Installing Xen Project Hypervisor on Debian 9

An in-depth beginner's guide

The purpose of this tutorial is to describe how to install and configure a Xen Project[1] hypervisor with control and guest domains using Debian[2] as the base operating system. Note that this tutorial uses Xen version 4.8.5-pre[3] as included in the current stable release of Debian 9.5 (stretch).

Hardware

Concerning hardware, a Lenovo ThinkCentre M83 SFF Pro desktop computer was used to test this tutorial with the following main components:

- Intel Core i5-4570 processor (4 cores),
- 8GB DDR3 memory,
- 120GB SSD,
- 500GB HDD, and
- Gigabit Ethernet port (with Internet connectivity).

Three-step process

Before we dive into the tutorial, here's a quick tip: it's helpful to think of the Xen system installation as a three-step process:

1. installing the Debian base operating system,
2. installing the Xen hypervisor[4] and control domain ('Dom0'[5]), and
3. installing guest domains ('DomU'[6]).

For the uninitiated, the term 'domain'[7] is used in Xen parlance to refer to a specific virtual machine[8] or instance.

Xen background

The Xen hypervisor runs directly on the computer hardware, as such it is a Type 1 or 'bare metal' hypervisor[9]. The control domain, Dom0, is a privileged virtual machine that runs on top of the hypervisor, from where the user may create and control unprivileged guest domains, or DomU. In a typical Xen setup, a DomU operating system only knows about itself and cannot directly control the hypervisor, control domain, or any other guest domains.

1 https://xenproject.org/
2 https://www.debian.org/
3 https://packages.debian.org/stretch/xen-system-amd64
4 https://wiki.xenproject.org/wiki/Hypervisor
5 https://wiki.xenproject.org/wiki/Dom0
6 https://wiki.xenproject.org/wiki/DomU
7 https://wiki.xenproject.org/wiki/Domain
8 https://wiki.xenproject.org/wiki/Virtual_Machine
9 https://en.wikipedia.org/wiki/Hypervisor

Concerning the hypervisor's relationship to Dom0 and DomU virtual machines, the most important points to remember are:

- Dom0 and DomU are virtual machines running on top of the Xen hypervisor,
- Dom0 is privileged and DomU is unprivileged, and
- domains are, therefore, separate entities from the hypervisor.

It's also good to know that the Xen installation process has been drastically simplified over the years through the hard work of dedicated individuals in the free open source software community. In fact, the installation process has been simplified to such an extent that many old Xen tutorials and books have you doing things that are either automatically configured today, or are no longer applicable in current kernel or operating system versions. Hopefully this tutorial helps to sort out, summarize, and explain some of the less-documented instructions, and also provides a bit more background and reference resources where necessary.

Alright, better get started! A detailed explanation of each installation step is provided below with links to additional reference material found throughout the tutorial.

Step 1 – Installing the Debian base operating system

During the Xen installation process, the Debian base operating system is used to install and configure the Xen hypervisor and Dom0 virtual machine only. After Xen is installed and configured, GRUB[10] is used to boot directly into the Xen hypervisor and Dom0 operating system where DomU can then be installed and configured.

For more information on how Xen's various pieces fit together, check out the Xen Project Software Overview[11] (sections 'What is the Xen Project Hypervisor?' and 'Introduction to Xen Project Architecture'), and Xen Project Beginner's Guide[12] (sections 'What is this Xen Project software all about?' and 'A brief look at Xen Project architecture').

1.1 Download the Debian .iso

For this tutorial, a *network install* or *netinst* image of the current stable release, stretch, can be downloaded from Debian's website, debian.org[13]. Choose the appropriate .iso for your computer's architecture.

Once downloaded, the ~300MB *debian-9.5.0-amd64-netinst.iso* image can be written to an appropriately sized USB thumb drive using *dd*, the GNU coreutils program. Note that there are many detailed references available online describing how to safely write an operating system image to a USB drive using *dd*. For my goto *dd* reference, refer to the instructions provided on raspberrypi.org[14]. Similar instructions may be found on debian.org[15].

CAUTION! If dd is run incorrectly it can wipe out your entire HDD – ensure that you have properly identified your USB thumb drive as detailed below, and that you have removed all files from the USB drive that you wish to keep.

1.2 Determine the USB drive name and unmount the drive

Before running the *dd* command, identify the device and partition name(s) on the USB drive by running the *lsblk* command. Run the command first with the USB drive removed, then once again with the drive plugged in. The name that appears when the USB device is plugged in – for example, 'sdX' – is, therefore, the USB drive.

```
$ lsblk
```

Once the USB drive has been positively identified as 'sdX', with a single partition of 'sdX1', unmount the partition by running the *umount* command as root:

```
# umount /dev/sdX1
```

[N.B. Substitute your USB drive's partition instead of 'sdX1'.]

10 https://www.gnu.org/software/grub/
11 https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview
12 https://wiki.xenproject.org/wiki/Xen_Project_Beginners_Guide
13 https://www.debian.org/CD/netinst/
14 https://www.raspberrypi.org/documentation/installation/installing-images/linux.md
15 https://www.debian.org/CD/faq/#write-usb

Note that your device may have more than one partition (e.g., sdX1 and sdX2). Run the *umount* command to unmount all mounted partitions on your USB drive.

1.3 Copy the .iso file to the USB drive using *dd*

dd can then be used to write the .iso image to the unmounted USB drive by running the following command as root:

```
# dd bs=4M if=debian-9.5.0-amd64-netinst.iso of=/dev/sdX
```

[N.B. Substitute your USB drive's name instead of 'sdX'. Note that the command uses the device name not partition. Also, the name of the .iso file you download should follow the 'if=' statement.]

Finally, run the *sync* command to ensure that it's safe to remove the USB drive.

```
$ sync
```

1.4 Run Debian installer and partition the primary disk drive

Once the USB drive is loaded with the .iso, boot the computer with the USB drive plugged in to initiate the Debian installer. Make sure the computer's BIOS is configured to boot from the USB drive ahead of the system's disk drive(s).

Follow the Debian installer's prompts setting up networking and user accounts as necessary. When the installer's partition configuration section is reached choose the 'manual' option, and setup the primary disk drive partitions as follows:

- 4GB primary partition, filesystem type 'ext4', mark as 'bootable', mount on '/'
- 1GB swap partition
- with the remaining disk space, create a Volume Group partition with the name *vg0* (do not create a Logical Volume, and do not create a mount point)

[N.B. The primary disk on the test computer, 'sda', is a 120GB SSD.]

For more information on how to use the Debian installer for partition configuration, refer to the Debian GNU/Linux Installation Guide[16] or chapter '4.2.13.2 Manual Partitioning' in the Debian Administrator's Handbook[17].

To give context to the partition choices, above, note that the minimal install of Debian 9.5 – with SSH server and standard system utilities installed, and no desktop environment – used 836MB of the 4GB primary partition, 'sda1', on the test computer. Even with packages *xen-system-amd64* and *xen-tools* installed, which we'll be using later on, primary partition usage came in at only 1.1GB.

Similarly, after booting into the Debian base operating system, the *free* command registered only 61MB of memory usage. This is informative for determining the swap partition size. For most Xen system setups, the hypervisor and Dom0 do not require large amounts of disk space or memory, so create your partitions with this in mind.

16 https://www.debian.org/releases/stable/amd64/ch06s03.html.en#di-partition
17 https://debian-handbook.info/browse/stable/sect.installation-steps.html

As we have no need for a Xen hypervisor/Dom0 desktop environment, none was installed; however, SSH server and standard system utilities were installed by default.

Let the Debian installer finish and boot into the newly installed Debian base operating system.

[N.B. The secondary disk drive on the test computer, a 500GB HDD, was left untouched during the Debian installer process as it was previously formatted with a single ext4 filesystem partition, ‘sdb1’. Step 3, below, will illustrate how to provide access to and mount this drive in a DomU.]

Step 2 – Installing the Xen hypervisor and control domain (‘Dom0’)

The next step consists of installing the Xen hypervisor, and configuring the Dom0 virtual machine through the Debian base operating system. Once all preliminary configuration is complete, we will boot into the Xen hypervisor/Dom0 virtual machine to explore the new system.

Note that all code provided in Step 2 may be executed (i) via a SSH session, or (ii) with a monitor connected to the Debian base operating system computer. However, once Xen is installed, the computer will be configured to boot directly into the Xen hypervisor/Dom0: Xen will be the GRUB default. In that case, if you would like to get back into the Debian base operating system once Xen is installed, connect a monitor to view and adjust the boot selection on the GRUB menu.

[N.B. If a monitor is not easily assessable, you can temporarily change the GRUB default to the Debian base operating system, then change it back to the Xen hypervisor/Dom0 when you’re done. One way to do this is by changing the name of the GRUB/Xen configuration file from ‘/etc/default/grub.d/xen.cfg’ to ‘/etc/default/grub.d/xen.cfg.disabled’, then run *update-grub*. When you’re ready to have Xen as the GRUB default again, just rename the file ‘xen.cfg’. Remember to run the command *update-grub* after each configuration change otherwise it won’t take effect!]

2.1 Verify the system’s configuration

Verify the base operating system’s partition configuration by running the command *lsblk*:

```
$ lsblk
```

If the tutorial’s partition scheme for the base operating system was followed, the command will generate the following output:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	111.8G	0	disk	
sda1	8:1	0	3.7G	0	part	/
sda2	8:2	0	954M	0	part	[SWAP]
sda3	8:3	0	1K	0	part	
sda5	8:5	0	107.1G	0	part	
sdb	8:16	0	465.8G	0	disk	
sdb1	8:17	0	465.8G	0	part	

The example output, above, includes the following configurations on the primary disk, ‘sda’:

- a 4GB ‘sda1’ partition with mount point ‘/’, showing up as 3.7G on the test computer,
- a 1GB ‘sda2’ swap partition with mount point ‘[swap]’, showing up as 954M on the test computer, and
- the remaining disk space of 107.1G on ‘sda5’ with no mount point.

To get more information on the Volume Group partition, ‘sda5’, created during the Debian installation process, run the *vgs* command as root:

```
# vgs
```

This generates the following output on the test computer:

VG	#PV	#LV	#SN	Attr	Vsize	VFree
vg0	1	0	0	wz--n-	107.13g	107.13g

If there is a Volume Group configured – as there should be – then its Volume Group name (e.g., ‘vg0’) will be listed under the column header ‘VG’.

If something went wrong, lookup the commands *pvcreate* and *vgcreate* in the man pages, and review the section ‘Add LVM (Logical Volume Manager)’ in the xmodulo.com Xen tutorial[18]. To create the LVM Volume Group, run the following two commands as root substituting ‘sdaX’ with your desired Volume Group partition:

```
# pvcreate /dev/sdaX

# vgcreate vg0 /dev/sdaX
```

Now verify that the Volume Group was properly configured by running *vgs* as root, as previously described above.

[N.B. The Volume Group is required in Step 3 to create partitions for the DomU virtual machine. Note that the recommended method for allocating disk and swap partitions to virtual machines in Xen is through the creation of Logical Volumes in a Volume Group.]

2.2 Install Xen hypervisor

Use *apt-get* to update the Debian base operating system package index files, and upgrade all currently installed packages. As root run the following command:

```
# apt-get update && apt-get upgrade
```

Next, use *apt-get* to install the Xen meta-package[19]. Run the following command as root, adjusting the architecture suffix to suit your hardware:

```
# apt-get install xen-system-amd64
```

[N.B. The Debian stretch repository provides three architecture options: amd64, arm64, and armhf. As the test computer has an Intel Core i5, this tutorial uses amd64 as an example.]

2.3 Verify GRUB’s menu default

Installing the *xen-system-amd64* package should automatically update the GRUB menu default; however, it is good form to take a moment and verify the configuration.

View the GRUB/Xen configuration file ‘/etc/default/grub.d/xen.cfg’ using *less* by running the following command:

```
$ less /etc/default/grub.d/xen.cfg
```

18 http://xmodulo.com/install-xen-hypervisor.html
19 https://packages.debian.org/stretch/xen-system-amd64

Look for the comments and code located at the end of the file that match the following:

```
# Make booting into Xen the default if not changed above. Finding the
# current string for it always has been a problem.
#
if [ "$XEN_OVERRIDE_GRUB_DEFAULT" = "" ]; then
	echo "WARNING: GRUB_DEFAULT changed to boot into Xen by default!"
	echo "	Edit /etc/default/grub.d/xen.cfg to avoid this warning."
	XEN_OVERRIDE_GRUB_DEFAULT=1
fi
if [ "$XEN_OVERRIDE_GRUB_DEFAULT" = "1" ]; then
	GRUB_DEFAULT="Debian GNU/Linux, with Xen hypervisor"
fi
```

At the top of the same file the following line of code should be, by default, commented out:

```
#XEN_OVERRIDE_GRUB_DEFAULT=0
```

Note that the above code configures GRUB to boot into the Xen hypervisor/Dom0 virtual machine by default. If this code or configuration is missing from your system, copy the example code provided above into your '/etc/default/grub.d/xen.cfg' file. For an alternative configuration method, refer to the section 'Prioritise Booting Xen Over Native' in the debian.org Xen tutorial[20].

2.4 Bridge the network interface

The easiest way to provide DomU access to the computer's Ethernet device is through a Linux Ethernet bridge to Dom0. In this tutorial, the Debian package *bridge-utils*[21] is used to configure the bridge. The *bridge-utils* package should have been installed automatically when the *xen-system-amd64* meta-package was installed.

Verify that *bridge-utils* is installed in the operating system by running the command *dpkg -l* [lower-case L], and piping the output through *grep*:

```
$ dpkg -l | grep bridge-utils
```

If the program is installed, the output should look something like this:

```
ii  bridge-utils    1.5-13+deb9u1    amd64    Utilities for configuring the Linux Ethernet bridge
```

Once it is confirmed that *bridge-utils* is installed, it's time to configure the system's network settings. For all the steps below, it's good form to make backup copies of the original configuration files before modifying them.

As root, make a copy of the file '/etc/network/interfaces':

```
# cp /etc/network/interfaces /etc/network/interfaces.backup
```

Now edit the 'interfaces' file using the *nano* file editor:

```
# nano /etc/network/interfaces
```

20 https://wiki.debian.org/Xen
21 https://packages.debian.org/stretch/bridge-utils

The interfaces file on the test computer contains the following lines by default:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eno1
iface eno1 inet dhcp
```

Note that the primary network interface in your computer may have a different name than ‘eno1’. Always use the name found in your computer’s ‘interfaces’ file. In other words, don’t use ‘eno1’ unless it’s the name found on your system.

Edit the ‘interfaces’ file to include the following modifications and new lines making sure to substitute ‘eno1’ with your system’s interface name:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eno1
iface eno1 inet manual

# Setup Xen interface bridge
auto xenbr0
iface xenbr0 inet dhcp
   bridge_ports eno1
```

[N.B. In *nano*, use ‘CTRL+ O’ to save, and ‘CRTL + X’ to exit.]

It is important to set the primary network interface to ‘manual’ and the bridge to ‘dhcp’, if you want to have your networking configured via DHCP. Also, note that ‘xenbr0’ is the standard name used for the Linux Ethernet bridge in Xen installations.

Once the ‘interfaces’ file is properly configured and saved, restart the systemd networking unit with the following command, executed as root:

```
# systemctl restart networking
```

Then check that the bridge is properly configured by running the *ip addr* command, looking for mention of ‘xenbr0’:

```
$ ip addr
```

Running this command on the test computer produces the following output:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master xenbr0 state UP group default qlen 1000
   link/ether 44:39:c4:35:29:e7 brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.140/24 brd 192.168.1.255 scope global eno1
      valid_lft forever preferred_lft forever
3: xenbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 44:39:c4:35:29:e7 brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.140/24 brd 192.168.1.255 scope global xenbr0
      valid_lft forever preferred_lft forever
   inet6 fe80::4639:c4ff:fe35:29e7/64 scope link
      valid_lft forever preferred_lft forever
```

You can also check that the bridge and Ethernet device are properly associated by running the command *brctl show* as root:

```
# brctl show
```

Running this command on the test computer produces the following output:

bridge name	bridge id	STP enabled	interfaces
xenbr0	8000.4439c43529e7	no	eno1

For more information on configuring a Linux Ethernet bridge, setting up networking with a static ip address, or any other Xen networking options, review the following pages on wiki.xenproject.org (listed in order of complexity):

- Xen Project Beginners Guide[22], see section ‘Setup Linux Bridge for guest networking’
- Network Configuration Examples (Xen 4.1+)[23]
- Xen Networking[24]

Debian also provides detailed network and bridge configuration information on their website:

- Network Configuration[25]
- Bridge Network Connections[26]

22 https://wiki.xenproject.org/wiki/Xen_Project_Beginners_Guide
23 https://wiki.xenproject.org/wiki/Network_Configuration_Examples_(Xen_4.1%2B)
24 https://wiki.xenproject.org/wiki/Xen_Networking
25 https://wiki.debian.org/NetworkConfiguration
26 https://wiki.debian.org/BridgeNetworkConnections

2.5 Autoballoon and memory allocation

According to the Xen Project guides 'Tuning Xen for Performance'[27] and 'Xen Project Best Practices'[28], it's best to disable Xen's 'autoballoon' feature and allocate a set amount of memory to Dom0 for the following two reasons:

1. as the Linux kernel calculates various network related parameters based on the amount of memory detected at boot, these parameters could potentially be incorrect if memory allocation changes after boot, and
2. memory meta-data is created based on the amount of memory detected at boot; therefore, RAM is wastefully filled up with meta-data for memory that is not longer available to the system.

Note that 1GB of swap space was allocated during the Debian base operating system's partitioning process. For the same reasons as provided in Section 1.4, above, 1GB of memory will be allocated to the Dom0 virtual machine. To ensure this allocation is properly carried out, three Xen configuration files must be modified to set Dom0's memory limit and disable the 'autoballoon' feature.

2.5.1 Edit /etc/xen/xl.conf

As root, create a backup copy of file '/etc/xen/xl.conf':

```
# cp /etc/xen/xl.conf /etc/xen/xl.conf.backup
```

Then open the 'xl.conf' file in *nano*, and look for the commented-out line '#autoballoon="auto"'.

```
# nano /etc/xen/xl.conf
```

Delete the '#' character and change '"auto"' to '0', so the line shows 'autoballoon=0'. On the test computer, the modified file now reads:

```
## Global XL config file ##

# Control whether dom0 is ballooned down when xen doesn't have enough
# free memory to create a domain.  "auto" means only balloon if dom0
# starts with all the host's memory.
autoballoon=0

# full path of the lockfile used by xl during domain creation
#lockfile="/var/lock/xl"

# default output format used by "xl list -l"
#output_format="json"

# first block device to be used for temporary VM disk mounts
#blkdev_start="xvda"

# default option to run hotplug scripts from xl
# if disabled the old behaviour will be used, and hotplug scripts will be
# launched by udev.
#run_hotplug_scripts=1
```

27 https://wiki.xenproject.org/wiki/Tuning_Xen_for_Performance
28 https://wiki.xenproject.org/wiki/Xen_Project_Best_Practices

```
# default backend domain to connect guest vifs to.  This can be any
# valid domain identifier.
#vif.default.backend="0"

# default gateway device to use with vif-route hotplug script
#vif.default.gatewaydev="eth0"

# default vif script to use if none is specified in the guest config
#vif.default.script="vif-bridge"

# default bridge device to use with vif-bridge hotplug scripts
#vif.default.bridge="xenbr0"

# Reserve a claim of memory when launching a guest. This guarantees immediate
# feedback whether the guest can be launched due to memory exhaustion
# (which can take a long time to find out if launching huge guests).
# see xl.conf(5) for details.
#claim_mode=1

# Specify global vcpu hard affinity masks. See xl.conf(5) for details.
#vm.cpumask="0-7"
#vm.pv.cpumask="0-3"
#vm.hvm.cpumask="3-7"
```

Save the changes and exit *nano*.

[N.B. Note that it does not appear to affect the configuration if the zero is surrounded in quotes or not, for example, “0” versus 0.]

2.5.2 Edit /etc/xen/xend-config.sxp

As root, create a backup copy of file ‘/etc/xen/xend-config.sxp’:

```
# cp /etc/xen/xend-config.sxp /etc/xen/xend-config.sxp.backup
```

Then open the ‘xend-config.sxp’ file in *nano*, and look for the lines:

```
(dom0-min-mem 196)
(enable-dom0-ballooning yes)
```

Change these lines to:

```
(dom0-min-mem 1024)
(enable-dom0-ballooning no)
```

On the test computer, the sections that contained these lines now read:

```
# dom0-min-mem is the lowest permissible memory level (in MB) for dom0.
# This is a minimum both for auto-ballooning (as enabled by
# enable-dom0-ballooning below) and for xm mem-set when applied to dom0.
(dom0-min-mem 1024)

# Whether to enable auto-ballooning of dom0 to allow domUs to be created.
# If enable-dom0-ballooning = no, dom0 will never balloon out.
(enable-dom0-ballooning no)
```

Save the changes and exit *nano*.

2.5.3 Edit /etc/default/grub.d/xen.cfg

As root, create a backup copy of file '/etc/default/grub.d/xen.cfg':

```
# cp /etc/default/grub.d/xen.cfg /etc/default/grub.d/xen.cfg.backup
```

Then open the 'xen.cfg' file in *nano* and add the line:

```
GRUB_CMDLINE_XEN_DEFAULT="dom0_mem=1024M,max:1024M"
```

The modified 'xen.cfg' file on the test computer now reads:

```
#
# Uncomment the following variable and set to 0 or 1 to avoid warning.
#
#XEN_OVERRIDE_GRUB_DEFAULT=0

echo "Including Xen overrides from /etc/default/grub.d/xen.cfg"

#
# When running update-grub with the Xen hypervisor installed, there are
# some additional variables that can be used to pass options to the
# hypervisor or the dom0 kernel.

# The following two are used to generate arguments for the hypervisor:
#
#GRUB_CMDLINE_XEN_DEFAULT=""
#GRUB_CMDLINE_XEN=""
#
# For example:
#
# dom0_mem=<size>[M]:max=<size>[M]
#   Sets the amount of memory dom0 uses (max prevents balloning for more)
# com[12]=<speed>,<data bits><parity><stopbits>
#   Initialize a serial console from in the hypervisor (eg. 115200,8n1)
#   Note that com1 would be ttyS0 in Linux.
# console=<dev>[,<dev> ...]
#   Redirects Xen hypervisor console (eg. com1,vga)

GRUB_CMDLINE_XEN_DEFAULT="dom0_mem=1024M,max:1024M"

#
# The next two lines are used for creating kernel arguments for the dom0
# kernel. This allows to have different options for the same kernel used
# natively or as dom0 kernel.
#
#GRUB_CMDLINE_LINUX_XEN_REPLACE_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT"
#GRUB_CMDLINE_LINUX_XEN_REPLACE="$GRUB_CMDLINE_LINUX"
#
# For example:
#
# earlyprintk=xenboot
#   Allows to send early printk messages to the Xen hypervisor console
# console=hvc0
#   Redirects the Linux console to the hypervisor console

#
# Make booting into Xen the default if not changed above. Finding the
# current string for it always has been a problem.
#
```

```
if [ "$XEN_OVERRIDE_GRUB_DEFAULT" = "" ]; then
	echo "WARNING: GRUB_DEFAULT changed to boot into Xen by default!"
	echo "	Edit /etc/default/grub.d/xen.cfg to avoid this warning."
	XEN_OVERRIDE_GRUB_DEFAULT=1
fi
if [ "$XEN_OVERRIDE_GRUB_DEFAULT" = "1" ]; then
	GRUB_DEFAULT="Debian GNU/Linux, with Xen hypervisor"
fi
```

Save the changes and exit *nano*.

[N.B. I had some issues at this point in the configuration process. In hindsight, I realize I was confused when I compared the syntax used in other Xen tutorials to the example in the 'xen.cfg' file provided on the test computer. The main confusion was concerning whether a space was permitted after the comma, and whether I should be using 'max=1024M' versus 'max:1024M'.

Even with the example 'dom0_mem=<size>[M]:max=<size>[M]' provided in the 'xen.cfg' file on the test computer, the syntax '"dom0_mem=1024M,max:1024M"' is what worked in the end.]

IMPORTANT: You must update GRUB or the changes will not take effect!

To update GRUB run the following command as root:

```
# update-grub
```

2.6 Allocate Dom0 virtual CPUs

As the test computer only has four cores in its processor, no configuration was made to allocate CPUs to Dom0. By default, Xen makes all processor cores available to Dom0, who then shares these cores with DomU when the virtual machines are created (see below on how to allocate cores to DomU). This configuration is in line with the Xen Project guide 'Tuning Xen for Performance'[29], which states under the 'Dom0 vCPUs' section that "In general you should not assigned less than 4 vCPUs to Dom0".

For more specific information on allocating vCPUs in your Xen system, refer to the following guides:

- 'Tuning Xen for Performance'[30] under the section 'Dom0 vCPUs'
- 'Xen - Debian Wiki'[31] under the section 'Configure dom0 CPUs'

2.7 Disable domain saving and restoring

As mentioned in section 'Configure guest behaviour on host reboot' of the 'Xen - Debian Wiki'[32] guide, it is a good idea to configure how DomU responds when Dom0 shuts down or reboots. Applying the configuration below ensures that Dom0 will not try to save or hibernate guest virtual machines when shutting down, which may have unforeseen consequences.

29 https://wiki.xenproject.org/wiki/Tuning_Xen_for_Performance
30 https://wiki.xenproject.org/wiki/Tuning_Xen_for_Performance
31 https://wiki.debian.org/Xen
32 https://wiki.debian.org/Xen

The configuration is made in the file '/etc/default/xendomains'. Make a backup copy of the file before modifying it. As root, run the following command:

```
# cp /etc/default/xendomains /etc/default/xendomains.backup
```

Then open the 'xendomains' file using *nano* and replace,

```
XENDOMAINS_SAVE=/var/lib/xen/save
XENDOMAINS_RESTORE=true
```

with the lines:

```
XENDOMAINS_SAVE=
XENDOMAINS_RESTORE=false
```

The '/etc/default/xendomains' file on the test computer now reads:

```
# The xendomains script can send SysRq requests to domains on shutdown.
# If you don't want to MIGRATE, SAVE, or SHUTDOWN, this may be a possibility
# to do a quick and dirty shutdown ("s e i u o") or at least sync the disks
# of the domains ("s").
#
# XENDOMAINS_SYSRQ=

# Set this to a non-empty string if you want to migrate virtual machines
# on shutdown. The string will be passed to the xm migrate DOMID command
# as is: It should contain the target IP address of the physical machine
# to migrate to and optionally parameters like --live. Leave empty if
# you don't want to try virtual machine relocation on shutdown.
# If migration succeeds, neither SAVE nor SHUTDOWN will be executed for
# that domain.
#
# XENDOMAINS_MIGRATE=

# Directory to save running domains to when the system (dom0) is
# shut down. Will also be used to restore domains from if # XENDOMAINS_RESTORE
# is set (see below). Leave empty to disable domain saving on shutdown
# (e.g. because you rather shut domains down).
# If domain saving does succeed, SHUTDOWN will not be executed.
#
XENDOMAINS_SAVE=

# This variable determines whether saved domains from XENDOMAINS_SAVE
# will be restored on system startup.
#
XENDOMAINS_RESTORE=false

# This variable sets the directory where domains configurations
# are stored that should be started on system startup automatically.
# Leave empty if you don't want to start domains automatically
# (or just don't place any xen domain config files in that dir).
# Note that the script tries to be clever if both RESTORE and AUTO are
# set: It will first restore saved domains and then only start domains
# in AUTO which are not running yet.
# Note that the name matching is somewhat fuzzy.
#
XENDOMAINS_AUTO=/etc/xen/auto

# On xendomains stop, a number of xm commands (xm migrate, save, shutdown,
```

```
# shutdown --all) may be executed. In the worst case, these commands may
# stall forever, which will prevent a successful shutdown of the machine.
# If this variable is non-zero, the script will set up a watchdog timer
# for every of these xm commands and time it out after the number of seconds
# specified by this variable.
# Note that SHUTDOWN_ALL will not be called if no virtual machines or only
# zombies are still running, so you don't need to enable this timeout just
# for the zombie case.
# The setting should be large enough to make sure that migrate/save/shutdown
# can succeed. If you do live migrations, keep in mind that live migration
# of a 1GB machine over Gigabit ethernet may actually take something like
# 100s (assuming that live migration uses 10% of the network # bandwidth).
# Depending on the virtual machine, a shutdown may also require a significant
# amount of time. So better setup this variable to a huge number and hope the
# watchdog never fires.
#
XENDOMAINS_STOP_MAXWAIT=300
```

Save the changes and exit *nano*.

Restart the computer and boot into the Xen hypervisor/Dom0 virtual machine.

All configuration in the Debian base operating system is complete. Once you are ready, run the *reboot* command as root:

```
# reboot
```

2.8 Boot into the Xen hypervisor/Dom0 and snoop around

If you have a monitor plugged in, after the computer reboots, you will see the GRUB menu with "Debian GNU/Linux, with Xen hypervisor" set as default. This is your new Xen hypervisor/Dom0 installation, and our current destination.

If you're using SSH on a headless system, you'll have to wait until the computer passes the GRUB menu, and loads the Xen hypervisor/Dom0 activating the SSH server daemon. Note that if the SSH server was enabled in the Debian base operating system, it will be enabled in the Xen hypervisor/Dom0.

Whether proceeding with or without a monitor, once you reach the familiar terminal login screen, **use your login and password from the Debian base operating system to login to the new Xen hypervisor/Dom0 virtual machine**.

Now that you're in Dom0, check your work to make sure that all configurations made in the Debian base operating system were passed on to Dom0. Start by running the *free* command to verify that Dom0 has been allocated its 1GB of memory:

```
$ free -h
```

This results in the following output in the Dom0 virtual machine on the test computer:

	total	used	free	shared	buff/cache	available
Mem:	926M	61M	775M	3.1M	88M	842M
Swap:	953M	0B	953M			

Running *top* will also provide verification that Dom0 has been properly allocated memory:

```
$ top
```

Here is line four and five of *top*'s output when run on the test computer:

```
KiB Mem :   948248 total,   794120 free,   63040 used,   91088 buff/cache
KiB Swap:   976892 total,   976892 free,   0 used.       863056 avail Mem
```

[N.B. Use the 'q' key to quit *top*.]

While *free* and *top* are useful native GNU/Linux programs, the Xen management tool *xl*[33] is essential while navigating Dom0 and examining other virtual machines. Note that ***xl* must always be run as root**, otherwise *bash* will complain: "command not found".

2.8.1 xl list

Run the command *xl list*, which lists all currently running Xen virtual machines along with select operating and configuration information:

```
# xl list
```

The test computer produces the following output:

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	1024	4	r-----	3.7

The output can be interpreted as,

There is one virtual machine running named Domain-0, which has the following characteristics:

- its ID is 0 (Dom0's ID is always 0),
- it has been assigned 1024MB of memory,
- it has the use of 4 virtual CPUs,
- its current state is 'r' or 'running', and
- it has used 3.7 seconds of CPU time.

Specific information on *xl list* – and *xl* in general – can be found at xenbits.xen.org[34] or by running the command *man xl*.

2.8.2 xl top

The command *xl top* also provides detailed information on currently loaded Xen virtual machines:

```
# xl top
```

33 https://xenbits.xen.org/docs/unstable/man/xl.1.html
34 https://xenbits.xen.org/docs/unstable/man/xl.1.html

This results in the following output on the test machine (for practical reasons, the output of *xl top* was truncated in this example):

```
xentop – 22:20:35     Xen 4.8.5-pre
1 domains: 1 running, 0 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8275580k total, 1146048k used, 7129532k free    CPUs: 4 @ 3192MHz
NAME          STATE    CPU(sec)       CPU(%)   MEM(k)  MAXMEM(k)  VCPUS
Domain-0      -----r   3              0.1      1048576  1048576        4
```

From the output generated in the above commands, it appears that the configuration of memory, max memory, and vCPUs to Dom0 was successful.

[N.B. If you would like to verify network and bridge configuration in Dom0, refer to Section 2.4, above, for discussion of the commands *ip addr* and *brctl show*. These commands are available in Dom0, as they were in the Debian base operating system.]

Step 3 - Installing a guest domain ('DomU')

Now that the Xen hypervisor/Dom0 virtual machine is configured and up and running, it's time to configure and create a DomU. For the purpose of this tutorial, the new DomU virtual machine will be configured with the intent of turning it into a FTP server for users on the local network. For simplicity sake, the DomU will be setup to run Debian 9 stretch with a 4GB primary disk image, 1024MB swap partition, and 1024MB of allocated memory. DomU will also be given access to the 500GB HDD on the test computer ('sdb'), and its networking configuration will be set to DHCP.

The Debian package *xen-tools* will be used to configure and create the new DomU virtual machine. By default, *xen-tools* will also configure DomU's networking and setup OpenSSH.

3.1 Install Debian package *xen-tools*

Use *apt-get* to update the Xen hypervisor/Dom0 package index files, and upgrade all currently installed packages. As root run the following command:

```
# apt-get update && apt-get upgrade
```

Next, use *apt-get* to install the *xen-tools*[35] package from the Debian stretch repository, which is currently shipping *xen-tools* version 4.7-1. Run the following command as root:

```
# apt-get install xen-tools
```

3.2 Review *xen-tools* default values in 'xen-tools.conf' and the *xen-create-image* manual

Once *xen-tools* is installed, review the program's documentation and configuration files to get a better understanding of its functionality. Many of the default DomU image creation settings for *xen-tools* are located in the file '/etc/xen-tools/xen-tools.conf', which can be examined using the *less* command:

```
$ less /etc/xen-tools/xen-tools.conf
```

On the test computer, the following lines are uncommented in the 'xen-tools.conf' file thereby making them the default values when creating DomU virtual machine images:

```
#  Installation method.
install-method = debootstrap
…
#  Disk and Sizing options.
size        = 4G        # Root disk, suffix (G, M, k) required
memory      = 256M      # Suffix (G, M, k) required
swap        = 512M      # Suffix (G, M, k) required
fs          = ext4      # Default file system for any disk
dist        = `xt-guess-suite-and-mirror --suite`
                        # Default distribution is determined by Dom0's distribution
image       = sparse    # Specify sparse vs. full disk images (file based images only)
…
# Default kernel and ramdisk to use for the virtual servers
kernel      = /boot/vmlinuz-`uname -r`
initrd      = /boot/initrd.img-`uname -r`
```

35 https://packages.debian.org/stretch/xen-tools

```
…
#  Filesystem options for the different filesystems we support.
ext4_options         = noatime,nodiratime,errors=remount-ro
ext3_options         = noatime,nodiratime,errors=remount-ro
ext2_options         = noatime,nodiratime,errors=remount-ro
xfs_options          = defaults
reiserfs_options     = defaults
btrfs_options        = defaults
```

[N.B. The '/etc/xen-tools/xen-tools.conf' file on the test computer is 326 lines, so the uncommented lines listed above are only a small subset of all possible configuration.]

After reviewing the 'xen-tools.conf' file, review the manual page for the *xen-create-image*[36] command:

```
$ man xen-create-image
```

An extensive list of DomU configuration options are found in the manual page including additional default options concerning the allocation of VCPUs, e.g.,:

```
--vcpus=num
                Set the number of vcpus that the new instance will
                have instead of the default value of "1".
```

3.3 Test computer DomU configuration

For this tutorial, the following DomU configuration choices are accepted from the 'xen-tools.conf' file:

- **install-method = debootstrap**, the recommended Debian install-method;
- **fs = ext4**, the recommended filesystem type for the Debian root partition;
- **ext4_options = noatime,nodiratime,errors=remount-ro**, the default ext4 filesystem mount options [for more information refer to the manual pages *mount* and *fstab*];
- **kernel = /boot/vmlinuz-`uname -r`**, the location of the kernel to be used in creating the DomU virtual machine. Note that this is Dom0's kernel location; and,
- **initrd = /boot/initrd.img-`uname -r`**, the location of the boot loader initialized RAM disk to be used in creating the DomU virtual machine. Note that this is Dom0's 'initrd.img' file location.

[N.B. Concerning the configuration **image = sparse**, the *xen-create-image* manual page explains: "Full images are mandatory when using LVM, so this setting is ignored in that case." Therefore, as this tutorial is using LVM for its DomU disk allocation, the **image=** default configuration is not applicable.]

Additional configuration choices will be passed on the command line during DomU image creation, which include:

- **--hostname=ftpserver**, the chosen hostname for the DomU (e.g., user@ftpserver);
- **--lvm=vg0**, the name of the Volume Group from which to create the DomU 'root disk' and 'swap partitions';
- **--dist=stretch**, the GNU/Linux distribution and version to be installed on the DomU;

36 https://manpages.debian.org/stretch/xen-tools/xen-create-image.8.en.html

- **--size=4G**, the size chosen for the root partition on the DomU. Note that this configuration is technically redundant as 'xen-tools.conf' has the default set to 4G already;
- **--memory=1024M**, the amount of memory to be allocated to the DomU at startup;
- **--maxmem=1024M**, the maximum memory to ever be allocated to the DomU;
- **--swap=1024M**, the swap partition size to be created for the DomU;
- **--pygrub**, the DomU is to be booted using *pygrub*[37]; and,
- **--dhcp**, DomU networking will be configured via DHCP.

All other default DomU configuration options, not listed above, are accepted as outlined in the *xen-create-image*[38] and *xl.cfg*[39] manual pages. As mentioned above, since the default VCPU value is **--vcpus=1**, the new DomU will be assigned one virtual CPU at startup. Also, note that the default configuration for DomU *type* on x86 architecture is **--type="pv"**, which means the virtual machine will be paravirtualized[40].

3.4 Create the DomU virtual machine image using *xen-tools*

Now that the DomU's configuration has been established, it's time to create the virtual machine image file. As root, run the *xen-create-image* command to create the DomU virtual machine image and its corresponding Logical Volumes.

```
# xen-create-image --hostname=ftpserver --lvm=vg0 --dist=stretch
--size=4G --memory=1024M --maxmem=1024M --swap=1024M --pygrub
--dhcp
```

[N.B. The command above is one long line. If you don't like long text in the command line, you can append backslashes ('\') to your command like so,

```
# xen-create-image \
> --hostname=ftpserver \
> --lvm=vg0 \
> --dist=stretch \
> --size=4G \
> --memory=1024M \
> --maxmem=1024M \
> --swap=1024M \
> --pygrub \
> --dhcp
```

…]

37 https://wiki.xen.org/wiki/PyGrub
38 https://manpages.debian.org/stretch/xen-tools/xen-create-image.8.en.html
39 https://xenbits.xen.org/docs/unstable/man/xl.cfg.5.html
40 https://wiki.xen.org/wiki/Paravirtualization_(PV)

Running the *xen-create-image* command results in the following output on the test computer:

```
General Information
--------------------
Hostname        :   ftpserver
Distribution    :   stretch
Mirror          :   http://httpredir.debian.org/debian
Partitions      :   swap        1024M   (swap)
                    /           4G      (ext4)
Image type      :   full
Memory size     :   1024M
Max mem size    :   1024M
Bootloader      :   pygrub

Networking Information
----------------------
IP Address      :   DHCP [MAC: 00:16:3E:52:85:D6]


Creating swap on /dev/vg0/ftpserver-swap
Done

Creating ext4 filesystem on /dev/vg0/ftpserver-disk
Done
Installation method: debootstrap
Done

Running hooks
Done

No role scripts were specified.  Skipping

Creating Xen configuration file
Done

No role scripts were specified.  Skipping
Setting up root password
Generating a password for the new guest.
All done


Logfile produced at:
        /var/log/xen-tools/ftpserver.log

Installation Summary
---------------------
Hostname        :   ftpserver
Distribution    :   stretch
MAC Address     :   00:16:3E:52:85:D6
IP Address(es)  :   dynamic
SSH Fingerprint :   SHA256:S9gtygjZ9/N0tNIU0dVILcJfbA0/cDsl2P3BqspXqZk (DSA)
SSH Fingerprint :   SHA256:OdUpgMghre0yuTIh7qPopXCEoAedfKfrP9Bbfrto3Cw (ECDSA)
SSH Fingerprint :   SHA256:9fg0hjGKdqEyRnMf/5/VTloDr3XGXWjs9jnjrSAGFuw (ED25519)
SSH Fingerprint :   SHA256:THyGJr0MVPgOetszKnKfsqIiRV8A5RjC2QcpX4T/jkU (RSA)
Root Password   :   AfkDDadUpcYP888MViTBNdZ
```

The DomU image creation process took just under 4 minutes on the test computer as it was the first time the *xen-create-image* program was run, and because the DomU required files to be downloaded from the Debian mirror website.

Note that the *xen-create-image* command creates an ‘ftpserver.cfg’ DomU image file in the ‘/etc/xen/’ directory only: **to start the DomU, the *xl create* command must be run.**

3.5 Review the newly created DomU .cfg file

Before starting the DomU, review the newly created configuration file ‘/etc/xen/ftpserver.cfg’ using the *less* command:

```
$ less /etc/xen/ftpserver.cfg
```

The ‘ftpserver.cfg’ file on the test computer contains the following lines:

```
#
# Configuration file for the Xen instance ftpserver, created
# by xen-tools 4.7 on Sun Oct 21 15:23:22 2018.
#

#
#  Kernel + memory size
#

bootloader = '/usr/lib/xen-4.8/bin/pygrub'

vcpus       = '1'
memory      = '1024'
maxmem      = '1024'

#
#  Disk device(s).
#
root        = '/dev/xvda2 ro'
disk        = [
                  'phy:/dev/vg0/ftpserver-disk,xvda2,w',
                  'phy:/dev/vg0/ftpserver-swap,xvda1,w',
                ]

#
#  Physical volumes
#

#
#  Hostname
#
name        = 'ftpserver'

#
#  Networking
#
dhcp        = 'dhcp'
vif         = [ 'mac=00:16:3E:52:85:D6' ]

#
#  Behaviour
#
on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Most configuration options found in the .cfg file should be familiar, including the bootloader, vcpus, memory, maxmem, name, and dhcp KEY=VALUE pairs.

Under the 'Disk device(s)' section, we find the newly created Logical Volumes **phy:/dev/vg0/ftpserver-disk** and **phy:/dev/vg0/ftpserver-swap**, or partitions 'xvda2' and 'xvda1', respectively. Note that 'xvda2' has been auto-configured as the 'root' partition, and 'xvda1' as swap.

For 'Networking', on creation, the ftpserver DomU will have a virtual interface ['vif'] assigned the MAC address '00:16:3E:52:85:D6'. Note that DomU's IP and MAC addresses will be different from Dom0's.

Finally, under the 'Behaviour' section, the DomU is provided instructions on how to respond when the virtual machine is told to power off, reboot, or crashes. Note that **on_poweroff** is synonymous with the systemd *shutdown* command, and **on_reboot** with the systemd *reboot* command.

As described in the *xl.cfg* manual, under the value **destroy** it states that destroy is used to "destroy the domain", and that the value **restart** is used to "destroy the domain and immediately create a new domain with the same configuration".

For more information on configurable options available in the '/etc/xen/ftpserver.cfg' file, refer to the *xl.cfg* manual page[41].

[N.B. It is important to differentiate the *xl.cfg* 'on_poweroff=destroy' configuration from the *xl destroy {domain-id}* command as detailed in the *xl* manual. The *xl.cfg* 'destroy' value is essentially equivalent to running the systemd *shutdown* command on DomU, which should cause the virtual machine to shutdown gracefully. Contrast this with the *xl destroy {domain-id}* command in the *xl* manual where the value 'destroy' has the following description:

"Immediately terminate the domain specified by domain-id. This doesn't give the domain OS any chance to react, and is the equivalent of ripping the power cord out on a physical machine. In most cases you will want to use the **shutdown** command instead."

The better way to shutdown or reboot DomU from the Dom0 console is with the *xl shutdown {domain-id}* or *xl reboot {domain-id}* commands, which are equivalent to DomU running the systemd shutdown or reboot commands from its own console.]

3.6 Confirm the creation of the DomU Logical Volumes

Take a moment to check that the *xen-create-image* command properly created the DomU's Logical Volumes. The best way to do this is by using the *lsblk* command:

```
$ lsblk
```

41 https://xenbits.xen.org/docs/unstable/man/xl.cfg.5.html

On the test computer, this results in the following output:

```
NAME                      MAJ:MIN  RM   SIZE  RO  TYPE  MOUNTPOINT
sda                       8:0      0  111.8G   0  disk
    sda1                  8:1      0    3.7G   0  part  /
    sda2                  8:2      0    954M   0  part  [SWAP]
    sda3                  8:3      0      1K   0  part
    sda5                  8:5      0  107.1G   0  part
        vg0-ftpserver--swap 254:0  0      1G   0  lvm
        vg0-ftpserver--disk 254:1  0      4G   0  lvm
sdb                       8:16     0  465.8G   0  disk
    sdb1                  8:17     0  465.8G   0  part
```

From the above output, note that the swap and disk partitions have been successfully created as ‘vg0-ftpserver--swap’ and ‘vg0-ftpserver--disk’ on the ‘vg0’ Volume Group.

As root, running the *lvs* command will also provide detailed information about Dom0’s Logical Volumes:

```
# lvs
```

On the test computer, this results in the following output:

```
LV             VG  Attr       Lsize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
ftpserver-disk vg0 -wi-ao---- 4.00g
ftpserver-swap vg0 -wi-ao---- 1.00g
```

The output provides further confirmation that DomU’s Logical Volumes were successfully created by the *xen-create-image* command.

3.7 Configure DomU access to secondary disk (Part I)

If we were to start the DomU virtual machine now and run *lsblk* on its console, we would get the following output:

```
NAME  MAJ:MIN  RM  SIZE  RO  TYPE  MOUNTPOINT
xvda1 202:1    0   1G    0   disk  [SWAP]
xvda2 202:2    0   4G    0   disk  /
```

That is because DomU is only permitted to see the disks and partitions passed through its configuration file, ‘/etc/xen/ftpserver.cfg’, as specified under the ‘disk’ heading:

```
disk            = [
                      'phy:/dev/vg0/ftpserver-disk,xvda2,w',
                      'phy:/dev/vg0/ftpserver-swap,xvda1,w',
                  ]
```

Given the above configuration, the DomU cannot see the ‘sda’ partitions of Dom0, nor can it see the secondary disk ‘sdb’. As the ftpserver DomU requires access to the 500GB HDD sdb1 partition, we must provide the proper configuration in the DomU’s .cfg file.

Before modifying the file, as root, create a backup copy of ‘/etc/xen/ftpserver.cfg’:

```
# cp /etc/xen/ftpserver.cfg /etc/xen/ftpserver.cfg.backup
```

Next, use *nano* to modify the '/etc/xen/ftpserver.cfg' file so that the 'disk' section now reads:

```
disk            = [
                      'phy:/dev/vg0/ftpserver-disk,xvda2,w',
                      'phy:/dev/vg0/ftpserver-swap,xvda1,w',
                      'phy:/dev/sdb1,xvda3,w',
                    ]
```

Note that only one virtual machine may have the partition mounted, which is why we did not create a mount point for the sdb1 partition in Dom0. Once we boot the DomU, we will login to the new virtual machine, create a mount point for sdb1, and configure the *fstab*[42] file to auto-mount the partition.

Refer to the *xl-disk-configuration*[43] manual page, available online, for more specific disk configuration information.

3.8 Start the new DomU

To start the new DomU virtual machine, use the *xl* Xen management tool to run the *xl create* command as root, specifying the 'ftpserver.cfg' DomU image file previously created:

```
# xl create /etc/xen/ftpserver.cfg
```

[N.B. Running the *xl create* command with the -c flag – e.g., *xl create -c /etc/xen/ftpserver.cfg* – will cause the terminal window to be passed to the newly created DomU. Use the 'CTRL +]' key combination to exit the DomU terminal.]

Verify that the new 'ftpserver' DomU virtual machine is running with the *xl list* command:

```
# xl list
```

The test computer generates the following output:

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	1022	4	r-----	134.3
ftpserver	1	1024	1	-b----	1.9

Similarly, running the *xl top* command on the test computer provides the following information (truncated for presentation purposes):

```
xentop - 22:56:57   Xen 4.8.5-pre
2 domains: 1 running, 1 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8275580k total, 2193148k used, 6082432k free    CPUs: 4 @ 3192MHz
```

NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPUS	NETS
Domain-0	-----r	134	0.3	1046884	12.7	1048576	12.7	4	0
ftpserver	--b---	1	0.0	1048576	12.7	1049600	12.7	1	1

Access the new DomU's console with the *xl console* command:

```
# xl console ftpserver
```

42 https://manpages.debian.org/stretch/mount/fstab.5.en.html
43 https://xenbits.xen.org/docs/unstable/man/xl-disk-configuration.5.html

Login to the DomU as user ‘root’ with the password generated during the *xen-create-image* process, e.g.,:

Root Password : AfkDDadUpcYP888MViTBNdZ

The password set during the DomU image creation process may also be found in the directory ‘/var/log/xen-tools/’, under the .log file associated with your chosen hostname. Of course, use the root password generated during your installation process.

3.9 Configure DomU access to secondary disk (Part II)

Running the *lsblk* command reveals that the ‘sdb1’ partition, or virtual disk ‘xvda3’, has been successfully passed through to the DomU, but is not currently mounted:

```
$ lsblk
```

This generates the following output on the test computer:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
xvda1	202:1	0	1G	0	disk	[SWAP]
xvda2	202:2	0	4G	0	disk	/
xvda3	202:3	0	465.8G	0	disk	

The final task is to create a mount point for the partition, and ensure it auto-mounts on boot.

First, as root, create a mount point for ‘xvda3’ in the ‘/media/’ directory. The mount point will be a directory called ‘WD_500GB’:

```
# mkdir /media/WD_500GB
```

Next, create a backup copy of the file ‘/etc/fstab’:

```
# cp /etc/fstab /etc/fstab.backup
```

Before modifying the ‘fstab’ file, review the file’s contents with the *more* command:

```
# more /etc/fstab
```

[N.B. The ftpserver DomU did not have the *less* package installed by default. Using *more* or even *cat* is sufficient in this case, as the ‘fstab’ file is not long.]

The test computer DomU generated the following output:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>                       <dump>  <pass>
proc            /proc           proc    defaults                        0       0
devpts          /dev/pts        devpts  rw,noexec,nosuid,gid=5,mode=620 0       0
/dev/xvda1      none            swap    sw                              0       0
/dev/xvda2      /               ext4    noatime,nodiratime,errors=remount-ro 0  1
```

Now, use *nano* to add the following line to the ‘fstab’ file:

```
/dev/xvda3      /media/WD_500GB ext4    errors=remount-ro               0       2
```

DomU's 'fstab' should now read as follows:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>                       <dump>  <pass>
proc            /proc           proc    defaults                        0       0
devpts          /dev/pts        devpts  rw,noexec,nosuid,gid=5,mode=620 0       0
/dev/xvda1      none            swap    sw                              0       0
/dev/xvda2      /               ext4    noatime,nodiratime,errors=remount-ro 0  1
/dev/xvda3      /media/WD_500GB ext4    errors=remount-ro               0       2
```

Save the changes and exit *nano*. Now DomU is configured to mount the 'xvda3' partition to the '/media/WD_500GB/' directory every time the virtual machine is started.

Run the *mount* command to mount all file systems specified in the 'fstab' file:

```
# mount -a
```

Lastly, use the lsblk command to verify that all DomU partitions were successfully mounted:

```
# lsblk
```

DomU on the test computer generates the following output:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
xvda1	202:1	0	1G	0	disk	[SWAP]
xvda2	202:2	0	4G	0	disk	/
xvda3	202:3	0	465.8G	0	disk	/media/WD_500GB

3.10 DomU statistics from *free* and *df*, and *ip addr* output

For those of you who are curious as to how much memory and disk space is used by the Debian stretch DomU after installation, here is output generated from the *free* and *df* commands on the test computer:

```
$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	992M	34M	915M	1.4M	42M	937M
Swap:	1.0G	0B	1.0G			

```
$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	468M	0	468M	0%	/dev
tmpfs	100M	1.4M	98M	2%	/run
/dev/xvda2	3.9G	575M	3.1G	16%	/
tmpfs	497M	0	497M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	497M	0	497M	0%	/sys/fs/cgroup
/dev/xvda3	458G	73M	435G	1%	/media/WD_500GB

Additionally, running the *ip addr* command in the DomU outputs the following:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
   link/ether 00:16:3e:52:85:d6 brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.143/24 brd 192.168.1.255 scope global eth0
      valid_lft forever preferred_lft forever
   inet6 fe80::216:3eff:fe52:85d6/64 scope link
      valid_lft forever preferred_lft forever
```

A few more tips

Before we wrap up, here are a few navigational tips.

While in DomU's console:

- To shutdown the DomU, as root run the command *shutdown -h now*
- To reboot the DomU, as root run the command *reboot*
- To exit DomU's console and go back to Dom0 use the 'CTRL +]' key combination

While in Dom0's console:

- To shutdown a DomU, as root run the command *xl shutdown {domain-id}*
- To reboot a DomU, as root run the command *xl reboot {domain-id}*
- To exit Dom0's console and go back to DomU, as root run the command *xl console {domain-id}*
- To start a DomU for the first time, or to restart after shutdown, as root run the command *xl create {full-location-to-DomU .cfg file}*

[N.B. If it appears you are in a frozen terminal in any one of the transitions above, try pressing <Enter> again, and you will likely find out that you are not!]

It is important to note that the DomU operating system – like any *real* operating system – will retain all configuration history and file creation/destruction changes even after it is shutdown and restarted. Simply configure and use the DomU virtual machine as you would any computer.

Final words

If you encounter issues during any of the installation or configuration stages described above, make sure to thoroughly review the manual pages and/or search for relevant websites or tutorials specific to your issue – sometimes this is easier said then done, as it can be hard to figure out exactly what is causing the problem. For all intents and purposes, the Xen Project's wiki[44] is an excellent source of information, as are the other websites I've linked to throughout the tutorial.

Here's hoping that your Xen Project hypervisor installation went as smoothly as mine!

Now that everything is up and running, it's time to decide which Debian repository FTP package to install…

44 https://wiki.xenproject.org/wiki/Main_Page