Creating a KVM-based virtual machine using GUI and CLI

1. First I'll update the system using these commands:

```
junaid@abrar-21141023:~$ sudo apt update && sudo apt upgrade
[sudo] password for junaid:
```

2. Then using this command, I'll check if virtualization is enabled. If the value is more than 0, then virtualization is enabled

```
junaid@abrar-21141023:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
24
```

3. Now, we'll check if KVM virtualization is enabled

```
junaid@abrar-21141023:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

4. If not enabled, use this command to install the cpu-checker package

```
junaid@abrar-21141023:~$ sudo apt install -y cpu-checker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cpu-checker is already the newest version (0.7-1.3build1).
The following package was automatically installed and is no longer required:
  brave-keyring
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

5. Run the command below to install KVM and additional virtualization packages on Ubuntu 22.04

```
junaid@abrar-21141023:~$ sudo apt install -y qemu-kvm virt-manager libvirt-daemo
n-system virtinst libvirt-clients bridge-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
bridge-utils is already the newest version (1.7-1ubuntu3).
virt-manager is already the newest version (1:4.0.0-1).
virtinst is already the newest version (1:4.0.0-1).
libvirt-clients is already the newest version (8.0.0-1ubuntu7.10).
libvirt-daemon-system is already the newest version (8.0.0-1ubuntu7.10).
qemu-system-x86 is already the newest version (1:6.2+dfsg-2ubuntu6.21).
The following package was automatically installed and is no longer required:
  brave-keyring
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

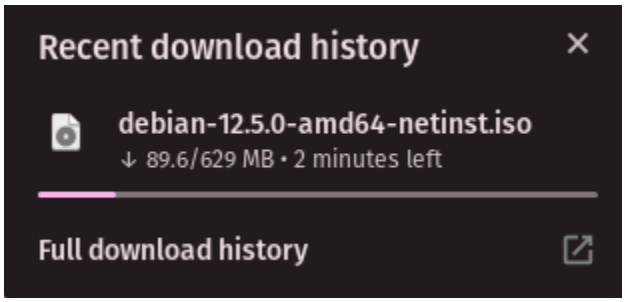
6. Start & Enable the Virtualization Daemon and confirm the virtualization daemon is running

```
junaid@abrar-21141023:~$ sudo systemctl enable --now libvirtd && sudo systemctl
start libvirtd
junaid@abrar-21141023:~$ sudo systemctl status libvirtd
● libvirtd.service - Virtualization daemon
     Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor pres>
     Active: active (running) since Tue 2024-06-11 23:57:39 +06; 21min ago
TriggeredBy: ● libvirtd-ro.socket
             ● libvirtd-admin.socket
             ● libvirtd.socket
       Docs: man:libvirtd(8)
             https://libvirt.org
   Main PID: 1216 (libvirtd)
      Tasks: 21 (limit: 32768)
     Memory: 43.7M
        CPU: 587ms
     CGroup: /system.slice/libvirtd.service
             ├─1216 /usr/sbin/libvirtd
             ├─1649 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/defa>
             └─1650 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/defa>
```

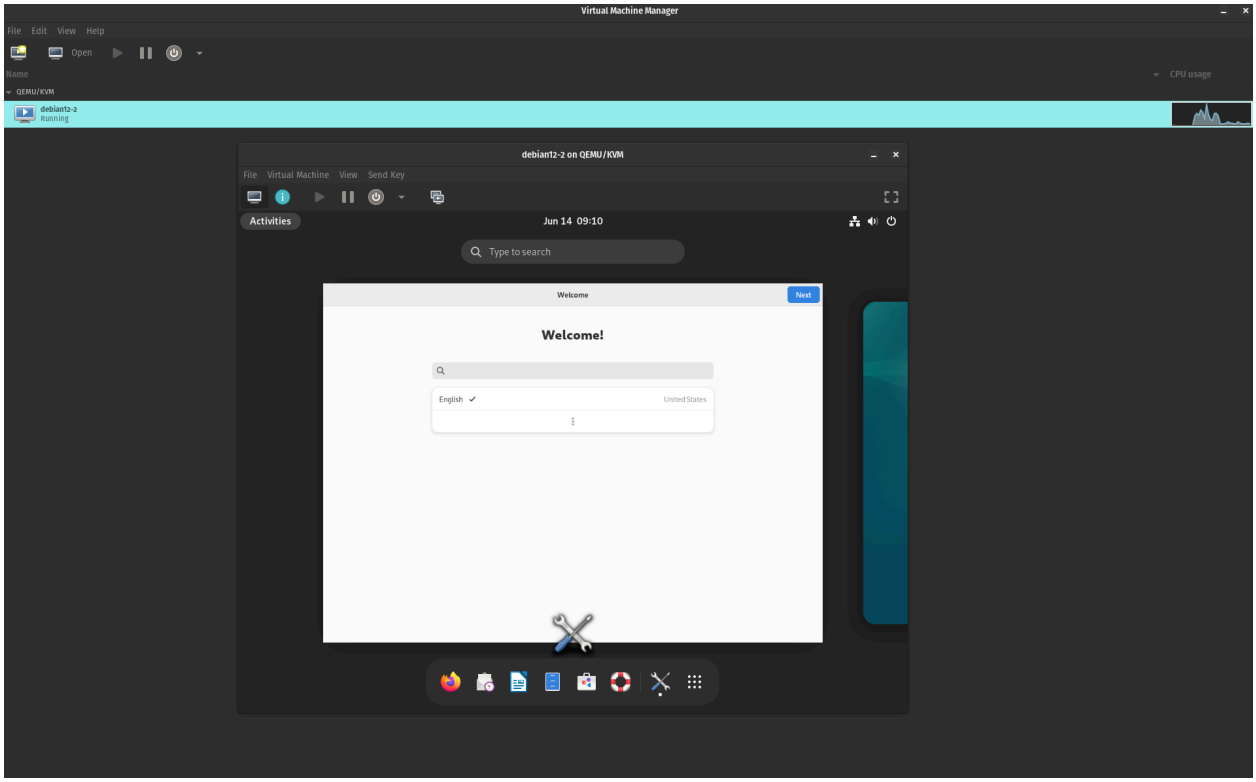
7. In addition, you need to add the currently logged-in user to the KVM and libvirt groups so that they can create and manage virtual machines

```
junaid@abrar-21141023:~$ sudo usermod -aG kvm junaid
junaid@abrar-21141023:~$ sudo usermod -aG libvirt junaid
```

8. For the virtual machine ISO, I've downloaded Debian 12.5.0



9. I've created a virtual machine using the Virtual Machine Manager GUI



Now, I will create a shared folder between my Host machine and Guest Machine(VM) using CLI

1. Installing ‘virtisofsd’ package on my host machine to enable the file sharing for QEMU/KVM

```
junaid@abrar-21141023:~$ sudo apt update
sudo apt install qemu-system-x86 qemu-utils virt-manager libvirt-daemon-system l
ibvirt-clients
```

2. Now, I will create a directory using the mkdir command named the “shared_folder” and make sure that the permissions of this directory allow access from the guest

```
junaid@abrar-21141023:~/Desktop$ mkdir -p shared
junaid@abrar-21141023:~/Desktop$ chmod 777 shared/
```

3. We need to start the ‘virtiofsd’ Daemon with the correct path to the shared folder. **This will create a socket for communication.**

```
junaid@abrar-21141023:~/Desktop$ sudo /usr/lib/qemu/virtiofsd --so
cket-path=/tmp/vhostqemu -o source=/home/junaid/Desktop/shared -o
cache=always -o no_posix_lock
virtio_session_mount: Waiting for vhost-user socket connection...
```

Note: Keep this host terminal running!

4. Configure the VM to use Virtofs
    - Open Virt-Manager
    - Select your VM
    - Click on “Details” tab and “Add Hardware”
    - Select Filesystem
    - In the “Driver”, select “virtio”
    - Set “Source Path”, in my case ‘/home/junaid/Desktop/shared’
    - Set “Target Path” to the name you want
    - Click “Finish”
5. Now we need to modify the VM configuration to include the virtiofs filesystem. This involves editing the VM's XML configuration
    - List the running VMs
    - Edit the VM's XML configuration using virsh edit and make sure the absolute path is there

```
junaid@abrar-21141023:~/Desktop$ sudo virsh list --all
 Id   Name          State
-----------------------------
 -    debian12-2    shut off
```

```
  </controller>
  <filesystem type='mount' accessmode='passthrough'>
    <driver type='virtiofs'/>
    <binary path='/usr/lib/qemu/virtiofsd'/>
    <source dir='/home/junaid/Desktop/shared'/>
    <target dir='shared'/>
    <alias name='fs0'/>
    <address type='pci' domain='0x0000' bus='0x07' slot='0x00' function='0x0'/>
  </filesystem>
  <interface type='network'>
```

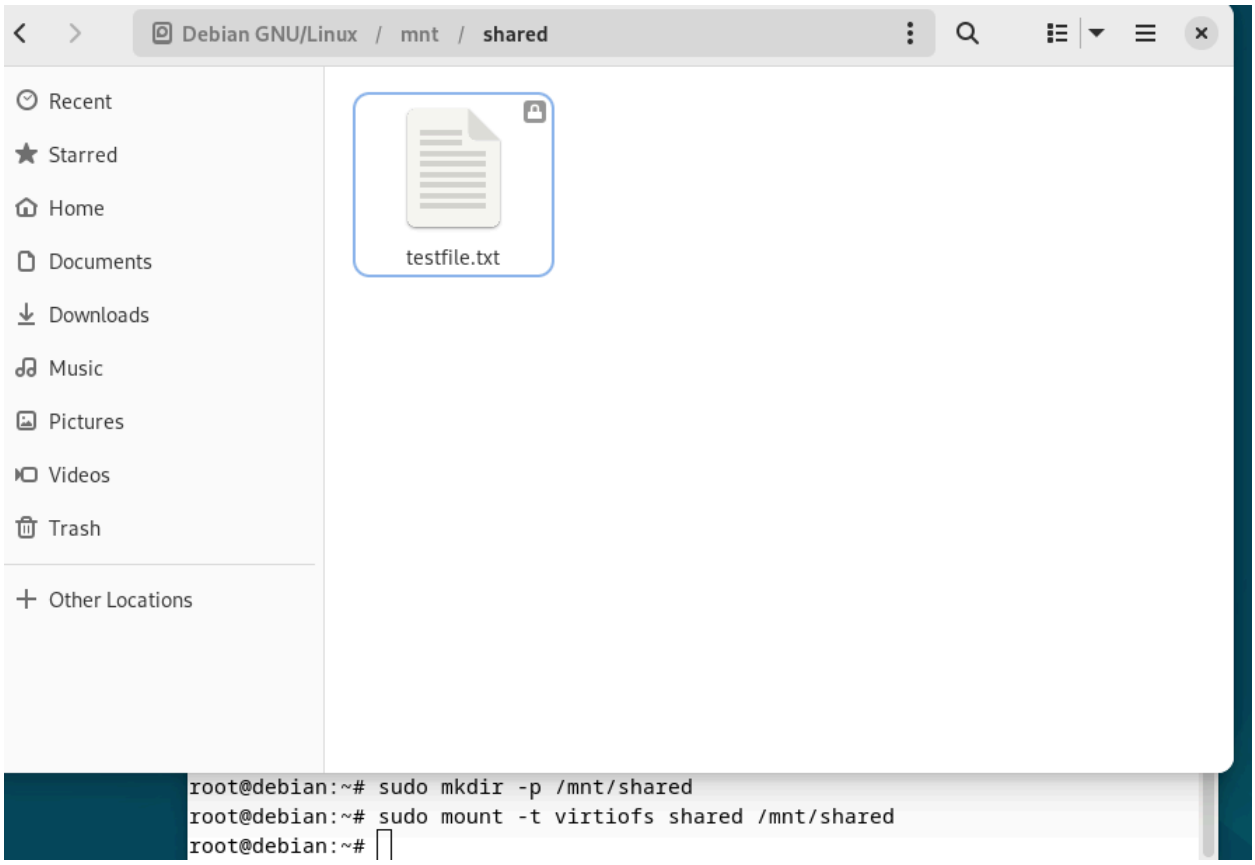
6. In my guest machine, now I need to make a directory for the shared folder. Before doing that, I’ll need to **switch the user to root**.

```
junaid@debian:~$ su -
Password:
root@debian:~# sudo mkdir -p /mnt/shared
root@debian:~# sudo mount -t virtiofs shared /mnt/shared
root@debian:~#
```

7. To test if my shared folder is working currently, I will echo a test file, which should also appear in my guest machine.

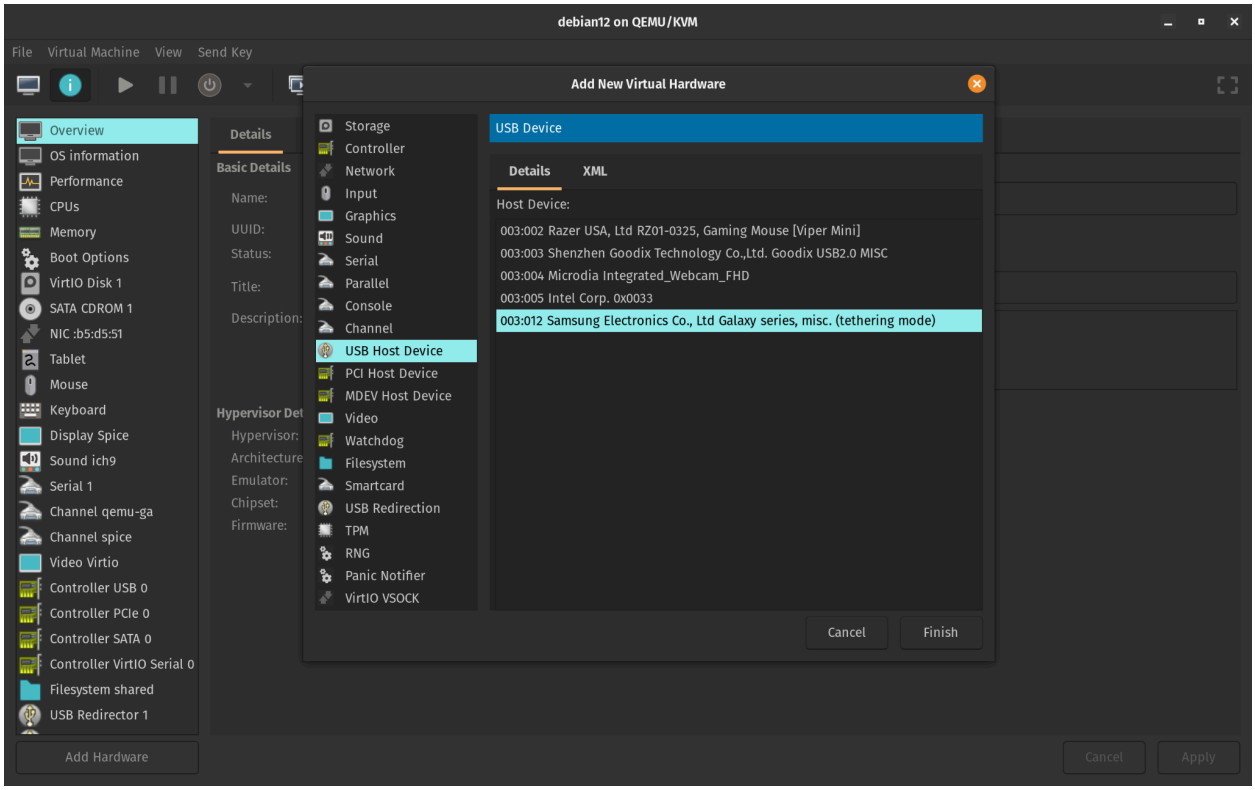
```
junaid@abrar-21141023:~/Desktop$ echo "Hello from Host" > /home/ju
naid/Desktop/shared/testfile.txt
```

And voila, here it is!



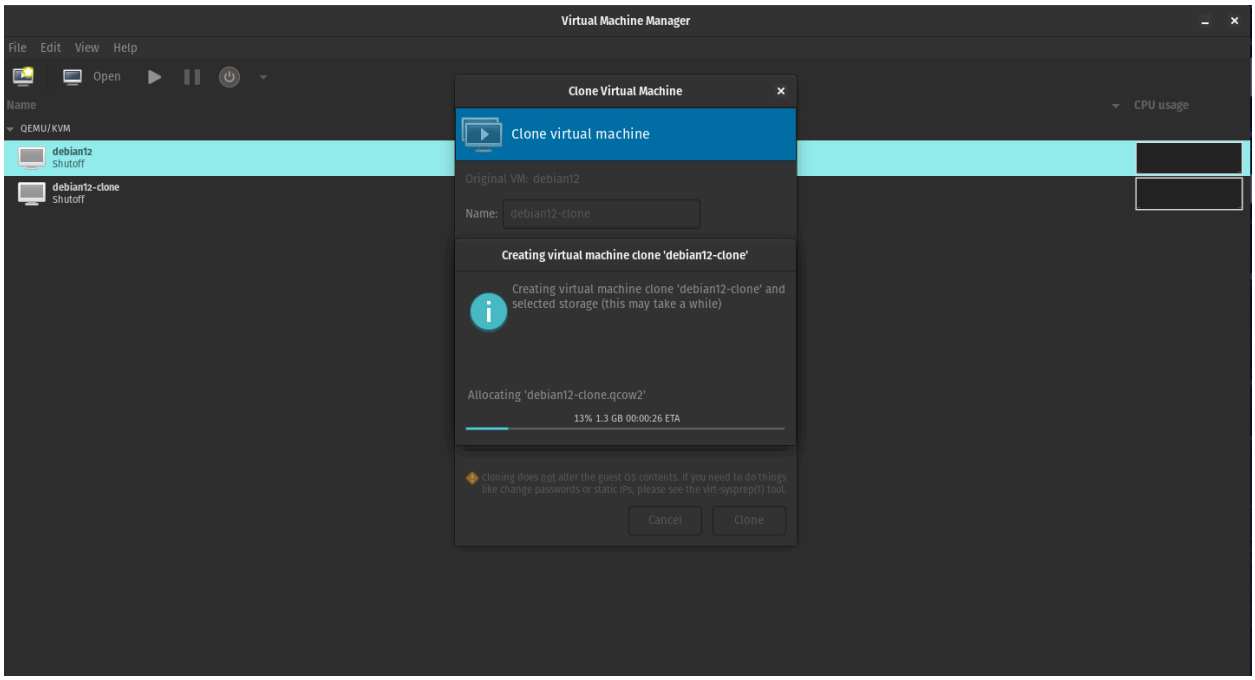
Step 6

Connecting an Android Phone to my Virtual Machine:

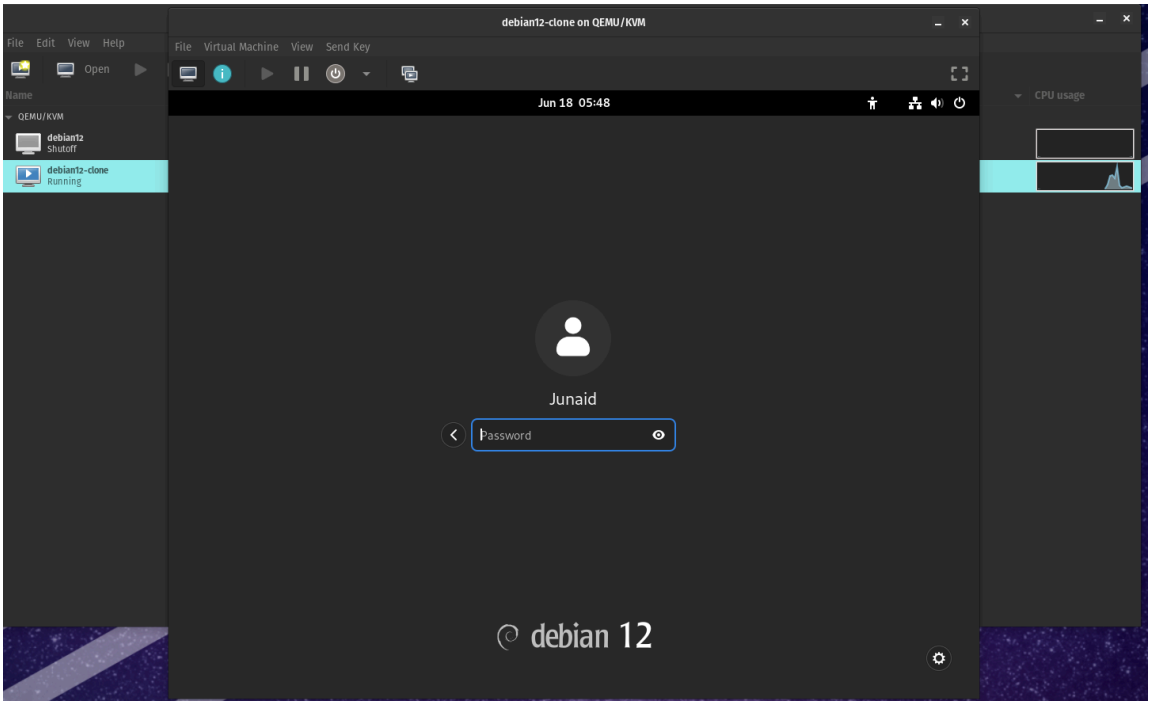


- Open VM settings
- Add new virtual hardware
- Add the USB from the list
- Apply changes

Step 7
Cloning my Virtual Machine using GUI:



- Successfully cloned



Cloning my Virtual Machine using CLI:

- **First, we need to use the virt-clone package to clone the VM**

```
junaid@abrar-21141023:~$ sudo apt-get install virtinst
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
virtinst is already the newest version (1:4.0.0-1).
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
```

- **Use this command to clone, specifying the original VM and clone VM's name and the absolute path**

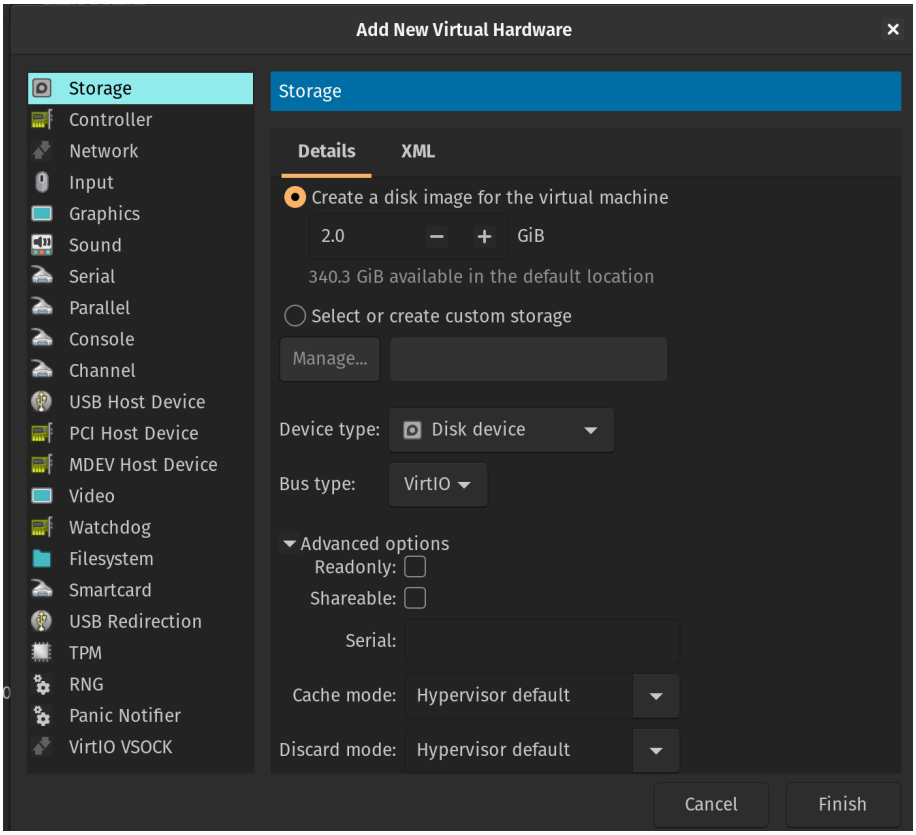
```
junaid@abrar-21141023:~$ sudo virt-clone --original debian12 --name debian12_cli
_clone --file /var/lib/libvirt/images/ubuntu_vm_clone.img
Allocating 'ubuntu_vm_clone.img'                          | 4.9 GB  00:06 ...

Clone 'debian12_cli_clone' created successfully.
```

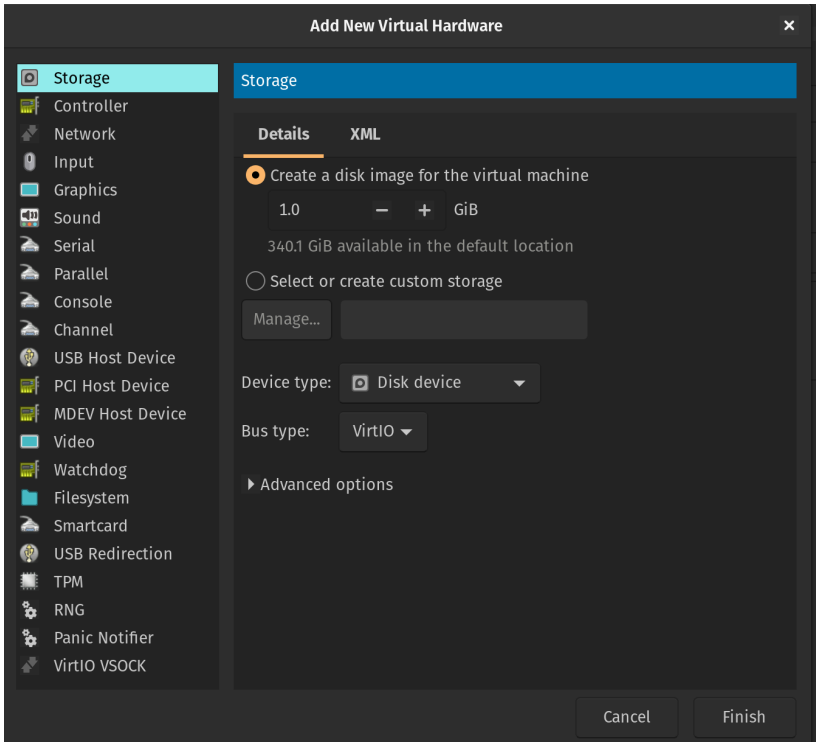
Step 8

Adding two hard disks to the cloned VM using GUI

- Open virt-manager
- Select the cloned VM
- Open VM details
- Configure size, format, location and add the hard disk



Second one,



Now I have three virtual disks. VirtIO Disk 1, VirtIO Disk 2, and VirtIO Disk 3.

Step 9

Now, we will again add virtual disks to the clone VM, but using CLI this time. Let's get to it.

First we need to create the necessary disk images using this commands

```
junaid@abrar-21141023:~$ sudo qemu-img create -f qcow2 /var/lib/libvirt/images/f
irst-disk.img 1G
[sudo] password for junaid:
Formatting '/var/lib/libvirt/images/first-disk.img', fmt=qcow2 cluster_size=6553
6 extended_l2=off compression_type=zlib size=1073741824 lazy_refcounts=off refco
unt_bits=16
```

It's time to create the second disk

```
junaid@abrar-21141023:~$ sudo qemu-img create -f qcow2 /var/lib/libvirt/images/s
econd-disk.img 1G
Formatting '/var/lib/libvirt/images/second-disk.img', fmt=qcow2 cluster_size=655
36 extended_l2=off compression_type=zlib size=1073741824 lazy_refcounts=off refc
ount_bits=16
```

Then attach the disk images to the VM

```
junaid@abrar-21141023:~$ virsh attach-disk debian12-clone /home/junaid/vm-images
/first-disk.img vdd --targetbus virtio --persistent
Disk attached successfully

junaid@abrar-21141023:~$ virsh attach-disk debian12-clone /home/junaid/vm-images
/second-disk.img vde --targetbus virtio --persistent
Disk attached successfully
```

Voila, disk images added successfully.