WEEK 1 – TASK 1.4P

Pass Task.
Release Date: 15 March, Due Date: 29 March, End Date: 5 April.

Learning Outcomes

- Basic understanding of Operating Systems and Linux OS
- Basic understanding of Command Line Interface (CLI)
- Understanding the basics of Virtual Machines (VMs)
- Setting up a VM and installing Kali Linux as host operating system

Instructions

Resources

An **answer sheet template** is available on OnTrack as a `**Resources**'. Please download the answer sheet and fill it with your answers. To upload on OnTrack, you need to convert the answer sheet template document to **PDF**. MS Word includes built-in PDF conversation capability.

All questions/tasks that have the icon below must be attempted for you to complete this task. If screenshots are required, please ensure that text in screenshots is readable. If you didn't know how to take a screenshot, please refer to Help Point on unit site (Discussions > Content > Help Point). Do not use your phone to take photos of your screen.

Remember that troubleshooting technical problems is part of learning in this field. Tasks are not step-by-step guide. You need to be in the driver seat and learn concepts by doing – as you would when you start your future job (many times even your supervisor doesn't know the answer to problems you face). Do your research patiently to solve issues you face and if you are stuck:

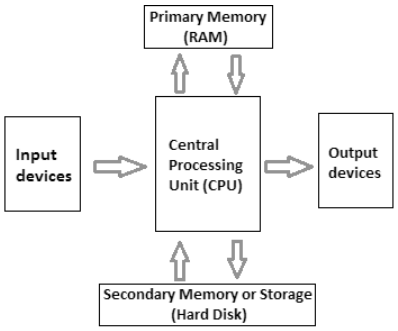
Help is always available in SIT182. Please go to **Discussions** and ask your questions about this task in **Task 1.4P**. All students are encouraged to participate and help peers with their questions. Helping others is a great way to learn and think about aspects you may have overlooked. You can also seek help from tutors during online and face-to-face pracs. Please do not raise your questions through Teams, OnTrack, or Email.

This task, being the first task, includes a lot of background information that will be helpful to those who need it. There are just 11 actual tasks to complete. All students are expected to know the concepts covered in this task hereon.

Section I: Background[1]

This unit is studied by many first-year student who in some cases may have little familiarity with computers. In this section we review some of the key concepts that will be assumed the students are aware of throughout the unit.

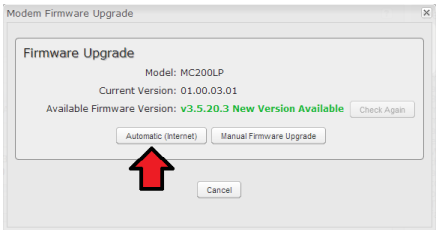
- A computer is a device that performs calculations. A typical modern computer is an electronic device that can perform a huge number of useful tasks for its owner.
- Any computer, small or large, has a central processing unit (CPU) that performs the calculations, or processing for the computer.
- A microcomputer is a computer small enough and cheap enough for the use of one person.
- The CPU in a microcomputer is a microprocessor, although many still refer to it simply as a CPU or processor.
- An important invention that led to the miniaturization of computers was the integrated circuit (IC), a small electronic component made up of transistors (tiny switches) and other miniaturized parts.
- Interaction with a computer is input/output (I/O). When we send something into the computer, we call it input. You are inputting through input devices when you type on the keyboard, tap on a touch screen, or talk to a computer through a microphone. Output is processed information of many types: sounds sent through the speakers, visual output to the display screen or printer and data files saved or sent over a network.
- In a microcomputer, the internal components include at least one microprocessor, random access memory (RAM) that acts as the main memory for holding active programs and associated data, firmware, and various other supporting circuitry, all installed onto a motherboard.
- Random-access memory (RAM) is volatile: when you turn off the computer the contents in RAM disappear.
- The typical microcomputer also has some form of storage, such as a hard drive, and it has at least one means each for input and output.



[1] *Part of this section is extracted from Survey of Operating Systems" by Jane Holcombe and Charles Holcombe [8th edition], Chapter 1- with amendments in text.*

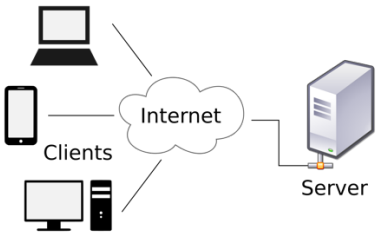
What is a firmware?
Each computer device you use has special software resident in integrated circuits called firmware containing small programs for providing basic communications between the operating system and the hardware.

- System firmware contains program code that informs the processor of the devices present and how to communicate with them.
- Most components and peripheral devices that connect to a computer (such as the video and network adapters, USB ports, and digital cameras) have their own firmware, which is often limited to small programs for providing basic communication between the operating system and the component.



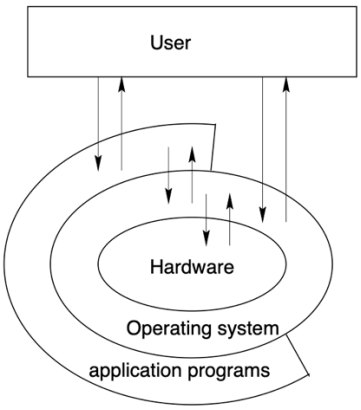
Internet of Things (IoT) devices: microcomputers exist in devices belonging to the Internet of things (IoT). These are devices we don't normally think of as computing devices. They include kitchen appliances, thermostats, utility meters, components in automobiles, light bulbs, and industrial control devices. They are not necessarily mobile, but they communicate on networks, often the Internet. IoT devices are increasingly used in industrial automation, connecting wirelessly, or via Ethernet, to automation networks.

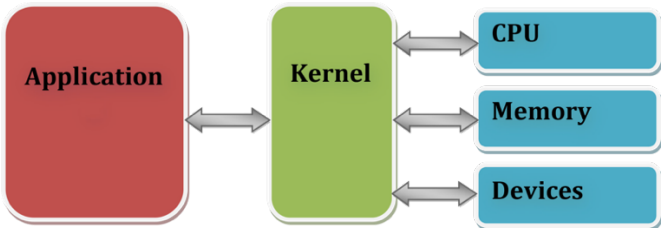
What is a Server?
A server is a computer that provides one or more services to other computers over a network. What services do servers provide? A file server stores data files for network-connected
users. If a server has one or more printers connected to it that it shares with users on the network, it is a print server. We call a server doing both tasks a file and print server; even though it sounds like two services, they combine into one service. A computer on the user end of these services is a client. Today's client computers include the PCs, laptops, tablets, and smartphones. A server can offer multiple services at the same time while also being a client to other servers.



What is an Operating System?

An operating system (OS) is a collection of programs that controls all of the interactions among the various system components, freeing application programmers from needing to include such functions in their programs. Every computer system must have at least one operating system to run other programs. An application (app) is software that allows a user to perform useful functions. Microsoft Word and Adobe Photoshop are applications. Applications send commands to the OS to interact with the hardware.





When a computer is turned on an operating system starts up (or "boots up," a derivation of the expression "lifting yourself by your own bootstraps"). Its main component, the kernel, remains in memory while the computer is running, managing low-level (close-to-the-hardware) OS tasks. When a programmer, also known as a "developer," writes an application, he or she designs the application to interact with the operating system and to make requests for hardware services through the operating system. To do this, a programmer must write the program to use the correct commands to request operating system services. The operating system, in turn, interacts with the hardware on behalf of the application and fulfills the requests the application made. An operating system performs several functions.

- Process management: helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

- **Memory management:** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.

- **File management**: It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

- **Device Management**: Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.

- **I/O System Management**: One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

- **Secondary-Storage Management**: Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.

- **Security**: Security module protects the data and information of a computer system against malware threat and authorized access. For example, Rachel is the accounting clerk in a small company. She has confidential information on her computer, and she doesn't want just anyone to be able to walk up to her computer and access the information stored there. Rachel can set up her computer so that others cannot access the data on her computer.

- **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

- **Networking**: A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.

- **Job accounting**: Keeping track of time & resource used by various job and users. Communication management: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems

OS's User Interface (UI)

The user interface (UI) is the software layer, sometimes called the shell, through which the user interacts with the OS. The UI includes the command processor, which loads programs into memory, as well as the many visual components of the operating system.

What is Unix?

UNIX is the oldest operating system still in use. UNIX grew out of an operating system developed for an early Digital Equipment Corporation (DEC) computer and went through several generations of changes before it emerged from the Bell Labs Computing Science

Research Center (Bell Labs) as UNIX version 6 in 1975, a portable operating system for minicomputers and mainframe computers.

A portable operating system is one that you can use on a variety of computer system platforms, with only minor alterations required to be compatible with the underlying architecture. Minicomputers and mainframe computers allowed multiple remote users to connect and use the computer's resources, and UNIX supported the time-sharing and multitasking features that made this possible.

UNIX exists in many different versions ("derivates"): Solaris, AIX, XENIX, HP-UX, SINIX, and Linux. If you are keen to review the history of operating systems, "Survey of Operating Systems" by Jane Holcombe and Charles Holcombe is a great reference.

What is Linux?

Linux is a computer operating system (similar to Microsoft Windows or Apple MacOS) originally developed by Linus Torvalds as a research project. Unlike other mainstream Operating Systems, Linux is made freely available and is Open Source. Linux is available in a variety of distributions, and it can be modified to run on nearly any computer. A distribution is a bundling of the Linux kernel and software—both enhancements to the OS and applications.

Different distributions of Linux exist because different people all have different opinions on what is most important. Each distribution represents the diverse choice of a group of individuals. Most modern-day distros spring from 'core' distributions, such as Redhat and Debian.

Ubuntu: Designed to be friendly to the average user, and the best desktop operating system that it can be. Ubuntu is based on Debian, and has a number of forks, including Kubuntu, Edubuntu, and Xubuntu. Today, Ubuntu (and its derivatives) is one of the more popular Linux Desktop operating system, and is currently in use on home computers, servers, business, education, government, and charity machines.

Today, Linux has found a home on smartphones (Android, Meego, Moblin, Maemo, Zaurus), Network Appliances (NAS, Firewall, Load Balancing), TVs (e.g., most Samsung TVs), Laptops, Netbooks, Desktops, Servers (including web servers), Games Consoles, IoT devices (Raspberry Pi devices can comfortably run Linux for example). Linux is usable in every area that Microsoft Windows is, and many more.

Different Linux Desktop Interfaces

There are two primary desktop interfaces in the Linux world that you are likely to encounter. GNOME and KDE. These are known as Window Managers

GNOME is the default Window Manager for Redhat and Ubuntu, and the one that you will be using on the Linux Terminal Server. The GNOME (pronounced "Gah-NOME") project's aim is to build a complete, user-friendly desktop based entirely on free software.

Like with the different Linux Distributions, whichever Desktop interface you choose is dependent onwhich you prefer, and which one fits your specific purpose.

Linux User Types

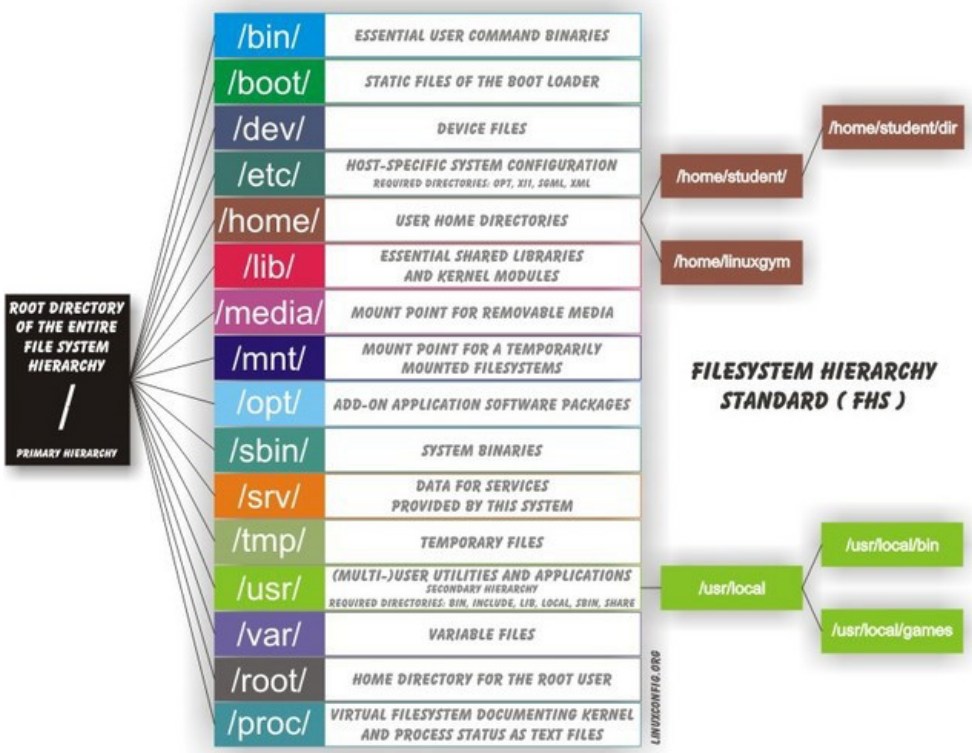
There are two user types:

- 'normal' users with restricted rights
- system administrator (root) with all privileges. The latter is responsible for the installation, configuration and maintenance of the system as well as the user administration.

Linux Filesystem

The Linux File System is different to that of Windows, in that the File System is not represented by Hard Drives and CD Drives like c:\ & d:\ etc. Instead, the whole system is represented by the root directory, or '/'. Files, folders, and drives, all exist within this. Within '/', exists a standard core directory structure:

(see next page)



The folder you operate in 95% of the time, is '~', which usually equates to '/home/<username>'.

Even devices exist as files within a Unix system. They are located in '/dev/', If you insert a CD, the drive will be mounted to '/media/cdrom'. Some systems do this automatically, however sometimes this needs to be done manually. An example command to run to do this would be: 'mount /dev/sdb1/media/cdrom'. This would allow you to access the CD-ROM by browsing '/media/cdrom/'. Before ejecting the disk, you would exit from the mounted directory, then run 'umount /media/cdrom', and then eject the disk from the computer. Some systems will respond to the command 'eject' for CD- ROMs, automatically unmounting and ejecting the drive in one command.

USB Sticks, Memory Cards, and Hard Disks are all mounted in the same way, having a corresponding '/dev/<filename>' file, and a location to mount it to. This could be in '/home/<username>/Storage', or in '/mnt'. It makes no difference to Linux, making it a very flexible and scalable OS.

Section II: CLI

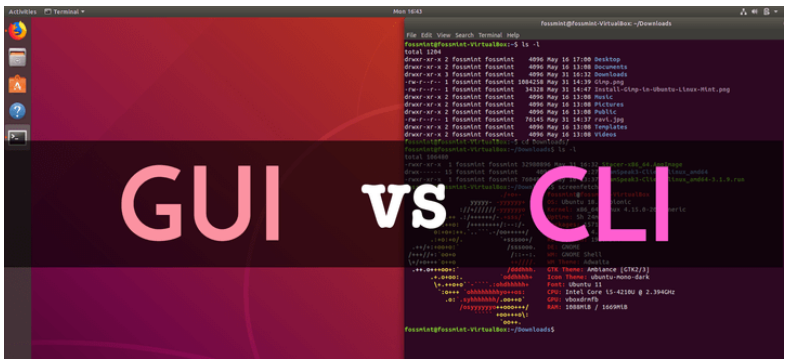
The Command Line Interface (CLI) is the most important interface to learn for Linux. Linux users will often be asked to perform commands through the use of the CLI, and not through a GUI.

To run commands covered in this section and complete the tasks, first setup Kali VM as covered in Section III.

Why we prefer the Command Line Interface over Graphical User Interface (GUI)?

Why would we use a command prompt over the Graphic User Interface that we all know and love? The reason is simple: Control. A command prompt gives us greater control and flexibility over the system or software we are creating. We have no way of knowing if configuring a service using a graphical user interface is doing exactly what we want it to, or if it's doing something that its interpreted as what we want. In the world of security, the risk is not worth taking.

The command line can be used if you wanted to perform a task multiple time. You could script the command, and then run the script, making sure that the command run is 100% accurate. If you wanted to run a command once a year, you would probably use the CLI (Command Line Interface) over the GUI (Graphic User Interface). GUI's will change, and documentation will need updating regularly. The CLI is a more static environment, requiring little to no documentation changes.



The command line interface is case-sensitive. It does understand the difference between a lower-case t and an upper-case T – Check what you're typing as one wrong character can cause a lot of problems.

There are many CLIs that can be used in any Unix-like system, two of the most popular shells are the Bourne Again Shell (BASH) and Z-Shell (zsh), which is actually based of bash originally.

Shell Scripts

Shell scripts are small programs consisting of UNIX commands and shell-specific program constructs (branches, loops etc), which behave like UNIX com- mands but are present in text form (instead of binary). These scripts are interpreted by the shell. The syntax of shell scripts differs (considerably) from shell to shell.

Let's now get our hands dirty (a bit).

A. Navigating the File System

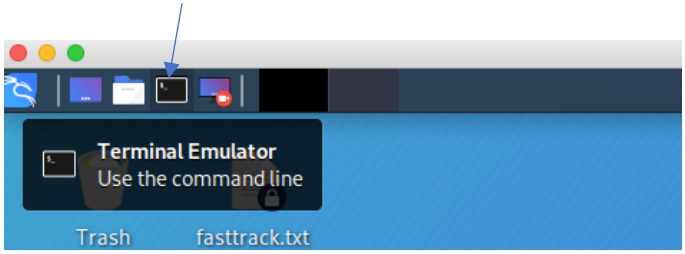
First, we need to learn how to move around the Linux file system. Once you have logged in, the directory you start in is your home directory. Each user is given theirown local home directory which is independent from everyone else's. First you need to take note ofyour home directory, as this is different for every user.

```
pwd
```

'pwd' or 'print working directory' is a useful command to check the full path of where you are currently located within the Linux File System.

This can save you from getting lost, or as a confirmation that you are working in the correct directory.

Task 1: run the above command in the "Terminal Emulator" of Kali Linux. What is your home directory? Include a screenshot of the output you get after running the command.



Now we will move to the root directory / and then run a command to list all directories inside of it.

Task 2: run the following commands and include a screenshot of the outputs you get.

`cd ~`	Changes Directory to your home directory (donated by ~)
`pwd`	Print the current working directory -
`cd /`	The command 'cd' stands for 'change directory'. This is used to move to different directories, and can be used with absolute or relative paths. / tells the system to navigate to the root directory of the filing system, irrespective of your current working directory.
`pwd`	Print the current Working Directory

You can also move deeper into the file system using relative paths.

First, run the following commands:

`cd ~`	First, return to your home directory
`pwd`	Print the current working directory to check out location.
`ls`	This will print (display on screen) a list of files and folders stored in the current working directory. In this instance, we're printing the contents of your Home Directory.

You will see a list of folders and files appear. This is the current contents of your home directory. We are going to move into the directory 'Desktop' using a relative path.

Task 3: run the following commands and include a screenshot of the outputs you get.

`cd Desktop`	This changes the current working directory to 'Desktop', which is located in your current working directory IMPORTANT: All Commands are Case Sensitive!
`pwd`	Print the current working directory to check out location.

Task 4: Run the following commands replacing <your home directory path> withthe path to your home directory (the one you found in Task 1). Include a screenshot of commands and the output you get.

(see next page)

cd ~	Changes Directory to your home directory (donated by ~)
pwd	Print the current working directory - You should be in /home/msai135 (where msai135 is your own username)
cd <your home directory path>	Changes directory using an absolute path. the '/' denotes the root of the filing system.
pwd	Print the current Working Directory to confirm your current location
cd ~	Return to your home directory
pwd	Print the current Working Directory to confirm your current location
cd ~/Desktop	Change Directory to your Desktop folder, which is absolute to the root(/) of the system.
pwd	Print the current working directory to confirm your current location
cd ../	The ../ tells the system to navigate up exactly one level, relative to your current working directory.
pwd	Print the current working directory to confirm your current location

As you can see, you can chain different folders together to move between many levels in a single command.

The command you ran was 'cd /' The '../' part of this is the path that you are moving to. This is what is known as a relative path, as you moved up one level (to /home) relative to your starting position.

B. Working with Files

First, create a directory (i.e., folder), move to it, and verify where you are:

cd ~	Change directory to your home directory
mkdir unixintro	Make directory 'unixintro'
cd unixintro	Change directory to 'unixintro'

Task 5: Now, first run "cd /" in the Terminal and then try to run the `mkdir unixintro". When trying to execute the `mkdir' command you will receive a permission error. Find a way to solve this problem and create the unixintro folder in / directory. In your answer sheet document, mention what command you used.

Hint: there are 2 different types of users. You will need to be more privileged to create folder in / directory.

Task 6: let's copy a file to `unixintro' directory you created in / directory. You want to copy "/usr/share/wordlists/fasttrack.txt". You will need to use the `cp' command for this. Include the full command.

Hint: https://shapeshed.com/unix-cp/ provides some helpful tips.

Task 7: Let's try to copy multiple files into "unixintro" directory you have created in / directory. This time you would like to use `cp' command and copy all files available in "/usr/share/wordlists/dirb/". Include the full command you need to use and a screenshot of the output you get when running the command.

Hint: https://shapeshed.com/unix-cp/ provides some helpful tips.

You may get errors like below for some of the files:

```
cp: -r not specified; omitting directory '/usr/share/wordlists/dirb/others'
cp: -r not specified; omitting directory '/usr/share/wordlists/dirb/stress'
cp: -r not specified; omitting directory '/usr/share/wordlists/dirb/vulns'
```

Task 8: Refer to the hint page and find out how to adjust the "cp" command to include these sub-directories when copying the content of "/usr/share/wordlists/dirb/". What is the updated `cp' command?

If you have managed to complete Task 8 successfully your `unixintro' should contain the following files and folders:

```
big.txt      common.txt    extensions_common.txt  mutations_common.txt  small.txt    stress
catala.txt   euskera.txt   indexes.txt            others                spanish.txt  vulns
```

Task 9:

A. What command should be used to move the `common.txt' file to `others' directory in `unixintro' directory? (Include the full command to move common.txt to `others' and run the command to move the file)
B. Change your directory to `others'. List files available in `others'. Remove/Delete the `common.txt' file you moved. Include the full commands you used for Task 9.B.

C. Improving your productivity in the CLI

Working within the confines of the Command Line Interface can sometimes be frustrating. The lack of a 'copy / paste' menu on the right click of the mouse can be a major headache when it comes to repeating commands.
[you could use mouse for copy/paste in Terminal Emulator, but let's assume we cannot so we learn CLI commands better.]

- Tab-to-complete

If you begin typing a command or path in the CLI, you can 'auto-complete' this by pressing the 'tab' key. For example, if you want to navigate to '/etc/samba/', you can just type '/et' and then hit tab, which will auto-complete it to '/etc/'. You can then append 'sam' and hit tab, and it will auto- complete to '/etc/samba/'. This is incredibly useful when accomplishing tasks quickly.
If the command does not complete after a single tab, press it again and the system will display a list of options that matches what you have entered so far.

- Editing a command

You can use the Left (back) and Right (forward) arrows on the keyboard to move backwards and forwards through a typed command. Very useful if you want to modify something right at the beginning of the long line you just typed.

- History

At any point, you can recall previous commands that you typed by using the Up and Down arrows on the keyboard. Up goes back through the commands you've used, in reverse order going back into the past. Down brings you back to the present until you reach a blank line to input a new command.

Your history is stored in '~/.bash_history' .
You can view it by running the command cat ~/.bash_history

Your history does not store the output from the terminal, nor your interaction with any programs or applications that have been run - only the commands that you inputted straight into the command line are stored.

- **Clear**

Sometimes you just want the shell to be cleared and not see previous commands and outputs. You can use the command `clear' for this.

If you are keen for more CLI exercises/tutorials, you may find http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html quite helpful.

Task 10: Search online about Linux Kernel. In your own words (maximum 100 words) discuss what is a Linux Kernel, what does it do, and where does it fit within an OS.

Section III: Setting up Kali Linux Virtual Machine (VM)

Getting to know about Kali and downloading Kali VM

Head to https://www.kali.org and explore Kali Linux's website. Open up Kali's Documentation and browse https://www.kali.org/docs/introduction/. Read through `What is Kali Linux' and `Should I use Kali Linux'. You are assumed to know about Kali Linux hereon.

Note: link on PDF may not work correctly when clicking on them. Type the URL in your browser's address bar if link was not working.

Download the following customised Kali VM for VirtualBox:
https://cloudstor.aarnet.edu.au/plus/s/GBcX4b5HAiMB376 *[about 3.5GB]*

If the above link doesn't work, try the following second link:
https://www.dropbox.com/s/ftfv419zxqopo8v/Kali-Linux-2020%20Docker%20Enabled.ova

Note: All students are encouraged to setup Kali VM on their own laptop. However, if you are in cyber labs of Deakin University (Burwood and Geelong), a Kali VM is already available for you to use. Your tutor will show you how to run Kali VM.

- Remember that lab machines are not persistent, and all content is wiped as soon as you logout.
- To login to cyber lab machines, click on Guest account (no Deakin username and password is needed).
- Cyber lab machines are isolated from Deakin services. Hence, you cannot access any Deakin domain (including OnTrack, Unit Site, and Deakin Email). You can access the web (e.g., Google) – traffic is monitored for illicit content and malicious activities.

Installing VirtualBox

The first step in using the virtual machines is to download and install VirtualBox by Oracle from the following site:

https://www.virtualbox.org/

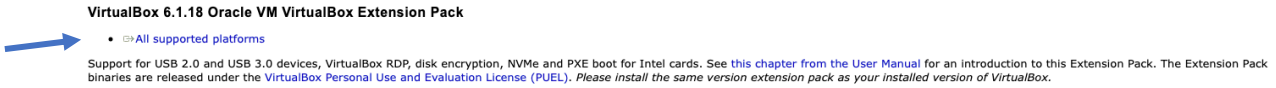
This software is free and available for the Windows, macOS, and Linux platforms.

- **Installing VM VirtualBox Extension Pack**

You will need the VirtualBox Extension Pack to enhance the experience of using a VM on your host OS.

Visit the following URL: https://www.virtualbox.org/wiki/Downloads

Then, download the extension pack.



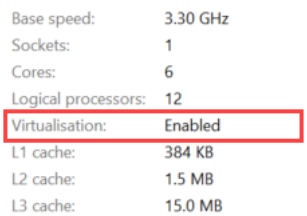
Click on the downloaded file and run it. VirtualBox should show you an installation prompt. Accept and Install.

You can now return to Section II and complete the tasks using Terminal Emulator in Kali Linux.

- BIOS Settings to run VMs

PCs: Virtualization is often disabled by default for many PCs. You can determine this using one of two methods.

The first method is by using Task Manager. Open Task Manager, either by right-clicking your taskbar and selecting Task Manager, or by pressing Ctrl+Alt+Delete and clicking the button there. In Task Manager, click on the Performance tab, and then selecting the CPU. The bottom half of the window will include summary information indicating whether virtualization is enabled or not:



Alternatively, start a command prompt and run the command 'systeminfo'. The output of this command will include a section indicating whether virtualization is enabled or not (usually at the end of the output):

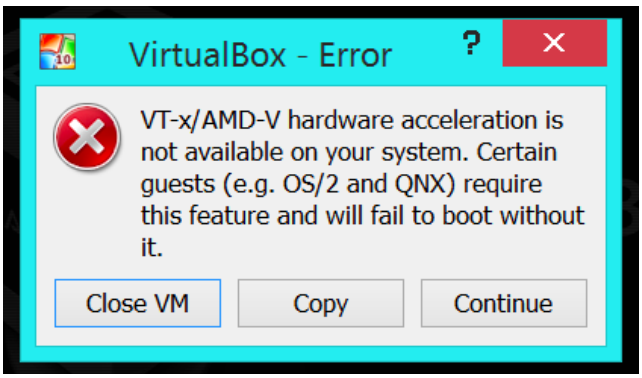
```
Hyper-V Requirements:      VM Monitor Mode Extensions: Yes
                           Virtualization Enabled In Firmware: Yes
                           Second Level Address Translation: Yes
                           Data Execution Prevention Available: Yes
```

If these are showing that virtualization is not enabled, you will need to enable these settings from your computer's BIOS. Usually, these settings are accessed by pressing a key when the computer is first turned on (delete, escape, F1, F4, etc.) whilst the initial splash screen is shown (usually the logo of the computer/motherboard manufacturer). If this isn't obvious, you'll need to check the documentation for your computer for the exact location if it doesn't indicate how to access the settings (Google can help here also).

Once you have accessed your computer's settings, you're looking for the ability to enable Intel VT-x and VT-d support, or for AMD CPUs the equivalent is usually indicated as Secure Virtual Machine (SVM). These settings are often associated with the CPU Settings or the Advanced Settings for the computer.

Apple Mac: Note that if you are using an Apple Mac computer, the settings should already be correct for that device. If virtualization is disabled, your Mac probably isn't powerful enough. You can find instructions on Apple's web site if you wish to try however (not recommended).

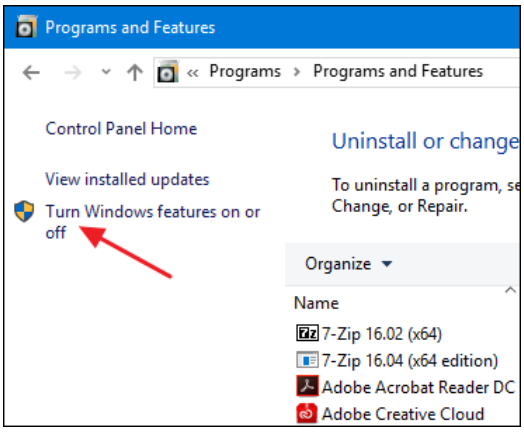
If you do not have virtualisation enabled, you will receive an error like below:



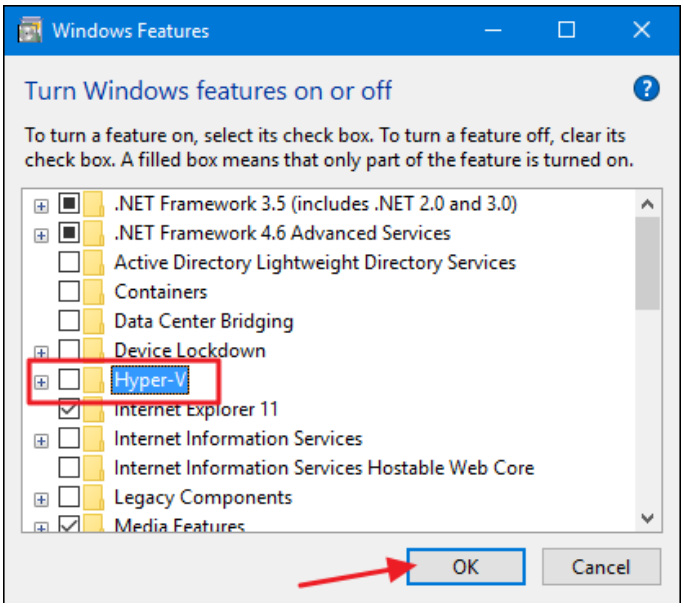
Other Troubleshooting tips:

If you have ensured about virtualisation capability (as suggested above), then another suggestion is to uninstall Hyper-V.

To solve this problem, you just need to uninstall Hyper-V. Hyper-V is an optional Windows feature, so uninstalling it is a little different than uninstalling a regular app. Head to Control Panel > Uninstall a Program. In the "Programs and Features" window, click "Turn Windows features on or off."



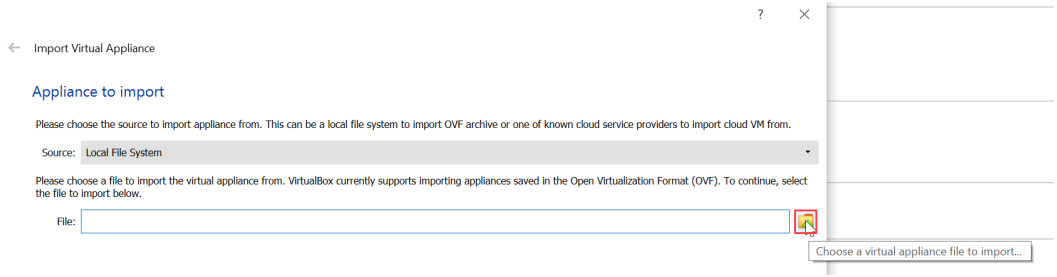
In the "Windows Features" window, clear the "Hyper-V" checkbox and then click "OK."



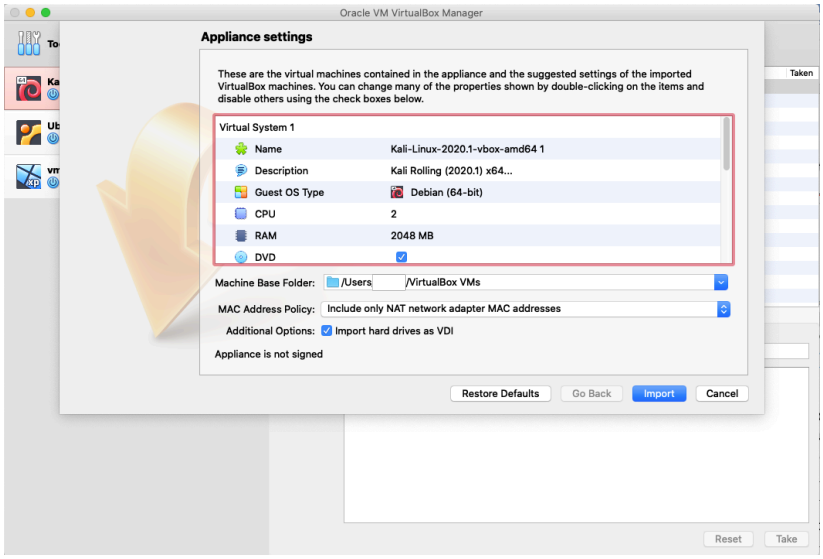
When Windows is done uninstalling Hyper-V, you'll need to restart your PC and then you can try using VirtualBox or VMware again.

Importing Kali Linux VM

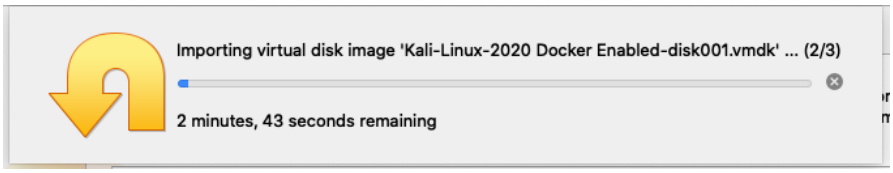
The next step is to import the VMs that you have downloaded. To do so, start VirtualBox, open the File menu, and select the Import Applicance… option. Click on the browse button (highlighted below), select the .ova file for Kali VM, and click Next.



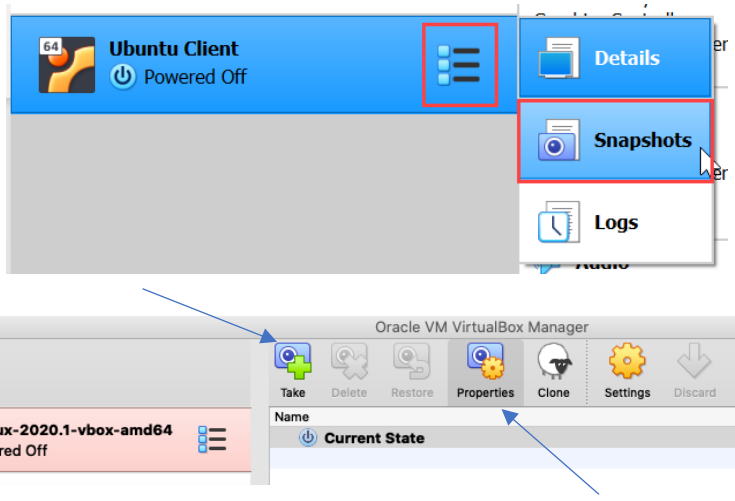
Confirm the settings that are presented to you in the next page. Generally, these settings shouldn't be modified unless you know what you're doing:



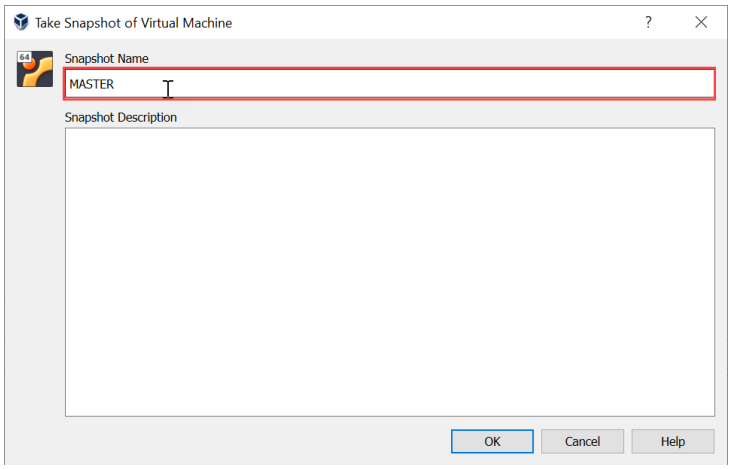
You will now see a progress bar indicating the VM is being imported. Wait for this to complete:



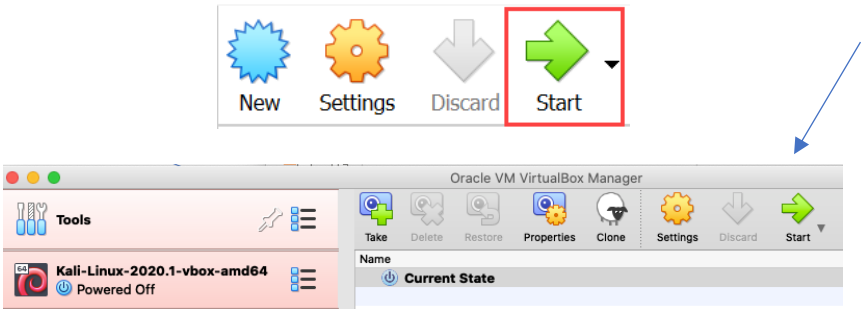
Finally, before using the VM, create a snapshot so that you can always rewind the VM to its initial state as follows. Click on the menu button next to the VM indicated and click on the Snapshots entry:



Make sure that the Current State is highlighted, then click on the Take button. Give the snapshot an appropriate name (MASTER in this case) and click the OK button.



To run the VMs, click on Start after choosing Kali.



Login Details for Kali

On the login screens, please use "kali" (without quotes) as both username and password.

What is a Virtual Machine (VM)?[2]

When early computer systems were being developed, hardware was designed first, and software written specifically for that hardware followed. Each system was essentially handcrafted with its own instruction set, and software was developed for that specific instruction set. This included operating system software, assemblers (later compilers), and application programs. With a small
user community, relatively simple programs, and fixed hardware, this tightly integrated approach worked quite well, especially while the basic concepts of the stored program computer were still evolving. But user communities grew, operating systems became more complex, and the number of application programs rap- idly expanded, so that re-writing and distributing software for each new computer system became a major burden.

[2] Extracted from "An Overview of Virtual Machine Architectures" by J. E. Smith and Ravi Nair, published by Elsevier Science.

The advantages of software compatibility and portability quickly became evident. Furthermore, hardware designers for the various computer companies gravitated toward certain common features. New designs were often similar to previous ones, although usually not identical. It was not until the development of the IBM 360 family in the early 1960s, that the importance of full software compatibility was fully recognized and articulated. The IBM 360 series had a number of models that could incorporate a wide range of hard- ware resources, thereby covering a broad spectrum of price and performance levels -- yet they were designed to run the same software. To successfully accomplish this, the interface between the hardware and software had to be precisely defined and controlled; the concept of the Instruction Set Architecture (ISA) came into being.

In addition, operating systems were developed that could shield application programs from specifics of the hardware, for example, the amount of memory available and the exact characteristics of disk drives. Operating systems were responsible for managing hardware resources, protecting running applications and user data, supporting controlled sharing, and providing other useful functions. Because application programs depend heavily on the services provided by the operating system, these operating system services and the operating system interfaces also became an important point for standardization. Today, there is a relatively small number of standard operating systems. Furthermore, operating systems no longer work in relative isolation; today they are responsible for controlled protection and sharing of data and resources, extending across net- works of computers.

The development of standard interfaces for matching software to underlying hardware has led to a very powerful environment for developing large, complex, and varied software systems. Because of standardized interfaces, software can be made much more flexible and portable than would otherwise be possible. The situation is not perfect, however, because there are a number of different, incompatible ISAs and operating systems. Hence, software portability is limited to those platforms that conform to the same standards as the ones for which the software was developed. User programs are tied to a specific instruction set and operating system. An operating system is tied to a computer that implements a specific instruction set, memory system, and I/O system interfaces. This lack of portability becomes especially restrictive in a world of net- worked computers where one would like software to move as freely as data.

These limitations on software are only there because the platform, or “machine”, on which the software exe- cutes is assumed to be a physical entity with certain properties: e.g., it runs a specific ISA and/or operating system. In brief, the software runs on a single type of real machine. Virtual Machines (VMs) eliminate this real machine constraint and enable a much higher degree of portability and flexibility. A virtual machine is implemented by adding software to an execution platform to give it the appearance of a different platform, or for that matter, to give the appearance of multiple platforms. A virtual machine may have an operating system, instruction set, or both, that differ from those implemented on the underlying real hardware.

The virtualization of architected elements enables a larger scale virtualization of computer system resources. And in some VM applications, this is the whole point of virtualization. That is, processors, memory, and I/O devices, including network resources can be virtualized. When the state of a process, or system, is virtualized it is essentially separated from a specific piece of physical hardware. This means that a virtual process or system can be temporarily associated with specific physical resources, and that the association can change over time. For example, a server with m processors can support n system virtual machines where n ≥ m, and the number of physical processors associated with a given virtual machine can change over time as the workloads of the virtual systems change. Similarly, a single physical network interface can serve a number of virtual networks by time-multiplexing the physical resource amongst a number of virtualized network inter- faces.

One type of virtual machine (VM) is illustrated in Figure 1 where virtualizing software is placed between the underlying machine and conventional software. In this example, virtualizing software translates the hardware ISA so that conventional software "sees" a different ISA from the one supported by hardware. As we shall see, virtualizing at the ISA level is only one possibility, but it is adequate for illustrating the range of VM applications. The virtualization process involves 1) the mapping of virtual resources, e.g., registers and memory, to real hardware resources and 2) using real machine instructions to carry out the actions specified by the virtual machine instructions, e.g., emulating the virtual machine ISA.

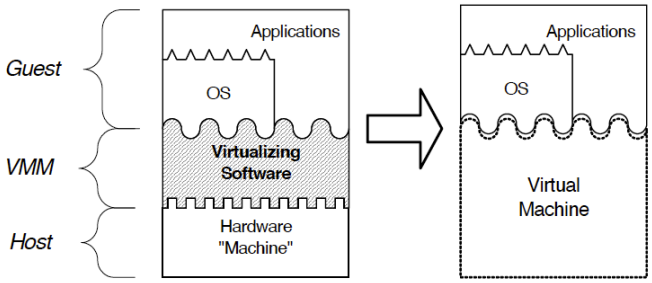


Figure 1

Figure 1 Virtualizing software can translate the ISA used by one hardware platform to another, forming a Virtual Machine, capable of executing software developed for a different set of hardware.

With regard to terminology (see Figure 1), we usually refer to the underlying platform as the "host", and the software that runs in the VM environment as the "guest". The virtualizing software will often be referred to as the "virtual machine monitor" (VMM), in accordance with the term used in the original VMs developed in the late 1960's.

We will need to stop here given the scope of this unit. However, if you are passionate to know more about VMs, then you may wish to read [3].

What are virtual machines used for?[3]

Virtual machine technology is used for many use cases across on-premises and cloud environments. More recently, public cloud services are using virtual machines to provide virtual application resources to multiple users at once, for even more cost efficient and flexible compute.

Virtual machines (VMs) allow a business to run an operating system that behaves like a completely separate computer in an app window on a desktop. VMs may be deployed to accommodate different levels of processing power needs, to run software that requires a different operating system, or to test applications in a safe, sandboxed environment.

Virtual machines have historically been used for server virtualization, which enables IT teams to consolidate their computing resources and improve efficiency. Additionally, virtual machines can perform specific tasks considered too risky to carry out in a host environment, such as accessing virus-infected data or testing operating systems. Since the virtual machine is separated from the rest of the system, the software inside the virtual machine cannot tamper with the host computer.

Advantages of virtual machines

- Virtual machines are easy to manage and maintain, and they offer several advantages over physical machines:
- VMs can run multiple operating system environments on a single physical computer, saving physical space, time and management costs.
- Virtual machines support legacy applications, reducing the cost of migrating to a new operating system. For example, a Linux virtual machine running a distribution of Linux as the guest operating system can exist on a host server that is running a non-Linux operating system, such as Windows.
- VMs can also provide integrated disaster recovery and application provisioning options.

Disadvantages of virtual machines

- While virtual machines have several advantages over physical machines, there are also some potential disadvantages:
- Running multiple virtual machines on one physical machine can result in unstable performance if infrastructure requirements are not met.
- Virtual machines are less efficient and run slower than a full physical computer. Most enterprises use a combination of physical and virtual infrastructure to balance the corresponding advantages and disadvantages.

[3] *Extracted from https://www.vmware.com/topics/glossary/content/virtual-machine*

The two types of virtual machines

Users can choose from two different types of virtual machines—process VMs and system VMs:

- A process virtual machine allows a single process to run as an application on a host machine, providing a platform-independent programming environment by masking the information of the underlying hardware or operating system. An example of a process VM is the Java Virtual Machine, which enables any operating system to run Java applications as if they were native to that system.
- A system virtual machine is fully virtualized to substitute for a physical machine. A system platform supports the sharing of a host computer's physical resources between multiple virtual machines, each running its own copy of the operating system. This virtualization process relies on a hypervisor, which can run on bare hardware, such as VMware ESXi, or on top of an operating system.

What are 5 types of virtualization?

All the components of a traditional data center or IT infrastructure can be virtualized today, with various specific types of virtualization:

- Hardware virtualization: When virtualizing hardware, virtual versions of computers and operating systems (VMs) are created and consolidated into a single, primary, physical server. A hypervisor communicates directly with a physical server's disk space and CPU to manage the VMs. Hardware virtualization, which is also known as server virtualization, allows hardware resources to be utilized more efficiently and for one machine to simultaneously run different operating systems.

- Software virtualization: Software virtualization creates a computer system complete with hardware that allows one or more guest operating systems to run on a physical host machine. For example, Android OS can run on a host machine that is natively using a Microsoft Windows OS, utilizing the same hardware as the host machine does. Additionally, applications can be virtualized and delivered from a server to an end user's device, such as a laptop or smartphone. This allows employees to access centrally hosted applications when working remotely.

- Storage virtualization: Storage can be virtualized by consolidating multiple physical storage devices to appear as a single storage device. Benefits include increased performance and speed, load balancing and reduced costs. Storage virtualization also helps with disaster recovery planning, as virtual storage data can be duplicated and quickly transferred to another location, reducing downtime.

- Network virtualization: Multiple sub-networks can be created on the same physical network by combining equipment into a single, software-based virtual network resource. Network virtualization also divides available bandwidth into multiple,

independent channels, each of which can be assigned to servers and devices in real time. Advantages include increased reliability, network speed, security and better monitoring of data usage. Network virtualization can be a good choice for companies with a high volume of users who need access at all times.

- Desktop virtualization: This common type of virtualization separates the desktop environment from the physical device and stores a desktop on a remote server, allowing users to access their desktops from anywhere on any device. In addition to easy accessibility, benefits of virtual desktops include better data security, cost savings on software licenses and updates, and ease of management.

Container vs virtual machine

Like virtual machines, container technology such as Kubernetes is similar in the sense of running isolated applications on a single platform. While virtual machines virtualize the hardware layer to create a "computer," containers package up just a single app along with its dependencies. Virtual machines are often managed by a hypervisor, whereas container systems provide shared operating system services from the underlying host and isolate the applications using virtual-memory hardware.

A key benefit of containers is that they have less overhead compared to virtual machines. Containers include only the binaries, libraries and other required dependencies, and the application. Containers that are on the same host share the same operating system kernel, making containers much smaller than virtual machines. As a result, containers boot faster, maximize server resources, and make delivering applications easier. Containers have become popular for use cases such as web applications, DevOps testing, microservices and maximizing the number of apps that can be deployed per server.

Virtual machines are larger and slower to boot than containers. They are logically isolated from one another, with their own operating system kernel, and offer the benefits of a completely separate operating system. Virtual machines are best for running multiple applications together, monolithic applications, isolation between apps, and for legacy apps running on older operating systems. Containers and virtual machines may also be used together.

Task 11: Reflection time. What did you learn that was new to you in Week 1 of OnTrack tasks (it could be what is covered in this task or any other information you learnt as you worked on the task)? Consider writing a paragraph summary of your reflection (3-4 lines suffices) or maybe compile a cheat sheet of CLI commands for future weeks.

References:

[1]: Survey of Operating Systems" by Jane Holcombe and Charles Holcombe [8th edition], Chapter 1.
[2]: "An Introduction to Linux", Royal Holloway University.
[3]: "An Overview of Virtual Machine Architectures" by J. E. Smith and Ravi Nair, published by Elsevier Science.
[4]: Virtual Machine, https://www.vmware.com/topics/glossary/content/virtual-machine