

# Assignment 02



**Abir Ahammed Bhuiyan**

**ID: 20101197**

Dept. of Computer  
Science and Engineering

**Section : 01**

**Subject : CSE484**

**Semester : Spring-2024**

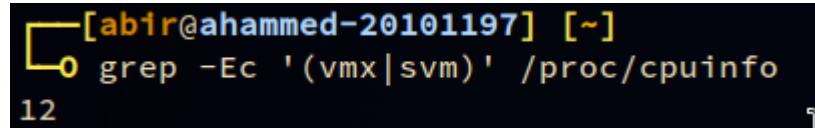
**Date : 15.02.2024**

# 1. Install KVM

## Check virtualization is enabled or not

```
grep -Ec '(vmx|svm)' /proc/cpuinfo
```

If the output of this command is more than 0 that means virtualization is enabled in our computer.



```
[abir@ahammed-20101197] ~
└─ grep -Ec '(vmx|svm)' /proc/cpuinfo
    12
```

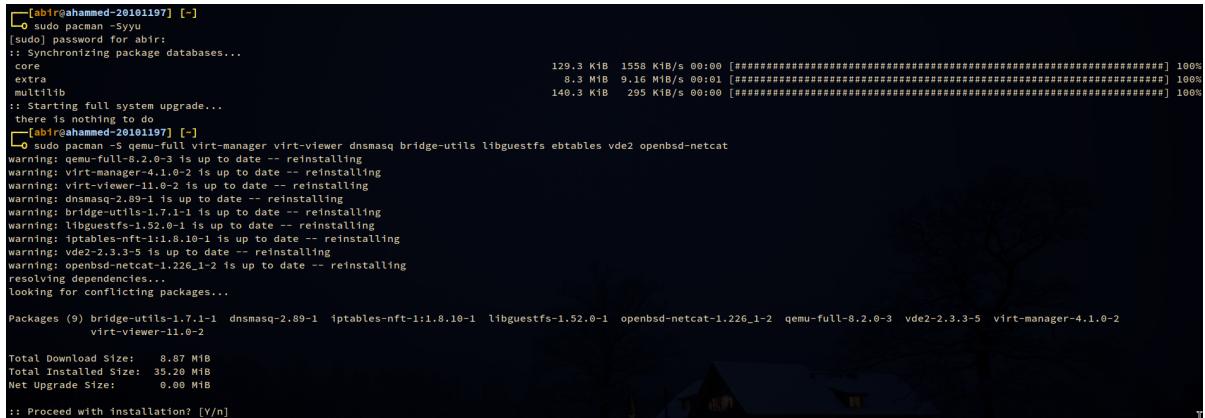
## Install the Required KVM Packages

Updating the existing packages in the `Arch Linux` system.

```
sudo pacman -Syyu
```

By running the below command will install the required packages to run `KVM` in our system.

```
sudo pacman -S qemu-full virt-manager virt-viewer dnsmasq bridge-utils libguestfs
ebtables vde2 openbsd-netcat
```



```
[abir@ahammed-20101197] ~
└─ sudo pacman -Syyu
[sudo] password for abir:
:: Synchronizing package databases...
core                                              129.3 KiB  158 KiB/s 00:00 [#####
extra                                             8.3 MiB  9.16 MiB/s 00:01 [#####
multilib                                         140.3 KiB  295 KiB/s 00:00 [#####
:: Starting full system upgrade...
there is nothing to do
[abir@ahammed-20101197] ~
└─ sudo pacman -S qemu-full virt-manager virt-viewer dnsmasq bridge-utils libguestfs ebttables vde2 openbsd-netcat
warning: qemu-full-8.2.0-3 is up to date -- reinstalling
warning: virt-manager-4.1.0-2 is up to date -- reinstalling
warning: virt-viewer-11.0-2 is up to date -- reinstalling
warning: dnsmasq-2.89-1 is up to date -- reinstalling
warning: bridge-utils-1.7.1-1 is up to date -- reinstalling
warning: libguestfs-1.52.0-1 is up to date -- reinstalling
warning: iptables-nft-1:1.8.10-1 is up to date -- reinstalling
warning: vde2-2.3.3-5 is up to date -- reinstalling
warning: openbsd-netcat-1.226_1-2 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...
Packages (9) bridge-utils-1.7.1-1 dnsmasq-2.89-1 iptables-nft-1:1.8.10-1 libguestfs-1.52.0-1 openbsd-netcat-1.226_1-2 qemu-full-8.2.0-3 vde2-2.3.3-5 virt-manager-4.1.0-2
virt-viewer-11.0-2

Total Download Size: 8.87 MiB
Total Installed Size: 35.20 MiB
Net Upgrade Size: 0.00 MiB

:: Proceed with installation? [Y/n]
```

## Configure the libvirtd Service

Start and enable `libvirtd` service with the below commands and checking status

```
sudo systemctl start libvirtd.service
sudo systemctl enable libvirtd.service
sudo systemctl status libvirtd.service
```

```
[abir@ahammed-20101197] [-]
└─$ sudo systemctl start libvirtd.service
[abir@ahammed-20101197] [-]
└─$ sudo systemctl enable libvirtd.service
[abir@ahammed-20101197] [-]
└─$ sudo systemctl status libvirtd.service
● libvirtd.service - libvirt legacy monolithic daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: disabled)
   Active: active (running) since Fri 2024-02-09 11:49:58 +06; 15s ago
     TriggeredBy: ● libvirtd.socket
                   ● libvirtd-ro.socket
                   ● libvirtd-admin.socket
   Docs: man:libvirtd(8)
         https://libvirt.org/
 Main PID: 51202 (libvirtd)
   Tasks: 20 (limit: 32768)
  Memory: 54.6M (peak: 54.9M)
    CPU: 318ms
   CGroup: /system.slice/libvirtd.service
           └─51202 /usr/bin/libvirtd --timeout 120

Feb 09 11:49:58 ahammed-20101197 systemd[1]: Starting libvirt legacy monolithic daemon...
Feb 09 11:49:58 ahammed-20101197 systemd[1]: Started libvirt legacy monolithic daemon.
Feb 09 11:49:58 ahammed-20101197 libvirtd[51202]: libvirt version: 10.0.0
Feb 09 11:49:58 ahammed-20101197 libvirtd[51202]: hostname: ahammed-20101197
Feb 09 11:49:58 ahammed-20101197 libvirtd[51202]: Cannot find 'dmidecode' in path: No such file or directory
Feb 09 11:49:58 ahammed-20101197 libvirtd[51202]: Cannot find 'dmidecode' in path: No such file or directory
```

We need to make some changes to the `libvirtd` configuration file located at `/etc/libvirt/libvirtd.conf`. After opening the file we have to uncomment (i.e. removing `#` characters) from matching lines below and save the file.

```
unix_sock_group = "libvirt"
unix_sock_rw_perms = "0770"
```

```
# Set the UNIX domain socket group ownership. This can be used to
# allow a 'trusted' set of users access to management capabilities
# without becoming root.
#
# This setting is not required or honoured if using systemd socket
# activation.
#
# This is restricted to 'root' by default.
unix_sock_group = "libvirt"

# Set the UNIX socket permissions for the R/O socket. This is used
# for monitoring VM status only
#
# This setting is not required or honoured if using systemd socket
# activation.
#
# Default allows any user. If setting group ownership, you may want to
# restrict this too.
#unix_sock_ro_perms = "0777"

# Set the UNIX socket permissions for the R/W socket. This is used
# for full management of VMs
#
# This setting is not required or honoured if using systemd socket
# activation.
#
# Default allows only root. If PolicyKit is enabled on the socket,
# the default will change to allow everyone (eg, 0777)
#
# If not using PolicyKit and setting group ownership for access
# control, then you may want to relax this too.
unix_sock_rw_perms = "0770"

# Set the UNIX socket permissions for the admin interface socket.
#
# This setting is not required or honoured if using systemd socket
# activation.
#
# Default allows only owner (root), do not change it unless you are
# sure to whom you are exposing the access to.
#unix_sock_admin_perms = "0700"

# Set the name of the directory in which sockets will be found/created.
#
# This setting is not required or honoured if using systemd socket
# activation.
#
#unix_sock_dir = "/run/libvirt"
"/etc/libvirt/libvirtd.conf" 536L, 17824B written
```

Next we have to add our current user to the libvirt group.

```
sudo usermod -aG libvirt $USER
```

Restart the `libvirtd` service.

```
sudo systemctl restart libvirtd.service
```

```

[abir@ahammed-20101197] [-]
└─$ sudo usermod -aG libvirt $USER
[abir@ahammed-20101197] [-]
└─$ sudo systemctl restart libvirtd.service
[abir@ahammed-20101197] [-]
└─$ sudo systemctl status libvirtd.service
● libvirtd.service - libvirt legacy monolithic daemon
  Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: disabled)
  Active: active (running) since Fri 2024-02-09 11:57:38 +06; 10s ago
TriggeredBy: ● libvirtd.socket
    ● libvirtd-ro.socket
    ● libvirtd-admin.socket
  Docs: man:libvirtd(8)
        https://libvirt.org/
 Main PID: 61546 (libvirtd)
   Tasks: 20 (limit: 32768)
  Memory: 48.7M (peak: 49.1M)
    CPU: 221ms
   CGroup: /system.slice/libvirtd.service
           └─61546 /usr/bin/libvirtd --timeout 120

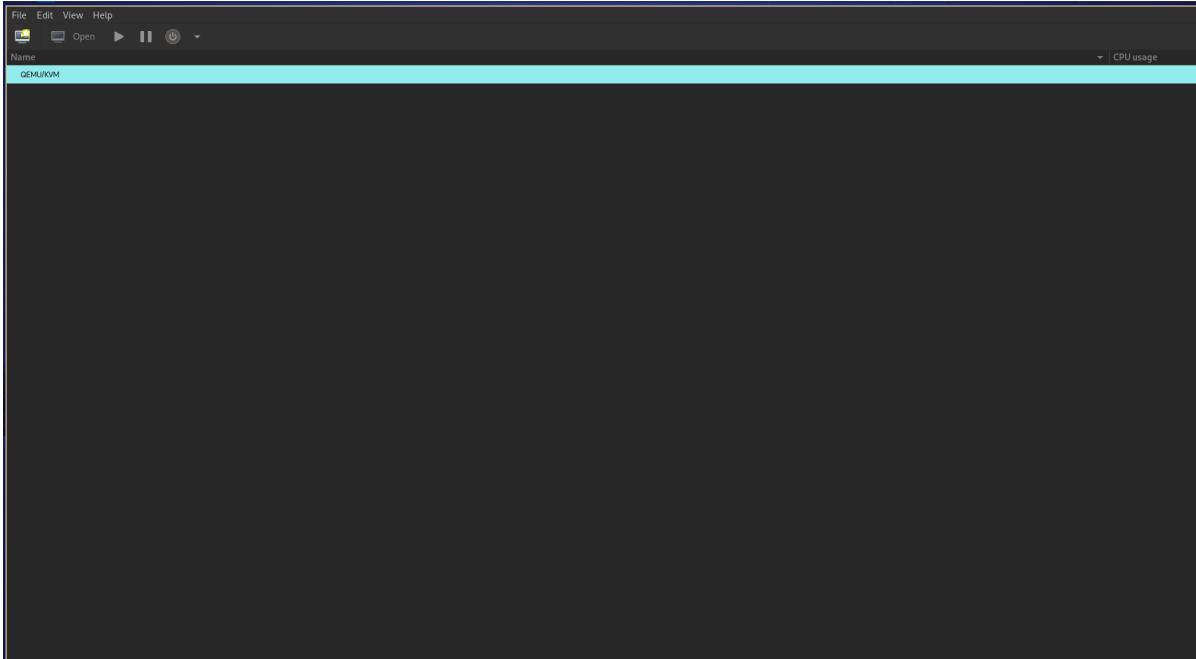
Feb 09 11:57:38 ahammed-20101197 systemd[1]: Starting libvirt legacy monolithic daemon...
Feb 09 11:57:38 ahammed-20101197 systemd[1]: Started libvirt legacy monolithic daemon.
Feb 09 11:57:38 ahammed-20101197 libvirtd[61546]: libvirt version: 10.0.0
Feb 09 11:57:38 ahammed-20101197 libvirtd[61546]: hostname: ahammed-20101197
Feb 09 11:57:38 ahammed-20101197 libvirtd[61546]: Cannot find 'dmidecode' in path: No such file or directory
Feb 09 11:57:38 ahammed-20101197 libvirtd[61546]: Cannot find 'dmidecode' in path: No such file or directory
[abir@ahammed-20101197] [-]
└─$ 

```

That is how the `KVM` has been installed in our system.

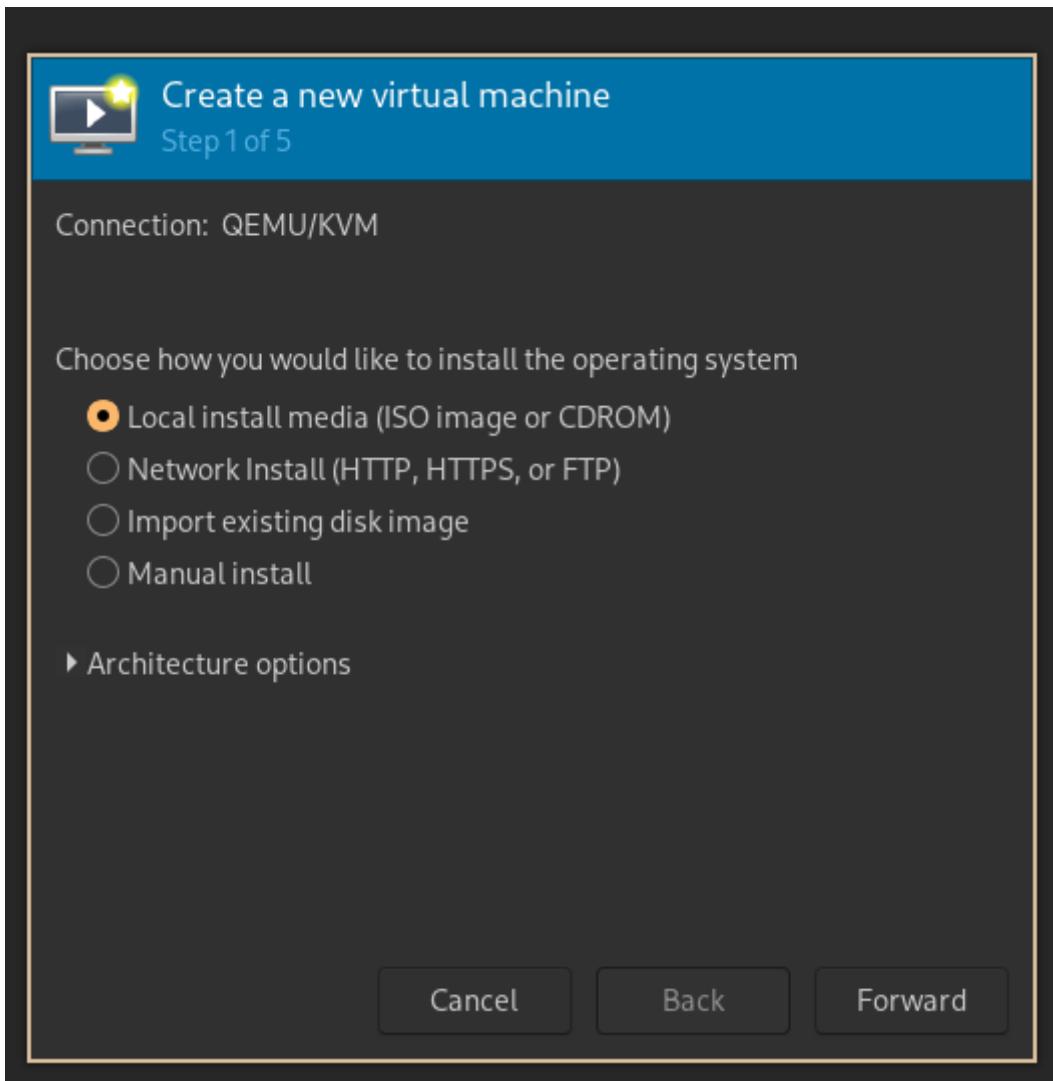
## 2. Create a VM using VMM (virtual machine manager) i.e. using GUI.

open `virt-manager`.

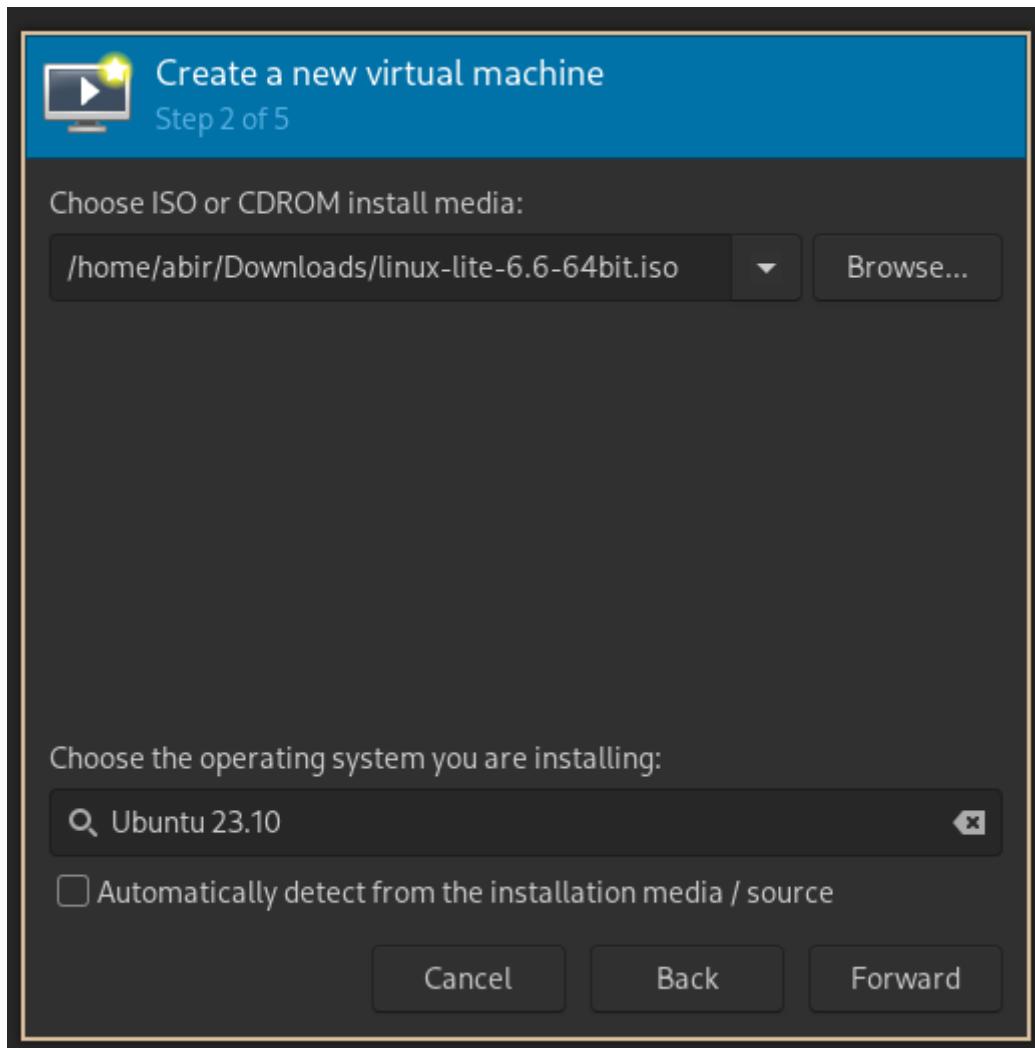


Click on `create a new virtual machine` button on the upper left corner. It will pop up a new window.

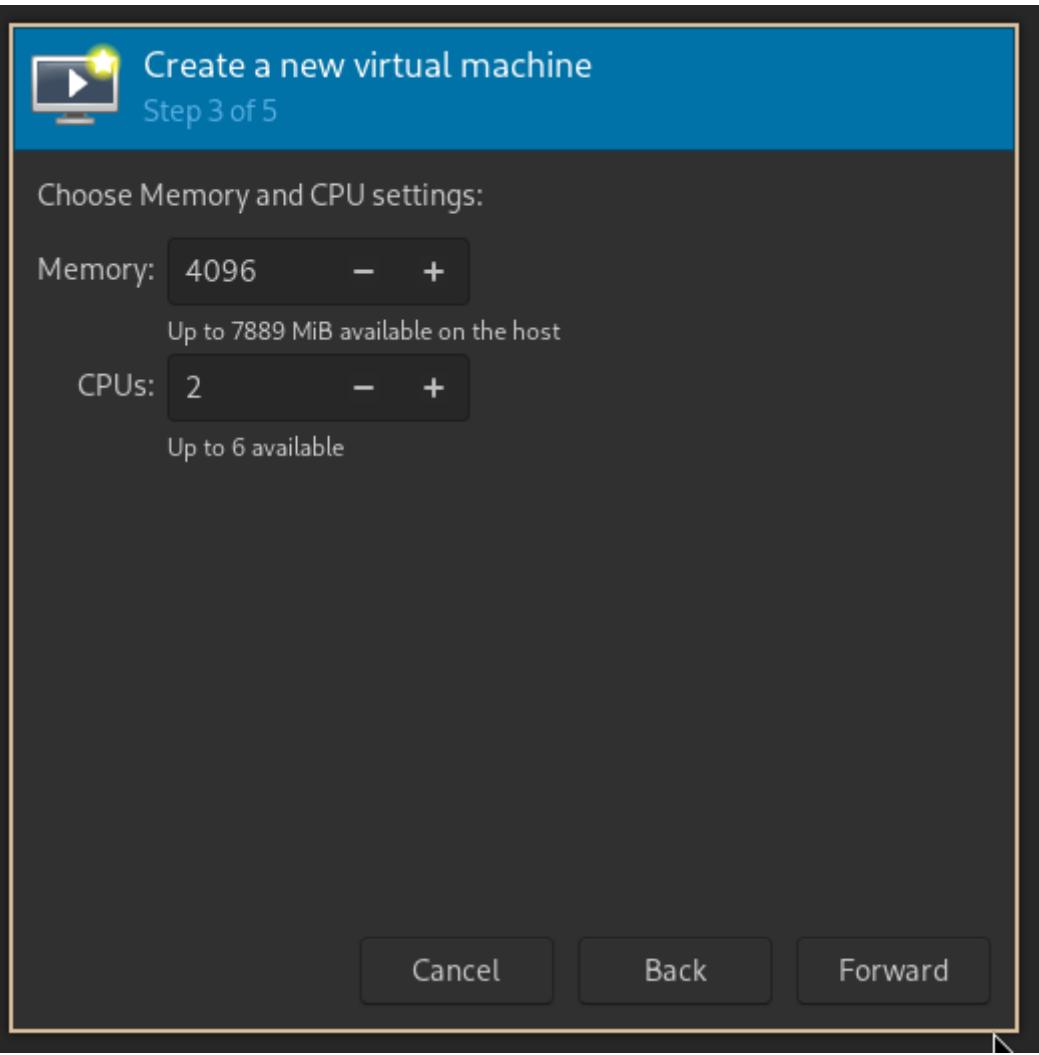
Click on the `Local install media (ISO image or CDROM)` and click forward.



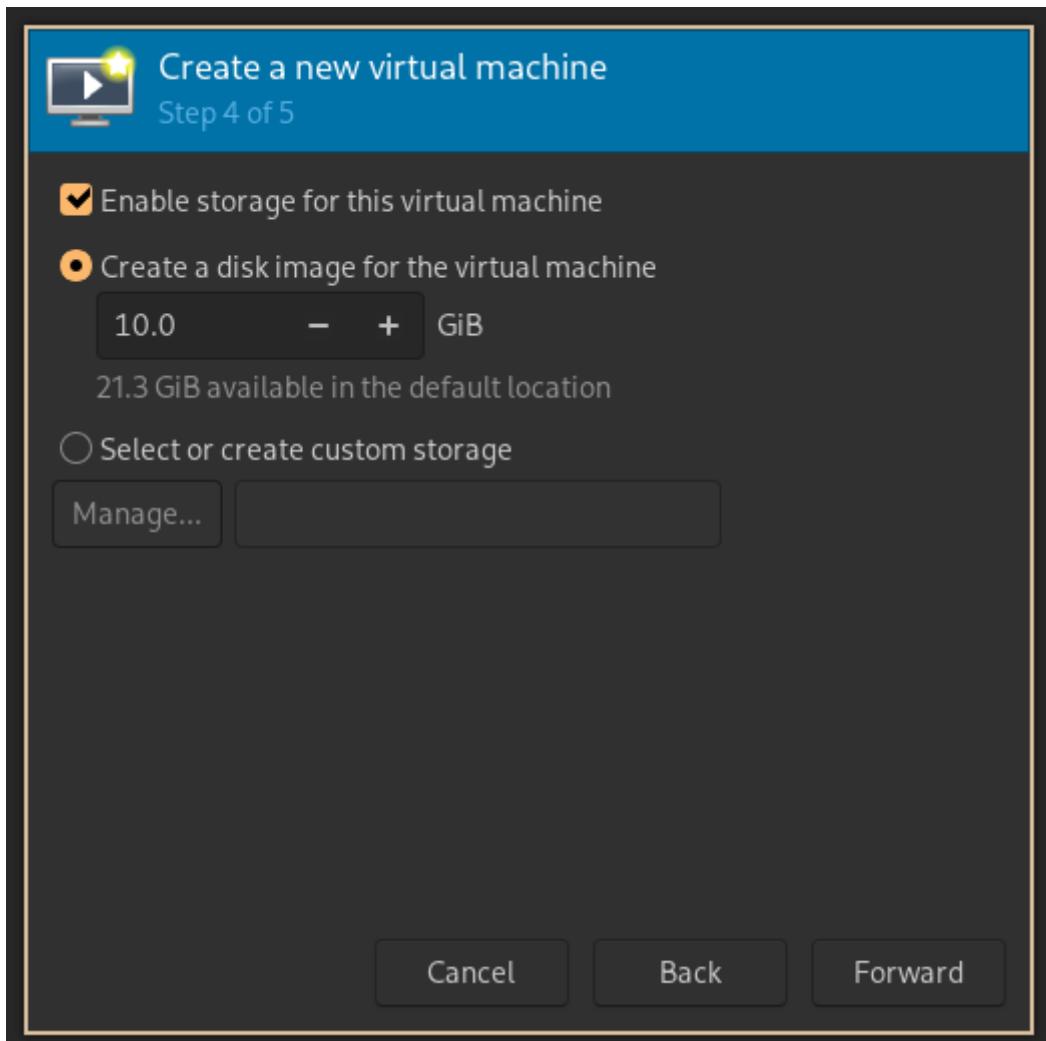
Here we are trying to install `linux-lite` that is based on `Ubuntu 23.10` hence we have selected the options like below. Click forward.



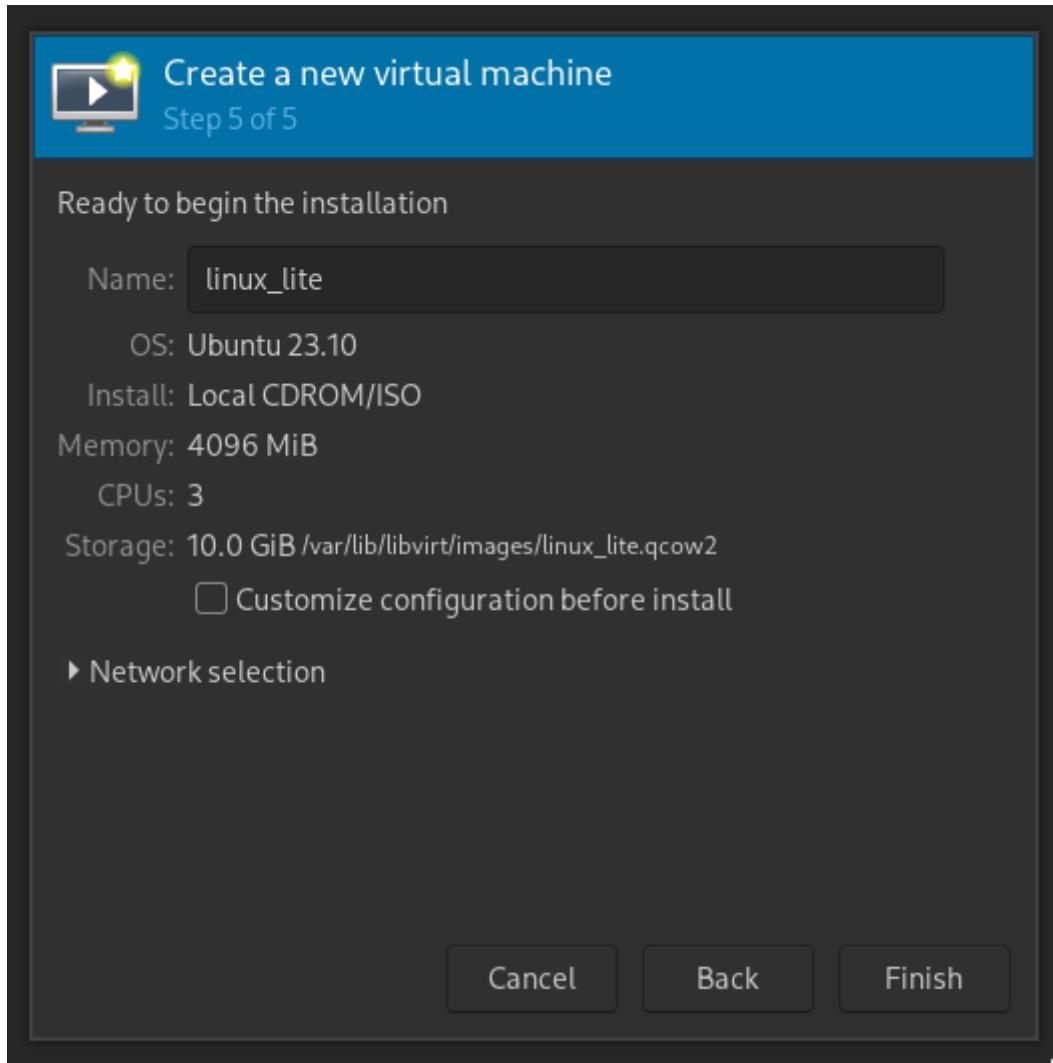
Here we have to allocated memory and CPU. For our case we have allocated 4GB of RAM along with 2 CPU cores. Click forward.



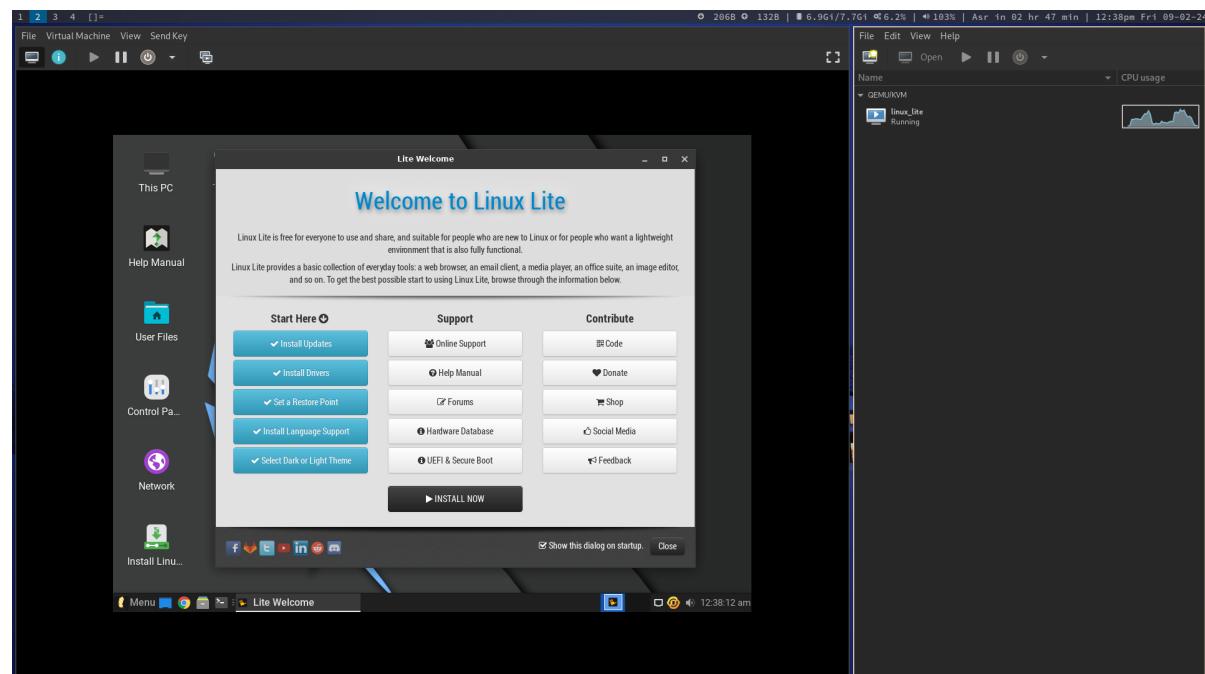
Here we have to specify storage size for the VHD (Virtual Hard Disk). We have allocated 10GB of space for the guest OS.



At last we have to give a name (in our case `linux_lite`) and click on finish.



Wow, we have successfully created a VM and it is running `Linux Lite`.



After running the VM just install the OS.

### 3. Create a kvm-based VM using `virt-install` cli.

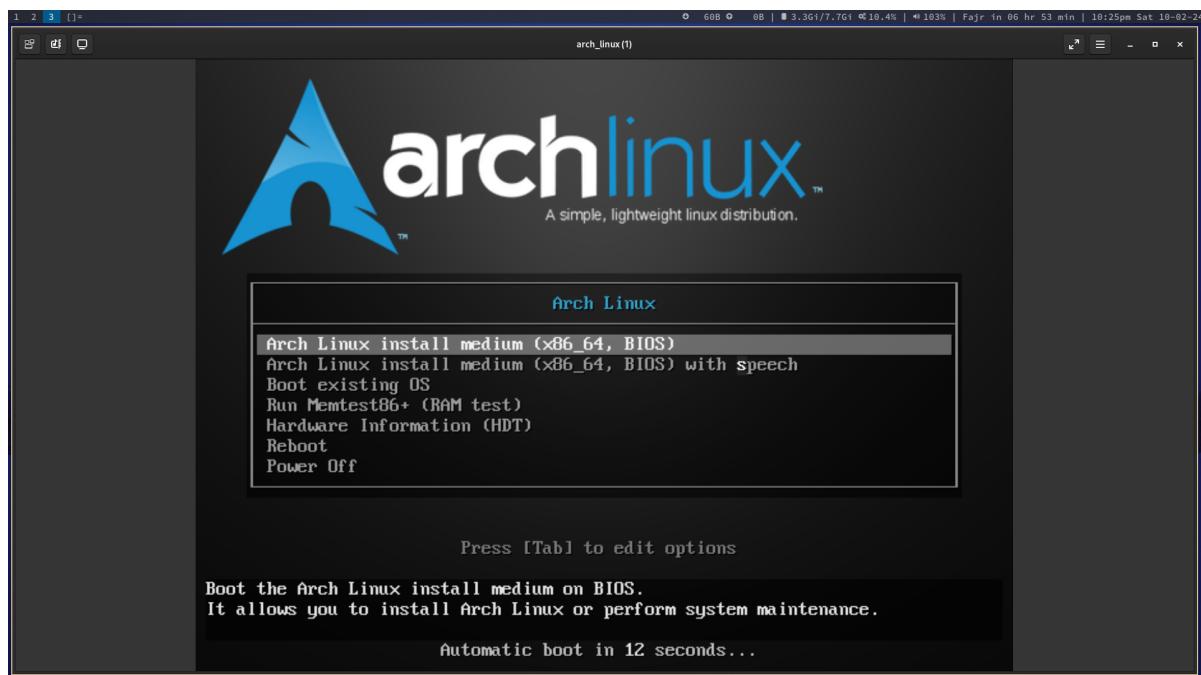
For this we will create a VM for `Arch Linux`. First, we have to fire `virt-install` command like below.

```
sudo virt-install --name arch_linux --memory 2048 --vcpus 2 \
--disk path=/var/lib/libvirt/images/arch-
os.qcow2,format=qcow2,bus=virtio,size=10 \
--cdrom /home/abir/Downloads/archlinux-x86_64.iso \
--os-variant archlinux
```

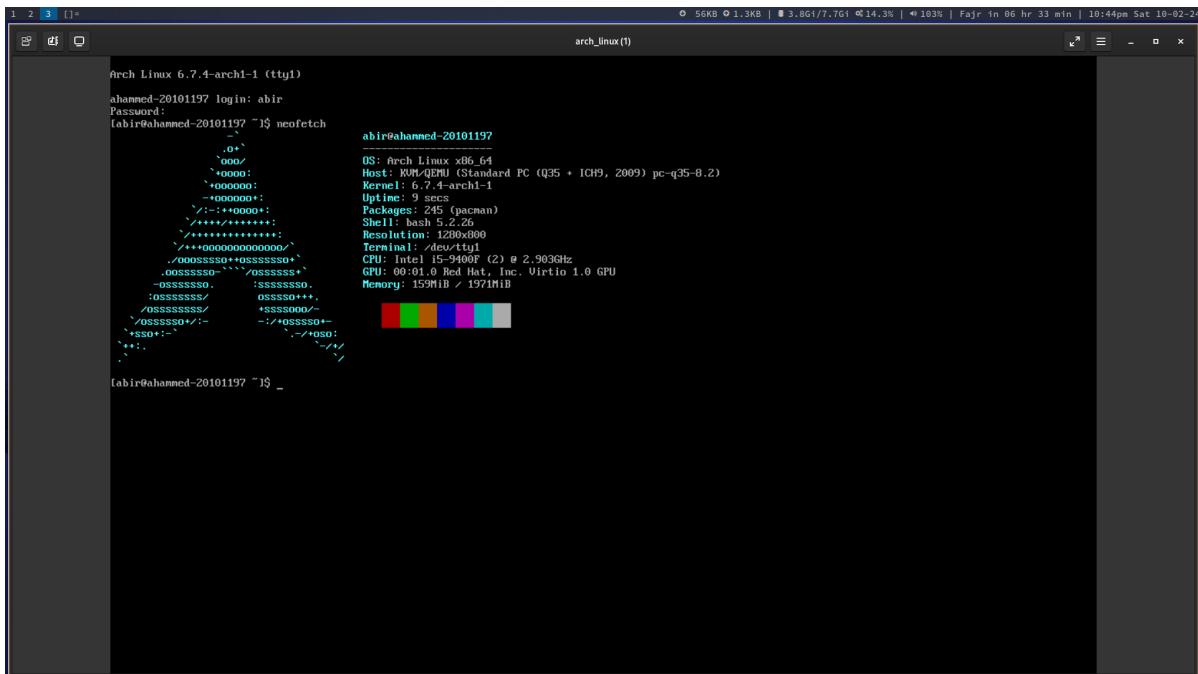


```
[abir@ahammed-20101197] [~]
o sudo virt-install --name arch_linux --memory 2048 --vcpus 2 \
--disk path=/var/lib/libvirt/images/arch-os.qcow2,format=qcow2,bus=virtio,size=10 \
--cdrom /home/abir/Downloads/archlinux-x86_64.iso \
--os-variant archlinux
```

After firing that command we will see a new window like below.



Install `Arch Linux` using `archinstall` command.

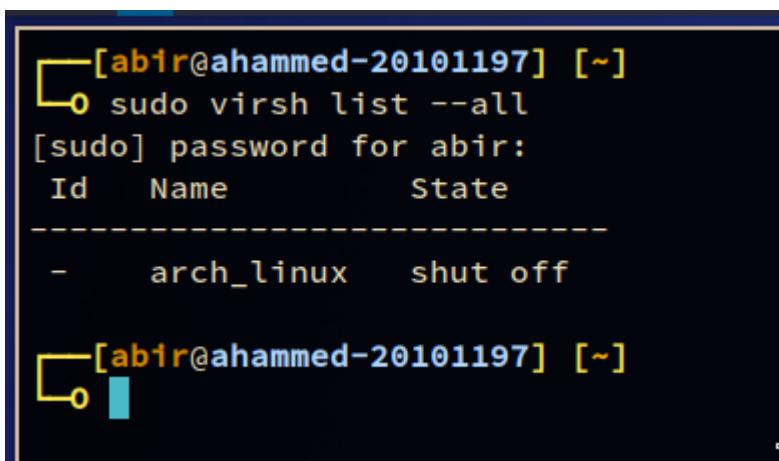


```
Arch Linux 6.7.4-arch1-1 (tty1)
abir@ahammed-20101197 login: abir
Password:
[abir@ahammed-20101197 ~]$ neofetch

[abir@ahammed-20101197 ~]$
```

Using the below command we can see that our desired VM has been created successfully.

```
sudo virsh list --all
```



```
[abir@ahammed-20101197 ~]
└─o sudo virsh list --all
[sudo] password for abir:
 Id   Name      State
 -----
 -   arch_linux  shut off

[abir@ahammed-20101197 ~]
```

## 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.

To achieve this we have to first make a directory and change its access permissions like below. Also, create a file to test our procedure.

```
[abir@ahammed-20101197] [~]
└ o mkdir shared_kvm_host
[abir@ahammed-20101197] [~]
└ o chmod 777 shared_kvm_host/
[abir@ahammed-20101197] [~]
└ o cd shared_kvm_host/
[abir@ahammed-20101197] [~/shared_kvm_host]
└ o echo "from host" > hello
[abir@ahammed-20101197] [~/shared_kvm_host]
└ o cat hello
from host
[abir@ahammed-20101197] [~/shared_kvm_host]
```

The below command is essential for creating a shared folder. The command will be used to open our VM's `config` file.

```
sudo EDITOR=vim virsh edit arch_linux
```

For editing we will be using `vim` text editor so we are using `EDITOR=vim`. You can use any editor you like.

```
[abir@ahammed-20101197] [~]
└ o sudo virsh list --all
  Id      Name          State
  --
  -      arch_linux    shut off

[abir@ahammed-20101197] [~]
└ o sudo EDITOR=vim virsh edit arch_linux
```

After opening the `config` file we have to insert the below lines to activate shared folder feature.

```
<memoryBacking>
  <source type='memfd'/>
  <access mode='shared'/>
</memoryBacking>
```

```
<filesystem type='mount' accessmode='passthrough'>
  <driver type='virtiofs' />
  <source dir='/home/abir/shared_kvm_host' />
  <target dir='shared_kvm_host' />
</filesystem>
```

After saving the `config` file just start the VM like below. `virt-viewer` will open our VM in GUI mode.

```
[abir@ahammed-20101197 ~]
└─$ sudo virsh list --all
   Id  Name      State
-----
-   arch_linux shut off

[abir@ahammed-20101197 ~]
└─$ sudo virsh start arch_linux
Domain 'arch_linux' started

[abir@ahammed-20101197 ~]
└─$ sudo virt-viewer arch_linux
```

Then we have to create a folder inside `guest os` and mount our shared folder using `mount` command like below.

```
mkdir shared_kvm
```

```
sudo mount -v -t virtiofs shared_kvm_host shared_kvm/
```

```
[abir@ahammed-20101197 ~]$ mkdir shared_kvm
[abir@ahammed-20101197 ~]$ sudo mount -v -t virtiofs shared_kvm_host shared_kvm/
[sudo] password for abir:
mount: shared_kvm_host mounted on /home/abir/shared_kvm.
[abir@ahammed-20101197 ~]$
```

Using the `cat` on the saved file in the shared folder we can verify that our shared folder is actually working.

```
[abir@ahammed-20101197 ~]$ cd shared_kvm/
[abir@ahammed-20101197 shared_kvm]$ ls
hello
[abir@ahammed-20101197 shared_kvm]$ cat hello
from host
[abir@ahammed-20101197 shared_kvm]$ _
```

Now change the text file to see if the change can be observed from the host also.

```
[abir@ahammed-20101197 shared_kum]$ ls -al
total 12
drwxrwxrwx 2 abir abir 4096 Feb 11 00:09 .
drwx----- 6 abir abir 4096 Feb 11 00:17 ..
-rw-r--r-- 1 abir abir 10 Feb 11 00:09 hello
[abir@ahammed-20101197 shared_kum]$ cat hello
from host
[abir@ahammed-20101197 shared_kum]$ echo "hello from guest" > hello
[abir@ahammed-20101197 shared_kum]$ cat hello
hello from guest
[abir@ahammed-20101197 shared_kum]$
```

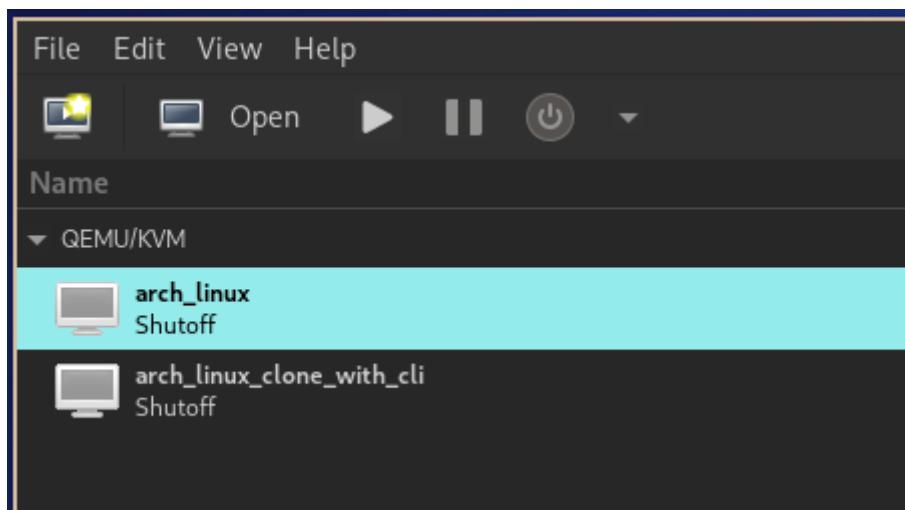
The changes from the `guest os` can also be seen from the `host os`.

```
[abir@ahammed-20101197] [~/shared_kvm_host]
└ o ls
  └ hello
    [abir@ahammed-20101197] [~/shared_kvm_host]
    └ o cat hello
      hello from guest
    [abir@ahammed-20101197] [~/shared_kvm_host]
    └ o
```

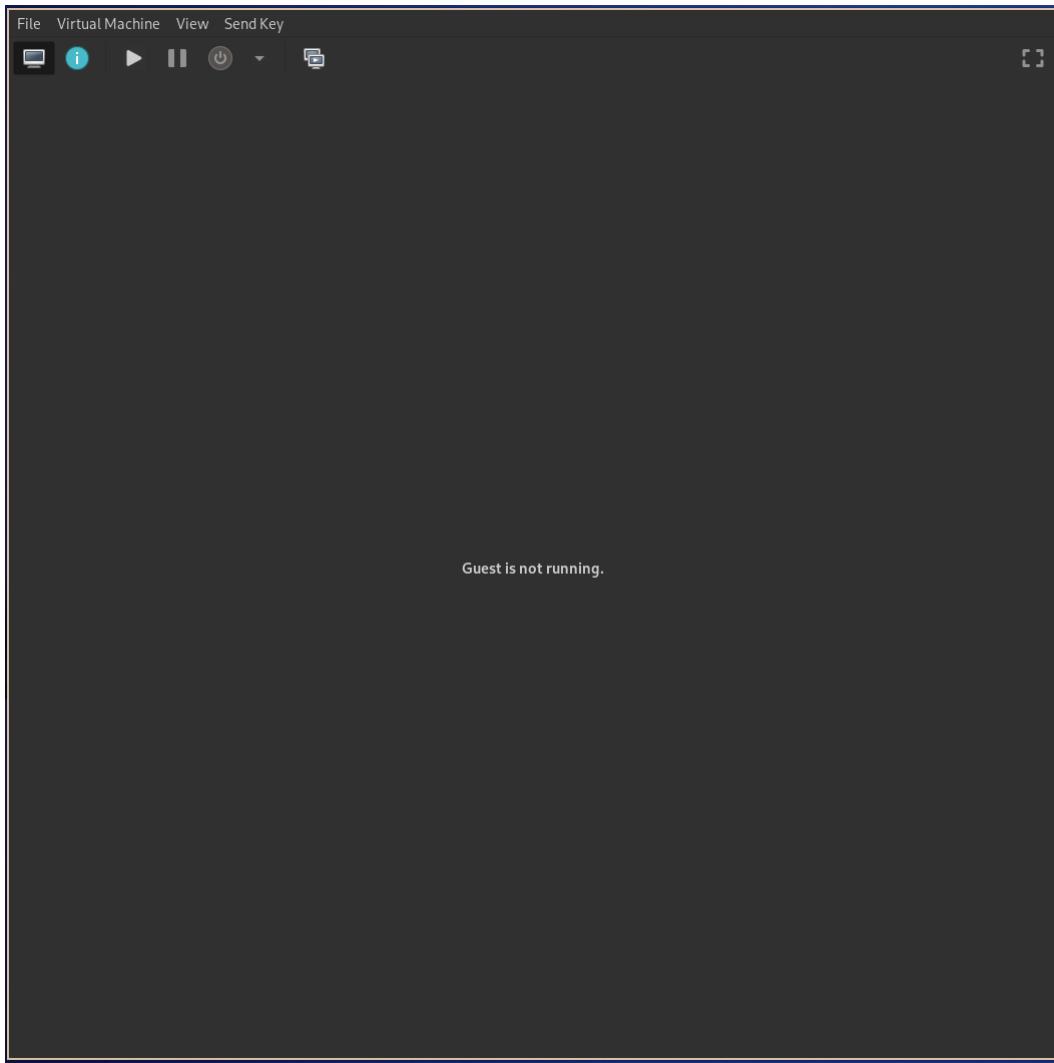
## 6. Connect the guest OS with a phone and access the phone's files.

First, connect your phone with the PC using a usb cable. Make sure on the phone file transfer is enabled.

Then in the PC open `virt-manager`.



Start the VM.



```
Arch Linux 6.7.4-arch1-1 (tty1)
ahammed-20101197 login: abir
Password:
Last login: Thu Feb 15 19:01:17 on tty1
[abir@ahammed-20101197 ~]$ neofetch
      _`_
     .o+`_
    `ooo/
   `+oooo:
  `+oooooo:
 -+ooooooo+:
 /:-:++oooo+:
 `/++++/++++++:
 `/+++++++/+++++:
 `/+++++++/+++++:
 `/+++ooooooooooooo/`_
 ./ooooooo+-+ooooooo+-+
 .ooooooo-````-ooooooo-
 -ooooooo.       :ooooooo.
 :ooooooo/       oooooo+++
 /ooooooo/-       +ooooooo/-
 \ooooooo+/-       -:/+oooo+-
 `ssoo+:-`       `.-/+oso:
 \++:.
 ``
```

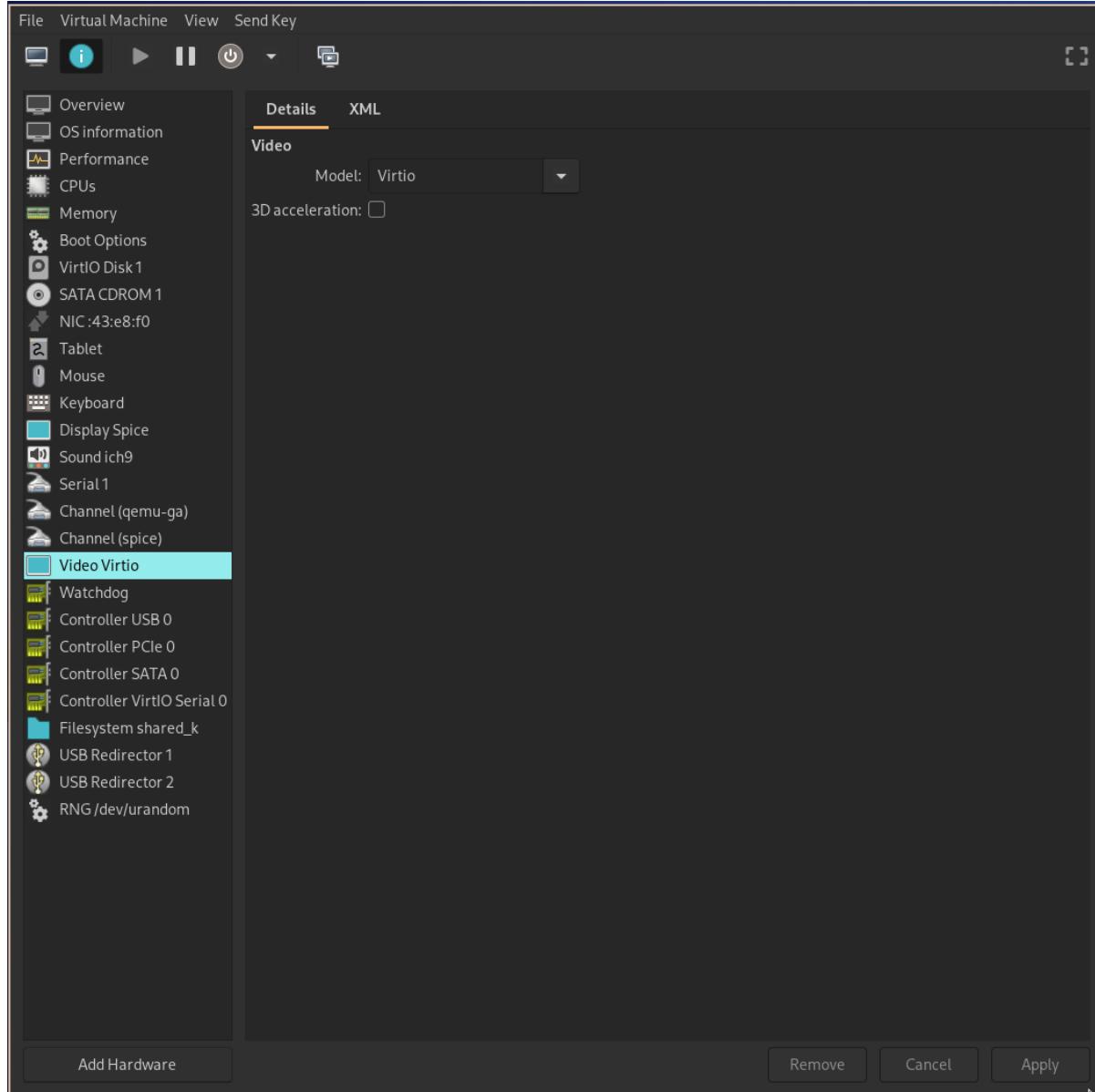
[abir@ahammed-20101197 ~]\$ \_

We have to install a package called `android-file-transfer` for this purpose.

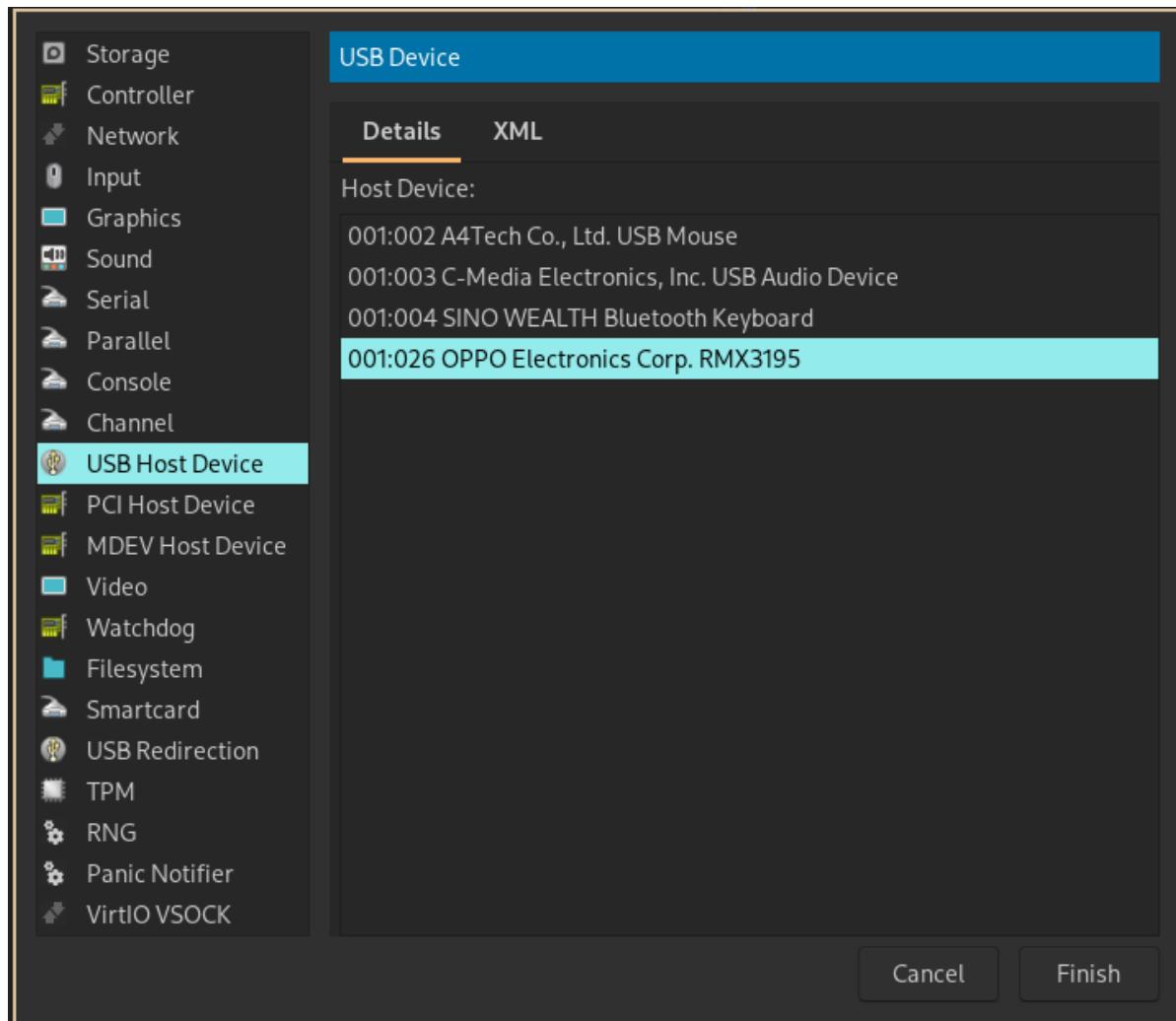
```
sudo pacman -S android-file-transfer
```

After installing the package click on the `info` icon it will open a new window like below.

From there select `Add Hardware`.



You will see a new window, from there select `USB Host Device` and select your phones model and click on `Finish`.

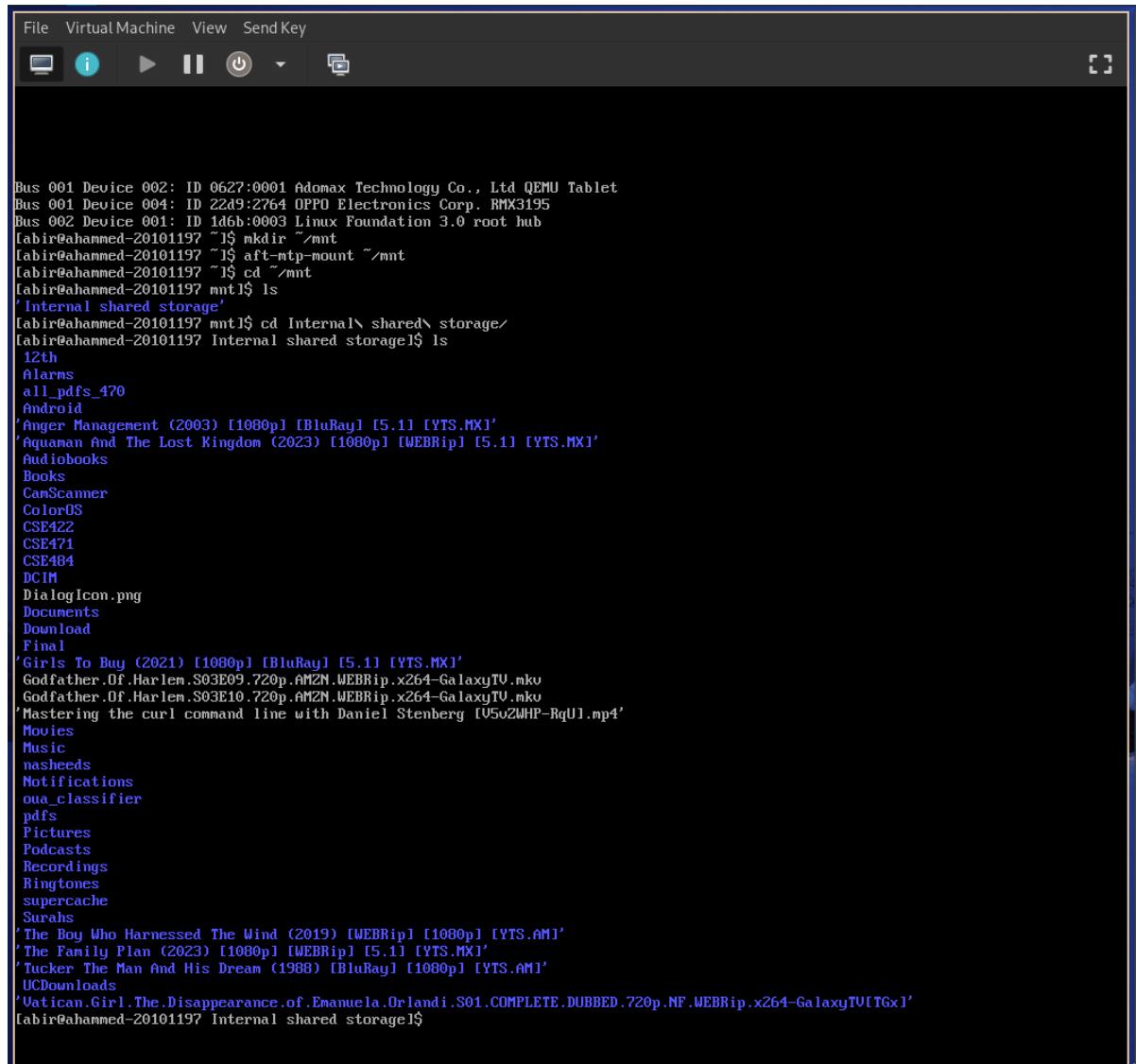


Now get back to the VM and check if the usb device is connected or not using `lsusb` command.

Then create a new directory (mount point) and mount your phone.

```
mkdir ~/mnt  
aft-mtp-mount ~/mnt
```

Then check inside the `~/mnt` directory and you will see your phone's content being shown.



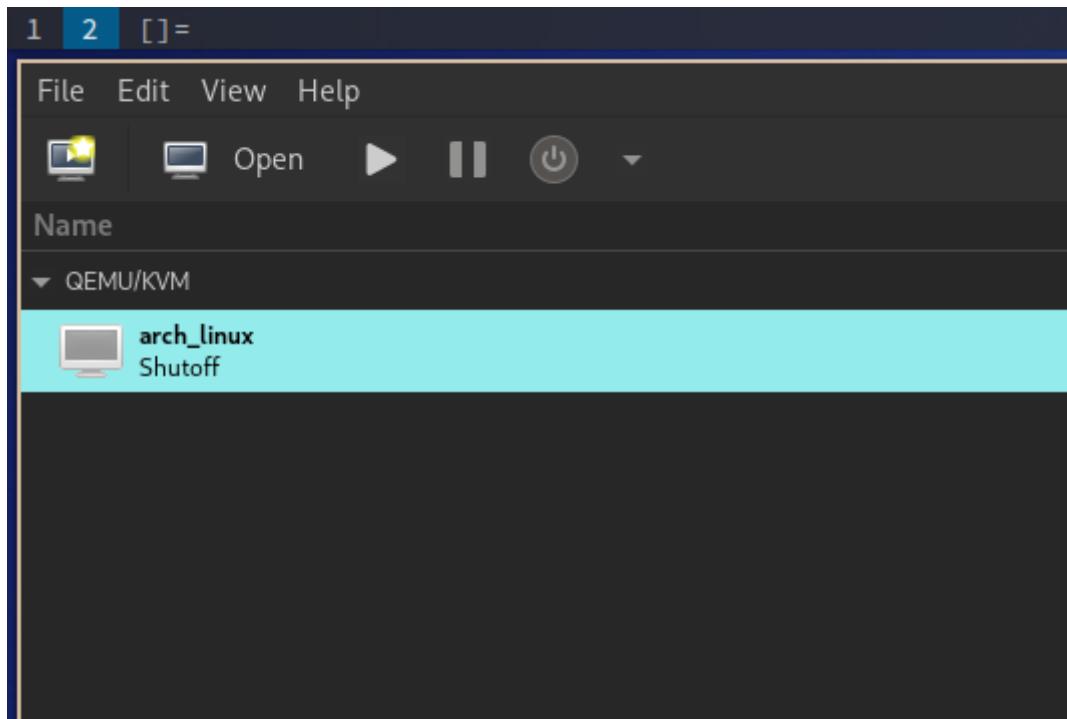
```
File Virtual Machine View Send Key
[ ] i ▶ || ⌂ ↻ [ ] [x]

Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd QEMU Tablet
Bus 001 Device 004: ID 22d9:2764 OPPO Electronics Corp. RMX3195
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
fabir@ahammed-20101197 ~]$ mkdir ~/mnt
fabir@ahammed-20101197 ~]$ apt-ntp-mount ~/mnt
fabir@ahammed-20101197 ~]$ cd ~/mnt
fabir@ahammed-20101197 mnt]$ ls
'Internal shared storage'
fabir@ahammed-20101197 mnt]$ cd Internal\ shared\ storage/
fabir@ahammed-20101197 Internal shared storage]$ ls
12th
Alarms
all_pdffs_470
Android
'Anger Management (2003) [1080p] [BluRay] [5.1] [YTS.MX]'
'Aquaman And The Lost Kingdom (2023) [1080p] [WEBRip] [5.1] [YTS.MX]'
Audiobooks
Books
CamScanner
ColorOS
CSE422
CSE471
CSE484
DCIM
DialogIcon.png
Documents
Download
Final
'Girls To Buy (2021) [1080p] [BluRay] [5.1] [YTS.MX]'
Godfather.Of.Harlem.S03E09.720p.AMZN.WEBRip.x264-GalaxyTV.mkv
Godfather.Of.Harlem.S03E10.720p.AMZN.WEBRip.x264-GalaxyTV.mkv
'Mastering the curl command line with Daniel Stenberg [VSvZWHP-RqU].mp4'
Movies
Music
nasheeds
Notifications
oua_classifier
pdfs
Pictures
Podcasts
Recordings
Ringtones
supercache
Surahs
'The Boy Who Harnessed The Wind (2019) [WEBRip] [1080p] [YTS.AM]'
'The Family Plan (2023) [1080p] [WEBRip] [5.1] [YTS.MX]'
'Tucker The Man And His Dream (1988) [BluRay] [1080p] [YTS.AM]'
UCDownloads
'Vatican.Girl.The.Disappearance.of.Emanuela.Orlandi.S01.COMPLETE.DUBBED.720p.NF.WEBRip.x264-GalaxyTUITGx1'
fabir@ahammed-20101197 Internal shared storage]$
```

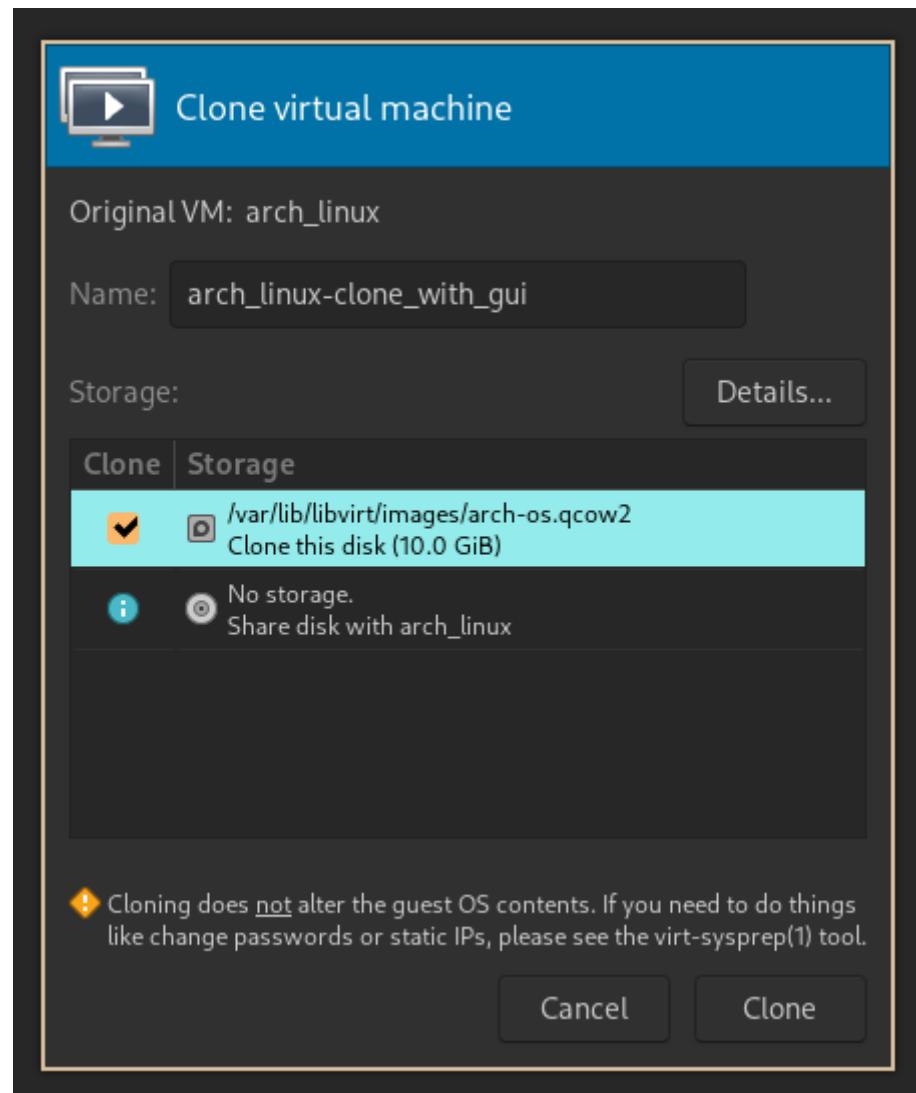
## 7. Clone a VM using GUI and using a kvm-based command.

### Using GUI

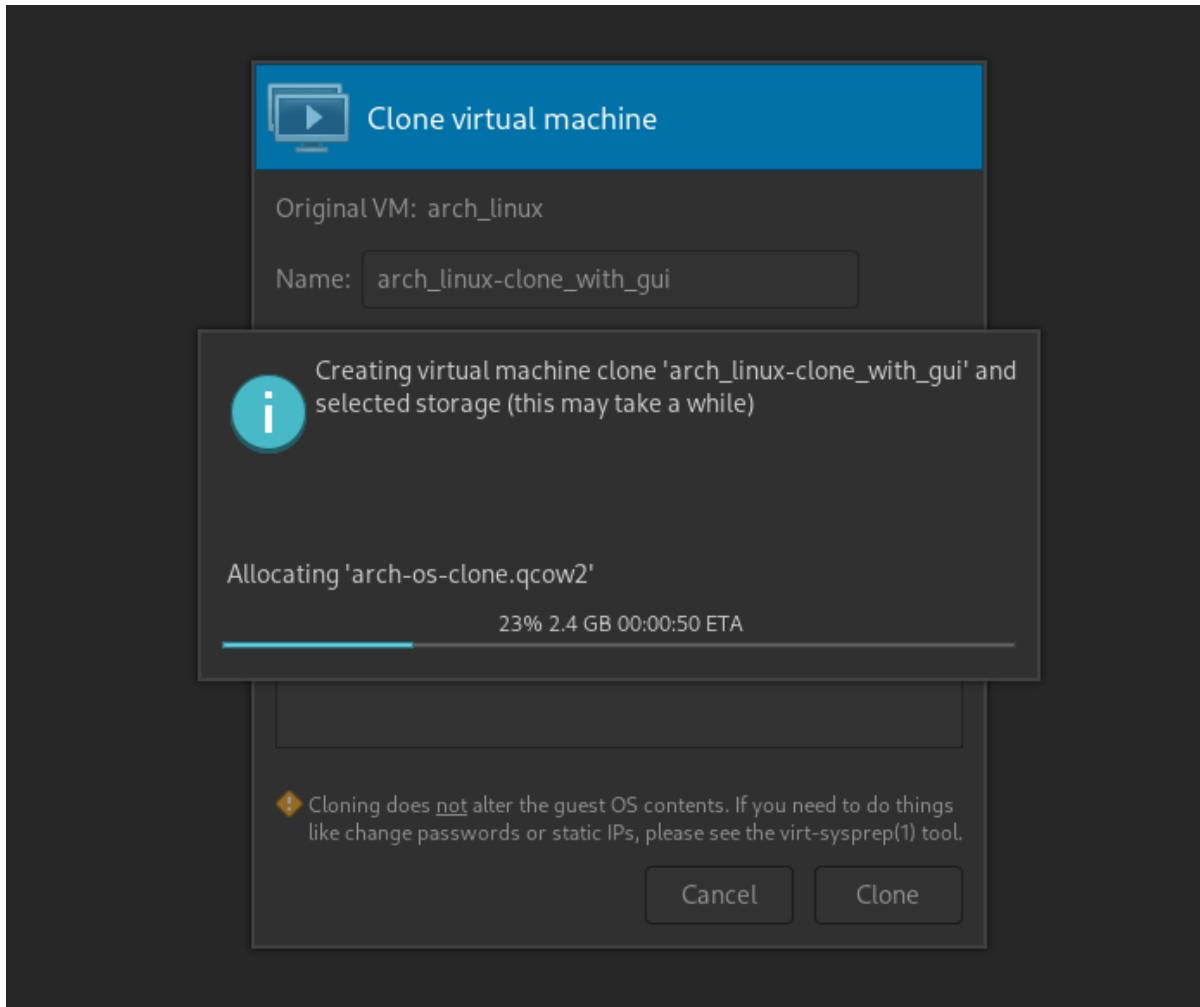
First, open `virt-manager`.



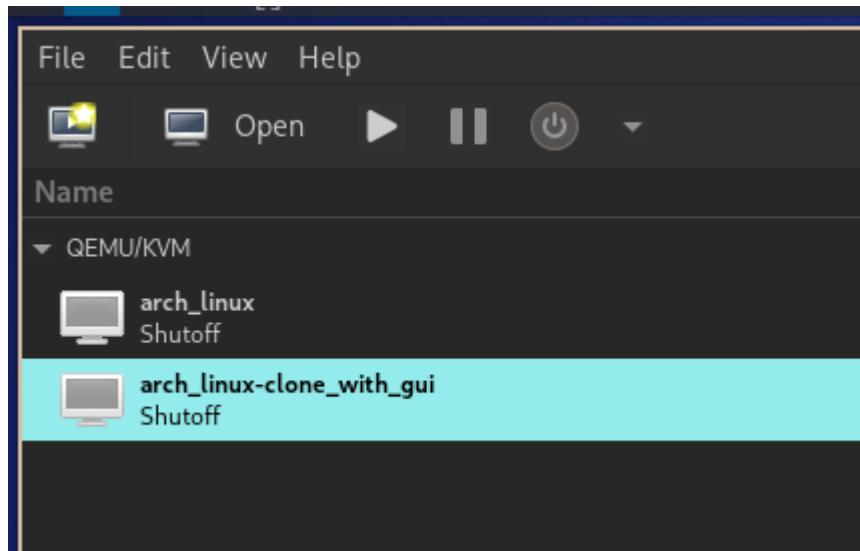
Right click on our selected os will open a menu. Just select `clone` from there.



Then just click on `clone` and we are done cloning.



In the `virt-manager` we can see our cloned VM.



## Using CLI

We can also clone a VM by using a command like below.

```
sudo virt-clone --original arch_linux --name arch_linux_clone_with_cli -f /var/lib/libvirt/images/arch_linux_clone.qcow2
```

Using the `virsh` command we can verify our work.

```
[abir@ahammed-20101197] [-]
└─$ sudo virsh list --all
Id  Name           State
--- 
-  arch_linux      shut off
[abir@ahammed-20101197] [-]
└─$ sudo virt-clone --original arch_linux --name arch_linux_clone_with_cli -f /var/lib/libvirt/images/arch_linux_clone.qcow2
Allocating 'arch_linux_clone' created successfully.
[abir@ahammed-20101197] [-]
└─$ sudo virsh list --all
Id  Name           State
--- 
-  arch_linux      shut off
-  arch_linux_clone_with_cli  shut off
[abir@ahammed-20101197] [-]
└─$ sudo virsh start arch_linux_clone_with_cli
Domain 'arch_linux_clone_with_cli' started
[abir@ahammed-20101197] [-]
```

Our cloned VM is running fine.

```
ahammed-20101197 login: abir
Password:
Last login: Sun Feb 11 11:19:07 on ttyS0
[abir@ahammed-20101197 ~]$ neofetch
              _`_
              .o+'
             `ooo/
             `+oooo:
             `+oooooo:
             +-oooooooo+:
             `/:-:+oooo+:
             `/++++/++++++:
             `/+++++++/++++++:
             `/+++ooooooooooooo/`_
             ./ooooooooo++oooooooo+`_
             .oooooooo-```/oooooooo+`_
             -oooooooo.       :oooooooo.
             :oooooooo/       oooooo++.
             /oooooooo/       +oooooooo/-_
             `/oooooooo+/-`-:/+oooooooo+-
             `+soo+/-`       `.-/+oso:
             `++:.          `--/+
             .`               `/

[abir@ahammed-20101197 ~]$
```

**abir@ahammed-20101197**

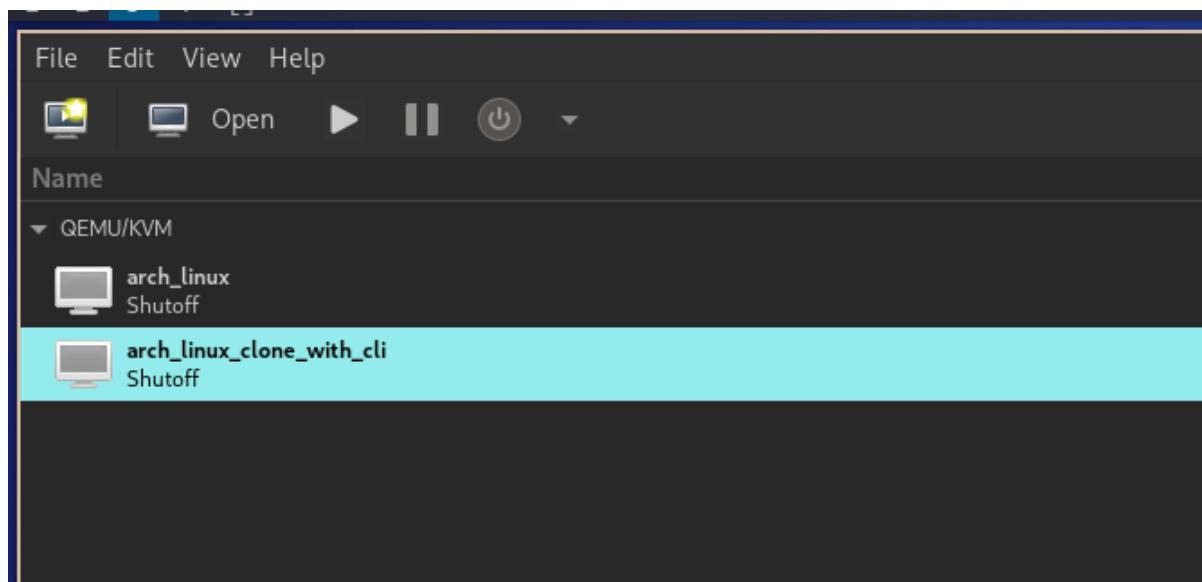
---

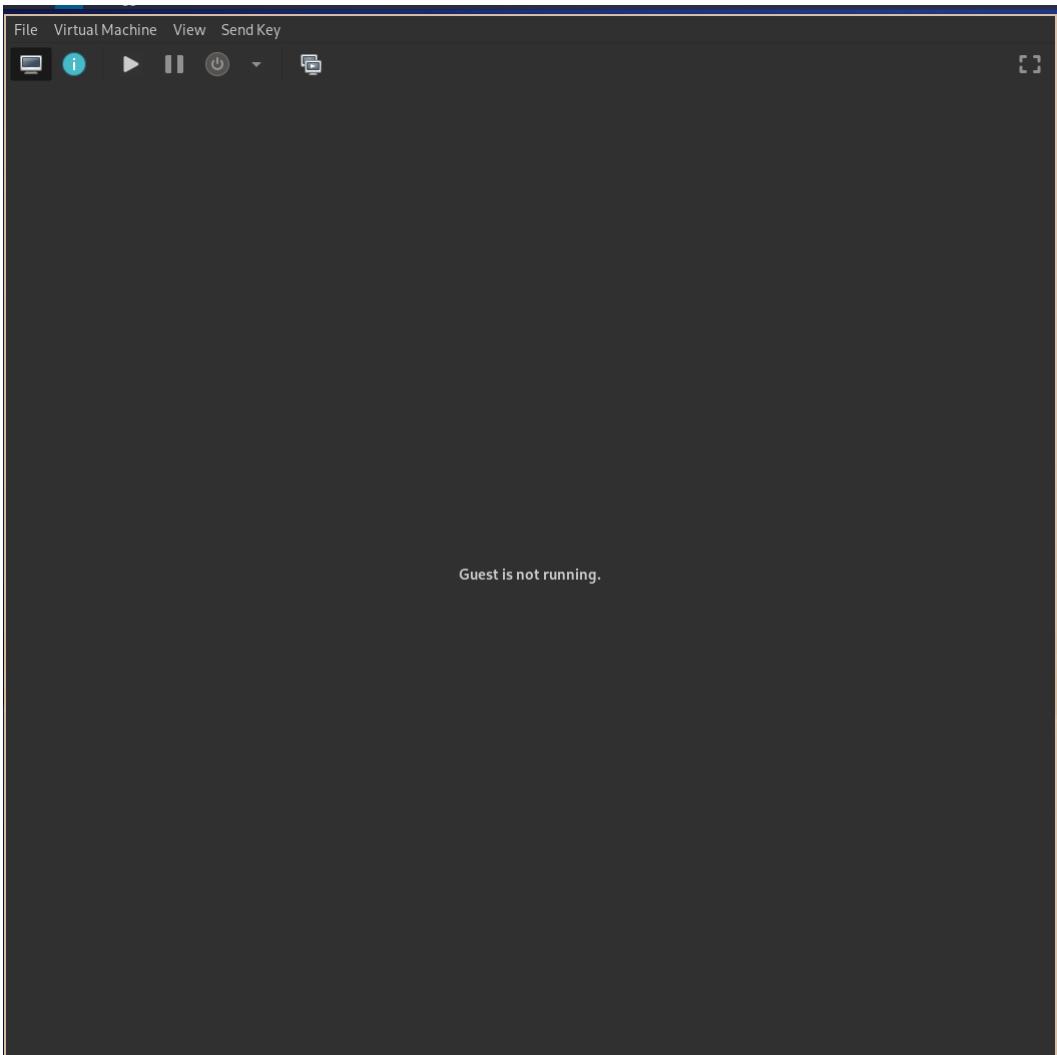
**OS:** Arch Linux x86\_64  
**Host:** KVM/QEMU (Standard PC (Q35 + ICH9, 2009) pc-q35-8.2)  
**Kernel:** 6.7.4-arch1-1  
**Uptime:** 2 mins  
**Packages:** 245 (pacman)  
**Shell:** bash 5.2.26  
**Resolution:** 1280x800  
**Terminal:** /dev/ttyS0  
**CPU:** Intel i5-9400F (2) @ 2.903GHz  
**GPU:** 00:01.0 Red Hat, Inc. Virtio 1.0 GPU  
**Memory:** 152MiB / 1971MiB



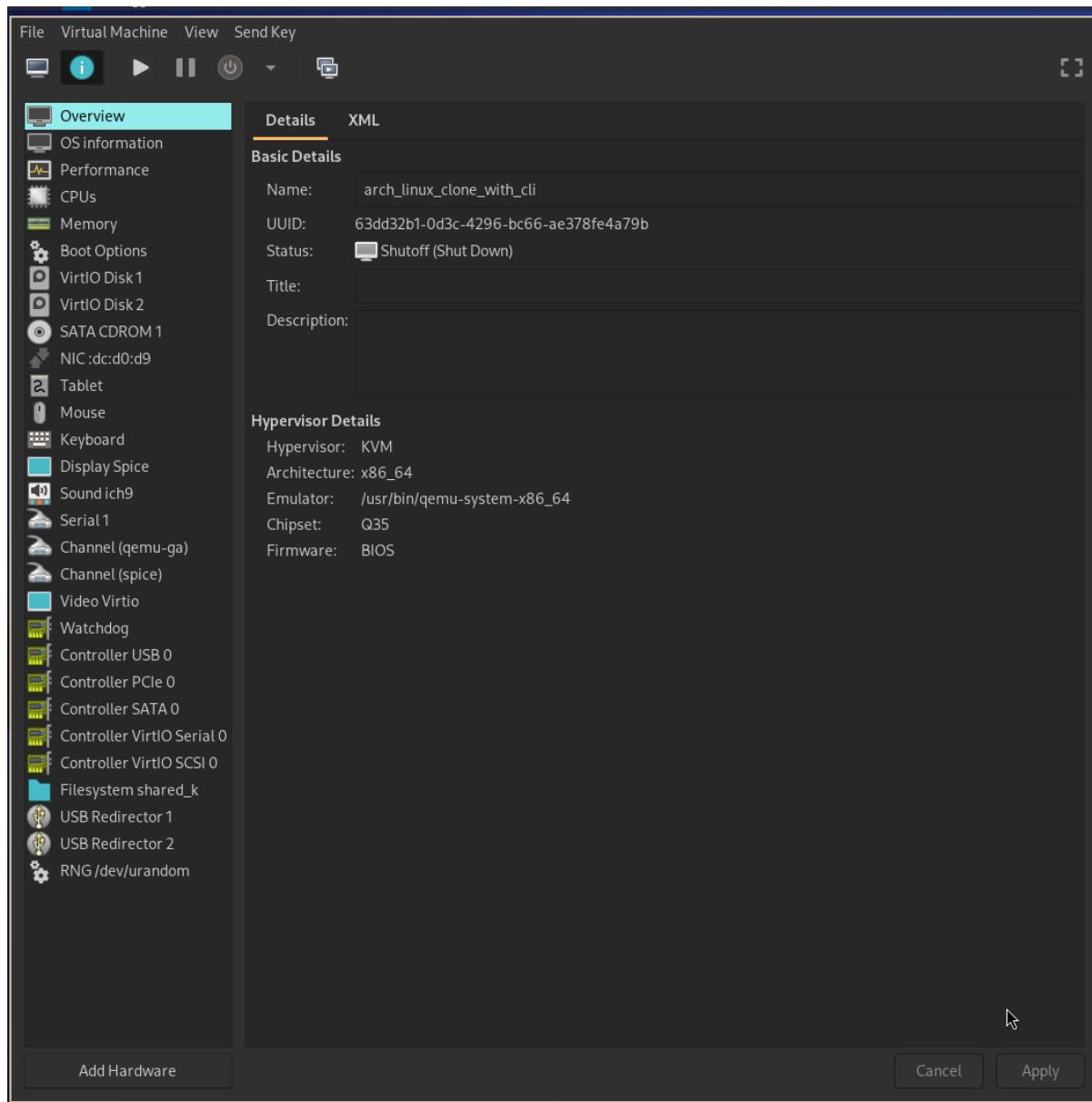
## 8. Add two hard disks in a new cloned virtual machine using GUI.

Open `virt-manager` select your VM but don't start it.

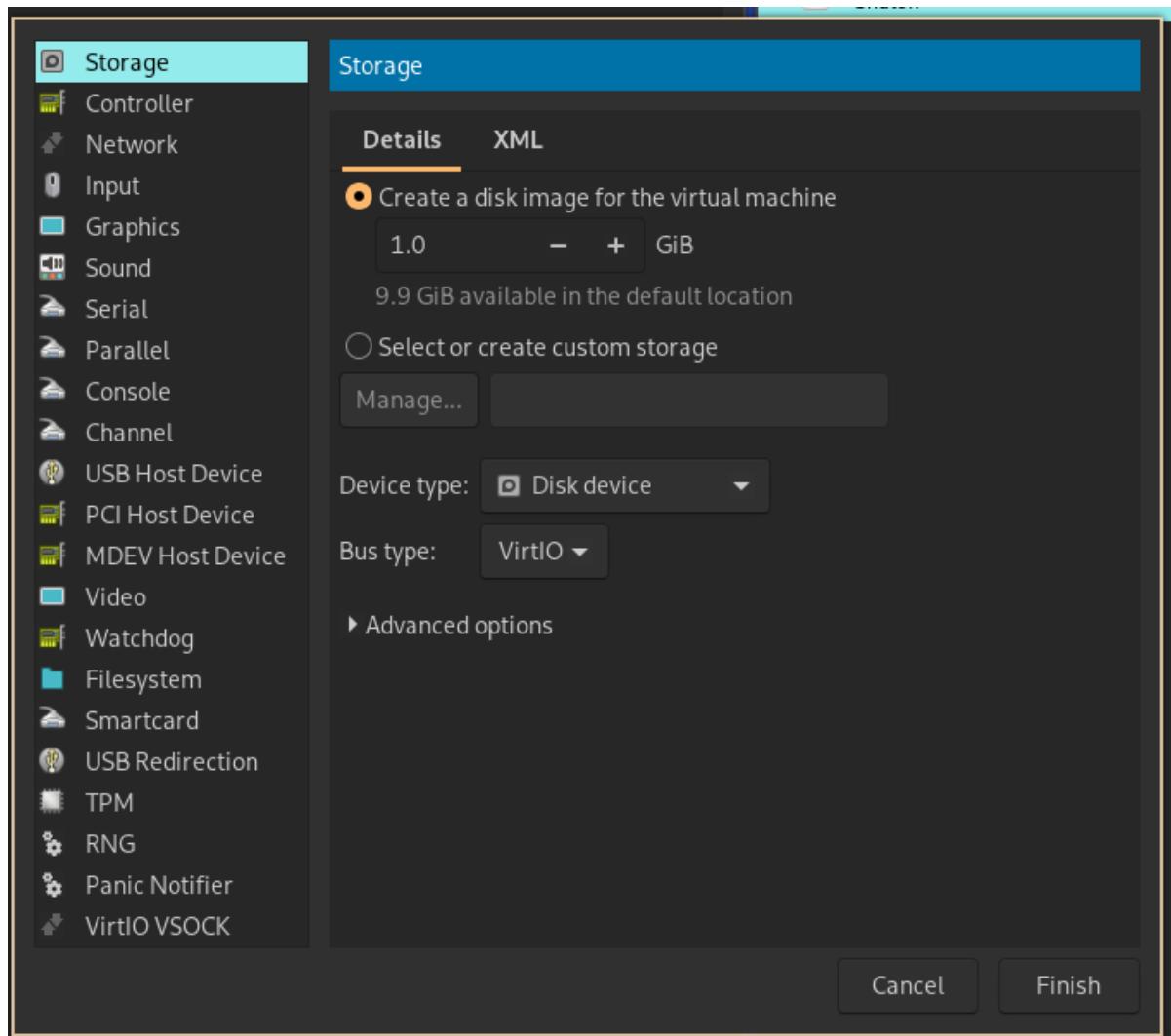




Click on the `info` icon and from there select `Add Hardware`.



Now select `Storage` and give size to your storage and makesure `Device type` is set to `Disk device` and `Bus type` is set to `VirtIO`.



This way create two storage. Start the VM using CLI or GUI and login.

We can use `lsblk` to justify if our storage devices are connected or not. From the below screen shot we can see two new storages named `/dev/vdb` and `/dev/vdc` have been recognized.

```

[abir@ahammed-20101197] [~]
└─$ sudo virsh list --all
[sudo] password for abir:
 Id   Name                State
 -----
 -   arch_linux            shut off
 -   arch_linux_clone_with_cli    shut off

[abir@ahammed-20101197] [~]
└─$ sudo virsh start arch_linux_clone_with_cli
Domain 'arch_linux_clone_with_cli' started

[abir@ahammed-20101197] [~]
└─$ sudo virsh console arch_linux_clone_with_cli
Connected to domain 'arch_linux_clone_with_cli'
Escape character is ^] (ctrl + ])

ahammed-20101197 login: abir
Password:
Last login: Thu Feb 15 00:09:53 on ttys0
[abir@ahammed-20101197 ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0     11:0    1 1024M  0 rom
vda     254:0    0   10G  0 disk
└─vda1  254:1    0  203M  0 part /boot
└─vda2  254:2    0  9.7G  0 part /
vdb     254:16   0    1G  0 disk
vdc     254:32   0    1G  1 disk
[abir@ahammed-20101197 ~]$
```

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

First, go to the `/var/lib/libvirt/images/` because we will create our hard disks there and fire the below command twice to create two hard disks.

```
sudo qemu-img create -f qcow2 hdd_1 1G
```

From the screen shot we can see that two new hard disks (hdd\_1 and hdd\_2) have been created.

```

[abir@ahammed-20101197] [~]
└ o cd /var/lib/libvirt/images/
[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o sudo qemu-img create -f qcow2 hdd_1 1G
[sudo] password for abir:
Formatting 'hdd_1', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=1G
[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o sudo qemu-img create -f qcow2 hdd_2 1G
Formatting 'hdd_2', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=1G
[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o ll
drwxr-xr-x root root 4.0 KB Thu Feb 15 00:49:02 2024 .
drwxr-xr-x root root 4.0 KB Sun Feb  4 00:30:27 2024 ..
.rw----- root root 10 GB Mon Feb 12 00:48:16 2024 arch-os.qcow2
.rw----- root root 3.0 GB Thu Feb 15 00:18:34 2024 arch_linux_clone.qcow2
.rw-r--r-- root root 192 KB Thu Feb 15 00:48:54 2024 hdd_1
.rw-r--r-- root root 192 KB Thu Feb 15 00:49:02 2024 hdd_2
[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o

```

Now we have to attach those hard disks using the below command.

```

sudo virsh attach-disk arch_linux_clone_with_cli /var/lib/libvirt/images/hdd_1
vdb
sudo virsh attach-disk arch_linux_clone_with_cli /var/lib/libvirt/images/hdd_2
vdc

```

Now start the VM and using `lsblk` command we can ensure that two new hard disks have been attached.

```

[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o sudo virsh start arch_linux_clone_with_cli
Domain 'arch_linux_clone_with_cli' started

[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o sudo virsh attach-disk arch_linux_clone_with_cli /var/lib/libvirt/images/hdd_1 vdb
Disk attached successfully

[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o sudo virsh attach-disk arch_linux_clone_with_cli /var/lib/libvirt/images/hdd_2 vdc
Disk attached successfully

[abir@ahammed-20101197] [/var/lib/libvirt/images]
└ o sudo virsh console arch_linux_clone_with_cli
Connected to domain 'arch_linux_clone_with_cli'
Escape character is ^] (Ctrl + ])

ahammed-20101197 login: abir
Password:
Last login: Thu Feb 15 00:16:29 on ttys0
[abir@ahammed-20101197 ~]$ lsblk
NAME   MAJ:MIN RM    SIZE RO TYPE MOUNTPOINTS
sr0     11:0    1 1024M  0 rom
vda    254:0    0   10G  0 disk
└─vda1 254:1    0   203M  0 part /boot
└─vda2 254:2    0   9.7G  0 part /
vdb    254:16   0 192.5K  0 disk
vdc    254:32   0 192.5K  0 disk
[abir@ahammed-20101197 ~]$
logout

Arch Linux 6.7.4-arch1-1 (ttys0)

ahammed-20101197 login:

```

# **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.**

---

## **In our machine:**

---

First we have copy the virtual disk image (i.e. `* .qcow2`) file from our computer to our friend's computer. To copy, we can use either pendrive or we can copy over secure shell. Here, we are copying `* .qcow2` file over secure shell more precisely we are using the `scp` command.

```
scp -r -v /var/lib/libvirt/images/arch-os.qcow2 ub41201@172.18.188.108:~/
```

We are copying `arch-os.qcow2` from our machine to the machine located at `172.18.188.108` also the user name is `ub41201`.

```
abir@ahammed-20101197:~$ scp -r -v /var/lib/libvirt/images/arch-os.qcow2 ub41201@172.18.188.108:~/
Executing: program /usr/bin/ssh host 172.18.188.108, user ub41201, command scp -v -r -t ~/OpenSSH_8.9p1 Ubuntu-3ubuntu0.6, OpenSSL 3.0.2 15 Mar 2022
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to 172.18.188.108 [172.18.188.108] port 22.
debug1: Connection established.
debug1: identity file /home/abir/.ssh/id_rsa type -1
debug1: identity file /home/abir/.ssh/id_rsa-cert type -1
debug1: identity file /home/abir/.ssh/id_ecdsa type -1
debug1: identity file /home/abir/.ssh/id_ecdsa-cert type -1
debug1: identity file /home/abir/.ssh/id_ecdsa_sk type -1
debug1: identity file /home/abir/.ssh/id_ecdsa_sk-cert type -1
debug1: identity file /home/abir/.ssh/id_ed25519 type -1
debug1: identity file /home/abir/.ssh/id_ed25519-cert type -1
```

```

abir@ahammed-20101197: ~
debug1: rekeying in progress
debug1: rekeying in progress
debug1: SSH2_MSG_KEX_ECDH_REPLY received
debug1: Server host key: ssh-ed25519 SHA256:TNCABq76r1cje139ct5VgL3tg3NvxhPS0UZpTMTqM/Q
debug1: ssh_packet_send2_wrapped: resetting send seqnr 33372
debug1: ssh_set_newkeys: rekeying out, input 1522356 bytes 16995 blocks, output 9662632572 bytes 1341
86356 blocks
debug1: rekey out after 134217728 blocks
debug1: dequeue packet: 94
debug1: SSH2_MSG_NEWKYS sent
debug1: expecting SSH2_MSG_NEWKYS
debug1: ssh_packet_read_poll2: resetting read seqnr 8420
debug1: SSH2_MSG_NEWKYS received
debug1: ssh_set_newkeys: rekeying in, input 1522368 bytes 16996 blocks, output 9662665360 bytes 4098
blocks
debug1: rekey in after 134217728 blocks
arch-os.qcow2
99% 10GB 93.8MB/s 00:00 ET

Adebug1: enqueue packet: 94
debug1: SSH2_MSG_KEXINIT sent
debug1: rekeying in progress
debug1: rekeying in progress
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ssh-ed25519
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: rekeying in progress
debug1: rekeying in progress
debug1: SSH2_MSG_KEX_ECDH_REPLY received
debug1: Server host key: ssh-ed25519 SHA256:TNCABq76r1cje139ct5VgL3tg3NvxhPS0UZpTMTqM/Q
debug1: ssh_packet_send2_wrapped: resetting send seqnr 32955
debug1: ssh_set_newkeys: rekeying out, input 1693124 bytes 17107 blocks, output 10736248568 bytes 134
185522 blocks
debug1: rekey out after 134217728 blocks
debug1: dequeue packet: 94
debug1: SSH2_MSG_NEWKYS sent
debug1: expecting SSH2_MSG_NEWKYS
debug1: ssh_packet_read_poll2: resetting read seqnr 8476
debug1: SSH2_MSG_NEWKYS received
debug1: ssh_set_newkeys: rekeying in, input 1693136 bytes 17108 blocks, output 10736281356 bytes 4098
blocks
debug1: rekey in after 134217728 blocks
scp: debug1: fd 6 clearing 0_NONBLOCK
scp: debug1: fd 0 clearing 0_NONBLOCK
arch-os.qcow2
100% 10GB 94.9MB/s 01:47

debug1: client_input_channel_req: channel 0 rtype exit-status reply 0
debug1: channel 0: free: client-session, nchannels 1
Transferred: sent 10745986728, received 1694780 bytes, in 108.0 seconds
Bytes per second: sent 99457473.5, received 15685.7
debug1: Exit status 0
abir@ahammed-20101197: ~

```

The transaction was successful. After that we can use `virsh migrate` command to directly migrate the config file from our machine to the destination machine.

```

sudo virsh migrate --offline --persistent --verbose archlinux
qemu+ssh://ub41201@172.18.188.108/system

```

```

abir@ahammed-20101197:~$ sudo virsh migrate --offline --persistent archlinux qemu+ssh://ub41201@172.18.188.108/system --verbose
ub41201@172.18.188.108's password:
Migration: [100 %]
abir@ahammed-20101197:~$

```

## In friend's machine:

After the successful migration, we just have to change our copied image's directory to `/var/lib/libvirt/images/`.

```

sudo cp arch-os.qcow2 /var/lib/libvirt/images/

```

```
ub41201@tahmid-20101555:~$ sudo cp arch-os.qcow2 /var/lib/libvirt/images/
[sudo] password for ub41201:
```

Then, we can use our regular `virsh` command to start our newly migrated VM.

```
ub41201@tahmid-20101555:~$ sudo virsh list --all
Id  Name      State
--  --
-   archlinux  shut off

ub41201@tahmid-20101555:~$ sudo virsh start archlinux
Domain 'archlinux' started

ub41201@tahmid-20101555:~$ sudo virsh console archlinux
Connected to domain 'archlinux'
Escape character is ^] (Ctrl + ])

ahammed-20101197 login: abir
Password:
Last login: Thu Feb 15 12:59:57 on ttys0
[abir@ahammed-20101197 ~]$ neofetch
               _`_
               .o+`_
               `ooo/
               `+oooo:
               `+oooooo:
               -+ooooooo+:
               `/:-:++oooo+:
               `/++++/++++++:
               `/+++++++/+++++:
               `/+++oooooooooooo/`_
               ./ooooooo+-+osssssso+`_
               .ooooooo-` ``/osssss+-`_
               -osssssso.      :ssssssso.      abir@ahammed-20101197
               :osssssss/      osssssso+++.      OS: Arch Linux x86_64
               /ossssssss/      +sssssooo/-      Host: KVM/QEMU (Standard PC (Q35 + ICH9, 2009) pc-q35-6.2)
               `osssssso+/-`-      -:/+osssss+-`_ Kernel: 6.7.4-arch1-1
               +ssso+:-`       ``-/+oso:      Uptime: 17 secs
               `++:.          ``-/+/`      Packages: 245 (pacman)
               .`             ``/`      Shell: bash 5.2.26
                                         Resolution: 1024x768
                                         Terminal: /dev/ttys0
                                         CPU: 11th Gen Intel i5-11400 (3) @ 2.592GHz
                                         GPU: 00:01.0 Red Hat, Inc. Virtio 1.0 GPU
                                         Memory: 154MiB / 2014MiB
               [color bar]
```