By Falko Timme

Published: 2006-10-20 13:38

The Perfect Xen 3.0.3 Setup For Debian Sarge

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 10/20/2006

This tutorial provides step-by-step instructions on how to install Xen (version **3.0.3**) on a **Debian Sarge (3.1)** system.

Xen lets you create guest operating systems (*nix operating systems like Linux and FreeBSD), so called "virtual machines" or *domU*s, under a host operating system (*dom0*). Using Xen you can separate your applications into different virtual machines that are totally independent from each other (e.g. a virtual machine for a mail server, a virtual machine for a high-traffic web site, another virtual machine that serves your customers' web sites, a virtual machine for DNS, etc.), but still use the same hardware. This saves money, and what is even more important, it's more secure. If the virtual machine of your DNS server gets hacked, it has no effect on your other virtual machines. Plus, you can move virtual machines from one Xen server to the next one.

I will use Debian Sarge for both the host OS (*dom0*) and the guest OS (*domU*). I will describe how to install Xen from the sources (which I recommend) in **chapter 4** and from the binary package (**chapter 5**). In an additional section at the end of chapter 4 (**chapter 4.5**) I will also show how to create a virtual local network with virtual machines, with *dom0* being the router.

This howto is meant as a practical guide; it does not cover the theoretical backgrounds. They are treated in a lot of other documents in the web.

This document comes without warranty of any kind! I want to say that this is not the only way of setting up such a system. There are many ways of achieving this goal but this is the way I take. I do not issue any guarantee that this will work for you!

1 Install The Debian Sarge Host System (*dom0*)

You can overall follow these instructions, but with a few changes:

- **http://www.howtoforge.com/perfect_setup_debian_sarge**
- **http://www.howtoforge.com/perfect_setup_debian_sarge_p2**

However, it's important that you type `linux26` at the boot prompt to install a kernel 2.6 system. `dom0`'s FQDN in this example will be `server1.example.com`, so I specify `server1` as `Hostname` and `example.com` as `Domain name`. `server1.example.com`'s IP address will be `192.168.0.100` in this tutorial.

When it comes to the partitioning, I select `Manually edit partition table`. I create the following partitions:

- `/boot` 100 MB (Primary) (Location for the new partition: Beginning) (`ext3`) (**Bootable flag: on** <-- important, otherwise your system will not boot!)
- `swap` 1GB (Logical) (Location for the new partition: Beginning)
- `/` 2GB (Logical) (Location for the new partition: Beginning) (`ext3`)
- `/vserver` the rest (Logical) (Location for the new partition: Beginning) (`ext3`)

(**Side note:** You can also install everything in one big partition (as described here: **http://www.howtoforge.com/perfect_setup_debian_sarge**), but then you have to keep in mind that the `Grub` stanzas I describe in this howto are slightly different. For example, when I write that I add

```
[...]

title Xen 3.0.3 / XenLinux 2.6
root (hd0,0)
kernel /xen.gz  dom0_mem=64000
module /vmlinuz-2.6-xen root=/dev/hda6 ro max_loop=255
module /initrd.img-2.6.16.29-xen

[...]
```

to `/boot/grub/menu.lst` then you should probably use

```
[...]

title Xen 3.0.3 / XenLinux 2.6
root (hd0,0)
kernel /boot/xen.gz  dom0_mem=64000
module /boot/vmlinuz-2.6-xen root=/dev/hda6 ro max_loop=255
module /boot/initrd.img-2.6.16.29-xen

[...]
```

in that file instead...)

When the Debian installer prompts `Choose software to install:` I make no selection and go on (*dom0* should run as few software as possible in order not to be vulnerable to attacks. To the outside world it will be accessible only over SSH.).

2 Configure *dom0*'s Network

Because the Debian Sarge installer has configured our system to get its network settings via DHCP, we have to change that now because a server should have a static IP address. Edit ***/etc/network/interfaces*** and adjust it to your needs (in this example setup I will use the IP address *192.168.0.100*):

```
vi /etc/network/interfaces
```

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# The first network card - this entry was created during the Debian installation
# (network, broadcast and gateway are optional)
```

```
auto eth0
iface eth0 inet static
        address 192.168.0.100
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
```

Then restart your network:

```
/etc/init.d/networking restart
```

Edit *`/etc/resolv.conf`* and add some nameservers:

```
vi /etc/resolv.conf
```

```
search server
nameserver 145.253.2.75
nameserver 193.174.32.18
nameserver 194.25.0.60
```

Then set *`dom0`*'s hostname:

```
echo server1.example.com > /etc/hostname
```

```
/bin/hostname -F /etc/hostname
```

3 Install Xen

There are two ways to install Xen: compile Xen and the Xen kernels from the sources, or install the binary package from the Xen website.

The last way is easier, but it has the disadvantage that the *domU* kernel that comes with the binary package has no support for *quota* and *iptables*, both features that I need in my virtual machines (*domU*). Plus, the *dom0* kernel has no support for the *dummy* network driver, which might come in handy for more advanced network setups. Also, if you use the binary package, there is only one kernel for both *dom0* and *domU*, and the network setup for the virtual machines cannot be done using configuration files on *dom0*, but has to be done within the virtual machines themself which is rather complicated if you want to create new virtual machines from a pre-made image. I also got the impression that the source install is much more stable/mature. On the binary install it happened to me that a virtual machine didn't boot up because of a kernel panic, and two minutes later the same unchanged virtual machine did boot up without problems. So I highly recommend to compile Xen from the sources, although this takes much more time.

In ***chapter 4*** I describe how to compile and install Xen from the sources which is a must if you need quota and iptables in your virtual machines. In ***chapter 5*** I describe how to install the Xen binary package which might be easier for beginners.

4 Installing From The Sources

Run the following commands:

```
apt-get remove exim4 exim4-base lpr nfs-common portmap pidentd pcmcia-cs pppoe pppoeconf ppp pppconfig

apt-get install iproute bridge-utils python-twisted gcc-3.3 binutils make zlib1g-dev python-dev transfig bzip2 screen ssh debootstrap libcurl3-dev libncurses5-dev x-dev
```

4.1 Install Xen

Now we download *xen-3.0.3_0-src.tgz* from ***http://www.xensource.com/xen/downloads/dl_303tarballs.html*** and unpack it:

```
cd /usr/src

wget http://bits.xensource.com/oss-xen/release/3.0.3-0/src.tgz/xen-3.0.3_0-src.tgz

tar -xvzf xen-3.0.3_0-src.tgz
```

Then we compile Xen. This will create one Xen kernel (`2.6.16.29-xen`). We have to do this before we can create individual kernels for `dom0` and `domU`. This can take a long time so be patient:

```
cd xen-3.0.3_0-src/

make world

make install

mv /lib/tls /lib/tls.disabled
```

Now Xen is installed. In order to start the Xen services at boot time, do the following:

```
update-rc.d xend defaults 20 21

update-rc.d xendomains defaults 21 20
```

We need a ramdisk for our new Xen kernel, therefore we do the following:

```
depmod 2.6.16.29-xen
```

```
apt-get install libhtml-template-perl libparse-recdescent-perl
```

```
wget http://downloads.howtoforge.com/files/yaird_0.0.12-8bpo1_i386.deb

dpkg -i yaird_0.0.12-8bpo1_i386.deb
```

(The original yaird package was located in ***http://backports.org/debian/pool/main/y/yaird/***, but was removed in the meantime, so I've made the package available under ***http://downloads.howtoforge.com/files/yaird_0.0.12-8bpo1_i386.deb***.)

```
mkinitrd.yaird -o /boot/initrd.img-2.6.16.29-xen 2.6.16.29-xen
```

The last command creates the ramdisk `/boot/initrd.img-2.6.16.29-xen`.

Next we add our new kernel to `Grub`, our bootloader. Edit `/boot/grub/menu.lst`, and before the line `### BEGIN AUTOMAGIC KERNELS LIST` add the following stanza:

```
vi /boot/grub/menu.lst
```

```
[...]

title Xen 3.0.3 / XenLinux 2.6
root (hd0,0)
kernel /xen.gz  dom0_mem=64000
module /vmlinuz-2.6-xen root=/dev/hda6 ro max_loop=255
module /initrd.img-2.6.16.29-xen

[...]
```

Make sure that `/dev/hda6` is your `/` partition. Keep in mind what I said about `Grub` and partitioning in chapter 1! I added `max_loop=255` to the `module` line to make sure that enough loop devices are available because or virtual machines will be mounted as loop devices.

Now reboot the system:

```
shutdown -r now
```

At the boot prompt, `Grub` should now list `Xen 3.0.3 / XenLinux 2.6` as the first kernel and boot it automatically. If your system comes up without problems, then everything is fine!

4.2 Compile A dom0 Kernel

Now we compile a `dom0` kernel:

```
cd /usr/src/xen-3.0.3_0-src/

make linux-2.6-xen0-config CONFIGMODE=menuconfig KERNELS="linux-2.6-xen0"
```

In the kernel comfiguration menu that shows up we enable `quota`, `iptables` and the `dummy` network driver as ***modules***. This is where you enable these modules:

```
File systems --> [*] Quota support
<M> Old quota format support
<M> Quota format v2 support

Device Drivers ---> Network device support ---> <M> Dummy net driver support

Networking ---> Networking options ---> [*] Network packet filtering (replaces ipchains) ---> Core Netfilter Configuration
---> <M> Netfilter Xtables support (required for ip_tables)

Networking ---> Networking options ---> [*] Network packet filtering (replaces ipchains) ---> IP: Netfilter Configuration
---> <M> IP tables support (required for filtering/masq/NAT)
```

`[*]` means: build into the kernel statically.

`<M>` means: build as a kernel module.

Next we build and install the *dom0* kernel:

```
make linux-2.6-xen0-build

make linux-2.6-xen0-install

depmod 2.6.16.29-xen0
```

Next we add our new kernel to *Grub*, our bootloader. Edit */boot/grub/menu.lst*, and before the line *### BEGIN AUTOMAGIC KERNELS LIST* add the following stanza ( ):

```
vi /boot/grub/menu.lst
```

```
[...]

title Xen 3.0.3 / XenLinux 2.6
root (hd0,0)
kernel /xen.gz  dom0_mem=64000
module /vmlinuz-2.6-xen0 root=/dev/hda6 ro max_loop=255

[...]
```

Make sure that */dev/hda6* is your */* partition. Keep in mind what I said about *Grub* and partitioning in chapter 1!

Now reboot the system:

```
shutdown -r now
```

At the boot prompt, `Grub` should now list `Xen 3.0.3 / XenLinux 2.6` as the first kernel and boot it automatically. If your system comes up without problems, then everything is fine!

4.3 Compile A domU Kernel

Afterwards we compile a kernel for `domU` (the virtual machines):

```
cd /usr/src/xen-3.0.3_0-src/

make linux-2.6-xenU-config CONFIGMODE=menuconfig KERNELS="linux-2.6-xenU"
```

In the kernel comfiguration menu that shows up we have to enable `quota` and `iptables` as ***modules*** (it is ***important*** that they are ***modules***. I could not get `iptables` to work in a virtual machine when I compiled it into the kernel statically!). This is where you enable these modules:

```
File systems --> [*] Quota support
<M> Old quota format support
<M> Quota format v2 support

Networking ---> Networking options ---> [*] Network packet filtering (replaces ipchains) ---> Core Netfilter Configuration
---> <M> Netfilter Xtables support (required for ip_tables)

Networking ---> Networking options ---> [*] Network packet filtering (replaces ipchains) ---> IP: Netfilter Configuration
---> <M> IP tables support (required for filtering/masq/NAT)
```

`[*]` means: build into the kernel statically.
`<M>` means: build as a kernel module.

After you have left the kernel configuration menu, do the following to build and install the `domU` kernel:

```
make linux-2.6-xenU-build

make linux-2.6-xenU-install
```

```
depmod 2.6.16.29-xenU
```

4.4 Create A Virtual Machine (domU)

Next we create an image of a virtual machine. It will be a basic Debian system. This image will be the template for all our virtual machines. Whenever we want to create a new virtual machine, we just copy this image, create a new Xen configuration file and boot the copy, and then we can go on and configure the copy to our needs (e.g install a mail server, web server, DNS server, etc. on it). All our images will be on the `/vserver` partition which should be the largest one we have.

```
mkdir /vserver/vm_base

mkdir /vserver/images
```

Now we create a 1 GB image file and a 500 MB swap image. In the end the virtual machines will have 1 GB space and 500 MB swap. These are just example values, in the real world you might want to have more space for your virtual machines (e.g. between 5 and 30 GB), so just increase the value of `count` to create larger images.

```
dd if=/dev/zero of=/vserver/images/vm_base.img bs=1024k count=1000

dd if=/dev/zero of=/vserver/images/vm_base-swap.img bs=1024k count=500
```

Then we format `/vserver/images/vm_base.img` with `ext3` and `vm_base-swap.img` with `swap`:

```
mkfs.ext3 /vserver/images/vm_base.img
```

When you see the following, answer with *y*:

```
/vserver/images/mail.img is not a block special device.
  Proceed anyway? (y,n) <-- y
```

```
mkswap /vserver/images/vm_base-swap.img
```

4.4.1 Install A Basic Debian In The Image

In order to install a basic Debian system in our image, we mount the image, run *debootstrap* and a few other commands:

```
mount -o loop /vserver/images/vm_base.img /vserver/vm_base

debootstrap --arch i386 sarge /vserver/vm_base/ http://ftp2.de.debian.org/debian
```

```
chroot /vserver/vm_base

apt-setup
```

You are asked the following question:

Archive access method for apt: <--

Then select a mirror close to you.

Afterwards, edit */etc/apt/sources.list* and replace *testing* with *stable*. That's how my */etc/apt/sources.list* looks:

```
vi /etc/apt/sources.list
```

```
deb http://ftp2.de.debian.org/debian/ stable main
deb-src http://ftp2.de.debian.org/debian/ stable main

deb http://security.debian.org/ stable/updates main
```

Then run

```
apt-get update
```

Now we set up our `locales`. If we do not do this now, we will see some ugly warnings during `base-config` like these:

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_DE:en_US:en_GB:en",
        LC_ALL = (unset),
        LANG = "en_US"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
```

They are not serious, but ugly... So we run

```
apt-get install localeconf
```

You will be asked a few questions:

Select locales to be generated. <--
Which locale should be the default in the system environment? <--
Manage locale configuration files with debconf? <--
Environment settings that should override the default locale: <--
Replace existing locale configuration files? <--
Default system locale: <-- e.g.

Next run

```
base-config
```

You will see a menu with installation options. This is what we do:

- *Configure timezone*
- *Set up users and passwords*
- *Select and install packages* (when it comes to *Choose software to install:*, you can choose whatever you like; I, however, choose nothing because I want to install a basic system.)
- *Finish configuring the base system*

Don't deal with the other menu items, you don't need them. Then we remove *nfs-common* and delete */etc/hostname*:

```
apt-get remove nfs-common


rm -f /etc/hostname
```

Then edit */etc/fstab*. It should look like this:

```
vi /etc/fstab
```

```
/dev/hda1       /       ext3    defaults        1       2
```

```
/dev/hda2       none       swap   sw          0     0
/dev/pts        devpts     gid=5,mode=620     0     0
none            /dev/shm   tmpfs  defaults    0     0
```

Change `/etc/network/interfaces` to look like this:

```
vi /etc/network/interfaces
```

```
auto lo
iface lo inet loopback
        address 127.0.0.1
        netmask 255.0.0.0
```

Then create `/etc/hosts`:

```
vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain   localhost

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
ff02::3 ip6-allhosts
```

Then we edit the scripts */etc/init.d/hwclock.sh* and */etc/init.d/hwclockfirst.sh* and add the line *exit 0* right at the beginning because otherwise these two scripts will really slow down the bootup of our virtual machines:

```
vi /etc/init.d/hwclock.sh
```

```
#!/bin/sh
# hwclock.sh    Set and adjust the CMOS clock, according to the UTC
#               setting in /etc/default/rcS (see also rcS(5)).
#
# Version:      @(#)hwclock.sh  2.00  14-Dec-1998  miquels@cistron.nl
#
# Patches:
#               2000-01-30 Henrique M. Holschuh <hmh@rcm.org.br>
#                - Minor cosmetic changes in an attempt to help new
#                  users notice something IS changing their clocks
#                  during startup/shutdown.
#                - Added comments to alert users of hwclock issues
#                  and discourage tampering without proper doc reading.

# WARNING:      Please read /usr/share/doc/util-linux/README.Debian.hwclock
#               before changing this file. You risk serious clock
#               misbehaviour otherwise.

exit 0
[...]
```

```
vi /etc/init.d/hwclockfirst.sh
```

```
#!/bin/bash
# hwclockfirst.sh Set system clock to hardware clock, according to the UTC
#             setting in /etc/default/rcS (see also rcS(5)).
#
#
# WARNING:    Runs without write permission on /etc, and before
#             mounting all filesystems! If you need write permission
#             to do something, do it in hwclock.sh.
#
# WARNING:    If your hardware clock is not in UTC/GMT, this script
#             must know the local time zone. This information is
#             stored in /etc/localtime. This might be a problem if
#             your /etc/localtime is a symlink to something in
#             /usr/share/zoneinfo AND /usr isn't in the root
#             partition! The workaround is to define TZ either
#             in /etc/default/rcS, or in the proper place below.
#
# REMEMBER TO EDIT hwclock.sh AS WELL!

# Set this to any options you might need to give to hwclock, such
# as machine hardware clock type for Alphas.

exit 0

HWCLOCKPARS=
[...]
```

Now we leave the chroot environment:

```
exit
```

Then we copy over the kernel modules to our virtual machine image and unmount the image:

```
cp -dpR /lib/modules/2.6.16.29-xenU /vserver/vm_base/lib/modules/

mv /vserver/vm_base/lib/tls /vserver/vm_base/lib/tls.disabled

fuser -k /vserver/vm_base

umount /vserver/vm_base
```

Now our virtual machine image template is ready!

4.4.2 Create And Start The First Virtual Machine

Now we create our first virtual machine, *vm01*, by making a copy of our template:

```
cp -pf /vserver/images/vm_base.img /vserver/images/vm01.img

cp -pf /vserver/images/vm_base-swap.img /vserver/images/vm01-swap.img
```

Then we create a Xen configuration file for *vm01*, */etc/xen/vm01-config.sxp*:

```
vi /etc/xen/vm01-config.sxp
```

```
name="vm01"
kernel="/boot/vmlinuz-2.6-xenU"
```

```
root="/dev/hda1"
memory=32
disk=['file:/vserver/images/vm01.img,hda1,w','file:/vserver/images/vm01-swap.img,hda2,w']

# network
vif=[ '' ]
dhcp="off"
ip="192.168.0.101"
netmask="255.255.255.0"
gateway="192.168.0.1"
hostname="vm01.example.com"

extra="3"
```

In `memory` you specify the RAM you want to allocate to that virtual machine (here: `32 MB`). In `disk` you specify which images to use and how to mount them (i.e., under which partition, e.g. `hda1`). This ***must*** correspond to the settings in the image's `/etc/fstab` file! In the network settings we tell `vm01` that its IP address is `192.168.0.101` (the main machine's (`dom0`) IP address is `192.168.0.100`), and what `hostname` it has.

If you want `vm01` to start automatically at the next boot of the system, then do this:

```
ln -s /etc/xen/vm01-config.sxp /etc/xen/auto
```

Now let's start `vm01`:

```
xm create -c /etc/xen/vm01-config.sxp
```

If nothing's wrong, `vm01` should come up without problems, and you should be able to login. By running

```
iptables -L
```

you should see that *iptables* is available on *vm01*. To leave *vm01*'s shell, type *CTRL+]* if you are at the console, or *CTRL+5* if you're using PuTTY. From the outside you should be able to connect to *192.168.0.101* via *SSH*.

Back on *dom0*'s shell, you can shutdown *vm01* by running

```
xm shutdown vm01
```

Here are some other Xen commands:

xm create -c /path/to/config - Start a virtual machine.
xm shutdown <name> - Stop a virtual machine.
xm destroy <name> - Stop a virtual machine immediately without shutting it down. It's as if you switch off the power button.
xm list - List all running systems.
xm console <name> - Login on a virtual machine.
xm help - List of all commands.

Now you can reboot the main system to see if *vm01* comes up automatically (if you created the symlink in */etc/xen/auto*):

```
shutdown -r now
```

4.4.3 Creating And Customizing Further Virtual Machines

You can create further virtual machines simply by copying the image template:

```
cp -pf /vserver/images/vm_base.img /vserver/images/vm02.img

cp -pf /vserver/images/vm_base-swap.img /vserver/images/vm02-swap.img
```

Then you have to create a Xen configuration file, e.g. */etc/xen/vm02-config.sxp*:

```
vi /etc/xen/vm02-config.sxp
```

```
name="vm02"
kernel="/boot/vmlinuz-2.6-xenU"
root="/dev/hda1"
memory=32
disk=['file:/vserver/images/vm02.img,hda1,w','file:/vserver/images/vm02-swap.img,hda2,w']

# network
vif=[ '' ]
dhcp="off"
ip="192.168.0.102"
netmask="255.255.255.0"
gateway="192.168.0.1"
hostname="vm02.example.com"

extra="3"
```

Start the machine:

```
xm create -c /etc/xen/vm02-config.sxp
```

If you get an error like this:

```
Using config file "/etc/xen/vm02-config.sxp".
  Error: Error creating domain: The privileged domain did not balloon!
```

then this means that the virtual machine tried to use more memory than is available. Edit the configuration file of the virtual machine and decrease the value of `memory` and try to start it again.

Create a symlink, if you want to start the virtual machine at boot time:

```
ln -s /etc/xen/vm02-config.sxp /etc/xen/auto
```

Now you can log into each machine, e.g. via `SSH`, and configure it as if it was a normal system.

You can create as many virtual machines as you like. Your hardware's the limit!

4.5 Create A Virtual Local Network From The Virtual Machines (Optional)

(This chapter is optional. What is described here works only if you installed Xen from the sources.)

In this chapter I want to create a virtual network with my virtual machines, i.e. a network that is different from the network of `dom0`.

You can find a drawing of what I want to do here: ***http://wiki.xensource.com/xenwiki/XenNetworkingUsecase#head-7f23d0f2248cb0c70458f9339b4405e2b1bfc271***

I did the same with Xen 2.0.7 here: ***http://www.howtoforge.com/perfect_xen_setup_debian_ubuntu_p6***. However, the way to achieve this with Xen 3 has changed completely. Xen 3 configures all the firewall rules, gateways, etc. ***automatically***. Furthermore, we don't need any `dummy` network interface anymore for our virtual network. It is important to know that Xen 3 assigns gateways from the `10.x.x.x` net to our virtual machines, so it is a good idea to also assign IP addresses from the `10.x.x.x` net to our virtual machines. If you give them IP addresses from the `192.168.3.x` net (as we did with Xen 2.0.7 on ***http://www.howtoforge.com/perfect_xen_setup_debian_ubuntu_p6***), then your virtual machines will have no access to the internet.

So we will give `vm01` the IP address `10.0.0.1` and `vm02` the IP address `10.0.0.2`.

First we edit `/etc/xen/xend-config.sxp` and disable bridging and enable NAT (network address translation) instead:

```
vi /etc/xen/xend-config.sxp
```

```
[...]

#(network-script network-bridge)
#(vif-script vif-bridge)

(network-script network-nat)
(vif-script vif-nat)

[...]
```

Then we change the configuration files of `vm01` and `vm02`:

`/etc/xen/vm01-config.sxp`:

```
vi /etc/xen/vm01-config.sxp
```

```
name="vm01"
kernel="/boot/vmlinuz-2.6-xenU"
root="/dev/hda1"
memory=32
disk=['file:/vserver/images/vm01.img,hda1,w','file:/vserver/images/vm01-swap.img,hda2,w']

vif=[ 'ip=10.0.0.1' ]
dhcp="off"
hostname="vm01.example.com"
ip="10.0.0.1"
netmask="255.0.0.0"
gateway="10.0.0.254"
```

```
extra="3"
```

/etc/xen/vm02-config.sxp:

```
vi /etc/xen/vm02-config.sxp
```

```
name="vm02"
kernel="/boot/vmlinuz-2.6-xenU"
root="/dev/hda1"
memory=32
disk=['file:/vserver/images/vm02.img,hda1,w','file:/vserver/images/vm02-swap.img,hda2,w']

vif=[ 'ip=10.0.0.2' ]
dhcp="off"
ip="10.0.0.2"
netmask="255.0.0.0"
gateway="10.0.0.254"
hostname="vm02.example.com"

extra="3"
```

Afterwards shut down *vm01* and *vm02*:

```
xm shutdown vm01

xm shutdown vm02
```

Wait a few seconds and control with `xm list` that `vm01` and `vm02` have shut down. Then reboot the system:

```
shutdown -r now
```

If `vm01` and `vm02` aren't started automatically at boot time, start them now:

```
xm create /etc/xen/vm01-config.sxp

xm create /etc/xen/vm02-config.sxp
```

Now you should be able to ping `vm02` from `vm01` and vice versa, and you also be able to ping `dom0` and hosts on the internet!

Now let's assume we have a web server on port 80 in `vm01` and a mail server on port 25 in `vm02`. As they are in their own network (`10.x.x.x`), we cannot access them from the outside unless we forward these ports to the appropriate vm. We can create the necessary port forwarding rules on `dom0` with the help of `iptables`:

```
iptables -A PREROUTING -t nat -p tcp -i eth0 --dport 80 -j DNAT --to 10.0.0.1:80

iptables -A PREROUTING -t nat -p tcp -i eth0 --dport 25 -j DNAT --to 10.0.0.2:25
```

If we connect to `dom0` now on port 80, we are forwarded to `vm01`. The same goes for port 25 and `vm02`.

Of course, the forwarding rules are lost when we reboot `dom0`. Therefore we put the rules into `/etc/network/if-up.d/iptables`, which is executed automatically when the system boots:

```
vi /etc/network/if-up.d/iptables
```

```
#!/bin/sh
```

```
### Port Forwarding ###
iptables -A PREROUTING -t nat -p tcp -i eth0 --dport 80 -j DNAT --to 10.0.0.1:80
iptables -A PREROUTING -t nat -p tcp -i eth0 --dport 25 -j DNAT --to 10.0.0.2:25
```

Now we have to make that script executable:

```
chmod 755 /etc/network/if-up.d/iptables
```

Whenever you need additional port forwarding rules, execute them on `dom0`'s shell and then append them to `/etc/network/if-up.d/iptables` so that they are available even after a reboot.

5 Installing The Binary Package

Run the following commands:

```
apt-get remove exim4 exim4-base lpr nfs-common portmap pidentd pcmcia-cs pppoe pppoeconf ppp pppconfig

apt-get install screen ssh debootstrap python python2.3-twisted iproute bridge-utils libcurl3-dev
```

5.1 Install Xen

Then download `xen-3.0.3_0-install-x86_32.tgz` from http://www.xensource.com/xen/downloads/dl_303tarballs.html, unpack it, and run the install script:

```
cd /usr/src

wget http://bits.xensource.com/oss-xen/release/3.0.3-0/bin.tgz/xen-3.0.3_0-install-x86_32.tgz
```

```
tar xvzf xen-3.0.3_0-install-x86_32.tgz

cd dist/

./install.sh

mv /lib/tls /lib/tls.disabled
```

Now Xen is installed. In order to start the Xen services at boot time, do the following:

```
update-rc.d xend defaults 20 21

update-rc.d xendomains defaults 21 20
```

We need a ramdisk for our new Xen kernel, therefore we do the following:

```
depmod 2.6.16.29-xen
```

```
apt-get install libhtml-template-perl libparse-recdescent-perl
```

```
wget http://downloads.howtoforge.com/files/yaird_0.0.12-8bpo1_i386.deb

dpkg -i yaird_0.0.12-8bpo1_i386.deb
```

(The original yaird package was located in ***http://backports.org/debian/pool/main/y/yaird/***, but was removed in the meantime, so I've made the package available under ***http://downloads.howtoforge.com/files/yaird_0.0.12-8bpo1_i386.deb***.)

```
mkinitrd.yaird -o /boot/initrd.img-2.6.16.29-xen 2.6.16.29-xen
```

The last command creates the ramdisk `/boot/initrd.img-2.6.16.29-xen`.

Next we add our new kernel to `Grub`, our bootloader. Edit `/boot/grub/menu.lst`, and before the line `### BEGIN AUTOMAGIC KERNELS LIST` add the following stanza:

```
vi /boot/grub/menu.lst
```

```
[...]


title Xen 3.0.3 / XenLinux 2.6
root (hd0,0)
kernel /xen.gz  dom0_mem=64000
module /vmlinuz-2.6-xen root=/dev/hda6 ro max_loop=255
module /initrd.img-2.6.16.29-xen


[...]
```

Make sure that `/dev/hda6` is your `/` partition. Keep in mind what I said about `Grub` and partitioning in chapter 1! I added `max_loop=255` to the `module` line to make sure that enough loop devices are available because or virtual machines will be mounted as loop devices.

Now reboot the system:

```
shutdown -r now
```

At the boot prompt, `Grub` should now list `Xen 3.0.3 / XenLinux 2.6` as the first kernel and boot it automatically. If your system comes up without problems, then everything is fine!

5.2 Create A Virtual Machine (domU)

(Please note: image creation depends on whether you installed Xen from the sources or from the binaries. If you installed Xen from the sources, please refer to chapter 4.4!)

Next we create an image of a virtual machine. It will be a basic Debian system. This image will be the template for all our virtual machines. Whenever we want to create a new virtual machine, we just copy this image, create a new Xen configuration file and boot the copy, and then we can go on and configure the copy to our needs (e.g install a mail server, web server, DNS server, etc. on it). All our images will be on the `/vserver` partition which should be the largest one we have.

```
mkdir /vserver/vm_base

mkdir /vserver/images
```

Now we create a 1 GB image file and a 500 MB swap image. In the end the virtual machines will have 1 GB space and 500 MB swap. These are just example values, in the real world you might want to have more space for your virtual machines (e.g. between 5 and 30 GB), so just increase the value of `count` to create larger images.

```
dd if=/dev/zero of=/vserver/images/vm_base.img bs=1024k count=1000

dd if=/dev/zero of=/vserver/images/vm_base-swap.img bs=1024k count=500
```

Then we format `/vserver/images/vm_base.img` with `ext3` and `vm_base-swap.img` with `swap`:

```
mkfs.ext3 /vserver/images/vm_base.img
```

When you see the following, answer with `y`:

```
/vserver/images/mail.img is not a block special device.
  Proceed anyway? (y,n) <-- y
```

```
mkswap /vserver/images/vm_base-swap.img
```

5.2.1 Install A Basic Debian In The Image

In order to install a basic Debian system in our image, we mount the image, run *debootstrap* and a few other commands:

```
mount -o loop /vserver/images/vm_base.img /vserver/vm_base

debootstrap --arch i386 sarge /vserver/vm_base/ http://ftp2.de.debian.org/debian
```

```
chroot /vserver/vm_base

apt-setup
```

You are asked the following question:

Archive access method for apt: <--

Then select a mirror close to you.

Afterwards, edit */etc/apt/sources.list* and replace *testing* with *stable*. That's how my */etc/apt/sources.list* looks:

```
vi /etc/apt/sources.list
```

```
deb http://ftp2.de.debian.org/debian/ stable main
deb-src http://ftp2.de.debian.org/debian/ stable main

deb http://security.debian.org/ stable/updates main
```

Then run

```
apt-get update
```

Now we set up our `locales`. If we do not do this now, we will see some ugly warnings during `base-config` like these:

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_DE:en_US:en_GB:en",
        LC_ALL = (unset),
        LANG = "en_US"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
```

They are not serious, but ugly... So we run

```
apt-get install localeconf
```

You will be asked a few questions:

```
Select locales to be generated. <--
  Which locale should be the default in the system environment? <--
  Manage locale configuration files with debconf? <--
  Environment settings that should override the default locale: <--
  Replace existing locale configuration files? <--
  Default system locale: <-- e.g.
```

Next run

```
base-config
```

You will see a menu with installation options. This is what we do:
- `Configure timezone`
- `Set up users and passwords`
- `Select and install packages` (when it comes to `Choose software to install:`, you can choose whatever you like; I, however, choose nothing because I want to install a basic system.)
- `Finish configuring the base system`

Don't deal with the other menu items, you don't need them. Then we remove `nfs-common` and delete `/etc/hostname`:

```
apt-get remove nfs-common
```

Then edit `/etc/fstab`. It should look like this:

```
vi /etc/fstab
```

```
/dev/hda1       /           ext3    defaults        1   2
/dev/hda2       none        swap    sw              0   0
/dev/pts        devpts      gid=5,mode=620          0   0
none            /dev/shm    tmpfs   defaults        0   0
```

Then create `/etc/hosts`:

```
vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain   localhost

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Then do this:

```
mkdir /lib/modules/2.6.16.29-xen

depmod 2.6.16.29-xen
```

Next we edit the scripts */etc/init.d/hwclock.sh* and */etc/init.d/hwclockfirst.sh* and add the line *exit 0* right at the beginning because otherwise these two scripts will really slow down the bootup of our virtual machines:

```
vi /etc/init.d/hwclock.sh
```

```
#!/bin/sh
# hwclock.sh    Set and adjust the CMOS clock, according to the UTC
#               setting in /etc/default/rcS (see also rcS(5)).
#
```

```
# Version:      @(#)hwclock.sh  2.00  14-Dec-1998  miquels@cistron.nl
#
# Patches:
#               2000-01-30 Henrique M. Holschuh <hmh@rcm.org.br>
#                - Minor cosmetic changes in an attempt to help new
#                  users notice something IS changing their clocks
#                  during startup/shutdown.
#                - Added comments to alert users of hwclock issues
#                  and discourage tampering without proper doc reading.

# WARNING:      Please read /usr/share/doc/util-linux/README.Debian.hwclock
#               before changing this file. You risk serious clock
#               misbehaviour otherwise.

exit 0
[...]
```

```
vi /etc/init.d/hwclockfirst.sh
```

```
#!/bin/bash
# hwclockfirst.sh Set system clock to hardware clock, according to the UTC
#               setting in /etc/default/rcS (see also rcS(5)).
#
#
# WARNING:      Runs without write permission on /etc, and before
#               mounting all filesystems! If you need write permission
#               to do something, do it in hwclock.sh.
#
# WARNING:      If your hardware clock is not in UTC/GMT, this script
#               must know the local time zone. This information is
```

```
#		stored in /etc/localtime. This might be a problem if
#		your /etc/localtime is a symlink to something in
#		/usr/share/zoneinfo AND /usr isn't in the root
#		partition! The workaround is to define TZ either
#		in /etc/default/rcS, or in the proper place below.
#
# REMEMBER TO EDIT hwclock.sh AS WELL!

# Set this to any options you might need to give to hwclock, such
# as machine hardware clock type for Alphas.

exit 0

HWCLOCKPARS=
[...]
```

Now we leave the chroot environment:

```
exit
```

Then we unmount the image:

```
mv /vserver/vm_base/lib/tls /vserver/vm_base/lib/tls.disabled

fuser -k /vserver/vm_base

umount /vserver/vm_base
```

Now our virtual machine image template is ready!

5.2.2 Create And Start The First Virtual Machine

Now we create our first virtual machine, `vm01`, by making a copy of our template:

```
cp -pf /vserver/images/vm_base.img /vserver/images/vm01.img

cp -pf /vserver/images/vm_base-swap.img /vserver/images/vm01-swap.img
```

In the binary Xen install we cannot specify our virtual machine's hostname and network configuration in a Xen configuration file, we must specify these details directly in the virtual machine. Therefore we must mount the image now and edit a few files:

```
mount -o loop /vserver/images/vm01.img /vserver/vm_base

chroot /vserver/vm_base
```

The hostname of our first virtual machine is `vm01.example.com`, therefore we do this:

```
echo "vm01.example.com" > /etc/hostname
```

Then we edit `/etc/network/interfaces` and put in our network configuration (IP address `192.168.0.101`, gateway `192.168.0.1`):

```
vi /etc/network/interfaces
```

```
auto lo
iface lo inet loopback
        address 127.0.0.1
        netmask 255.0.0.0
```

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.0.101
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
```

Then we leave the chroot environment and unmount the image:

```
exit

umount /vserver/vm_base
```

Next we create a Xen configuration file for *vm01*, */etc/xen/vm01-config.sxp*:

```
vi /etc/xen/vm01-config.sxp
```

```
name="vm01"
kernel="/boot/vmlinuz-2.6-xen"
root="/dev/hda1"
memory=32
disk=['file:/vserver/images/vm01.img,hda1,w','file:/vserver/images/vm01-swap.img,hda2,w']
# network
vif=[ '' ]
extra="3"
```

In *memory* you specify the RAM you want to allocate to that virtual machine (here: *32 MB*). In *disk* you specify which images to use and how to mount them (i.e., under which partition, e.g. *hda1*). This ***must*** correspond to the settings in the image's */etc/fstab* file!

If you want *vm01* to start automatically at the next boot of the system, then do this:

```
ln -s /etc/xen/vm01-config.sxp /etc/xen/auto
```

Now let's start *vm01*:

```
xm create -c /etc/xen/vm01-config.sxp
```

If nothing's wrong, *vm01* should come up without problems, and you should be able to login. To leave *vm01*'s shell, type *CTRL+]* if you are at the console, or *CTRL+5* if you're using PuTTY. From the outside you should be able to connect to *192.168.0.101* via *SSH*.

Back on *dom0*'s shell, you can shutdown *vm01* by running

```
xm shutdown vm01
```

Here are some other Xen commands:

xm create -c /path/to/config - Start a virtual machine.
xm shutdown <name> - Stop a virtual machine.
xm destroy <name> - Stop a virtual machine immediately without shutting it down. It's as if you switch off the power button.
xm list - List all running systems.
xm console <name> - Login on a virtual machine.
xm help - List of all commands.

Now you can reboot the main system to see if *vm01* comes up automatically (if you created the symlink in */etc/xen/auto*):

```
shutdown -r now
```

5.2.3 Creating And Customizing Further Virtual Machines

You can create further virtual machines simply by copying the image template:

```
cp -pf /vserver/images/vm_base.img /vserver/images/vm02.img

cp -pf /vserver/images/vm_base-swap.img /vserver/images/vm02-swap.img
```

Again, we must specify our network configuration like this:

```
mount -o loop /vserver/images/vm02.img /vserver/vm_base

chroot /vserver/vm_base
```

Now our hostname is `vm02.example.com` for example, therefore we do this:

```
echo "vm02.example.com" > /etc/hostname
```

Then we edit `/etc/network/interfaces` and put in our network configuration (e.g. IP address `192.168.0.102`, gateway `192.168.0.1`):

```
vi /etc/network/interfaces
```

```
auto lo
iface lo inet loopback
        address 127.0.0.1
        netmask 255.0.0.0

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.0.102
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
```

Then we leave the chroot environment and unmount the image:

```
exit

umount /vserver/vm_base
```

Then you have to create a Xen configuration file, e.g. `/etc/xen/vm02-config.sxp`:

```
vi /etc/xen/vm02-config.sxp
```

```
name="vm02"
kernel="/boot/vmlinuz-2.6-xen"
root="/dev/hda1"
memory=32
disk=['file:/vserver/images/vm02.img,hda1,w','file:/vserver/images/vm02-swap.img,hda2,w']
# network
vif=[ " ]
```

```
extra="3"
```

Start the machine:

```
xm create -c /etc/xen/vm02-config.sxp
```

If you get an error like this:

```
Using config file "/etc/xen/vm02-config.sxp".
  Error: Error creating domain: The privileged domain did not balloon!
```

then this means that the virtual machine tried to use more memory than is available. Edit the configuration file of the virtual machine and decrease the value of `memory` and try to start it again.

Create a symlink, if you want to start the virtual machine at boot time:

```
ln -s /etc/xen/vm02-config.sxp /etc/xen/auto
```

Now you can log into each machine, e.g. via `SSH`, and configure it as if it was a normal system.

You can create as many virtual machines as you like. Your hardware's the limit!

6 Links

- Xen: ***http://www.xensource.com/xen/***
- Debian: ***http://www.debian.org/***