Virtualization

KVM

KVM is a virtualization infrastructure for the Linux Kernel. If you're using a the standard Linux kernel from version 2.6.20 then you have support for KVM.

To determine your Linux kernel version to meet the software requirements:

```
# uname -r
2.6.32-279.el6.x86_64
```

!!! NB !!!
KVM doesn't perform any emulation but allows the /dev/kvm interface to be used by a user-space program like QEMU to perform the emulation of hardware.

KVM does have 2 very specific hardware requirements: 64 bit CPU and support for hardware assisted virtualization which is enabled in the BIOS.

To verify that your system meets the hardware prerequisites for KVM:

```
# grep flags /proc/cpuinfo | uniq | grep -E --color 'vmx|lm|svm'
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 rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep ermsf
```

Matching to the flag lm means that you have a 64 bit processor, vmx means that hardware assisted virtualization enabled in the BIOS for an Intel processor and svm means the same for an AMD processor.

Now that we've met the software and hardware requirements, we're able to use QEMU to create and manage virtual machines.

QEMU

QEMU is your Quick EMUlator and emulates a hardware environment for user-level processes. Virtualization across architectures is not supported, so one cannot run an AMD guest on an Intel host.

Paravirtualization vs Full Virtualization

Full virtualization uses a special kind of software called a hypervisor. The hypervisor interacts directly with the physical server's CPU and disk space. It serves as a platform for the virtual servers' operating systems. The hypervisor keeps each virtual server completely independent and unaware of the other virtual servers running on the physical machine. Each guest server runs on its own OS - you can even have one guest running on Linux and another on Windows. The hypervisor monitors the physical server's resources. As virtual servers run applications, the hypervisor relays resources from the physical machine to the appropriate virtual server. Hypervisors have their own processing needs, which means that the physical server must reserve some processing power and resources to run the hypervisor application. This can impact overall server performance and slow down applications.

The para-virtualization approach is a little different. Unlike the full virtualization technique, the guest servers in a para-virtualization system are aware of one another. A para-virtualization hypervisor doesn't need as much processing power to manage the guest operating systems, because each OS is already aware of the demands the other operating systems are placing on the physical server. The entire system works together as a cohesive unit.

Creating a Virtual Machine

A number of resources are available for creating Virtual Machines but we will focus on using the Virtual Machine Manager program called virt-manager and virt-install.

virt-manager is a GUI based program which allows you to easily create and further customize a virtual machine.

virt-install is a CLI based program which allows you to easily create a new virtual machine.

The command below creates a virtual machine called dev.web which uses 512M RAM and 1 Virtual CPU to install an Operating System from the URL ftp://192.168.122.1/pub/rhel6.3 using the disk /var/lib/images/dev.web.img

```
# virt-install -n dev.web -r 512 --vcpus=1 -l ftp://192.168.122.1/pub/rhel6.3 --disk /var/lib/images/dev.web.img
```

!!! TIP !!!
The disk used above was created as a sparse image file using the dd command:

```
# dd of=/var/lib/images/dev.web.img bs=1 seek=10G count=0
```

Connecting to a virtual machine’s console

The easiest way to get a console for your virtual machine is to use virt-manager and then double-click the virtual machine that you want to open a console to.

However, most enterprise level systems run headless and you will need to know how to access a console for a virtual machine across a remote connection. For this we use the virsh command.

To see the virtual machines you're running:

```
# virsh list
Id    Name                           State
----------------------------------------------------
2     RHWS300                        running
```

To connect to a virtual machine you can either specify the Name (RHWS300) or the ID (2):

```
# virsh console 2
Connected to domain RHWS300
Escape character is ^]
```

Once you are done working on that virtual machine, don't forget to logout or exit otherwise you will remain logged in. To finally disconnect from the virtual machine's console send the keys CTRL] to it.

Should you type in your username and not get any response then it you don't have a serial console configured on that virtual machine. You will now need to use virt-manager to connect to the virtual machine, log on and open up /boot/grub/grub.conf and add the following at the end of the line which begins with the word kernel:

```
console=ttyS0
```

For example:

```
# cat /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/VolGroup-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
```

```
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux (2.6.32-279.el6.x86_64)
  root (hd0,0)
  kernel /vmlinuz-2.6.32-279.el6.x86_64 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS
LANG=en_US.UTF-8 rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16
crashkernel=auto rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM
rhgb quiet console=ttyS0
  initrd /initramfs-2.6.32-279.el6.x86_64.img
```

You will need to reboot in order to make this change effective.

Starting Virtual Machines

The default auto start behavior is that any virtual machine that was started when the physical machine (the host) was shutdown, will automatically start when the physical machine boots up.

This may not be what you want and if you only want virtual machines to start automatically if you explicitly tell it to, then you need to change the following.

In /etc/sysconfig/libvirt-guests change the line which says:

```
ON_BOOT=start
```

to

```
ON_BOOT=ignore
```

Then restart the libvirtd with the command:

```
# service libvirtd restart
```

Now we need to go to the virtual machines and instruct them to start automatically when the physical machine starts (regardless of the virtual machine's state when the physical machine was shutdown or rebooted). You can use virt-manager for this purpose or virsh.

Using virt-manager:

Select by double clicking the virtual machine you want to auto start, then click VIEW > DETAILS > BOOT OPTIONS and tick Start virtual machine on host boot up

Using virsh:

```
# virsh autostart NAME
```

or

```
# virsh autostart ID
```

To disable the autostart status of a virtual machine using virsh use:

```
# virsh autostart NAME or ID --disable
```

Lab activity

Create 2 virtual machines for any operating systems you desire of which 1 must be Linux. Call the first VM1 and the second VM2.

VM1 should have the following specifications:

- 512MB RAM
- 1 CPU
- 8GB disk space
- 1 NIC mapped to eth0
- Should not be started automatically when the physical machine starts

VM2 should have the following specifications:

- 768MB RAM
- 2 CPU
- 10GB disk space
- 1 NIC mapped to eth0
- Should be started automatically when the physical machine starts

Make sure that you are able to invoke a console connection to your Linux virtual machine using the command virsh console VMx where x refers to the number of your Linux virtual machine.