System Installation and Basic Administration

COSC301 Laboratory Manual

Ubuntu

Ubuntu (pronouced “oo-BOON-too [https://www.ubuntu.com/about/about-ubuntu]”, not “oo-BUN-too”) is a South African ethical ideology focusing on people’s allegiances and relations with each other.

Many of you might have installed an operating system before, probably Windows, possibly Mac OS X or Linux, or possibly even dabbled with other operating systems. The desktop versions of Windows and Mac OS X, which are aimed at the mass market, are designed to be very easy for the user, with minimal choice. Linux systems have been moving in that direction for a long time now, and are now almost as easy, but because Linux caters for a more technical audience, there is still plenty of options to choose from during installation, all the way from “easy and quite painless” up to “frustrating and error-prone”, depending on the distribution you wish to install. Ubuntu and Redhat are both at the “easy and quite painless” end of the spectrum.

Installation of these “easy and painless” systems is comparatively fairly boring, but none-the-less important, and so we have included reading material in today's lab aimed at getting you to think about operating system installation differently (eg. how might you install and maintain a computer laboratory?), as well as trying to get you thinking about how one installation might vary from another (eg. workstation versus database server versus web server).

In brief, today we shall be installing our server, which will be used for the rest of the paper. Here is an overview of what you will need to accomplish today:

1. You will need to create suitable documentation for what you are doing today, such that another person, with the same training as yourself, can follow your instructions and arrive at the same result. **This is the prime self-assessment for this lab.**

2. Create a virtual machine in VirtualBox.

3. We shall install “Ubuntu 22.04 LTS Server Edition” into our machine. This will be a command-line only environment, for reasons discussed later. We will aim at a suitable “first” system, where we don’t have much of an idea of exactly how we should approach some tasks, e.g. how to partition disk storage.

4. We shall ensure all system updates come from an appropriate source and are applied to our server.

5. We shall install the VirtualBox Guest Additions, and make use of the “Shared Folders” feature.

6. We shall remove some of our ignorance that we had when installing our system for the first time, and figure out just how large different parts of the filesystem are.

7. As time allows, you will have a look at some of the extra reading material which covers important background material concerning storage, security issues during installation and different options for installing systems.

To cater for students at different levels of experience, and to help you manage your time and workload more effectively, some material is marked as optional.

8. In the next lab, which deals with post-installation, we will apply further configurations to introduce it into its new network setting and start performing just some of the many things we would generally do after having just installed a server.

9. From this lab onwards, you should start your second assignment based on what you learnt from the labs.

1. Thinking about your Documentation

It is recommended you read this section before coming to do the lab.

The largest assessable, though not the largest deliverable, in this laboratory will be your documentation. You need to create sufficient documentation such that one of your class-mates could follow your instructions and arrive at the same result. It is also very important for you to repeat them while doing your assignments.

First, a few ideas about documentation:

- You don’t need to include default options, except perhaps where they are significant. In real-life, some defaults change between operating system releases, and so detailing important default options can be important.

- Documentation becomes faulty if it is not maintained, and thus a liability. Therefore, documentation needs to be kept up-to-date with a list of revisions. A list of dated revisions is important for quickly determining what change might have caused a problem. In larger environments, “Configuration Management” and “Change Management” procedures are formal procedures for ensuring that you can easily back out of a change if it causes problems, minimising downtime. Missing documentation is even more of a liability, particularly in complex systems: it makes it hard to determine what services another service might be dependent on, and thus the effect of a service failing.

- What does this system do? What has it done in the past? When reading someone else's documentation, it is very useful to appreciate the task that the system was installed for. A lot of servers have had a long life and may have accumulated a lot of cruft over years of service. Knowing what a server does, why a service is needed, how important it is, and the key configuration of it is important information for a team-member to put their hands on.

- Of course, putting your hands on such documentation indicates that such documentation should be easy to find. However, in many environments you find very disorganised standards regarding where documentation should be stored, how it should be edited, etc. One person might use a Wiki, one might use a paper exercise book, one might use Google Docs and another might use a Microsoft Word document. This is not helpful if you are a backup systems engineer.

  Given the amount of work that is done on servers remotely, a paper exercise book can be too easy to forget to update. Similarly, don’t store your documentation for a server on the same server, lest you can’t get to it when you need it. Collaborative editing can occassionally be useful, particularly as it results in fewer revisions of the document floating around at one time. Being able to access the document is obviously very important, and so if your documentation is kept “in the cloud” (such as on Google Docs), then it would also be useful to occassionally keep the most recent version printed, with its attendent synchronisation problems.

So, what should documentation contain? To illustrate, here is a (slightly edited) table of contents of some documentation we have maintained for one of our servers. Note that each server would be different, so take this simply as a guide. You will not yet understand all that presented here; that's okay, we've put a † next to those items that you aim to complete for this lab.

```
Management of the server ‘NAME’
  Administrator †           Who is the administrator, and their contact details.
  Release Information †     What OS and version is this machine currently running?
  Recent Changes †          On every change, this gets updated manually
    5 November 2017
    4 November 2017
    …
  History                    The systems history and artefacts
  Hardware †                 CPU, Memory, Location, Physical Access
  General Management
    Filesystems †            How are the disks partitioned?
    Network Interfaces †     How is the system connected?
      inside                 IP address…, Connected to…
      outside                If this host has two interfaces
    Administration Rights † How to get them
    Authentication           How is user-authentication managed?
      Password Requirements These are additional to the system default behaviour
      Password Expiration    These are additional to the system default behaviour
    Software Upgrades †      Where does the software come from. How are updates made?
    Time Synchronisation     Is the time synchronised with a time server?
    Cron                     What periodic jobs are run on this host?
      backup                 Every host should have something similar
      report-users-old       These are particular to this host…
      …
    Firewall                 Where is it defined, brief description
    TCPWrappers              Useful for limiting access to network services
    Log maintenance and monitoring
    Backup (client)          This is IMPORTANT
      Bare-Metal Restoration Preparation
                             DOCUMENT how to restore onto an empty disk
      Restoration Procedure [Last tested: 12 October 2014]
                             And TEST that it works, periodically
  Role Management            What services does this server house?
  Web Server
    How important is it?
    Who should have access?
    What is its “normal” behaviour?
    What are its major configuration changes?
    Any particular policies that need to be catered to?
  Local E-mail Server
    Another fairly standard service…
    …but perhaps you have some added monitoring which
    you should document.
  SSH Server
    Another very standard service…
    …but perhaps with some added service-specific notes.
  GIT Repositories for Research Students (via SSH)
    A lot of servers do fairly unique things, so be sure to document such things well,
    particularly with regard to any management tools that might be developed.
```

You may create your documentation anywhere you wish, so long as it is *not* stored on the server itself. You do not have to follow the outline above, but you do need to at least include the material marked with a †.

2. Adding Server1 to VirtualBox

In this section, we shall create a new machine in VirtualBox.

Before installing our operating system, we should first consider some differences between different releases of operating systems, such as server-class or desktop-class. Desktops are generally optimised for interactivity and foreground processes, and pay some attention to security without inconveniencing the user too much. In contrast servers are generally optimised for security rather than featuring the latest, cutting-edge software. Because core server daemons don't change much over time (compared to desktop software), servers don't need, and often don't run, desktop environments. If we take Ubuntu as an example we see that Ubuntu offers Long Term Support [https://ubuntu.com/about/release-cycle] (LTS) releases, which are supported for three years for the desktop version, and five years for the server product. Non-LTS releases are only supported for six months.

Now let's get on with installing a server. For the rest of this section, please pay close attention, as it can be easy to skip along and miss out an important task.

Procedure 1. Creating the Virtual Machine “cosc301-server1”

1. In VirtualBox, click on Machine→New to start creating our new virtual machine.

   You should now see the “Create New Virtual Machine” wizard. Click Continue

2. Give the virtual machine the name “cosc301-server1”, which is what VirtualBox will call it. Giving our virtual machines a common prefix can be useful when sorting them (consider the case if you have other Virtual Machines for other papers).

3. Click on the ISO Image dropdown and after clicking 'Other', from the same folder you found the iso used when creating Client2, select: "ubuntu-22.04.1-live-server-amd64.iso"

   After this click "Skip unattended installation".

4. In the "Hardware" tab, choose the default value 2048MB as the amount of memory. Remember, default values are only suggestions. It's not uncommon to come across machines with a decent amount of memory. The machines in the lab have 16GB. Plenty of space to give our virtual machine 2048MB. Keep in mind though, there may be minimum requirements for the operating system and any applications you may be running. We are only going to be using a server (and so no GUI) and no applications that require large amounts of memory. So we'll go with the default.

5. When configuring the virtual "Hard Disk”, make sure Create a virtual hard disk now is selected, as we do want to create a new hard disk for the virtual machine. Make sure that "Pre-allocate Full Size" is NOT checked. These should be the defaults, but at the moment we just want to be very careful. Click Finish.

We have now created and configured the virtual hardware for our server. But we still want to update one of the settings for the new virtual machine. In the VirtualBox window, click on the machine called “cosc301-server1”, and then click on Settings to access the settings dialog.

Click on Audio and *un-tick* the Enable Audio. This is because a server doesn’t need audio, and it introduces a bit of complexity that could otherwise cause a problem, such as the virtual machine crashing due to some audio-related bug.

There are some other configurations we will do, but we shall leave those until the relavent sections, to better explain them. Do have a brief look around all the other parts of the Settings, but don’t make any other changes, lest you cause it to behave differently from what we expect. Close the Settings dialog by clicking OK. You are now ready to install the operating system.

Disk or disc?

Just in-case this causes you confusion, *disc* refers to the round, flat CDs and DVDs, while *disk* refer to things like magnetic hard-disks and floppy-disks. Interestingly enough, disks contain discs, but don't resemble a geometrical disc on the outside. It can be all terribly confusing, and the Usage Note at dictionary.com's entry for "compact disk" [http://dictionary.reference.com/browse/compact%20disk#sharethis] gives some explanation, but you'll probably find it also depends on whether you use British or American English.

3. Installing Server1 with Ubuntu 22.04 LTS Server

We are installing a Linux server, and like any Unix-like server system, it does not require a graphical environment. Indeed, there are good reasons for not *wanting* a graphical environment. Chief among these is complexity. Complexity is the often-times enemy of stability and security (ie. things are more likely to fail in ways that could easily disrupt the rest of the system). A graphical interface also consumes rather a lot of system resources, and we would typically want background processes to have preferential treatment, so it would also be rather sluggish.

Graphical user interfaces are also not very scalable in terms of management operations *on the system itself*. Graphical user interfaces can however be wonderful when *coordinating* management operations across a large number of machines.

Because this is likely the first time for you (although for a number of you, you might have been down this road a number of times already), we shall aim for a "first" system; one where we don't really have any experience on which to base some of our decisions. These decisions include questions relating to how much memory should have been allocated to your system, how should you disk be partitioned into filesystems and what software should you install.

We've already told you how much memory you should allocate to your server, although later on we shall see how much is actually used. Figuring out how best to set up your storage (partitioning your disks and formatting the partitions with a filesystem) can be somewhat onerous, and it will often depend on what you will be doing with a particular machine. There are plenty of guidelines, but without seeing for ourselves how large different parts of the filesystem should be, it is not particularly useful, so we shall let Ubuntu help us out with some defaults. Finally, with Ubuntu, the choice of which software to installed is made very very simple, to a point where it can be a bit more annoying if you know exactly what you want, so we can basically leave that question aside for now.

So, our "first" system is basically going to be installed practically using all default values, then we shall have a look at what the installed system looks like, and start gaining some experience which we could then use to reinstall the server — although we're not going to reinstalling the server today because we don't have time.

"Plan To Throw The First One Away"

This is a quote from a famous software engineer by the name of Fred Brookes, who wrote a series of well-regarded essays on Software Engineering called the *Mythical Man Month* (ISBN: 0201835959). This saying is perhaps more true of installing an unfamiliar operating system than it is of software engineering, as you will inevitably learn more about the system and realise that you could have made wiser decisions when you installed the system.

Important

As you proceed through the installation, be certain to record suitable documentation as instructed in a previous section.

Procedure 2. Running the Ubuntu Installation

1. In the VirtualBox main window, click on your new server, which you called “cosc301-server1”, and click on Start to start the virtual machine. Choose "Try or Install Ubuntu Server". It should soon prompt you what language to install. Choose English by pressing the **return** key. When asked about updating the system, choose Continue without updating.

2. The installation should start, and you should be asked a a question regarding keyboard. Select the default by pressing the **return** key. When asked, choose Ubuntu Server (not minimal).

3. Ubuntu will attempt to configure the network interface using DHCPv4. You can press **return** to accept the default setting.

   Then you will be asked to configure a proxy server, which we don't need. So just press **return**.

   Regarding Ubuntu archive mirror, accept the default by pressing **return**.

   Then you will be prompted with filesystem setup. Just choose the default Use An Entire Disk by pressing **return**. Then choose the only virtual disk as the local disk for installation by pressing **return**.

4. Now comes what could be our first major set of decisions: partititioning. We only have one disk in our virtual system (we could have made others, if we wanted to experiment with different disk management strategies, but that is not what we want at this stage).

   We could manually create a bunch of different filesystems, but at the current stage, you don’t know enough to appreciate it and you may end up making some parts of your system too small and have to repeat the entire the procedure all-over again; remember, we’re creating a “first” system.

   So we just accept the default partitions by pressing **return** on Done. The default has two partitions. One partition is for root filesystem /, which has a type of “ext4”, and this is where all of our system will be installed. Another partition is for grub, the booting software.

   For the next Confirm destructive action notice, just choose Continue.

5. Now you will be asked a number of important questions. The first question is the name of the first user. It is the administrative user account. This is a normal user who is also given the power to use the **sudo** command. Use the full name “Miss A. Laneous”. The server name should be “server1”, which is consistent with the name the machine will have on the network.

For the username, we shall continue our own standard nomenclature by using a username "mal". The password will be `Quack1nce4^`.

6. Then you will be asked to install SSH or not. Don't install it at the current stage. We will install and configure it in a later lab.

   Next you will be asked about Featured Server Snaps. Don't install any such package as the default installation will be sufficient for us.

7. Now begins the wait of around 10 minutes, while the "base" system is installed. This installs a minimal bootable system onto the disk.

Tip

While it is installing, go and read some of the other sections of this lab, and return to here when you are ready to continue.

8. Installation complete! After being prompted with this message, reboot the server. Then you will be asked to remove the installation medium (just press **Return**). The system will reboot, and it should reboot into your new system. This is generally referred to as "the moment of truth" because you are seeing if your installation actually worked.

9. You should see the following on screen, after some initial startup messages:

```
Ubuntu 22.04 LTS server1 tty1

server1 login:
```

   This is your login prompt. Use Ctrl+Alt+F3 to choose a different window to login as the first presented window is the console which has various log messages appearing in it. You can now login as the user "mal" with the password `Quack1nce4^`. After a bit of processing, it will greet you with some information regarding the system statistics (load, memory user, number of processes, etc.) as well as tell you how many packages can be updated.

10. Update the apt software packages with the following commands.

```
$ sudo apt update
$ sudo apt upgrade
```

11. Install the ifupdown and net-tools software packages, which allow us to configure network interfaces with **ifup** and **ifdown** commands, and also use ifconfig.

```
$ sudo apt install ifupdown net-tools
```

12. Edit the configuration file `/etc/netplan/00-installer-config.yaml` to disable **netplan**. Initially the network in server1 is managed by **netplan**. In this paper, we are using the ifupdown package to manage the network interfaces. To disable **netplan**, change `/etc/netplan/00-installer-config.yaml` as below.

```
network: {config: disabled}
#network:
#    ethernets:
#        enp0s3:
#          dhcp4: true
#    version 2
```

Basically we uncomment the line for disabling the network and comment out the lines that set up the network interface `enp0s3`.

13. Since **netplan** is disabled, in the booting process of the system, there is a oneshot system service **systemd-networkd-wait-online** waiting for the network to be configured. This is annoying as it waits up to two minutes until giving up. To disable the service to prevent the system from waiting on a network connection, use the following command:

```
$ sudo systemctl disable systemd-networkd-wait-online.service
```

You also need to use the following command to prevent the service from starting if requested by another service:

```
$ sudo systemctl mask systemd-networkd-wait-online.service
```

Mind you that, though masking systemd-networkd-wait-online.service, as suggested above, may help fix the waiting, all other services depending on it will fail. That means, all services needed to wait until the network is online will fail. However, fear not! you will manually configure the network correctly in the rest of the lab.

4. Disabling the Unaccelerated Framebuffer

If you start playing with the system now, you'll quickly find that it feels very slow, particularly when output is scrolling on screen. That's because Ubuntu is using something called a "framebuffer", which allows it to display output suitable for all the different human languages that Ubuntu supports. In our case, we don't need such enhanced support, and if it were to use the standard old VGA interface, it would be much faster, so we shall turn it off to get our performance back.

Edit, using **sudo**, the file `/etc/default/grub`, and uncomment the following line by removing the #:

```
#GRUB_TERMINAL=console
```

Now, as root (using **sudo**), run the command **`update-grub`** in order to affect the change. This will prevent GRUB from trying to use a framebuffer but will not prevent Ubuntu from setting one up later. To prevent that, as root, add the following line to the end of the file `/etc/modprobe.d/blacklist-framebuffer.conf`:

```
blacklist vga16fb
```

Note

In this case there is no command you need to run in order to affect the changes. This was not always the case in earlier versions of Ubuntu or Debian. Indeed, figuring out how to disable the framebuffer can be an exercise in frustration, as there have been many ways this could be done in the past, and many don't work today.

Now reboot (**sudo reboot**) and when it comes back up, you should notice the window is slightly different shape, and when you run a command that produces a lot of output (such as **dmesg** to output the kernel logs), it should scroll very very quickly.

5. Disabling Cloud-Init

By default, Ubuntu ships with cloud-init. It is the industry standard multi-distribution method for cross-platform cloud instance initialisation. Since you are not doing a cloud setup here, it can be safely disabled; otherwise the terminal will be filled up with messages from cloud-init. Also what you are doing in this paper is much more than cloud-init can offer, as you are effectively configuring a much more powerful cloud instance at a much lower, more manageable level in Ubuntu/Linux systems.

To disable cloud-init, simply run this command:

```
sudo touch /etc/cloud/cloud-init.disabled
```

6. Connecting to the Network

Okay, so at this stage, we should have created the (virtual) hardware for Server1, installed it with Ubuntu and reclaimed some performance. But the network interface `enp0s3` should be down as **netplan** is disabled. Now we're going to change the interface name to a proper name that makes better sense and is easy to remember, and to configure it to connect to the Internet again.

Procedure 3. Connecting to the Network

1. First, check the detail of the interface `enp0s3`, which is not "UP" yet.

```
$ /sbin/ifconfig enp0s3
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:e3:4d:42
          no lines saying "inet" or "inet6"
          BROADCAST MULTICAST  MTU:1500  Metric:1
          …
```

2. Because we will occasionally need access to the Internet in order to get software packages, and for other tasks, we shall be keep our current network interface attached to the Internet, and later on we shall add another interface that we shall use for offering services to our internal network. To reduce confusion, we shall rename our current interface from "enp0s3" to "outside".

   Rename the interface by creating an appropriate `/etc/systemd/network/70-outside.link` file, as we practiced in an earlier lab about basic interface management (ie. adjust the "Name" and "MACAddress" parameters).

   You will need to create this file for both of your adapters. You could use the name of the network you are connecting to as the name of the link file to help you keep track of what's going on.

Warning

Don't forget to run **sudo update-initramfs -u**.

This command must be run every time changes are made to systemd configuration files.

Reboot when you have completed the edit and ensure the renaming has worked by listing the interfaces with **/sbin/ifconfig -a**. At this stage, it should not yet have an address.

3. Our outer interface is connected to the VirtualBox "NAT" attachment, and so should be configured using DHCP. To affect this, edit the file `/etc/network/interfaces`, *adding* the following lines.

```
auto lo
iface lo inet loopback

auto outside
iface outside inet dhcp
```

It's as easy as that. Now test that you can get an address using DHCP by bringing up the interface:

```
$ sudo ifup outside
… You will see a bunch of output
The following line can be ignored, as the file will be created…
chown: failed to get attributes of `/et/resolv.conf': No such file or directory
bound to 10.0.2.15 -- renewal in 36648 seconds. Success!
```

4. As a final "moment of truth", reboot the virtual machine (**sudo reboot**). After you log in again, check the interface details:

```
$ /sbin/ifconfig
outside   Link encap:Ethernet  HWaddr 08:00:27:e3:4d:42
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0    Success!
          inet6 addr: fe80::a00:27ff:fee3:4d42/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2070 (2.0 KB)  TX bytes:1788 (1.7 KB)
```

7. Software Updates

Now we should have a freshly installed, updated machine that is connected to the network. This represents some of the very first things we would do to a new system post-installation; we shall do some further work in the following lab.

For software updates, the Ubuntu software is mirrored all over the world, and many people even provide their own mirrors or caches for their own local network. However, we are going to use the default APT proxy-cache configured in `/etc/apt/sources.list`. This allows us to avoid problems like unavailable mirrors/caches or non-active caches. It may cause more global Internet traffic which is not an issue anymore with today's Internet.

To apply all of the available updates, use the following command.

```
$ sudo apt-get dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
```

```
The following NEW packages will be installed:
  … a few
The following packages will be upgraded:
  … many
72 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 73.8MB of archives.
After this operation, 104MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
… a lot of work begins downloading and applying updates
```

We'll cover package installation more in the lab on post-installation, but the key thing to note here is that first we used **apt-get update** to update Server1's knowledge of what packages are currently available, and then we used **apt-get dist-upgrade** to perform a major upgrade of any packages. Typically, we would only use **apt-get upgrade** for day-to-day upgrades, but since this first update after installing could contain more significant[1] changes, it is best to use a command such as **apt-get dist-upgrade**.

apt-get and Friends

There are multiple tools similar to **apt-get**. For example, **aptitude** is a bit smarter, and generally replaces **apt-get**; **synaptic** is like a graphical tool similar to **aptitude**, and there are still others. This is not a course in Debian administration tools, so we shall just stick with **apt-get**.

While that is completing, let's look again at what we put in that `sources.list` file. Here's the general format:

```
deb URI release section …
…
```

`deb`
: Each line (ignoring comments and blank lines) will have either a `deb` or `deb-src` to describe either a binary or source package respectively. We shall only be dealing with binary packages, source packages are not often used. `.deb` is the standard file extension for a Debian package.

`URI`
: A URI (or URL if you prefer) specifies where such packages may be found, and will generally specify a method (such as via HTTP, FTP, or locally available on a CD-ROM or elsewhere in the filesystem), and usually a host for network-oriented access methods. In our particular entry, we also specified an optional port number, since the default port number for HTTP is port 80. We also needed to say where on the server the Ubuntu "archive" can be found; typically this will be `/ubuntu`, but that is not a requirement.

`release`
: This basically specifies the *version* of Ubuntu we are interested in installing. Because this is typically a moving target, we have three versions which we typically use, although five are available:

  `Jammy` is the codename for Ubuntu 22.04 LTS, so the `Jammy` version contains whatever is in the latest version of the official Ubuntu CDs. All Ubuntu hosts will have at least this.

  `Jammy-updates` are major software updates, that may add (or remove) features, but are not security updates. They represent the changes between "point" releases, such as

[1]Meaning that it would cause other previously-not-installed packages to be installed, or some existing packages to be removed.

"14.04" and "14.04.1". You can choose whether you want to have these installed or not; they may cause unplanned downtime due to changes.

`Jammy-security` are security updates for software that the Ubuntu Security team has put together. All machines should include this release.

`Jammy-backports` represent software that has been packaged for newer versions of Ubuntu, and has also been "back-ported" into the Jammy release. This is useful only if you need something on a Jammy box that is not available in Jammy, such as a particular feature of some software that is available in Ubuntu 14.10 but not 14.04. It is not commonly used.

`Jammy-proposed` is generally for people wanting to test updates (ie. developers and people who really need a bug fixed) to test something about to be put in `Jammy-updates`. It is not intended for general consumption.

Further, different versions of Ubuntu will have different code-names, so for example 18.04 LTS is "Bionic Beaver"(bionic), and 22.10 is "Kinetic Kudu" (kinetic). The codenames are in alphabetical order.

`section …`

The sections always have the names `main`, `restricted`, `universe` and `multiverse`.

These sections have different restrictions with regard to licencing and support. Everything will have at least `main`. A desktop system will commonly have all of them to incorporate software for which there is no good open-source product available. Wireless drivers often require the `restricted` section, as they are generally binary-only or have some firmware with a restrictive copyright. Only `main` is looked after by the Ubuntu Security team, so if you don't need something in the other sections, it is good to omit those sections.

`universe` contains a lot of other useful programs that you would often want, so you can generally include that also.

Okay, hopefully by now you will have finished performing the updates, which will have very likely installed a new version of the kernel; after which we should reboot into the new kernel. Note that you should choose the first kernel after reboot.

```
$ reboot
```

You can now proceed onto the next section to help integrate your virtual machine with the host.

8. Installing Guest Additions

In this section, we are going to install the VirtualBox Guest Additions, which enable the guest (virtual machine) to have some greater integration with the host. Historically, this used to be necessary in order to make use of the shared folders feature (allowing us to easily share files between the guest and the host) however this is no longer the case.

On a desktop system, it would also enable clipboard integration for copy-paste, graphics performance enhancements as well as a number of other things. On other virtualisation platforms, such guest additions are also used, and can help with host memory utilisation, and enhanced performance using specialised drivers for virtual network cards etc, so in general guest additions are useful for all classes of guests.

Note

If you do not have enough time, you can skip this section, which should not affect the rest of the lab. It is recommended that you come back to this section in future when you have time.

Procedure 4. Installing the Guest Additions

1. To install the guest additions, with the VirtualBox window titled “cosc301-server1 [Running]” in the foreground, select Devices → Insert Guest Additions CD image… from the VirtualBox menu at the top of the screen. This inserts a virtual CD (an ISO image) into the virtual CD-ROM drive of the guest.

2. Because Ubuntu Server will not automatically mount the disc with the Guest Additions on it, we shall have to mount it manually. “Mounting” a filesystem means to attach it and make it available somewhere under the filesystem hierarchy.

   When we performed this step, we discovered that Ubuntu Server did not have the filesystem table (file `/etc/fstab`) configured with instructions where to mount the CD-ROM device, so we shall add that now. *Add* the following line to the bottom of `/etc/fstab`:

```
/dev/cdrom /mnt/cdrom iso9660 ro
```

   After creating the mount point "/mnt/cdrom", you should be able to mount the filesystem which is on the CD-ROM, into the system’s filesystem, making its contents available under the directory `/mnt/cdrom`:

```
$ sudo mount /mnt/cdrom
$ ls /mnt/cdrom
32Bit        runasroot.sh                VBoxWindowsAdditions.exe
64Bit        VBoxLinuxAdditions.run      VBoxWindowsAdditions-x86.exe
AUTORUN.INF  VBoxSolarisAdditions.pkg
autorun.sh   VBoxWindowsAdditions-amd64.exe
```

   Now we need to run the **VBoxLinuxAdditions.run** command, which will build and install the Guest Additions appropriate to our particular Linux kernel and environment.

```
$ cd /mnt/cdrom
$ sudo ./VBoxLinuxAdditions.run  run as root!
Verifying archive integrity... All good.
…
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel
modules. This may take a while.
…
This system is currently not set up to build kernel modules.   Oops!
Please install the gcc make perl packages from your distribution.
… some more messages
```

   According to the instructions above, we need to install a set of building tools, so run the command below.

```
$ sudo apt-get install build-essential
…
Do you want to continue [Y/n]? Y
Get:1 …
…
Selecting …
Preparing …
```

```
Unpacking …
…
Processing triggers …
…
```

Okay, so now hopefully all the packages will be installed which will allow the Guest Additions to install correctly. Let’s try and reinstall the Guest Additions now:

```
$ cd /mnt/cdrom     You’re probably still there
$ sudo ./VBoxLinuxAdditions.run
Verifying archive integrity... All good.
…
Removing installed version 7.0.6 Guest Additions...........
…
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel
modules. This may take a while.
this takes a while, don’t panic
you will also see warnings about missing libXt libraries (since X isn't installed)
…
VirtualBox Guest Additions: Building the modules for kernel 5.15.0…  Success!
…
VirtualBox Guest Additions: Running kernel modules will not be replaced until
the system is restarted
```

3. Now you are finished installing the Guest Additions. Reboot the server machine.

Procedure 5. Setting up Shared Folders

1. In this procedure, we shall configure VirtualBox to share a folder on your hosts desktop (we shall call it `VBoxShare`) with the guest. This allows files to be moved into and out of the virtual machine to the host, without the need for any networking.

2. Click on “cosc301-server1 [Running]”, then click "Settings". At the bottom of the Settings window, you will see Shared folders. Click it.

   Click on the folder icon with a green plus icon; this will add a new entry. In the Folder Path drop-down box, select Other… and navigate to J: drive. Click on "New Folder" and create a folder called `VBoxShare`. Then click on "Select Folder".

   > **Tip**
   >
   > It can be useful to share a single folder among many guests.

   Because we want this to always be available, tick the box labelled Make Permanent, and then click OK. Click OK again to close the Shared Folders dialog.

   We have now configured our host and the VirtualBox virtual machine with the shared folder, but the operating system inside the virtual machine still needs to be configured to do something useful with it.

3. Add the following line to `/etc/fstab`:

```
VBoxShare /media/host vboxsf defaults,uid=mal,gid=mal
```

   The first field (`VBoxShare`) is referring to the Folder Name that was specified in the previous step. We are going to make it available on the directory `/media/host`, which we shall very soon create. Because we want our regular user to have convenient access to it, we shall specify that everything is owned by the user and group called “mal”.

4. Create the directory that shall be used to access the filesystem (this is typically called a “mount-point”).

```
$ sudo mkdir /media/host
```

5. Add the text `vboxsf` to the file `/etc/modules`. This causes the `vboxsf` kernel module to be loaded earlier in the boot process so that it is ready for trying to mount the shared folder when the computer boots.

6. Time for the moment of truth. Reboot and ensure that it still comes up smoothly:

```
$ reboot
```

7. Time to test. Log in as the user “mal”. Inspect the currently mounted filesystems and see if /media/host is mounted:

```
$ mount | grep VBox
VBoxShare on /media/host type vboxsf (uid=1000,gid=1000,rw) Success!
```

   Now create a file inside `/media/host`:

```
$ echo “Hello” > /media/host/hello.txt
There should be no error message produced.
```

   On your host’s desktop, go into the `VBoxShare` folder and check that a file called `hello.txt` is present and that it contents match.

8. Now test in the opposite direction. Copy a file (such as a screenshot) from the host into the `VBoxShare` folder on the host. Inside the virtual machine, use **ls -l** to inspect the permissions. If you see something like this, then it is working well and you can go onto to next section.

```
$ ls -l /media/host
…
-rw-r--r-- 1 mal mal         6 2010-12-09 11:55 hello.txt
-rw-r--r-- 1 mal mal    288593 2010-12-09 11:56 some-picture.png
…
```

Use this for Keeping your Work Handy

Students often want to take the work they have done in these labs and take it home to set up their own home network. You will find this shared folder a reasonably convenient way to export files so you can move them from the VBoxShare folder to removable media etc.

9. Fixing some of our Install-Time Ignorance

You may have noticed the time zone is not quite right. Use **sudo dpkg-reconfigure tzdata** to set the right time zone.

Now we shall have a brief look at how large various parts of the filesystem are on our system, and how much memory is currently used. This will give a baseline for a Ubuntu Server, which we can then add-to when considering requirements for further installations.

First, let’s just take a peek at memory utilisation:

```
$ free -tm
         total       used       free     shared  buff/cache  available
Mem:      1976        170        803          1        1001       1639
Swap:     2047          0       2047
Total:    4024        170       2851
```

The above output gives the information of memory in the virtual machine, which includes both physical memory and swap memory. For example, we have total physical memory of around 2048MB (1976MB excluding a small amount that the kernel permanently uses for itself at startup). Currently 170MB is used, 803MB is free. 1MB is used as shared memory between processes, and 1001MB is used as memory buffers for file systems. In file systems, to improve performance, memory buffers are used to keep the disk file contents in memory. Some memory buffers/caches are reclaimable but others not. If a memory buffer is reclaimable, it is counted as available along with the free memory. In total there is 1639MB available including the free memory in the above output.

Note also that the Used Swap is currently 0. We want this to be very little (you may find a few MB are used, but so long as its not actively accessing swap frequently, that’s okay).

So what processes *are* running on a basic Ubuntu Server with VirtualBox Guest Additions installed?

```
$ pstree
init─┬─VBoxService───6*[{VBoxService}]
     ├─atd
     ├─cron
     ├─dhclient3
     ├─6*[getty]
     ├─login───bash───pstree
     ├─rsyslogd───2*[{rsyslogd}]
     ├─udevd───2*[udevd]
     └─upstart-udev-br
```

pstree is a useful little command that shows all of the processes in an easily digested way. It is not as available as the venerable old **ps** command, but it is by far easier to use for casual purposes. Currently, you don’t need to understand what everything is for, but it’s useful to develop a feel for what is “normal” in a system… documenting what is “normal” behaviour can also be useful.

Okay, so that’s memory and processes, let’s now have a quick look at network activity and then we’ll look at the filesystem. We will practice this plenty of times later on, but here is how we can see what processes are listening for network connections:

```
$ sudo lsof -Pni    as root!
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient3 538 root   4u  IPv4   3066      0t0  UDP *:68
```

Ubuntu has a policy of not running with any network-reachable processes by default, and that certainly seems to be the case, as the only thing listening on the network is **dhclient3**, which is the DHCP client – that’s how we got our network address, so it needs to be running. This sort of thing is useful to know for later on when you begin securing your operating system.

Okay, time to look at the filesystem usage. First, how much is actually used?

```
$ df -h      Mnemonic: disk full
Filesystem            Size  Used Avail Use% Mounted on
```

```
/dev/sda2             9.8G  4.0G  5.4G  43% /
… The rest can be ignored, they are virtual filesystems
```

Okay, so of the single data partition that Ubuntu Server created for us when we installed, 43% is used. How is that usage distributed through the filesystem?

```
$ sudo du -xm --max-depth 2 / | awk '$1 > 2'
14      /opt/VBoxGuestAdditions-7.0.6
14      /opt
214     /usr/share
134     /usr/bin
31      /usr/sbin
25      /usr/include
141     /usr/src
1955    /usr/lib
4       /usr/libexec
2500    /usr
97      /var/cache
28      /var/log
368     /var/lib
492     /var
6       /etc
5059    /
```

What that command does is simply to report the "disk usage" (**du**) in megabytes (`-m`) of everything in the root (`/`), without going across onto other (typically virtual) filesystems (`-x`) and without recursing further than two levels. The **awk** command is being used to filter the output, only outputting lines that have a value in the first field (megabytes used) greater than 2.

We have noted those lines that are given for files in the same partition. However, for those files that are not in the same partition, **du -x** does not show. For example, `/usr/local` and `/home` often have their own partitions so they are not shown in the above lines. Therefore, the final entry only shows the total for the current entire partition for the root (`/`) filesystem. But if you wanted to put `/var` on a separate partition, you would need to subtract the usage of `/var` from the usage of `/` in order to get accurate estimation of the usage of of `/`.

It is important to realise that this is only a baseline, and as a system grows, parts of the system will increase (particularly in places such as `/var`, and will differ a lot from one system to another depending on the nature of the services being run, so you would need to repeat this process after you have developed your services and run them for a while.

Right, we've done a lot in this lab, so let's move to self-assessment.

10. Self-assessment

There has been plenty of things to do today, but there are only two assessments for this lab. The first is to perform the installation steps shown previously, and second (which you should do at the same time) is to document it.

11. Appendix: Deploying Many Machines

This is optional material that is made available for those students who find nothing particularly new in installing Ubuntu. There is no work that needs to be done, but provides some valuable further insight into some real-world related tasks.

If you install multiple, or many, machines by hand, you are wasting a lot of valuable time, plus you're much more likely to make a mistake. Thus, we need a way of being able to get a working system onto many machines in a short amount of time. This is what operating system deployment is all about.

11.1. Deploying Using Disk Images

Perhaps the most common paradigm for deploying workstations is that of the *disk image*. This is probably due to the predominance of Windows in the IT space, and the experience IT staff have with this paradigm. Norton Ghost is a very well-known piece of software for the creation and deployment of disk images, which has been around for many years and supports nice features such as multicast image distribution and also now supports Linux filesystems. A selection of free disk imaging products can be found at The Free Country [http://www.thefreecountry.com/utilities/backupandimage.shtml].

As imaging goes, there are basically two common methods: *sector-based* and *file-based* images. Sector-based images are made by copying each sector on disk, and so don't care about what file-system is used on top of them. File-based imagers need to understand the particular filesystems involved. File-based imagers create smaller images and can offer richer features (such as knowing what files not to copy) but can be more complex and even more timeconsuming.[2]

Imagers work somewhat as follows: first the machine to be *imaged* is prepared, typically removing any temporary files, defragmenting the filesystem and zeroing out any free-space; file-based imagers can often do this for you. This is all done to reduce the size of the resulting, typically compressed, image.

On Linux and similar machines, which use filesystems that automatically defragment as needed, both defragmentation and zeroing of free-space can be done using the following commands as root: **dd if=/dev/zero of=/largefile; rm /largefile**[3] The command works by writing a file containing zeros to the disk, the file will be written until all the space is consumed. Near-filling of the filesystem is one of the events that causes a defragmentation event on filesystems such as ext3. As a side effect, the parts of the filesystem that were free-space are now filled with long strings of zeros (NUL bytes) which compress very nicely. Deleting the file doesn't cause these blocks to be overwritten. As another side-effect, any old data blocks will have been overwritten, but you should not rely on this for security.

Apple have made this easy in Mac OS X. In the Disk Utility program, you can use the Erase Free Space… Unlike the basic naïve **dd** approach, you can securely remove old data this way.

Most imaging programs understand the Windows file-systems sufficiently well enough to ignore those data-blocks that are not currently allocated. Windows machines will need to have a tool called SysPrep run on them to remove their identity, otherwise after deploying the image, multiple instances of the same computer (same identity) will show up on the network, which causes problems. This identity is called the Security ID or SID.

The imaging program is not run from the system you are imaging, rather you boot from another system, either from a bootable CD or floppy-disk, or you can remove the hard-disk to be imaged, and plug it into a different machine as the non-boot device. Apple have made this very easy, as Apple Macintosh machines have a feature called *Target-Disk Mode*[4]. You boot

[2]So **dd** is a very naïve sector—based imager whereas **dump** is a file-based imager—we shall see them in a later lab about filesystems.
[3]This assumes that there is only one filesystem on the disk. If there are multiple, the command should be run replacing `/largefile` with `/mountpoint/largefile` for each local filesystem.
[4]An excellent example of how the legacy BIOS impedes PC innovatation, and a sign of things to come with EFI.

the machine to be imaged, holding down **T**. Now the storage devices on the machine can be accessed as a Firewire storage device, and can be imaged from a neighbouring Mac. A very nice feature, especially for laptops.

When you create the image file, you generally store it on a network, on a machine with plenty of hard disk space, and is compressed as it is created. This is particularly true for Windows deployments, because with Windows, you generally need one image for each hardware configuration.

Deployment can be done individually, typically booting the machine to be imaged either over a network, CD or floppy, or mounted on another machine. The process is then run in reverse, writing the disk image contents onto a selected hard disk.

However, because you often want to deploy many machines at once, it can be much easier to boot each into the imaging software, such as Norton Ghost, and then *multicast* the image from the imaging server (generally on the same subnet as the clients) to all the clients at once, which reduces network traffic. This assumes that your network can handle multicast traffic; many cannot[5].

One advantage of disk images, that Apple takes advantage of, is that you can use them for network booting. Apple calls this *NetBoot*, as opposed to *NetInstall*, which is for installing (er, deploying) an image over the network to the hard disk. With NetBoot, clients boot from the network, and don't require a hard disk, but rather run from the read-only image on the network. This is useful when you have a lab of Macs that need to run a particular environment for a short period of time, or when you want to reset the machine to a known state very easily. It does require a fast network though. Each client will typically have some writable storage on the server (called a "shadow file") so the entire computer still feels writable, and users will have home directories on the network.

Disk images are most commonly used for deploying a minimal base-system. Other technologies are often used in tandem for deploying applications and managing configuration; one particularly well-known, if not well-loved by users, example of this is Novell's ZEN (Zero Effort Networking) product.

11.2. Scripted Installations

Installation can be a tedious process, and deployment can be impractical when you have heterogeneous devices. Therefore, what would be really nice is the ability to perform an unattended or scripted installation. This can commonly be performed either by booting from the network, or providing some install-time parameter that points to a set of answers for the questions asked during installation. This file is commonly retrieved via HTTP, but may also be made available on a floppy etc. If booting from the network, this information may be configured based on the particular machine.

In the Linux space, two tools are commonly used: *Kickstart* for RPM-based systems, such as Fedora; while Debian-based systems use *pre-seed* files, which can answer questions not only at operating-system install-time, but also when individual packages are installed later. Kickstart also has a nifty little program to make the process of creating a kickstart file much easier, and a suitable Kickstart file is made when you install the OS manually, which provides a useful starting point.

Windows also supports scripted installations. Mac OS X notably does not; Apple prefers people to use imaging technologies for a base install. Given Apple's fairly uniform hardware

[5]You need switches that employ *IGMP Snooping*, otherwise the traffic turns into broadcast traffic, which floods the network and can bring a local network to its knees.

offering and simple installation process, there is less need for scripted installations, so long as the operating system you are installing is at least as modern as the most modern machine you wish to deploy onto. Further client administration is expected to be done using Apple’s OpenDirectory and Remote Desktop products.

11.3. The “Golden Client” Methodology

One way to deploy, and maintain deployed machines, is to use a golden-client methodology, as made popular by software such as SystemImager [http://wiki.systemimager.org/].

The basic concept is thus: you install a system (your *Golden Client*) and get it to the state you want to deploy to. You run some software which uploads all the files (excluding certain files which should be unique to the particular machine) to a server. You then run an initial install-client on the machines you want to deploy onto, which downloads the files from the server. Some scripting may be used to provide for tasks such as partitioning the hard disks. A concept of “classes” may be used to provide different files to different machines, so for example you might have a class for machines with ATI graphics cards and another class for NVidea, or you might have a class for particular machines which need some software with a limited number of licences.

When you want to make an update, you make the update to your Golden Client, run the agent on the Golden Client to synchronise the changes up to the server. You then cause all the other clients (perhaps by initiating a command over the network, or by some scheduled task) which runs a *differencing engine* to make the clients' files the same as the client files installed on the server.

This has the advantage of making updates relatively easily, especially when you have a lot of software being installed from different formats. However, there can be a fair bit of work involved in ensuring you don’t upload something to the server that should be local to the machine. The granularity is typically limited to a file which means modifications to isolated parts of a file (such as adjusting a configuration file) become rather more difficult. Configuration engines such as **cfengine**, **puppet**, and **chef** are much more capable in this regard.

11.4. Dealing with Heterogeity

Deployment can be much easier when the hardware platform you are deploying to is consistent. There is some room for flex, depending on the operating system in question. For example, Windows can be quite perplexed if it finds itself running with different hardware than the last time it booted, so Microsoft have a tool called SysPrep that removes some of the more unique aspects of the underlying registry, ready for deployment.

Linux systems tend to be rather more forgiving about suddenly finding itself on different hardware. So long as the system is not configured to particular hardware, and there is some hardware detection performed at bootup, then the process can be quite smooth. A lot of problems can be worked around by editing startup-scripts which check to see what devices are used in the system (**lspci**) and, for example, re-pointing a symbolic link to use the most suitable X.org configuration file, depending on what video card is installed. Thankfully, modern Linux systems have done a lot of work to make this less and less necessary.

Mac OS X, with its narrow hardware divergence, has no apparent problems with this either.

When deploying, you want to avoid as much local configuration as possible. For example, avoid unnecessary local user accounts and instead using network-based accounts and home directories. Linux systems have less ability to store their configuration in *directory servers*, which we we shall cover later in this paper. However work is being done to make Linux more Enterprise friendly, especially by Novell (which acquired SuSE Linux, another major player in the European Linux space). Microsoft has done a good job on making many information elements configurable via a directory server. Apple has also done a lot of good work, and is maturing nicely. Linux has a harder time because of its plain-text configuration file heritage of Unix.

11.5. Maintenance

Inevitably, you will need to maintain these machines, and this will include some degree of maintenance to a) possibly numerous disk images, b) installation scripts, or c) a golden client.

Disk images are easily the most amount of work to update, especially if you have to update multiple images, because you need to restore the image, update the system, recreate the image, and repeat for all images.

Installation scripts probably won't need a lot of work, often just adding or removing a directive to install a particular package, although this might be managed in some other way, such as **cfengine** [http://www.cfengine.com].

A Golden Client will need updated, synced to the server, and then synced to all the clients. This is typically fairly fast, and it is an effective way maintaining a lab of similar machines, especially where the files have been customised post-installation.