KVM: Virtual Machines

[**References:**](#_64hr7vgqf28o) **1**

[**Introduction:**](#_u27vsk5ubtwa) **1**

[**Check That CPU Supports Virtualization:**](#_rscac3qmxsh0) **2**

[**Installation:**](#_kj17hx8obx15) **3**

[Installing the needed packages:](#_qoj1ebj7qp7e) 3

[Creating the Network Bridge:](#_8j9ou2thfrks) 3

[Modify ifcfg-enpXXX:](#_m6z8zo8da2p3) 6

[Create ifcfg-enpXXX:](#_rhvfg1ao6vno) 6

[Restart the network:](#_j7ovwsa79019) 6

[**Creating a virtual machine:**](#_r693vy3buy8b) **7**

[Virtual Machine Tools from GITHUB:](#_ws0eexwkbn1f) 7

[Description of Commands:](#_srxxugi58lpk) 7

[Procedure:](#_jwhbghngqmfd) 9

[Resizing the root (/) partition:](#_nkuwvonw0oso) 9

[Install the GITHUB VM tools package:](#_yx1jmkb5ziuu) 10

[Create Virtual Machine Kickstart file:](#_kcjlqk4uwhqy) 11

[Create the Virtual Machine:](#_ueagjd9gc3fs) 11

[The DHCP Kludge:](#_2xniwy6b1p1c) 11

[Running CreateCentOS7:](#_1jfeycswauo6) 12

[VirtClone:](#_147jqqxtgsxo) 13

[**virt-manager:**](#_5u3bwkpsevuq) **13**

[**Other GUI Virtual Machine Systems:**](#_t0yelrf5632e) **14**

# References:

<https://www.cyberciti.biz/faq/how-to-install-kvm-on-centos-7-rhel-7-headless-server/>

<https://wiki.archlinux.org/index.php/Network_bridge>

# Introduction:

KVM stands for**Kernel-Based Virtual Machine (VM)**, is a **virtualization software** which provides an ability to run a multiple **guest** operating systems with the help of **hardware virtualization extensions**. It supports a wide variety of guest operating system’s such as **Linux**, **Windows** and much more.

Some of the benefits of running guest operating systems:

* Create a **specific** **virtual** **machine** for a **specific purpose.**
* Store the **virtual** machines (a **disk image**) on reliable hardware.
* Easily move the **virtual machine** to **different hardware** in the event of a failure.
* Make **backups** and **snapshots** of a VM to run a previous operating system configuration.
* **Increase** **security** by **isolating** **services** on the VM from real hardware.
* **Rapidly** create new VMs for **testing** new software and/or operating system configurations.

|  |
| --- |
| Using **virtualization,** you run any operating supported the hardware. In other words, with a **CentOS 7** host you can run a **different** type of **Linux** or **Microsoft Windows**. |

|  |
| --- |
| We will use **virtual machines** on the **central server** to run **Cobbler** and **Puppet**. We may use it to run the HTCondor daemons.  **We will not use virtual machines on the individual workstations.** |

# Check That CPU Supports Virtualization:

**KVM** will work only if the CPU has the support of hardware virtualization, either **Intel VT** or **AMD-V**. To find whether your CPU supports **virtualization** features, run the following command.

|  |
| --- |
| grep -E '(vmx|svm)' /proc/cpuinfo | uniq |

If there is **no output**, then the hardware **does not support virtualization**. The issue may be that it is not **enabled** in the **BIOS** or that the **hardware** (CPU) does not support virtualization. If the system **does support** virtualization, the **output** will look **similar** to:

|  |
| --- |
| flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant\_tsc arch\_perfmon pebs bts rep\_good nopl xtopology nonstop\_tsc aperfmperf pni pclmulqdq dtes64 monitor ds\_cpl **vmx** smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4\_1 sse4\_2 popcnt aes lahf\_lm epb tpr\_shadow vnmi flexpriority ept vpid dtherm arat |

# Installation:

**KVM** requires the installation of a number of packages including the networking package **bridge-utils**. The package **bridge-utils** will enable us to configure the Ethernet card as a network bridge, and then assign unique Ethernet addresses and IP addresses to the Virtual Machines.

## Installing the needed packages:

The command below installs the needed software.

|  |
| --- |
| yum install qemu-kvm qemu qemu-img virt-manager libvirt libvirt-python \  ibvirt-client virt-install virt-viewer bridge-utils |

You then have to start and enable the **libvirt** daemon.

|  |
| --- |
| **libvirt** is an open-source API, daemon and management tool for managing platform virtualization. It can be used to manage KVM, Xen, VMware ESX, QEMU and other virtualization technologies. These APIs are widely used in the orchestration layer of hypervisors. |

|  |
| --- |
| systemctl start libvirtd systemctl enable libvirtd |

And now test to make sure that the **kvm kernel** module has been **loaded**:

|  |
| --- |
| [root@htc192 gdm]# lsmod | grep kvm  kvm 566340 0  irqbypass 13503 1 kvm |

## Creating the Network Bridge:

f you want to **assign IP addresses** to your **virtual machine**s and make them **accessible** from your **LAN** you need to **setup a network bridge**. A **bridge** is a piece of software used to **unite two or more** network **segments**. A bridge behaves like a **virtual network switch**, working transparently (the other machines do not need to know or care about its existence). Any **real devices** (e.g. **eth0**) and **virtual devices** (e.g. **br0**) can be connected to it.

By **configuring** a **network bridge** and then **configuring** the **virtual machine** to use it as a **network interface**, our **virtual machine** will look like any **other machine of the network**. It can run **services**, you **login** to it and it can **mount disks**. It can even be an HTCondor node although real hardware always outperforms virtual machines.

The remote access machines in Colorado are VMs:

|  |
| --- |
| pbsgold-laptop> ssh culogin03.colorado.edu  drjohn@culogin03.colorado.edu's password:  Last login: Wed May 2 12:52:48 2018 from 198.11.29.126  ===============================================================================  University of Colorado High Energy Physics  Unauthorized or improper use of this system is strictly forbidden.  Boulder, Colorado    culogin03.colorado.edu  QEMU Virtual CPU version (cpu64-rhel6) (cpus/cores: 1))  Operating system: Scientific Linux release 7.4 (Nitrogen)  Total RAM: 3.70 GiB Total swap: 2.00 GiB  ===============================================================================  culogin03> df -h  Filesystem Size Used Avail Use% Mounted on  /dev/vda3 48G 3.1G 45G 7% /  devtmpfs 1.9G 0 1.9G 0% /dev  tmpfs 1.9G 0 1.9G 0% /dev/shm  tmpfs 1.9G 145M 1.8G 8% /run  tmpfs 1.9G 0 1.9G 0% /sys/fs/cgroup  /dev/vda1 497M 176M 322M 36% /boot  hepusers:/amd/hepusers/nfs/hepusers 1.4T 45G 1.4T 4% /nfs/hepusers  tmpfs 380M 0 380M 0% /run/user/278  hepadmin01:/amd/hepadmin01/nfs/admin01 1.8T 595G 1.2T 34% /nfs/admin01  denali:/usr/local 1.8T 97G 1.7T 6% /misc/local |

Now that **bridge-utils** is installed, we can set up the network bridge. To do this we need to **modify two files** and then **restart** the **network**.

|  |
| --- |
| **WARNING**: It is highly **recommended** that you **do not** do this over a **network connection**. If you make a **mistake**, you may lose your network connection and be **locked out of the machine**. |

* The files we need to modify are located in **/etc/sysconfig/network-scripts**.
* **ifcfg-enpXXX** - The exact name of the file will depend on the name of the Ethernet interface. This can be different on every machine. In the directions I am using **enpXXX** as a placeholder for the device name. You can find you exact device by looking at:

|  |
| --- |
| ls -l /etc/sysconfig/network-scripts/ifcfg-e\* |

* **ifcfg-br0** - The bridge file. It does not currently exist on your machine

### Modify ifcfg-enpXXX:

Change **DEVICE=”enpXXX”** to **DEVICE=”br0”** and add the line **TYPE=”Bridge”**

|  |
| --- |
| cd /etc/sysconfig/network-scripts/  mkdir -p SaveConfigs  cp ifcfg-enpXXX SaveConfigs/  mv ifcfg-enpXXX ifcfg-br0 vi ifcfg-br0 |

**After the changes**, the file should like **similar** to, but **not exactly** like:

|  |
| --- |
| UUID="8ffa05c6-a97a-43b2-b3ca-4eec4a5ec1cb"  DNS1="172.16.0.1"  IPADDR="172.16.9.192"  GATEWAY="172.16.1.2"  NETMASK="255.255.240.0"  BOOTPROTO="static"  DEVICE="**br0**"  ONBOOT="yes"  IPV6INIT="no"  **TYPE="Bridge"** |

### Create ifcfg-enpXXX:

|  |
| --- |
| cd /etc/sysconfig/network-scripts/  vi ifcfg-enpXXX |

The contents of the new file should be very similar to:

|  |
| --- |
| TYPE=Ethernet  BOOTPROTO=static  DEVICE=**enpXXX**  ONBOOT=yes  **BRIDGE=br0** |

### Restart the network:

|  |
| --- |
| systemctl restart network |

If everything is correct, you should still have network connectivity but now have a network bridge. You can use ifconfig to display the configured network devices:

|  |
| --- |
| ifconfig |

With part of the output similar to:

|  |
| --- |
| **br0** Link encap:Ethernet HWaddr A0:36:9F:59:AE:E0  **inet addr:128.138.133.25** Bcast:128.138.133.255 Mask:255.255.255.0  inet6 addr: fe80::a236:9fff:fe59:aee0/64 Scope:Link  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  RX packets:1735500142 errors:0 dropped:0 overruns:0 frame:0  TX packets:1774113760 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:0  RX bytes:1250311258541 (1.1 TiB) TX bytes:799226149273 (744.3 GiB)  **eth0**  Link encap:Ethernet HWaddr A0:36:9F:59:AE:E0  inet6 addr: fe80::a236:9fff:fe59:aee0/64 Scope:Link  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  RX packets:2256831911 errors:0 dropped:16 overruns:0 frame:0  TX packets:2079208093 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:1000  RX bytes:1318106934007 (1.1 TiB) TX bytes:819372158998 (763.0 GiB) |

You should be able to ping other hosts and the web browser on the machine should still function properly.

# Creating a virtual machine:

libvirt comes with a number of tools to help manage virtual machines. I have packaged a few of them in GIT repository. During this exercise, you will still have to do a little manual intervention.

## Virtual Machine Tools from GITHUB:

The files are located on GITHUB at:

|  |
| --- |
| [**https://github.com/taktse/Virtual-Machines**](https://github.com/taktse/Virtual-Machines) |

## 

## Description of Commands:

The two tools you can use are:

* **bin/CreateCentOS7VMKickstart** - Create a kickstart file for the Virtual Machine. It will be created in the subdirectory kickstarts/, but then you will need to copy it to:

|  |
| --- |
| /nfs/htc180/pub/ks |

* **bin/CreateCentOS7** - The runs the command **virt-install**. It expects:
  + **Boot ISO /mnt/boot.iso -** You can find a copy of this image in:

|  |
| --- |
| /nfs/htc180/pub/centos/7/os/x86\_64/images/boot.iso |

* + **Kickstart file**: This kickstart file is the created using **CreateCentOS7VMKickstart.** After it is generated, you will put a copy on the class server in /nfs/htc180/pub/ks. It can then be accessed through the class web server at**:**

|  |
| --- |
| http://172.16.9.180/software/ks/${hostname}-ks.cfg |

* + **Bridge device: br0** - This is the bridge device
  + **Number CPUs: 1** - Number of CPUs for this VM
  + **Memory: 2048MB** - Memory for this VM
  + **VM storage directory**: /var/lib/libvirt/images - The VM images are stored.

|  |
| --- |
| The above **parameters** can be changed by **exporting variables**. For example:  **export ISO=/nfs/htc180/pub/centos/7/os/x86\_64/images/netboot.iso**  **Changes** the initial **boot device**. |

|  |
| --- |
| **Please do not just use the scripts. Please read them and try to understand what they do. You will need this to create VMs for the class server when you are configuring Cobbler, Puppet, Condor and a web server for the class.** |

## Procedure:

Before you begin installing a VM on the class nodes, you will have to modify the partitions on the disk. We installed the initial operating system using the default partition layout. This layout allocates 50GB for the root partition and the remainder goes to /home. There will not be enough room to install a VM on the / (root) partition so we want to resize the partitions. We can do this (Inshallah) with minimal disruption to the operating system.

|  |
| --- |
| **Please be careful. It is possible to loose files and/or corrupt the entire system when resizing the disk.** |

|  |
| --- |
| You don’t have to worry too much as this is the last time we will use these hosts before reinstalling all the of the operating systems. They are now disposable machines. |

### Resizing the root (/) partition:

Since there is currently **no free** **space** on the **disk**, we **first** have to **decrease** the size of /**home**. It is **not possible** to **nondestructively** **decrease** the size of a **partition** so we **first need to backup /home. For safety, make two copies:**

|  |
| --- |
| cd /home  tar -czvf /tmp/home.tar.gz .  tar -czvf /nfs/htc-data/`hostname -s`/home.tar.gz . |

|  |
| --- |
| **Notice the use of `hostname -s` in the path name.** |

Now we **umount /home, remove** the **/home logical volume**, **create** a **new home logical volume, format it** with the **xfs** file system and then **remount** it.

**(That is a lot to do. Please be careful about what you are doing.)**

|  |
| --- |
| umount /home  lvremove /dev/centos\_htcXXX/home  lvcreate -L 20GB -n home centos\_htcXXX  mkfs.xfs /dev/centos\_htcXXX/home  mount -a |

If you are daring, you could try:

|  |
| --- |
| export HOSTNAME=`hostname -s`  umount /home  lvremove /dev/centos\_${HOSTNAME}/home  lvcreate -L 20GB -n home centos\_${HOSTNAME}  mkfs.xfs /dev/centos\_${HOSTNAME}/home  mount -a |

Now restore the backed up files to /home and extend the root partition:

|  |
| --- |
| cd /home  tar -xzvf /tmp/home.tar.gz  lvextend -r -l +100%FREE /dev/mapper/centos\_htc188-root |

Again if you are daring, you can use:

|  |
| --- |
| export HOSTNAME=`hostname -s`  cd /home  tar -xzvf /tmp/home.tar.gz  lvextend -r -l +100%FREE /dev/mapper/centos\_${HOSTNAME}-root |

If **everything** went well, then /**home** is now **20GB** and **/** is **>800GB**. You can find out the size of the partitions using the df command:

|  |
| --- |
| df -h |

### Install the GITHUB VM tools package:

|  |
| --- |
| cd  git clone https://github.com/taktse/Virtual-Machines.git  cd Virtual-Machines |

## 

### Create Virtual Machine Kickstart file:

To gain experience in creating **VMs**, we will use the unused IP addresses assigned to the class: **htc193** through **htc200**. Please choose the appropriate hostname from the list below:

|  |  |
| --- | --- |
| Cobbler/Puppet Server | Test VM Hostname |
| htc181 | htc193 |
| htc185 | htc194 |
| htc186 | htc195 |
| htc189 | htc196 |
| htc191 | htc197 |

To generate a kickstart that is specific for a virtual machine, use:

|  |
| --- |
| bin/CreateCentOS7VMKickstart htcXXX |

Now place that kickstart on the class server. It will be served to the VM during installation from the web server.

|  |
| --- |
| cd kickstarts  sftp root@htc180  > cd /nfs/htc180/pub/ks  > put htcXXX-ks.cfg |

Now put a boot image in /tmp:

|  |
| --- |
| cp /nfs/htc180/pub/centos/7/os/x86\_64/images/boot.iso /tmp/boot.iso |

After copying **boot.iso** to **/tmp** and your **kickstart file** (htcXXX-ks.cfg) to **/nfs/htc180/pub/ks**, you can create a VM using **CreateCentOS7**.

### Create the Virtual Machine:

#### The DHCP Kludge:

When **creating** a virtual machine, it **needs** a **network connection** in order to download the **kickstart** file. This means that it needs a **DHCP** server. This presents us with a problem. We **can not run** an open **DHCP** server since we are on a **campus** **network**. We know **this is a bad thing** to do because we tried it last week and **got in trouble** with the **campus networking staff.** We would **prefer** to have our **own** network and more **preferably** our own **private network**. ***Inshallah***, we will have an additional network for the class server **in about 2 weeks**. To “**kludge**” this problem will:

* Start a VM installation
* Disconnect the room from the campus network
* Start a local DHCP server
* Wait until the installation starts
* Turn off the DHCP server
* Reconnect the room to the campus network.

#### Running CreateCentOS7:

When you are ready to install the VM, let me know and I will help.

|  |
| --- |
| /root/Virtual-Machine/bin/CreateCentOS7 htcXXX |

|  |
| --- |
| **Please note:** The current **kludge** will **not work** from a **remote machine**. The **installation** will **work remotely** when we have a **private network.** |

|  |
| --- |
| **Notice that the command to exit the Virtual Machine console is ^]**  **Escape character is ^]** |

After the VM **installation** **finishes**, you can **exit** the VM **console** using **^].** You can **interact** with **VMs** using the command **virsh**.

Useful **virsh** (virtual machine) commands:

|  |
| --- |
| * **virsh list** - List all the running VMs on a machine * **virsh list --all** - List all VMs (running and shutdown) on a machine. * **virsh autostart VMNAME** - Start the VM on reboot/. * **virsh list --autostart** - List the VMs that will start on reboot. * **virsh console VMNAME** - Connect a console to the VM * **^]** - Characters to exit VM console * **virsh destroy htcXXX** - Shutdown a VM. **This does not delete the VM!** * **virsh undefine htcXXX** - Remove definition of the VM. It will not longer be listed using virsh list. * **rm /var/lib/libvirt/images/htc196.qcow2** - **Permanently** remove a VM from a machine. **Use with caution.** |

## VirtClone:

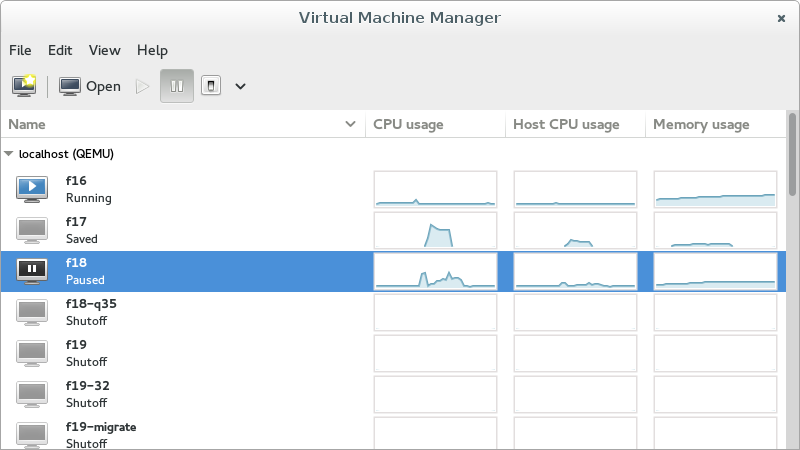
This is an **additional command** included with the GITHUB package **Virtual-Machines**. It will create a **clone** of a **VM** using the command **virt-clone**. If the VM is **running**, it will **suspend** it during the cloning and then **unsuspend** it when the clone is finished. Making a **clone** is much **faster** than **creating** a **VM** although it requires you boot the VM and then **fix the network configuration and hostname**.

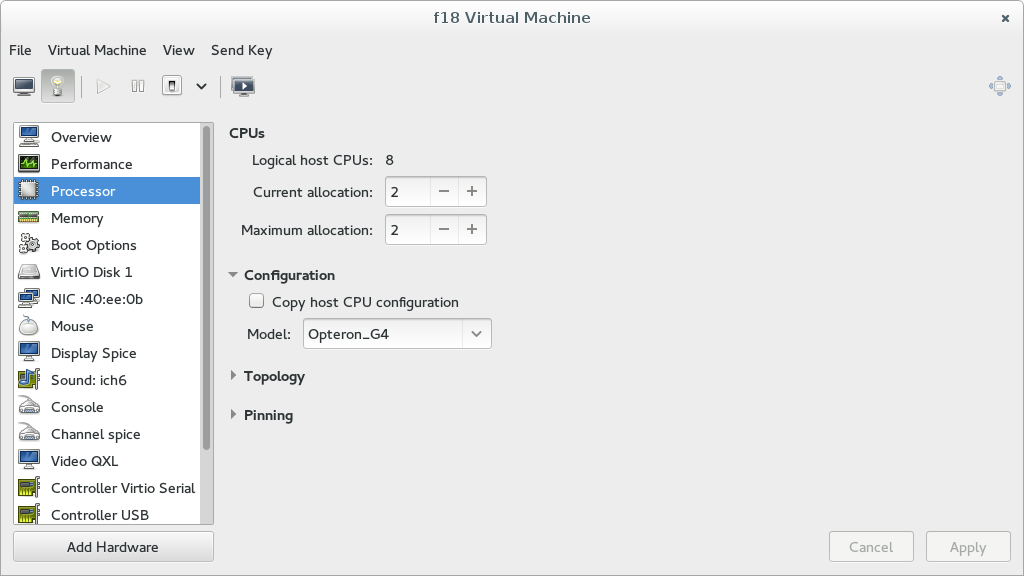
Usage:

|  |
| --- |
| VirtClone OldVM NewVM  For example:  VirtClone htcXXX puppet-server |

# virt-manager:

The **virt-manager** application is a desktop user interface for managing virtual machines through libvirt. It primarily targets **KVM** VMs, but also manages **Xen** and **LXC** (linux containers). It presents a summary view of running domains, their live performance & resource utilization statistics. Wizards enable the creation of new domains, and configuration & adjustment of a domain’s resource allocation & virtual hardware. An embedded VNC and SPICE client viewer presents a full graphical console to the guest domain.





# Other GUI Virtual Machine Systems:

A few other open source virtual machine, cloud infrastructure software packages:

* [**OpenNebula**](http://www.opennebula.org/) is a cloud computing platform for managing heterogeneous distributed data center infrastructures. The OpenNebula platform manages a data center's virtual infrastructure to build private, public and hybrid implementations of infrastructure as a service. The two primary uses of the OpenNebula platform are data center virtualization solutions and cloud infrastructure solutions.
* [**oVirt**](http://www.ovirt.org/) is free, open-source virtualization management platform. It was founded by Red Hat as a community project on which Red Hat Enterprise Virtualization is based. It allows centralized management of virtual machines, compute, storage and networking resources, from an easy-to-use web-based front-end with platform independent access.
* [**OpenStack**](http://openstack.org/) is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service (IaaS), whereby virtual servers and other resources are made available to customers.The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center.