The Quick Guide to QEMU Setup

By Manoj Kumar on May 1, 2011 in For You & Me, Home / Quick Tips, How-Tos · 11 Comments

In this series of articles, we will explore the basics of QEMU, OS installation, QEMU networking and embedded system development for the ARM architecture. In this first part, let's begin with the basics.

When I first began using computers, I was amazed. It required just a mouse-click to play songs, movies or games to entertain every age. It was like magic to me! Over time, I learnt about compiled programs, and source code. My curiosity very soon made source code my passion. Even though compiled software packages are easily available, I love compiling from source. And that is just what I do for QEMU.

QEMU is one of the best emulators out there. Still, very few people use its full capabilities. Though we deal with the basics in this article, look forward to some interesting stuff later in the series!

Building QEMU from source

The first thing is to download the QEMU source code; the current version as of this writing is 0.14 and you'll find it here. Extract the tar ball and go to the extracted directory:

```
$ tar -zxvf qemu-0.14.0.tar.gz
$ cd qemu-0.14.0
```

Run the configuration script. We will build QEMU for i386. (It can be built for other architectures too, like ARM, PPC, SPARC, etc.) Let's install the Ubuntu distro in the virtual machine — that's the reason we've chosen to build QEMU for the i386 architecture:

```
$ ./configure –target-list=i386-softmmu
```

Hopefully, you will not run into any trouble during the configure script run. If there's any issue, it will probably be some missing library or header files, which you can look for, and install.

Once you are done with the configure script, compile the source code with the `make` command. After compilation, QEMU binaries should be installed in their proper locations. On my Fedora system, I used the `su` command to get the necessary root privileges and install the binaries using `make install`.

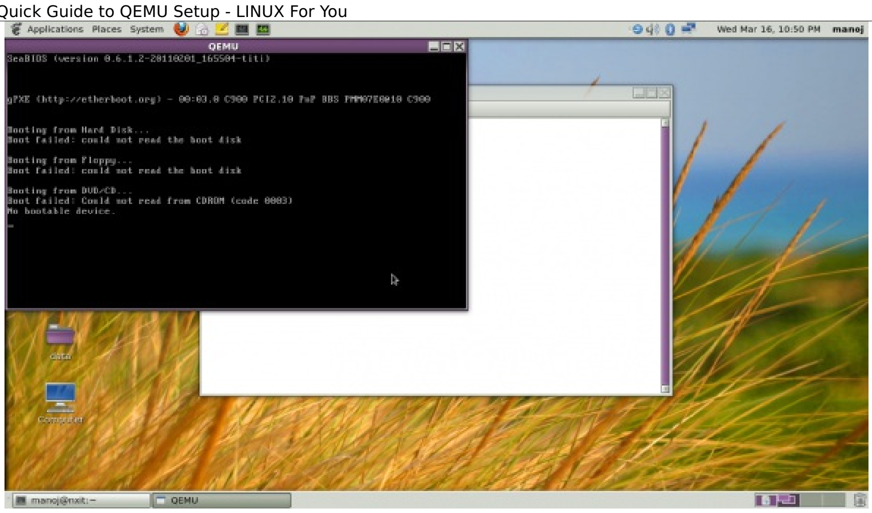
To confirm that QEMU has been successfully installed, run `qemu`, and a pop-up window like what appears in Figure 1 will confirm the successful installation of QEMU.

Search for: Search

Get Connected

RSS Feed Twitter

4



```
SeaBIOS (version 0.6.1.2-20110201_165504-titi)


gPXE (http://etherboot.org) - 00:03.0 C900 PCI2.10 PnP BBS PMM07E0@10 C900


Booting from Hard Disk...
Boot failed: could not read the boot disk

Booting from Floppy...
Boot failed: could not read the boot disk

Booting from DVD/CD...
Boot failed: Could not read from CDROM (code 0003)
No bootable device.
_
```

Figure 1: Testing QEMU after installation

Creating a new virtual machine

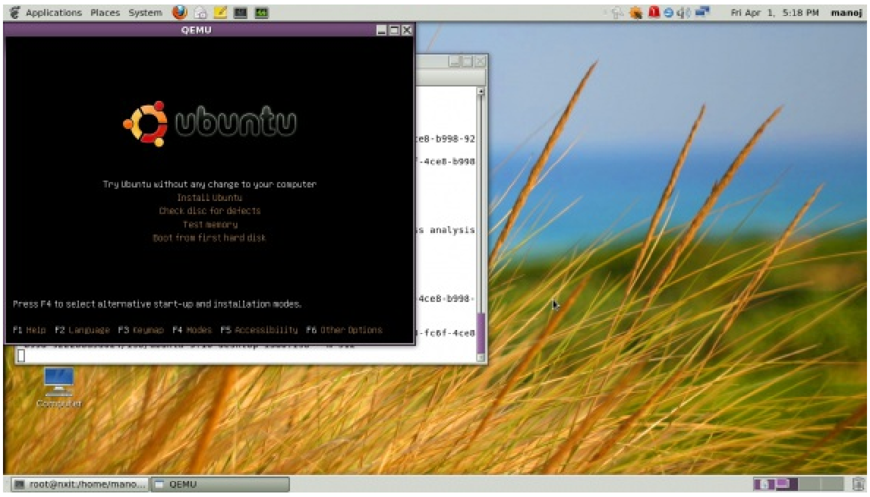
If you are familiar with other virtualisation software, you might wonder how to go about creating a new VM. The first step is to create the hard disk image for the VM. So let's install the Ubuntu 9.10 OS in the VM; a disk image of 10 GB is sufficient for this purpose. To create it, run the following commands:

```
$ qemu-img create ubuntu.img 10G
$ ls -lh ubuntu.img
-rw-r--r--. 1 root root 10G Mar 11 11:54 ubuntu.img
```

The next step is to install Ubuntu (I already have a downloaded Ubuntu 9.10 (Karmic) ISO image in my current working directory):

```
$ qemu -hda ubuntu.img -boot d -cdrom ./ubuntu-9.10-desktop-i386.iso -m 512
```

In the above command, the `-hda` option specifies the disk image file; and `-cdrom` is the CD-ROM or ISO image to use as the "optical drive" for the VM. The `-m` option specifies how much RAM this virtual machine will be given to use (in this case, I have allocated 512 MB of RAM; your choice should be based on your needs and hardware). Finally, we instruct QEMU to boot the VM from the ISO image by using the `-boot d` option. Once you run the above command, the VM will boot up and present the Ubuntu boot menu (see Figure 2).



```
Try Ubuntu without any change to your computer
Install Ubuntu
Check disc for defects
Test memory
Boot from first hard disk

Press F4 to select alternative start-up and installation modes.

F1 Help  F2 Language  F3 Keymap  F4 Modes  F5 Accessibility  F6 Other Options
```

Figure 2: Installing Ubuntu in QEMU

Follow the same installation steps you would use on a real machine. Once installed, you can boot the VM from the disk image with the following commands:

```
$ qemu -m 512 -hda ubuntu.img
```

Figure 3 shows the VM running after booting from the virtual hard disk.

Popular Comments Tag cloud

August 13, 2013 · 46 Comments · Diksha P Gupta
'India has immense under-utilised talent in the cloud security space'

June 20, 2013 · 5 Comments · sophie-samuel
New and amazing features of Linux

June 20, 2013 · 3 Comments · Priyanka Sarkar
What it Takes to be an Open Source Expert

August 24, 2013 · 3 Comments · Priyanka Sarkar
Secure Your Career with 'Ethical Hacking'!

September 21, 2013 · 1 Comments · Mihalis-T
Getting Started with WireShark

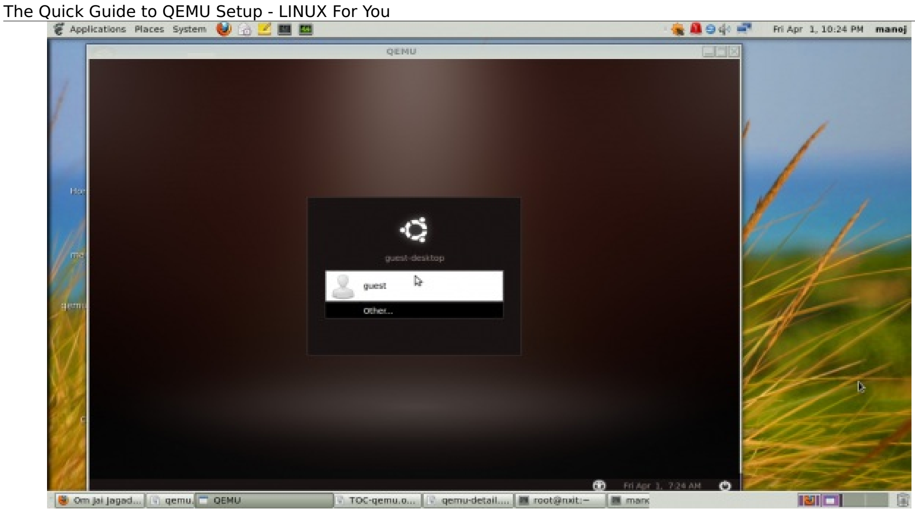


Figure 3: Booting the installed operating system

The next thing we need to do is set up networking.

QEMU networking

Setting up networking on QEMU is tricky work. Let's make use of the virtual network kernel devices TAP and TUN, which are different from hardware Ethernet devices; TAP and TUN are supported only in the kernel (i.e., only in software). TAP operates at the data-link layer, and TUN at the network layer.

QEMU can use the TAP interface to provide full networking support to the guest operating system. Before this, we need to install the VPN (Virtual Private Network) package on the host machine, and set up a bridge between the host and guest OS. Install the `openvpn` and `bridge-utils` packages:

```
# yum install openvpn
# yum install bridge-utils
```

Now, we will create two scripts for `qemu—qemu-ifup` and `qemu-ifdown`, as given below:

```
    #qemu-ifup
 1  /sbin/ifconfig eth1 down
 2  /sbin/ifconfig eth1 0.0.0.0 promisc up
 3  openvpn --mktun --dev tap0
 4  ifconfig tap 0 0.0.0.0 up
 5  brctl addbr br0
 6  brctl addif br0 eth1
 7  brctl addif br0 tap0
 8  brctl stp br0 off
 9  ifconfig br0 10.10.10.2 netmask
10  255.255.255.0
```

This script will be used to start QEMU networking. In the first line, the Ethernet device is disabled. For the interface to be a part of a network bridge, it must have an IP address of 0.0.0.0, which is what we have done in the second line. In lines 3 and 4, we create and bring up the TAP device/interface `tap0`. In the next few steps, a bridge is created with `eth1` and `tap0` as parts of this bridge. Finally, we assign an IP address to this bridge.

Following is what the `qemu-ifdown` script looks like:

```
   #qemu-ifdown
1  ifconfig eth1 down
2  ifconfig eth1 -promisc
3  ifup eth1
4  ifconfig br0 down
5  brctl delbr br0
6  openvpn --rmtun --dev tap0
```

This script will be used to shutdown QEMU networking; it is almost self-explanatory, shutting down both the interfaces, deleting the bridge, and the `tap0` device.

Copy these two files to your `/etc` directory, and test them:

```
# /etc/qemu-ifup
Wed Apr 6 15:53:50 2011 TUN/TAP device tap0 opened
Wed Apr 6 15:53:50 2011 Persist state set to: ON

# ifconfig br0
br0       Link encap:Ethernet HWaddr 00:25:11:74:5B:0C
          inet addr:10.10.10.2 Bcast:10.10.10.255 Mask:255.255.255.0
          inet6 addr: fe80::225:11ff:fe74:5b0c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:7539 (7.3 KiB)

#ifconfig tap0
tap0      Link encap:Ethernet HWaddr C2:10:27:8C:B8:35
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

The virtual device `tap0` and bridge `br0` are up, so our script is working fine. Now bring it down, using the qemu-ifdown script:

```
# /etc/qemu-ifdown
Error: Connection activation failed: Device not managed by NetworkManager
Wed Apr 6 15:56:44 2011 TUN/TAP device tap0 opened
Wed Apr 6 15:56:44 2011 Persist state set to: OFF
```

Everything is set up correctly; now it's time to boot the Ubuntu VM with full networking support. Start the networking (as root), and boot the VM (as an ordinary user):

```
# /etc/qemu-ifup
```

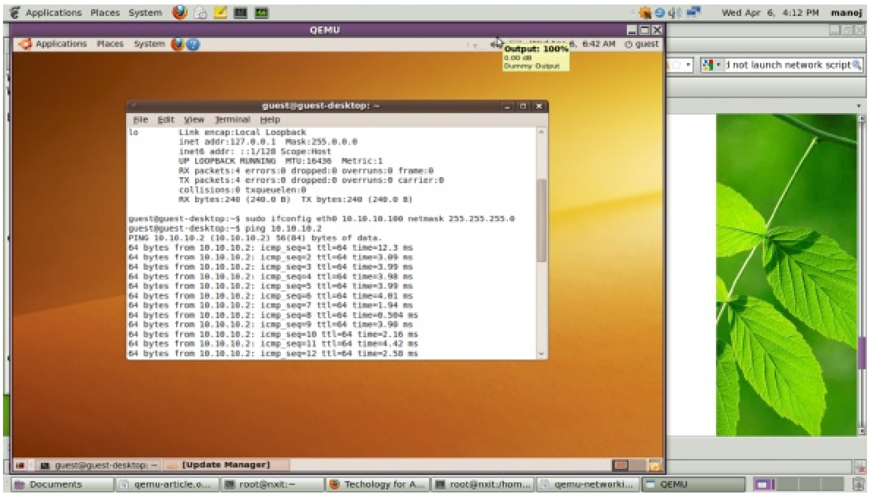
```
$ qemu -m 512 -hda ubuntu.img -net nic -net tap,ifname=tap0,script=no
```

When the machine boots up, assign an IP address to the `eth0` interface inside the VM:

```
$ sudo ifconfig eth0 10.10.10.100 netmask 255.255.255.0
```

Now try to ping the bridge IP (results are shown in Figure 4):

```
$ ping 10.10.10.2
```



```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:240 (240.0 B)  TX bytes:240 (240.0 B)

guest@guest-desktop:~$ sudo ifconfig eth0 10.10.10.100 netmask 255.255.255.0
guest@guest-desktop:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=12.3 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=3.09 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=3.99 ms
64 bytes from 10.10.10.2: icmp_seq=4 ttl=64 time=3.98 ms
64 bytes from 10.10.10.2: icmp_seq=5 ttl=64 time=3.99 ms
64 bytes from 10.10.10.2: icmp_seq=6 ttl=64 time=4.01 ms
64 bytes from 10.10.10.2: icmp_seq=7 ttl=64 time=1.94 ms
64 bytes from 10.10.10.2: icmp_seq=8 ttl=64 time=0.504 ms
64 bytes from 10.10.10.2: icmp_seq=9 ttl=64 time=3.90 ms
64 bytes from 10.10.10.2: icmp_seq=10 ttl=64 time=2.16 ms
64 bytes from 10.10.10.2: icmp_seq=11 ttl=64 time=4.42 ms
64 bytes from 10.10.10.2: icmp_seq=12 ttl=64 time=2.58 ms
```

Figure 4: QEMU networking using kernel virtual device

The network is working fine in the VM, now experiment with it. Try setting up a proxy server on the host OS, and connect to the Internet from the guest OS.

This was just an introduction to QEMU. You can look forward to a lot more in subsequent articles on how to use QEMU for embedded systems development.

Related Posts:

- Using QEMU for Embedded Systems Development, Part 3
- Using QEMU for Embedded Systems Development, Part 1
- Using QEMU for Embedded Systems Development, Part 2
- Virtualisation and Disk Management in OpenIndiana
- Kernel Development & Debugging Using the Eclipse IDE

Tags: architectures, ARM, bridge IP, Building QEMU, configuration script, Ethernet, Fedora, kernel virtual device, LFY May 2011, QEMU, RAM, tap0 device, TUN/TAP, ubuntu, virtual device, virtual machine, virtualisation software, VPN

Article written by:

Manoj Kumar

The author is a freelance developer and trainer. He leads a team in Linux kernel programming, Linux administration, cluster computing, embedded systems and QT/GTK programming on Linux. View and participate in the latest discussions on his Yahoo Group.

Connect with him: Website

Previous Post
Advanced Nmap: A Recap

Next Post
Securing Apache, Part 9: Attacks that Target PHP-based Instances

ALSO ON LINUX FOR YOU

Getting Started with WireSh
1 comment

The Semester Project-IV File Systems: Formatting a Pen

Cyber Attacks Explained: Th Botnet Army 1 comment

Learn the Art of Linux Troubleshooting 2 comments

11 comments

Leave a message...

Newest ▾ Community Share ↗

Peter • 4 months ago
Hi I followed the steps but on bootup of qemu it hangs at "Stopping configur network devices, waiting for network configuration" and when it times up, it bootup and I do not have eth0. I am wondering what could possibly cause th
△ | ▽ Reply Share ›

Suresh • 7 months ago
Hi Anil,

When i executed the second step ie "qemu -hda ubuntu.img -boot d -cdrom 9.10-desktop-i386.iso -m 512" it displays VNC server running on `127.0.0.1 the terminal got hanged ... i dnt know what is the problem... please help me

Thanks
Suresh
6 △ | ▽ Reply Share ›

Suresh > Suresh • 7 months ago
SDL enabled....
5 △ | ▽ Reply Share ›

Christoph P. • 8 months ago
The command $ ./configure -target-list=i386-softmmu must be changed to ./configure --target-list=i386-softmmu on my system to work. I downloaded http://download.savannah.gnu.o...
1 △ | ▽ Reply Share ›

Christoph P. > Christoph P. • 8 months ago
notice the double minus "--" in front of "target-list"
1 △ | ▽ Reply Share ›

Marxubuntu Buttonz • 2 years ago
this sounds exciting two thumbs up !!
△ | ▽ Reply Share ›

Prasad • 2 years ago
^ anand

you need a vnc client to see qemu's output....

install vncviewer & try the following command

vncviewer 127.0.0.1:5900
1 △ | ▽ Reply Share ›

Colombian > Prasad • a year ago

you must install libsdl-dev
sudo apt-get install libsdl-dev
and again configure.

△ | ▽ Reply Share ›

anand vamsi • 2 years ago

hi
im a newbie to qemu, when i executed the second step ie "qemu -hda ubunt
boot d -cdrom ./ubuntu-9.10-desktop-i386.iso -m 512" it displays VNC serve
on `127.0.0.1:5900 and the terminal got hanged ... i dnt know what is the
please help me out .

2 △ | ▽ Reply Share ›

lahwahleh > anand vamsi • 2 years ago

Hello, how did you solve this problem? I'm having the same problem he
you.

8 △ | ▽ Reply Share ›

nikhil • 2 years ago

Great article,
Gets everyone who'd like to use qemu started up.
You could also tell us if it is better than virtual box and (or) vmware player fo
virtualization.
Keep up the great work.
Eagerly looking for your next article.

△ | ▽ Reply Share ›

Reviews How-Tos Coding Interviews Features Overview Blogs

Search

Popular tags

Linux, ubuntu, Java, MySQL, Google, python, Android, Fedora, PHP, C, html, web applications, India, Microsoft, unix, Windows, Red Hat, Oracle, Security, Apache, xml, LFY April 2012, FOSS, GNOME, http, JavaScript, LFY June 2011, open source, RAM, operating systems

For You & Me
Developers
Sysadmins
Open Gurus
CXOs
Columns