KVM Virtual Machine Manager and Virtual Machines on external drives

by Lorenzo Bettini

Last year, I blogged about my first experiences with KVM and Virtual Machine Manager.

Then, I stopped using KVM because I've always found VirtualBox easier for my experiments. In particular, with VirtualBox, it is trivial to store virtual machines on an external drive (I mean, a fast external SD, of course): you specify to use a directory on the external drive, and all information about the virtual machine will be stored there. Then, you attach the drive to another computer with VirtualBox and open the virtual machine from the external drive. Easy!

Things are more complicated with KVM, QEMU, and Virtual Machine Manager. Even making QEMU access an external drive requires additional configuration steps.

In this blog post, I'll summarize the steps to achieve that.

I'll first show the manual export/import procedure for the machines' metadata information. Then, I'll show a different approach based on symlinks.

It was time to try KVM again because it's faster than VirtualBox.

I'll describe the installation steps for EndeavourOS and pure Arch Linux. I guess the steps for other distributions are similar.

Installation and configuration

Let's install a few packages for KVM, QEMU, and the Virtual Machine Manager:

```
sudo pacman -S --needed virt-manager qemu-base libvirt edk2-ovmf dnsmasq vde2 bridge-utils iptables-nft dmidecode
```

If you get this message, accept to remove "iptables":

```
iptables-nft and iptables are in conflict. Remove iptables? [y/N]
```

To use your user without entering the root password, we need to edit the file “/etc/libvirt/libvirtd.conf” and uncomment the following lines:

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

Or, append them at the end of the file:

```
sudo tee -a /etc/libvirt/libvirtd.conf > /dev/null <<EOT
unix_sock_group = "libvirt"
unix_sock_rw_perms = "0770"
EOT
```

Add your user account to the “libvirt” group.

```
sudo usermod -a -G libvirt $(whoami)
```

Now comes the crucial part for letting QEMU handle machines on external drives: we need to add our user to “/etc/libvirt/qemu.conf”. This can be done by setting the appropriate entries in the file or by simply appending the entries at the end of the file:

```
sudo tee -a /etc/libvirt/qemu.conf > /dev/null <<EOT
user = "$(whoami)"
group = "$(whoami)"
EOT
```

If you want to start the virtualization service and the default virtual network automatically at boot, you run:

```
sudo systemctl enable libvirtd.service
sudo virsh net-autostart default
```

Since I'm not using virtual machines daily, I prefer to start them when needed, so I don't run the above commands. Of course, I must remember to run these commands (note, for the network is "start" instead of "autostart") before starting the "Virtual Machine Manager":

```
sudo systemctl start libvirtd.service
sudo virsh net-start default
```

Remember you can always use:

```
systemctl status libvirtd.service
```

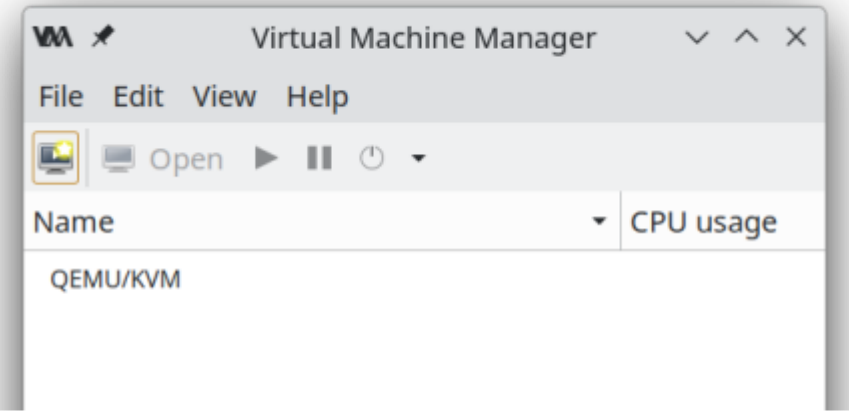
to see the service status and possible errors shown when running this command.

OK, time to reboot now.

Let's create a virtual machine on an external drive

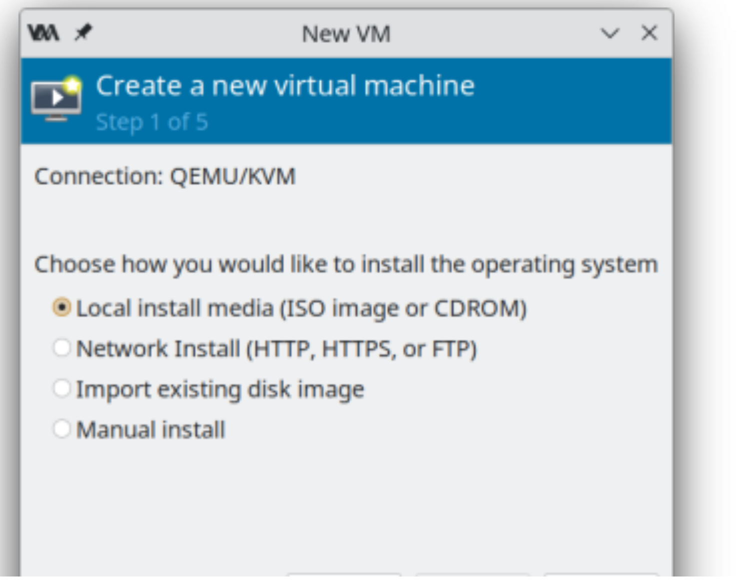
I created a directory "kvm/images" on my external USB SD to store the virtual machine images.

Let's start the "Virtual Machine Manager" program. We should see "QEMU/KVM":



Let’s create a new virtual machine with the leftmost toolbar button.

I specify a local ISO.



New VM

Create a new virtual machine

Step 1 of 5

Connection: QEMU/KVM

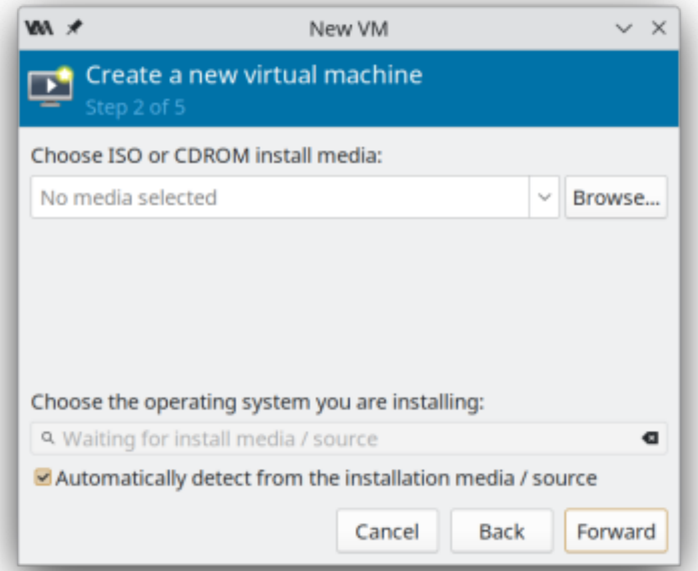
Choose how you would like to install the operating system

[✓ Local install media (ISO image or CDROM)]

Network Install (HTTP, HTTPS, or FTP)

Import existing disk image

Manual install



Choose ISO or CDROM install media:

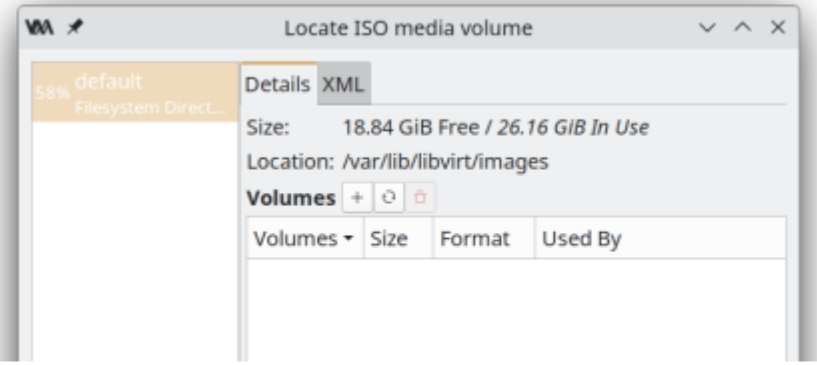
No media selected

Choose the operating system you are installing:

Waiting for install media / source

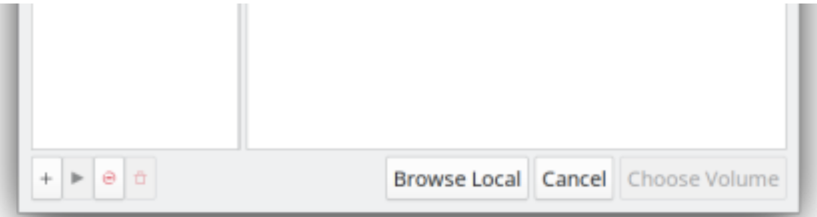
[✓] Automatically detect from the installation media / source

I don’t create a pool for ISOs and use “Browse Local” to select an ISO in my external drive.

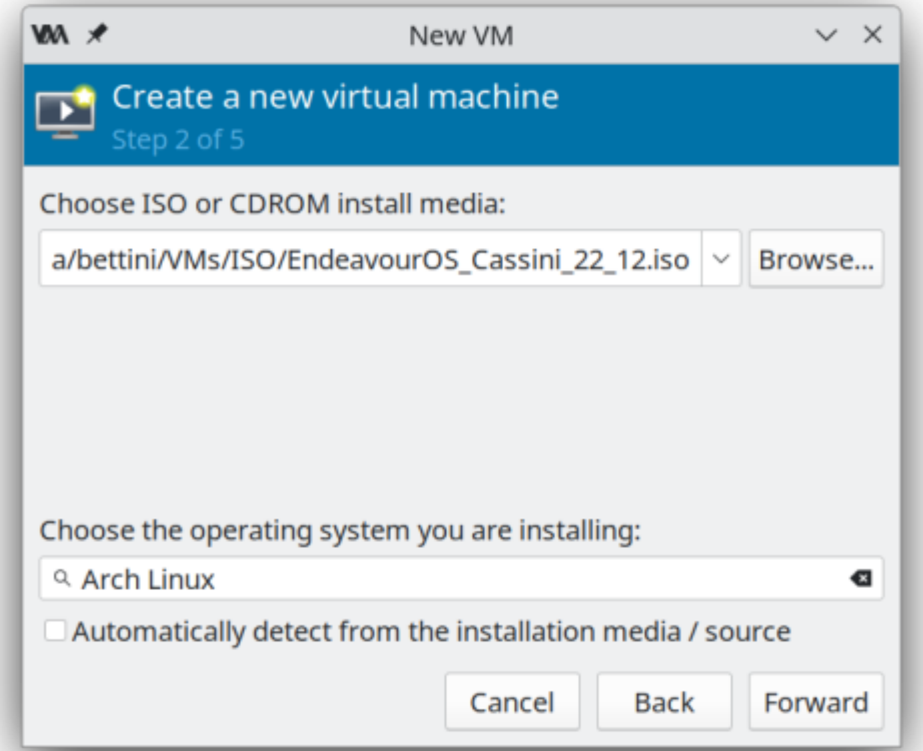


Size: 18.84 GiB Free / *26.16 GiB In Use*

Location: /var/lib/libvirt/images



In this example, I will install EndeavourOS on the virtual machine. I have to select the operating system manually (start typing, and you get completions):



Choose ISO or CDROM install media:

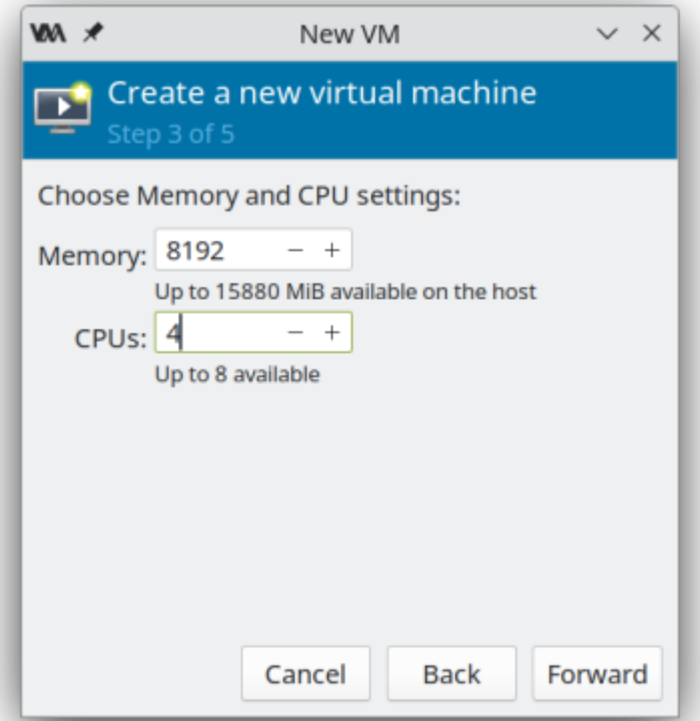
a/bettini/VMs/ISO/EndeavourOS_Cassini_22_12.iso

Choose the operating system you are installing:

Arch Linux

☐ Automatically detect from the installation media / source

Time to allocate resources for the virtual machine. I'm giving the VM half my RAM and half my cores:


New VM

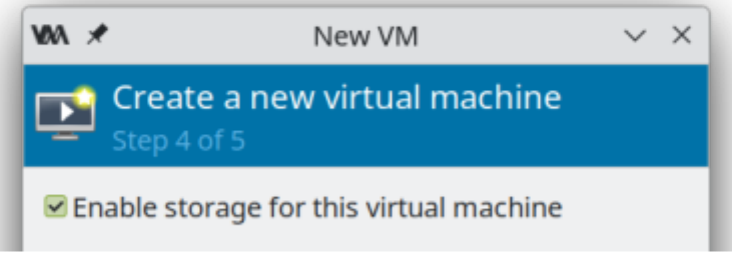
Create a new virtual machine
Step 3 of 5

Choose Memory and CPU settings:

Memory: 8192
Up to 15880 MiB available on the host

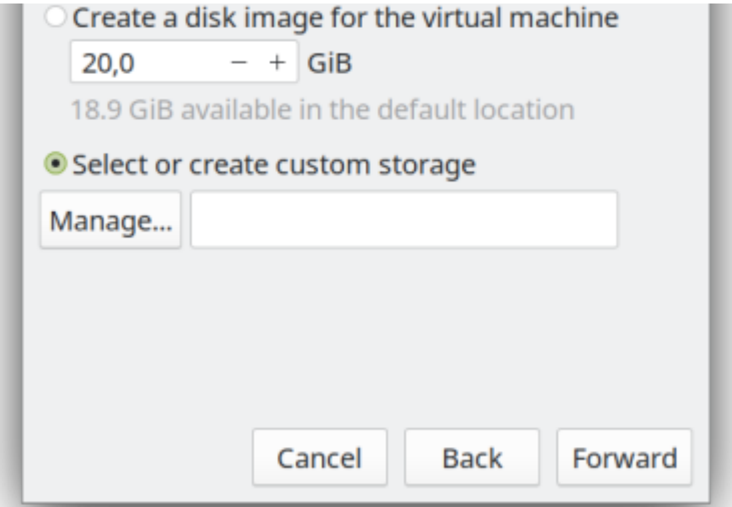
CPUs: 4
Up to 8 available

Now here's the essential part of disk selection. Remember, I want to use my external drive, so I select custom storage and press "Manage":

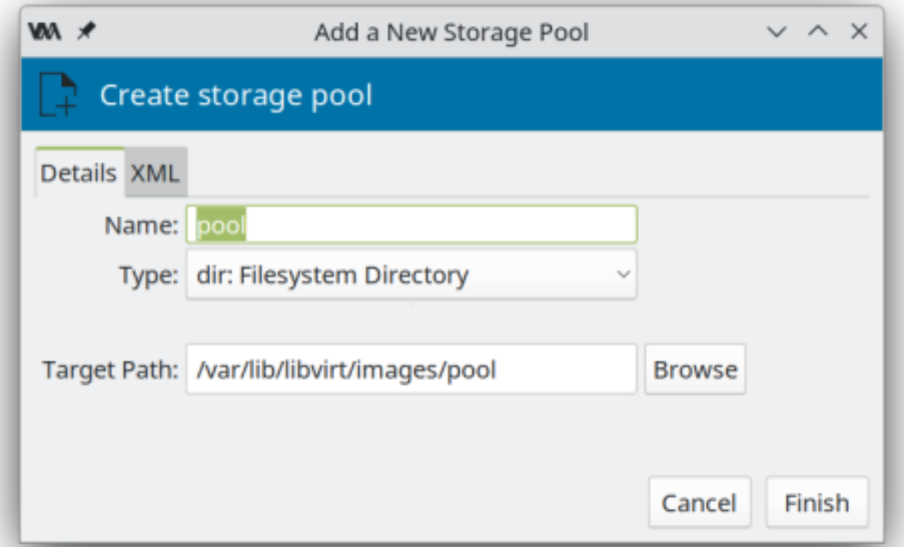

New VM

Create a new virtual machine
Step 4 of 5

☑ Enable storage for this virtual machine

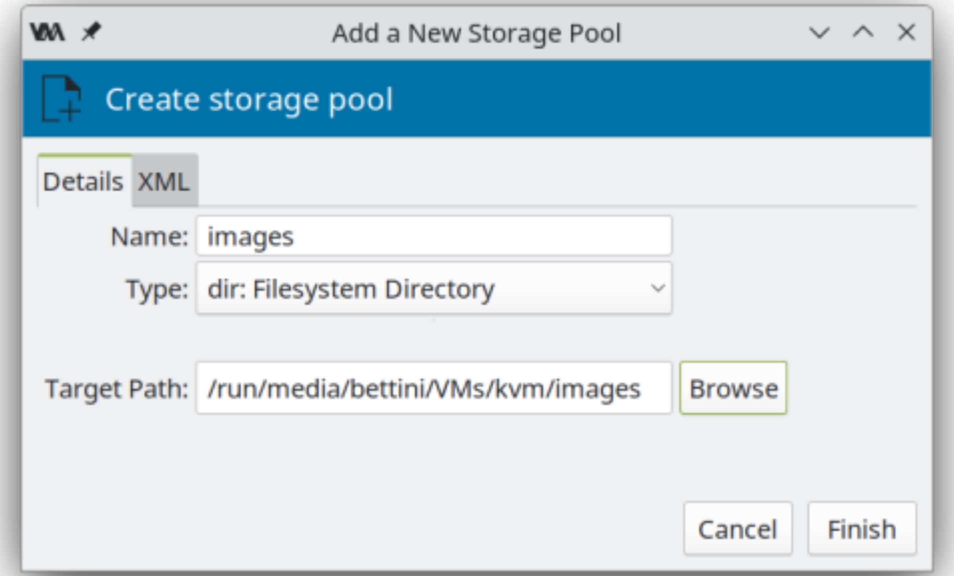

Create a disk image for the virtual machine
20,0 GiB
18.9 GiB available in the default location
Select or create custom storage
Manage...
Cancel
Back
Forward

In the following dialog, I use the “+” button in the bottom left corner to create a new pool:



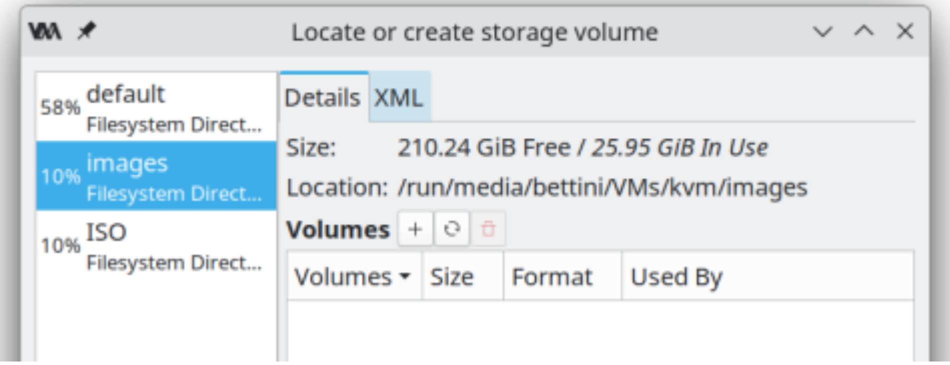
Name: pool
Type: dir: Filesystem Directory
Target Path: /var/lib/libvirt/images/pool

I give the pool the name "images" and specify the directory I mentioned above on my external drive:

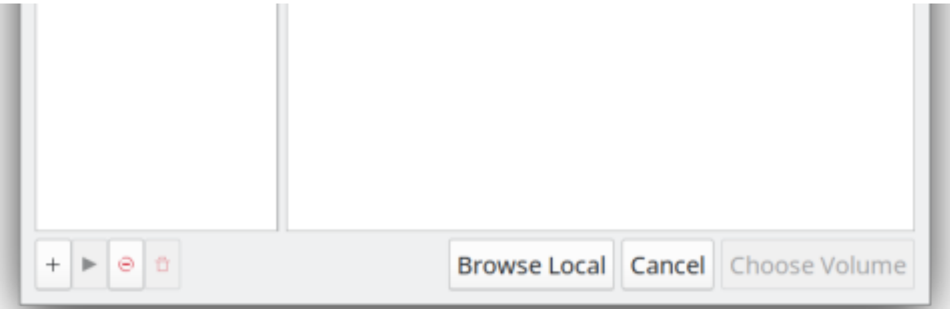


Name: images
Type: dir: Filesystem Directory
Target Path: /run/media/bettini/VMs/kvm/images

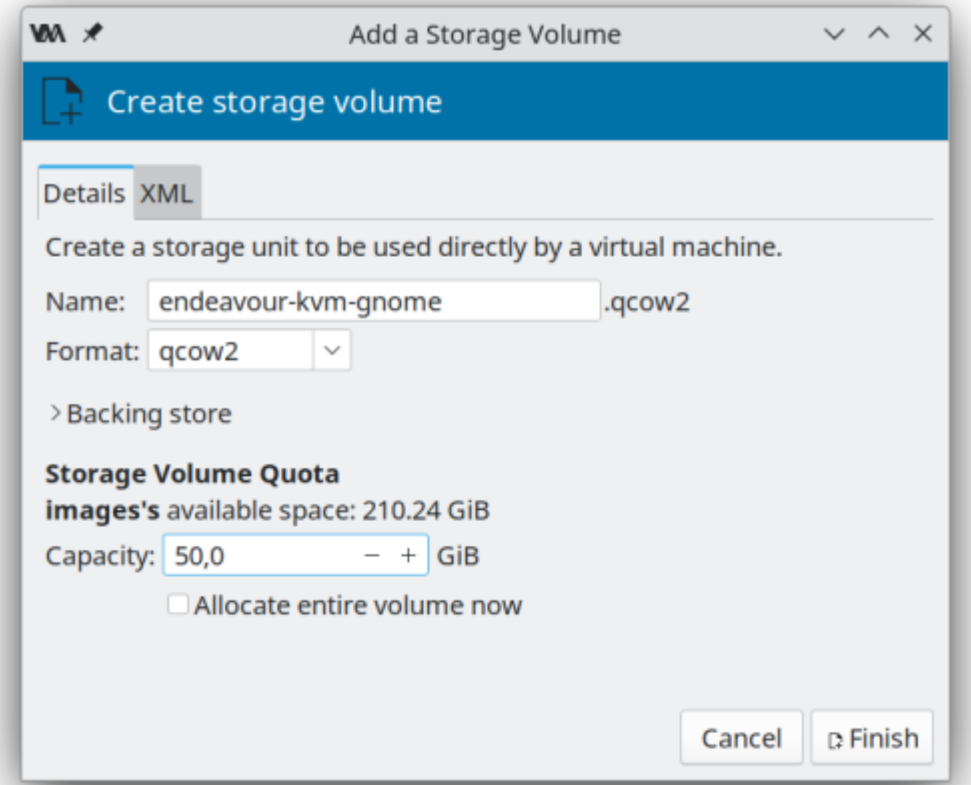
After pressing "Finish", I select the created pool and add a "Volume" (with the other "+" button)



Size: 210.24 GiB Free / *25.95 GiB In Use*
Location: /run/media/bettini/VMs/kvm/images



I give the disk image a proper name and enough size (recall that the image will NOT allocate all the size immediately, but only on-demand):



Create a storage unit to be used directly by a virtual machine.

Name: endeavour-kvm-gnome .qcow2

Format: qcow2

› Backing store

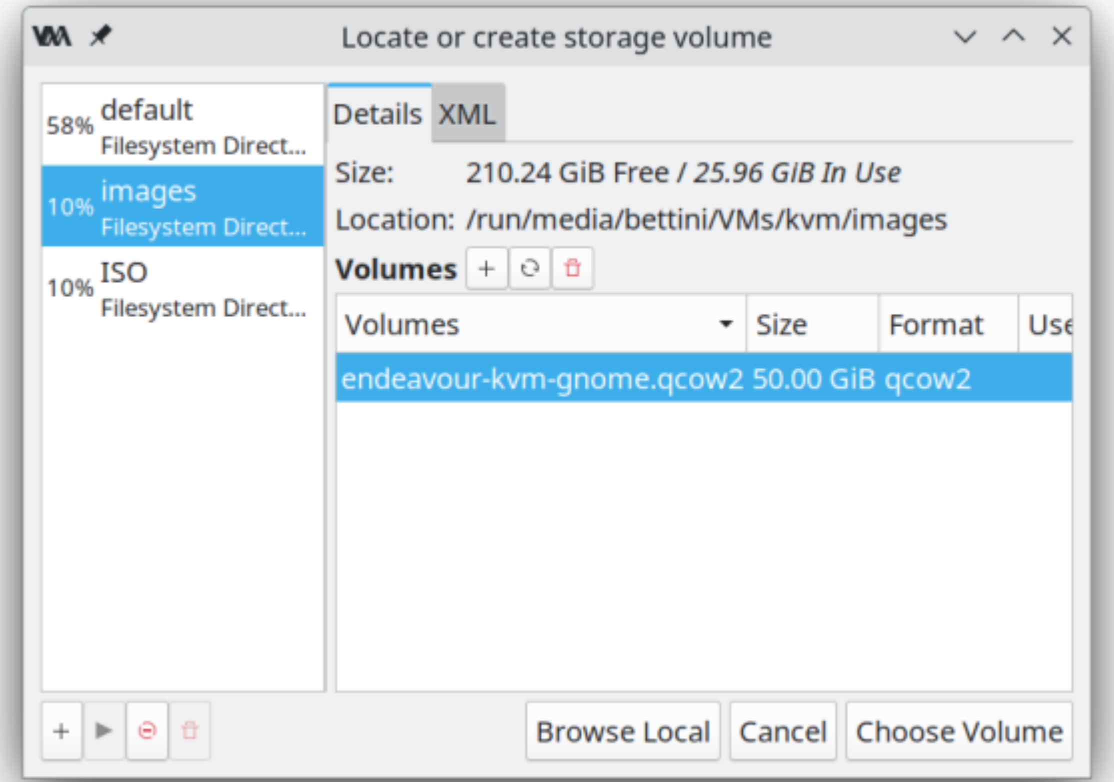
Storage Volume Quota

images's available space: 210.24 GiB

Capacity: 50,0 GiB

☐ Allocate entire volume now

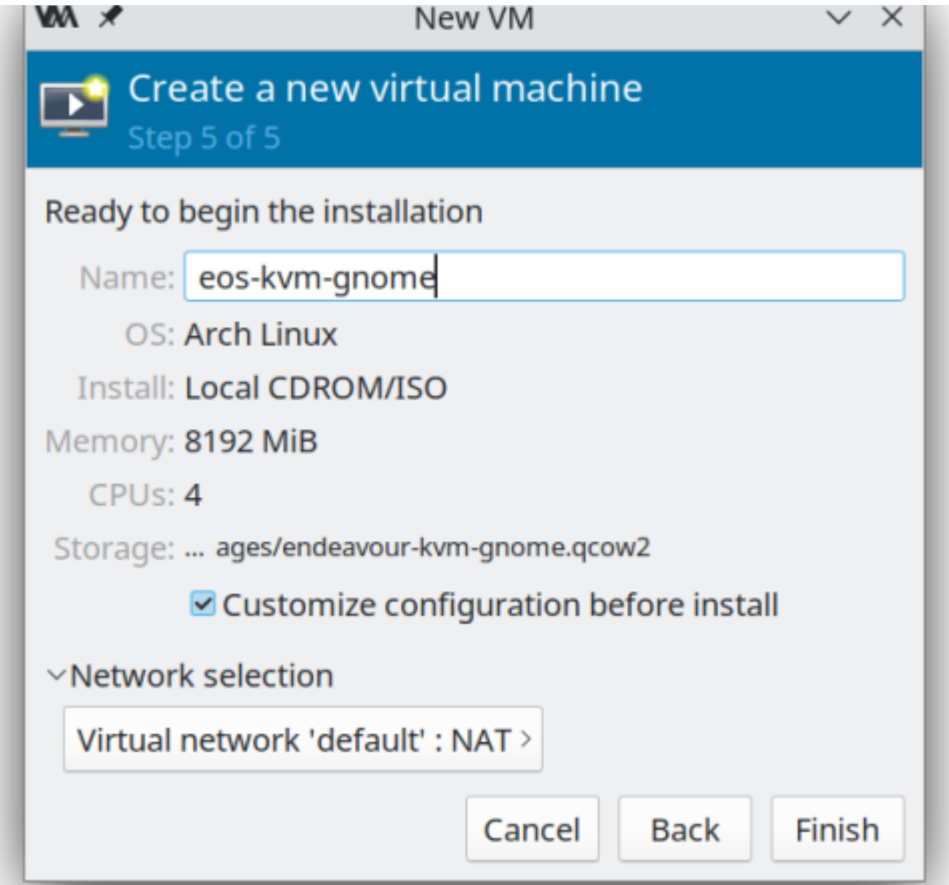
Select the volume and press "Choose Volume":



Size: 210.24 GiB Free / *25.96 GiB In Use*
Location: /run/media/bettini/VMs/kvm/images

Volumes	Size	Format	Use
endeavour-kvm-gnome.qcow2	50.00 GiB	qcow2	

On the final dialog, make sure the default network is selected and that you check "Customize configuration before install" (note that I also changed the name for the virtual machine):



Ready to begin the installation

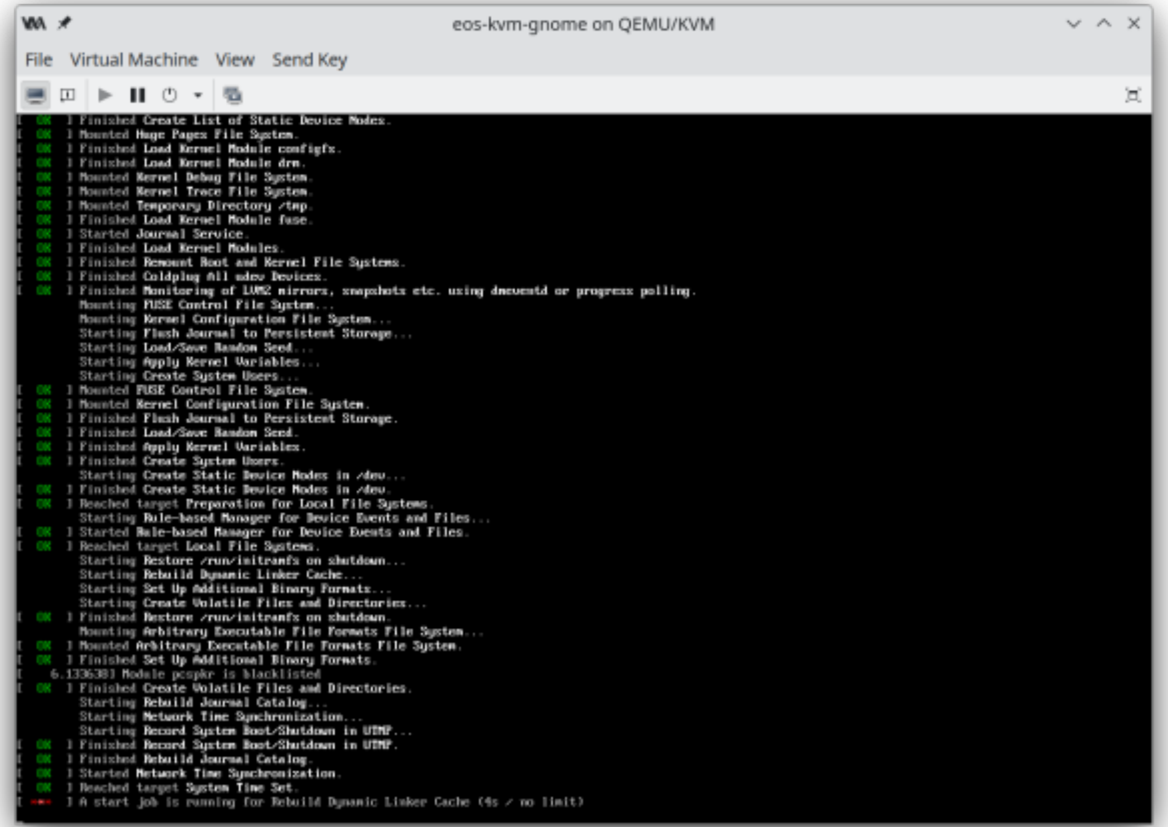
Name: eos-kvm-gnome
OS: Arch Linux
Install: Local CDROM/ISO
Memory: 8192 MiB
CPUs: 4
Storage: ... ages/endeavour-kvm-gnome.qcow2
[✓] Customize configuration before install

Network selection
Virtual network 'default' : NAT

Let's press "Finish," and get to the configuration dialog. I changed the Firmware from "BIOS" to "UEFI", pressed "Apply," and finally, we can start the installation with "Begin Installation".

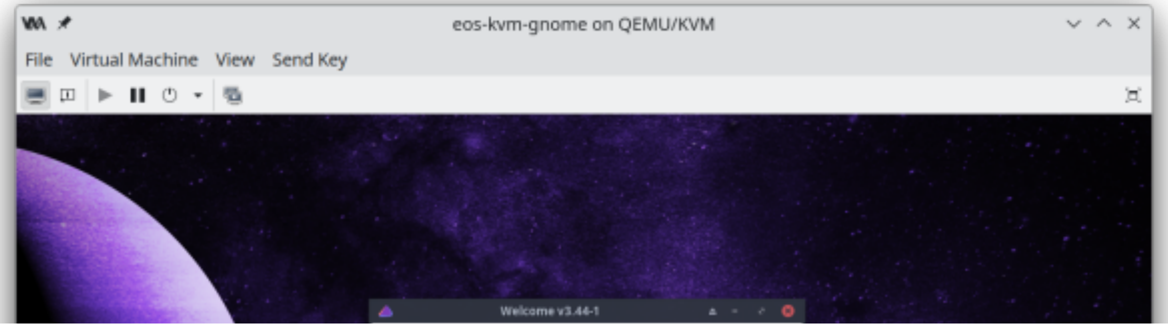
We should not get any error from QEMU because it cannot access the external drive, thanks to the configuration shown above in the qemu.conf file!

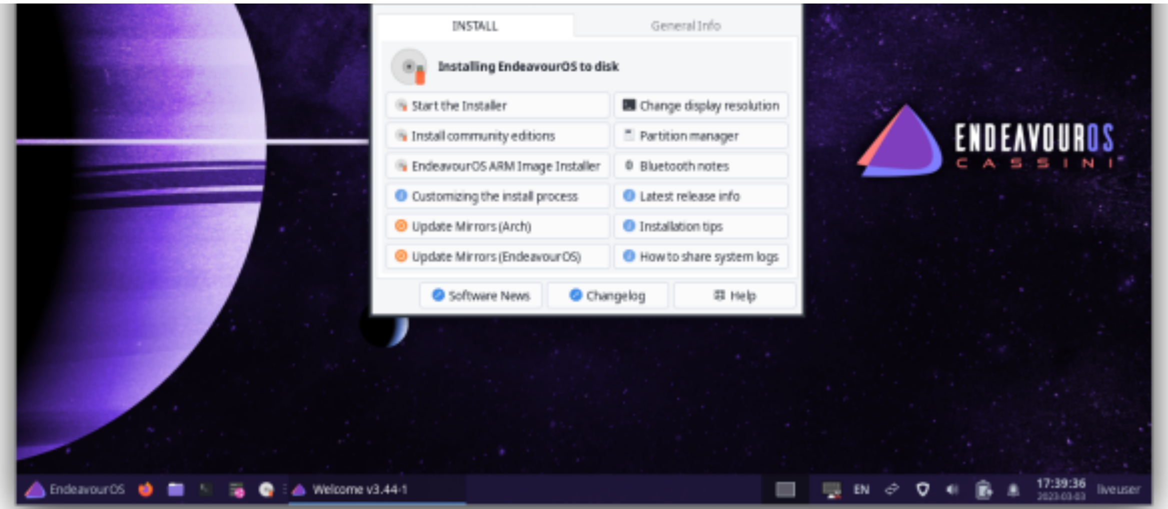
After the GRUB menu, we should see the installer log:



```
[  OK  ] Finished Create List of Static Device Nodes.
[  OK  ] Mounted Huge Pages File System.
[  OK  ] Finished Load Kernel Module configfs.
[  OK  ] Finished Load Kernel Module drm.
[  OK  ] Mounted Kernel Debug File System.
[  OK  ] Mounted Kernel Trace File System.
[  OK  ] Mounted Temporary Directory /tmp.
[  OK  ] Finished Load Kernel Module fuse.
[  OK  ] Started Journal Service.
[  OK  ] Finished Load Kernel Modules.
[  OK  ] Finished Remount Root and Kernel File Systems.
[  OK  ] Finished Coldplug All udev Devices.
[  OK  ] Finished Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling.
         Mounting FUSE Control File System...
         Mounting Kernel Configuration File System...
         Starting Flush Journal to Persistent Storage...
         Starting Load/Save Random Seed...
         Starting Apply Kernel Variables...
         Starting Create System Users...
[  OK  ] Mounted FUSE Control File System.
[  OK  ] Mounted Kernel Configuration File System.
[  OK  ] Finished Flush Journal to Persistent Storage.
[  OK  ] Finished Load/Save Random Seed.
[  OK  ] Finished Apply Kernel Variables.
[  OK  ] Finished Create System Users.
         Starting Create Static Device Nodes in /dev...
[  OK  ] Finished Create Static Device Nodes in /dev.
[  OK  ] Reached target Preparation for Local File Systems.
         Starting Rule-based Manager for Device Events and Files...
[  OK  ] Started Rule-based Manager for Device Events and Files.
[  OK  ] Reached target Local File Systems.
         Starting Restore /run/initramfs on shutdown...
         Starting Rebuild Dynamic Linker Cache...
         Starting Set Up Additional Binary Formats...
         Starting Create Volatile Files and Directories...
[  OK  ] Finished Restore /run/initramfs on shutdown.
         Mounting Arbitrary Executable File Formats File System...
[  OK  ] Mounted Arbitrary Executable File Formats File System.
[  OK  ] Finished Set Up Additional Binary Formats.
[    6.133638] Module pcspkr is blacklisted
[  OK  ] Finished Create Volatile Files and Directories.
         Starting Rebuild Journal Catalog...
         Starting Network Time Synchronization...
         Starting Record System Boot/Shutdown in UTMP...
[  OK  ] Finished Record System Boot/Shutdown in UTMP.
[  OK  ] Finished Rebuild Journal Catalog.
[  OK  ] Started Network Time Synchronization.
[  OK  ] Reached target System Time Set.
[ ***  ] A start job is running for Rebuild Dynamic Linker Cache (4s / no limit)
```

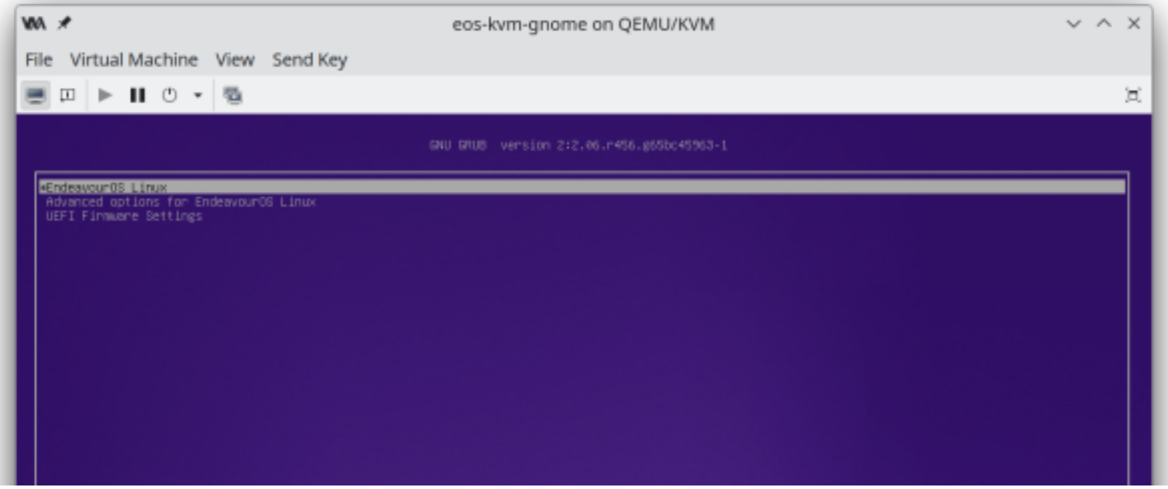
And then, the EndeavourOS installer dialog:





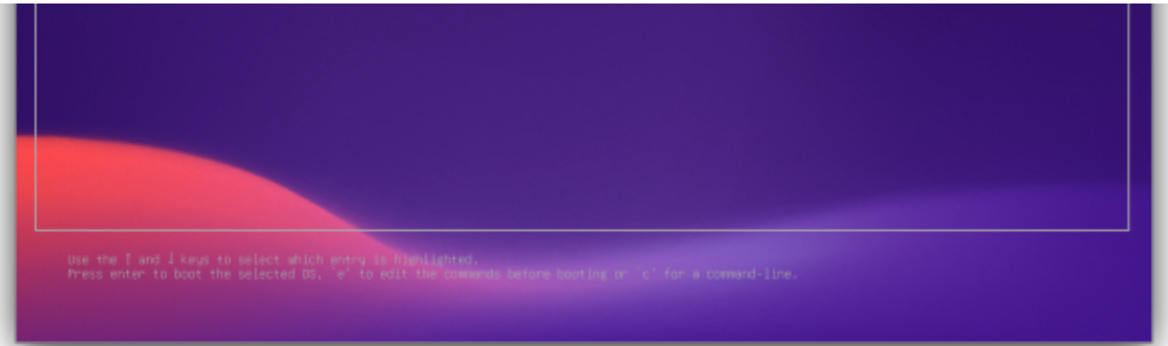
Since I've already blogged about EndeavourOS installation, I'll skip the detailed steps. I'll install the GNOME desktop environment and let the installer use the whole disk space with the BTRFS filesystem and SWAP with hibernate (later, I might want to check whether hibernate works in the VM).

In a few minutes, the installation finishes! We get to the GRUB menu of the installed system:



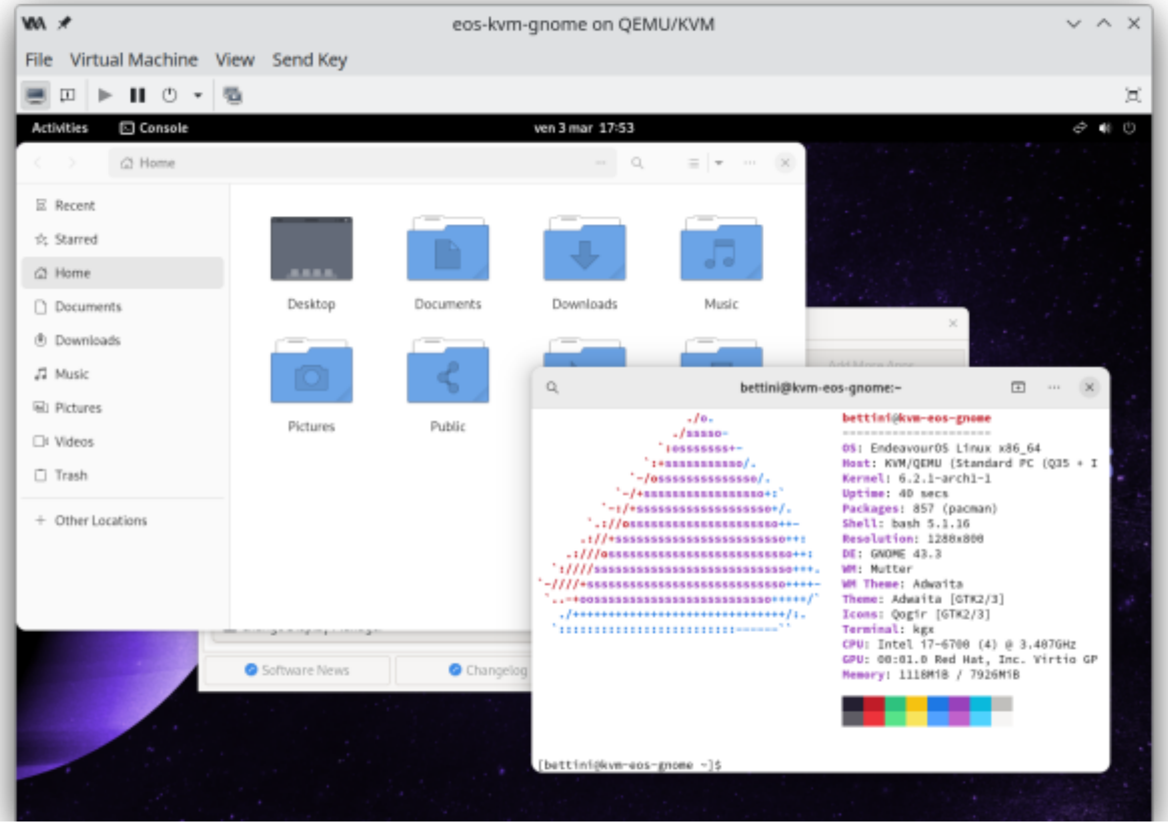
```
GNU GRUB  version 2:2.06.r456.g65bc45963-1

*EndeavourOS Linux
 Advanced options for EndeavourOS Linux
 UEFI Firmware Settings
```



```
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line.
```

And to the installed GNOME desktop:



```
bettini@kvm-eos-gnome
---------------------
OS: EndeavourOS Linux x86_64
Host: KVM/QEMU (Standard PC (Q35 + I
Kernel: 6.2.1-arch1-1
Uptime: 40 secs
Packages: 857 (pacman)
Shell: bash 5.1.16
Resolution: 1280x800
DE: GNOME 43.3
WM: Mutter
WM Theme: Adwaita
Theme: Adwaita [GTK2/3]
Icons: Qogir [GTK2/3]
Terminal: kgx
CPU: Intel i7-6700 (4) @ 3.407GHz
GPU: 00:01.0 Red Hat, Inc. Virtio GP
Memory: 1118MiB / 7926MiB

[bettini@kvm-eos-gnome ~]$
```

The disk image is correctly created in the external drive:

```
❯ ll /run/media/bettini/VMs/kvm/images
.rw------- 54G bettini 3 mar 17:46  endeavour-kvm-gnome.qcow2
```

And the information about the virtual machine is in the:

```
/etc/libvirt/qemu
├── autostart
├── networks
│   ├── autostart
│   │   └── default.xml -> /etc/libvirt/qemu/networks/default.xml
│   └── default.xml
└── eos-kvm-gnome.xml
```

Export the virtual machine

First, let’s shut down the machine.

Let’s export the virtual machine to use it from another computer. I understand that having the same software on the other host is crucial. Since I’m using EndeavourOS or Arch on my main computers, that is not a problem.

But isn’t the virtual machine already in an external drive? Why do I have to export it?

That’s the main difference with VirtualBox I mentioned at the beginning. The disk image is on an external drive, but the virtual machine information (configuration and metadata) is on a local XML file (see the listing of “/etc/libvirt/qemu” above; the XML file of the virtual machine is “eos-kvm-gnome.xml”, after the name I gave to the virtual machine when I created it).

Remember that the XML has an absolute path pointing to the disk image on the external drive:

```
<source file='/run/media/bettini/VMs/kvm/images/endeavour-kvm-gnome.qcow2'/>
```

So, again, in the other computers, the mount point of the external drive must be the same; otherwise, the absolute path must be manually adapted.

We could copy the XML file directly on the external drive (somewhere near the disk image to be easily found), e.g.:

```
sudo cp /etc/libvirt/qemu/eos-kvm-gnome.xml /run/media/bettini/VMs/kvm
```

Alternatively, if we don’t remember the location of the XML file, we can use the “dump” command.

For example, we can first list the current machines (in the example, I have only one):

```
❯ sudo virsh list --all
Id   Name          State
--------------------------------
-    eos-kvm-gnome   shut off
```

And then, we dump its XML configuration:

```
sudo virsh dumpxml eos-kvm-gnome > /run/media/bettini/VMs/kvm/eos-kvm-gnome.xml
```

We’re ready to import and use the VM on another computer

Import the virtual machine

I have already installed and configured KVM on another computer, following the same procedure at the beginning of the post.

Since I haven't enabled the services at boot time, I run the following:

```
sudo systemctl start libvirtd.service
sudo virsh net-start default
```

I connect the external drive and ensure it's mounted (remember, on the same mount point as in the other computer).

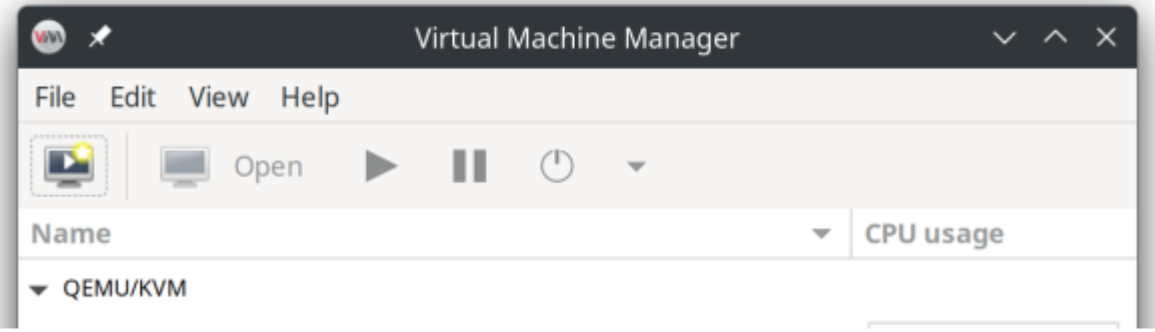
Then, I create the virtual machine information locally by using the XML file on the drive I created above:

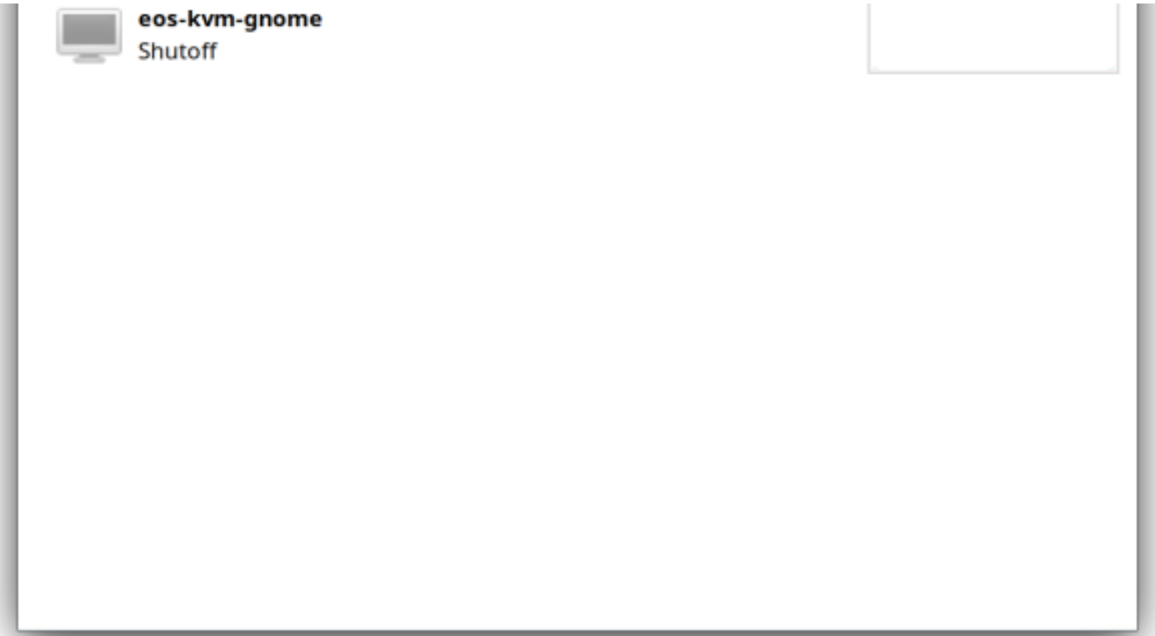
```
❯ sudo virsh define /run/media/bettini/VMs/kvm/eos-kvm-gnome.xml
Domain 'eos-kvm-gnome' defined from /run/media/bettini/VMs/kvm/eos-kvm-gnome.xml
```

We can verify that the XML is now in the directory of QEMU:

```
❯ ls -l /etc/libvirt/qemu/
total 8
drwxr-xr-x 1 root root 0 16 gen 20.56 autostart
-rw------- 1 root root 7693 4 mar 17.51 eos-kvm-gnome.xml
drwxr-xr-x 1 root root 40 4 mar 17.27 networks
```

Let's start "Virtual Machine Manager," and we can see the virtual machine:

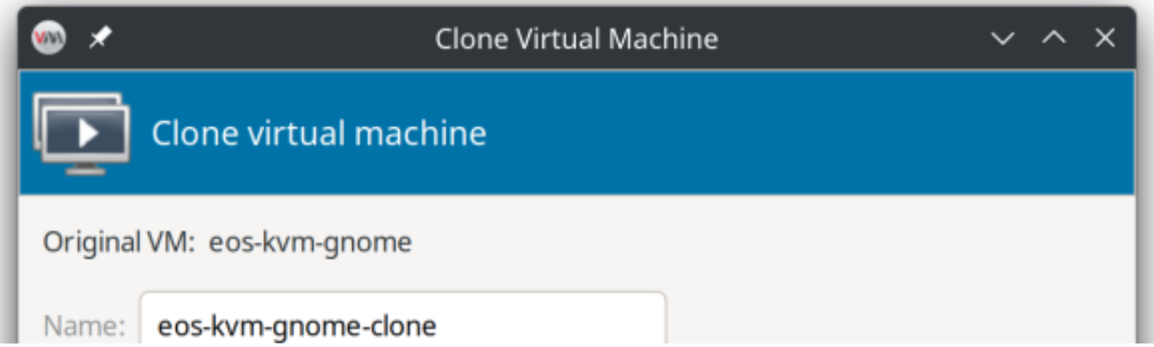




We can start it, and it should work as on the other computer.

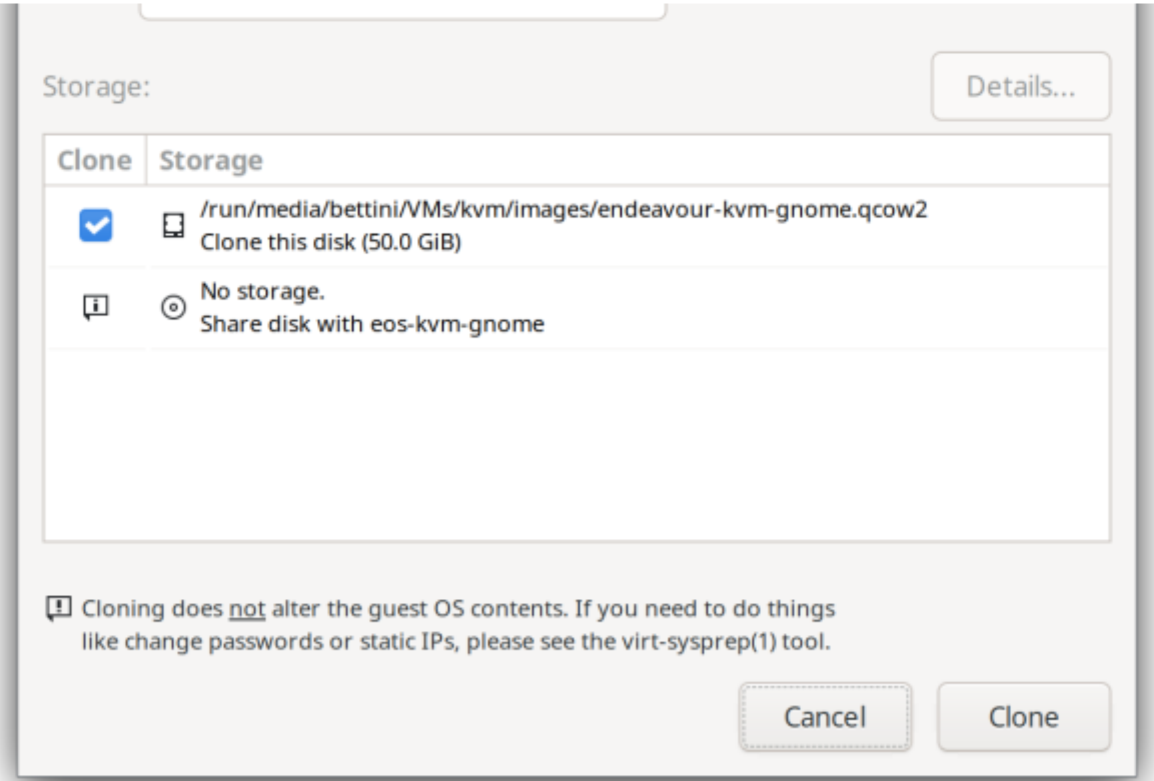
Cloning and Snapshots

Let’s create a clone of this virtual machine, e.g., with the context menu of the machine in the main user interface.



Original VM: eos-kvm-gnome

Name: eos-kvm-gnome-clone

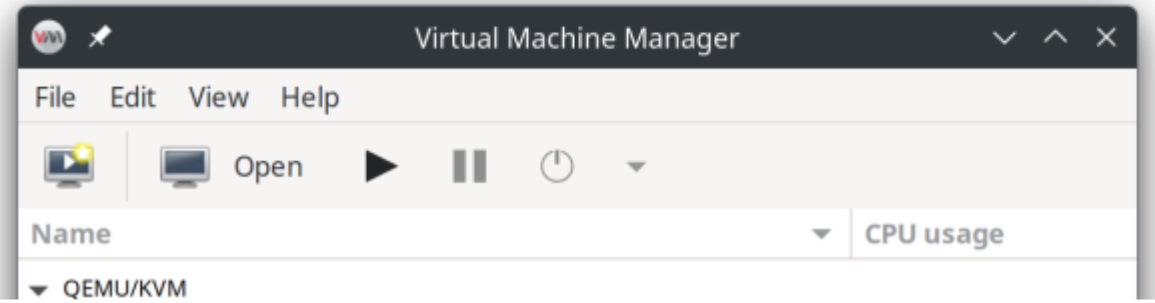

Storage:

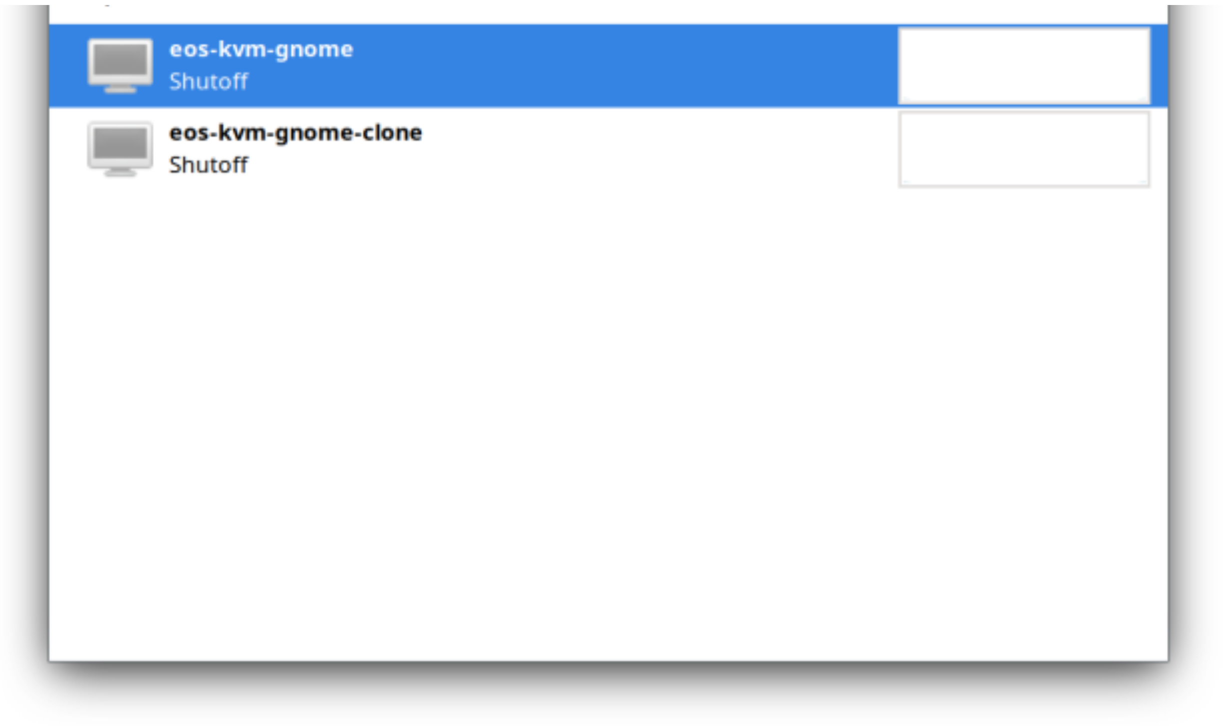
Clone	Storage
✓	/run/media/bettini/VMs/kvm/images/endeavour-kvm-gnome.qcow2 Clone this disk (50.0 GiB)
	No storage. Share disk with eos-kvm-gnome

Cloning does not alter the guest OS contents. If you need to do things like change passwords or static IPs, please see the virt-sysprep(1) tool.

The destination path is based on the path of the current machine, the external drive, which is good.

Let’s wait for the clone to finish, and then we have two virtual machines:

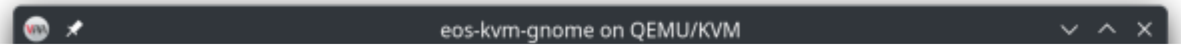


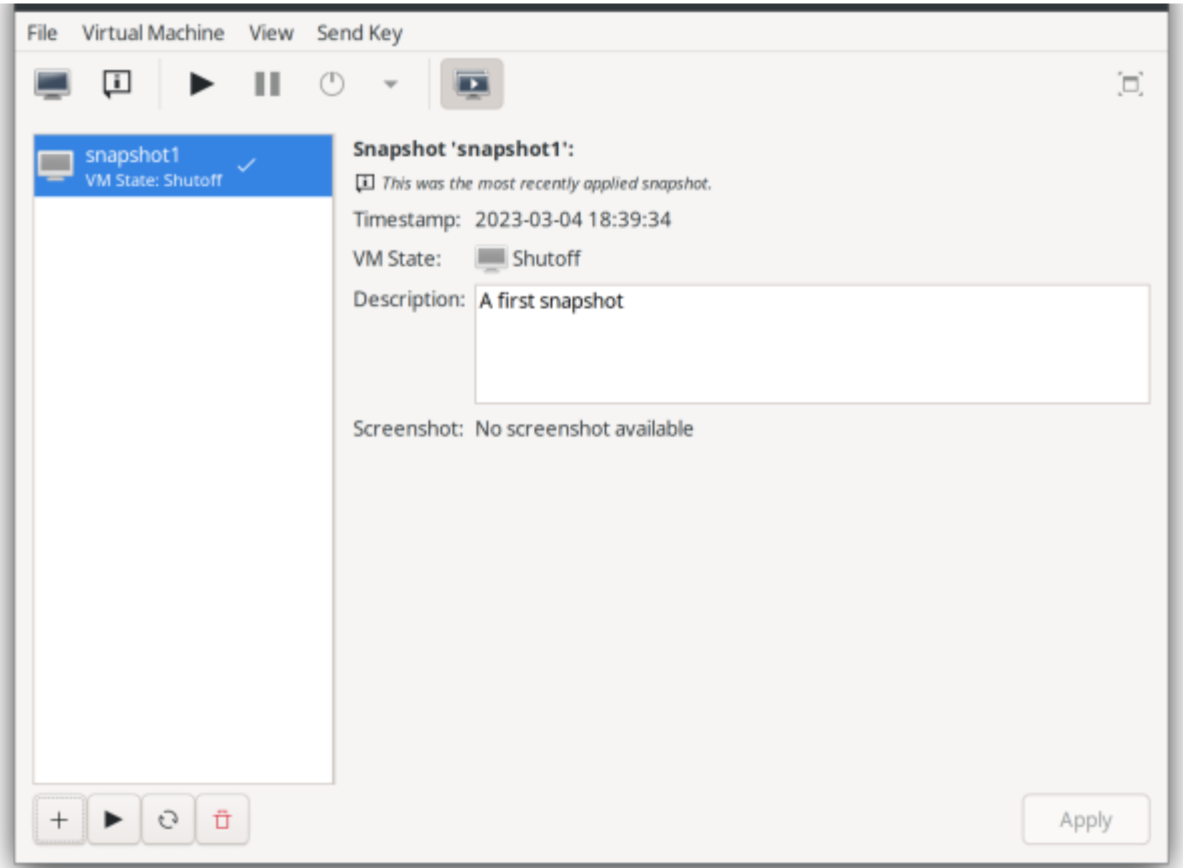


If I want this clone to be usable on other computers, I repeat the export procedure for this new virtual machine:

```
❯ sudo virsh list --all
Id   Name                  State
--------------------------------------
-    eos-kvm-gnome         shut off
-    eos-kvm-gnome-clone   shut off
❯ sudo virsh dumpxml eos-kvm-gnome-clone > /run/media/bettini/VMs/kvm/eos-kvm-gnome-clone.xml
```

I'll leave this clone virtual machine as it is for now, and I'll create a snapshot in the other virtual machine, the original one.





Snapshot 'snapshot1':

This was the most recently applied snapshot.

Timestamp: 2023-03-04 18:39:34

VM State: Shutoff

Description: A first snapshot

Screenshot: No screenshot available

Snapshot information is stored somewhere else, NOT in the XML of the virtual machine:

```
❯ ll /var/lib/libvirt/qemu/snapshot/eos-kvm-gnome
.rw------- 8,4k root  4 mar 18:39  snapshot1.xml
```

So we need them as well if we want to use them on another computer.

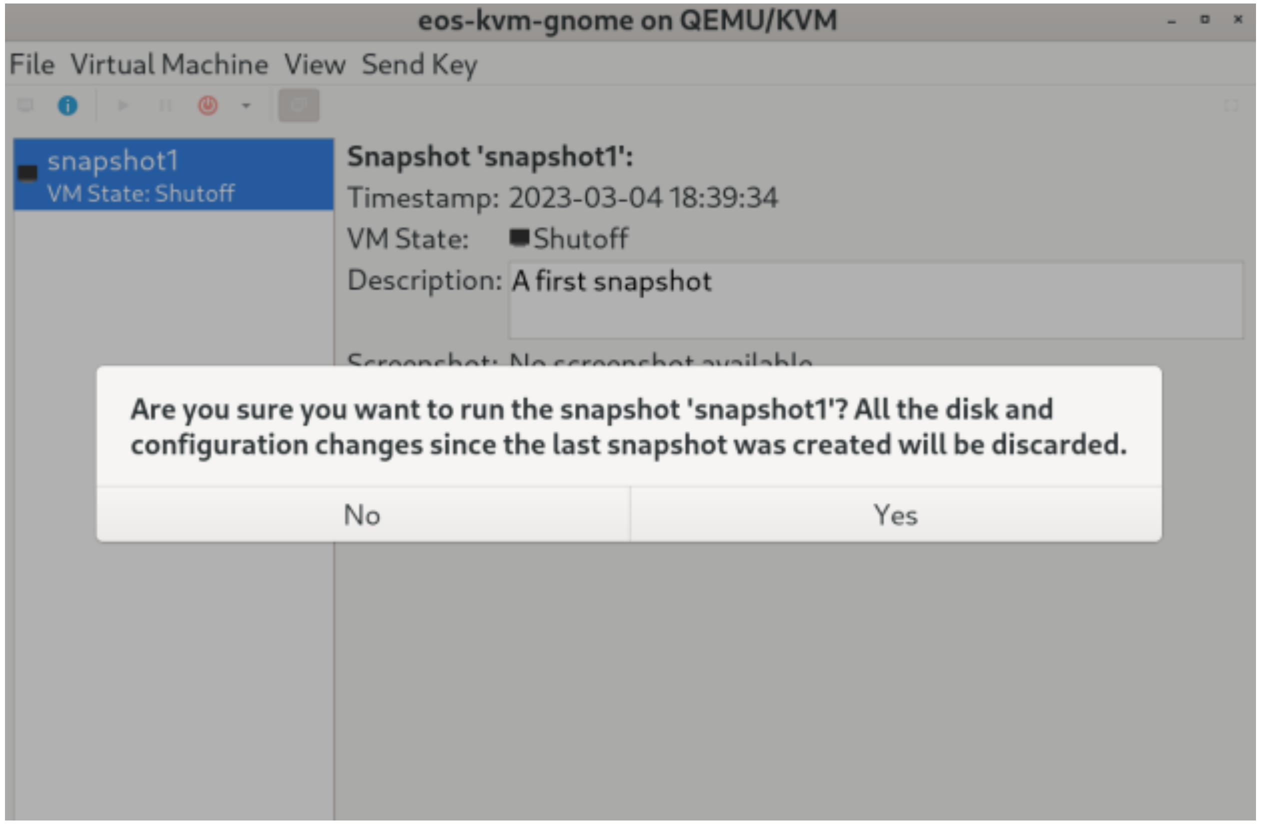
```
❯ sudo virsh snapshot-list --name eos-kvm-gnome
snapshot1
```

```
❯ sudo virsh snapshot-dumpxml --domain eos-kvm-gnome --snapshotname snapshot1 > /run/media/bettini/VMs/kvm/eos-kvm-gnome-snapshot1.xml
```

To add the snapshot to the other computer, I have to run:

```
❯ sudo virsh snapshot-create --redefine eos-kvm-gnome /run/media/bettini/VMs/kvm/eos-kvm-gnome-snapshot1.xml
Domain snapshot snapshot1 created from '/run/media/bettini/VMs/kvm/eos-kvm-gnome-snapshot1.xml'
❯ ll /var/lib/libvirt/qemu/snapshot/eos-kvm-gnome
.rw------- 8,4k root  4 mar 20:27 snapshot1.xml
```

However, keep in mind that if you try to start a snapshot, you get this warning:



Snapshot 'snapshot1':
Timestamp: 2023-03-04 18:39:34
VM State: Shutoff
Description: A first snapshot
Screenshot: No screenshot available

Are you sure you want to run the snapshot 'snapshot1'? All the disk and configuration changes since the last snapshot was created will be discarded.

So if you don't want to lose the current state, create another snapshot for the current state before restoring a previous one. Moreover, if the snapshot's state is "Shutoff", "starting" the snapshot only restores it. Then, you must start the virtual machine.

A different approach: symlinks

In the previous sections, I showed how machine information (including snapshots) and images could be put on external drives. Besides the machine images residing on external drives from the beginning, the machine metadata is still on your hard disk. In fact, you must first export them (e.g., on the external drive) and then import them on another computer.

A more radical approach consists of keeping the metadata on the external drive only and creating symlinks in each computer's libvirt/qemu directories.

On the first computer, the XML files of machine information and snapshots have to be copied onto the external drive. IMPORTANT: don't dump information as we did above; you need to copy the original XML files themselves. Dumping does not generate the exact XML files stored on the libvirt/qemu directories. In fact, as shown above, the dumped XML files must be imported with dedicated commands.

In my case, on the first computer, I run:

```
sudo cp -a /etc/libvirt/qemu /run/media/bettini/VMs/kvm/
sudo cp -a /var/lib/libvirt/qemu/snapshot /run/media/bettini/VMs/kvm/
```

So, on the external drive, I end up with these contents:

```
$ tree /run/media/bettini/VMs/kvm
/run/media/bettini/VMs/kvm
├── images
│   ├── endeavour-kvm-gnome-clone.qcow2
│   └── endeavour-kvm-gnome.qcow2
├── qemu
│   ├── autostart
│   ├── eos-kvm-gnome-clone.xml
│   ├── eos-kvm-gnome.xml
│   └── networks
│       ├── autostart
│       └── default.xml
└── snapshot
    └── eos-kvm-gnome
        └── snapshot1.xml
```

On the same computer, I run the following commands (make sure the "libvirtd.service" is not running):

```
sudo rm -rf /etc/libvirt/qemu
sudo ln -s /run/media/bettini/VMs/kvm/qemu /etc/libvirt
sudo rm -rf /var/lib/libvirt/qemu/snapshot
sudo mkdir -p /var/lib/libvirt/qemu
sudo ln -s /run/media/bettini/VMs/kvm/snapshot /var/lib/libvirt/qemu/
```

Now, I can start the "libvirtd.service" and the default network, and I make sure I can still access all my machines stored on the external drive, including all the machine information.

Of course, if you have never created virtual machines and want to start creating them on the external drive, it is enough to run the above commands. Then, start creating machines. Remember to select the external drive for the image location.

Then, on the other computers where I have already installed the same software for KVM, QEMU, etc., I first ensure the "libvirtd.service" is not running (in case stop it). Then, I connect my external drive and run the above commands (these will remove possible existing machines' information, so be careful).

Of course, the above commands must be run only the first time.

Now, I can start the "libvirtd.service" and the default network, and I can access all my machines stored on the external drive, including all the machine information. Every modification (an image content or a machine configuration) will be stored on the external drive.

This approach works if you want to store ALL your machines on the external drive. You won't have to keep the information in sync because they are stored in a single place.

If you need to keep some machines on your computers and others on different external drives, you must use the above-shown manual procedure for exporting and importing. It is then up to you to remember to re-export/re-import if you change a machine's configuration or a snapshot.

Happy virtualization!
🙂