Experiment 15: Virtualisation Environment

Aarya R Shankar

May 14, 2017

1 Aim

Virtualisation environment (e.g., xen, kqemu, virtualbox or lguest) to test an applications, new kernels and isolate applications. It could also be used to expose students to other alternate OSs like *BSD

2 Procedure

Sometimes we may want to install a new OS, but already 4 partitions are made in the disks, or we may want to test a software application, which may or may not harm your computer, etc. For these purposes, we install and use a virtualisation environment in our present operating system, and it will act as a separate machine. Hence, a single machine can act as different machines. There are several virtualisation softwares available for this purpose. Here, we'll be using KVM.
KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.
Note: It is preferred to use 64 bit kernel system as the host machine, as they can support 32 bit as well as 64 bit. In addition to that, 64 bit does not restrict RAM usage of the OS in the virtual environment (guest OS) to 2 GB, which will inevitably decrease its speed.
To know if our kernel is 64 bit, execute:
`uname -a`
Expected output: `x86_64 in the output`
To know if our processor supports 64-bit instructions, execute:
`egrep -c ' lm ' /proc/cpuinfo`

Expected output: `1 or higher`
To know if ou processor support hardware virtualisation, execute:
`egrep -c '(vmx|svm)' /proc/cpuinfo`
Expected output: `1 or higher`

Assuming that all the outputs are as expected, let us now install KVM, by executing:

```
sudo apt-get install qemu-kvm libvirt-bin bridge-utils
```

To install the GUI tool, execute: (Not necessary)

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

If we are not a root user and we want to launch VMs without root privileges, log out and log in after executing:

```
sudo adduser username libvirt
```

Now run:

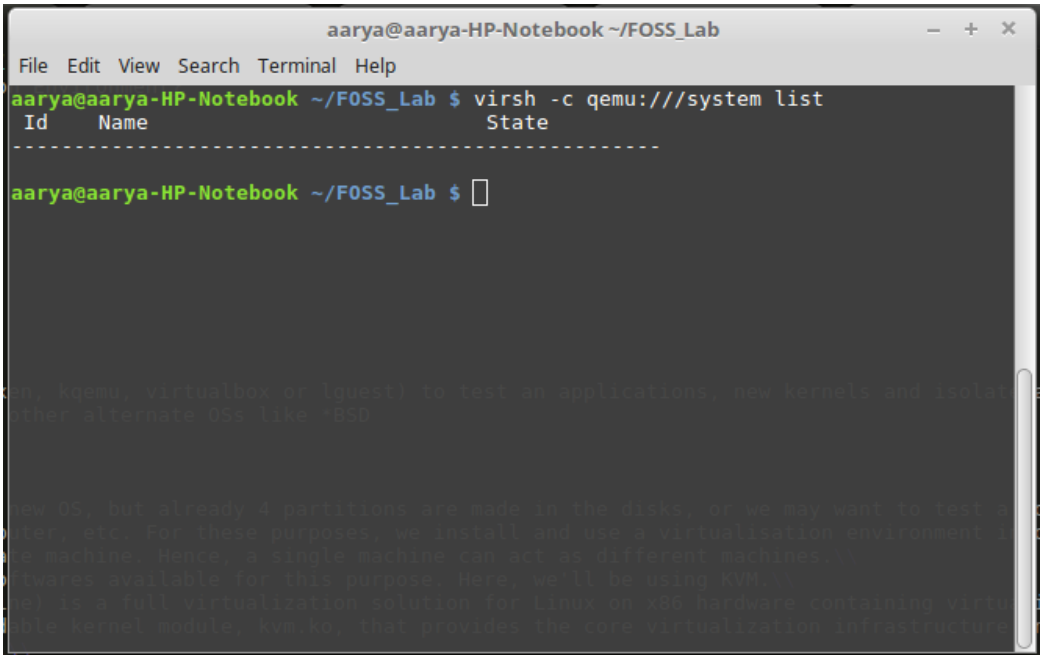
```
virsh -c qemu:///system list
```

If the output is an empty set of VMs, it means that the KVM is installed successfully.

Now we have successfully set up the KVM, we need to create a hard disk image, which are of 2 types: raw and qcow2.
A raw disk image file is an exact bit-for-bit copy of the hard drive, that is, a complete copy of all of the data stored on the source drive. It provides the lease I/O overhead, but it causes lots of space-wastage as the unused space of the guest OS can't be used by the host OS.
A qcow2 disk image only allocates the physical space needed by actual data stored to the virtual disk, ie, entire virtual disk size doesnt need to be allocated on the hard drive when its created. Due to this, the space usage is optimal, but this type has a processing, as well as I/O overhead, due to which its performance will not be as good as the former.

To create a hard disk image of 10 GB in raw format, execute:

```
qemu-img create -f raw firstvirtimg.img 10G
```



To create a hard disk image of 10 GB in qcow2 format, execute

```
qemu-img create -f qcow2 firstvirtimg.qcow2 10G
```

To verify the image, execute:

```
ls -lh firstvirtimg.img
```

To resize the disk image (raw or qcow2 type) to 12GB, execute:

```
qemu-img resize firstvirtimg.img +2G
```

Now execute:

```
ls -lh firstvirtimg.img
```

We can see that the size of the disk image increased by 2GB.
To convert a qcow image 'test.img' to a raw 'test.raw' file, run:

```
qemu-img convert test.img -O raw test.raw
```

To convert a raw 'test.img' file to a qcow image 'test.qcow', run:

```
qemu-img convert test.img -O qcow2 test.qcow2
```

To verify the image, execute:
`ls -lh firstvirtimg.img`
To resize the disk image (raw or qcow2 type) to 12GB, execute:

```
qemu-img resize firstvirtimg.img +2G
```

Sometimes we need to revert a change we made in the guest system. For this, we make a backup image or a storage image and a overlay image to keep the changes we make to the storage image. So when we want to reverse the last change, we only need to create an overlay image of the current storage image. To create an overlay image, execute:

`qemu-img create -f qcow2 -b firstvirtimg.qcow2 firstvirtimg.ovl`
Now we can run our QEMU VM as usual by using the overlay image so as to save the new changes in it, by executing the command:

```
qemu-system-i386 -enable-kvm firstvirtimg.ovl
```

To run Arch Linux without a virtual hard drive in the virtualised system, execute:
`kvm -m 2048 -cdrom archlinux-2017.02.01-dual.iso -boot d`

Here `kvm` is a script which executes `qemu-system-x86_64 -enable-kvm`
`-enable-kvm` enables kvm extensions
`-m num` sets the RAM of the virtual system to num megabytes
`-cdrom` uses the iso file as CD-ROM image
`-boot d` specifies CD-ROM drive (indicated by d) as the first one in boot order.

The KVM has successfully run the arch linux iso file. It is also possible to install and run any other operating system or application in the KVM.

3 Result

The KVM is successfully set up. The virtual machine has been successfully booted with the iso file of Arch Linux.