Assignment 2 : Everything is Virtual?

Name: NAIMUR ISLAM NAVID
Id: 24141160
Brac University
CSE Department

Note: I am currently using Virtual Box from Windows and in the Virtual Box I have installed Ubuntu 22.04 version, now I'm going to install nested Ubuntu in my virtual machine using KVM. The steps and the Process are shown below.

Task-1

Install KVM

Update Ubuntu 22.04

At first to get off the ground, launch the terminal and update local package index as follows.

$ sudo apt update

```
naimur@Navid-24141160:~$ sudo apt update
[sudo] password for naimur:
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://us.archive.ubuntu.com/ubuntu jammy/main Translation-en_CA [4,224
Get:6 http://us.archive.ubuntu.com/ubuntu jammy/universe Translation-en_CA [1,
2 B]
Fetched 5,356 B in 2s (3,492 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
115 packages can be upgraded. Run 'apt list --upgradable' to see them.
naimur@Navid-24141160:~$
```

Check if Virtualization is Enabled

Before proceeding any further, we need to check if our CPU supports KVM virtualization. For this to be possible, our system needs to either have a VT-x(vmx) Intel processor or an AMD-V (svm) processor.

This is achieved by running the following command. if the output is greater than 0, then virtualization is enabled. Otherwise, virtualization is disabled and you need to enable it.

$ egrep -c '(vmx|svm)' /proc/cpuinfo

```
naimur@Navid-24141160:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
6
naimur@Navid-24141160:~$
```

Here the value returned 6 which means, our system supports KVM installation.

In addition, we can verify if KVM virtualization is enabled by running the following command:

$ kvm-ok

For this to work, we need to have installed the cpu-checker package, otherwise, we will bump into the error 'Command 'kvm-ok' not found'. Therefore, install the cpu-checker package as follows.

$ sudo apt install -y cpu-checker

```
naimur@Navid-24141160:~$ kvm-ok
Command 'kvm-ok' not found, but can be installed with:
apt install cpu-checker
Please ask your administrator.
naimur@Navid-24141160:~$ sudo apt install cpu-checker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  msr-tools
The following NEW packages will be installed:
  cpu-checker msr-tools
0 upgraded, 2 newly installed, 0 to remove and 115 not upgraded.
Need to get 17.1 kB of archives.
After this operation, 67.6 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 msr-tools amd64
```

Then run the kvm-ok command, and if KVM virtualization is enabled, we should get the following output.

```
naimur@Navid-24141160:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
naimur@Navid-24141160:~$
```

That means our system is ready for installing kvm.

Install KVM on Ubuntu 22.04

Next, run the command below to install KVM and additional virtualization packages on Ubuntu 22.04.

$ sudo apt install -y qemu-kvm virt-manager libvirt-daemon-system virtinst libvirt-clients bridge-utils

Let us break down the packages that we are installing:

- **qemu-kvm** – An opensource emulator and virtualization package that provides hardware emulation.
- **virt-manager** – A Qt-based graphical interface for managing virtual machines via the libvirt daemon.
- **libvirt-daemon-system** – A package that provides configuration files required to run the libvirt daemon.
- **virtinst** – A set of command-line utilities for provisioning and modifying virtual machines.
- **libvirt-clients** – A set of client-side libraries and APIs for managing and controlling virtual machines & hypervisors from the command line.
- **bridge-utils** – A set of tools for creating and managing bridge devices.

Start & Enable Virtualization Daemon.

With all the packages installed, enable and start the Libvirt daemon.

$ sudo systemctl enable --now libvirtd
$ sudo systemctl start libvirtd

```
naimur@Navid-24141160:~$ sudo systemctl enable --now libvirtd
naimur@Navid-24141160:~$ sudo systemctl start libvirtd
naimur@Navid-24141160:~$
```

Confirm that the virtualization daemon is running as shown.

$ sudo systemctl status libvirtd

```
naimur@Navid-24141160:~$ sudo systemctl status libvirtd
● libvirtd.service - Virtualization daemon
     Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; ve
     Active: active (running) since Sat 2024-11-23 17:00:21 +06; 4min
TriggeredBy: ● libvirtd-ro.socket
             ● libvirtd-admin.socket
             ● libvirtd.socket
       Docs: man:libvirtd(8)
             https://libvirt.org
   Main PID: 7089 (libvirtd)
      Tasks: 21 (limit: 32768)
     Memory: 9.5M
        CPU: 572ms
     CGroup: /system.slice/libvirtd.service
             ├─7089 /usr/sbin/libvirtd
             ├─7215 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dns
             └─7216 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dns

Nov 23 17:00:21 Navid-24141160 systemd[1]: Started Virtualization daem
Nov 23 17:00:21 Navid-24141160 dnsmasq[7215]: started, version 2.90 ca
Nov 23 17:00:21 Navid-24141160 dnsmasq[7215]: compile time options: IP
Nov 23 17:00:21 Navid-24141160 dnsmasq-dhcp[7215]: DHCP, IP range 192.
```

In addition, we need to add the currently logged-in user to the kvm and libvirt groups so that they can create and manage virtual machines.

$ sudo usermod -aG kvm $USER
$ sudo usermod -aG libvirt $USER

```
naimur@Navid-24141160:~$ sudo usermod -aG kvm $USER
naimur@Navid-24141160:~$ sudo usermod -aG libvirt $USER
naimur@Navid-24141160:~$
```

The $USER environment variable points to the name of the currently logged-in user. To apply this change, you need to log out and log back again.

With these steps, our KVM is already installed into our Ubuntu 22.04 Host

Task-2

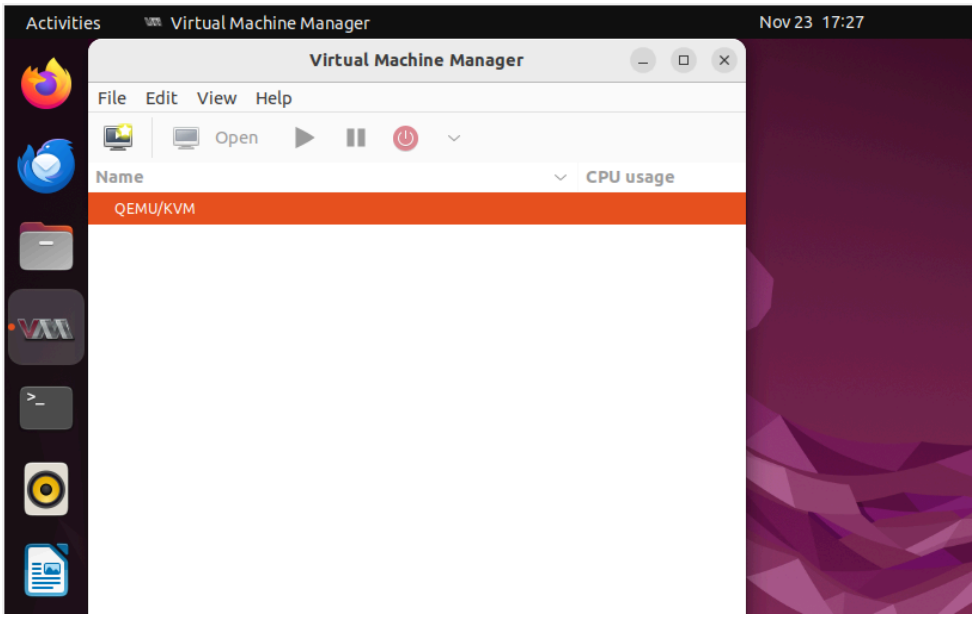
Create a VM using VMM (virtual machine manager) using GUI.

In this part, we are going to create a VM using VMM using a GUI.

With KVM installed, we can begin creating your virtual machines using the virt-manager GUI tool. To get started, use the GNOME search utility and search for ‘Virtual machine Manager’.

Click on the icon that pops up.

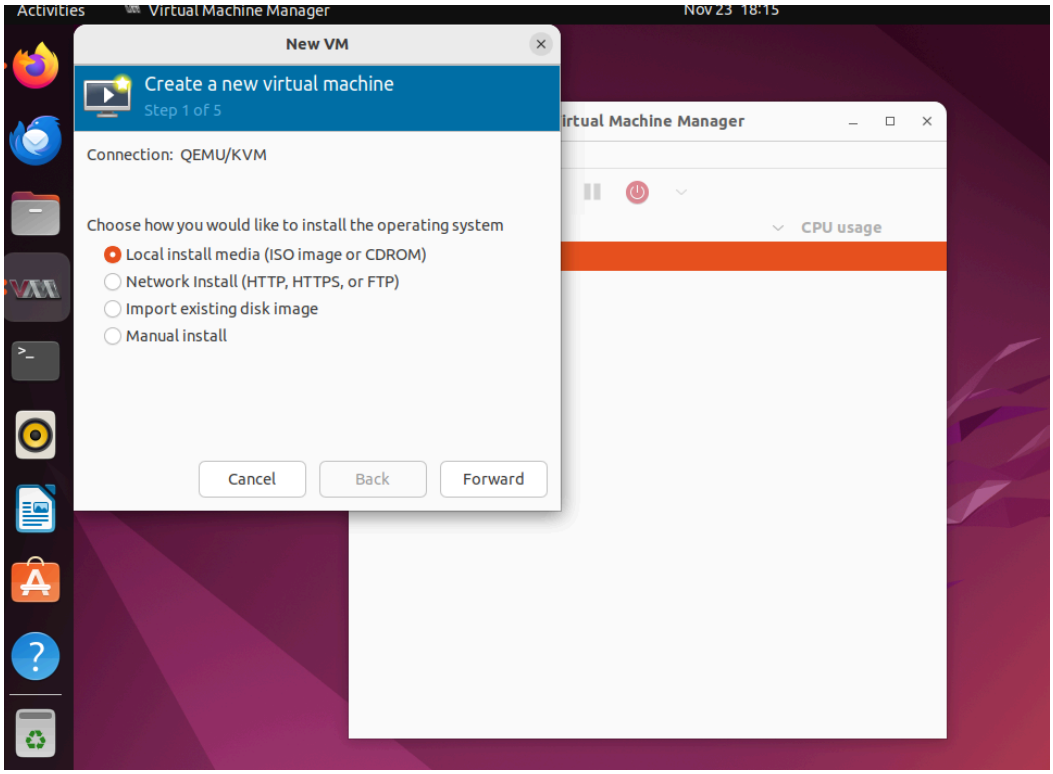
This launches the Virtual Machine Manager Interface.



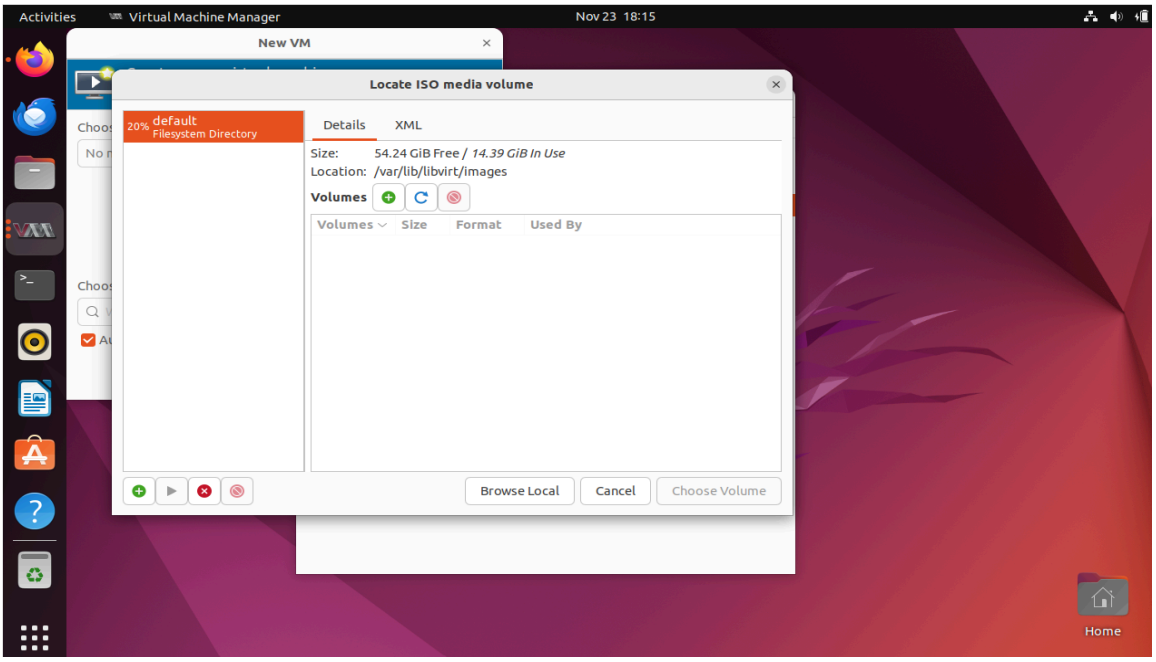
Click on “File” then select “New Virtual Machine”. Alternatively, we can click on the button shown. This pops open the virtual machine installation wizard which presents you with the following four options:

- Local install Media (ISO image or CDROM)
- Network Install (HTTP, HTTPS, and FTP)
- Import existing disk image
- Manual Install

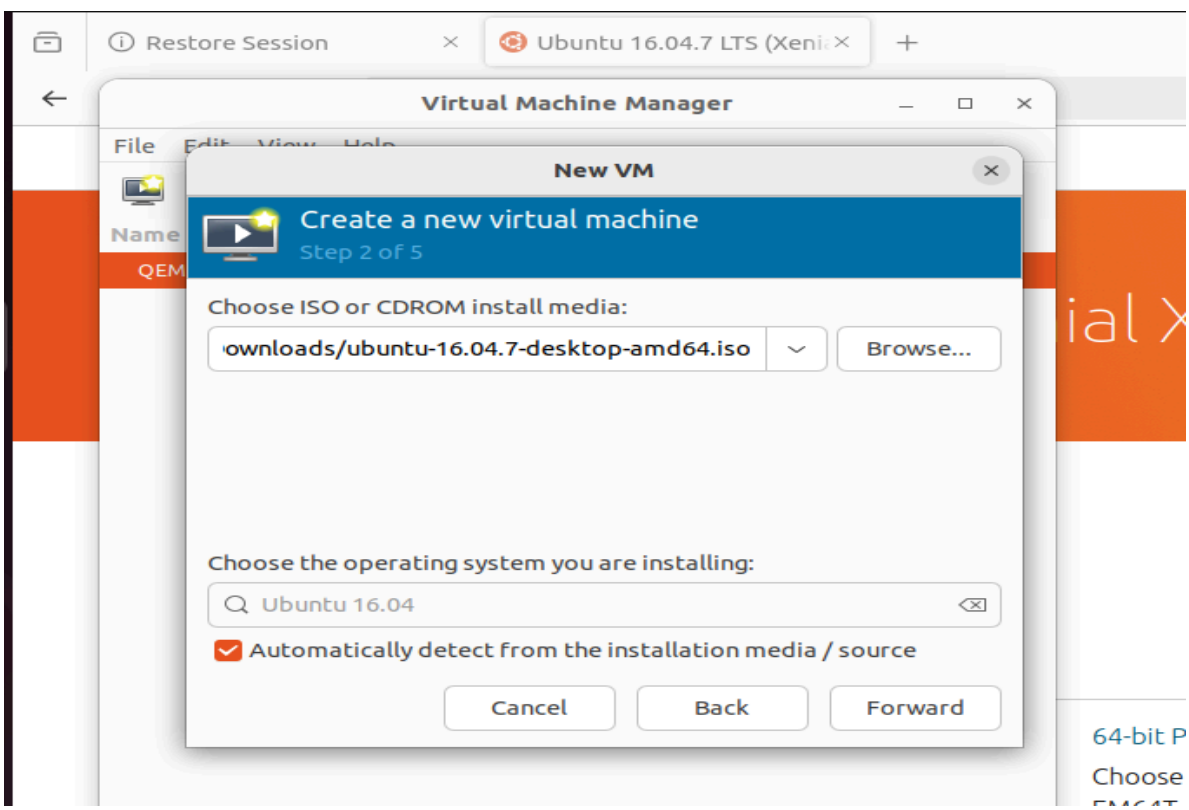
In this guide, we have downloaded a Ubuntu 16.04 ISO image, select the first option and click ‘Forward’.



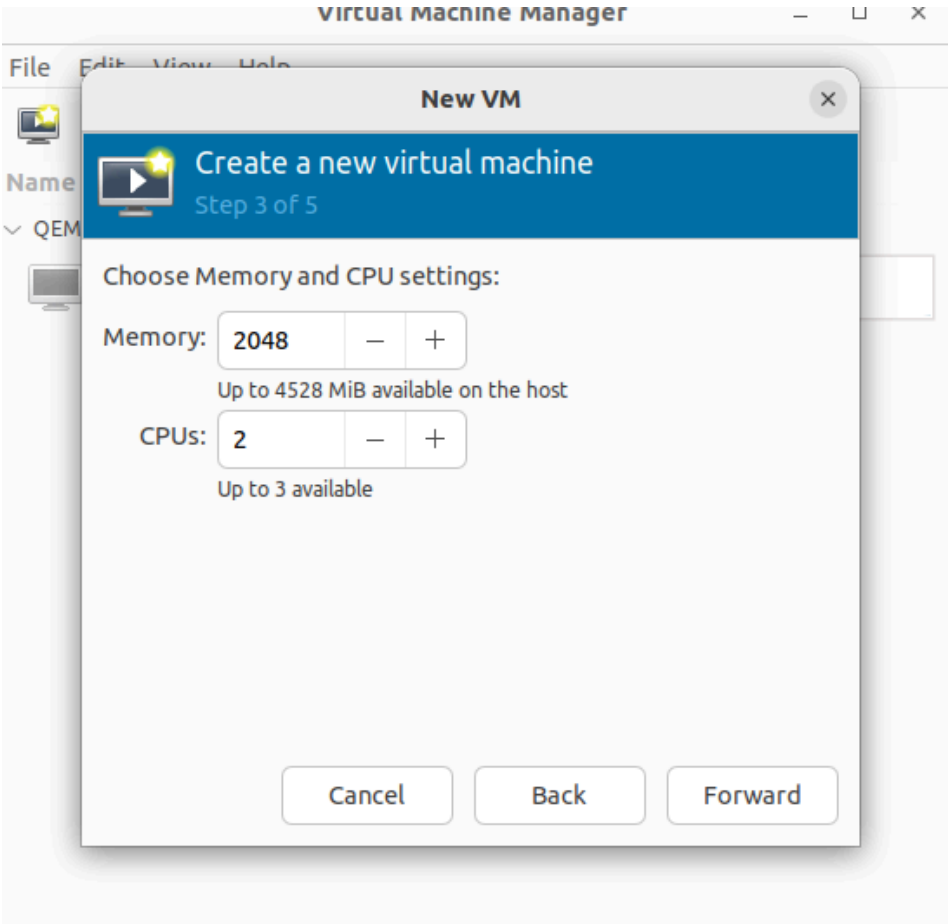
After clicking Forward this type of interface will show up and then click on Browse Local



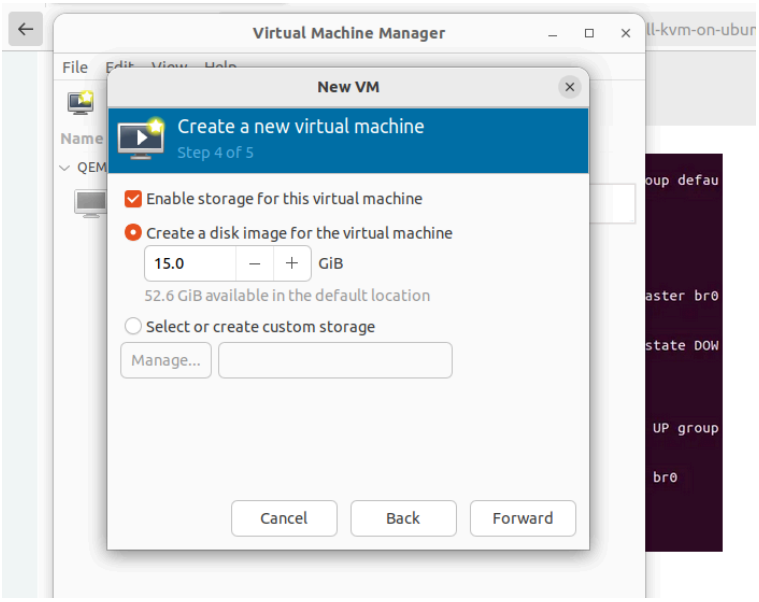
After clicking browse local this kind of interface (below) will show up and then we need give the ISO image and we need to click forward it will automatically detect the iso system.



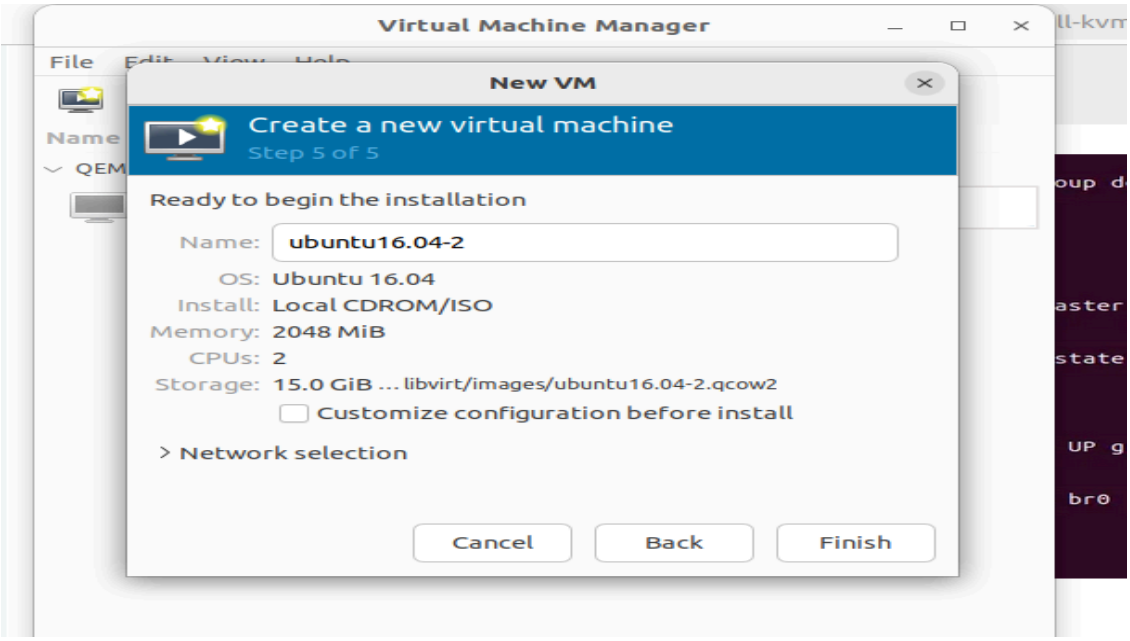
Then on the next we need to give memory and cpus to it and click to forward



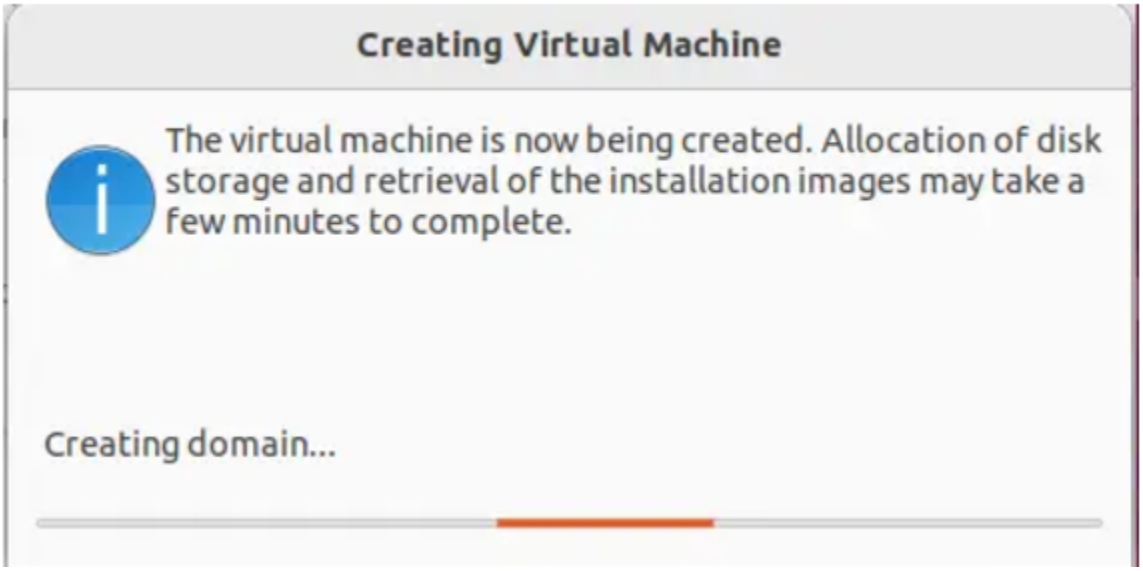
After that we need to allocate disk space for our Vm and click on forward.



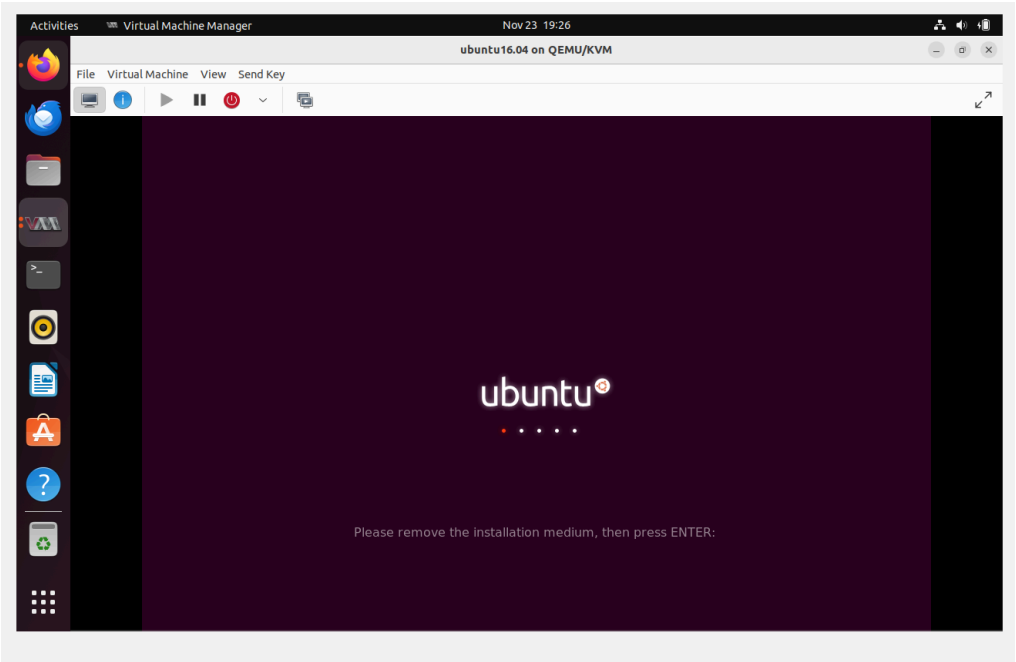
Finally it will show all the details on a interface like this and we can change our vm name as we like and finally we click to finish to install the VM



After clicking finish it will start creating the domain.

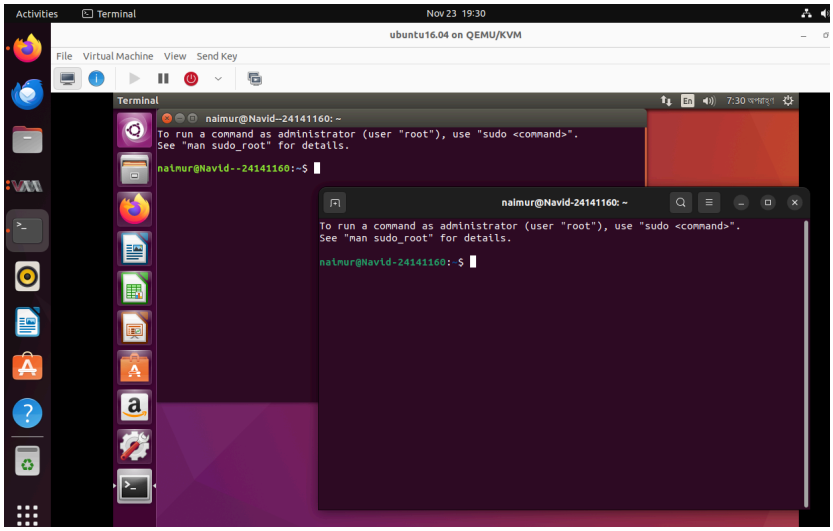


After installing it will directly move to the ubuntu setup. After setting up Ubuntu, finally the interface will show up.



Now if you need help with the Ubuntu setup then follow this link:
https://ubuntu.com/tutorials/install-ubuntu-desktop#7-create-your-login-details

Here im also attaching my two terminals image, One from virtual box and another from KVM



And thats How we install nested the VM using GUI.

Task-3

Create a kvm-based VM using “virt-install” cli. Check all the cli options and try it out.

In this part We have to create a KVM based Vm using “virt-install” cli.

First lets check all the cli for virt install and for that the command we will be using will be:

$ sudo virt-install --help

```
naimur@Navid-24141160:~$ sudo virt-install --help
[sudo] password for naimur:
usage: virt-install --name NAME --memory MB STORAGE INSTALL [opti
ons]

Create a new virtual machine from specified install media.

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --connect URI         Connect to hypervisor with libvirt URI

General Options:
  -n NAME, --name NAME  Name of the guest instance
  --memory MEMORY       Configure guest memory allocation. Ex:
                        --memory 1024 (in MiB)
                        --memory memory=1024,currentMemory=512
  --vcpus VCPUS         Number of vCPUs to configure for your gue
st. Ex:
                        --vcpus 5
                        --vcpus 5,maxvcpus=10,cpuset=1-4,6,8
                        --vcpus sockets=2,cores=4,threads=2
  --cpu CPU             CPU model and features. Ex:
                        --cpu coreduo,+x2apic
                        --cpu host-passthrough
                        --cpu host
```

We also need to check if the libvirt and its all dependencies are there and aslo need to ensure if virt-install is installed. To execute these we use commands :

$sudo apt update
$sudo apt install -y qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virt-manager virtinst
$sudo systemctl enable --now libvirtd

And now for installing the kvm based vm with cli we will use this command:

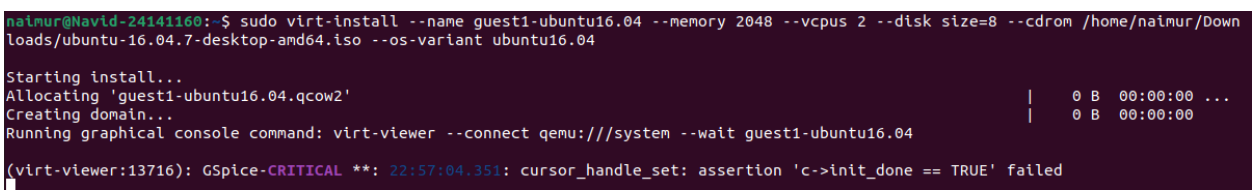
$sudo virt-install - -name guest1-ubuntu16.04 --memory 2048 -- vcpus 2 --disk size=8 --cdrom /home/naimur/Downloads/ubuntu-16.04.7-desktop-amd64.iso - -os-variant ubuntu16.04

By using this command we can easily create a kmv based Vm.

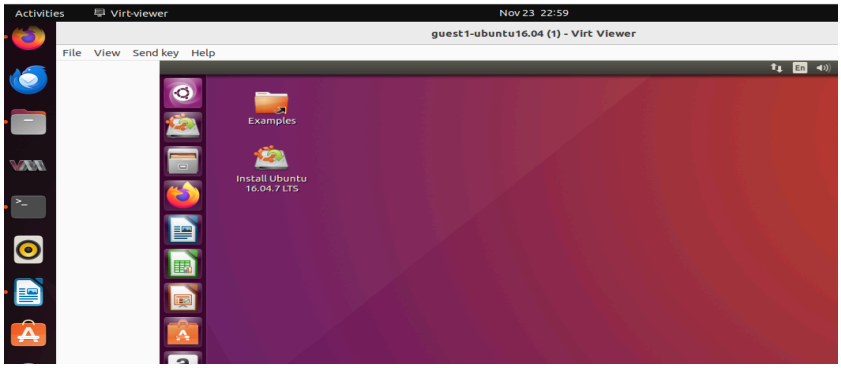
Explanation of the options:

- **--name guest1-ubuntu16.04:** Sets the name of the virtual machine.
- **--memory 2048:** Allocates 2GB of RAM to the VM.
- **--vcpus 2:** Allocates 2 virtual CPUs to the VM.
- **--disk size=8:** Creates a virtual disk with 8GB of storage.
- **--cdrom /home/naimur/Downloads/ubuntu-16.04.7-desktop-amd64.iso:** Specifies the path to the installation ISO for Ubuntu 16.04.
- **--os-variant ubuntu16.04:** Specifies the operating system variant (Ubuntu 16.04 in this case).

Complete set of commands and their explanation can be found here
https://manpages.ubuntu.com/manpages/focal/man1/virt-install.1.html

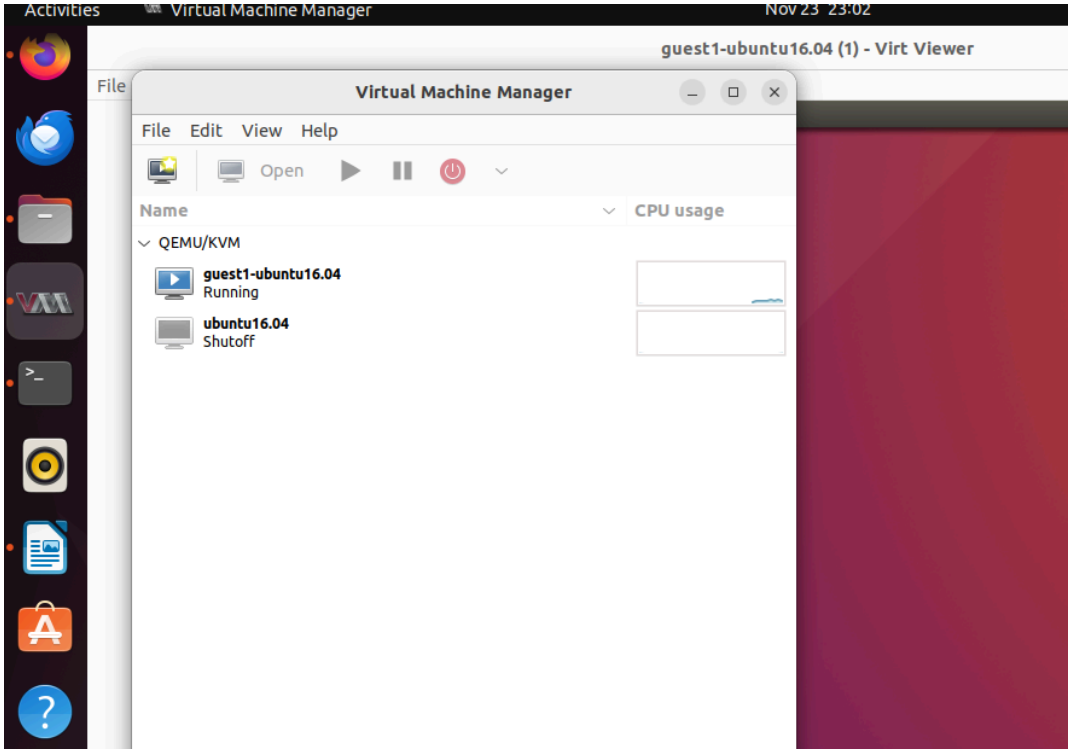


After executing the command a Virt viewer will automatically pop up and show the set up



Here i did not completely install Ubuntu 16.04 in kvm unlike the previous one. I'm using TRY Ubuntu only to show that it worked. If you want you can install the complete the ubuntu. For the guideline use the link
https://ubuntu.com/tutorials/install-ubuntu-desktop#7-create-your-login-details

Im also attaching the KVM to show the 2 VM that i installed



Task-4

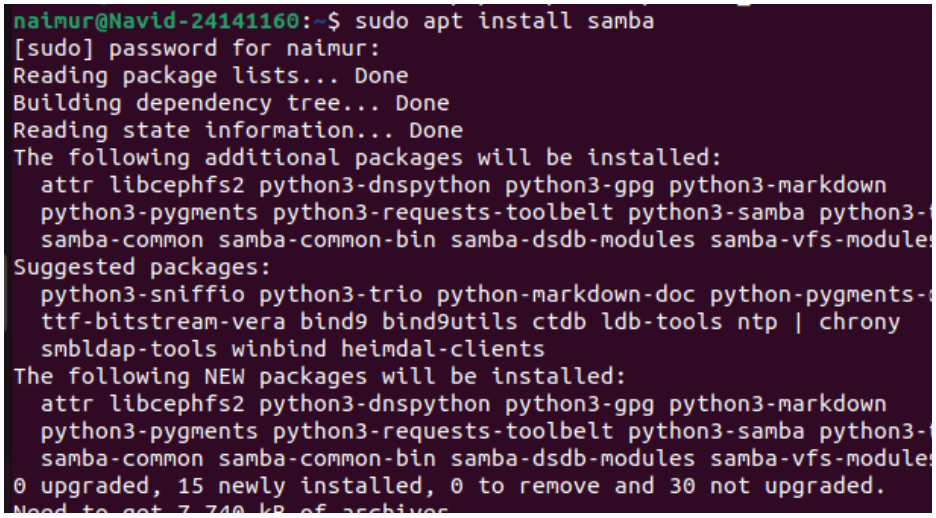
In this task its said we need to make documentation.

Task-5

Make a shared folder between host os and guest os using cli. If any changes happen in Guest OS then the same change should be reflected in the Host-OS. This must be applicable for vice versa.

For this part Im going to Use Samba package. As before Im going to use same Host os (ubuntu 22.04) and Guest os (16.o4) and with the help of Samba i will create a share_folder between them.
At first we need to update the apt and then install Samba on host os. The commands are:

$sudo apt update
$sudo apt install samba



```
naimur@Navid-24141160:~$ sudo apt install samba
[sudo] password for naimur:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  attr libcephfs2 python3-dnspython python3-gpg python3-markdown
  python3-pygments python3-requests-toolbelt python3-samba python3-
  samba-common samba-common-bin samba-dsdb-modules samba-vfs-module
Suggested packages:
  python3-sniffio python3-trio python-markdown-doc python-pygments-
  ttf-bitstream-vera bind9 bind9utils ctdb ldb-tools ntp | chrony
  smbldap-tools winbind heimdal-clients
The following NEW packages will be installed:
  attr libcephfs2 python3-dnspython python3-gpg python3-markdown
  python3-pygments python3-requests-toolbelt python3-samba python3-
  samba-common samba-common-bin samba-dsdb-modules samba-vfs-module
0 upgraded, 15 newly installed, 0 to remove and 30 not upgraded.
```

After that we will create the shared_folder. The commands are

$mkdir -p /home/naimur/shared_folder
$chmod 777 /home/naimur/shared_folder

The first command will create a directory and the second command will give the permissions for read and write.

```
naimur@Navid-24141160:~$ mkdir -p /home/naimur/shared_folder
naimur@Navid-24141160:~$ chmod 777 /home/naimur/shared_folder
```

Now we need to configure the samba file for that open the config file

$sudo nano /etc/samba/smb.conf

```
naimur@Navid-24141160:~$ sudo nano /etc/samba/smb.conf
```

And then write the bottom of the file this-

[shared_folder]
path = /home/naimur/shared_folder
browseable = yes
writable = yes
guest ok = yes
force user = naimur

Here instead of naimur you have to use your username for your os.

```
# admin users are members of.
# Please note that you also need to set appropriate Unix permissions
# to the drivers directory for these users to have write rights in it
;   write list = root, @lpadmin

[shared_folder]
path = /home/naimur/shared_folder
browseable = yes
writable = yes
guest ok = yes
force user = naimur

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

Press **clt+o** for save, save the file name by pressing enter and exit by **clt+x**

Then restart the Samba service

$sudo systemctl restart smbd

```
naimur@Navid-24141160:~$ sudo systemctl restart smbd
```

Allow Samba Through the Firewall (Optional)

$sudo ufw allow samba

```
naimur@Navid-24141160:~$ sudo ufw allow samba
Rules updated
Rules updated (v6)
```

Now check the ip address of your host

$ip addr show

```
naimur@Navid-24141160:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_code
oup default qlen 1000
    link/ether 08:00:27:19:f8:53 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixr
       valid_lft 83586sec preferred_lft 83586sec
    inet6 fd00::7eea:ebf6:a3a8:410c/64 scope global temporary dynar
       valid_lft 86386sec preferred_lft 14386sec
    inet6 fd00::6433:7ef2:bf29:14a5/64 scope global dynamic mngtmpa
oute
       valid_lft 86386sec preferred_lft 14386sec
    inet6 fe80::7221:9dce:3493:7c7/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Here my Ip address is 10.0.2.15. You need to get yous.

Now on the Guest server

Install Samba client

$sudo apt update

```
naimur@navid--24141160:~$ sudo apt update
[sudo] password for naimur:
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:2 http://bd.archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://bd.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:4 http://bd.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
194 packages can be upgraded. Run 'apt list --upgradable' to see them.
naimur@navid--24141160:~$
```

$sudo apt install cifs-utils

```
naimur@navid--24141160:~$ sudo apt install cifs-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  keyutils libldb1 libsmbclient libwbclient0 python-crypto python-ldb
  python-samba python-tdb samba-common samba-common-bin samba-libs
Suggested packages:
  smbclient winbind python-crypto-dbg python-crypto-doc heimdal-clients
The following NEW packages will be installed:
```

Then Mount the shared folder

$sudo mkdir -p /mnt/shared_folder

```
naimur@navid--24141160:~$ sudo mkdir -p /mnt/shared-folder
naimur@navid--24141160:~$
```

Breakdown of the command:

- **sudo**: Runs the command as a superuser (administrator). This is required because creating directories in system-wide directories like /mnt usually requires elevated privileges.
- **mkdir**: The command used to create a directory.
- **-p**: Ensures that the directory is created without error if it already exists, and also creates any necessary parent directories.
- **/mnt/shared_folder**: The directory path where you want to create the folder. /mnt is typically used for mounting storage devices or creating shared directories.

Now we need to Mount the Samba there

$sudo mount -t cifs //10.0.2.15/shared_folder /mnt/shared_folder -o guest,uid=$(id -u),gid=$(id -g)

Breakdown:

mount: The command used to attach a filesystem to a directory.

-t cifs: Specifies the type of filesystem to mount, in this case, **CIFS** (Common Internet File System), which is the protocol used for shared folders in Windows and Samba.

//host_ip_address/shared_folder: The network share you want to mount.

- host_ip_address: The IP address of the remote host where the shared folder is located (e.g., 192.168.1.100).
- shared_folder: The name of the shared folder on the remote machine.

/mnt/shared_folder: The local directory where the network share will be mounted. This should be an existing directory (e.g., /mnt/shared_folder), as specified in the previous mkdir command.

-o guest,uid=$(id -u),gid=$(id -g):

- **guest**: This option tells the system to mount the share with guest access (no username/password required). If the share requires credentials, you would replace guest with username=username,password=password.
- **uid=$(id -u)**: This sets the UID (User ID) of the user who will own the mounted files. $(id -u) dynamically fetches the current user's UID.
- **gid=$(id -g)**: This sets the GID (Group ID) of the user who will own the mounted files. $(id -g) dynamically fetches the current user's GID.

```
naimur@navid--24141160:~$ sudo mkdir -p /mnt/shared_folder
naimur@navid--24141160:~$ sudo mount -t cifs //10.0.2.15/shared_folder /mnt/shar
ed_folder -o guest,uid=$(id -u),gid=$(id -g)
naimur@navid--24141160:~$
```

Note: We need to be very careful. These commands are case sensitive and also be aware of the spaces used in the commands.

Now we are done with the creating . as for verify lets chk by

$ ls /mnt/shared_folder

Lets input a test text

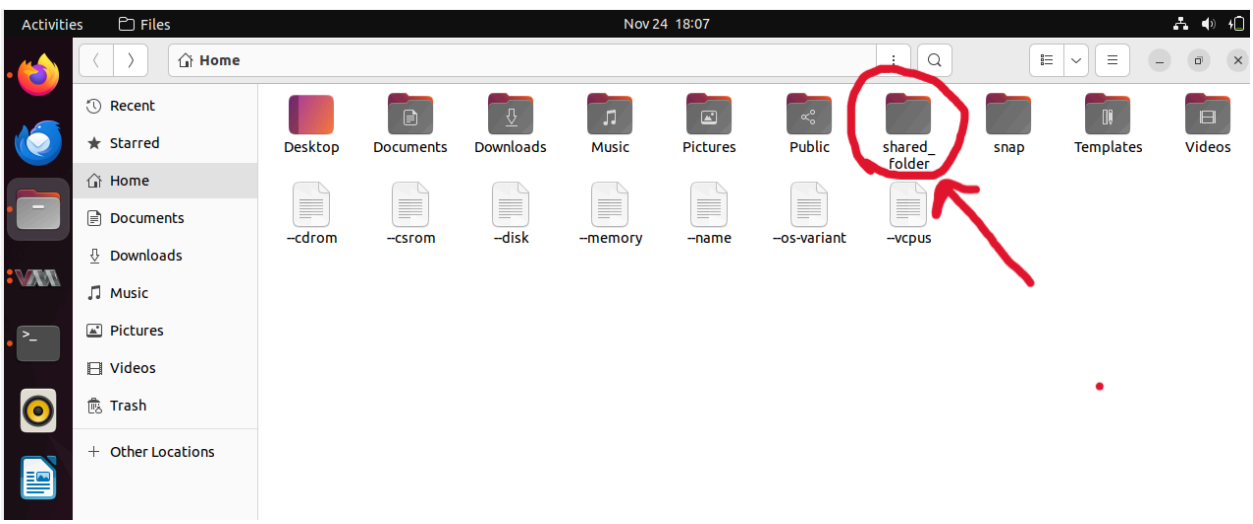
echo "Hello from Guest" > /mnt/shared_folder/test.txt

```
naimur@navid--24141160:~$ ls mnt/shared_folder
ls: cannot access 'mnt/shared_folder': No such file or directory
naimur@navid--24141160:~$ ls /mnt/shared_folder
naimur@navid--24141160:~$ echo "Hello from guest os"> /mnt/shared_folder/test.tx
t
naimur@navid--24141160:~$
```

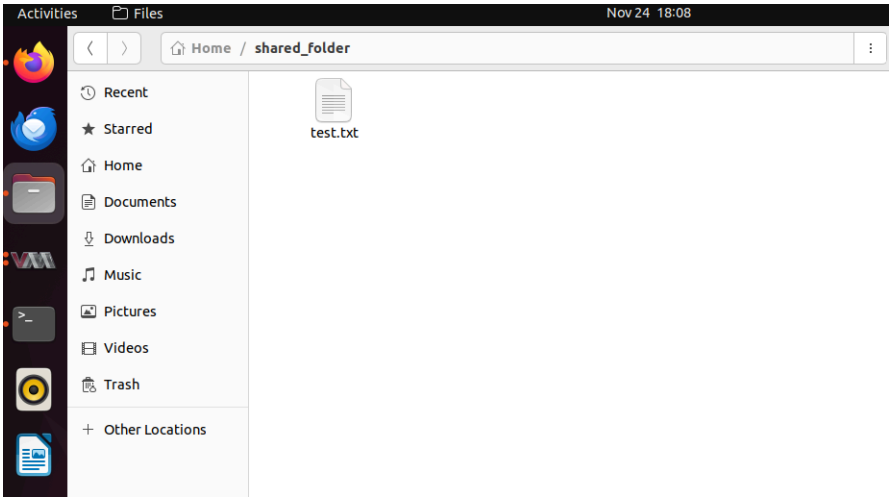
We also need to work on that on our Host os

```
naimur@Navid-24141160:~$ /mnt/shared_folder
bash: /mnt/shared_folder: No such file or directory
naimur@Navid-24141160:~$ cd /mnt/shared_folder
bash: cd: /mnt/shared_folder: No such file or directory
naimur@Navid-24141160:~$ sudo mkdir -p /mnt/shared_folder
[sudo] password for naimur:
naimur@Navid-24141160:~$ sudo chmod 777 /mnt/shared_folder
naimur@Navid-24141160:~$ ls -ld /mnt/shared_folder
drwxrwxrwx 2 root root 4096 Nov 24 18:32 /mnt/shared_folder
naimur@Navid-24141160:~$ sudo mount -t cifs //127.0.0.1/shared_folder /mnt/share
d_folder -o guest,uid=$(id -u),gid=$(id -g)
naimur@Navid-24141160:~$
```

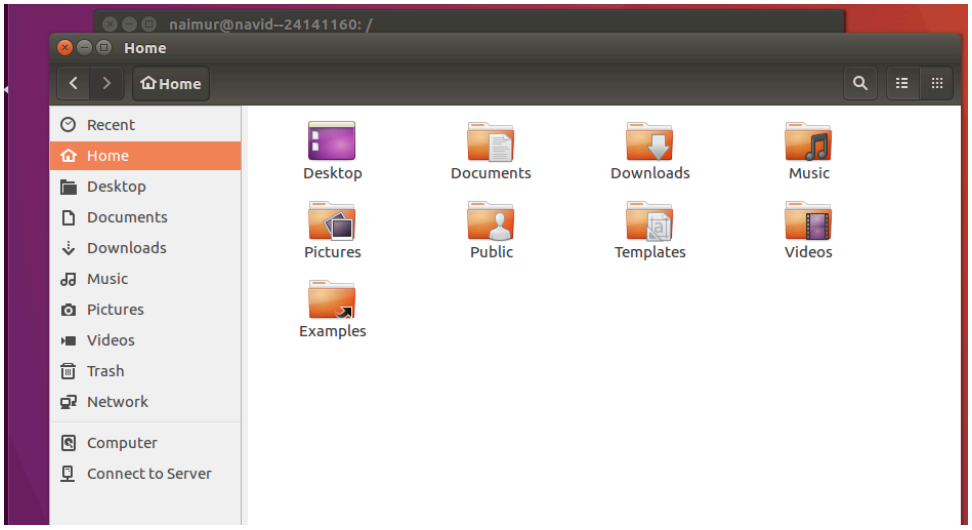
Now lets check our host if the folder was actually created



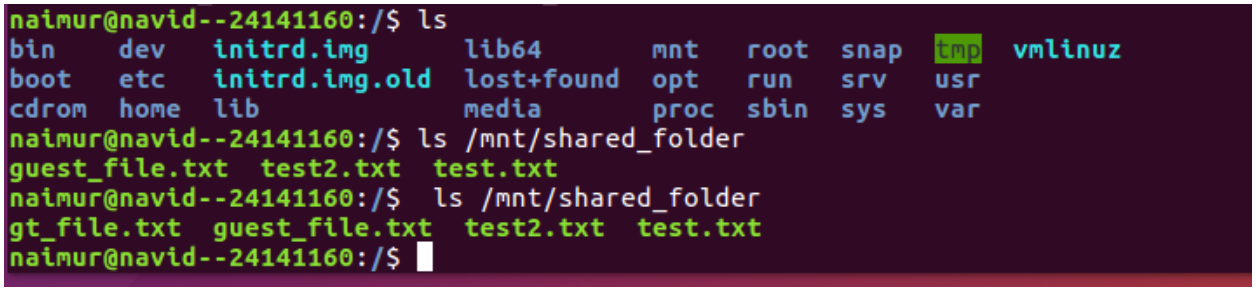
See it created the folder. Now what about the file that we wrote in our command panel, lets see if its inside or not



Now lets check out Guest os



Theres no shared folder showing because ins inside in /mnt.



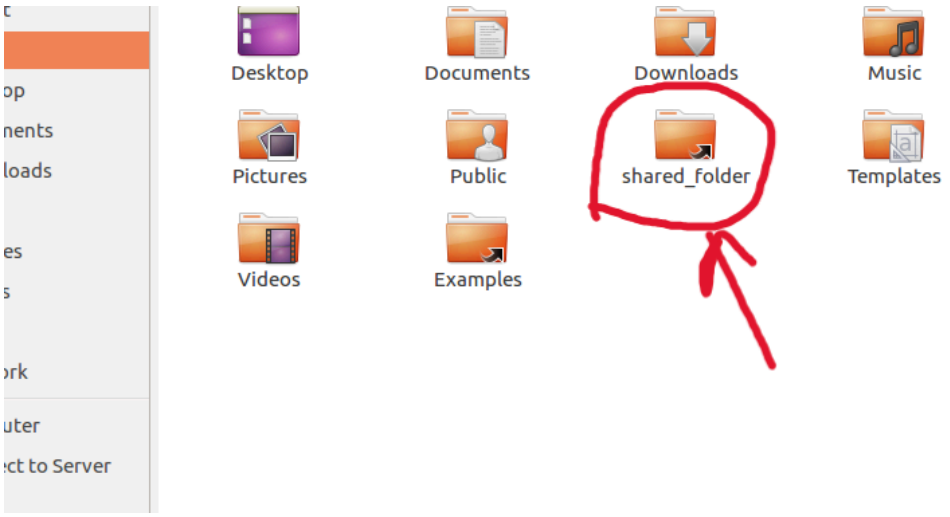
Note: there are 3 more file showing because i created 2 more for test purpose.

To show in here we can create a symbolic link to do that. To create a symbolic link we can use command-

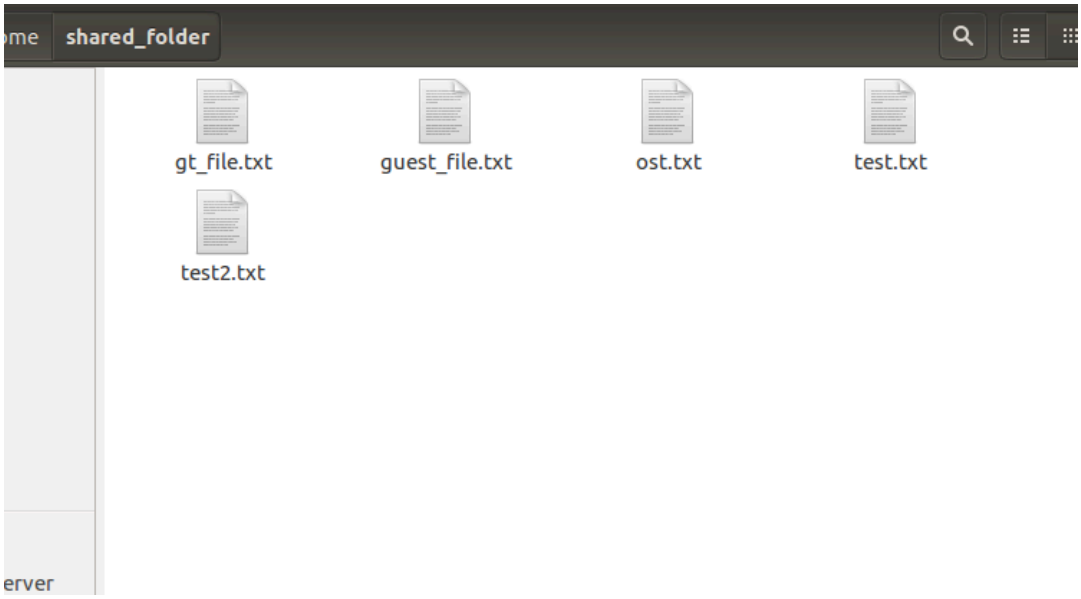
$ln -s /mnt/shared_folder /home/naimur/shared_folder

```
naimur@navid--24141160:~$ ln-s/mnt/shared_folder /home/naimur/shared_folder
bash: ln-s/mnt/shared_folder: No such file or directory
naimur@navid--24141160:~$ ln -s /mnt/shared_folder /home/naimur/shared_folder
naimur@navid--24141160:~$
```

Now thets check again



Now it works and if we click into it then we will find all those files



And thus We created a shared folder between host os and guest os

Task-6

Connect the guest OS with a phone and access the phone's files.

There are several methods to connect a phone to a guest (Ubuntu) operating system. Mounting MTP is one of them. By using GNOME Virtual File System (GVFS) we can access the phone file system, but the major drawback of this method is it's required a wired connection to perform this method we have to connect our phone with the pc with help of a usb connection.

On the other hand, to perform this task wirelessly we can use KDE Connect, which facilitates wireless communications and data transfer between devices over local networks through guest operating systems and phones.

In this section im going to show the MTP wired system.

Before doing the initial steps lets first update and upgrade our system

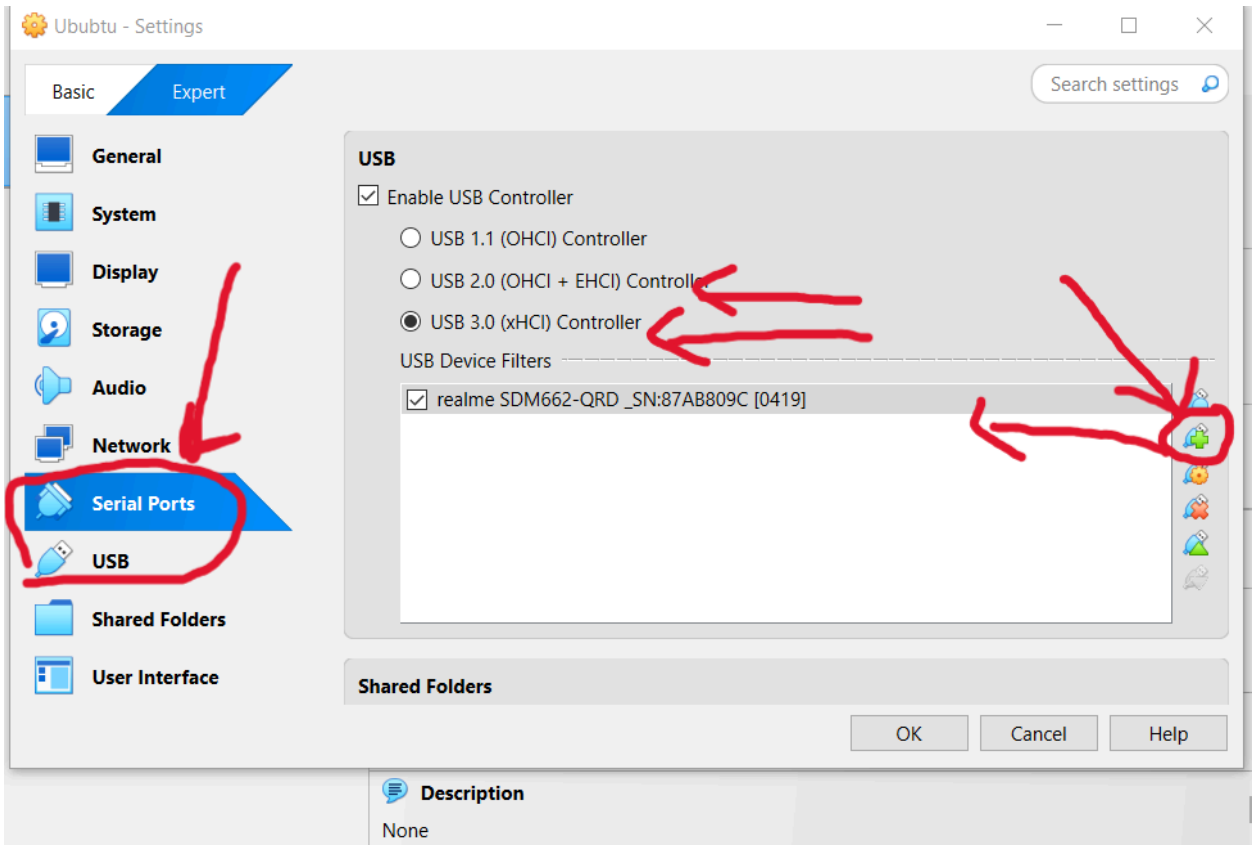
$sudo apt update
$sudo apt upgrade

```
E: Unable to locate package libmpt
haimur@Navid-24141160:~$ sudo apt update
sudo apt upgrade
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [1
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 D
a [43.2 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/restricted a
etadata [208 B]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd
adata [125 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/multiverse a
etadata [208 B]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
```

Then to connect our guest system with our phone we need to configure our virtual machine (in this case im using virtual machine).

Enable USB Controller

- Open VirtualBox.
- Select Ubuntu guest OS and go to **Settings > USB**.
- Enable the USB controller (choose USB 2.0 or USB 3.0 based on your phone's USB capabilities).



- Then click on shown USB + icon and choose your mobile device.
- Then click ok and reboot your vm machine

Now we need to open guest os terminal and install necessary mtp-tools

$ sudo apt install libmpt mtp-tools gvfs-backends

```
naimur@Navid-24141160:~$ sudo apt install libmtp mtp-tools gvfs-backends
[sudo] password for naimur:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package libmtp
```

Well we found error in this. The reason is libmtp is not support in ubuntu version 22.04

So instead we will write libmpt9 instead id libmpt

$ sudo apt install libmpt9 mtp-tools gvfs-backends

```
naimur@Navid-24141160:~$ sudo apt install libmtp9 mtp-tools gvfs-backends
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libmtp9 is already the newest version (1.1.19-1build1).
mtp-tools is already the newest version (1.1.19-1build1).
gvfs-backends is already the newest version (1.48.2-0ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
naimur@Navid-24141160:~$
```

It worked.

Now lets restart our system and connect our USB again then we will see a mobile icon in our guest os

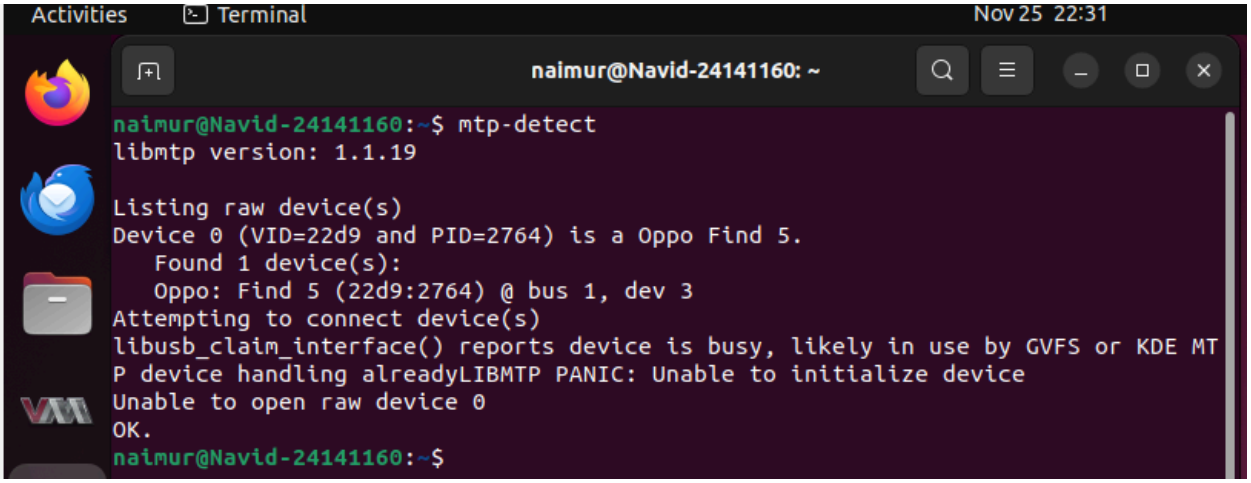
We can also see if our Virtual machine hets the connection by commanding

$lsusb

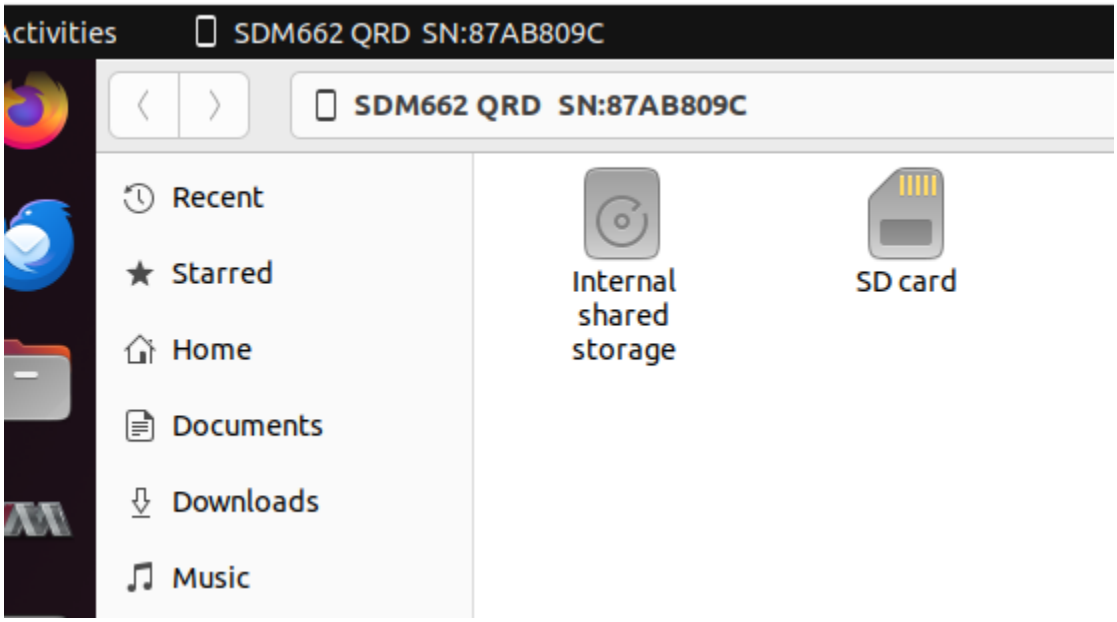
```
naimur@Navid-24141160:~$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 22d9:2765 OPPO Electronics Corp. Oppo N1
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
naimur@Navid-24141160:~$
```

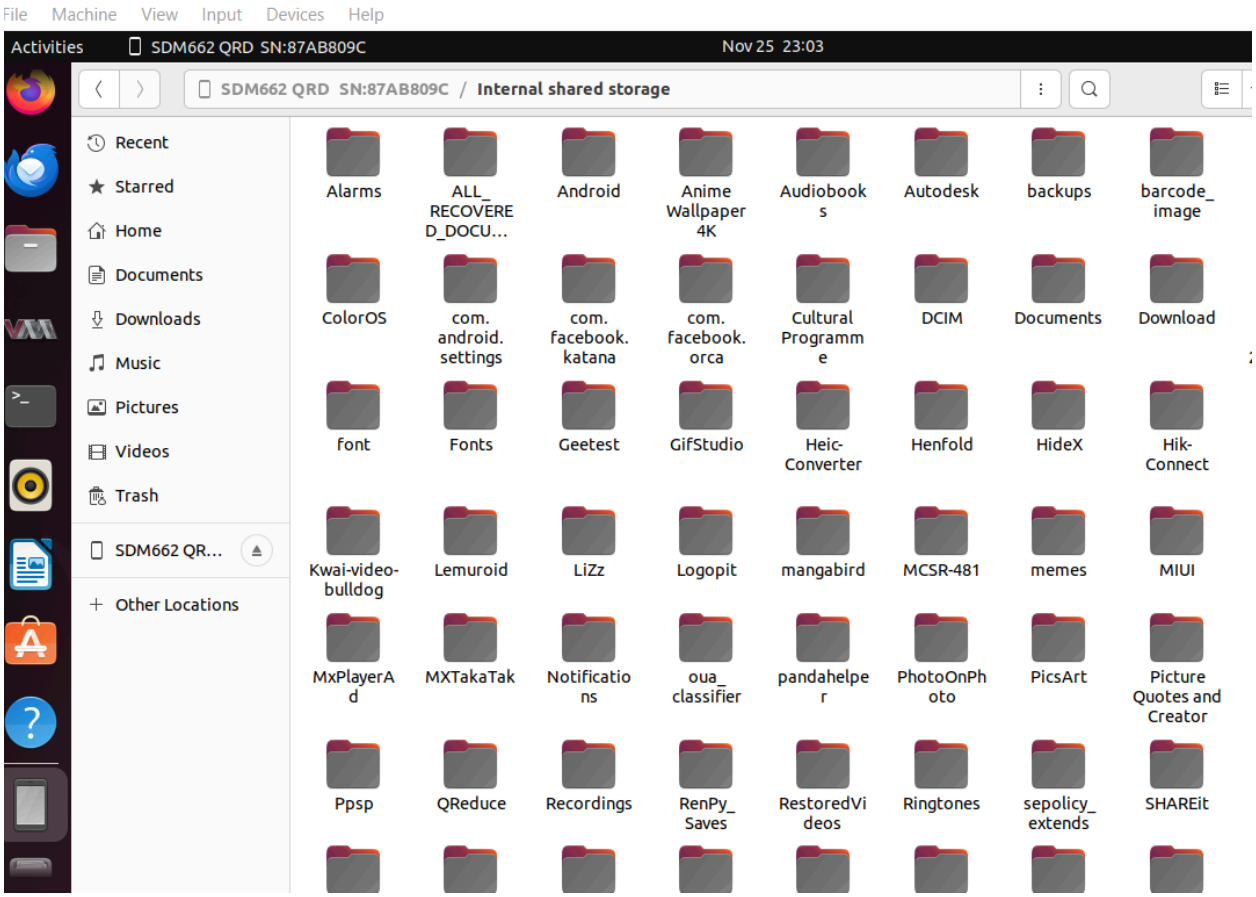
And if wee find our device in this then it means its connected. We can also check by commanding

$mtp-detect



Nice. That means our connection is successful. Now lets click on mobile icon and access phones files.





And thats how we connect our guest os with our phone.
If incase if you cant access the phone then you need to configure on your phone. You need to enable File Transfer option when you plug the USB in your phone. If you still cant then go to settings and enable the developer mode and go to developer action and enable MTP or File Transfer option.

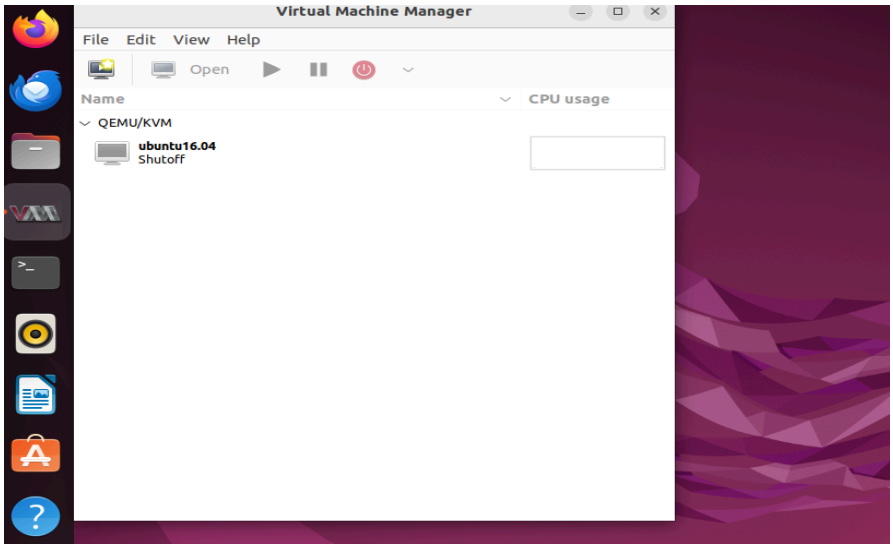
Now we can also connect our phone wireless. To perform this task wirelessly we can use KDE Connect, which facilitates wireless communications and data transfer between devices over local networks through guest operating system and phone.

Task-7

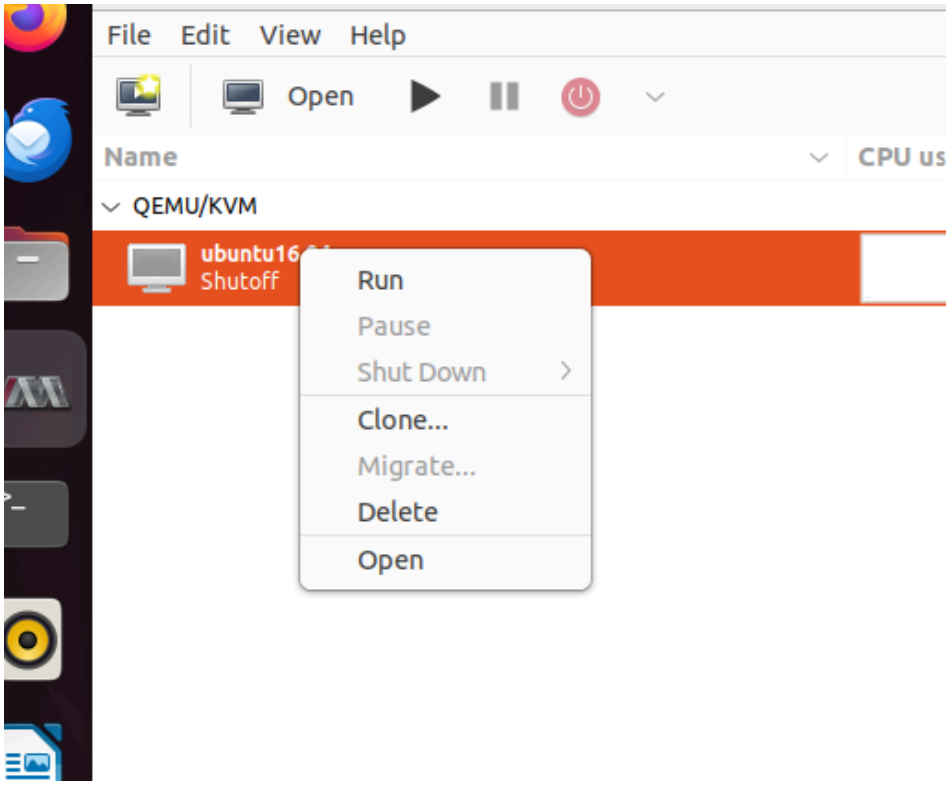
Clone a VM using GUI and using a kvm-based command.

We can clone a vm using virtual machine manager (vmm) using it's graphical user interface (GUI).

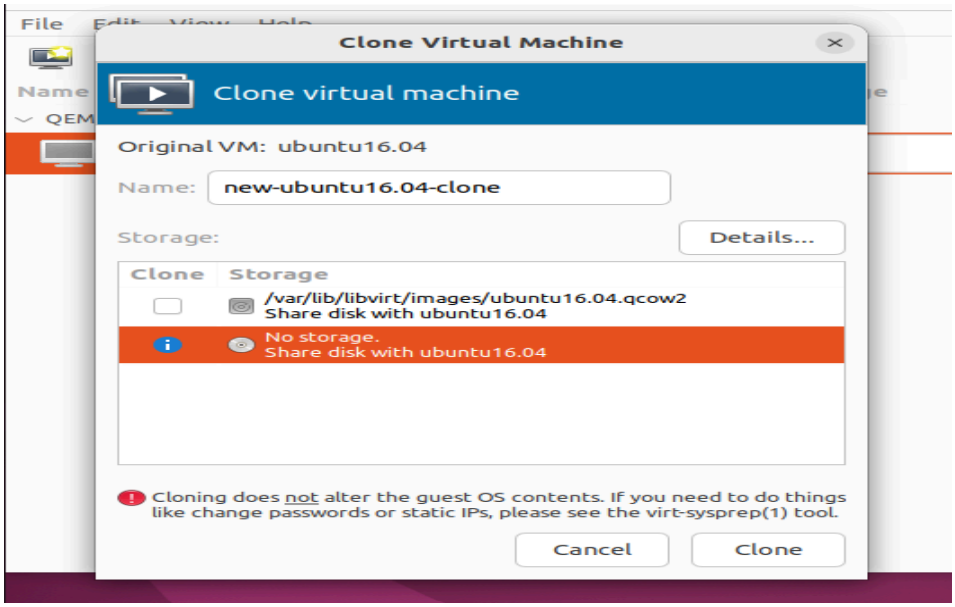
Now to do this, lets first open our Vm(KVM)

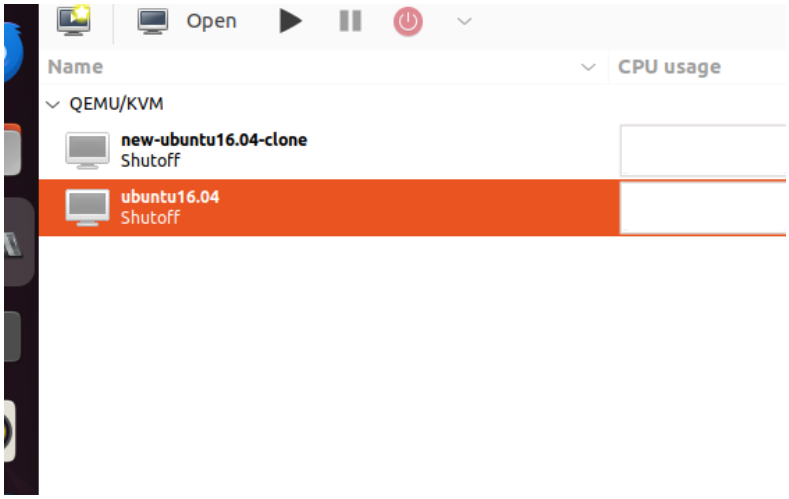


Now right click on our guest os and we will see clone option. Click on that



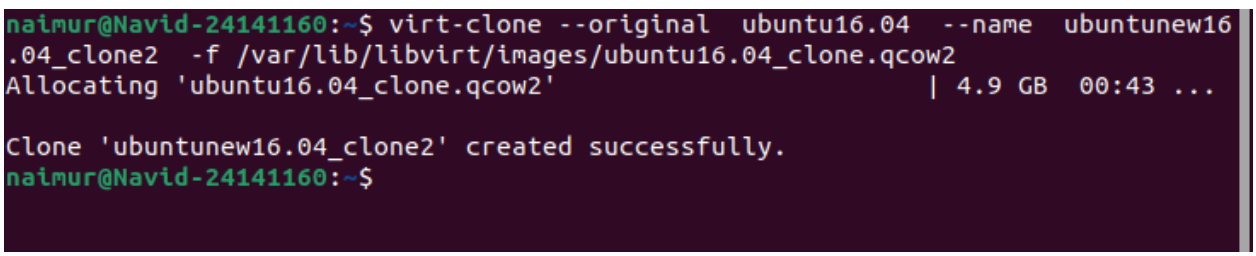
Now lets give a new name and click on clone. It will automatically create a clone.

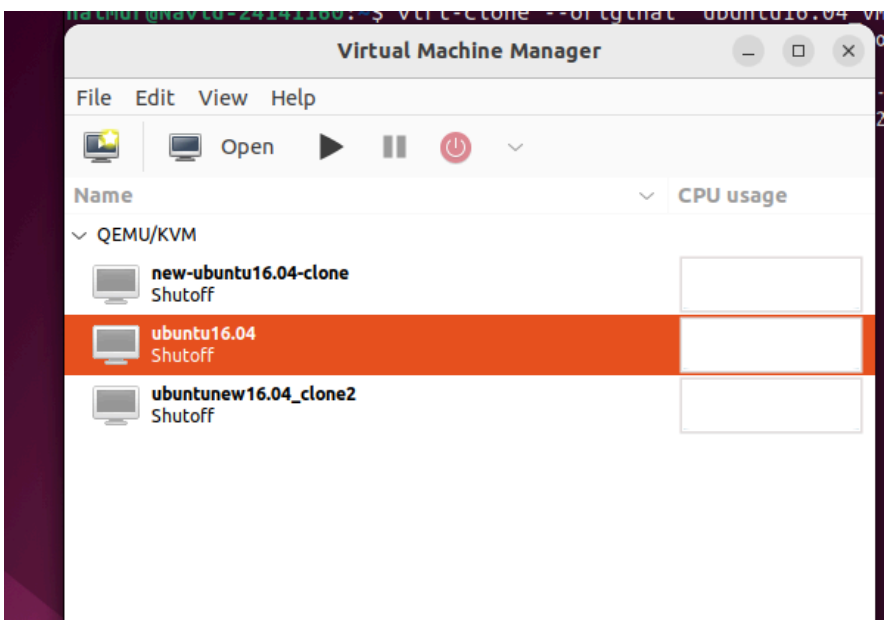




We can also create the clone by using a single command

$virt-clone --original ubuntu16.04 --name ubuntunew16,04_clone2 -f /var/lib/libvirt/images/ubuntu16.04_clone.qcow2



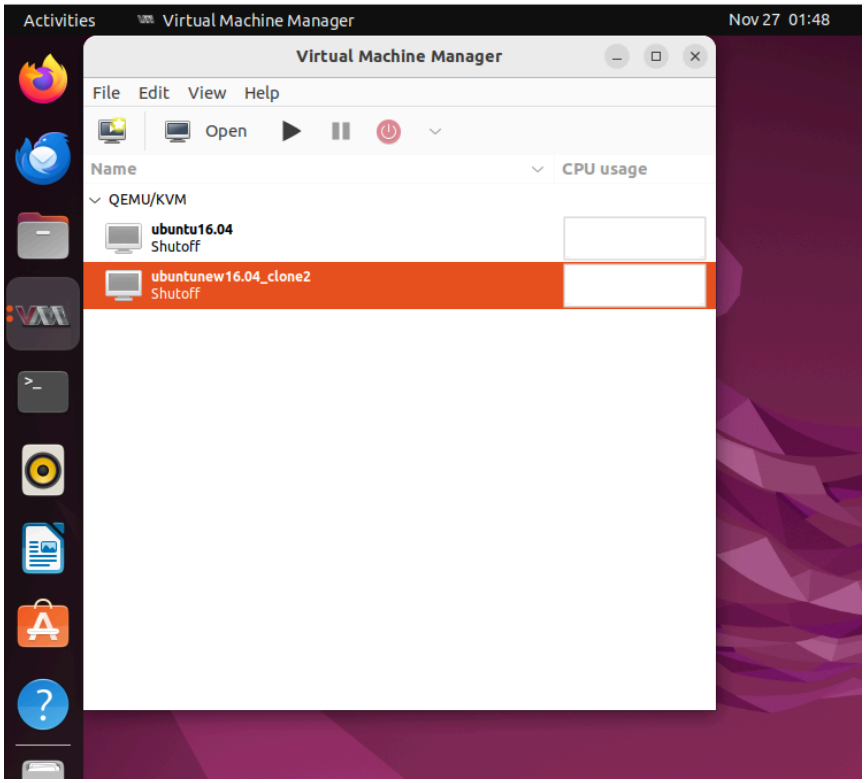


As we can see with the help of that command we have easily created another clone.

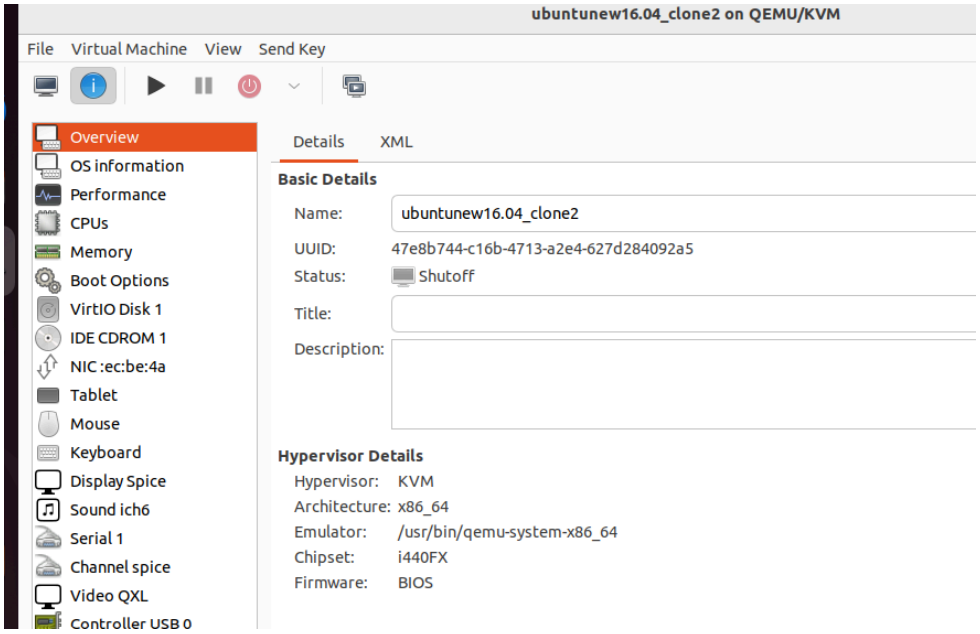
Task -8

Add two hard disks in a new cloned virtual machine using GUI.

First lets select our clone vm from kvm.

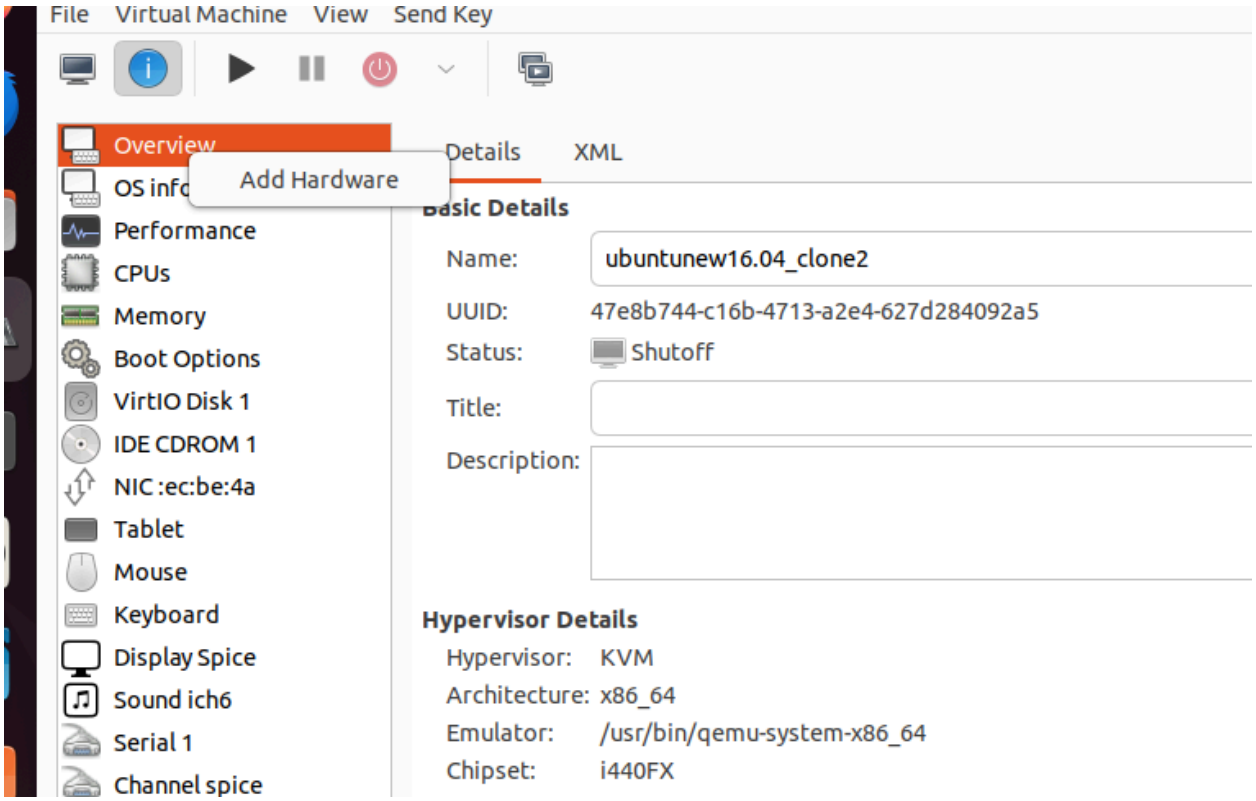


Now if we go to the about section (blue i icon) we can see all the details like given in below image.

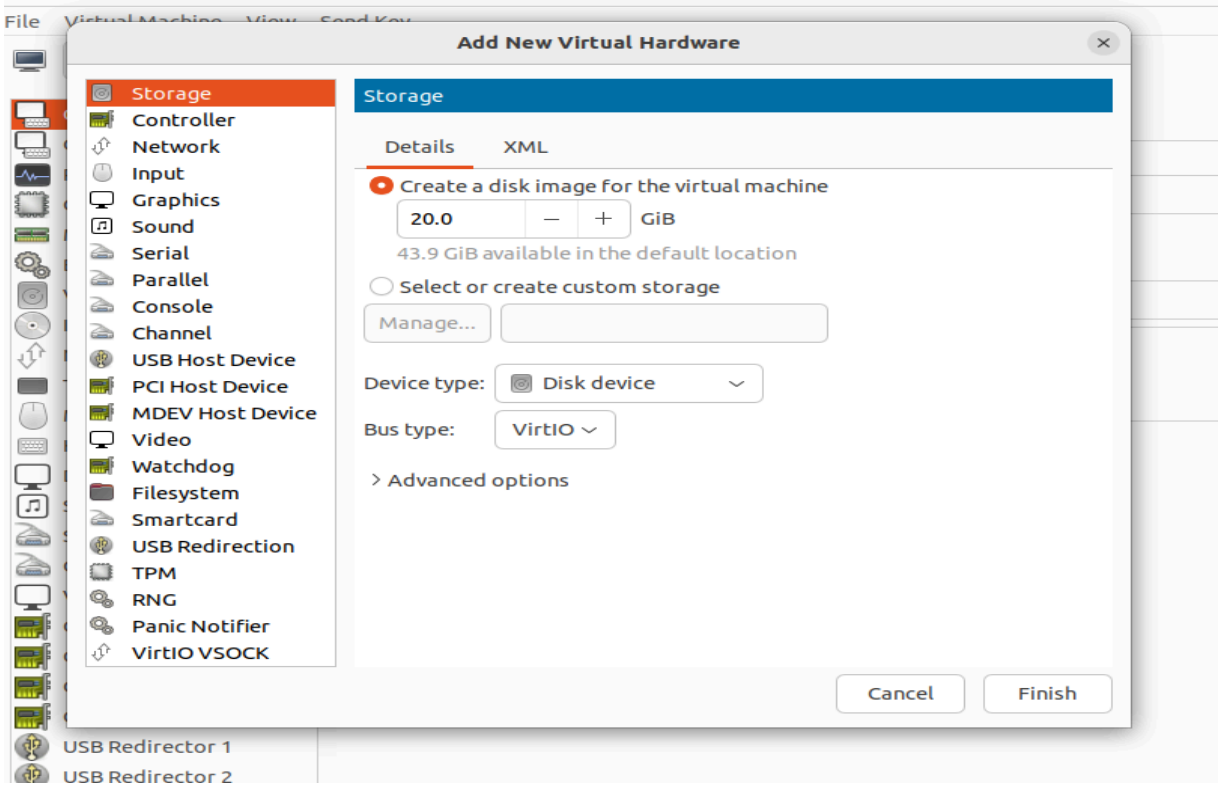


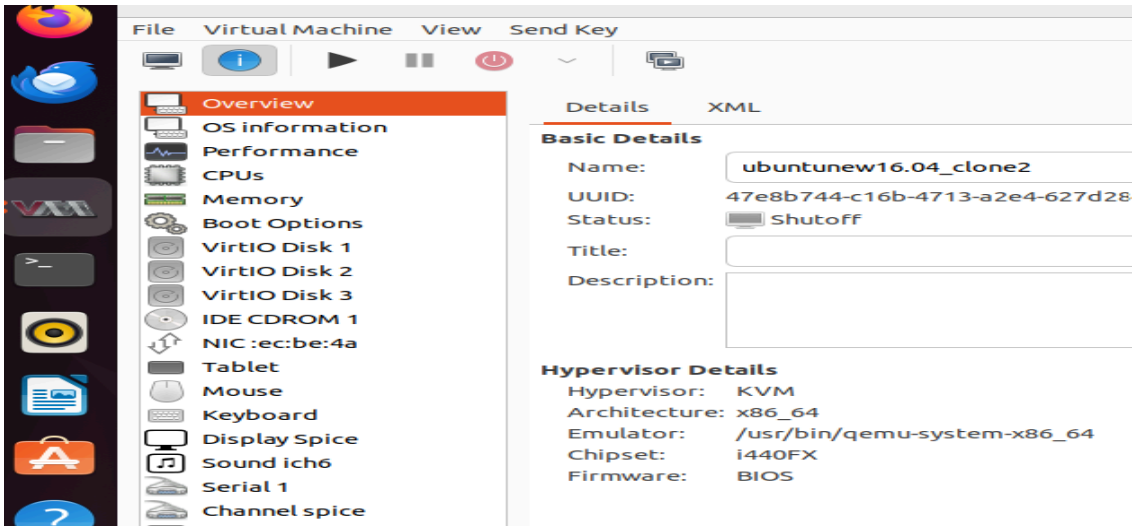
Here we can see our Vm has currently one virtual disk. So now our target is to create and two new virtual storage disk.

First we need to right click of the mouse on the overview section then we will find the option add hardware and we have to click on that



Now we need to give the required storage Gb and click on finish. It will automatically create a virtual storage disk. We just need to create one more and we are done creating two virtual disks using GUI in our VM.





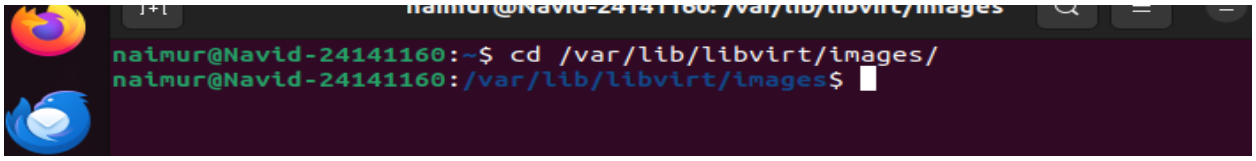
Task-9

Add two hard disks in a new cloned virtual machine using a kvm-based command.

In this part we need to add two virtual disk using KVm based command.

First lets open our terminal and we have to create a new disk image. For that we have to go to the kvm default directory by using this command

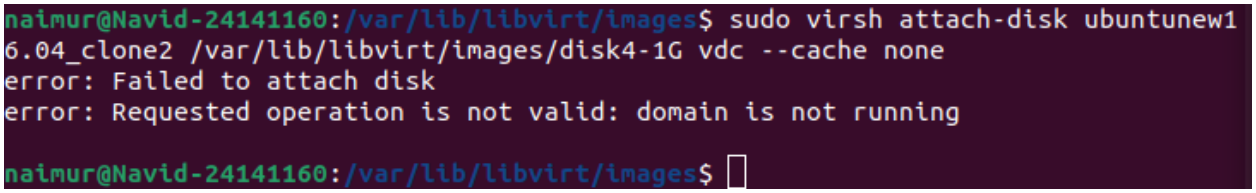
$ **cd /var/lib/libvirt/images/**



Then to create a new disk image of qcow2 format have to write the following command

$sudo qemu-img create -f qcow2 disk4-1G 1G

Here we are creating a 1GB storage size with the name of disk4.



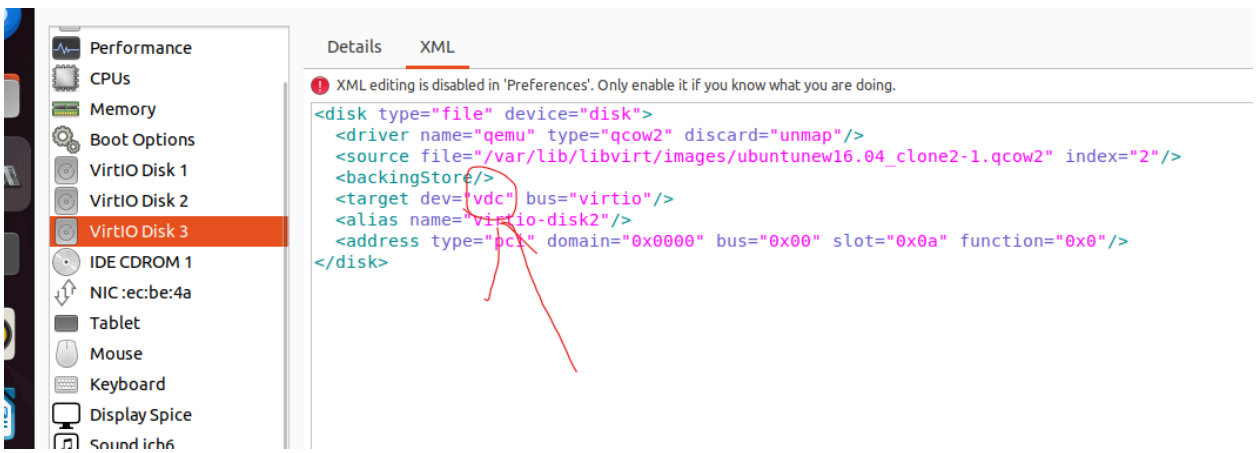
The error is showing because to create we need to power on and run the vm. After that it will create the disk

```
naimur@Navid-24141160:/var/lib/libvirt/images$ sudo qemu-img create -f qcow2 dis
k4-1G 1G
[sudo] password for naimur:
Formatting 'disk4-1G', fmt=qcow2 cluster_size=65536 extended_l2=off compression_
type=zlib size=1073741824 lazy_refcounts=off refcount_bits=16
naimur@Navid-24141160:/var/lib/libvirt/images$
```

Now, we have to attach the disk to the virtual machine for this run the following command

$ sudo virsh attach-disk ubuntu20.04_clone /var/lib/libvirt/images/disk4-6G vdc --cache none

Here our virtual clone machine name is ubuntunew16.04_clone2 and we added the path that we created and define the dev type as vdd, because we have three another disk which are vda, vdb and vdc.

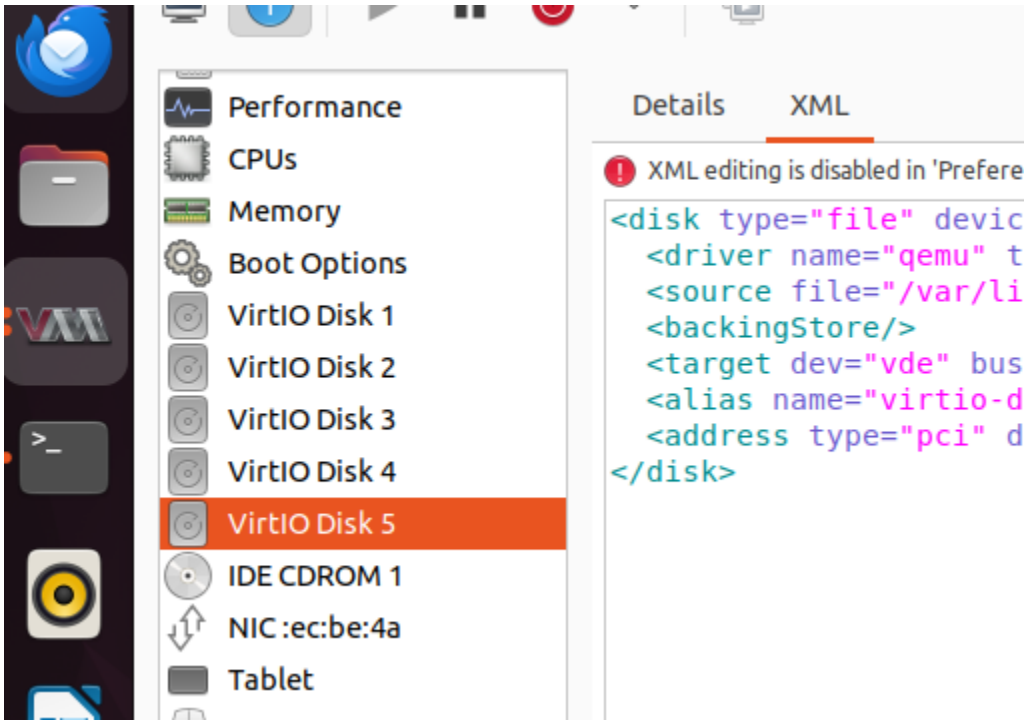



```
naimur@Navid-24141160:/var/lib/libvirt/images$ sudo virsh attach-disk ubuntunew1
6.04_clone2 /var/lib/libvirt/images/disk4-1G vdd --cache none
Disk attached successfully

naimur@Navid-24141160:/var/lib/libvirt/images$
```

Now the disk attach is successful.

Now all we need to do is to create another one as the same process just will have to give the type vde and it will create and add successfully.

```
naimur@Navid-24141160:/var/lib/libvirt/images$ sudo qemu-img create -f qcow2 dis
k5-1G 1G
Formatting 'disk5-1G', fmt=qcow2 cluster_size=65536 extended_l2=off compression_
type=zlib size=1073741824 lazy_refcounts=off refcount_bits=16
naimur@Navid-24141160:/var/lib/libvirt/images$ sudo virsh attach-disk ubuntunew1
6.04_clone2 /var/lib/libvirt/images/disk4-1G vde --cache none
error: Failed to attach disk
error: internal error: unable to execute QEMU command 'device_add': Failed to ge
t "write" lock

naimur@Navid-24141160:/var/lib/libvirt/images$ sudo virsh attach-disk ubuntunew1
6.04_clone2 /var/lib/libvirt/images/disk5-1G vde --cache none
Disk attached successfully
```



And thus we create and attach 2 disk using KVM command.

Task-10

How to migrate a VM to another host? Show step-by-step commands and output about migrating any of your VMs. Migrate your VM to your friend's host machine and vice versa. Must show the output after successful migration.

Steps to Migrate a VM Installed via Virt Manager to Another Host

Copy the VM Image to a New Directory

First, create a directory on your current machine to store the VM image temporarily:

$mkdir ~/Desktop/Migrate_VM

```
naimur@Navid-24141160:~$ mkdir ~/Desktop/Migrate_VM
```

Next, copy the .qcow2 VM disk image from the default KVM/libvirt directory to this new directory:

$sudo cp /var/lib/libvirt/images/ubuntu16.04_VM.qcow2 ~/Desktop/Migrate_VM/

```
naimur@Navid-24141160:~$ sudo cp /var/lib/libvirt/images/ubuntu16.04.qcow2 ~/Desktop/Migrate_VM/
naimur@Navid-24141160:~$
```

Change Directory Permissions

Change the permissions of the directory so that the VM image file can be accessed without restrictions:

$sudo chmod -R a+rwx ~/Desktop/Migrate_VM

```
naimur@Navid-24141160:~$ sudo chmod -R a+rwx ~/Desktop/Migrate_VM
naimur@Navid-24141160:~$
```

Transfer the .qcow2 File to the New Host

Use a secure file transfer method like scp to copy the .qcow2 file to the new host:

$scp ~/Desktop/Migrate_VM/ubuntu16.04_VM.qcow2 naimur@Navid-24141160: /home/naimur/Migrate_VM

```
naimur@Navid-24141160:~$ scp ~/Desktop/Migrate_VM/ubuntu16.04.qcow2 naimur@Navid-24141160:/home/naimur/Mig
rate_VM
naimur@navid-24141160's password:
ubuntu16.04.qcow2                                             100%   15GB 120.5MB/s   02:07
naimur@Navid-24141160:~$
```

Install the VM on the New Host

On the new host machine, use **Virt Manager** or **VirtualBox** to install the migrated VM:

1. Open Virt Manager or VirtualBox.
2. Select the option to create a new VM.
3. Import the existing disk image from the directory where the .qcow2 file was copied.
4. Provide the required resources (e.g., memory size, CPU count).
5. Click "Finish."

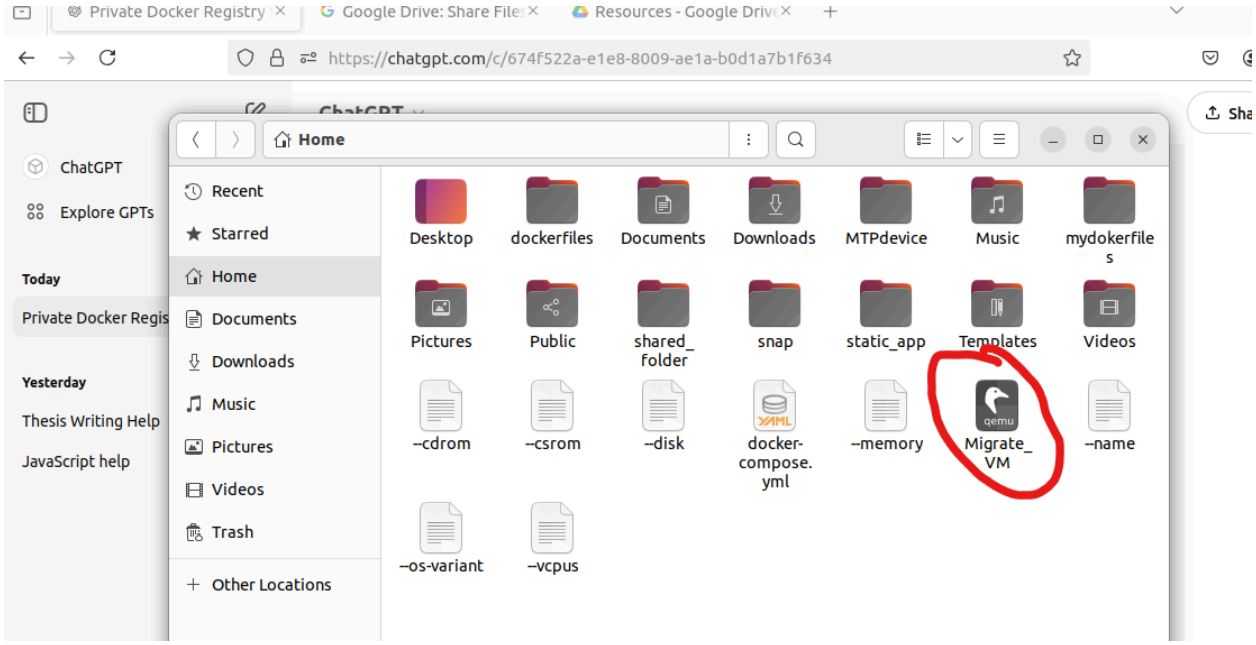
For transfer use google drive or portable device. The new VM will retain the same configurations as the original VM because it uses the same disk image.

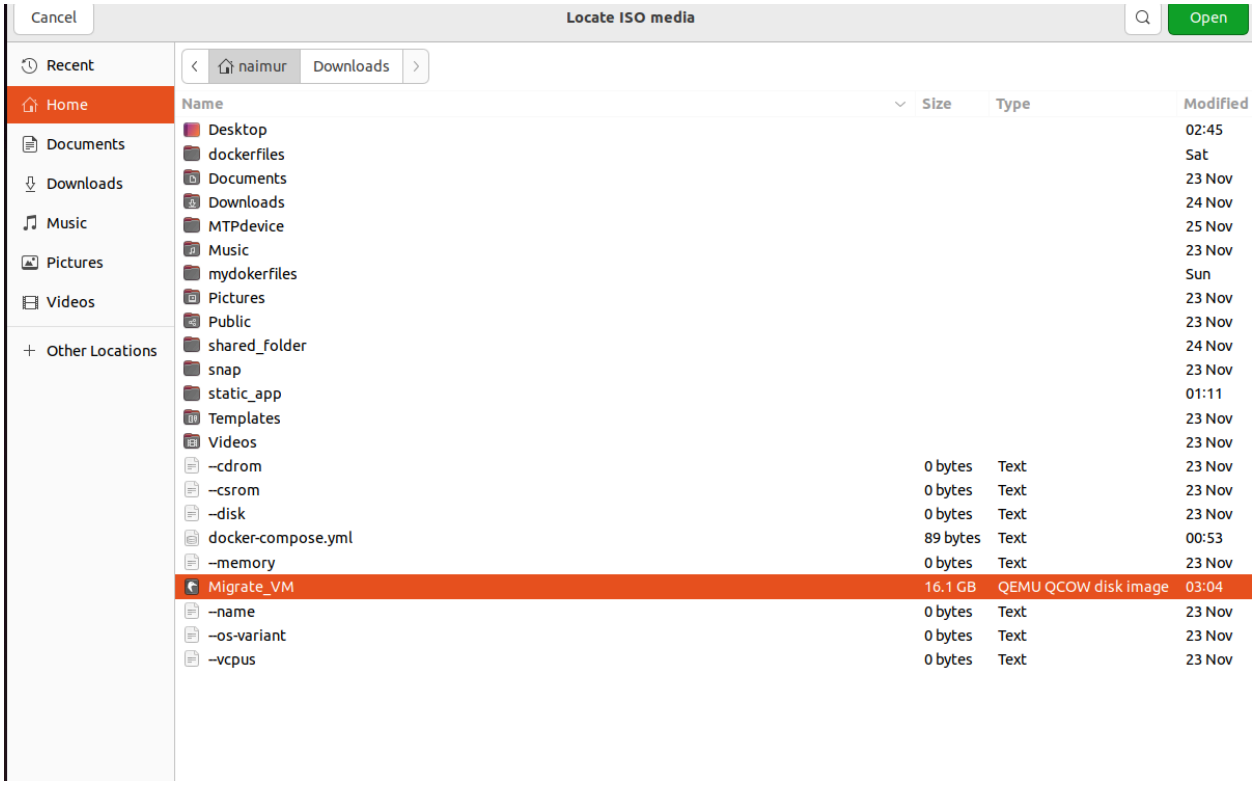
Start the Migrated VM

Start the VM on the new host machine. You should now be able to operate the migrated VM exactly as it was on the original host.

Reverse Migration (Optional)

If needed, you can follow the same steps in reverse to migrate the VM back to the original host.


Private Docker Registry
Google Drive: Share File
Resources - Google Drive
https://chatgpt.com/c/674f522a-e1e8-8009-ae1a-b0d1a7b1f634
ChatGPT
Explore GPTs
Today
Private Docker Regis
Yesterday
Thesis Writing Help
JavaScript help
Home
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
Desktop
dockerfiles
Documents
Downloads
MTPdevice
Music
mydokerfiles
Pictures
Public
shared_folder
snap
static_app
Templates
Videos
--cdrom
--csrom
--disk
docker-compose.yml
--memory
Migrate_VM
--name
--os-variant
--vcpus


Cancel
Locate ISO media
Open
Recent
Home
Documents
Downloads
Music
Pictures
Videos
Other Locations
naimur
Downloads
Name
Size
Type
Modified
Desktop 02:45
dockerfiles Sat
Documents 23 Nov
Downloads 24 Nov
MTPdevice 25 Nov
Music 23 Nov
mydokerfiles Sun
Pictures 23 Nov
Public 23 Nov
shared_folder 24 Nov
snap 23 Nov
static_app 01:11
Templates 23 Nov
Videos 23 Nov
--cdrom 0 bytes Text 23 Nov
--csrom 0 bytes Text 23 Nov
--disk 0 bytes Text 23 Nov
docker-compose.yml 89 bytes Text 00:53
--memory 0 bytes Text 23 Nov
Migrate_VM 16.1 GB QEMU QCOW disk image 03:04
--name 0 bytes Text 23 Nov
--os-variant 0 bytes Text 23 Nov
--vcpus 0 bytes Text 23 Nov