## Installing and Using KVM

Document Author: arho.virkki@vtt.fi

## Cent OS 7 Setup

### Install the KVM environment

Access the remote machine with `ssh -X` and

```
ssh -X abscissa.vtt.fi
```

and gain the root privileges

```
su -
yum groupinstall "Virtualization Hypervisor"
yum groupinstall "Virtualization Client"
yum groupinstall "Virtualization Platform"
yum groupinstall "Virtualization Tools"
```

### Manage guest operating systems with *virt-manager*

Log in into the system

```
ssh -X abscissa.vtt.fi
su -
```

and use the existing X Window system (on Ubuntu Linux, Mac OS X). The other option is to set up remote access as in Virtualization Deployment and Administration_Guide:

```
ssh-keygen -t rsa
ssh-copy-id -i .ssh/id_rsa.pub root@abscissa.vtt.fi
```

Then start *libvirt* daemon

```
ssh root@abscissa.vtt.fi
systemctl enable libvirtd.service
systemctl start libvirtd
```

Now launch

```
virt-manager
```

All details can be configured graphically, including *bridged networking*, which is most conveniently done through MacVTap. Manual editing of the the */etc/network/interfaces* configuration file is not necessary.

### Configure a software bridge

There are multiple different ways to configure network in CentOS 7. For details, see: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/.

### Short terminology

**Network bonding**: To bind multiple network interfaces together into a single, bonded, channel. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

**Network teaming**: Same as network bonding, but different (newer) implementation.

**Network bridge** (we need this): A network bridge is a link-layer device which forwards traffic between networks based on MAC addresses. It makes forwarding decisions based on a table of MAC addresses which it builds by listening to network traffic and thereby learning what hosts are connected to each network. A software bridge can be used within a Linux host in order to emulate a hardware bridge, for example in virtualization applications for sharing a NIC with one or more virtual NICs.

The graphical tool is *nm-connection-editor*, whereas the text-based user interface can be launched with

```
nmtui
```

The status of the network can be inspected with

```
systemctl status network
```

To create a virtual (software) bridge

1. Delete or inactivate (+disable auto-start) the physical device (e.g. enp11s0)
2. Create the bridge (e.g. bridge0)
3. Bind the physical device (e.g. enp11s0) to the bridge as slave
4. Ensure that the bridge starts automatically on server boot.

References: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Virtualization_Deployment_and_Administration_Guide/sect-Installing_the_virtualization_packages-Installing_virtualization_packages_on_an_existing_Red_Hat_Enterprise_Linux_system.html

## Ubuntu 14.04 Setup

### Prerequisites

First, check the compatibility

```
sudo apt-get install cpu-checker
kvm-ok
```

Install the virtualization packages

```
sudo apt-get install qemu-kvm libvirt-bin
sudo adduser $USER libvirtd
```

and log out to enable the new group settings.

### Configure a Virtual Bridge

To make the virtual machines available to outside network, we need to configure a virtual bridge (i.e. a network switch or a smart hub) to connect these machines. Install bridge-utils (if not already installed)

```
sudo apt-get install bridge-utils
```

And modify the network interface settings in */etc/network/interfaces* according to the example:

```
auto br0
iface br0 inet static
        address 192.168.1.220
        netmask 255.255.255.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
        bridge_ports eth0
        bridge_stp on
        bridge_maxwait 0
```

```
        dns-nameservers 192.168.1.1
        dns-search vtt.fi
```

We simply changed *eth0* into *br0*, and added some lines to control the bridge behaviour (*bridge_ports*, *bridge_stp*,*bridge_maxwait*). To activate the setup, the simplest thing is to reboot the machine.

Once up again, issue

```
brctl show
```

To see the activated bridges. To drop the default bridge that came along with the packages, issue

```
sudo ip link set dev virbr0 down
sudo brctl delbr virbr0
```

Now we are set with a very clean setup.

## Installation of virtual machines

Now we are ready to install and manage virtual machines. There are several options to view, install and manage virtual machines explained in Ubuntu Server Documentation https://help.ubuntu.com/lts/serverguide/libvirt.html.

### Graphical Method (*virt-manager*)

The *virt-manager* is developed (mainly) for Linux-based workstations and can be installed with

```
sudo apt-get install virt-manager qemu-system
```

To connect local local and remote hosts, issue, for example

```
virt-manager -c qemu:///system
virt-manager -c qemu+ssh://abscissa.vtt.fi/system
```

The Windows versions can be found at: http://www.spice-space.org/download.html. New connections can also be made graphically from the UI.s

A new virtual machine can be createdby clicking the top left display icon in the GUI. When connected to remote host, the installation media should be copied under `/var/lib/libvirt/images/` to make it available for the virt-manager.

During the process, the wizard will create an XML file describing the VM's settings under */etc/libvirt/qemu/*, and a disk image under */var/lib/libvirt/images/*.

The virtual machines can be viewed and used with *virt-manager*, or alternatively with *virt-viewer* which connects directly to the virtual machine. The following example connects to machine *RemoteSrv1*. Graphical Method (virt-manager)

```
virt-viewer --connect qemu+ssh://abscissa.vtt.fi/system  RemoteSrv1
```

### Command-line Method

There are multiple options to systematize virtual machine installation with different prepackaged softare bundled. For more information, see: https://help.ubuntu.com/lts/serverguide/virtualization.html

## Management of virtual machines

### Graphical Method (*virt-manager*)

Choose File -> Add Connection -> QEMU/KVM and fill in the connection details. Administration interface resembles VMWare Workstation and Virtualbox, and everything can be tuned graphically.

### Command-line Method (*virsh*)

For details, see https://www.centos.org/docs/5/html/5.2/Virtualization/chap-Virtualization-Managing_guests_with_virsh.html

The *virsh* shell can be started locally with no options or by giving the connection uri explicitly:

```
virsh -c qemu:///system
virsh -c qemu+ssh://abscissa.vtt.fi/system
```

The virsh commands are given as the second argument, e.g.

```
virsh -c qemu+ssh://abscissa.vtt.fi/system list
```

To avoid always writing the connection uri, the default connection can also be set as environment variable

```
export VIRSH_DEFAULT_CONNECT_URI="qemu+ssh://abscissa.vtt.fi/system"
```

The virtual machine description can be saved into an xml which can be used te recreate the guest later.

```
virsh dumpxml RemoteBox1 > RemoteBox1.xml
```

Example command (run on the server)

```
virsh list --all
virsh suspend Win7
virsh resume Win7
virsh save Win7 Win7SnapShot
virsh restore Win7SnapShot
virsh reboot Win7
virsh shutdown Win7
```

## Copying virtual machines between hosts

http://ostolc.org/kvm-move-guest-to-another-host.html

On the old machine

```
virsh shutdown Win7
virsh dumpxml Win7 > /tmp/Win7.xml
less /tmp/Win7.xml
cp /tmp/Win7.xml arho@192.168.1.200:/tmp/
rsync -e ssh -avut /var/lib/libvirt/images/Win7.img arho@192.168.1.200:v
```

On the new machine

```
sudo su
mv /tmp/Win7.img /var/lib/libvirt/images/
chown libvirt-qemu:kvm /var/lib/libvirt/images/Win7.img
chmod 600  /var/lib/libvirt/images/Win7.img
virsh define /tmp/Win7.xml
virsh start Win7
```

Once we have tested that the machine works in the new host, we can delete it from the old machine.

```
virsh undefine Win7
virsh list --all
```

### Editing the network settings

The xml file defines the virtual machine settings. For example, if the new host does not support bridged networking, we simply change the *bridge* entry in the xlm from

```
<interface type='bridge'>
  <mac address='52:54:00:f4:6e:7c'/>
  <source bridge='br0'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

into

```
<interface type='network'>
  <mac address='52:54:00:f4:6e:7c'/>
  <source network='default'/>
  <model type='rtl8139'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

References:

https://help.ubuntu.com/lts/serverguide/virtualization.html http://wiki.libvirt.org/page/Networking#Debian.2FUbuntu_Bridging https://help.ubuntu.com/community/KVM http://wiki.libvirt.org/page/Main_Page https://www.centos.org/docs/5/html/5.2/Virtualization/chap-Virtualization-Managing_guests_with_virsh.html https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/chap-Virtualization-Managing_guests_with_virsh.html http://www.tuxradar.com/content/howto-linux-and-windows-virtualization-kvm-and-qemu https://help.ubuntu.com/lts/serverguide/network-configuration.html#name-resolution

## KVM Advanced Usage

### File systems utilities

Install some extra tools:

```
sudo yum install libguestfs-tools virt-top
```

Now the following commands do what is expected…

```
virt-ls -d ktptest -l /
virt-cat -d ktptest /etc/passwd
virt-edit -d ktptest /etc/fstab # the machine must be shut off
virt-df -d ktphadoop -h
virt-top
```

See: http://www.server-world.info/en/note?os=CentOS_7&p=kvm&f=9

### Snapshots

KVM can make and restore snapshots from running virtual machines. At the time of this writing, this works only / has been tested best with qcow2 disk images.

### Snapshots with *virt-manager* (GUI)

Recent *virt-manager* (>1.0 default in CentOS 7 but not in Ubuntu 14.04) includes a button for creating and restoring snapshots. For details, see http://blog.wikichoon.com/2014/03/snapshot-support-in-virt-manager.html.

#### Snapshots with *virsh* (CLI)

```
virsh help snapshot

virsh snapshot-create-as RemoteBox2 Snap3 "Descriprion here..."
```

For details, see: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Virtualization_Deployment_and_Administration_Guide/sect-Managing_guest_virtual_machines_with_virsh-Managing_snapshots.html

### Convert Virtual Machines between KVM and WMWare

#### From KVM to VMWare

Shut down the KVM machine and convert its virtual disk into VMWare format with *qemu-img*. For example,

```
qemu-img convert RemoteBox2.img -O vmdk RemoteBox2.vmdk
```

The other option is to download a free (Windows) tool from https://www.starwindsoftware.com/converter.

The KVM virtual machine configuration can be exported into libvirt XML with

```
virsh dumpxml RemoteBox2 > RemoteBox2.xml
```

At this writing, there are no tools to convert this file directly into VMWare *vmx* definition file. We need to re-create the machine definition with e.g. VMWare workstation.

1. Download free 30-day evaluation copy of VMWare Workstation from http://www.vmware.com/try-vmware
2. Start creating a new custom virtual machine and pick a suitable virtual harware, and escially, "I will install the operating system later" (since it is already installed), and choose the existing *vmdk* image as the virtual disk.
3. Boot the converted virtual machine to check that it work.

#### From VMWare to KVM

KVM supports *vmdk* disk images directly. For a quick launch, open *virt-manager*, choose "Create a new virtual machine" -> "Import existing disk image".

The disk image can also be converted to *qcow2* to support hot snapshots and rollbacks. Example

```
qemu-img convert -f vmdk -O qcow2 centos7.vmdk centos7.qcow2
```

For details, see e.g.

http://docs.openstack.org/image-guide/content/ch_converting.html https://access.redhat.com/articles/1351473 http://www.linux-kvm.org/page/How_To_Migrate_From_Vmware_To_KVM http://manpages.ubuntu.com/manpages/utopic/man1/vmware2libvirt.1.html