Translation(s) : English - Italiano - Русский

QEMU

DebianPkg: QEMU is a fast processor emulator using dynamic translation to achieve good emulation speed. It is a free open-source alternative to VMware.

As QEMU requires no host kernel patches to run, it is very safe and easy to use.

Contents

Operation

QEMU has two operating modes:

- Full system emulation. In this mode, QEMU emulates a full system (for example a PC), including a processor and various peripherals. It can be used to launch different Operating Systems without rebooting the PC or to debug system code.
- User mode emulation (Linux host only). In this mode, QEMU can launch Linux processes compiled for one CPU on another CPU. For example, it can be used to launch DebianPkg: Wine or to ease cross-compilation and cross-debugging.

Furthermore, there are two options for running QEMU:

	Pros	**Cons**
QEMU *(plain)*	Doesn't require a kernel module	Not as fast as the others
QEMU / kvm	Fastest	Requires the kvm module + x86 CPU w/ virtualization extensions + granting the user R/W access to `/dev/kvm`

Installation

The QEMU emulator is packaged as DebianPkg: qemu, which can be installed using your preferred package management tool.

Usage

Setting up a stable system

Debian developer Aurelien Jarno maintains a list of ready-to-use Debian stable QEMU images at http://people.debian.org/~aurel32/qemu

Setting up a testing/unstable system

QEMU is especially handy to set up an emulated testing/unstable system when working on the Debian installer itself or on the boot system, or when trying some experimental features without impact on the productive system. A sid system can be set up with the following steps:

- Create the hard disk image with:

  ```
  $ qemu-img create debian.img 2G
  ```

  or with the qcow2 disk image format if you want to use QEMU's own "Copy On Write" image format:

  ```
  $ qemu-img create -f qcow2 debian.qcow 2G
  ```

- Download a current boot image, e.g. the businesscard image at http://cdimage.debian.org/cdimage/daily-builds/daily/arch-latest/i386/iso-cd/
- Boot the image with:

  ```
  $ qemu -hda debian.img -cdrom debian-testing-i386-businesscard.iso -boot d -m 256
  ```

- When the usual debian boot screen appears, boot into "expert" mode.
- Install the system as usual; to set up a sid system choose "unstable" when being asked by the installer.

After the installation is done, the system can be booted with:

```
$ qemu -hda debian.img -m 256
```

Backing up the disk image

The disk image "debian.img" is a sparse file. After installing a Debian base system, it fits on a CD-ROM even without compression:

```
$ tar c --sparse -f backup.tar debian.img
```

This creates a tar file of about 320M (supposed that the image contains a 1.9GB ext3 root filesystem and a 250MB swap partition). After unpacking with tar xf, the sparse file is restored and can be booted immediately.

Better still, convert from a sparse file into the qemu's own "Copy On Write" image. This conversion will save the same space and still be runnable:

```
$ qemu-img convert -c debian.img -O qcow debian_recompressed.img
```

If the guest system's image is still larger than reasonable, then open up the Guest system and run "`dd if=/dev/zero of=/tmp/junk ; sync ; rm /tmp/junk`". That will push out deleted file scraps, recompression should work then.

Networking

Guests on NATed internal network

By default, QEMU invokes the `-nic` and `-user` options to add a single network adapter to the guest and provide NATed external Internet access. The host and guest will not see each other.

Host and guests on same network

To create a bridge between host and guests, do the following (tested on DebianSqueeze). Please note that all these changes must be done on the host system.

1. Install DebianPkg: bridge-utils.
2. Edit `/etc/network/interfaces`:
    1. Remove the 'auto' line and change the 'method' of your physical, wired network adapters from 'auto' to 'manual':

    ```
    #auto eth0
    iface eth0 inet manual
    ```

    2. Add a stanza for the bridge:

    ```
    auto br0
    iface br0 inet dhcp
       pre-up ip tuntap add dev tap0 mode tap user <username>
       pre-up ip link set tap0 up
       bridge_ports all tap0
       bridge_stp off
       bridge_maxwait 0
       bridge_fd      0
    ```

```
    post-down ip link set tap0 down
    post-down ip tuntap del dev tap0 mode tap
```

Explanation: this stanza auto loads a bridge and configures it using DHCP. A tap device is created owned by <username> and is brought up; please note that <username> is the username on the host system. On the 'bridge_ports' line, 'all' adds all physical interfaces to the bridge; virtual interfaces have to be listed explicitly. The three other bridge directives will speed up the activating of the bridge. See DebianMan: bridge-utils-interfaces(5) - note that the man page warns against adding wireless adapters to the bridge (see also BridgeNetworkConnections).

3. Launch QEMU:

```
$ qemu -hda imagefile.img -net nic -net tap,ifname=tap0,script=no,downscript=no
```

- `ifname=tap0` - the tap name here corresponds with the name in the bridge stanza above.
- `script=no,downscript=no` disable the scripts `/etc/qemu-ifup` and `/etc/qemu-ifdown` as they are not needed.

The guest virtual network adapters will be configured by the `/etc/network/interfaces` file in the guest file system. DHCP will work so this is simplest way to go.

4. To run additional guests, duplicate the lines in the bridge stanza for tap1, tap2 as needed and change the ifname argument in the command line. If you run more guests from the same image file, udev will rename the interface to avoid duplication (e.g. eth0 => eth1), so add extra interface stanzas in the guest interfaces file to configure these.

QEMU networking with VDE

Virtual Distributed Ethernet (VDE) provides is a virtual switch that can connect multiple virtual machines together, both local and remote. With VDE it is possible to create a virtual network of QEMU machines running on one or more real computers.

1. Install DebianPkg: vde2 and DebianPkg: uml-utilities (for tunctl).
2. Add users which will be running VM's to the vde2-net group.
3. Add to `/etc/network/interfaces`:

```
auto mytap
iface mytap inet static
    address 10.0.3.1
    netmask 255.255.255.0
    vde2-switch -t mytap
```

4. If automatic configuration of guest network interfaces is desired, install a DHCP server such as dnsmasq as well. Configure it to assign addresses only on the TAP interface (e.g. 'mytap' in the above example). If you use dhcpd, make sure mytap is added to /etc/default/dhcp3-server and the INTERFACES directive. anm
5. Either reboot or run the following:

```
# modprobe tun
# ifup mytap
# /etc/init.d/dnsmasq restart #if used
$ newgrp vde2-net #run as user starting Qemu VM's
```

6. Finally, start the VM:

```
$ qemu -net nic -net vde,sock=/var/run/vde2/mytap.ctl debianimage.img
```

If running more than one VM, you need to provide a unique MAC address:

```
$ qemu -net nic,macaddr=52:54:00:12:01:00 -net vde,sock=/var/run/vde2/mytap.ctl debianimage.img
```

(in DebianLenny, vdeqemu was needed for running qemu with VDE, but recent versions of qemu has vde enabled already.) If copying a MAC address from an existing interface, make it unique by altering any of the last 6 digits at the end. The first 6 designate the card manufacturer and so should not be made up. If running kvm instead of QEMU, the command name is 'vdekvm'.

7. QEMU VM's networked under VDE are not automatically NATed as they are under QEMU's default User Mode networking. For guest external network access you must enable NAT/Masquerading via iptables. An iptables rules-building program such as DebianPkg: shorewall makes this easier. You may also use NAT/Masquerade script from tldp

More info on VDE:

- VDE Forum
- VDE wiki

Other

Mounting QEMU images

Never mount a QEMU image while QEMU is using it, or you are likely to corrupt the filesystem(s) within.

raw

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 an offset of 32,256:

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

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

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

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).

qcow2

You will need a 2.6.26 kernel or newer, so Lenny will do. Load the ndb module:

```
# modprobe nbd max_part=8
```

If you leave off the max_part attribute, partitions are not supported and you’ll be able to access the disk, but not have device nodes for any of the partitions.

Assuming your qcow2 image filename is *imagename.qcow*, run:

```
# kvm-nbd --connect=/dev/nbd0 imagename.qcow
```

Now you can check the partition table with:

```
# fdisk -l /dev/nbd0
```

and mount, e.g., the first partition on /mnt with:

```
# mount /dev/nbd0p1 /mnt
```

See Also

QEMU related tools:

- KVM
- qemubuilder - creates Debian packages sandboxed with QEMU
- QEMU frontends:
  - DebianPkg: qemu-launcher
  - DebianPkg: qemulator (since Wheezy superseded by DebianPkg: virtualbricks)

Alternative or similar tools:

- SystemVirtualization tools
- Emulator tools

External Links

- http://wiki.qemu.org/ - QEMU homepage (visited 04/2012)
- http://qemu-buch.de/cgi-bin/moin.cgi/FrequentlyAskedQuestions - QEMU FAQ (visited 04/2012)
- #qemu - IRC channel

- Other Guides
  - Networking KVM (applies to QEMU) (visited 04/2012) *Note: the command 'qemu-system-x86_64' in this guide on Debian Etch is 'kvm'*
  - Debian HOW-TO : QEMU Virtual Machine (visited 04/2012)
  - Qemu Networking. (visited 04/2012)
  - Understanding networking in QEMU Guests (visited 04/2012)

CategoryEmdebian