

BSc in Computer Engineering
CMP4103
Computer Systems and Network Security

Lecture 5
Virtualisation

Eng Diarmuid O'Briain, CEng, CISSP



Department of Electrical and Computer Engineering,
College of Engineering, Design, Art and Technology,
Makerere University

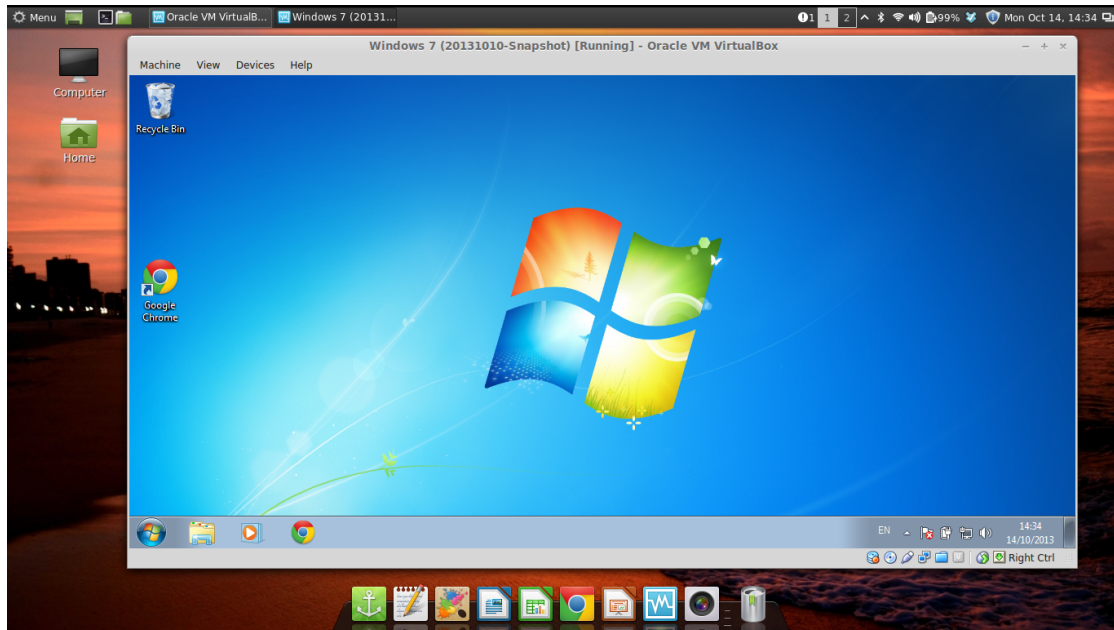
Copyright © 2017 Diarmuid Ó Briain

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table of Contents

1. VIRTUALISATION.....	4
2. HYPERVISOR.....	5
3. XEN.....	7
3.1 XEN HYPERVISOR.....	7
3.2 XEN ARCHITECTURE.....	8
3.3 GUEST TYPES.....	9
3.4 TOOLSTACKS, MANAGMENT APIs AND CONSOLES.....	10
3.5 THE MODELS.....	11
3.6 CONTROL DOMAIN (DOM0).....	11
4. KERNEL VIRTUAL MANAGER (KVM).....	12
5. CONTAINERS WITH DOCKER.....	13
5.1 THE CONTAINER CONCEPT.....	13
5.2 WHAT IS DOCKER.....	14
5.3 DOCKER HUB.....	14
6. VIRTUALBOX.....	15
6.1 GETTING VIRTUALBOX.....	15
6.2 BUILDING A VM ON VIRTUALBOX.....	16
6.3 INSTALLING AN OS ON A VM.....	21
6.4 RUNNING THE VM.....	23
7. LAB EXERCISE.....	24
8. BIBLIOGRAPHY.....	27

1. Virtualisation



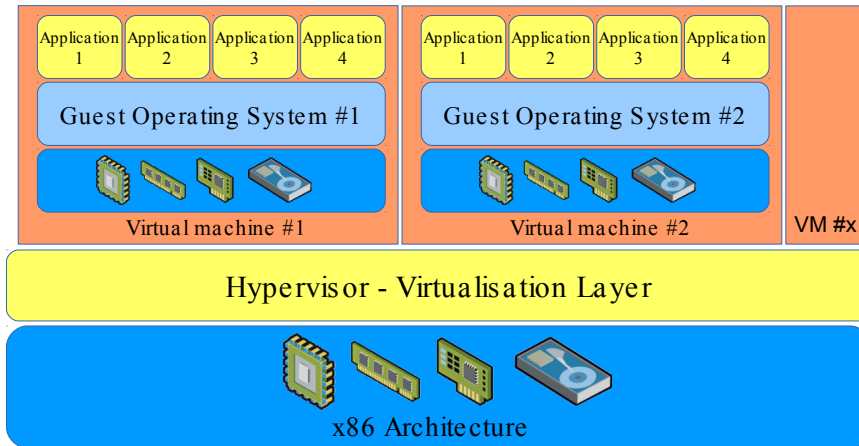
Virtualisation is the creation of a virtual computer called a Virtual Machine (VM) that acts like a real computer with an Operating System (OS). Software executed on these VMs is separated from the underlying hardware resources. The diagram above shows a computer running Linux which is hosting a VM that looks like a separate computer with Microsoft Windows 7 OS. In this case virtualisation is provided by Oracle VirtualBox.



2. Hypervisor

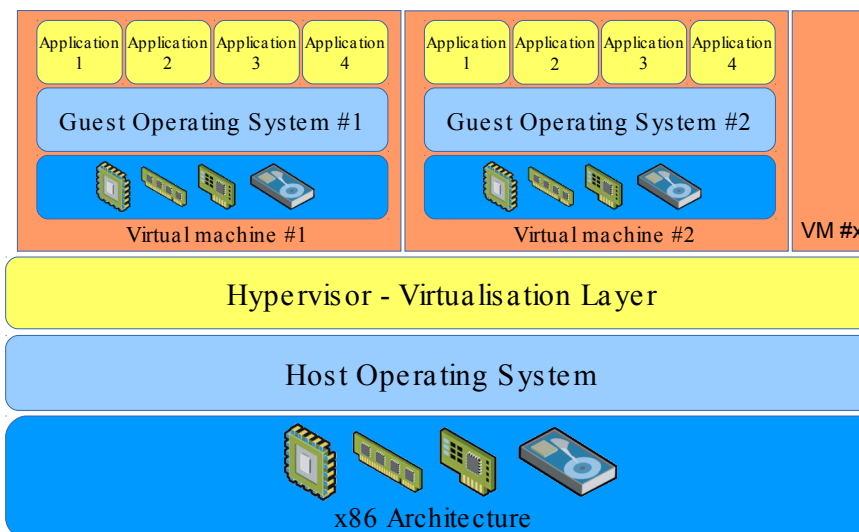
A hypervisor or VM Manager, is software that permits multiple OSs to share a single hardware host computer. Each OS appears to have the host's processor, memory, and other resources all to itself. However, the hypervisor is actually controlling the host processor and resources, allocating what is needed to each OS in turn and making sure that the guest OSs (VMs) cannot disrupt each other.

2.1.1 Type-1 Hypervisor



A Type-1 hypervisor (baremetal hypervisor) is a client hypervisor that interacts directly with hardware that is being virtualised. It is completely independent of the host OS, unlike a Type-2 hypervisor, and boots before the host OS. Currently, Type-1 hypervisors are being used by all the major players in the desktop virtualisation space, including but not limited to VMware vSphere (ESX) Hypervisor, Xen Hypervisor and Microsoft Hyper-V Server.

2.1.2 Type-2 Hypervisor



A Type-2 hypervisor is a client hypervisor that sits on top of a host OS. Unlike a Type-1 hypervisor, a Type-2 hypervisor relies heavily on the host OS. It cannot boot until the host OS is already up and running and, if for any reason the host OS crashes, all end-users are affected. This is a big drawback of Type-2 hypervisors, as they are only as secure as the host OS on which they rely. Also, since Type-2 hypervisors depend on an OS, they are not in full control of the end user's machine. One advantage of a Type-2 Hypervisor is the hypervisor functionally is practically identical on all of the host platforms, typically common file and image formats. All this results in the ability to run VMs created on one host on another host with a different host OS. i.e. you may have created a Microsoft Windows VM on Linux move it to another computer with a different host OS, say BSD UNIX and the Hypervisor will be able to run the guest OS in a VM. Some examples include Oracle VirtualBox, VMware Player and Microsoft Virtual Server.

2.1.3 Type-1 -vs- Type-2 Hypervisor

For high performance, Return on Investment (ROI) and scalability a Type-1 hypervisor is far superior to a Type-2. This is due to the independence of the Type-1 hypervisor from the OS.

3. Xen

The Xen Project team is a global open source community that develops the Xen Hypervisor, contributes to the Linux ParaVirtual OPerationS (PVOPS) framework, the Xen Cloud Platform and Xen Advanced RISC Machine (ARM).

Xen (zɛn) has its origins in the ancient greek term Xenos, which can be used to refer to guest-friends whose relationship is constructed under the ritual of xenia 'guest-friendship', which in term is a wordplay on the idea of guest OSs as well as a community of developers and users. The original website was created in 2003 to allow a global community of developers to contribute and improve the hypervisor.

3.1 Xen Hypervisor

Xen hypervisor is a type-1 or baremetal hypervisor, which makes it possible to run many instances of an OS or indeed different OSs in parallel on a single host. The hypervisor is used as the basis for a number of different commercial and open source applications, such as: server virtualisation, Infrastructure as a Service (IaaS), desktop virtualisation, security applications, embedded and hardware appliances. It enables users to increase server utilisation, consolidate server farms, reduce complexity, and decrease total cost of ownership. Xen is the basis of some of the worlds largest Clouds, Amazon Web Services (AWS) and Citrix XenServer to name a few.

3.1.1 Xen features

Small footprint and interface (is around 1MB in size). Because Xen uses a microkernel design, with a small memory footprint and limited interface to the guest, it is more robust and secure than other hypervisors.

OS agnostic

Most installations run with Linux as the main control stack called domain0 (dom0). But a number of other OSs can be used instead, including NetBSD and OpenSolaris.

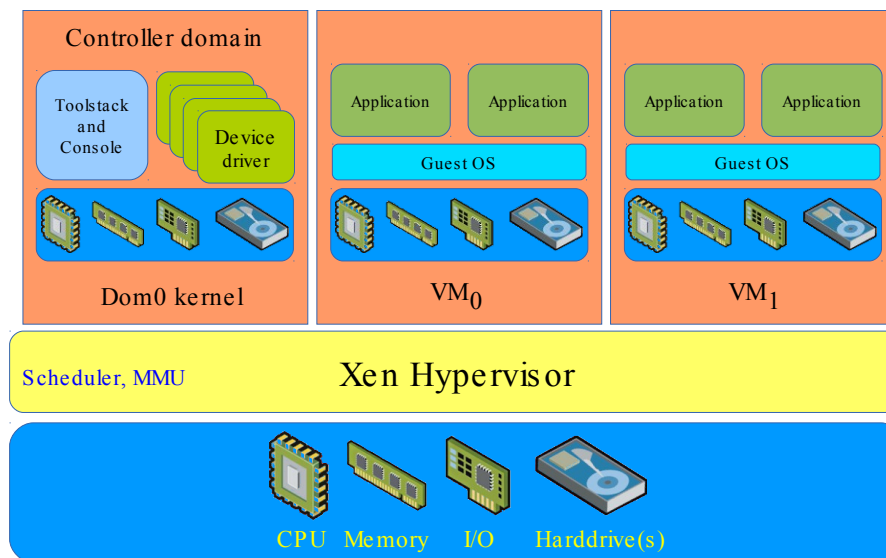
Driver Isolation

Xen has the capability to allow the main device driver for a system to run inside a VM. If the driver crashes, or is compromised, the VM containing the driver can be rebooted and the driver restarted without affecting the rest of the system.

Para-Virtualisation

Fully PV guests have been optimised to run as a VM. This allows the guests to run much faster than with Hardware Extensions (HVM). Additionally, Xen can run on hardware that doesn't support virtualisation extensions.

3.2 Xen Architecture



3.2.1 Xen Hypervisor

The Xen hypervisor runs directly on the hardware and is responsible for handling CPU, Memory and interrupts. It is the first program running after exiting the bootloader. On top of Xen run a number of VMs. A running instance of a VM in Xen is called a domain or guest. A special domain, called domain 0 (dom0) contains the drivers for all the devices in the system. Domain 0 also contains a control stack to manage VM creation, destruction, and configuration.

3.2.2 Virtual Machines

VMs are virtualised environments, each running their own OS and applications. Xen supports two different virtualisation modes: Paravirtualisation (PV) and Hardware-assisted or Full Virtualisation (HVM). Both guest types can be used at the same time on a single Xen system. It is possible to use techniques used for Paravirtualisation in an HVM guest. This essentially creates a continuum between PV and HVM. This approach is called 'PV on HVM'. Xen guests are totally isolated from the hardware: in other words, they have no privilege to access hardware or I/O functionality. Thus, they are also called the unprivileged domains (or DomU).

3.2.3 Control Domain (Dom0)

Dom0 is a specialised VM with special privileges like the capability to access the hardware directly, handles all access to the system's I/O functions and interacts with the other VMs. It also exposes a control interface for system control. The Xen hypervisor is not usable without Dom0, which is the first VM started by the system.

3.2.4 Toolstack and Console

Dom0 contains a toolstack that allows a user to manage VM creation, destruction, and configuration. The toolstack exposes an interface that is either driven by a command line console, by a Graphical User Interface (GUI) or by a cloud orchestration stack like OpenStack or CloudStack.

3.2.5 Xen-enabled OS

A Xen Dom0 requires a Xen-enabled kernel. PV guests require a PV-enabled kernel. Linux distributions that are based on recent Linux kernel are Xen-enabled and usually contain packages that contain the Xen Hypervisor and Tools Xen (Toolstack and Console).

3.3 Guest Types

3.3.1 Xen Paravirtualisation (PV)

Paravirtualisation is an efficient and lightweight virtualisation technique introduced by Xen, later adopted by other virtualisation platforms (i.e. VMware call it VM Interface (VMI)). PV does not require virtualisation extensions from the host CPU. However, PV guests require a Xen-PV-enabled kernel and PV drivers, so the guests are aware of the hypervisor and can run efficiently without emulation or virtual emulated hardware. Linux kernels have been Xen-PV enabled from 2.6.24 using the Linux PVOPS framework.

3.3.2 Xen Full Virtualisation (HVM)

Full Virtualisation or Hardware-assisted virtualisation uses virtualisation extensions from the host CPU to virtualise guests. HVM requires Intel VT or AMD-V hardware extensions. Xen uses the **Qemu** processor emulator to emulate PC hardware, including BIOS, IDE disk controller, VGA graphic adapter, USB controller, network adapter etc. Virtualisation hardware extensions are used to boost performance of the emulation. Fully virtualised guests do not require any kernel support. This means that Windows OSs can be used as Xen HVM guest. HVM guests are usually slower than PV guests, because of the required emulation.

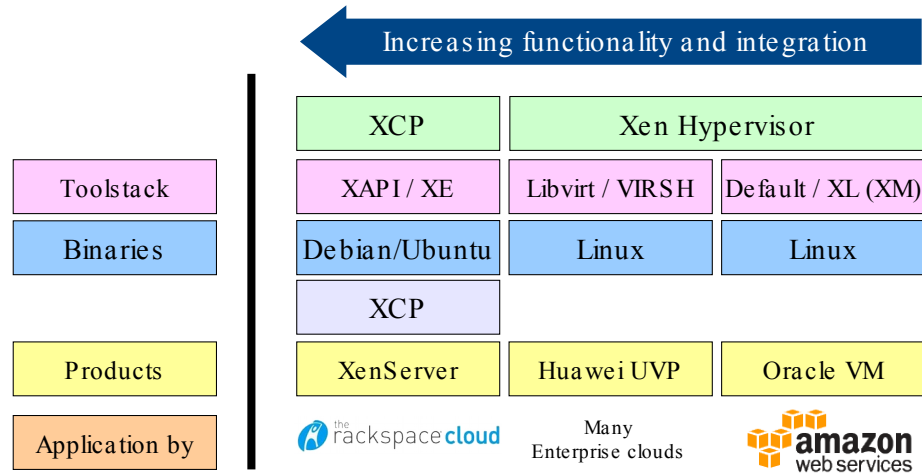
3.3.3 PV on HVM (PVHVM)

To boost performance, HVM guests can use special PV device drivers (PVHVM). These drivers are optimised PV drivers for HVM environments and bypass the emulation for disk and network IO, thus giving PV like (or better) performance on HVM systems. This means that it is possible to get optimal performance on guests OSs such as Microsoft Windows.

3.3.4 PV in an HVM Container (PVH)

There is another virtualisation mode called PVH since Xen 4.3. This is essentially a PV guest using PV drivers for boot and I/O. Otherwise it uses HW virtualisation extensions, without the need for emulation. PVH combines the best trade-offs of all virtualisation modes, while simplifying the Xen architecture.

3.4 Toolstacks, Management APIs and Consoles



* Huawei Unified Virtualisation Platform (UVP)

Xen can run with a number of different toolstacks. Each toolstack exposes an Application Programming Interface (API), which will run different tools. The diagram demonstrates choices, commercial products and examples of hosting vendors using specific APIs.

3.4.1 Default / XL

xl is a lightweight command line toolstack built using libxenlight. **xl** is shipped along with Xen and is the default toolstack for Xen.

3.4.2 libvirt / virsh

libvirt is a virtualisation API/toolkit used to manage various virtualisation technologies. The **libvirt** Xen driver provides the ability to manage Kernel-based VM (KVM) VMs. **virsh** is a Virtual Shell (virsh) is a command program that uses the libvirt API to create, delete, run, stop, and manage the hypervisor.

3.4.3 XAPI / XE

The Xen management API (XAPI) is the default toolstack for XenServer and the Xen Cloud Project (XCP). XenServer and XAPI are fully open source and XAPI has been ported to Linux distributions. **xe** is the command-line tool to configure and operate the XCP platform while **xscnsole** is a text-based User Interface (UI) to do basic tasks and configurations. XAPI is the API that cloud orchestration software used in:

- CloudStack
- OpenNebula
- OpenStack

3.4.4 Toolstack comparison

FEATURES	XL	XAPI	LIBVIRT
Purpose-built for Xen	✓	✓	✗
Basic VM Operations	✓	✓	✓
Managed Domains	✗	✓	✓
Live Migration	✓	✓	✓
PCI Passthrough	✓	✓	✓
Host Pools	✗	✓	✗
Flexible, Advanced Storage Types	✗	✓	✗
Built-in advanced performance monitoring (RRDs)	✗	✓	✗
Host Plugins (XAPI)	✗	✓	✗

3.5 The models

Well there are three top level models to choose from. The choice is the tried and trusted xl and libvirt, XenServer an XCP - XAPI shrink wrapped CentOS or an XCP-XAPI on Debian or Fedora based OS.

- Xen with a choice of Linux or Unix host distributions
 - Choice of distributions.
 - Xl and libvirt
- Xen Cloud Platform shrink wrapped as XenServer
 - Single ISO with CentOS as Dom0
 - XCP-XAPI
- Xen Cloud Platform with Xen API
 - Project Kronos
 - XCP-XAPI toolstack installed onto a pre-existing Debian and Ubuntu based OS deployment.
 - Project Zeus
 - XCP-XAPI toolstack installed onto a pre-existing Fedora and CentOS based OS deployment.

3.6 Control Domain (Dom0)

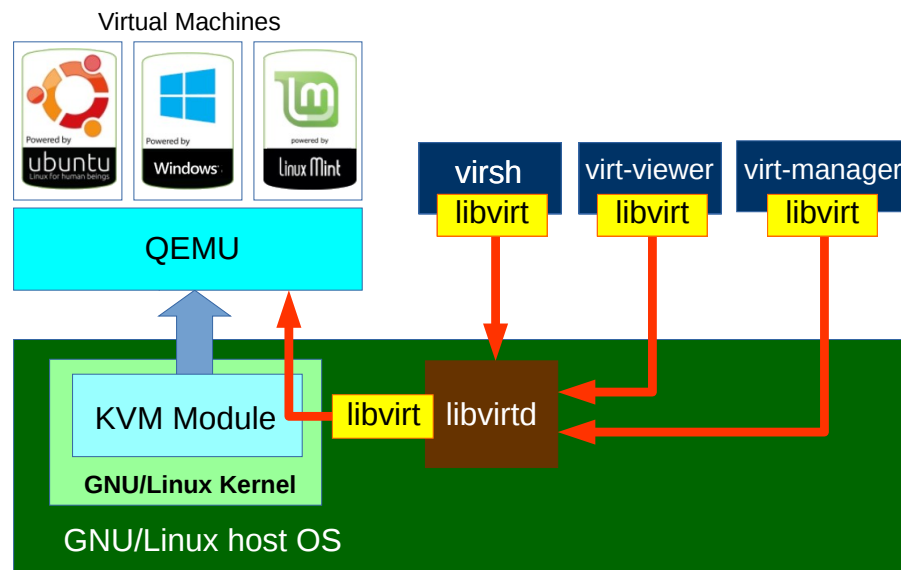
Xen requires a kernel as control domain. The choice of Dom0 comes down to:

- Familiarity with a specific distribution, BSD Unix, Debian, Fedora.
- Xen Hypervisor version that ships with the distribution of choice.
- Commercial support if required.

If you use XCP, you typically will not be interfacing much with Dom0.

[Additional material is available in the optional
Lab 5\(a\) – Xen Lab](#)

4. Kernel Virtual Manager (KVM)



The KVM hypervisor is included in major GNU/Linux releases today. As a result it has become the hypervisor of choice within the GNU/Linux community so it is available within the various distribution repositories. In fact KVM is a GNU/Linux kernel module that permits programs within user space to access either the Intel or AMD processor virtualisation features. As a result KVM VMs actually run as user space processes.

KVM uses the Quick EMUlator (QEMU), a generic and open source machine emulator and virtualiser for I/O hardware emulation. It can emulate a variety of processors on a guest processor and combined with the KVM kernel module it can approach native speeds. All combinations of 32-bit and 64-bit host and guest systems are supported, except 64-bit guests on 32-bit hosts.

KVM is managed via the libvirt API and tools such as *virsh*, *virtinstall*, *virt-clone*, *virt-viewer* and *virt-manager*.

KVM is a Type-1 hypervisor that runs directly on x86 hardware. The GNU/Linux interface makes it look like it is a hosted hypervisor running on it, but in fact each VM is running on the bare metal with the host GNU/Linux OS providing a launchpad for the hypervisor and then engaging in a co-processing relationship with the hypervisor.

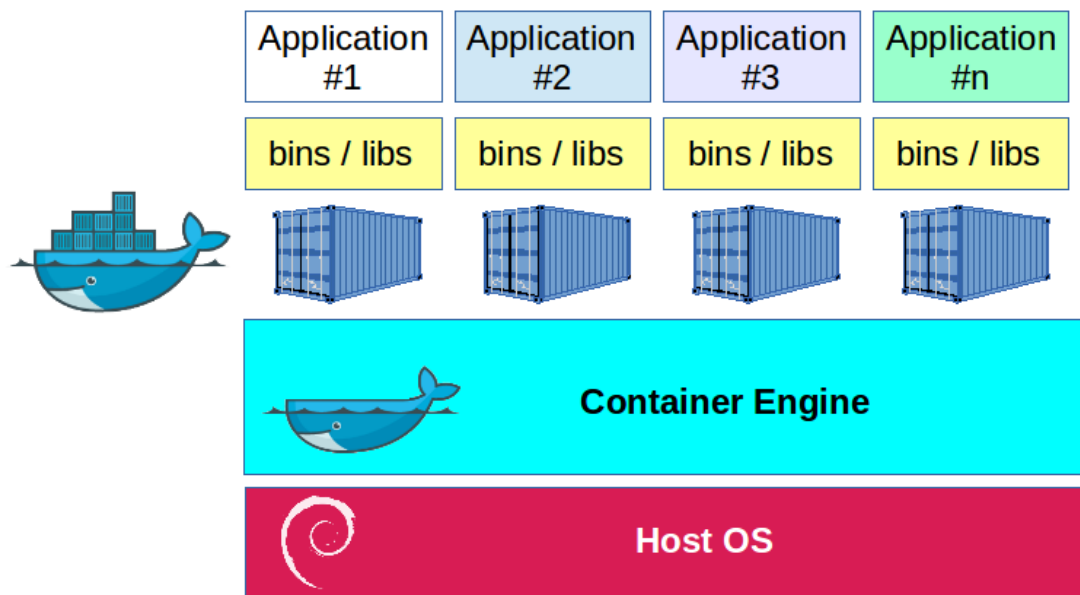
On x86 hardware, KVM relies on the hardware virtualisation instructions that are embedded in the processors and therefore these advanced chipset features must be enabled. Using these instructions the hypervisor and each guest VM run directly on the bare metal, and most of the resource translations are performed by the hardware.

Additional material is available in the optional
Lab 5(b) – KVM Lab

5. Containers with Docker



5.1 The Container concept



With container-based virtualisation the virtualisation layer runs as an application within the OS. The OS kernel runs on the hardware node with several isolated guest VM installed on top of it. The isolated guests are called containers.

However compared to standard virtualisation where each guest run a completely installed OS, with container based virtualisation there is only one host OS and each container runs a partial instance of the OS. The host OS ensure isolation of resources among all containers making each appear as a VM. The container contains any binaries or libraries that are unique to its function and has access to the binaries and libraries of the host OS.

In this way the container is only a partial OS which avoids the performance problems related to hardware access by means of driver virtualisation associated with traditional hypervisors and VMs.

5.2 What is Docker

The Linux kernel introduced Control GROUPS (**cgroups**). **cgroups** allow for the allocation of resources (CPU, memory, block I/O, network, etc.) to user defined groups of processes on the system. Additionally the kernel has a feature called **namespace** which allows the creation of different and separate instances of process groups that are isolated from each-other. Bringing **cgroups** and **namespace** together GNU/Linux has the concept of the Linux Container (LXC), where each LXC is an isolated environment for applications or OS virtualisation.

Docker is a utility that uses LXC to package an application and all its dependencies in an LXC that can then run on any GNU/Linux machine in a form of lightweight virtualisation.

5.3 Docker Hub

Docker Hub is centralised registry service of docker container images. It allows users to gain access to a vast number of pre-built images.

[Additional material is available in the optional
Lab 5\(c\) – Containers with Docker Lab](#)

6. VirtualBox

6.1 Getting VirtualBox

While the installation and operation of a Type-1 Hypervisor like Xen would be ideal the hardware is not available for such a lab. To have some practical experience with virtualisation therefore from this point VirtualBox Type-2 Hypervisor will be considered.

6.1.1 GNU/Linux

Using a Debian GNU/Linux based OS review the packages available for VirtualBox.

```
$ apt-cache search virtualbox | grep ^virtualbox
virtualbox-nonfree - Oracle Virtualbox virtualization software.
virtualbox-4.3 - Oracle VM VirtualBox
virtualbox-5.0 - Oracle VM VirtualBox
virtualbox - x86 virtualization solution - base binaries
virtualbox-dbg - x86 virtualization solution - debugging symbols
virtualbox-dkms - x86 virtualization solution - kernel module sources
for dkms
virtualbox-guest-additions-iso - guest additions iso image for
VirtualBox
virtualbox-guest-dkms - x86 virtualization solution - guest addition
module source for dkms
virtualbox-guest-source - x86 virtualization solution - guest
addition module source
virtualbox-guest-utils - x86 virtualization solution - non-X11 guest
utilities
virtualbox-guest-x11 - x86 virtualization solution - X11 guest
utilities
virtualbox-qt - x86 virtualization solution - Qt based user interface
virtualbox-source - x86 virtualization solution - kernel module
source
```

```
$ sudo apt-get install virtualbox-5.0
```

6.1.2 Apple OS X

Download and install the binary file below.

<http://download.virtualbox.org/virtualbox/5.0.4/VirtualBox-5.0.4-102546-OSX.dmg>

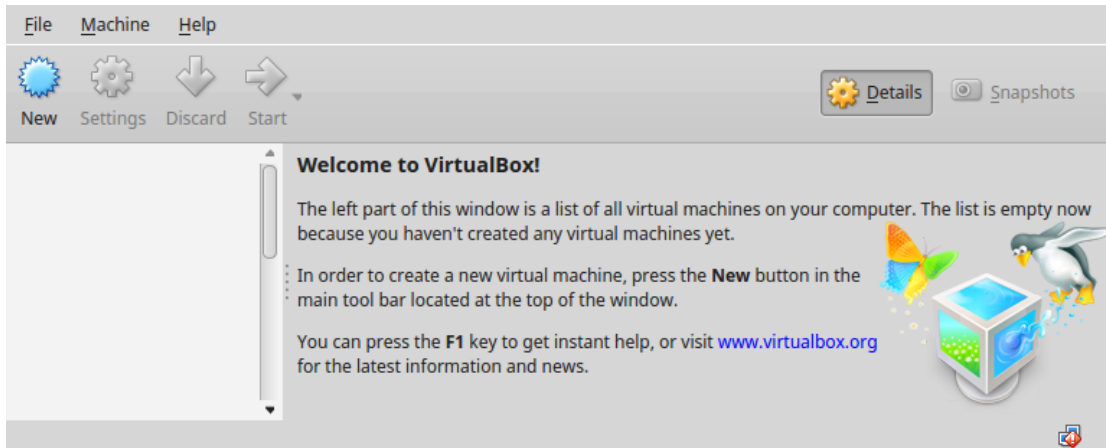
6.1.3 Microsoft Windows

Download and install the binary file below.

<http://download.virtualbox.org/virtualbox/5.0.4/VirtualBox-5.0.4-102546-Win.exe>

6.2 Building a VM on VirtualBox

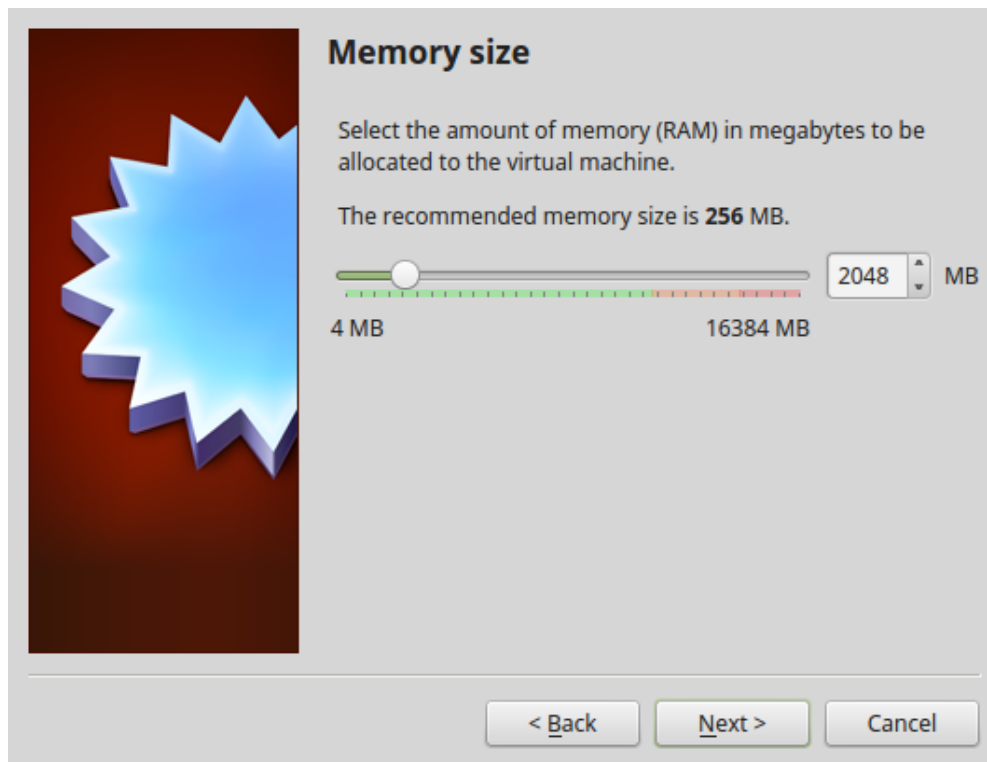
Run VirtualBox and the following pane is displayed.



Select New and the following pane I displayed permitting the selection of OS and a preferred name. For the purpose of the exercise Linux Mint v17.2 is being installed.



The next pane permits the percentage of host memory that can be allocated to the OS. In this case 2 GB of memory has been selected from a host system with 16 GB.



The next step is to create a virtualised Hard Disk.



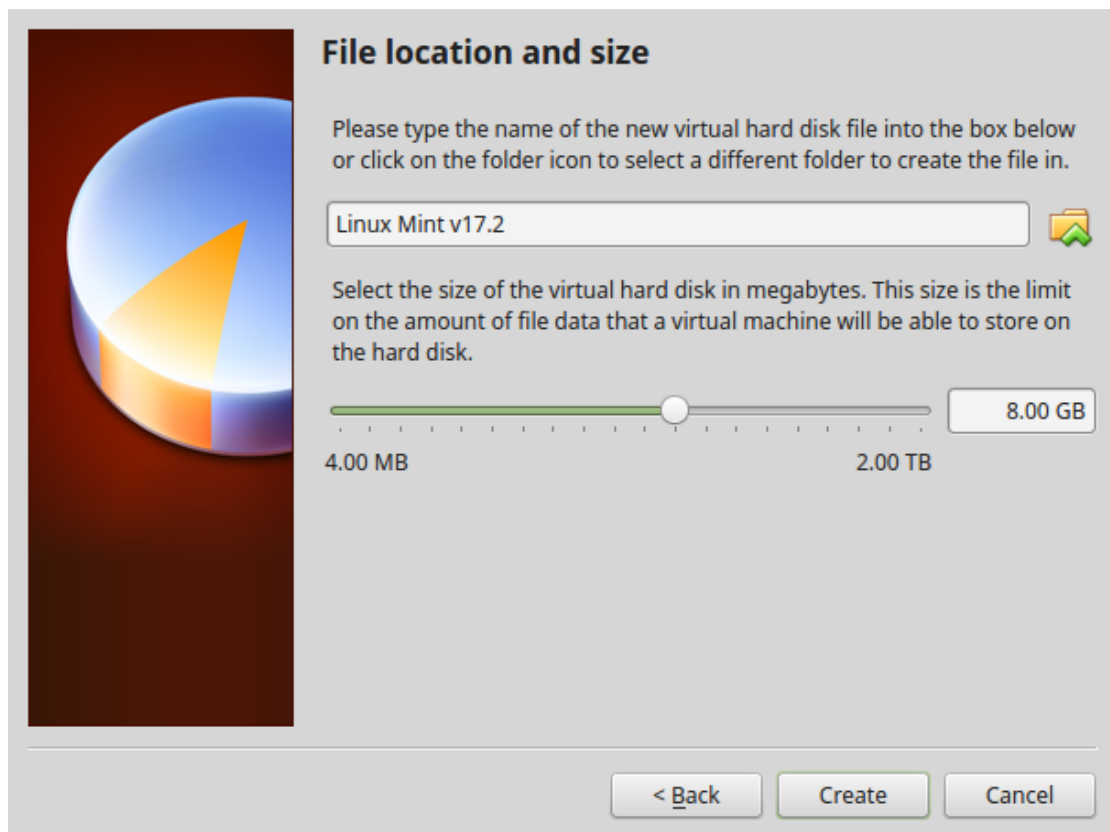
Select the VM Disk (VMDK), as this file format for virtual hard disk drives containers, while initially developed by VMware is now an open format and is one of the disk formats used in the Open Virtualization Format for virtual appliances. If the VM is never going to be migrated to another Hypervisor other than VirtualBox then VirtualBox Disk Image (VDI) maybe selected either.



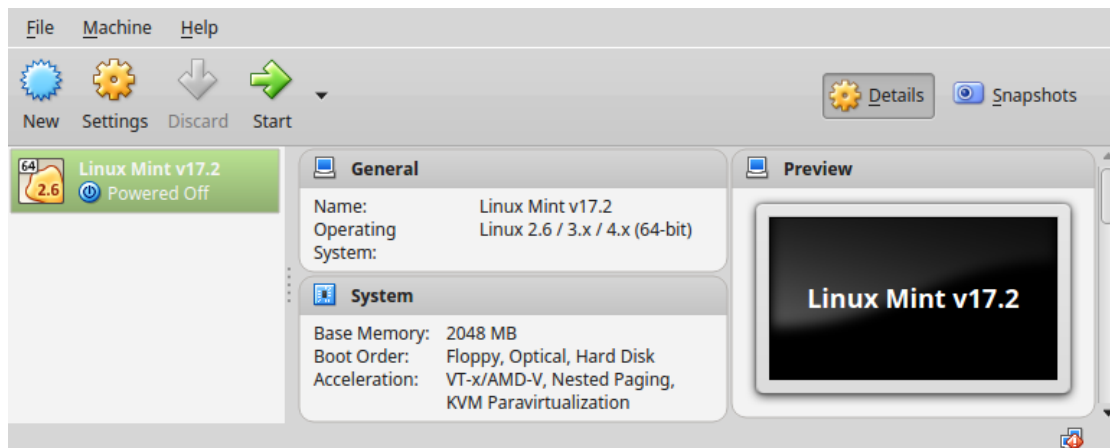
Allow for dynamically allocated to allow it grow as required by the VM.



Give the new virtual hard disk a name and a maximum size.

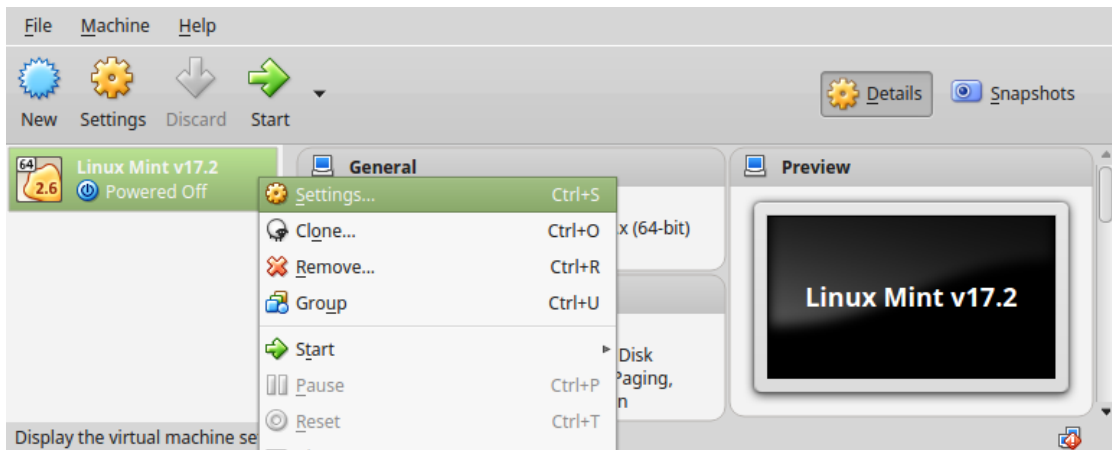


A new VM has been created.

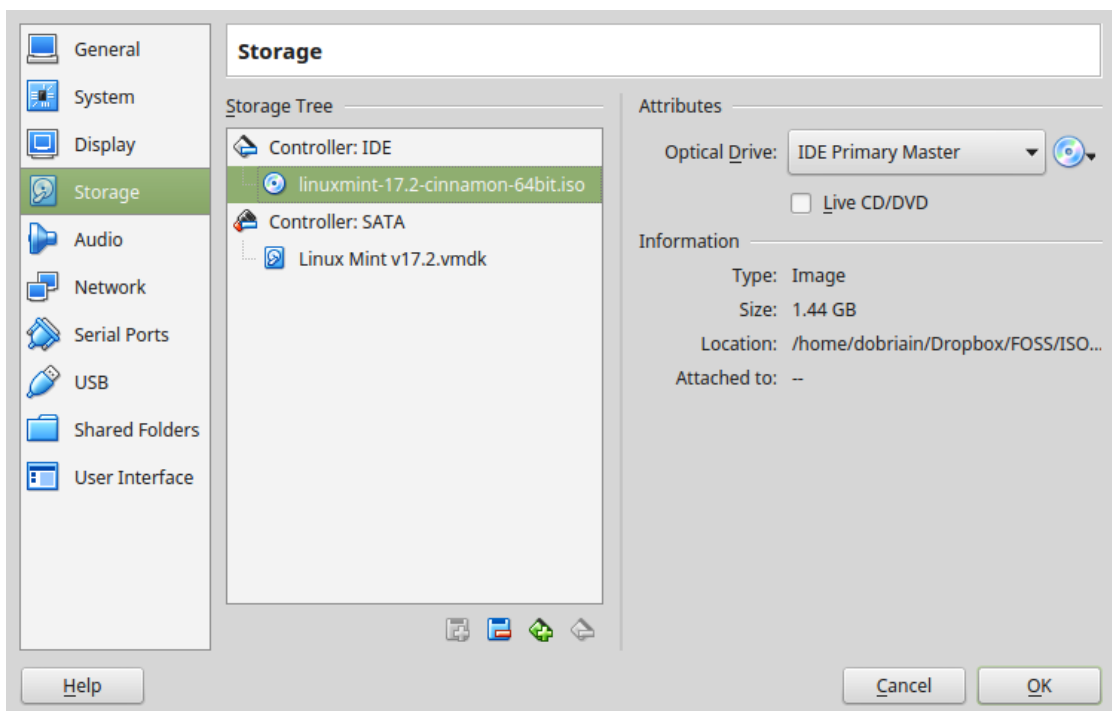


6.3 Installing an OS on a VM

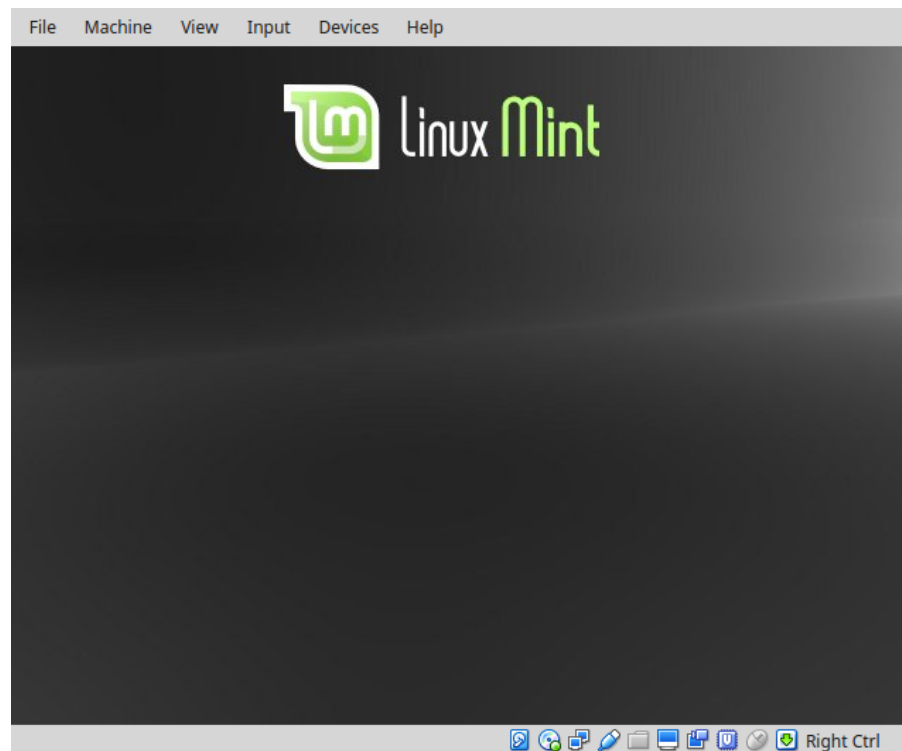
Before running the VM it is important that we assign a device where an OS installation exists, an ISO for example.



A downloaded Linux Mint v17.2 .iso file is selected, this allows the running VM to boot and install the new image.



No when the VM is selected and the green arrow clicked to start, the VM will boot into the Linux Mint install script.

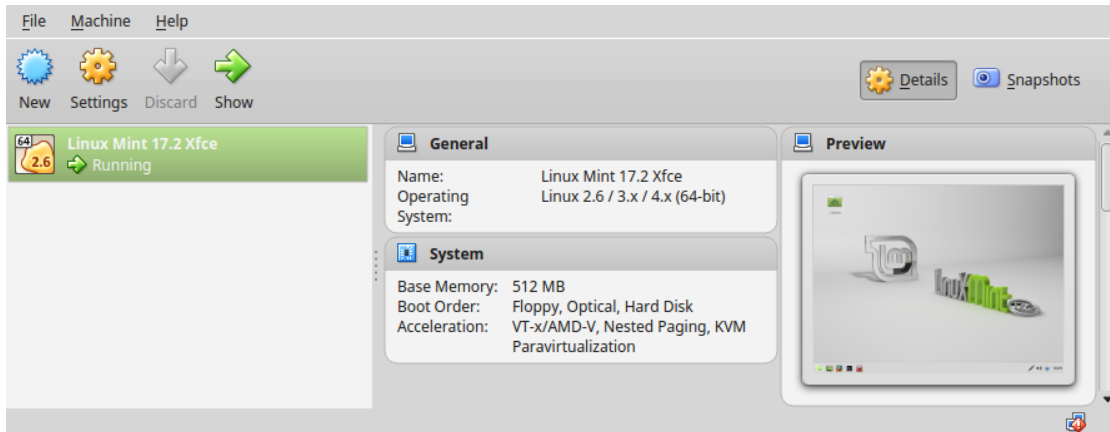


It may be unnecessary to install the OS into the VM as some distributions are supplied in .vmdk format. In this case instead of creating a virtual hard drive select the existing .vmdk file.

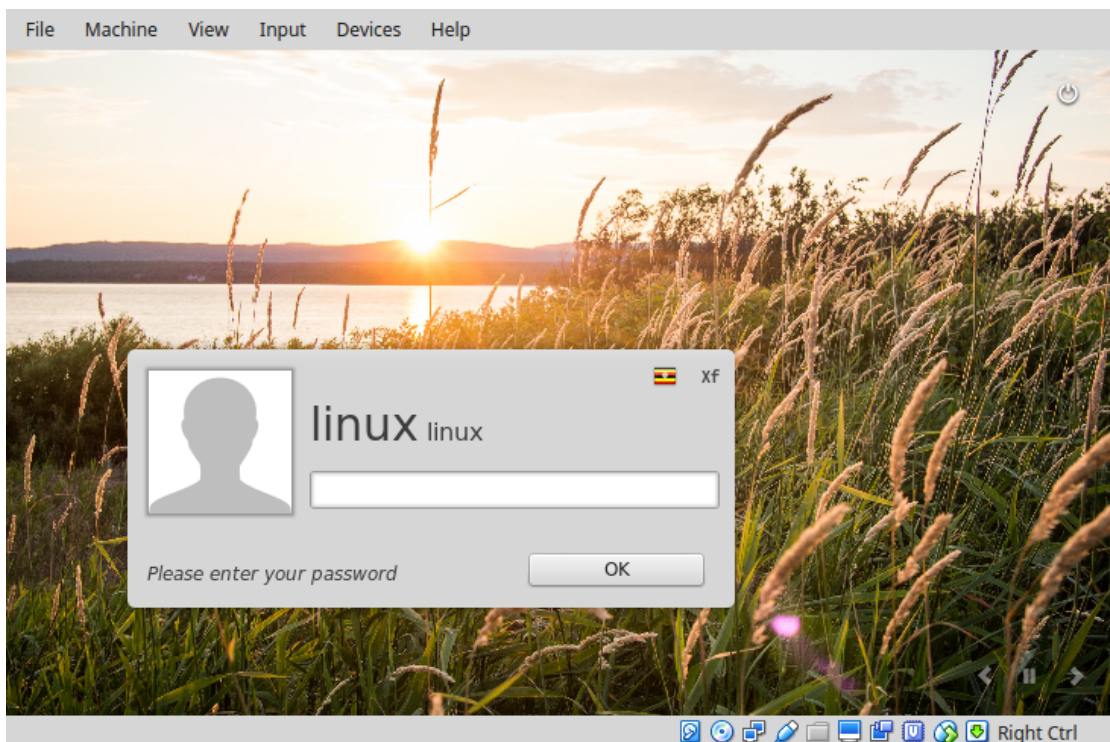


6.4 Running the VM

Now that the VM is installed run it. Here is the Kali Linux VM created from the .vmdk file.



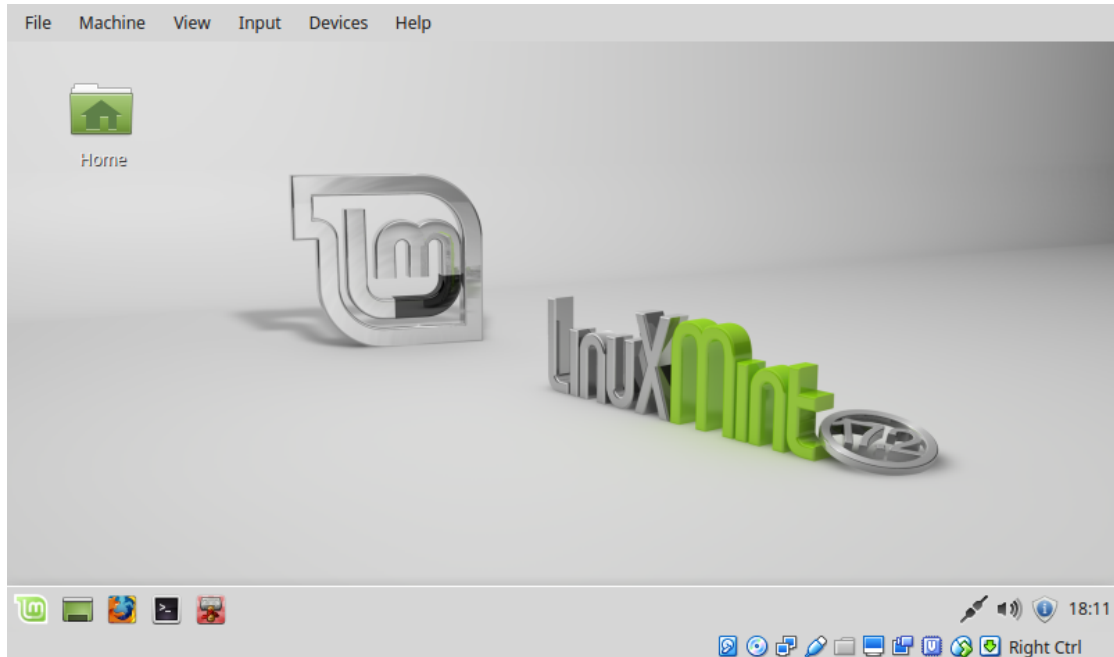
Clicking in the green arrow starts the selected VM.



7. Lab Exercise

Using the Linux Mint Linux image provided install VirtualBox, build the Open Virtual Appliance (.ova) image, install and run.

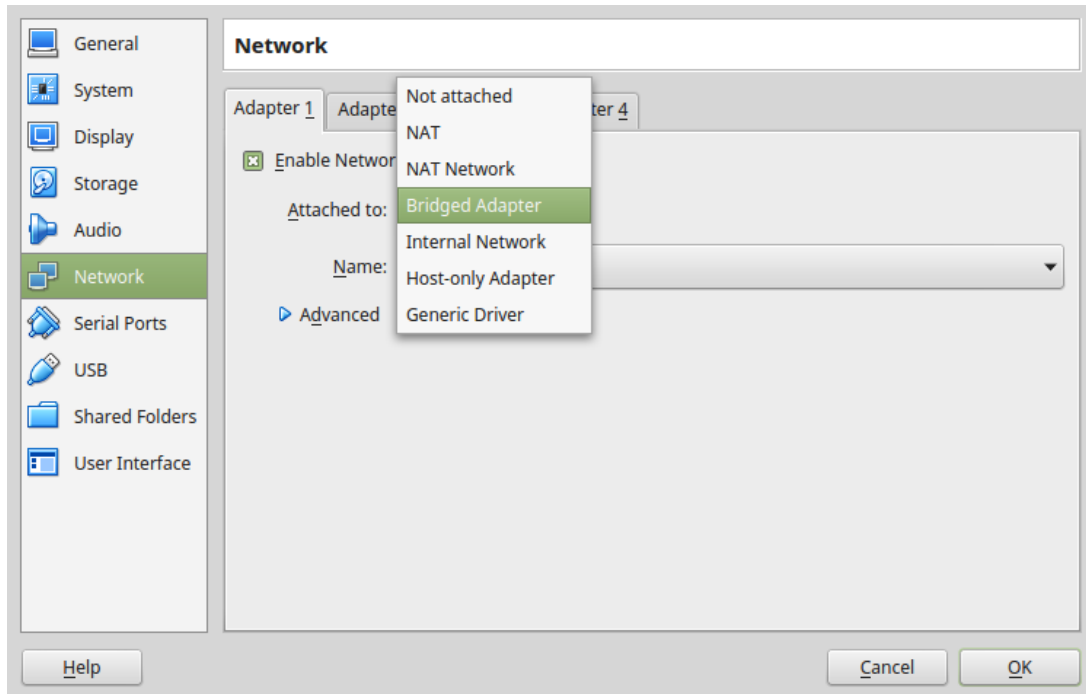
Login to the image with the default root username (linux) and password (mint).



Run up a shell by clicking on the fourth icon on the bottom toolbar and confirm connectivity with the Internet.

```
linux@linux-VirtualBox ~ $ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 08:00:27:1a:02:bd brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
        valid_lft 86044sec preferred_lft 86044sec
    inet6 fe80::a00:27ff:fe1a:2bd/64 scope link
        valid_lft forever preferred_lft forever
```

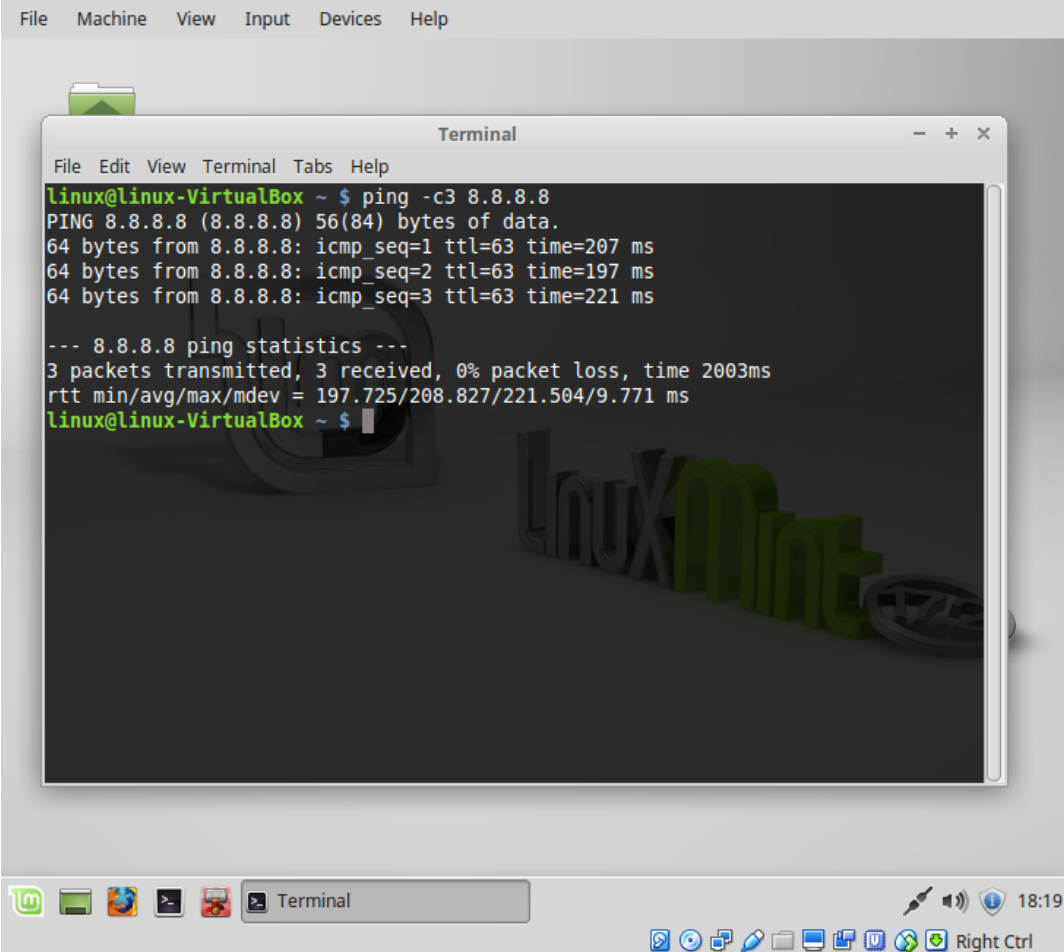

The IP Address is assigned by Network Address Translation (NAT) to the VM. It is possible to bridge the VM ethernet interface (eth0) with the active interface on the host to get an IP address from the real world Dynamic Host Configuration Protocol (DHCP) Server.



Whichever system is used the Internet Protocol (IP) Packet InterNet Groper (PING) test to the main google nameserver at 8.8.8.8 should elicit a response.

```
linux@linux-VirtualBox ~ $ ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=307 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=462 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=298 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 298.745/356.467/462.706/75.217 ms
linux@linux-VirtualBox ~ $
```



The image shows a screenshot of a Linux terminal window within a virtual machine. The terminal window is titled "Terminal" and has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal output shows a ping command being executed: `linux@linux-VirtualBox ~ $ ping -c3 8.8.8.8`. The output displays three successful ping requests to 8.8.8.8, each with 64 bytes of data and varying response times (207 ms, 197 ms, and 221 ms). Below the individual results, a summary line reads: `--- 8.8.8.8 ping statistics ---`, followed by: `3 packets transmitted, 3 received, 0% packet loss, time 2003ms` and `rtt min/avg/max/mdev = 197.725/208.827/221.504/9.771 ms`. The terminal prompt returns to `linux@linux-VirtualBox ~ $`. The background of the terminal window features a dark theme with a large, stylized "Linux" logo in green and white. The virtual machine's desktop environment is visible, including a taskbar at the bottom with icons for the terminal, a file manager, and other applications. The system tray on the right shows the time as 18:19 and the text "Right Ctrl".

```
File Machine View Input Devices Help

Terminal
File Edit View Terminal Tabs Help
linux@linux-VirtualBox ~ $ ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=207 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=197 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=221 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 197.725/208.827/221.504/9.771 ms
linux@linux-VirtualBox ~ $
```

8. Bibliography

von Hagen W (2008). Professional Xen Virtualization. Wiley / Wrox.

Takemura C, Crawford L (2009). The Book of Xen: A Practical Guide for the System Administrator. No Starch Press.

Dunlap G, Maresca S, West J (2012). From Datacenter to the Cloud - Featuring Xen and XCP. PFW Research LLC.

Xen. (2013). Xen Wiki. [ONLINE] Available at: http://wiki.xenproject.org/wiki/Main_Page. [Accessed 06 December 2013]. The Linux Foundation.

Citrix. (2013). Citrix delivers Cloud Solutions that enable Mobile Workstyles - Citrix. [ONLINE] Available at: <http://www.citrix.com/>. [Accessed 06 December 2013]. Citrix Systems Inc.

XenServer | Open Source Server Virtualization. (2013). XenServer | Open Source Server Virtualization. [ONLINE] Available at: <http://www.xenserver.org/>. [Accessed 06 December 2013]. Citrix Systems Inc.

Oracle VM VirtualBox . (2013). Oracle VM VirtualBox . [ONLINE] Available at: <https://www.virtualbox.org/>. [Accessed 06 December 2013]. Oracle Corporation.

Vmware. (2015). Virtual Disk Format 1.1. Available at: <http://www.vmware.com/app/vmdk/?src=vmdk> [Accessed 09 September 2015]. Vmware Inc.

This page is intentionally blank