QEMU/Images

Once QEMU has been installed, it should be ready to run a guest OS from a disk image. This image is a file that represents the data on a hard disk. From the perspective of the guest OS, it actually is a hard disk, and it can create its own filesystem on the virtual disk.

You can download a few guest OS images from the QEMU website (http://wiki.qemu.org/Testing), including a simple 8 MB image of a Linux distro (which is meant primarily for testing; note that it lacks the e1000 driver (http://www.nuke24.net/docs/2012/QEMUNetworking.txt) and therefore cannot do networking out-of-the-box). To run it, download and unzip the image in a folder and run the QEMU command.

```
qemu linux-0.2.img
```

When you do not have the plain command *qemu* try to run:

```
qemu-system-i386 linux-0.2.img
```

Replace *linux-0.2.img* with the name of your guest OS image file. If it has a GUI and you want to use your mouse with it, double-click on the window and QEMU will grab your mouse. To make QEMU release your mouse again, hold down the Control and Alt keys simultaneously, then let go - your mouse will be released back to X.

Aurélien Jarno of Debian has prepared a number of pre-packaged Debian QEMU images for several architectures, including ARM, Sparc, PowerPC, x86_64, and i386. They can be found at http://people.debian.org/~aurel32/qemu/

For some of the test kernels that you get, for example, the Sparc test image, you might get the error "No NFS Server available giving up" or some other message about needing to insert a "Root Floppy" or even a message specifying to fix the root option. If so then specify "root=/dev/ram" as an addition kernel command line option. This will then cause the test system to work. This is specifically true for the Sparc test files available from http://wiki.qemu.org/Testing, but probably for some of the others as well. If you are using the manager program for windows then just add the "root=/dev/ram/" in the advanced tab. Also turn off video and sound and if using qemu-system-sparcw.exe then change the filename to qemu-system-sparc.exe because some renamed it.

Contents

Image types

QEMU supports several image types. The "native" and most flexible type is *qcow2*, which supports copy on write, encryption, compression, and VM snapshots.

QEMU currently can use these image types or formats:

raw
: (default) the raw format is a plain binary image of the disc image, and is very portable. On filesystems that support sparse files, images in this format only use the space actually used by the data recorded in them.

cloop
: Compressed Loop format, mainly used for reading Knoppix and similar live CD image formats

cow
: copy-on-write format, supported for historical reasons only and not available to QEMU on Windows

qcow
: the old QEMU copy-on-write format, supported for historical reasons and superseded by qcow2

qcow2
: QEMU copy-on-write format with a range of special features, including the ability to take multiple snapshots, smaller images on filesystems that don't support sparse files, optional AES encryption, and optional zlib compression

vmdk
: VMware 3 & 4, or 6 image format, for exchanging images with that product

vdi
: VirtualBox 1.1 compatible image format, for exchanging images with VirtualBox.

vhdx
: Hyper-V compatible image format, for exchanging images with Hyper-V 2012 or later.

vpc
: Hyper-V legacy image format, for exchanging images with Virtual PC / Virtual Server / Hyper-V 2008.

Creating an image

To set up your own guest OS image, you first need to create a blank disc image. QEMU has the `qemu-img` command for creating and manipulating disc images, and supports a variety of formats. If you don't tell it what format to use, it will use raw files. The "native" format for QEMU is qcow2, and this format offers some flexibility. Here we'll create a 3 GB qcow2 image to install Windows XP on:

```
qemu-img create -f qcow2 winxp.img 3G
```

The easiest way to install a guest OS is to create an ISO image of a boot CD/DVD and tell QEMU to boot off it. Many free operating systems can be downloaded from the Internet as bootable ISO images, and you can use them directly without having to burn them to disc.

Here we'll boot off an ISO image of a properly licensed Windows XP boot disc. We'll also give it 256 MB of RAM, but we won't use the kqemu kernel module just yet because it causes problems during Windows XP installation.

```
qemu -m 256 -hda winxp.img -cdrom winxpsp2.iso -boot d
```

To boot from a real CD or DVD, tell QEMU where to find it. On Linux systems, you can usually use a logical device name like `/dev/cdrom` or `/dev/dvd`, or the physical name of the device, e.g. `/dev/sr0`

```
qemu -m 256 -hda winxp.img -cdrom /dev/cdrom -boot d
```

QEMU will boot from the ISO image or CD/DVD and run the install program. If you have two screens, move the QEMU screen off to the spare one where you can keep an eye on the installer, but get on with something else - it will take a while!

Once the guest OS has installed successfully, you can shutdown the guest OS (e.g. in Windows XP, click on Start and then Shutdown). Once it has shutdown, start QEMU up with the kqemu kernel module to give it a little more speed.

```
qemu -m 256 -hda winxp.img -cdrom winxpsp2.iso -enable-kvm
```

If you are running an x86-64 Linux (i.e. 64-bit), you will need to run the x86-64 version of QEMU to be able to utilise kqemu:

```
qemu-system-x86_64 -m 256 -hda winxp.img -cdrom winxpsp2.iso -enable-kvm
```

Using multiple images

QEMU can utilise up to four image files to present multiple virtual drives to the guest system. This can be quite useful, as in the following examples:

- a pagefile or swapfile virtual disc that can be shared between QEMU guests
- a common data drive where all data is stored, accessible from each QEMU guest but isolated from the host
- giving additional space to a QEMU guest without reconfiguring the primary image
- separating competing I/O operations onto different physical drive spindles by placing the separate QEMU images on different physical drives
- emulating a multi-drive physical environment for testing / learning

Bear in mind that only one instance of QEMU may access an image at a time - shared doesn't mean shared simultaneously!

To use additional images in QEMU, specify them on the command line with options -hda, -hdb, -hdc, -hdd.

```
qemu -m 256 -hda winxp.img -hdb pagefile.img -hdc testdata.img -hdd tempfiles.img -enable-kvm
```

NB: QEMU doesn't support both -hdc and -cdrom at the same time, as they both represent the first device on the second IDE channel.

Copy on write

The "cow" part of qcow2 is an acronym for copy on write, a neat little trick that allows you to set up an image once and use it many times without changing it. This is ideal for developing and testing software, which generally requires a known stable environment to start off with. You can create your known stable environment in one image, and then create several disposable copy-on-write images to work in.

To start a new disposable environment based on a known good image, invoke the `qemu-img` command with the `backing_file` option and tell it what image to base its copy on. When you run QEMU using the disposable environment, all writes to the virtual disc will go to this disposable image, not the base copy.

```
qemu-img create -f qcow2 -o backing_file=winxp.img test01.img
qemu -m 256 -hda test01.img -enable-kvm &
```

NB: 1) don't forget to copy any important data out of the disposable environment before deleting it. When developing and testing software in copy-on-write virtual environments, it is a good idea to use version control software like Subversion or CVS on a server external to your virtual environment. Not only is it easy to keep copies of your work outside your virtual environment, it is also very easy to set up a new virtual environment from version control. 2) backing_file image must be set as read-only on VMs, that will be using it directly. Otherwise, other images, based on this backing_file image will be corrupted, if there is some change have been made to the backing_file image.

Mounting an image on the host

Sometimes it is helpful to be able to mount a drive image under the host system. For example, if the guest doesn't have network support, the only way to transfer files into and out of the guest will be by the storage devices it can address.

Linux and other Unix-like hosts can mount images created with the *raw* format type using a loopback device. From a root login (or using sudo), mount a loopback with offset.

```
mount -o loop,offset=32256 /path/to/image.img /mnt/mountpoint
```

To determine the correct offset you can run

```
fdisk -l /path/to/image.img
```

and the offset you need is the *start* of a partition multiplied by sector size. For example if start is 128 and sector size is 512 then the offset is 65536.

For example, to copy some files across to a FreeDOS hard drive image:

```
mkdir -p /mnt/freedos
mount -o loop,offset=32256 freedos-c.img /mnt/freedos
cp oldgames /mnt/freedos
umount /mnt/freedos
```

NB: never mount a QEMU image while QEMU is using it (unless -snapshot is used), or you are likely to corrupt the filesystem on the image.

Note: if you have an image without partitions you should omit the ,offset=32256 part. This is for instance the case if you want to mount linux-0.2.img (which can be found at the qemu web site at the time of writing).

For other types of qemu images, you can use qemu-nbd

```
modprobe nbd max_part=16
qemu-nbd -c /dev/nbd0 image.qcow2
partprobe /dev/nbd0
mount /dev/nbd0p1 /mnt/image
```

Using **fdisk** you can get information regarding the different partitions in nbd0.

```
$ fdisk /dev/nbd0
 Command (m for help): p
 Disk /dev/nbd0: 4294 MB, 4294967296 bytes
 255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors
 Units = sectors of 1 * 512 = 512 bytes
 Sector size (logical/physical): 512 bytes / 512 bytes
 I/O size (minimum/optimal): 512 bytes / 512 bytes
 Disk identifier: 0x000183ca
      Device  Boot     Start        End      Blocks   Id  System
 /dev/nbd0p1   *        2048      499711      248832   83  Linux
 /dev/nbd0p2          501758     8386559     3942401    5  Extended
 /dev/nbd0p5          501760     8386559     3942400   8e  Linux LVM
```

LVM type partitions cannot be mounted using 'mount'. In such a case the image could be mounted with:

```
$ vgscan
  Reading all physical volumes. This may take a while...
  Found volume group "ub1110server-qemu" using metadata type lvm2
$ vgchange -ay
```

```
   2 logical volume(s) in volume group "ub1110server-qemu" now active
$ mount /dev/ub1110server-qemu/<LogicalVolumeName> /mnt/image
```

Replace <*LogicalVolumeName*> with the name of the logical volume present in the volume group. Finally, after the usage it's important to unmount the image and reverse other steps (depending on how the image was mounted):

```
$ umount /mnt/image
$ qemu-nbd -d /dev/nbd0
$ vgchange -an VolGroupName
$ killall qemu-nbd
```

"nbd" stands for "Network Block Device". Here we're using them locally, but one can also export the image for other hosts to use/mount. See qemu-nbd(8), nbd-client(8) for details.

The same can be done for raw images, but it's a lot less efficient than the loop method described above.

Another alternative for vmdk and raw images is to use vdfuse (http://forums.virtualbox.org/viewtopic.php?f=26&t=33355) which allows to access any image supported by VirtualBox.

Copying an image to a physical device

It may be desired to copy a diskimage to a physical device. An example may be if building a cluster, it might be easier to get everything ready in qemu, then write the final diskimage to all of the hard drives. Of course your image will need to contain all of required configuration and drivers for the new system to boot properly.

The diskimage will need to be in raw format.

```
$ qemu-img convert -O raw diskimage.qcow2 diskimage.raw
```

Then you just dd it onto the hard drive.

```
# dd if=diskimage.raw of=/dev/sdX
```

Or, let qemu-img directly write onto the drive in one command:

```
# qemu-img convert -O raw diskimage.qcow2 /dev/sdX
```

Of course you need to be careful that you write it to the correct drive!

Getting information

The `qemu-img` program can tell you about the format, virtual size, physical size, and snapshots inside an image.

```
$ qemu-img info test.vmdk
(VMDK) image open: flags=0x2 filename=test.vmdk
image: test.vmdk
file format: vmdk
virtual size: 20M (20971520 bytes)
disk size: 17M
```

Converting image formats

The `qemu-img` program can be used to convert images from one format to another, or add compression or encryption to an image. Specify the source and target files for the image, and select from the following options:

- `-f` *`fmt`* – optional, specify the format of the input file (QEMU can usually detect it)
- `-O` *`fmt`* – specify the format of the output file
- `-e` – use encryption in the output file (you will be prompted for a password)
- `-c` – use compression in the output file (can't be used with encryption)
- `-6` – when converting to vmdk (VMware) format, make it compatible with VMware 6

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

Exchanging images with VirtualBox

qemu also supports the *vdi* format, so you can convert in both ways:

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

qemu can use the vdi format for booting as well as a base image; because VirtualBox also supports this type of image, called there differencing image, a single (read-only) base copy could be shared simultaneously.

In any case, you may use the *raw* format (or an *nbd* device):

To convert a QEMU image for use with VirtualBox, first convert it to *raw* format, then use VirtualBox's conversion utility to convert and compact it in its native format. Note that the compact command requires the full path to the VirtualBox image, not just the filename.

```
qemu-img convert -O raw test.qcow2 test.raw
(1) VBoxManage convertdd test.raw test.vdi
VBoxManage modifyvdi /full/path/to/test.vdi compact
```

(1) or try:

```
VBoxManage convertfromraw -format VDI test.raw test.vdi
or
VBoxManage clonehd -format VDI test.raw test.vdi
```

Alternatively, you may use qemu-nbd to make the image available as raw as an nbd device, which VBoxManage can then convert.

```
qemu-nbd -c /dev/nbd0 test.qcow2
VBoxManage convertfromraw -format VDI /dev/nbd0 test.vdi
```

To convert a VDI back to raw, one may use VBoxManage again:

```
VBoxManage clonehd -format RAW test.vdi test.raw
```

or

```
VBoxManage internalcommands  converttoraw file.vdi file.raw
```

To use a VDI image from KVM without converting it, vdfuse may be used again.

```
vdfuse -f test.vdi ~/some-dir
kvm -hda ~/some-dir/EntireDisk ...
```
