How To Manage KVM Virtual Machines With Virt-Manager

1. What is Virt-Manager

**Virt-Manager** is a graphical user front end for the Libvirt library (is a toolkit to manage virtualization platforms ) which provides virtual machine management services. Virt-manager interface makes it easy for the user to create, delete and manipulate virtual machines without going through the terminal.

2. Install And Configure KVM In Ubuntu

2.1 What is KVM

Kernel-based Virtual Machine (KVM) is an open source virtualization technology built into Linux®. Specifically, KVM lets you turn Linux into a hypervisor that allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).

Using KVM, you can easily setup a virtualization environment in a Linux machine and host a wide range of guest operating systems including Linux, Windows, BSD, Mac OS and many.

Install Kvm and all required dependencies to setup a virtualization environment on your Ubuntu 20.04 LTS sever using command:

**$ sudo apt install qemu qemu-kvm libvirt-clients libvirt-daemon-system virtinst bridge-utils**

Once KVM is installed, start libvertd service (If it is not started already):

**$ sudo systemctl enable libvirtd**

**$ sudo systemctl start libvirtd**

Check the status of libvirtd service with command:

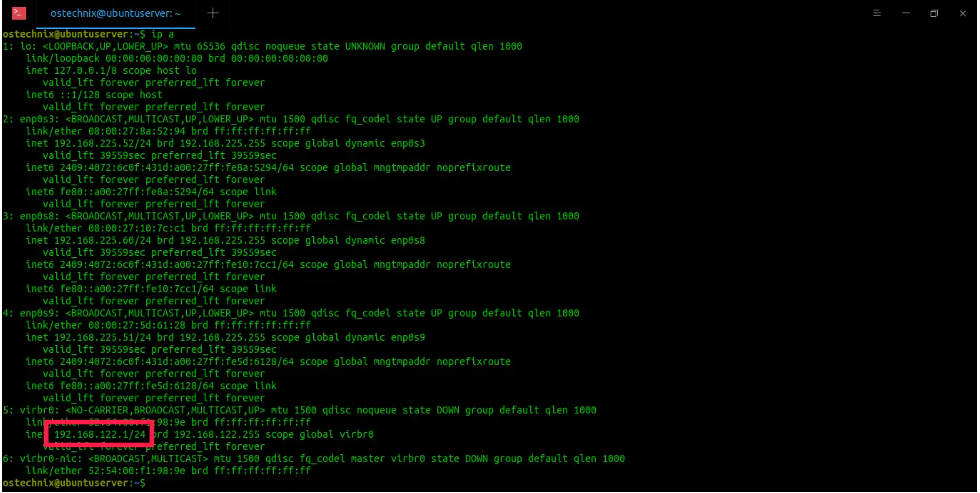
**$ systemctl status libvirtd**

2.2. Setup Bridge networking with KVM in Ubuntu

A bridged network shares the real network interface of the host computer with other VMs to connect to the outside network. Therefore each VM can bind directly to any available IPv4 or IPv6 addresses, just like a physical computer.

Have a look at the IP address of the KVM default virtual interfaces using "ip" command:

**$ ip a**



As you can see, KVM default network virbr0 uses 192.168.122.1/24 IP address. All the VMs will use an IP address in the 192.168.122.0/24 IP range and the host OS will be reachable at 192.168.122.1. You should be able to ssh into the host OS (at 192.168.122.1) from inside the guest OS and use scp to copy files back and forth

Before setting up a public bridged network, we should disable Netfilter for performance and security reasons. Netfilter is currently enabled on bridges by default.

To disable netfilter, create a file called **/etc/sysctl.d/bridge.conf**:

**$ sudo vi /etc/sysctl.d/bridge.conf**

Add the following lines:

**net.bridge.bridge-nf-call-ip6tables=0**

**net.bridge.bridge-nf-call-iptables=0**

**net.bridge.bridge-nf-call-arptables=0**

Save and close the file.

Then create another file called **/etc/udev/rules.d/99-bridge.rules** :

**$ sudo vi /etc/udev/rules.d/99-bridge.rules**

Add the following line:

**ACTION=="add", SUBSYSTEM=="module", KERNEL=="br\_netfilter", RUN+="/sbin/sysctl -p /etc/sysctl.d/bridge.conf"**

This will set the necessary flags to disable netfilter on bridges at the appropriate place in system start-up. Save and close the file. Reboot your system to take effect these changes.

Next, we should disable the default networking that KVM installed for itself.

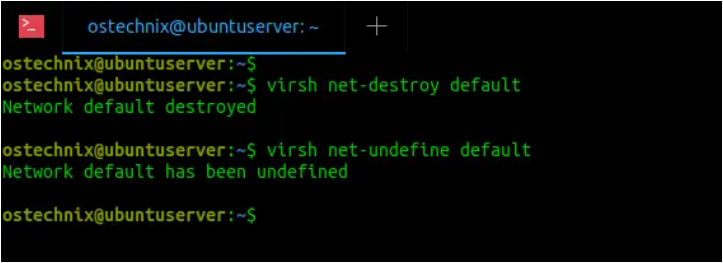
Find the name of KVM default network interfaces using "ip link" command:

**$ ip link**

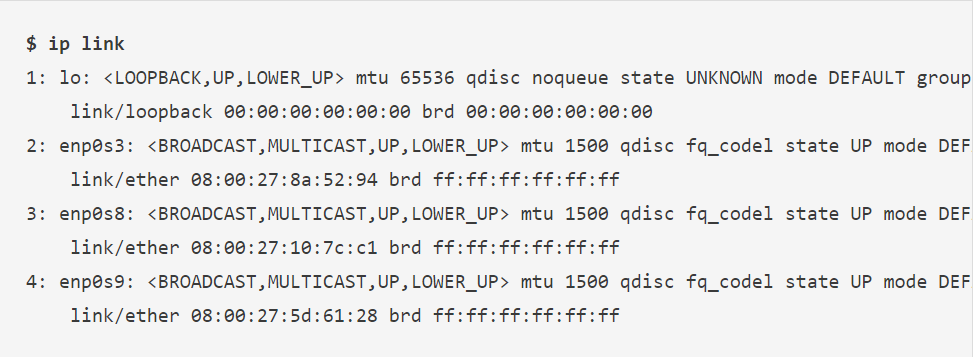


As you see in the above output, the entries "virbr0" and "virbr0-nic" are the KVM networks.

Let us remove the default KVM network with command:



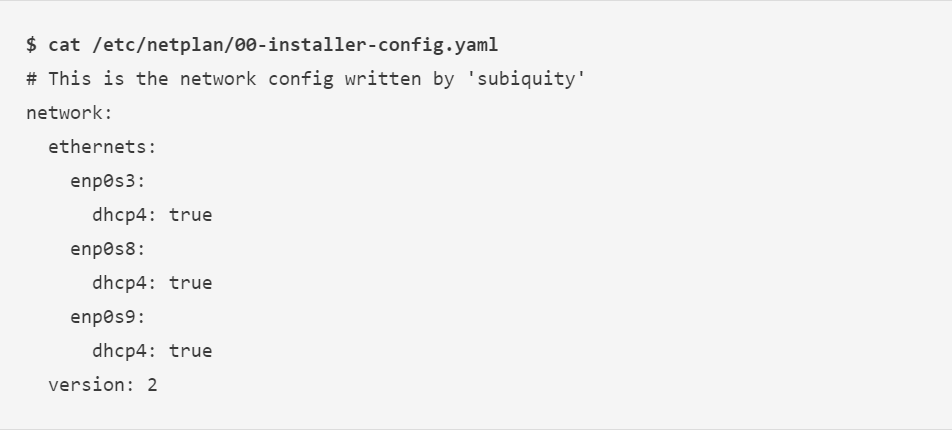
Now run "ip link" again to verify if the virbr0 and virbr0-nic interfaces are actually deleted:



Now, let us setup the KVM public bridge to use when creating a new VM.

To create a network bridge in host, edit **/etc/netplan/00-installer-config.yaml** file and add the bridge details.

Here is the default contents of the **00-installer-config.yaml** file in my Ubuntu 20.04 LTS server.



Then edit the default config file using your favorite editor:

**$ sudo vi /etc/netplan/00-installer-config.yaml**



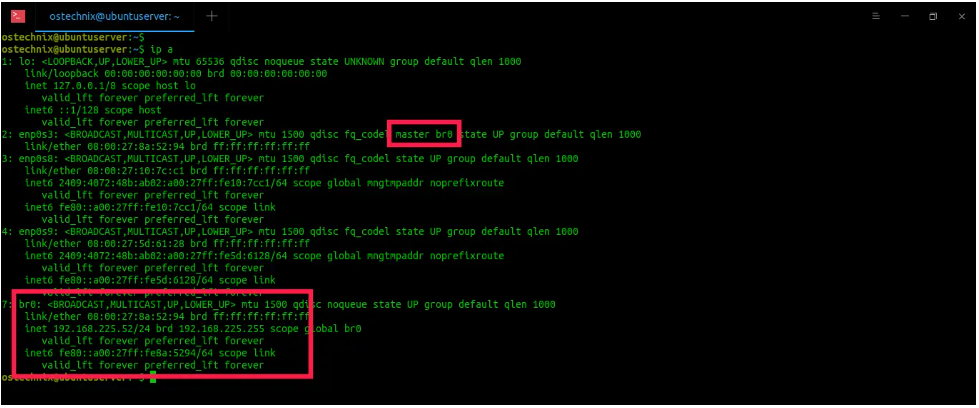
Here, the bridge network interface "br0" is attached to host's network interface "enp0s3". The ip address of br0 is 192.168.225.52. The gateway is 192.168.225.1. I use Google DNS servers (8.8.8.8 and 8.8.4.4) to connect to Internet. Make sure the space indentation are exactly same as above. If the line indentations are not correct, the bridged network interface will not activate. Replace the above values that matches with your network.

After modifying the network config file, save and close it. Apply the changes by running the following command:

**$ sudo netplan --debug apply**

Now check if the IP address has been assigned to the bridge interface:

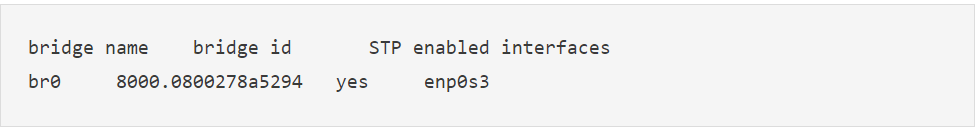
**$ ip a**



As you see in the above output, the bridged network interface br0 is assigned with IP address 192.168.225.52 and the enp0s3 entry now has "master br0" entry. It means that enp0s3 belongs to the bridge.

You can also use **"brctl"** command to show the bridge status:

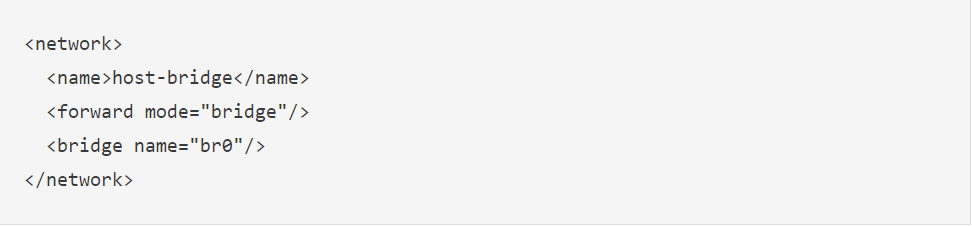
**$ brctl show br0**



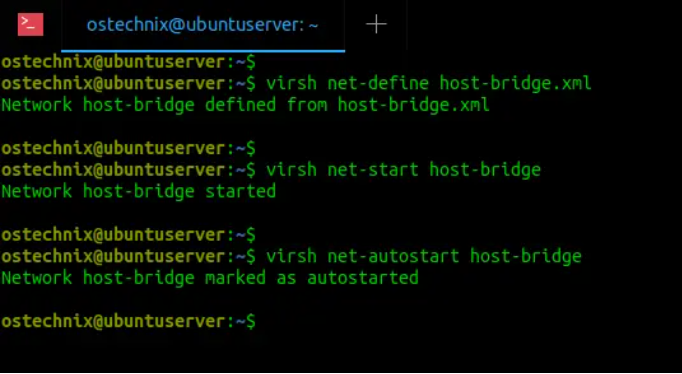
Now we should configure KVM to use this bridge. To do that, create a an XML file called host-bridge.xml :

**$ vi host-bridge.xml**

Add the following lines:



Run the following commands to start the newly created bridge and make it as default bridge for VMs:



To verify if the bridge is active and started, run:

**$ virsh net-list –all**



**Congratulations! We have successfully setup KVM bridge and it is active now.**

**3. Install Virt-Manager in Linux**

**$ sudo apt install virt-manager**

**3.1. Start and enable libvirtd service**

Make sure libvirtd service is enabled and running using command:

**$ sudo systemctl status libvirtd**

If it is not started, run the following commands to enable and start libvirtd service:

**$ sudo systemctl enable libvirtd**

**$ sudo systemctl start libvirtd**

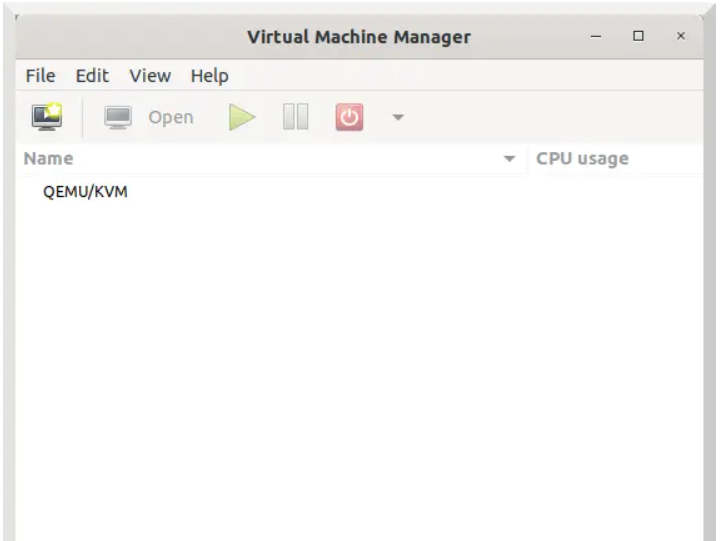
**3.2. Add user to libvirt group**

Every time you launch Virt-manager, you will be asked to enter the sudo password of your user. To avoid this, add your user to the libvirt group using command:

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

**4. Manage KVM Virtual Machines With Virt-Manager**

Once Virt-manager is installed, launch it from Dash or menu. The default interface of Virt-manager should look like below:



**4.1 Create a new virtual machine**

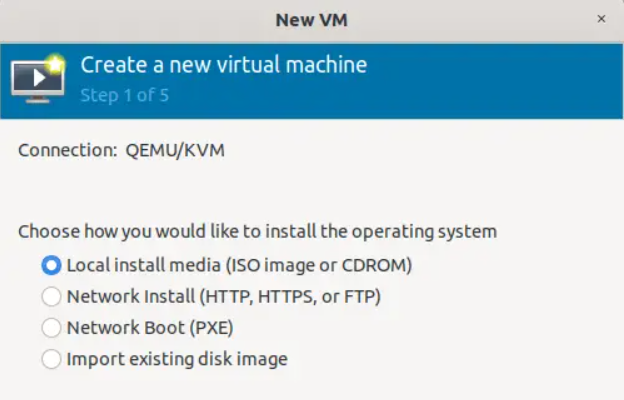
Go to File -> New Virtual Machine from Virt-manager main window. Choose how would you like to install the guest OS. You will be given four choices as listed below:

Local install media (E.g. ISO or CDROM),

Network install such as HTTP, HTTPS, or FTP,

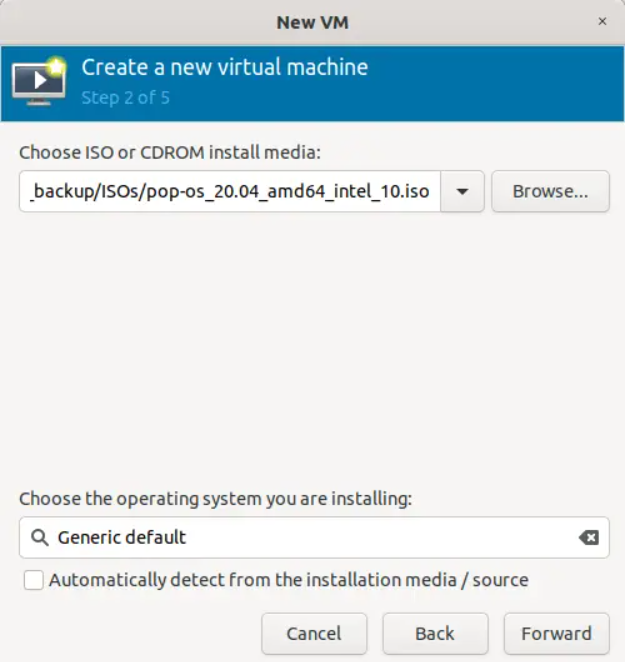
Network Boot (E.g. PXE),

I go with "Local install media" option. Click Forward to continue:

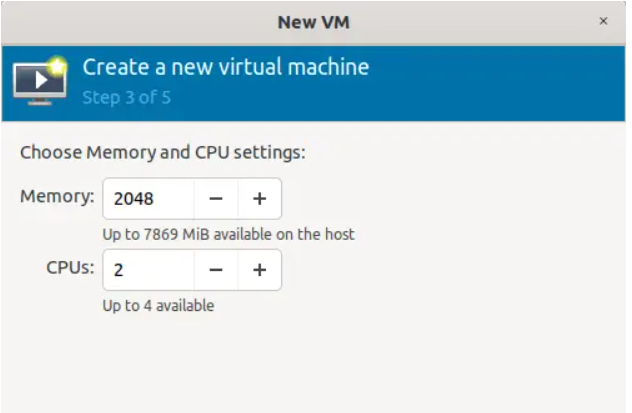


Choose the ISO file location of the OS. If you have the physical CD/DVD medium, choose it instead. You operating system type will automatically be selected depending on the installation medium you choose.

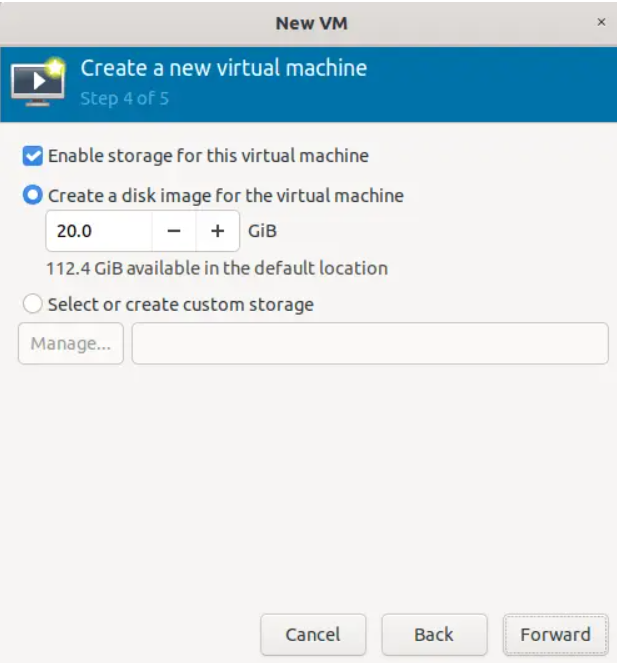
If Virt-manager can't detect the OS type, uncheck the option that says "Automatically detect from the installation media / source" and select "Generic default" as your OS type.



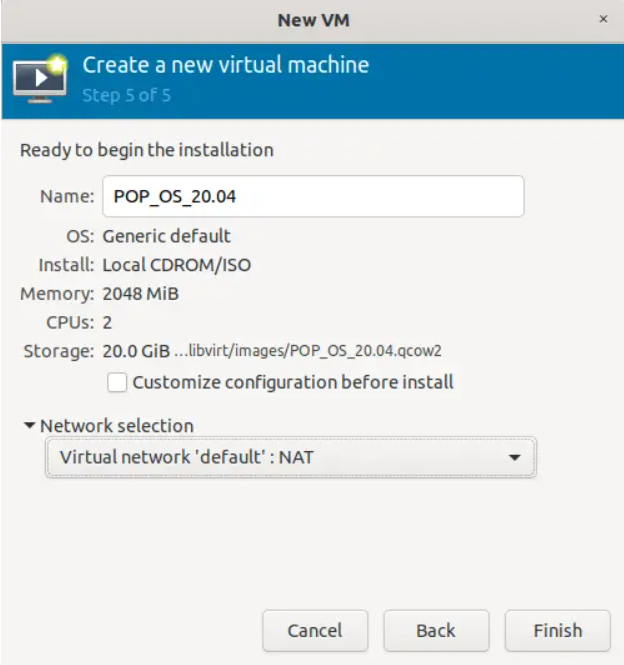
Choose memory size and number of cores for virtual CPU:



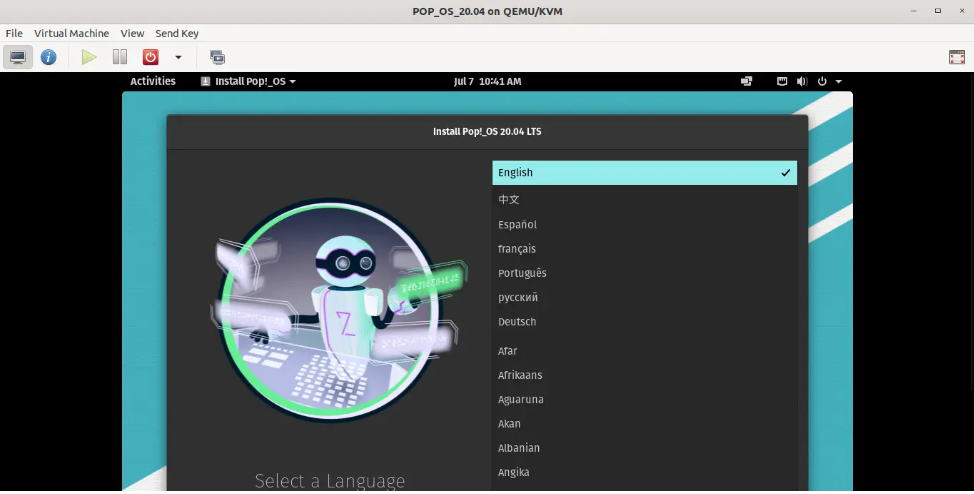
Choose the disk size for the Kvm guest:



Enter the name for your Kvm guest. The name should not contain any blank spaces. Click Finish to create the virtual machine.

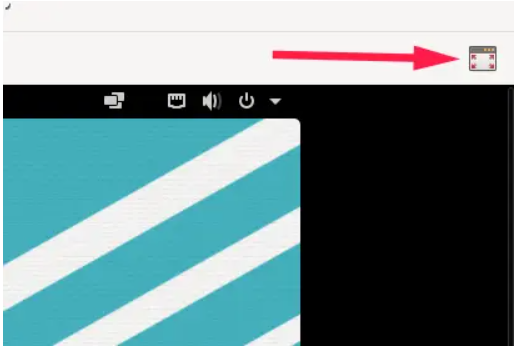


Once the Virtual machine is created, you will be automatically taken to the virtual machine's graphical console window.

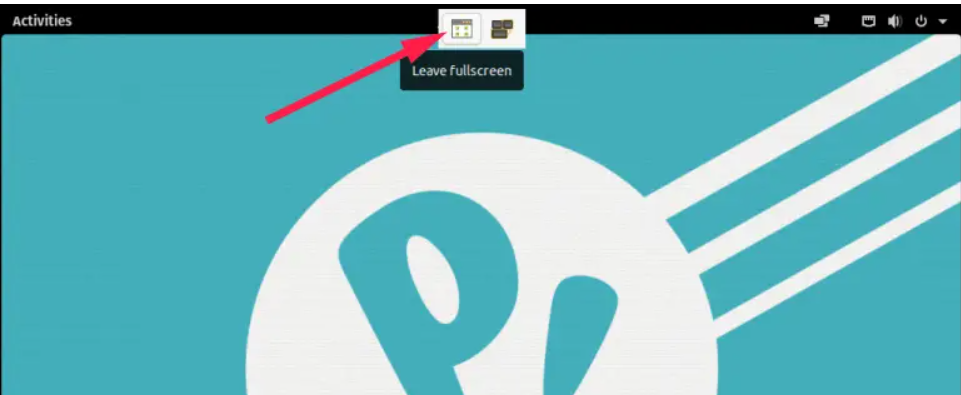


**4.2. Switch between normal mode and full screen screen mode**

By default, the guest will start in normal window mode. You can toggle to full screen mode by clicking on the "Switch to full screen view" button on the top right side of console window.

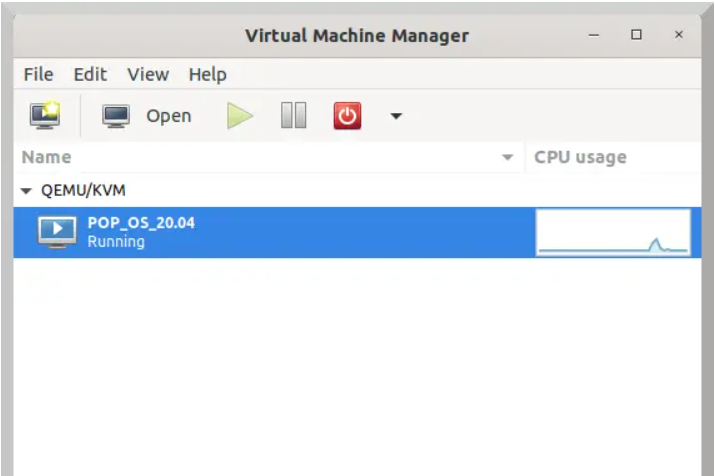


To go back to normal window, move the mouse cursor to the top middle side (over the date and time of VM) and you will see "Leave full screen" option, just click on it to switch to normal window.



**4.3. View all running kvm guests and their resource usage**

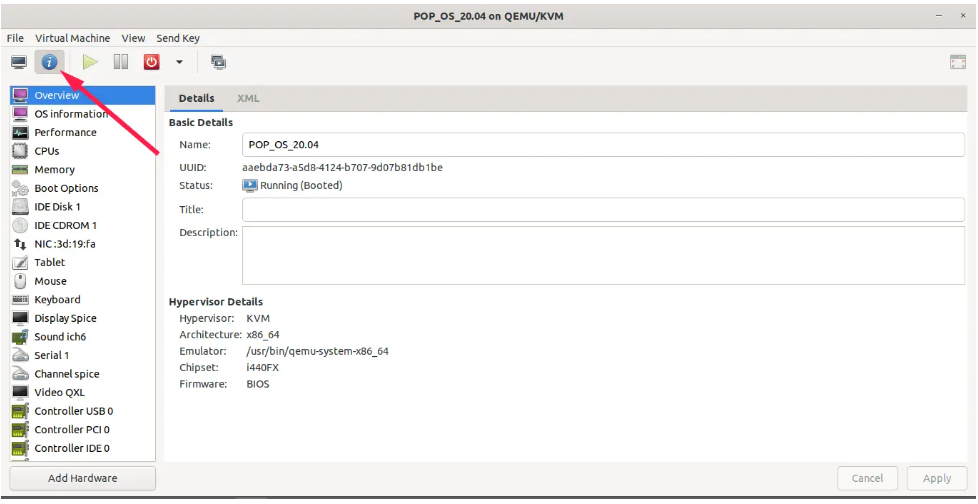
The Virt-manager main window shows all running kvm guests and resource utilization by the guests.



From here, you can start, pause, save the current state of a VM, restart and shutdown VMs.

**4.4. View Kvm virtual machine details**

The virtual hardware details window shows the information about all hardware resources configured to the kvm guest.



Some virsh command

1. Run "virt list" command to see the list of running VMs:

**$ sudo virsh list**

1. Run "virt list --all" command to see the list of running and nonrunning VMs:

**$ sudo virsh list --all**

1. To start a VM, run:

**$ sudo virsh start Ubuntu-18.04**

1. To restart a running VM, do:

**$ sudo virsh reboot Ubuntu-18.04**

1. To pause a running VM, do:

**$ sudo suspend Ubuntu-18.04**

1. To resume a suspended VM, do:

**$ sudo virsh resume Ubuntu-18.04**

1. To power off a running VM, do:

**$ sudo virsh shutdown Ubuntu-18.04**

1. To completely remove a VM, do:

**$ sudo virsh undefine Ubuntu-18.04**

1. **Virsh has a lots of commands and options. To learn all of them, refer virsh help section:**

**$ virsh --help**