A Time Machine and File Server with RAID, Samba and Xen

Scot W. Stevenson <scot.stevenson@gmail.com>
First version: 7. July 2008
This version: 7. July 2008

About This Text

Storage we needed, storage was dear
Storage to allay our backupy fear
For our MacBooks a-two
And our videos many

So here is the story -
A machine that was made
With nothing but cheap parts
And Xen, Samba, RAID.

Use these notes for your pleasure -
But with no guarantee
With one wish I do send them:
Be they useful to thee!

About This Text, Slightly Less Poetic

These are my notes on how I set up a Ubuntu Linux server at home to provide Time Machine backup and home video storage to our MacBooks. It is a simple step-by-set list of what I did, with very little background on the why. I wrote this text for myself so it will easier to do it all over again if alien face-huggers burst through the floor and destroy the computer with their acid blood. I am publishing it on the Internet in case somebody else has a similar project in the works.

Most of these commands must be executed as root. I have tried to remember to insert a sudo in the examples, but probably forgot it someplace.

Disclaimer

Copyright and License

```
is available at GNU Copyleft.
```

Overview of the Setup

We need to provide storage on our Intranet for Time Machine backups from two MacBooks and for large amounts of material from home videos (this is only for slow backup – stuff in use is kept on a smaller Firewire drive directly attached to an iMac). We use Xen (http://www.xen.org) to virtualize the computer so I can install other operating systems ~~to fool around with~~ for research purposes. This gives us the following virtual machines:

– The home domain is called **hive**. We don't really need one, but Xen feels better if it can find at least one level beyond the machine names.

– The Domain 0 (dom0) virtual machine is named **core**. It can only be accessed through the console and has a most minimal setup: No ssh, no zero configuration, no mail, no printing, no nothing. It only has the mdadm software for RAID control and LVM for volume management. There is only a single user, named **alice**. Yes, "hive" and "alice" are *Resident Evil* references. No, the password is not "redqueen".

– The actual server is the first Domain U (domU) machine named **xen1**. It receives the drives from core and offers them to the net via Samba. Xen1 can be accessed via ssh and has zero configuration.

– There are plans to install two or more other domUs named **xen2** and **xen3**: Ubuntu Desktop and OpenSolaris. In my spare time (snicker).

The computer will be on 24/7.

The Hardware

Two 250 Gbyte SATA hard drives from two different manufacturers for the RAID 1 (mirrored) Time Machine storage. Some testing had shown that 100 Gbytes per person is enough for my wife and I (http://www.possum.in-berlin.de/texts/time_machine_disk_use.html).

Three 500 Gbyte SATA hard drives from three different manufacturers for the RAID 5 main ("bulk") storage. This is where the videos and non-Time Machine backups go. One SATA interface on the motherboard is left over for expansion or a hot spare.

The Time Machine and bulk storage parts are kept separate so that we can a) spin down the large drives which are very rarely needed instead of having them running constantly, which saves energy and wear; b) the smaller, cheaper drives are less expensive to replace when they go bye-bye. The drawback is that we give up some storage space.

One IDE 20 Gbyte hard drive. A leftover from earlier computers. Holds the operating system.

One LG GSA-4040B DVD/CD/DVD-RAM IDE drive/burner. Used for the installation and then later as DVD-RAM storage. For some reason, Apple doesn't support DVD-RAM.

Intel DG965WHMKR motherboard (http://www.intel.com/products/motherboard/DG965WH/index.htm) Chosen for the six SATA interfaces, it also has two old-style ATA interfaces for the operating system and a DVD/CD drive. Includes a built-in graphics chip and Gigabit Ethernet LAN. The board supports the Core 2 Quad Q9300 CPU (http://processorfinder.intel.com/Details.aspx?sSpec=SLAWE), which in turn has HVT technology for direct virtualization. However, it is not clear if the board itself supports HVT.

Intel Pentium Dual Core E2140, 1600 Mhz, 800 MHz FSB (http://www.intel.com/products/processor_number/chart/pentium_dual-core.htm). Inexpensive, with 64-bit (EM64T, note that Ubuntu calls this “amd64” even if it is an Intel processor). This is to be replaced with a Core 2 Quad Q9300 when finances allow.

Four Kingston ValueRAM 512 Mbyte DDR2-800 memory sticks. The Intel DG965WHMKR motherboard has four slots, but can only take a total of four Gbyte RAM with the fastest memory access. The 512 Mbyte sticks will be replaced by 1 Gbyte sticks over time.

Make sure the case is big and well ventilated enough for all the heat that the drives will produce.

Installing Ubuntu

Our processor supports EM64T, so we need the AMD64 server edition of Ubuntu (http://www.ubuntu.com/products/WhatIsUbuntu/serveredition). We use 8.04 “Hardy Heron” because it has Long Term Support (LTS), in this case until 2013.

The installation is standard; let the script put everything on the IDE drive by itself. Once the system is running, we do a first update:

```
sudo apt-get update
sudo apt-get upgrade
```

Install the time daemon and a *real* version of vim:

```
sudo apt-get install ntp
sudo apt-get install vim
```

Configure vim. Create a file .vimrc in the home folder with

```
set textwidth=75
set autoindent
set tabstop=4
set expandtab
set shiftwidth=4
filetype indent on
syntax enable
```

This is a setup for Python. We do not configure the rest of the system yet.

Installing Xen

Note that ubuntu-xen-server is the same package for i386 and amd64 for Hardy. There is a second package `ubuntu-xen-desktop` which includes just what you think it does, but was broken in early versions of Hardy and probably not what we want to be using anyway.

Before we install, stop apparmor:

```
sudo /etc/init.d/apparmor stop
sudo update-rc.d -f apparmor remove
```

Then:

```
sudo apt-get install ubuntu-xen-server
```

After installation, before reboot, there are some things to change. The following line was required earlier, but doesn't seem to be anymore:

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

Edit /etc/modules to make sure we have the line

```
loop max_loop=64
```

Reboot to make sure that everything works. Afterwards, uname -a should give us something like:

```
Linux core 2.6.24-19-xen #1 SMP Wed Jun 18 16:08:38 UTC 2008 x86_64
GNU/Linux
```

Now we make sure that /etc/hosts has the following:

```
127.0.0.1   localhost
127.0.1.1   core  core.hive
```

In /etc/dhcp3/dhclient.conf, insert a line at the beginning

```
send host-name "core";
```

We will have to do this for the guests, too.

Creating the RAIDs

Get the files:

```
sudo apt-get install mdadm
```

Format the drives with fdisk. Give them the type fd ("Linux RAID autodetect") and install one partition that spans the whole drive.

The RAID devices are set up with (replace the drive names with real ones)

```
mdadm --create /dev/md0 --level=raid1 --raid-devices=2 \
/dev/sdx1 /dev/sdy1

mdadm --create /dev/md1 --level=raid5 --raid-devices=3 \
/dev/sda1 /dev/sdb1 /dev/sdc1
```

where the first RAID contains the two smaller drives. Wait for resync by watching the progress in

```
cat /proc/mdstat
```

This will produce output in the form of:

```
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5]
[raid4] [raid10]
md0 : active raid1 sdd1[1] sdc1[0]
      244195904 blocks [2/2] [UU]
      [>....................]  resync =  2.0% (5115200/244195904)
finish=40.8min speed=97472K/sec
```

When it is finished, edit /etc/mdadm/mdadm.conf to look something like this:

```
DEVICE partitions
CREATE owner=root group=disk mode=0660 auto=yes
HOMEHOST core

# MAILADDR root
PROGRAM /home/alice/mdadm-warning-beeps

ARRAY /dev/md0 level=raid1 num-devices=2 \
UUID=98dc9bbf:a00ae1d0:82b0cb0c:c934d285
```

```
ARRAY /dev/md1 level=raid5 num-devices=3 \
UUID=0474df00:3479df39:82b0cb0c:c934d285
```

We got the last two device lines by running

```
sudo mdadm --examine --scan
```

and then include the output in the bottom part of the configuration file.

Note the MAILADDR and PROGRAM lines: For security reasons and because of the complexity, we don't have a mail program installed on this machine. To make sure that we detect a problem with the RAIDs, we use this little shell script, mdadm-warning-beeps:

```
while true
do
    echo Oh no! Problem detected with mdadm RAID array!
    cat /proc/mdstat
    echo "\a"
    sleep 10
done
```

This produces a warning beep every ten seconds and prints out some diagnostic information. Once you have this script in place and mdadm has been restarted, you can test this setup with

```
sudo mdadm --monitor ---test /dev/md0
```

It should produce beeps and the writing; you will have to kill the process by hand when you are done. Now, make sure the computer is someplace where you can hear it!

If there already are RAIDs present from a previous build, the configuration file might have been generated automatically during boot. However – reboot and see if everything is okay:

```
sudo mdadm --detail /dev/md0
```

This will give output such as

```
        Version : 00.90.03
  Creation Time : Sun Feb 30 19:03:27 2008
     Raid Level : raid1
     Array Size : 244195904 (232.88 GiB 250.06 GB)
  Used Dev Size : 244195904 (232.88 GiB 250.06 GB)
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Thu Apr  3 00:16:44 2008
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           UUID : 98dc9bbf:a00ae1d0:82b0cb0c:c934d285 (local to host core)
         Events : 0.14

    Number   Major   Minor   RaidDevice State
       0       8       49        0      active sync   /dev/sdd1
       1       8       33        1      active sync   /dev/sdc1
```

We can already format the RAID we will be using for Time Machine with

```
    sudo mkfs.ext3 -m 0 -T largefile /dev/md0
```

We use the "-m 0" option so there no space reserved for the superuser; this is a drive for data storage only. "-T largefile" optimized the file system for, uh, large files.

Setting the Spin-Down Time

There is a problem with the Linux software RAID and setting the spin-down time for the hard drives (see /usr/share/doc/hdparm/README.Debian.gz). There are probably clever ways of doing this, but since the computer is on 24/7 anyway, I just have a shell script that I run after each boot:

```
#!/bin/sh

echo Setting Time Machine RAID /dev/md0, 2 hours
tm_drives=`mdadm --detail /dev/md0 | grep "/dev/sd" | awk '{print $7}'`
echo Drives are: $tm_drives

for d in $tm_drives
do
    hdparm -S 244 $d
done

echo

echo Setting Storage RAID /dev/md1, 30 minutes
tm_drives=`mdadm --detail /dev/md1 | grep "/dev/sd" | awk '{print $7}'`
echo Drives are: $tm_drives

for d in $tm_drives
do
    hdparm -S 241 $d
done
```

This script lives in the home directory but is owned by root. Call it script with

```
sudo ./setraidspin
```

once you are sure that the RAIDs are up and running. The options for hdparm are a bit counter-intuitive, so if you want to change the time, make sure to read the man page carefully.

Installing LVM

We want to be as flexible as possible with the bulk storage on the RAID 5. We use the Linux Volume Manager (LVM) make this possible. We need to get

```
sudo apt-get install lvm2
```

We do not need lvm-common. Then, in step 1, create the **physical volume**:

```
sudo pvcreate /dev/md1
```

Check the sizes of the physical volume with

```
sudo pvdisplay /dev/md1
```

Step 2, create the **volume group** with the name "deep" (or whatever):

```
sudo vgcreate deep /dev/md1
```

We can see how much space we have allocated and free with

```
sudo vgdisplay **deep**
```

which gives us something like this:

```
    --- Volume group ---
    VG Name               deep
    System ID
    Format                lvm2
    Metadata Areas        1
    Metadata Sequence No  11
    VG Access             read/write
    VG Status             resizable
    MAX LV                0
    Cur LV                0
    Open LV               0
    Max PV                0
    Cur PV                1
    Act PV                1
    VG Size               931.52 GB
    PE Size               4.00 MB
    Total PE              238468
    Alloc PE / Size       0 / 0
    Free  PE / Size       238468 / 931.52 GB
    VG UUID               H53yTD-Ilv0-hqxp-6QtF-uCuY-BPLj-ot8oSd
```

For Xen, we don't have to create logical volumes by hand – what the virtual machines will think of as “their” hard drives – because the xen-create-image tool will do that for us. So we just take a large chunk of the volume group for the main storage:

```
sudo lvcreate --name storage --size 800G deep
```

That's leaving lots of space over for virtual machines, but who knows, maybe one day we will be installing Windows Vista (heh). Now we format the drive on this machine.

```
sudo mkfs.ext3 -m 0 -T largefile /dev/deep/storage
```

Again, the options make sure there is no space is reserved for the superuser and that it is optimized for large files.

We do not mount the drives and do not change /etc/fstab except to comment out the line to mount the CD/DVD/DVDRAM drive.

Creating the Server

It is time to create the virtual machine that will do the actual serving. The nice xen-create-image tool does all the heavy lifting for us. First, edit /etc/xen-tools/xen-tools.conf which contain the defaults.

```
dir            = /home/xen
install-method = debootstrap

size   = 20Gb     # Disk image size.
memory = 512Mb    # Memory size
swap   = 512Mb    # Swap size

fs     = ext3     # use the EXT3 filesystem for the disk image.
dist   = hardy    # Default distribution to install.
image  = full     # Specify sparse vs. full disk images.

dhcp = 1
```

```
passwd   = 1
accounts = 1

kernel      = /boot/vmlinuz-`uname -r`
initrd      = /boot/initrd.img-`uname -r`

mirror       = http://archive.ubuntu.com/ubuntu
cache        = 1
arch         = amd64
ext3_options = noatime,nodiratime,errors=remount-ro
disk_device  = xvda
```

Note these are the default parameters. We are going to override some of them for the server. Run the whole thing with

```
sudo xen-create-image --hostname=xen1 --memory=768Mb ---swap=1024Mb \
--lvm=deep
```

The reason that “swap” is larger than “memory” is because we will be adding more memory to the system later. This gives you:

```
General Information
--------------------
Hostname       :  xen1
Distribution   :  hardy
Partitions     :  swap            1024Mb (swap)
                  /               20Gb   (ext3)
Image type     :  full
Memory size    :  768Mb
Kernel path    :  /boot/vmlinuz-2.6.24-19-xen
Initrd path    :  /boot/initrd.img-2.6.24-19-xen

Networking Information
----------------------
IP Address     : DHCP [MAC: 00:16:3E:9A:E8:77]

Creating swap on /dev/deep/xen1-swap
Done

Creating ext3 filesystem on /dev/deep/xen1-disk
Done
```

There is also a log file in /var/log/xen-tools/xen1.log- And even more importantly, this produces a script in /etc/xen/ called xen1.cfg that looks something like

```
kernel      = '/boot/vmlinuz-2.6.24-19-xen'
ramdisk     = '/boot/initrd.img-2.6.24-19-xen'
memory      = '768'
root        = '/dev/xvda2 ro'
disk        = [
                  'phy:/dev/deep/xen1-swap,xvda1,w',
                  'phy:/dev/deep/xen1-disk,xvda2,w',
              ]
name        = 'xen1'
dhcp        = 'dhcp'
vif         = [ 'mac=00:16:3E:9A:E8:77' ]
on_poweroff = 'destroy'
on_reboot   = 'restart'
```

```
on_crash    = 'restart'
extra       = '2 console=xvc0'
```

We are going to add a few details. We want more than one virtual CPU, so we insert a

```
vcpus = '2'
```

Also, the server needs to have access to the storage files and DVDRAM, so we include the following lines in the “disk” entry:

```
'phy:/dev/deep/storage,xvda3,w',
'phy:/dev/md0/,xvda4,w',
'phy:/dev/scd0/,xvda5,w'
```

Before we start start the Domain, you can check to see what the parameters will look like with

```
sudo xm create xen1.cfg --dryrun
```

This gives you:

```
(vm
    (name xen1)
    (memory 768)
    (on_poweroff destroy)
    (on_reboot restart)
    (on_crash restart)
    (vcpus 2)
    (on_xend_start ignore)
    (on_xend_stop ignore)
    (image
        (linux
            (kernel /boot/vmlinuz-2.6.24-19-xen)
            (ramdisk /boot/initrd.img-2.6.24-19-xen)
            (ip :1.2.3.4::::eth0:dhcp)
            (root '/dev/xvda2 ro')
            (args '2 console=xvc0')
        )
    )
    (device (vbd (uname phy:/dev/deep/xen1-swap) (dev xvda1) (mode w)))
    (device (vbd (uname phy:/dev/deep/xen1-disk) (dev xvda2) (mode w)))
    (device (vbd (uname phy:/dev/deep/md0) (dev xvda4) (mode w)))
    (device (vbd (uname phy:/dev/scd0) (dev xvda5) (mode w)))
    (device (vif (mac 00:16:3E:9A:E8:77)))
)
```

If you think you got everything right, start it by

```
sudo xm create xen1.cfg -c
```

The -c at the end attaches the console to the domain. To stop that, press ctrl-] (which, by the way, is a real pain with a German keyboard). The server should start.

Configuring the Server

This is pretty much the same as we did for core, just more so. Start off with

```
sudo apt-get update
sudo apt-get upgrade
```

Install the real version of vim:

```
sudo apt-get install vim
```

Later, configure vim for the individual users. Add the names of our domain to /etc/hosts:

```
127.0.0.1   localhost
127.0.1.1   xen1  xen1.hive
```

Wait a moment. Users? What users? The install script only gave us root!

Creating Users

Add all the users we want via

```
adduser user1
adduser user2
```

and so forth. We want to get rid of root, so first we have to enable one user – "user1" in our example – to switch to root.

```
addgroup admin
adduser user1 admin
```

You should not log on with root anymore. Now we make a group that will let us share things:

```
addgroup house
adduser user1 house
adduser user2 house
```

We are going to use this for the DVD RAM.

Installing Zero Configuration

Now it is time to stop fooling around on the console. Install zero configuration (which Apple calls "Bonjour", http://www.apple.com/macosx/technology/bonjour.html), but Linux people call avahi with

```
sudo apt-get install avahi-daemon
```

Edit in /etc/avahi/avahi-daemon.conf:

```
host-name=xen1
use-ipv6=no
use-ipv4=yes
dissallow-other-stacks=yes
```

Reboot after we have done this, though we shouldn't have to, just to be on the safe side. We should now be able to access the computer via ssh and Bonjour from the Macs with

```
ssh xen1.local
```

Ssh was installed by default on xen1.

Creating the Mount Points

On the xen1, create the directories

```
/mnt/storage
/mnt/timemachine
/media/dvdram
```

In /etc/fstab on DomU xen1, include these lines:

```
# Storage Hard Drive
/dev/xvda3 /mnt/storage ext3 noatime,nodiratime,errors=remount-ro 0 1

# Time Machine Hard Drive
/dev/xvda4 /mnt/timemachine ext3 noatime,nodiratime,errors=remount-ro 0 1

# DVDRAM
/dev/xvda5    /media/dvdram       auto    defaults    0   0
```

Mount the file systems by hand so that we can add some directories:

```
sudo mount /mnt/timemachine
sudo mount /mnt/storage
sudo mount /media/dvdram
```

If the RAIDs were set up on an earlier system, make sure that the ownership is still correct. Then, each user gets his own directory for his Time Machine backup data:

```
sudo mkdir /mnt/timemachine/user1
sudo chown user1:user1 /mnt/timemachine/user1
```

Now decide on folders for the stuff you are going to put on the bulk storage. In our case, this is a single folder for video files:

```
sudo mkdir /mnt/storage/Video
```

The capital “V” with “Video” is so agrees with the name that Mac OS X sees through Samba.

On the DVD RAM, we set up a common area for all users.

```
sudo mkdir /media/dvdram/common
sudo chown user1:house /media/dvdram/common
```

Since the amount of space is limited, this is a pretty secure place to do this.

Setting up Samba

```
sudo apt-get install samba
```

Stop Samba with

```
sudo /etc/init.d/samba stop
```

and edit the configuration file /etc/samba/smb.conf to look something like this (things to be changed are set in bold). Note that the users have their access to the home directory commented out, because we don't want them logging onto this machine anyway.

```
[global]
   workgroup = WHATEVER
   server string = %h server
   dns proxy = no
   log file = /var/log/samba/log.%m
   max log size = 1000
   syslog = 0
   panic action = /usr/share/samba/panic-action %d

   security = user
   encrypt passwords = true
   passdb backend = tdbsam
   obey pam restrictions = yes

;  guest account = nobody
   invalid users = root
   unix password sync = yes
   passwd program = /usr/bin/passwd %u
   passwd chat = *Enter\snew\s*\spassword:* %n\n
*Retype\snew\s*\spassword:* %n\n *password\supdated\ssuccessfully* .
   pam password change = yes
   map to guest = bad user
   socket options = TCP_NODELAY

[homes]
   comment = Core Home
   browseable = no
   read only = no
   create mask = 0775
   valid users = %S

[Video]
   comment = Video Storage
   path = /mnt/storage/Video
   browsable = yes
   read only = no
   create mask = 0775
   guest ok = yes
   valid users = user1 user2

[Backup]
   comment = Core Time Machine
   path = /mnt/timemachine/%u
   browsable = yes
   read only = no
   valid users = user1 user2
```

In our case, we also have the DVD RAM drive, so we add another section for that:

```
[DVDRAM]
   comment = Core DVDRAM
   path = /media/dvdram/common
   browsable = yes
   read only = no
   create mask = 0775
   valid users = user1 user2
```

Now add the users who can access Samba, for example:

```
sudo smbpasswd -L -a user1
sudo smbpasswd -L -e user1
```

The first line adds the user to the password file, the second one enables him. If we have users who are allowed to use Samba, but whom we don't want to give a login to the system, change the shell in /etc/passwd to /bin/true to read:

```
user1:x:1000:1000:User One's Name,,,:/home/user1:/bin/true
```

Restart Samba with

```
sudo /etc/init.d/samba start
```

Since this computer is primarily a machine for Time Machine backups, that is the next step.

Setting up the Time Machine Server

First, a word of warning: This is hack, a *real* hack, and before you follow these instructions, you might want to re-read the disclaimer at the beginning of this text, especially the parts about losing all your data. This has worked for me, it has worked for a lot of other people, but it might not work for you. Apple does not support this *at all* – they suggest you buy a Time Capsule (http://www.apple.com/de/timecapsule/) for you backups – and they might change the rules with any given update. At the time of writing, this was for OS X 10.5.4.

What we do is trick the Macs into believing they are talking to an Apple server or a Time Capsule.

First, on the server, we go to the individual user's folder and create a hidden file that tells Time Machine it can use the drive for storage:

```
touch /mnt/timemachine/user1/.com.apple.timemachine.supported
```

Note the leading dot in the file name. Now, we set up Time Machine on the Mac. Tell your computer – we will give it the name *white* in this text – that it can use external drives that are not from Apple:

```
defaults write com.apple.systempreferences \
TMShowUnsupportedNetworkVolumes 1
```

You might have to include a sudo at the beginning of the line. Next, we mount the drive we exported via Samba with white's finder

```
Go --> Connect to Server --> smb://core.local/Backup
```

which should appear on the desktop. Remember this is actually the folder /mnt/timemachine/user1 on the virtual machine xen1.

Open the Time Machine preferences on white and use Change Disk to select Backup as our storage for Time Machine. The computer will start a countdown towards the next backup. Let it – it will write a sparse bundle disk image to Backup. All we need is the name of that file, which will be something like

```
White_001b6fbce56f.sparsebundle
```

Once we have the name, we can abort the backup. If Time Machine doesn't delete the sparse bundle from Backup by itself, we do so.

Open the Disk Utility application

```
Applications --> Utilities --> Disk Utility
```

and create New Image. The parameters are

```
Volume Name:       MyBackup
Volume Size:       Custom... (100 Gbyte)
Volume Format:     Mac OS Extended (Journaled)
Encryption:        none
Partitions:        no partition map
Image Format:      sparse bundle disk image
```

Don't worry if the 100 Gbyte is larger than the space left on your hard drive: The magic of sparse bundle disk images is that it grows with the amount of data you put in. Save it with any name you like, because we are going to change it anyway. OS X will automatically attach the drive. Unmount it again, change the name of the sparse bundle to the one that Time Machine created. Then copy the sparse bundle to the mounted backup drive.

Open the preferences for Time Machine again a start the backup. MyBackup should be mounted on the desktop. The first backup can take hours, so if you Apple is a MacBook, attach it to a power source and make sure nobody shuts the lid halfway through. That will really ruin your day, believe me.

Reboot the server. Make sure that Samba works. If it does, you're done. Party!

Notes on Booting

This setup requires various things to be done by hand at boot:

1. Make sure the RAIDs have come up right with cat /proc/mdstat
2. If yes, set the spin-down time with the setraidspin script
3. Start xen1, the domU that does the work, with sudo xm create xen1.cfg
4. Log off

If you reboot the machine very infrequently like me, you might want to write this list down someplace.

Future Versions of This Text

will include installing other domU types such as OpenSolaris and Ubuntu Desktop and how to get a graphical login.

Colophon

This text was written in NeoOffice 2.2.4, the Mac OS X version of OpenOffice. The HTML- and PDF-Versions were created by NeoOffice's export function.

Sources

LVM:

https://help.ubuntu.com/community/Installation/RAID1%2BLVM

Samba:

http://ubuntuforums.org/showthread.php?t=202605&highlight=samba+howto

Xen:

http://www.virtuatopia.com/index.php/Xen_Virtualization_Essentials
http://www.howtoforge.com/ubuntu-8.04-server-install-xen-from-ubuntu-repositories
http://www.howtoforge.org/high-performance-xen-on-ubuntu-8.04-amd64

Thanks

First thanks go to my uncle **Gary W. Marklund** who gave me the Unix bug when I was young.

Disclaimer and Copyright sections taken from **The Linux Documentation Project** (http://tldp.org/). **Rolf Heckemann** made me aware of the problems with spin-down settings and RAID array.

Revision History

Document started 26. June 2008. First published version 7. July 2008