed.

```
(root@kali)-[~]
# ls -l
total 0
```

- To include hidden files **ls -al**:

```
(root㉿kali)-[~]
# ls -al
total 38
drwx----- 1 root root 160 Sep 23 09:45 .
drwxr-xr-x 1 root root 180 Sep 23 09:09 ..
-rw-r--r-- 1 root root 5551 Feb 25 2024 .bashrc
-rw-r--r-- 1 root root 571 Feb 25 2024 .bashrc.original
drwx----- 1 root root 120 Sep 23 09:12 .cache
drwx----- 2 root root 60 Sep 23 09:12 .config
drwx----- 3 root root 60 Sep 23 09:12 .dbus
-rw-r--r-- 1 root root 11656 Feb 25 2024 .face
lrwxrwxrwx 1 root root 11 Feb 25 2024 .face.icon -> /root/.face
dr-x----- 2 root root 0 Sep 23 09:12 .gvfs
-rw------- 1 root root 20 Sep 23 09:45 .lessht
-rw-r--r-- 1 root root 161 Feb 15 2024 .profile
drwx----- 2 root root 3 Feb 25 2024 .ssh
-rw-r----- 1 root root 5 Sep 23 09:10 .vboxclient-display-svga-x11-tty1-contr
rol.pid
-rw-r--r-- 1 root root 10868 Feb 25 2024 .zshrc
```

- To see human-readable sizes (KB, MB, GB) **ls -alh**:

```
(root㉿kali)-[~]
# ls -alh
total 38K
drwx----- 1 root root 160 Sep 23 09:45 .
drwxr-xr-x 1 root root 180 Sep 23 09:09 ..
-rw-r--r-- 1 root root 5.5K Feb 25 2024 .bashrc
-rw-r--r-- 1 root root 571 Feb 25 2024 .bashrc.original
drwx----- 1 root root 120 Sep 23 09:12 .cache
drwx----- 2 root root 60 Sep 23 09:12 .config
drwx----- 3 root root 60 Sep 23 09:12 .dbus
-rw-r--r-- 1 root root 12K Feb 25 2024 .face
lrwxrwxrwx 1 root root 11 Feb 25 2024 .face.icon -> /root/.face
dr-x----- 2 root root 0 Sep 23 09:12 .gvfs
-rw------- 1 root root 20 Sep 23 09:45 .lessht
-rw-r--r-- 1 root root 161 Feb 15 2024 .profile
drwx----- 2 root root 3 Feb 25 2024 .ssh
-rw-r----- 1 root root 5 Sep 23 09:10 .vboxclient-display-svga-x11-tty1-contr
rol.pid
-rw-r--r-- 1 root root 11K Feb 25 2024 .zshrc
```

```
-rw-r--r-- 1 root root 10868 Feb 25 2024 .zshrc
```

-rw-r--r--	1	root	root	10868	Feb 25 2024	.zshrc
File Permissions	Number of links	Owner	Group Member Ship	File size (bytes)	Last Modified Date and Time	File Name

## ➤ Word Count (wc):

- To count lines, words, and characters in a file **wc filename**:

```
(root@kali)-[~]
# wc .zshrc
258 932 10868 .zshrc
```

```
258 932 10868 .zshrc
```

258	932	10868
Lines	Words	Characters

## ➤ Quick Help:

- To see a quick summary of options (instead of full manual) **ls --help**:

```
(root@kali)-[~]
# ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all          do not ignore entries starting with .
  -A, --almost-all   do not list implied . and ..
  --author          with -l, print the author of each file
  -b, --escape       print C-style escapes for nongraphic characters
  --block-size=SIZE  with -l, scale sizes by SIZE when printing them;
                    e.g., '--block-size=M'; see SIZE format below

  -B, --ignore-backups
  -c
More
Downwards
  do not list implied entries ending with ~
  with -lt: sort by, and show, ctime (time of last
  change of file status information);
  with -l: show ctime and sort by name;
  otherwise: sort by ctime, newest first
```

## ➤ Processes and Services:

- To list running services/processes:
  - **Ps -aux**
- To view output **page by page**:
  - **Ps -aux | more**
- To monitor system performance in real time:
  - **top**
  - This shows processes sorted by **CPU** and **memory usage**.
  - To quit **top**, press **q**.

## ➤ Zombie Process:

- A **zombie process** is a child process whose **parent process has already terminated**.
- Since it has no parent to clean it up, it lingers until the system reclaims it.

## ➤ Quit Commands:

- In most Linux help/manual/programs (like man, more, top), you can quit by pressing:
  - **q**

## ➤ Permissions Overview:

- In Linux file and directory permissions are represented like this:
  - **-rw-r--r--**
- This string has a total of 10 characters:
  - The **first character** is the **file type (nature)**, not part of permissions.

-	d	l
regular file	directory	symbolic link

- The next **9 characters** represent permissions.

## ➤ Permission Types:

r	w	x	-
Read	Write	Execute	No Permission

The order is always the same: **read → write → execute**

## ➤ Permission Groups:

- Permissions are divided into three sets:
  - **Owner (user)** → the user who owns the file.
  - **Group** → other users in the same group.
  - **Others** → all other users.
- **Example:**

<b>rw-</b>	<b>r--</b>	<b>r--</b>
Owner can read and write	Group can only read	Other users can only read

## ➤ Checking Permissions:

- To see file and directories permissions **ls -l**:

```
[root@kali)~]# ls -l
total 0
drwxr-xr-x 2 root root 40 Sep 24 09:40 M_Taha
-rw-r--r-- 1 root root 0 Sep 24 09:39 Taha
```

- To see file permissions **ls -l [filename]**.

```
[root@kali)~]# ls -l Taha
-rw-r--r-- 1 root root 0 Sep 24 09:39 Taha
```

- To see directory permissions (not contents) **ls -ld [dirname]**.

```
[root@kali)~]# ls -ld M_Taha
drwxr-xr-x 2 root root 40 Sep 24 09:40 M_Taha
```

## ➤ Groups in Linux:

- In **Windows**, new users are added to predefined groups.
- In **Linux**, when new user is created, a **new group with the same name** is also created by default.
- To check which group a user belongs to:
  - **id username**

```
(root㉿kali)-[~]
# id kali
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialout),24(cdrom),
floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),100(users),101(netdev),
108(debian-tor),121(wireshark),126(bluetooth),127(vboxsf),142(scanner),152(kali),
er)
```

## ➤ Changing Permissions:

- Permissions can be represented by numbers:

Read(r)	Write(w)	Execute(x)	No Permission(-)
4	2	1	0

- Add the values together for each group.
- Example:**

- Create a file:

- touch Taha
- Default Permissions are: -rw-r--r--

```
(root㉿kali)-[~]
# touch Taha

(root㉿kali)-[~]
# ls -l Taha
-rw-r--r-- 1 root root 0 Sep 24 10:02 Taha
```

- If we want **owner, group, and others** to have **read + write (rw-rw-rw-)**:

Owner	Group	Others
r(4)+w(2)	r(4)+w(2)	r(4)+w(2)
6	6	6

- Change Permissions:

- chmod 666 Taha

- New Permissions: **-rw-rw-rw-**

```
(root㉿kali)-[~]
└─# touch Taha

(root㉿kali)-[~]
└─# ls -l Taha
-rw-r--r-- 1 root root 0 Sep 24 10:02 Taha

(root㉿kali)-[~]
└─# chmod 666 Taha

(root㉿kali)-[~]
└─# ls -l Taha
-rw-rw-rw- 1 root root 0 Sep 24 10:02 Taha
```

## ➤ File and User Management:

- Change ownership of a file: **chown [username] [file]**

The screenshot shows a terminal window with the following steps:

- Created a File**: The user runs `# touch Taha`. A blue callout box with the number 1 points to this command.
- See its properties and permissions and here you can see the owner is root**: The user runs `# ls -l Taha`. A blue callout box with the number 2 points to this command. The output shows the file was created by root:root.
- Using the command: chown [username] [file]**: The user runs `# chown kali Taha`. A blue callout box with the number 3 points to this command.
- Now you can see that the owner is changed from root to kali**: The user runs `# ls -l Taha` again. A blue callout box with the number 4 points to this command. The output now shows the file was created by kali:root.

```
File Edit View Search Terminal Help
root@kali: ~
└─# touch Taha
Created a File
└─# ls -l Taha
See its properties and permissions and here you can see the owner is root
└─# chown kali Taha
Using the command: chown [username] [file]
└─# ls -l Taha
Now you can see that the owner is changed from root to kali
└─#
```

- The user must already exist for this to work. If the user does not exist we can add the user using the command: **useradd [username]**.

```
(root@kali)-[~]
# useradd Taha
```

- To change group of a file we can use the command:  
**chgrp [groupname] [file]**

The screenshot shows a terminal window with the following session:

```
File Edit View Search Terminal Help
root@kali: ~
# ls -l Taha
-rw-r--r-- 1 kali root 0 Sep 26 09:46 Taha
# chgrp plugdev Taha
# ls -l Taha
-rw-r--r-- 1 kali plugdev 0 Sep 26 09:46 Taha
#
```

Annotations with arrows point to specific parts of the terminal output:

1. Points to the first 'root' entry in the ls command output. A callout box says: "If we see the properties of file Taha we can see it belongs to root group".
2. Points to the 'root' entry in the chgrp command. A callout box says: "So we will change the group using the command: chgrp [groupname] [file]".
3. Points to the second 'root' entry in the ls command output after the group change. A callout box says: "we can see that the group has been changed from root to plugdev".

- To add a user to a group we can use: **usermod -aG [groupname] [username]**
  - a → append (don't remove existing groups)

- -G → group(s) to add the user into

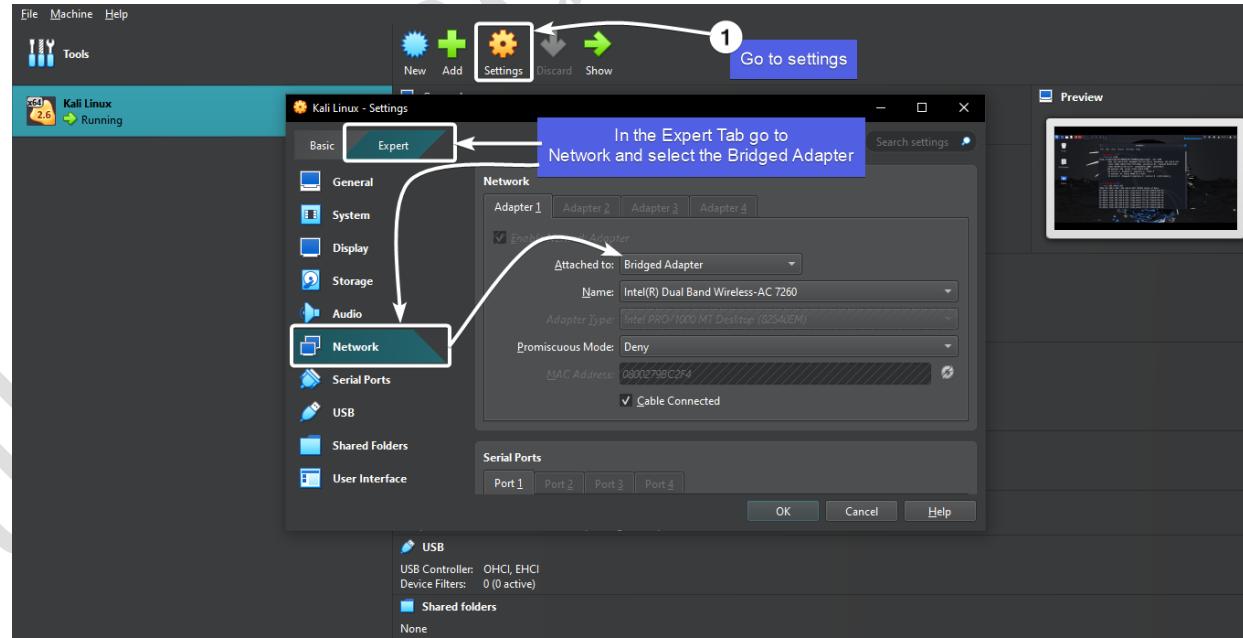
The terminal session shows the following steps:

1. Created a user and see the groups its part of  
# useradd Taha
2. We see here all the groups that kali is part of and choose one from the list  
# id Taha
3. if we see the groups for user Taha we can see Taha is added in the scanner group  
# usermod -aG scanner Taha  
# id Taha

⚠ If you use a **small -g** instead of **-G**, the user will be removed from all their other groups and only added to the one you specify.

## ➤ Network Settings:

- In the vm setting select the bridged adapter if not previously selected and restart the virtual machine:



- We know that in Bridged Adapter setting the virtual machine will have the internet connection that the physical machine have so after selecting and restarting the virtual machine type the command **ifconfig eth0** to check the Network Settings.

- **Look for:**
    - inet → IPv4 Address
    - inet6 → IPv6 Address
    - ether → MAC Address

⚠ If you don't see an inet line → no IPv4 is assigned.  
⚠ If you don't see an inet6 line → no IPv6 is assigned.
  - **Test Connectivity with ping:**
    - `ping -c 4 [ip-address]`

⚠ If you ping from your Virtual Machine (VM) to your Physical Machine (Windows), you might see:
    - Destination Host Unreachable
    - 100% Packet Loss.
  - This usually happens because **Windows Firewall blocks ICMP (ping) requests by default**. It doesn't mean your network is broken — just that your VM can't get a reply from the host until you allow ICMP in the firewall.
- ❖ **What is ICMP:**
- ICMP = *Internet Control Message Protocol*.
  - It's used by network devices (like routers, servers, and PCs) to send error messages and operational information.
  - ping is the most common use of ICMP — it sends an **ICMP Echo Request** and expects an **ICMP Echo Reply**.
  - If ICMP is blocked (like by Windows Firewall), ping will always fail, even if the network itself is fine.