Setting Up Virtual Machines with KVM

ir. Toon Macharis
# Table of Contents

1. Overview ......................................................................................................................... 1  
2. Host Setup ....................................................................................................................... 2  
3. Network Set Up ............................................................................................................... 14  
4. Installing KVM .................................................................................................................. 20  
5. Creating an LVM Partition ............................................................................................... 29  
6. Creating Logical Volumes with LVM ............................................................................... 35  
7. Virtual Machine Manager ................................................................................................. 38  
8. Bridged Networking ......................................................................................................... 68  
9. IP Forwarding .................................................................................................................... 90
Chapter 1. Overview

This book will explain how to set up a physical Debian/Ubuntu server and multiple virtual Debian Ubuntu servers using KVM. The explanation will occur mainly through screenshots of all the steps needed to set up multiple KVM virtual machines on a local Ubuntu desktop. The whole process can be emulated on a single PC, so everyone can try it out on his own laptop without having to risk doing something wrong on a remote server. Once you are familiar with the whole procedure, you can apply it on a remote server. Debian is a free operating system that can be downloaded from http://www.debian.org. Ubuntu is a free operating system that can be downloaded from http://www.ubuntu.com. I would recommend novice users to try out Ubuntu as a desktop system, whereas Debian is my preferred server operating system.

This book has 8 chapters:

- Chapter 2, Host Setup: this chapter explains how to disable the root login on a server for security reasons.
- Chapter 3, Network Set Up: this chapter explains how to configure IP addresses for a physical server.
- Chapter 4, Installing KVM: this chapter explains how to install KVM and bridged networking for KVM on a local or remote server.
- Chapter 5, Creating an LVM Partition: this chapter explains how to create an LVM partition that will be used to create logical disks for the virtual machine.
- Chapter 6, Creating Logical Volumes with LVM: this chapter explains how to create logical volumes with Logical Volume Manager (LVM). These logical volumes can be used as disks for virtual machines.
- Chapter 7, Virtual Machine Manager: this chapter explains how to install Virtual Machine Manager and how to install a virtual server on a remote server.
- Chapter 8, Bridged Networking: this chapter explains how to enable bridged networking for a virtual server, how to assign an IP address to a virtual server, how to disable the root login on the virtual server for security reasons and how to install a SSH server, so the virtual server can be accessed remotely.
- Chapter 9, IP Forwarding: this chapter explains how IP addresses that are assigned to a physical server can be assigned to virtual servers running on the physical server.
Chapter 2. Host Setup

This chapter explains how to disable the root login on a server for security reasons.

The default installation of Debian has an active root user account. The installation also sets up a regular user account. In the following example, the regular user account name is too. It is a bad practice to keep the root user account active and to let an SSH server listen to the default port 22, because the Internet is full of spiders that attempt brute force attacks. The more obvious or powerful a setting, the more likely they are used in brute force attacks. Most SSH attacks attempt to login at port 22, because this is the default port for SSH and the user name that is used the most in a log in attempt is root, since this user always exists and if he logs in, he has full control over the machine.

The example assumes no SSH server has been installed and we access the server locally. We will install an SSH server though, so we can access the server remotely. First log in with the following commands:

```
debian-root login: root
Password: password
```

Install sudo so we can give the regular user account administrative privileges, which will allow us to disable the root account:

```
debian-root:~# apt-get install sudo
```

An alternative and better way to install programs is to use aptitude instead of apt-get as in:

```
debian-root:~# aptitude install sudo
```

However, these are older screenshots in which I was still using apt-get, which is older than aptitude.
After installing `sudo`, enter:

debian-root:~# visudo

This will open the sudo configuration file in an editor.

The line:

```
root    ALL=(ALL) ALL
```

specifies that the root user has all permissions.
Add the following line:

```
    toon    ALL=(ALL) ALL
```

to give all permissions to the user `toon` also. Press `control-X` To quit the editor. Press `Y` to save the changes.

Press `enter` to confirm the changes:
Install an SSH server with the following command:

```bash
debian-root:~# apt-get install ssh
```

When prompted, press Y:
After the SSH server has been installed, log out:

debian-root:~# exit

We could now log in remotely to the server and proceed from the remote session. However in this example we just proceed with local access to the server.

Log in with the administrative user:

debian-root login: toon

Password: password
Go to the folder that contains the configuration of the SSH server:

toon@debian-root:~$ cd /etc/ssh/

List the files in the directory:

toon@debian-root:/etc/ssh$ ls

Open the SSH configuration file with an editor:

toon@debian-root:/etc/ssh$ sudo nano sshd_config

Note that only an administrator is allowed to modify the file sshd_config. The current user has been made an administrator because we have added the following to the sudo configuration file before:

```
toon    ALL=(ALL) ALL
```

Every time a user who is an administrator executes a command, he does not execute the command as an administrator by default. If a user wants to execute a command as an administrator, he has to explicitly type sudo before the command. The very first time a user executes a command with sudo he gets the 3 warnings that are shown in the screenshot. The first time a user executes a command with sudo in a session, he has to authenticate with his password, thus enter:

```
[sudo] password for toon: password
```
Here we see the configuration for the SSH server. If you have installed a X.Org server on the physical server and want to be able to tunnel the desktop remotely over SSH, make sure that you set the following setting:

**X11Forwarding yes**

The default port on which the SSH server is listening is 22:

**Port 22**
Host Setup

- Change this to another number between 1024-65535:

  **Port 2222**

- By default the *root* user can log in with SSH:

  PermitRootLogin yes
Disable the root user logging in with SSH:

PermitRootLogin no

Press control+X To quit the editor. Press Y to save the changes.

Press enter to confirm writing the changes.
Go to the `/etc/init.d/` directory, which contains start and stop scripts of several Linux services:

```
toont@debian-root:/etc/ssh$ cd /etc/init.d/
```

Look for the SSH script:

```
toont@debian-root:/etc/init.d$ ls s*
```

Restart the SSH server, so it uses the new SSH configuration:

```
toont@debian-root:/etc/init.d$ sudo /etc/init.d/ssh restart
```

Now we will not only prevent the `root` user from logging in through SSH, we will prevent the `root` user from logging in from anywhere. To disable the `root` user logging in, enter:

```
toont@debian-root:/etc/init.d$ sudo passwd -l root
```
or alternatively:

```bash
toon@debian-root:/etc/init.d$ sudo passwd --lock root
```

Try logging in as the `root` user:

```bash
toon@debian-root:/etc/init.d$ su
Password: password
```

The authentication fails. Now enable the log in of the `root` user again:

```bash
toon@debian-root:/etc/init.d$ sudo passwd -u root
```

or alternatively:

```bash
toon@debian-root:/etc/init.d$ sudo passwd --unlock root
```

Verify that the `root` user can log in again:

```bash
toon@debian-root:/etc/init.d$ su
Password: password
```

Log the `root` user out again:

```bash
debian-root:/etc/init.d# exit
```

Disable the `root` log in:

```bash
toon@debian-root:/etc/init.d$ sudo passwd -l root
```

Log out, we can access the server remotely over SSH, we do not need to be logged in locally:

```bash
toon@debian-root:/etc/init.d$ exit
```
Locally, we see the log in prompt:

```
Debian GNU/Linux 5.0 debian-root tty1
debian-root login: 
```
Chapter 3. Network Set Up

This chapter explains how to configure IP addresses for a physical server.

First we check which network interfaces are up and running. For this, we enter:

```
toon@ubuntu-toon:~$ sudo ifconfig
[sudo] password for toon: password
```

![Image of ifconfig output]

The network interfaces at my laptop are shown in the following screenshot. To add IP addresses to the eth0 interface, we will modify the `/etc/network/interfaces` file. For this, enter:

```
toon@little-ubuntu:~$ sudo nano /etc/network/interfaces
```
Initially, only the loopback interfaces is defined in `/etc/network/interfaces`:

```
auto lo
iface lo inet loopback
```

We assign the IP address 192.168.0.2 to eth0 by adding the following text (adjust where needed):

```
iface eth0 inet static
  address 192.168.0.2
  netmask 255.255.255.0
```

On another line, we add the route that forwards packets to the default gateway:

```
routes
  default via 192.168.0.1 dev eth0
```

The above configuration ensures that the host can communicate with the default gateway and receive an IP address from it.
auto eth0
iface eth0 inet static
    address 192.168.0.2
    network 192.168.0.0
    netmask 255.255.255.0
    gateway 192.168.0.1
# The DNS information below is copied to /etc/resolv.conf
dns-nameservers 8.8.8.8 8.8.4.4
dns-search pacita.org

We assign the IP address 192.168.0.3 to the same eth0 interface by adding the following text:

auto eth0:1
iface eth0:1 inet static
    address 192.168.0.3
    netmask 255.255.255.0

We add the IP addresses 192.168.0.4, 192.168.0.5 and 192.168.0.6 to the eth0 interface with the following text:

auto eth0:2
iface eth0:2 inet static
    address 192.168.0.4
    netmask 255.255.255.0

auto eth0:3
iface eth0:3 inet static
    address 192.168.0.5
    netmask 255.255.255.0

auto eth0:4
iface eth0:4 inet static
    address 192.168.0.6
    netmask 255.255.255.0
After making the changes to `/etc/network/interfaces`, press `control+X` and then `Y`.

Press `enter` to confirm the changes:

Now we have made the changes to the networking configuration, we will restart the networking for the changes to take effect.

**Warning**

Be careful before restarting the networking on a remote server. If the configuration is incorrect, you may not be able to access your server remotely anymore!

For the changes to take effect, enter:

```
toon@ubuntu-toon:~$ sudo /etc/init.d/networking restart
```
Now enter the following to check that IP addresses have been assigned to the network interface `eth0`:

```
toon@ubuntu-toon:~$ sudo ifconfig
```

The IP addresses 192.168.0.2, 192.168.0.3, 192.168.0.4, 192.168.0.5 and 192.168.0.6 have been assigned to the network interface `eth0`:
Test if the IP addresses work by trying to log in to the SSH server at one of the IP addresses. In the previous chapter, we let the SSH server listen at port 2222. Enter:

```
toon@ubuntu-toon:~$ ssh -p 2222 toon@192.168.0.2
Are you sure you want to continue connecting (yes/no)? yes
toon@192.168.0.2's password: password
```

We're logged in!
Chapter 4. Installing KVM

This chapter explains how to install KVM and bridged networking for KVM on a local or remote server. Much of the information about installing KVM can also be found at https://help.ubuntu.com/community/KVM

Execute the following command:

toon@ubuntu-toon:~$ egrep -c '(vmx|svm)' /proc/cpuinfo

If the console returns a number bigger than 0, all should be fine. 0 means that the CPU does not support hardware virtualization, a number bigger than 0 means that the CPU supports hardware virtualization. Execute the following command:

toon@ubuntu-toon:~$ egrep -c ' lm ' /proc/cpuinfo

If the console returns 0, it means the CPU is not 64 bit. If the console returns a number bigger than 0, it means the CPU is 64 bit. It is recommended, but not required that the CPU is 64 bit. Execute the following command:

toon@ubuntu-toon:~$ uname -m

If the console returns x86_64, you are running a 64 bit kernel. If the console returns one of i386, i486, i586 or i686, you are running a 32 bit kernel. It is recommended, but not required to run a 64 bit kernel.

If the settings above are fine, execute the following command to install KVM:

toon@ubuntu-toon:~$ sudo aptitude install qemu-kvm libvirt-bin bridge-utils

Press Y when prompted.

Check the groups the user is in by executing the command:

toon@ubuntu-toon:~$ groups
To use KVM, the user has to be in the group `libvirtd`. This group is created during installation of KVM, however the user is only added to this group after logging out and logging in again. Therefore, execute the command:

```
toorn@ubuntu-toon:~$ logout
```

Log in again with the following commands:

```
toorn@ubuntu-toon:~$ ssh -p 2222 toorn@192.168.0.2
Password: password
```

Now check the user has been added to the `libvirtd` group:

```
toorn@ubuntu-toon:~$ groups
```

We will use bridged networking, which would be the most efficient way to emulate physical ethernet cards at the virtual machines. For this, we have to install the `libcap2-bin` package:

```
toorn@ubuntu-toon:~$ sudo aptitude install libcap2-bin
```
Execute the following command to give the CAP_NET_ADMIN capability to KVM:

```
toon@ubuntu-toon:~$ sudo setcap cap_net_admin=ei /usr/bin/qemu-system-*
```

Execute the following command to modify the users that have the CAP_NET_ADMIN capability:

```
toon@ubuntu-toon:~$ sudo nano /etc/security/capability.conf
```

The contents of the file `/etc/security/capability.conf` look like:
Add the following line to give the user that runs KVM the inheritable CAP_NET_ADMIN capability:

```
cap_net_admin toon
```

**Note**

Give this capability only to the user that runs KVM. It is recommended to create a separate user, e.g. `kvmuser` whose only responsibility is to run KVM. In the example, I use my own user account, because I am the only active user at my laptop and it is easy to assign all permissions to myself. However, in a production environment with multiple physical users, the user that runs KVM should not be a user that is associated with a physical person.

If the line has been added, press `control+X` and then `Y` to save the changes.
Press **enter** to confirm the changes:

We will now modify the network configuration to use bridged networking. Execute:

```
toon@ubuntu-toon:~$ sudo nano /etc/network/interfaces
```
Installing KVM

Below you see the old configuration for the *eth0* interface:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
  address 192.168.0.2
  network 192.168.0.0
  netmask 255.255.255.0
  broadcast 192.168.0.255
  gateway 192.168.0.1

# The DNS information below is copied to /etc/resolv.conf
# dns-nameservers 8.8.8.8 8.8.4.4 # Google's public DNS servers
# dns-search pacita.org # domain is added to server name before resolution

auto eth0:1
iface eth0:1 inet static
  address 192.168.0.3
  netmask 255.255.255.0
```

Change the configuration for the *eth0* interface to:

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

Copy the old configuration from the *eth0* interface to the bridge network interface *br0*:

```
auto br0
iface br0 inet static
  address 192.168.0.2
  network 192.168.0.0
```
netmask 255.255.255.0
broadcast 192.168.0.255
gateway 192.168.0.1

# The DNS information below is copied to /etc/resolv.conf
dns-nameservers 8.8.8.8 8.8.4.4
dns-search pacita.org

and add the following lines to the br0 interface:

bridge_ports eth0  
bridge_stp off  
bridge_fd 0  
bridge_maxwait 0

Press control+X and then Y to save the changes.

Press enter to confirm the changes:
Restart the networking with the following command:

```
toon@ubuntu-toon:~$ sudo /etc/init.d/networking restart
```

**Warning**

Be careful before restarting the networking on a remote server. If the configuration is incorrect, you may not be able to access your server remotely anymore!

Log out:

```
toon@ubuntu-toon:~$ exit
```

Log in again to make sure the network is ok:

toon@ubuntu-toon:~$ ssh -p 2222 toon@192.168.0.2
   toon@192.168.0.2's password: password

And log out again:

toon@ubuntu-toon:~$ exit
Chapter 5. Creating an LVM Partition

This chapter explains how to create a Logical Volume Manager (LVM) partition that will be used to create logical disks for the virtual machine. There are 4 layers in an LVM partition:

1. The partition of type LVM.
2. A physical volume, which corresponds with an LVM partition.
3. A volume group, which comprises multiple physical volumes and multiple hard drives.
4. A logical volume, which is a part of a volume group and appears to be equivalent to a real hard drive.

The advantages of using LVM over regular partitions are that the volume group can span over multiple physical disks that can be added or removed on the fly, that the logical volumes can be easily resized and that a live snapshot can be taken of the logical volumes, so they can be backed up consistently without bringing the server down.

The first thing we do is checking the existing partitions with `fdisk`. Execute:

```
toon@ubuntu-toon:~$ sudo fdisk -l
[sudo] password for toon: password
```

The disk `/dev/sda` has an NTFS partition, an ext3 partition (Linux) and a swap partition and is fully used. The disk `/dev/sdb` has no partitions and is unused. If you would have no free space, you could use either `GParted` (if you install Xorg on your host server, which I don't) or `resize2fs` (if you do not install Xorg on your host server) to resize partitions and make some free space. Partitions can only be resized when they are not mounted. We assign new partitions to `/dev/sdb` by executing:

```
toon@ubuntu-toon:~$ sudo fdisk /dev/sdb
```
Creating an LVM Partition

Enter `m` to see all the available options:

Command (m for help): `m`

Enter `/` to see the possible partition types:

Command (m for help): `/`
Creating an LVM Partition

The Linux LVM partition type is identified by `8e`. Enter `n` to create a new partition:

```
Command (m for help): n
```

We have to decide whether to create a primary or an extended partition. There can only be 4 primary partitions per disk, so if we would create more than 4 partitions at the disk, we would have to use extended partitions. However, I will only create 1 single LVM partition that spans the whole disk, thus I choose for a primary partition:

```
Command action
```

Creating an LVM Partition

Since I only create 1 single partition, I can choose which number between 1 and 4 I choose. I choose 1:

```
Partition number (1-4): 1
```

The partition starts from the beginning of the disk, thus from cylinder 1:

```
First cylinder (1-60801, default 1): 1
```

The partition spans the whole disk until the last cylinder 60801:

```
Last cylinder, +cylinders or +size(K,M,G) (1-60801, default 60801): 60801
```

We set the type of the partition with the following command:

```
Command (m for help): t
```

We enter 8e, because that is the code for Linux LVM as we have seen before:

```
Hex code (type L to list codes): 8e
```

Enter p to print out the changes we have made to the partition table:

```
Command (m for help): p
```

```
Command (m for help): w
```

Execute the following command to check the changes to the partition table have been written correctly:

```
```
Creating an LVM Partition

```
toon@ubuntu-toon:~$ sudo fdisk -l
```

For the disk `/dev/sda`, we see the same partitions as before. For the disk `/dev/sdb`, we see the LVM partition we just created. We can also only list the partitions of disk `/dev/sdb` with the following command:

```
toon@ubuntu-toon:~$ sudo fdisk -l /dev/sdb
```

We only see the partitions of the disk `/dev/sdb`:
Creating an LVM Partition

<table>
<thead>
<tr>
<th>Device</th>
<th>Start</th>
<th>End</th>
<th>Blocks</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sdb1</td>
<td>1</td>
<td>60801</td>
<td>488364801</td>
<td>8e</td>
<td>Linux LVM</td>
</tr>
</tbody>
</table>

toont@ubuntu-toon:$ exit

Disk /dev/sdb: 500.1 GB, 50010762016 bytes
255 heads, 63 sectors/track, 60801 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xece3f2746

toont@ubuntu-toon:$ sudo fdisk -l /dev/sdb

Disk /dev/sdb: 500.1 GB, 50010762016 bytes
255 heads, 63 sectors/track, 60801 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xece3f2746
Chapter 6. Creating Logical Volumes with LVM

This chapter explains how to create logical volumes with Logical Volume Manager (LVM). These logical volumes can be used as disks for virtual machines. An LVM partition consists of 4 layers:

1. The partition of type LVM.
2. A physical volume, which corresponds with an LVM partition.
3. A volume group, which comprises multiple physical volumes and multiple hard drives.
4. A logical volume, which is a part of a volume group and appears to be equivalent to a real hard drive.

The previous chapter only explains how to create a partition of type LVM. This chapter explains how to create the other layers. The website http://www.howtoforge.com/linux_lvm explains LVM more in depth.

To create physical volumes, volume groups and logical volumes, we first have to install LVM2:

toon@ubuntu-toon:~$ sudo aptitude install lvm2
[sudo] password for toon: password

Next we check the available LVM partitions:

toon@ubuntu-toon:~$ sudo fdisk -l

/dev/sdb1 is the LVM partition we created in the previous chapter. From this partition, we make a physical volume with the command:

toon@ubuntu-toon:~$ sudo pvcreate /dev/sdb1

We check the existing physical volumes with the command:

---

The natural text is: 

Chapter 6. Creating Logical Volumes with LVM

This chapter explains how to create logical volumes with Logical Volume Manager (LVM). These logical volumes can be used as disks for virtual machines. An LVM partition consists of 4 layers:

1. The partition of type LVM.
2. A physical volume, which corresponds with an LVM partition.
3. A volume group, which comprises multiple physical volumes and multiple hard drives.
4. A logical volume, which is a part of a volume group and appears to be equivalent to a real hard drive.

The previous chapter only explains how to create a partition of type LVM. This chapter explains how to create the other layers. The website http://www.howtoforge.com/linux_lvm explains LVM more in depth.

To create physical volumes, volume groups and logical volumes, we first have to install LVM2:

toon@ubuntu-toon:~$ sudo aptitude install lvm2
[sudo] password for toon: password

Next we check the available LVM partitions:

toon@ubuntu-toon:~$ sudo fdisk -l

/dev/sdb1 is the LVM partition we created in the previous chapter. From this partition, we make a physical volume with the command:

toon@ubuntu-toon:~$ sudo pvcreate /dev/sdb1

We check the existing physical volumes with the command:
Creating Logical Volumes with LVM

toon@ubuntu-toon:~$ sudo pvdisplay

We see that /dev/sdb1 is a physical volume. We create the volume group *kvmgroup* from the physical volume /dev/sdb1 with the command:

toon@ubuntu-toon:~$ sudo vgcreate kvmgroup /dev/sdb1

We check the existing volume groups with the command:

toon@ubuntu-toon:~$ sudo vgdisplay

We see that *kvm0* is a volume group. We create the logical volume *kvm0* in the volume group *kvmgroup* with the command:

toon@ubuntu-toon:~$ sudo lvcreate -L 100G -n kvm0 kvmgroup
Creating Logical Volumes with LVM

```bash
toone@ubuntu-toone:~$ sudo lvcreate --name kvm0 --size 16G kvmgroup
```

The size of the logical volume is set to 16 GB.

We check the existing logical volumes with the command:

```bash
toone@ubuntu-toone:~$ sudo lvdisplay
```

We created the logical volume `/dev/kvmgroup/kvm0` with a size of 16 GB.
Chapter 7. Virtual Machine Manager

This chapter explains how to install Virtual Machine Manager and how to install a virtual server on a remote server. Virtual Machine Manager is a graphical user interface for KVM.

To install Virtual Machine Manager, execute the command:

```
toon@ubuntu-toon:~$ sudo aptitude install virt-manager
```

When prompted, press Y. Virtual Machine Manager has to be installed on your local PC, not on the server that hosts the virtual machines.

After installation, start Virtual Machine Manager from the command line with the command:

```
toon@ubuntu-toon:~$ virt-manager
```
You see the window below: Add a connection to the remote server by selecting from the menu: **File > Add Connection...**

Select **QEMU/KVM** as Hypervisor, **Remote tunnel over SSH** as Connection and enter **aaaa@bbbb:cccc** as Hostname, in which **aaaa** is the kvm user at the server, **bbbb** is the...
server hostname or IP address and cccc is the port at which the SSH server is listening. Click **Connect** to connect with the server.

When prompted, enter the password to log in. Once connected, right click on the connection, e.g. right click on `192.168.0.3:2222 (QEMU)` and select **Details**.
Select the Storage tab and click on the plus image in the bottom left corner.

Enter `kvmgroup`, the name of the volume group we created in the previous chapter in the Name field and select `logical: LVM Volume Group` for the Type field. Click Forward.
Set the Target Path to /dev/kvmgroup and leave the other fields blank. Click Finish.

We see the logical volume kvm0 that we created before appearing in the volume group kvm-group. We can also create logical volumes through the GUI. Click New Volume.
Enter `kvm1` in the Name field. Set Max Capacity and Allocation to 16000 MB. Click Finish.

We have now created the logical volume `kvm1` with a size of 16000 MB or 15.62 GB.
Switch back to the console. Check the directory you are currently in with the command:

```
toon@ubuntu-toon:~$ pwd
```

Create a subdirectory `kvm` and go to this subdirectory:

```
toon@ubuntu-toon:~$ mkdir kvm
```
```
toon@ubuntu-toon:~$ cd kvm
```

Download the latest Debian ISO image (probably different from the one shown in the screenshot) with `wget`:

```
toon@ubuntu-toon:~/kvm$ wget http://cdimage.debian.org/debian-cd/5.0.6/amd64/iso-cd/debian-506-amd64-netinst.iso
```
After downloading, check that the ISO image is present:

```
toon@ubuntu-toon:~/kvm$ ls
```

Switch back to the window that shows the storage at the server. Click on the plus image in the bottom left corner.
Enter *downloads* in the Name field and *dir: Filesystem Directory* in the Type field. Click *Forward*.

Enter the path to which we downloaded the Debian ISO, `/home/toon/kvm` in the Target Path field. Leave all the other fields blank and click *Finish*. 
We can now see the Debian ISO image.

Right click on the connection in the first window, e.g. right click on 192.168.0.3:2222 (QEMU) and select New. Enter Debian-Lenny in the Name field. Enter 192 (QEMU/KVM) in the Connection field. Select Local install media (ISO image or CDROM) and click Forward.
Set the OS type to Linux and the Version to Debian Lenny (or whichever that applies). Select Use ISO image and click Browse....
Select the *downloads* storage pool and select the Debian ISO image. Click *Choose Volume*.

![Virtual Machine Manager screen showing storage pool selection]

Click *Forward*.

![Virtual Machine Manager screen showing ISO image selection]

Set the RAM and CPUs of the virtual machine as desired and click *Forward*.
Select **Enable storage for this virtual machine.**

**Caution**

Do not select **Create a disk image on the computer’s hard drive.** If you select this option, the file system of the virtual machine will run on top of the file system of the host, which is not efficient at all. Instead select **Select managed or other existing storage.**

Click **Browse....**
Select the `kvmgroup` storage pool and select the logical volume `kvm0`. Click *Choose Volume*.

Click *Forward*. 
Leave the settings as they are and click Finish. The virtual machine will now start and boot from the ISO image. Click on the Open image of the first window to see the console of the virtual machine.
Select *Install* and press enter.

**Caution**

a graphical install is fancier than a regular install, but is also slower on a remote installation!
Select your language and press enter.

Select your country and press enter.
Select your region and press **enter**.

Select your country and press **enter**.
Select your keyboard layout and press enter.

Enter a host name for the server, select Continue and press enter.
Select a domain name for the server, select **Continue** and press **enter**.

Select *Guided - use entire disk* and press **enter**. *(we are already using LVM underneath.)*
Select the virtual disk and press enter.

Select the partitioning scheme you prefer. I prefer to select Separate /home partition, because then the user generated data is separated from the operating system files. In case of a back up you might only care about the /home partition, because the rest can be easily restored with a clean install. Press enter.
Select **Finish partitioning and write changes to disk** and press **enter**.

Select **Yes** and press **enter**.
Enter a root password (not password, except for demonstration purposes), select Continue and press enter.

Enter the same password, select Continue and press enter.
Enter your name (not mine), select *Continue* and press *enter*.

Enter a user name with which you will log in to the server, select *Continue* and press *enter*. 
Enter a password (not *password*, except for demonstration purposes) for the user name you just entered, select *Continue* and press *enter*.

Enter the same password, select *Continue* and press *enter*. 
Select the country from which you want to use the Debian mirror and press enter.

Select one of the available mirrors and press enter.
Fill in the proxy information, select **Continue** and press **enter**. The proxy information is to be left blank, unless you are behind a proxy.

Select whether you want to participate in the package usage survey and press **enter**.
Deselect all options, select **Continue** and press **enter**. I prefer to install as little as possible to my server and only add things when I need them. This would make a server more stable and safe, because programs that are not needed are not running and thus cannot be used to break into the server (if they would be listening to some ports).

Select **Yes** and press **enter**.
Select *Continue* and press enter.

After the server has been shut down, go to the info screen of the virtual machine, go to the CDROM and click *Disconnect*. 
Make sure that the Debian ISO image is not connected to the CDROM of the virtual machine anymore, so next time you start the virtual machine, you do not boot from the Debian ISO image (to reinstall Debian), but instead you boot from the virtual hard drive of the virtual machine.
Chapter 8. Bridged Networking

This chapter explains how to enable bridged networking for a virtual server, how to assign an IP address to a virtual server, how to disable the root login on the virtual server for security reasons and how to install an SSH server, so the virtual server can be accessed remotely.

The Virtual Machine Manager graphical user interface allows to modify most of the virtual machine settings. However, it appears not to be possible to set up a network card that uses bridged networking on a remote host. Therefore, we will have to set up the network card by manually editing the XML configuration file of the virtual machine. Make sure that the bridge network interface br0 has been set up as explained in Chapter 4, Installing KVM. We log in to the remote host from the command line with the following commands:

```
toon@ubuntu-toon:~$ ssh -p 2222 toon@192.168.0.3
toon@192.168.0.3's password: password
```

We go to the configuration directory of KVM with the following command:

```
toon@ubuntu-toon:~$ cd /etc/libvirt/qemu/
```

and see the configuration files of the virtual machines with:

```
toon@ubuntu-toon:/etc/libvirt/qemu$ ls
```

We edit the configuration file of the Debian-Lenny virtual machine with the following commands:

```
toon@ubuntu-toon:/etc/libvirt/qemu$ sudo nano Debian-Lenny.xml
[sudo] password for toon: password
```

Before editing the configuration file, make sure that the virtual machine has been turned off.

We see the original hardware that is attached to the virtual machine represented by XML elements. Network interfaces are defined by interface elements. The interface element that is already there represents the default network interface that uses NAT to access the internet through the host.
We add a network interface by adding the following text to the configuration:

```xml
<interface type='network'>
  <mac address='52:54:00:1d:c2:2a'/>
  <source bridge='br0'/>
  <model type='virtio'/>
</interface>
```

In this, the MAC address is unique and thus always different! You could add a network card to the virtual machine through Virtual Machine Manager. Only you don’t click Finish, but Cancel in the end, so the network card isn’t added. Yet, the UI generated a valid MAC address you could use while you were configuring the network card. The value for the bridge attribute is the name of the bridge network interface as set up in Chapter 4, Installing KVM. The value for the type attribute should be virtio (which has near native network performance). Once you added the new network interface, press control+X and then Y to save the changes:
Press **enter** to confirm the changes:

```
<disk>
  <interface type='network'>
    <mac address='52:54:00:8c:37:29'/>
    <source network='default'/>
    <model type='virtio'/>
  </interface>
  <interface type='bridge'>
    <mac address='52:54:00:1d:2c:2a'/>
    <source bridge='br0'/>
    <model type='virtio'/>
  </interface>
  <console type='pty'>
    <target port='0'/>
  </console>
  <console type='pty'>
    <target port='0'/>
  </console>
  <input type='mouse' bus='ps2'/>
  <graphics type='vnc' port='-1' autostart='yes'/>
</disk>
```

Reload the libvirt-bin service:

```
toon@ubuntu-toon:/etc/libvirt/qemu$ sudo service libvirt-bin reload
```

Close the command line prompt:

```
toon@ubuntu-toon:/etc/libvirt/qemu$ exit
toon@ubuntu-toon:~$ exit
```
Start a new command line prompt and start Virtual Machine Manager:

```
toon@ubuntu-toon:~$ virt-manager
```

Connect to the remote host. Enter your password when prompted. Select the Debian-Lenny virtual machine. Either double click on Debian-Lenny or on Open to open the window of the Debian-Lenny virtual machine:
Click on the play button to start the virtual machine:
Click the console button in the window of the Debian-Lenny virtual machine to see the console of the virtual machine. Once the login prompt appears, log in with the root user account (we continue the fresh Debian installation from Chapter 7, Virtual Machine Manager).

Lenny login: root
Password: password

Once logged in, we execute the command:

Lenny:~# ifconfig -a

This shows us all the network interfaces connected to the virtual machine, also the ones that are not configured and up yet. The network interface eth1 is not configured and up yet. We ran this command to find out the name of the network interface we manually added to the virtual machine by changing Debian-Lenny.xml above. This name is eth1, but it could as well have been eth2, eth3 or eth4. We need to know the name of the network interface to configure it in /etc/network/interfaces:

Lenny:~# nano /etc/network/interfaces
This is the original network configuration, only `eth0` and the loopback interface `lo` are configured:

```
Lenny:~# ifconfig -a

eth0  Link encap:Ethernet  HWaddr 52:54:08:6c:37:29
       inet addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:57 errors:0 dropped:0 overruns:0 frame:0
       TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:3690 (3.6 KiB)  TX bytes:726 (726.0 B)

eth1  Link encap:Ethernet  HWaddr 52:54:08:1d:c2:2a
       inet addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
       UP BROADCAST RUNNING 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:1000
       RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo    Link encap:Local Loopback
       inet addr:127.0.0.1  Mask:255.0.0.0
       UP LOOPBACK RUNNING  MTU:16436  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:0
       RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
Lenny:~# nano /etc/network/interfaces

auto lo
iface lo inet loopback
```

We assign the IP address 192.168.0.10 to the new network interface by adding the following text:

```
auto eth1
```

iface eth1 inet static
address 192.168.0.10
netmask 255.255.255.0

Press **control+X** and then **Y** to save the changes:

Press **enter** to confirm the changes:
Restart networking with the following command:

```bash
Lenny:~# /etc/init.d/networking restart
```

**Caution**

The networking of the virtual machine could be down if you did not configure it correctly!

Check all the network interfaces that are up and running with the following command:

```bash
Lenny:~# ifconfig
```

We see that the `eth1` network interface is up with the IP address that we assigned to it. However, the `eth0` network interface appears to be down. The `eth0` provided access to the Internet through the host. Currently we have no access to the Internet anymore. However, the `eth0` as well as the `eth1` network interface will be restored if we restart the server:

```bash
Lenny:~# reboot
```
After restarting the server, at the log in prompt, we enter again:

Lenny login: root
Password: password

We check again all network interfaces that are up and running:
After restarting the server, both the *eth0* and *eth1* network interface are up and running. The *eth0* interface provides access to the Internet (through the host by using NAT). The *eth1* interface allows the host to access the virtual machine at the IP address 192.168.0.10. We will now install *sudo*, so we can give administrative privileges to a regular user account, so we can desactivate the *root* user for security reasons:

```
Lenny:~# ifconfig
```

```
Setting kernel variables (/etc/sysctl.conf)... done.
Mounting local filesystems... done.
Activating swapfile swap... done.
Setting up networking....
Configuring network interfaces... done.
Setting console screen modes and fonts.
INIT: Entering runlevel: 2
Starting enhanced syslogd: Raylogd.
Starting ACPI services....
Starting periodic command scheduler: crond.

Debian GNU/Linux 5.0 Lenny tty1

Lenny login: root
Password:
Last login: Fri Nov 12 14:42:26 CET 2010 on tty1
Linux Lenny 2.6.28-2-xen-edd #1 SMP Thu Sep 16 15:56:30 UTC 2010 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software: the exact  
   version) in /usr/share/doc/*/copyright.  

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
   permitted by applicable law.
```

Lenny:~# aptitude install sudo
After installation, we configure `sudo` with the `visudo` command:

```
Lenny:~# visudo
```

Except for the root user, no one has administrative privileges by default:
We add the following line to give administrative rights to the user toon:

```
  toon    ALL=(ALL) ALL
```

Instead of giving individual users administrative rights, it may be better to give these rights to a group and add users that need to have these rights to the group. Add the following line instead to give administrative rights to the group admin:

```
%admin   ALL=(ALL) ALL
```

Press `control`+`X` and then `Y` to save the changes:
Press enter to confirm the changes:

Now log out the root user:

Lenny:~# exit

Log in with a regular user that has been given administrative privileges:
Bridged Networking

Lenny login: toon
Password: password

Disable the root login as soon as possible:

```bash
toon@Lenny:~$ sudo passwd -l root
```

The first time you execute `sudo`, you get the advice:

```
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.
```

Enter your password:

```
[sudo] password for toon: password
```

and install an SSH server:

```bash
toon@Lenny:~$ sudo aptitude install ssh
```
When prompted, press Y and confirm with enter:

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

1) Respect the privacy of others.
2) Think before you type.
3) With great power comes great responsibility.

Isudal password for toon:
Password changed.
toon@Lenny:~$ sudo aptitude install ssh

Configure the SSH server with:

toon@Lenny:~$ sudo nano /etc/ssh/sshd_config
Bridged Networking

First look for the following line:

**Port 22**

This line defines on which port the SSH server is listening.

Change the port on which the SSH server is listening to a number between 1024 - 65535, e.g. 2234:
Bridged Networking

You should not let the SSH server listen to port 22, the internet is full of spiders that attempt brute force attacks mainly on this port.

Find the line that allows root login:

PermitRootLogin yes
Change the line to:

```
PermitRootLogin no
```

You do not want a brute force attack succeeding in getting root access to your server. Press `control+X` and then `Y` to save the changes:

Press `enter` to confirm the changes:
Restart the SSH server with the following command:

```
toon@Lenny:~$ sudo /etc/init.d/ssh restart
```

Exit the console:

```
toon@Lenny:~$ exit
```

You will now be able to log in from the host with SSH.

```
From a console window, log in to the host:

```
toon@ubuntu-toon:~$ ssh -p 2222 toon@192.168.0.3
```

From the host, log in to the virtual machine guest:

```
toon@ubuntu-toon:~$ ssh -p 2234 toon@192.168.0.10
```

When prompted, enter `yes` and confirm with `enter`. Enter the password for the virtual machine:

```
toon@192.168.0.10's password: password
```
By mistake, I entered twice the password of the host, before entering the password of the guest, so I needed 3 attempts to log in. Check the hostname once you are logged in:

toon@Lenny:~$ hostname

The hostname is *Lenny*, which means that we are logged in on the virtual machine. Log out from the virtual machine:

toon@Lenny:~$ exit

Log out from the remote host:

toon@ubuntu-toon:~$ exit

And quit the command line:

toon@ubuntu-toon:~$ exit
Chapter 9. IP Forwarding

This chapter explains how IP addresses that are assigned to a physical server can be assigned to virtual servers running on the physical server.

To reroute Internet traffic that enters on an IP address of the host machine, to a private IP address of a virtual machine, one can use `iptables`. For this a rule has to be added to the PREROUTING chain of the nat table that has a DNAT target. I found a good explanation about the chains that are traversed by `iptables` at this website: http://www.faqs.org/docs/iptables/traversingoftables.html The traversal of the chains is also illustrated by the following image:
IP Forwarding

- ethernet in
  - PREROUTING mangle + nat
    - forward
      - no
      - yes
        - INPUT mangle + filter
          - FORWARD mangle + filter
            - local process
              - no
              - send out
                - yes
                  - OUTPUT mangle + nat + filter
          - yes
            - POSTROUTING mangle + nat
            - ethernet out
To forward the TCP traffic that enters at one specific port of an IP address to another port and IP address, one has to add a rule as:

```
toon@ubuntu-toon:~$ sudo iptables --table nat --append PREROUTING --protocol tcp --destination 192.168.0.4 --destination-port 1234 --jump DNAT --to-destination 192.168.0.10:2234
```

One could then access the SSH server which is listening to port 2234 at IP address 192.168.0.10 with the following command:

```
toon@ubuntu-toon:~$ ssh -p 1234 toon@192.168.0.4
```

However this can not be tested on a single PC only, because the iptables rules are only executed for Internet traffic that comes in or goes out through the network card. Network traffic with a local origin and destination does not pass the network card and therefore does not traverse the iptables chains. (I could test iptables from within the virtual servers, but the aim is to set it up at the host server, which would route Internet traffic to the virtual servers.) Forwarding the Internet traffic to another IP address in the PREROUTING nat table is not sufficient. The INPUT/FORWARD chains also have to accept the Internet traffic. If the default policy of the INPUT/FORWARD chains would be DROP, a rule has to be explicitly added to accept Internet traffic such as:

```
toon@ubuntu-toon:~$ sudo iptables --append FORWARD --protocol tcp --destination 192.168.0.10 --destination-port 2234 -j ACCEPT
```

or

```
toon@ubuntu-toon:~$ sudo iptables --append INPUT --protocol tcp --destination 192.168.0.10 --destination-port 2234 -j ACCEPT
```

![SSH Terminal Session](image)