

Gowamr1

Virtualisation IN720 2018

Assignment 3: Xen



Mark Gowans

Table of Contents

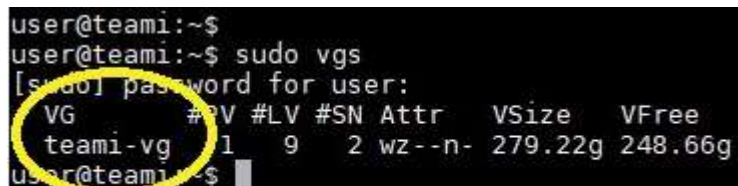
Set up a volume for a guest image	1
Obtain files necessary for running a Linux installer	2
Download Kernels	2
Make Config File	3
Boot a guest running the installer and carry out the installation	5
Perform post installation finalisation of the guest image	8
Modify Config.....	8
Restart VM	9
Prepare volumes for cloned guests	10
Stop VM	10
Make New Volumes	11
Write Xen configuration files for the cloned guests	11
Create the guest domains.....	13
Basic functionality of the running guests.....	13

Set up a volume for a guest image

First you want to find the name of our volume group.

Enter the command

`vgs`



```
user@teami:~$  
user@teami:~$ sudo vgs  
[sudo] password for user:  
VG          #PV #LV #SN Attr   VSize   VFree  
teami-vg    1   9  2 wz--n- 279.22g 248.66g  
user@teami:~$
```

then create a new volume for our guest

Enter the command

`sudo lvcreate -L 10G -n guest_volume /dev/<name of volume group (VG) from last command>`

This creates a volume named `guest_volume`

Obtain files necessary for running a Linux installer

We need to download the netboot installer for Ubuntu 14.04

Make a directory to hold the downloaded files

```
mkdir -p /var/lib/xen/images/ubuntu-netboot/trusty
```

then make that new directory the working directory

```
cd /var/lib/xen/images/ubuntu-netboot/trusty
```

Download Kernels.

First one (remember this command is just one line).

After the command is entered, something will download.

```
wget http://ucmirror.canterbury.ac.nz/ubuntu/dists/trusty/main/installer-  
amd64/current/images/netboot/xen/vmlinuz
```

Second (again, one command and there will be a download)

```
wget http://ucmirror.canterbury.ac.nz/ubuntu/dists/trusty/main/installer-  
amd64/current/images/netboot/xen/initrd.gz
```

```
user@teami:/etc/xen$ sudo wget http://ucmirror.canterbury.ac.nz/ubuntu/dists/trusty/main/installer-  
-amd64/current/images/netboot/xen/vmlinuz  
--2018-11-14 16:01:03-- http://ucmirror.canterbury.ac.nz/ubuntu/dists/trusty/main/installer-amd64/  
/current/images/netboot/xen/vmlinuz  
Resolving ucmirror.canterbury.ac.nz (ucmirror.canterbury.ac.nz)... 132.181.7.179  
Connecting to ucmirror.canterbury.ac.nz (ucmirror.canterbury.ac.nz)[132.181.7.179]:80... connected  
.  
HTTP request sent, awaiting response... 200 OK  
Length: 5778968 (5.5M) [text/plain]  
Saving to: 'vmlinuz'  
  
100%[=====] 5,778,968 10.9MB/s in 0.5s  
  
2018-11-14 16:01:04 (10.9 MB/s) - 'vmlinuz' saved [5778968/5778968]  
  
user@teami:/etc/xen$ sudo wget http://ucmirror.canterbury.ac.nz/ubuntu/dists/trusty/main/installer-  
-amd64/current/images/netboot/xen/initrd.gz  
--2018-11-14 16:01:27-- http://ucmirror.canterbury.ac.nz/ubuntu/dists/trusty/main/installer-amd64/  
/current/images/netboot/xen/initrd.gz  
Resolving ucmirror.canterbury.ac.nz (ucmirror.canterbury.ac.nz)... 132.181.7.179  
Connecting to ucmirror.canterbury.ac.nz (ucmirror.canterbury.ac.nz)[132.181.7.179]:80... connected  
.  
HTTP request sent, awaiting response... 200 OK  
Length: 21256771 (20M) [application/x-gzip]  
Saving to: 'initrd.gz'  
  
100%[=====] 21,256,771 11.1MB/s in 1.8s  
  
2018-11-14 16:01:29 (11.1 MB/s) - 'initrd.gz' saved [21256771/21256771]  
  
user@teami:/etc/xen$
```

Make Config File

We now need to make a config file for Xen to inform it how our VM should be set up. We can use files already made from the Xen installation

Change current directory to /etc/xen

```
cd /etc/xen
```

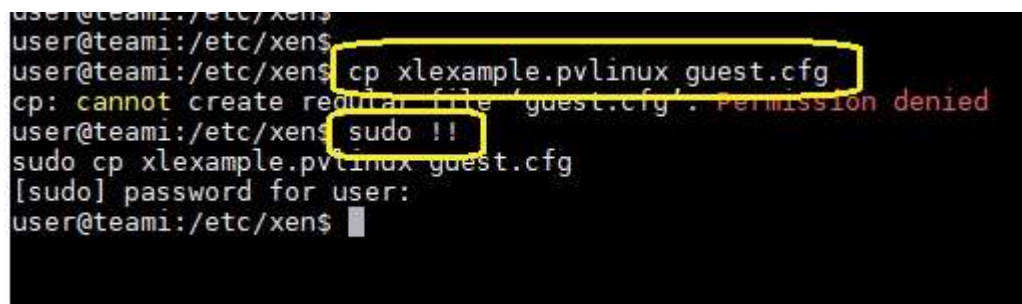
then copy a configuration file and rename the copy something that relates to our guest name.

```
sudo cp xlexample.pvlinux guest.cfg
```

If you find a command doesn't work because of **Permission denied**, it is because sudo is needed in front of it. Just type

```
sudo !!
```

It reruns the last command with sudo at the front



```
user@team1:/etc/xen$  
user@team1:/etc/xen$ cp xlexample.pvlinux guest.cfg  
cp: cannot create regular file 'guest.cfg': Permission denied  
user@team1:/etc/xen$ sudo !!  
sudo cp xlexample.pvlinux guest.cfg  
[sudo] password for user:  
user@team1:/etc/xen$
```

Now we edit the file so Xen can run our VM

```
sudo vim guest.cfg
```

Press i to insert text.

- Set name to `guestvm`
- Set kernel to `/var/lib/xen/images/ubuntu-netboot/trusty/vmlinuz` (note that this kernel is from the file we downloaded earlier)
- Set ramdisk to `/var/lib/xen/images/ubuntu-netboot/trusty/initrd.gz` (also from the file we downloaded earlier)
- Set memory to `1024`
- Set vcpus to `1`
- Set disk to `/dev/<name of volume group>/guest_volume,raw,xvda,rw`

```
# =====
# Example PV Linux guest configuration
# =====
#
# This is a fairly minimal example of what is required for a
# Paravirtualised Linux guest. For a more complete guide see xl.cfg(5)
#
# Guest name
name = "guestvm"
#
# 128-bit UUID for the domain as a hexadecimal number.
# Use "uuidgen" to generate one if required.
# The default behavior is to generate a new UUID each time the guest is started.
#uuid = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
#
# Kernel image to boot
kernel = "/var/lib/xen/images/ubuntu-netboot/trusty/vmlinuz"
#
# Ramdisk (optional)
ramdisk = "/var/lib/xen/images/ubuntu-netboot/trusty/initrd.gz"
#
# Kernel command line options
extra = "root=/dev/xvda1"
#
# Initial memory allocation (MB)
memory = 1024
#
# Maximum memory (MB)
# If this is greater than 'memory' then the slack will start ballooned
# (this assumes guest kernel support for ballooning)
#maxmem = 512
#
# Number of VCPUS
vcpus = 1
#
# Network devices
# A list of 'vifspec' entries as described in
# docs/misc/xl-network-configuration.markdown
vif = [ '' ]
#
# Disk Devices
# A list of 'diskspec' entries as described in
# docs/misc/xl-disk-configuration.txt
disk = [ '/dev/teami-vg/guest_volume,raw,xvda,rw' ]
#
~
```

Once these modifications have been done, press ESC, then `:wq`. This saves the changes and closes Vim.

```
~
~
~
:wq
```

Boot a guest running the installer and carry out the installation

Now that our config file is set up, we need to run the VM.

```
sudo xl create -c /etc/xen/guest.cfg
```

Notice that the second part of that command is pointing to the config file we just copied and edited.

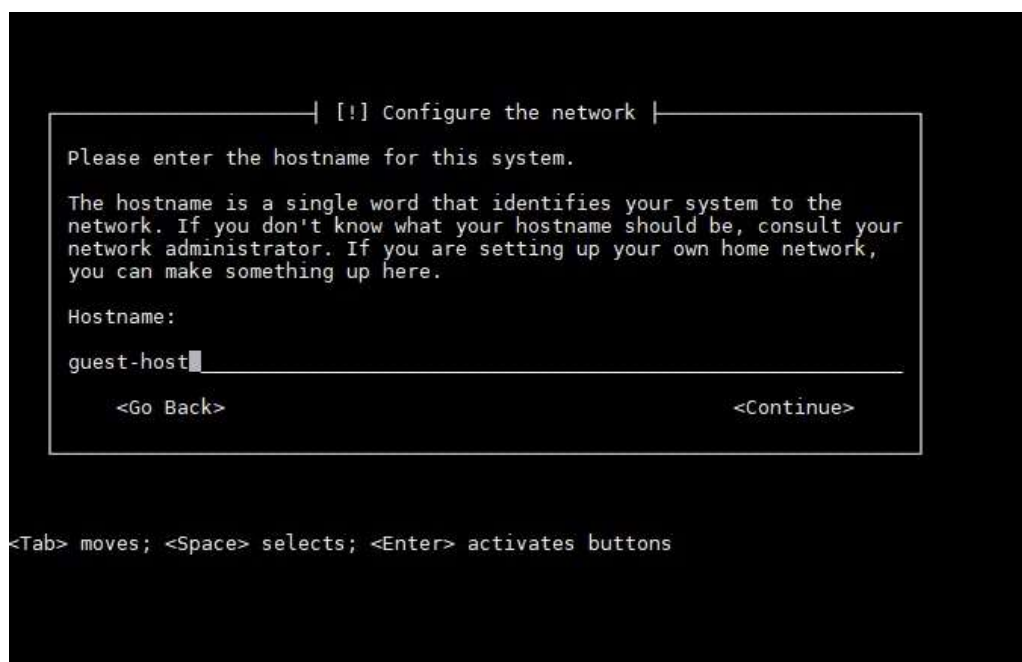
This creates our guest and opens up to its console.

From this point you will see a normal Ubuntu installation dialogue. This is pretty standard, keep everything to the default. Country set as New Zealand, detect keyboard, etc.

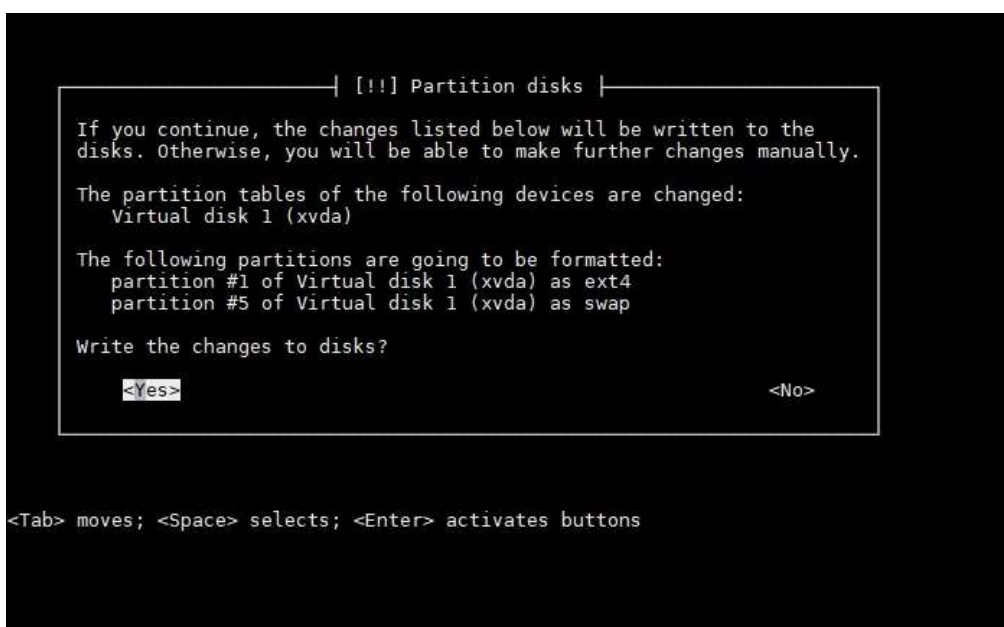
Except User and Password. I suggest *user* and *P@ssw0rd* for User and Password. It is up to you but make sure you remember what you have set.

There are a couple of other things you will have to set:

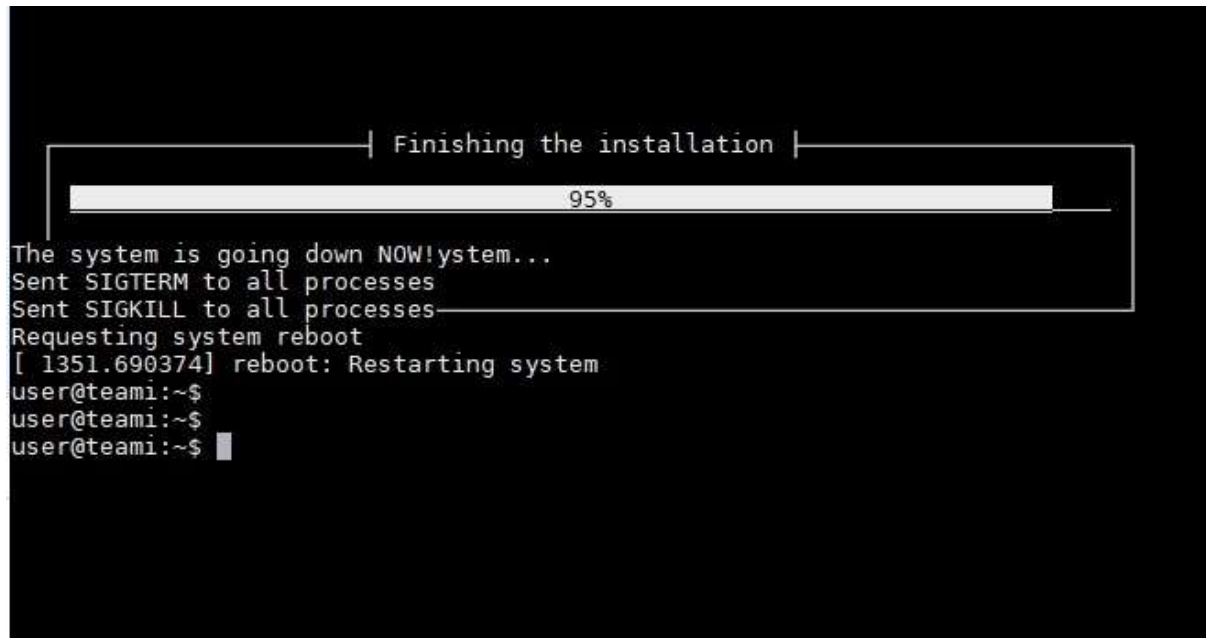
Hostname – guest-host (note: you cannot have ‘_’ in the name)



Partitions – use the entire disk



Once this has finished, the installer will reboot and you will be back in the Dom0 console.

A terminal window with a black background and white text. At the top, a progress bar is shown within a rectangular frame. The frame has a title bar that says "Finishing the installation". Inside the frame, a horizontal bar is filled with white, and the text "95%" is centered on it. Below the progress bar, the following text is displayed: "The system is going down NOW!system...", "Sent SIGTERM to all processes", "Sent SIGKILL to all processes", "Requesting system reboot", "[1351.690374] reboot: Restarting system", "user@teami:~\$", "user@teami:~\$", and "user@teami:~\$".

```
Finishing the installation
95%
The system is going down NOW!system...
Sent SIGTERM to all processes
Sent SIGKILL to all processes
Requesting system reboot
[ 1351.690374] reboot: Restarting system
user@teami:~$
user@teami:~$
user@teami:~$
```

Perform post installation finalisation of the guest image

Modify Config

Before we can use our guest we need to modify our config file again.

```
sudo vim /etc/xen/guest.cfg
```

Comment out, using `#`, the kernel and ramdisk lines (the ones we edited in the last step).

At the bottom of the file, add this line

```
bootloader = "/usr/lib/xen-4.4/bin/pygrub"
```

```
# =====
# Example PV Linux guest configuration
# =====
# This is a fairly minimal example of what is required for a
# Paravirtualised Linux guest. For a more complete guide see xl.cfg(5)

# Guest name
name = "guestvm"

# 128-bit UUID for the domain as a hexadecimal number.
# Use "uuidgen" to generate one if required.
# The default behavior is to generate a new UUID each time the guest is started.
#uuid = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"

# Kernel image to boot
#kernel = "/var/lib/xen/images/ubuntu-netboot/trusty/vmlinuz"

# Ramdisk (optional)
#ramdisk = "/var/lib/xen/images/ubuntu-netboot/trusty/initrd.gz"

# Kernel command line options
extra = "root=/dev/xvda1"

# Initial memory allocation (MB)
memory = 1024

# Maximum memory (MB)
# If this is greater than 'memory' then the slack will start ballooned
# (this assumes guest kernel support for ballooning)
#maxmem = 512

# Number of VCPUS
vcpus = 1

# Network devices
# A list of 'vifspec' entries as described in
# docs/misc/xl-network-configuration.markdown
vif = [ '' ]

# Disk Devices
# A list of 'diskspec' entries as described in
# docs/misc/xl-disk-configuration.txt
disk = [ '/dev/team1-vg/guest_volume,raw,xvda,rw' ]

bootloader = "/usr/lib/xen-4.4/bin/pygrub"
```

Press ESC, then `:wq`.

We have just told our VM to stop using the installer's kernel and start using the Xen's pygrub bootloader.

Restart VM

Now we need to stop our VM and start it again.

```
sudo xl shutdown guestvm
sudo xl create -c /etc/xen/guest.cfg
```

The screen may stop displaying text after about 30 seconds. Just press Enter a few times and the command prompt should show up.

This will boot in to our new VM. Log in with the credentials you set during the installation.

If you want to leave the VM, just press and hold ctrl and] although this may cause your command prompt to go weird.

```
Ubuntu 14.04.5 LTS guest-host hvc0
guest-host login: * Stopping System V runlevel compatibility          [ OK ]

Ubuntu 14.04.5 LTS guest-host hvc0
guest-host login: user
Password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-162-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

user@guest-host:~$
user@guest-host:~$
user@guest-host:~$ user@team1:~$ user@team1:~$ user@team1:~$ user@team1:~$ user@team1:~$
i:~$ user@team1:~$ user@team1:~$ user@team1:~$
```

You can still type but nothing shows. Just type
sudo reboot

```
user@team1:~$ user@team1:~$ user@team1:~$ user@team1:~$
Broadcast message from user@team1
(/dev/pts/1) at 16:41 ...

The system is going down for reboot NOW!
user@team1:~$
```

And your system will reboot. It may take a couple of minutes after the boot.

Make sure the VM is all up to date.

```
sudo apt update
```

```
sudo apt dist-upgrade
```

Now your VM is all up to date! This is important because all the clones will be copied from this VM. It will save you individually updating all the clones after making them.

Prepare volumes for cloned guests

It can be rather laborious to do the past few steps every time we want to make a new guest. We will now explore a way to use what we have already done to make new guests each time.

Stop VM

Make sure that the guest we made earlier is not running.

We will check this by showing a list of all VMs currently running.

```
sudo xl list
```

```
user@teami:~$
user@teami:~$
user@teami:~$ sudo xl list
Name                ID    Mem VCPUs    State    Time(s)
Domain-0            0    325     8    r-----   19.0
guestvm             1   1024     1    -b-----   0.1
user@teami:~$
```

If you see guestvm in that list, you should remove it by stopping it

```
sudo xl shutdown guestvm
```

Check the VMs list again

```
sudo xl list
```

You should only see Dom0 in the list.

Make New Volumes

Now we are ready to make new volumes for our clones

First we will take a snapshot of the system

```
modprobe dm-snapshot
```

Now we will look back at the first chapter of this tutorial and use very similar commands.

You will need to remember your volume group.

```
sudo vgs
```

```
user@teami:~$
user@teami:~$ sudo vgs
[sudo] password for user:
VG      #PV #LV #SN Attr   VSize   VFree
teami-vg 1   9   2 wz--n- 279.22g 248.66g
user@teami:~$
```

```
sudo lvcreate -s -L 1G -n clone1 /dev/< name of volume group>/guest_volume
```

```
sudo lvcreate -s -L 1G -n clone2 /dev/<name of volume group>/guest_volume
```

```
user@teami:~$
user@teami:~$
user@teami:~$
user@teami:~$
user@teami:~$
user@teami:~$ modprobe dm-snapshot
user@teami:~$ sudo lvcreate -s -L 1G -n guest-clone1 /dev/teami-vg/guest_volume
Logical volume "guest-clone1" created
user@teami:~$ sudo lvcreate -s -L 1G -n guest-clone2 /dev/teami-vg/guest_volume
Logical volume "guest-clone2" created
user@teami:~$
user@teami:~$
user@teami:~$
user@teami:~$
```

We have just created two volumes based on our original volume. The new volumes are only 1GB each because they each use the files from the original VM. The new volumes only need to store the difference.

Write Xen configuration files for the cloned guests

We need to have config files for both the new guests. We can copy the config file we made earlier and make some small modifications.

Make sure you are in the /etc/xen directory

```
cd /etc/xen
```

```
sudo cp guest.cfg guest-clone1.cfg
```

```
sudo cp guest.cfg guest-clone2.cfg
```

For each of the new config files we need to change some things.

The name in each config file needs to be unique. Example for clone1.cfg:

Set name – `guest-clone1`

Example for clone2.cfg

Set name – `guest-clone2`

We also need to change the starting memory for both config files

Set memory – `512`

Set disk to the path of the clone volume `/dev/<volume name>/guest-clone1,raw,xvda,rw`

Here is an example of guest-clone1.cfg

```
# =====
# Example PV Linux guest configuration
# =====
#
# This is a fairly minimal example of what is required for a
# Paravirtualised Linux guest. For a more complete guide see xl.cfg(5)
#
# Guest name
name = "guest-clone1"
#
# 128-bit UUID for the domain as a hexadecimal number.
# Use "uuidgen" to generate one if required.
# The default behavior is to generate a new UUID each time the guest is started.
#uuid = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
#
# Kernel image to boot
#kernel = "/var/lib/xen/images/ubuntu-netboot/trusty/vmlinuz"
#
# Ramdisk (optional)
#ramdisk = "/var/lib/xen/images/ubuntu-netboot/trusty/initrd.gz"
#
# Kernel command line options
extra = "root=/dev/xvda1"
#
# Initial memory allocation (MB)
memory = 512
#
# Maximum memory (MB)
# If this is greater than 'memory' then the slack will start ballooned
# (this assumes guest kernel support for ballooning)
#maxmem = 512
#
# Number of VCPUS
vcpus = 1
#
# Network devices
# A list of 'vifspec' entries as described in
# docs/misc/xl-network-configuration.markdown
vif = [ '' ]
#
# Disk Devices
# A list of 'diskspec' entries as described in
# docs/misc/xl-disk-configuration.txt
disk = [ '/dev/teami-vg/guest-clone1,raw,xvda,rw' ]
#
bootloader = "/usr/lib/xen-4.4/bin/pygrub"
```


Create the guest domains

We're all set now to run our guest clones.

```
sudo xl create /etc/xen/guest-clone1.cfg
```

```
sudo xl create /etc/xen/guest-clone2.cfg
```

When we omit the '-c' from the xl create command, it makes the VM but doesn't boot to the console.

```

user@team1:~$
user@team1:~$
user@team1:~$
user@team1:~$ sudo xl create /etc/xen/guest-clone1.cfg
Parsing config from /etc/xen/guest-clone1.cfg
user@team1:~$ sudo xl list

```

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	325	2	r-----	29.9
guest-clone1	4	512	1	r-----	3.2

```

user@team1:~$

```

Basic functionality of the running guests

Some basic commands to check that the VM clone is working like a real machine:

ifconfig – This command shows NIC information (same as ipconfig for Windows)

```

user@guest-host:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:08:73:01
          inet addr:10.118.0.166  Bcast:10.118.3.255  Mask:255.255.252.0
          inet6 addr: fe80::216:3eff:fe08:7301/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8724  errors:0  dropped:2  overruns:0  frame:0
          TX packets:48  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1430867 (1.4 MB)  TX bytes:4892 (4.8 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0  errors:0  dropped:0  overruns:0  frame:0
          TX packets:0  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

user@guest-host:~$

```

top – Shows running processes (same as Task Manager for Windows)

```
top - 17:25:36 up 13 min, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 61 total, 1 running, 60 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 499536 total, 232132 used, 267404 free, 8692 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free. 187892 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	33384	2648	1456	S	0.0	0.5	0:01.47	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworker/u2:0
7	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.02	rcuos/0
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	khelper
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xenwatch
17	root	20	0	0	0	0	S	0.0	0.0	0:00.50	xenbus

ping – used to check if devices can talk to other devices on networks. In this example, I pinged Google, which is an external address. This shows that the VM clone has internet access

```
user@guest-host:~$
user@guest-host:~$
user@guest-host:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=54.1 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=53.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=53.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=53.8 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4012ms
rtt min/avg/max/mdev = 53.867/53.959/54.132/0.194 ms
user@guest-host:~$
user@guest-host:~$
user@guest-host:~$
```