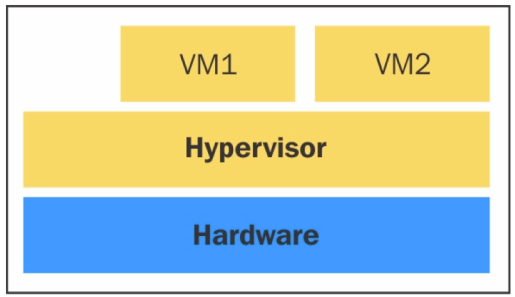
Containers, Docker, and more Vagrant

References

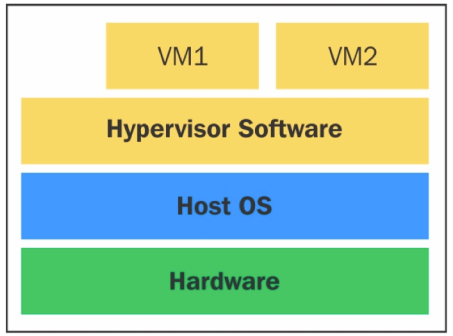
- Learning Docker - Second Edition by Jeeva S. Chelladhurai; Vinod Singh; Pethuru RajPublished ,by Packt Publishing, 2017
- https://www.docker.com/what-container
- http://www.thehyperadvisor.com/vmware/get-hyper-v-2012-running-vmware-fusion-6-x/
- https://stackoverflow.com/questions/30379381/docker-command-not-found-even-though-installed-with-apt-get
- https://stackoverflow.com/questions/39325394/initialize-permission-denied-rb-sysopen-vagrant-up
- https://atlas.hashicorp.com/minimal/boxes/xenial64
- https://github.com/moby/moby/issues/30762
- https://www.vagrantup.com/docs/vagrantfile
- https://www.vagrantup.com/docs/virtualbox/
- https://www.vagrantup.com/docs/virtualbox/configuration.html
- https://www.vagrantup.com/docs/provisioning/

Virtual Machine Review

- Virtual Machines (VMs) abstract hardware
  - In a type 1 vm the hypervisor runs directly on the hardware
  - In a type 2 vm the hypervisor runs on the host OS



Type 1



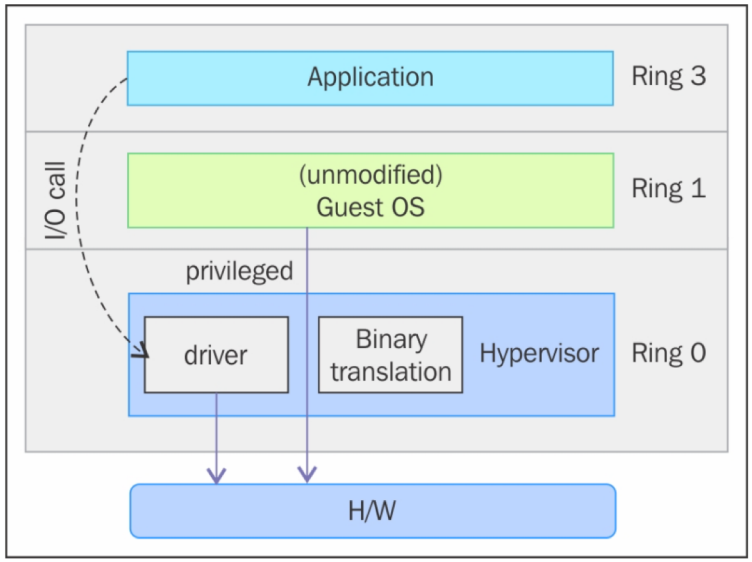
Type 2

Virtual Machine Review

- Besides where the Hypervisor resides (on top of hardware, or on top of the host OS), VM systems support Full Virtualization or Paravirtualization

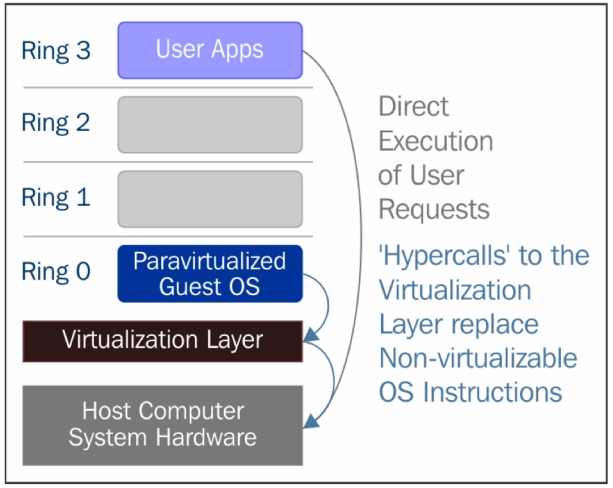
Virtual Machine Review

- Full virtualization
  - The guest OS runs in ring 1
  - The Hypervisor/VMM runs in ring 0



Virtual Machine Review

- Paravirtualization addresses the performance overhead of binary translation and emulation used in full virtualization by using modified versions of guest operating systems that access ring 0.



Virtual Machine Issues, Containers

- While Virtual Machine technologies have been well vetted and widely used, using VM's – especially for huge deployments – has overhead.
  - Each VM have a guest operating system
  - VM's take some time to spin up
  - VM's can get big

- A new technology – **containers** – provides a lightweight way to package and deploy software.

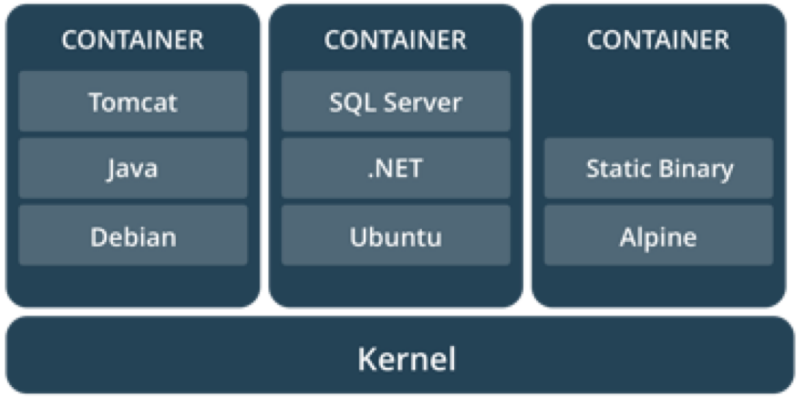
Containers

- A container makes use of the host environments OS, while providing an isolated environment in which software can be run
- A container does not need a guest OS.

Containers

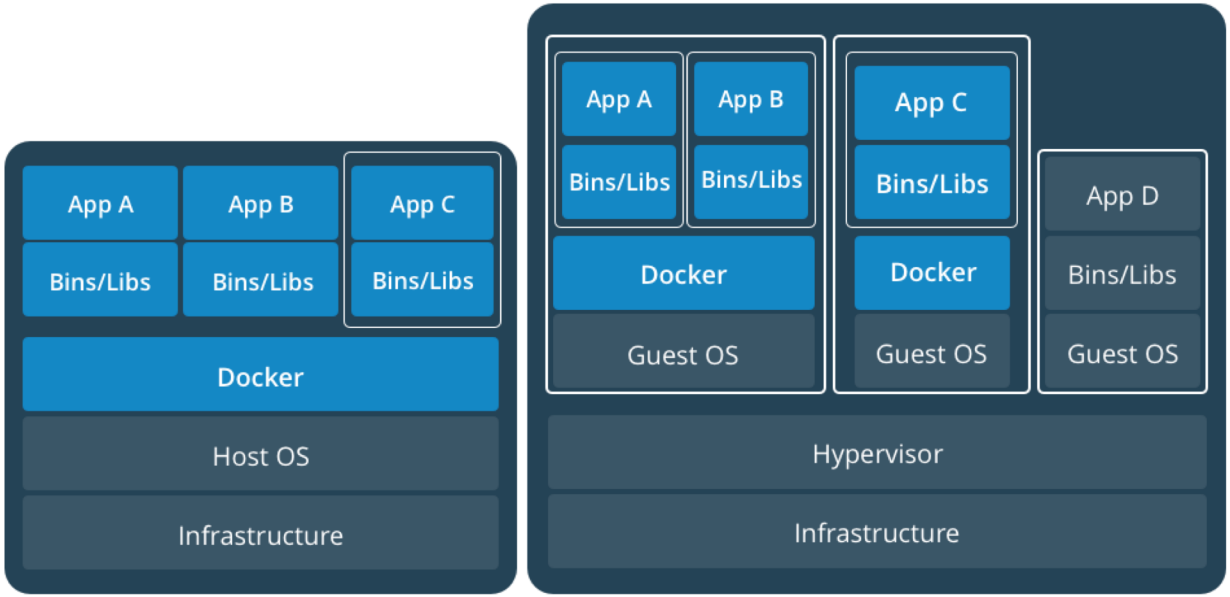
- Basically – a container only contains what it needs to run the applications embedded within it. This typically includes:
  - the applications
  - libraries and framework used by the applications
  - unlike a VM deployment, a container does NOT need an (guest) OS

Containers



From https://www.docker.com/what-container - each container contains the software it needs to run while utilizing the host environment's kernel

Containers



From https://www.docker.com/what-container#/virtual_machines: a containerized environment vs. a VM environment

Containers: Docker

- The use of container technologies is growing exponentially
- Software based on containers is: lightweight (lighter than a VM), portable, rapidly deployable, and easy extended.
- The leading container technology is called **Docker**.

Containers: Docker

- Once you start looking into Docker you will also see some technologies that have been developed to deploy and manage Docker containers. These include:
  - Kubernetes
  - Mesos
  - Docker Swarm
- We will cover these technologies in the Advanced DevOps class.

Docker

- Let's get starting leaning Docker by running looking at some simple examples.

- But first, lets download Docker. We will use **the Docker Community Edition:**

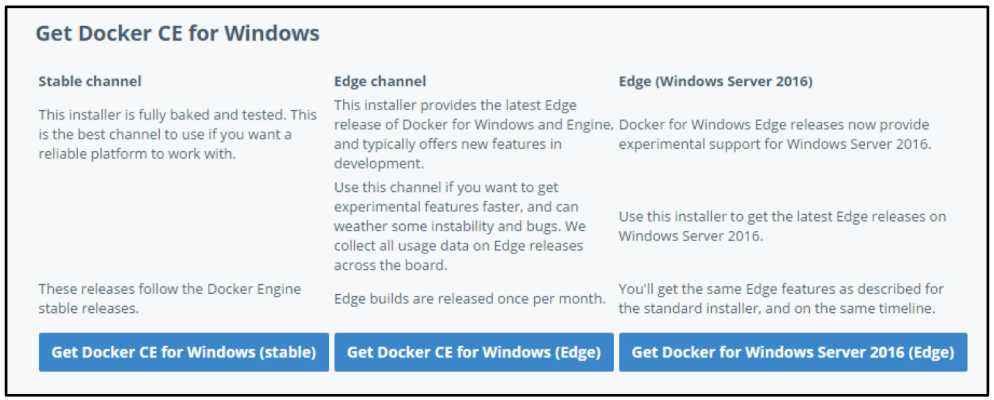
  https://www.docker.com/community-edition

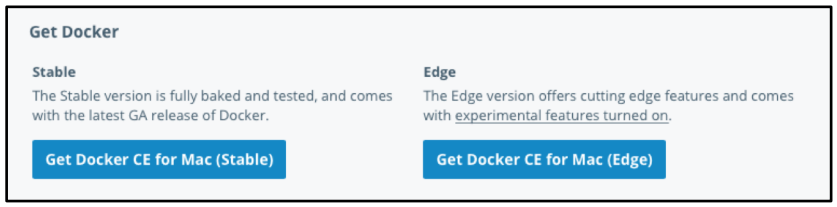
- Get the version for you system:

Docker

Docker also provides version for several Linux distributions and AWS

Docker



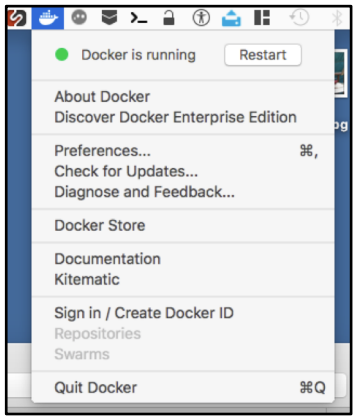


Download the "Stable Channel" version of Docker for your platform. Following the instructions on the site to install Docker

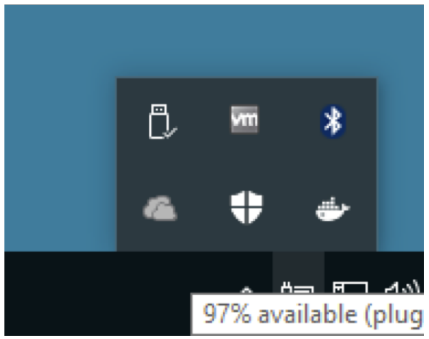
Docker

- Docker runs as a background service (you may have to start it manually).
- When Docker is running you will see the Docker Whale:

Mac: access Docker from the menu bar at the top



Docker in Windows 10



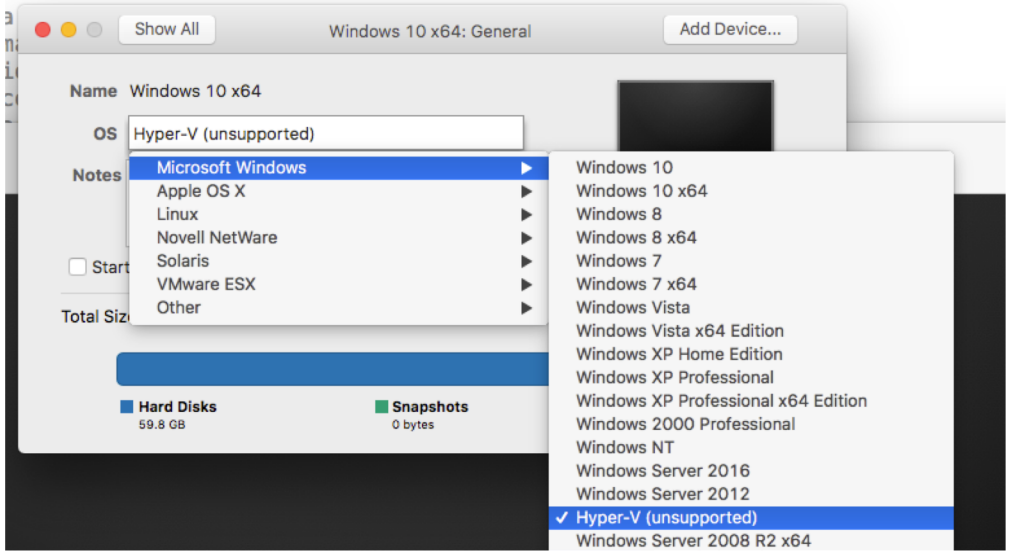
Installing Docker in a Virtual Machine

- While Docker is often considered a replacement for VMs, you can run Docker in a VM. To do so you must enable hypervisor applications within your VM. This is also called enabling "nested virtualization".

- The following slides show the settings what were enabled in VMWare Fusion running on a Mac to enable running Docker in a Windows (guest os) VM.

- Also, the settings are displayed for VirtualBox

Installing Docker in a Virtual Machine VMWare Fusion



For VMWare Fusion;

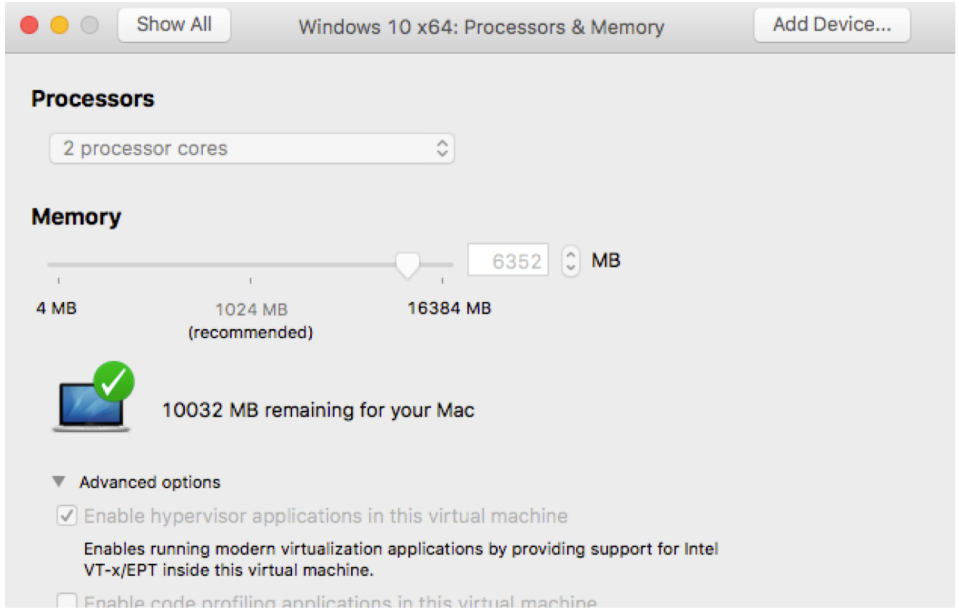
a) stop your VM
b) under Virtual Machine settings select general
c) change to OS to: Hyper-V (unsupported)

Installing Docker in a Virtual Machine VMWare Fusion

Next;

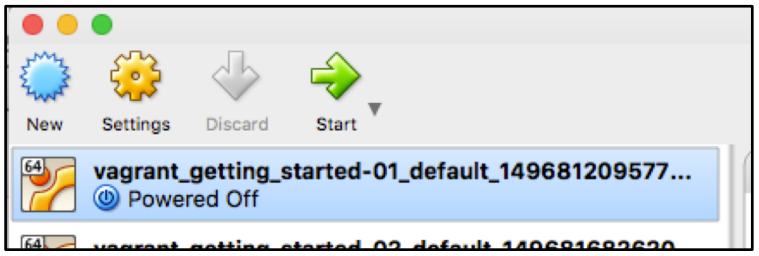
a) make sure VM is stopped
b) under Virtual Machine settings select Processor and Memory

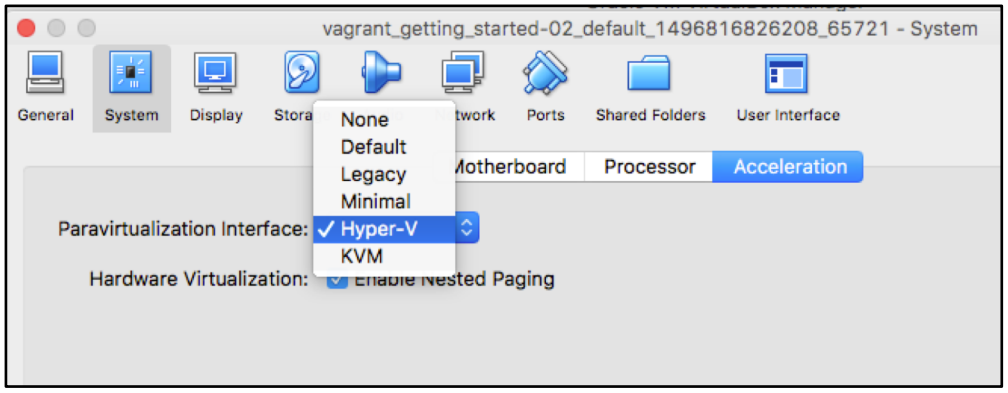
Installing Docker in a Virtual Machine VMWare Fusion



Under Advanced, check Enable Hypervisor application in this virtual machine

Installing Docker in a Virtual Machine VirtualBox





To enable Hyper-V when using VirutalBox

a) make sure VM is not running
b) Select Settings
c) Select System
d) Select Acceleration
e) In Paravirtual interface select:

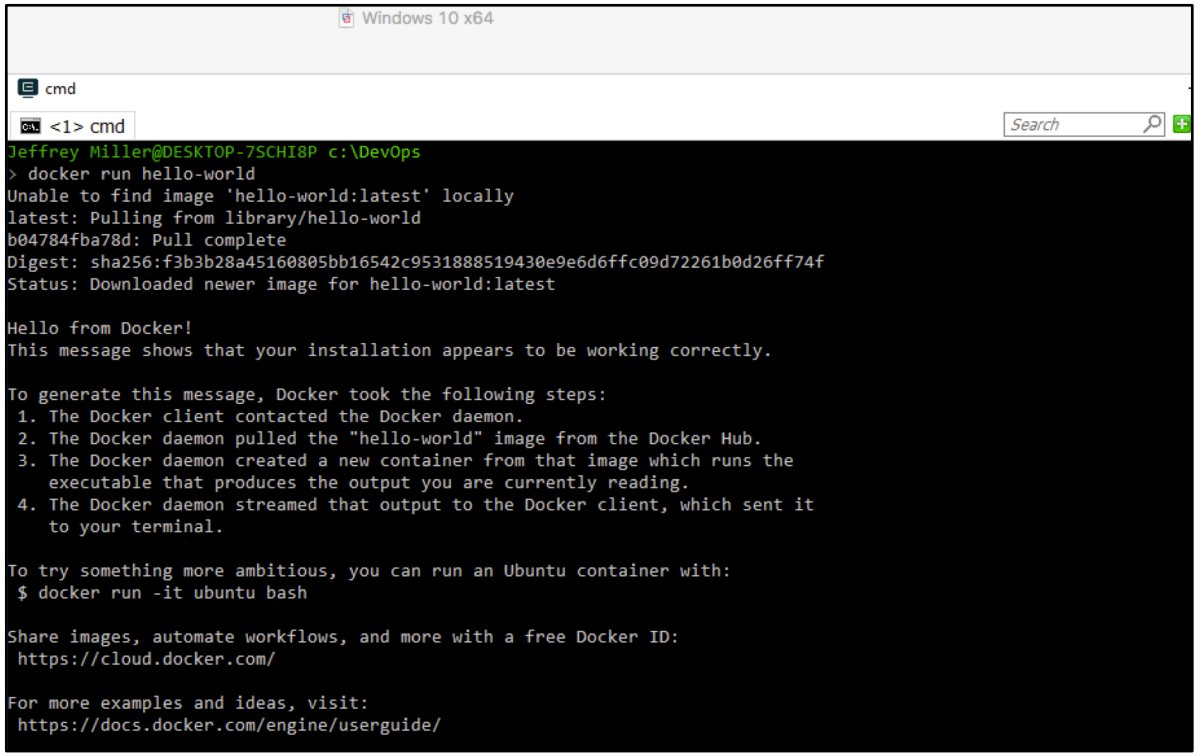
   Hyper-V or KVM – depending on what the host OS is.

Docker hello-world Example

- As is common when starting to work with a new technology – let's run the hello world docker example

- The following slides show docker hello-world in:
  - Windows 10, running in a VM in VMWare Fusion
  - Mac

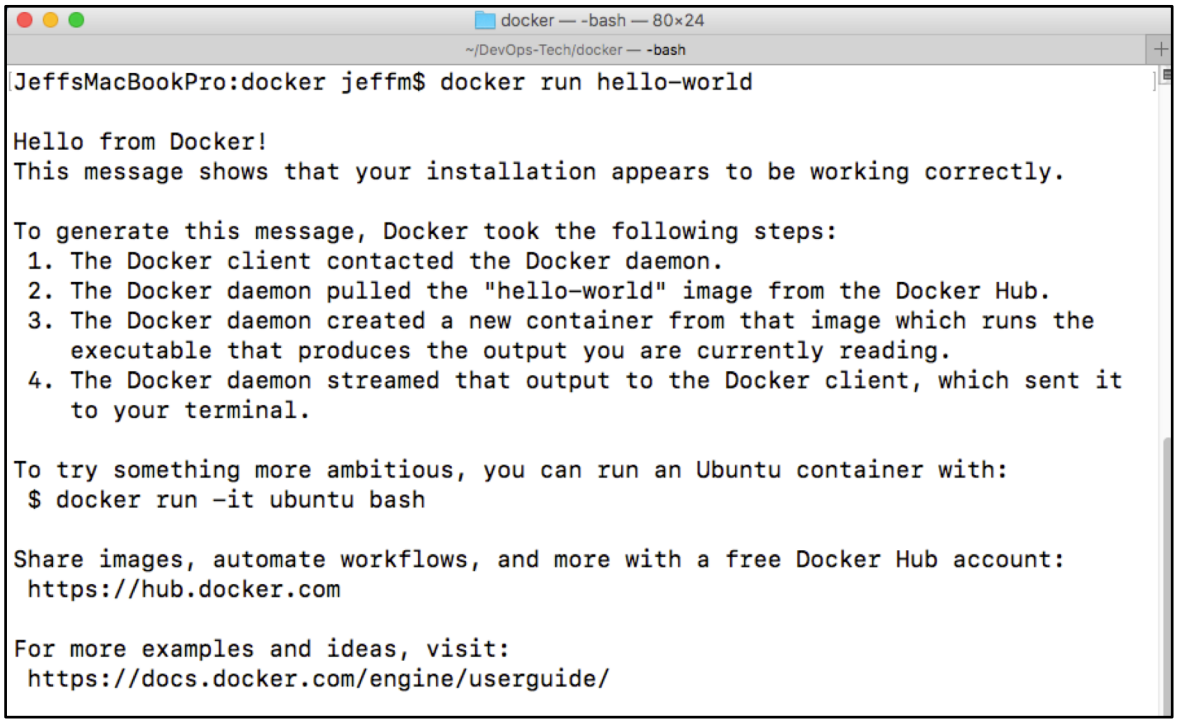
Docker hello-world Example Windows 10



In a command prompt:

docker run hello-world

Docker hello-world Example
Mac: Sierra



In a terminal:

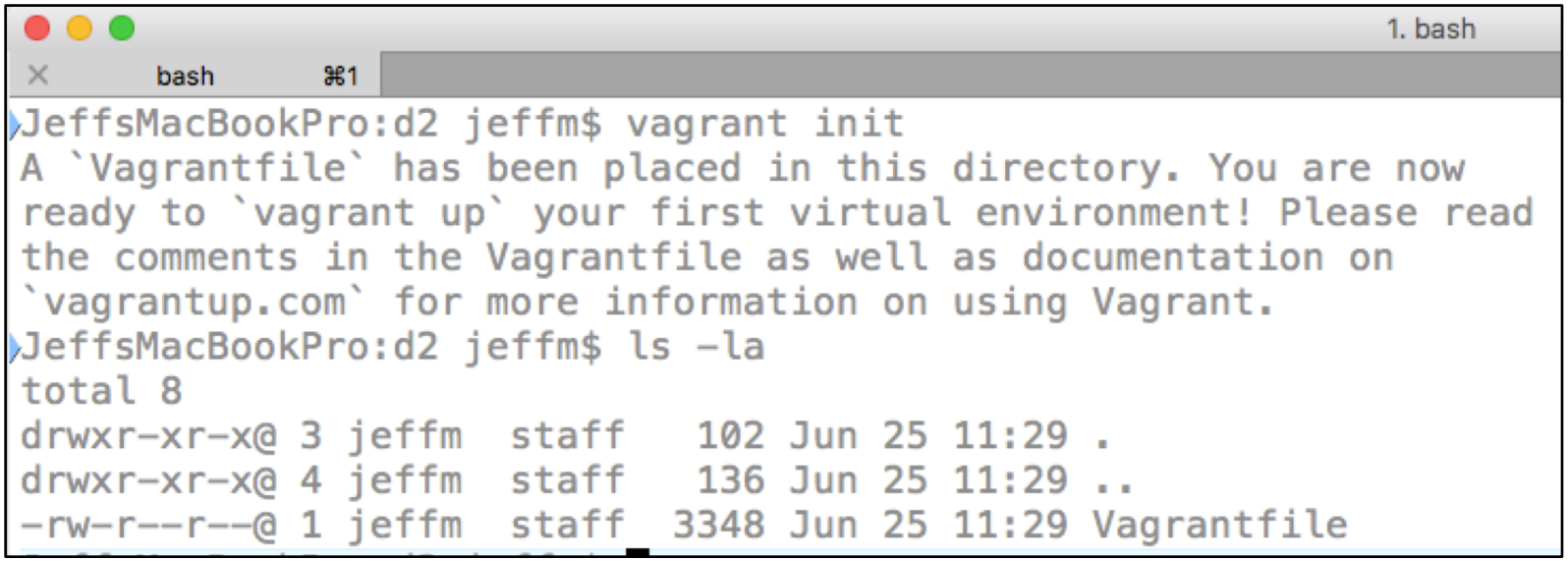
docker run hello-world

Example 02

- In the next example we will:
  - (again) look at the Vagrant Registry
  - Write a Vagrant file to:
    - get Ubuntu into a VirtualBox support VM
    - add a script into the Vagrant file to install Docker
    - run docker hello-world in

Example 02: vagrant init

- First, create a new folder – your instructor called it **d2**

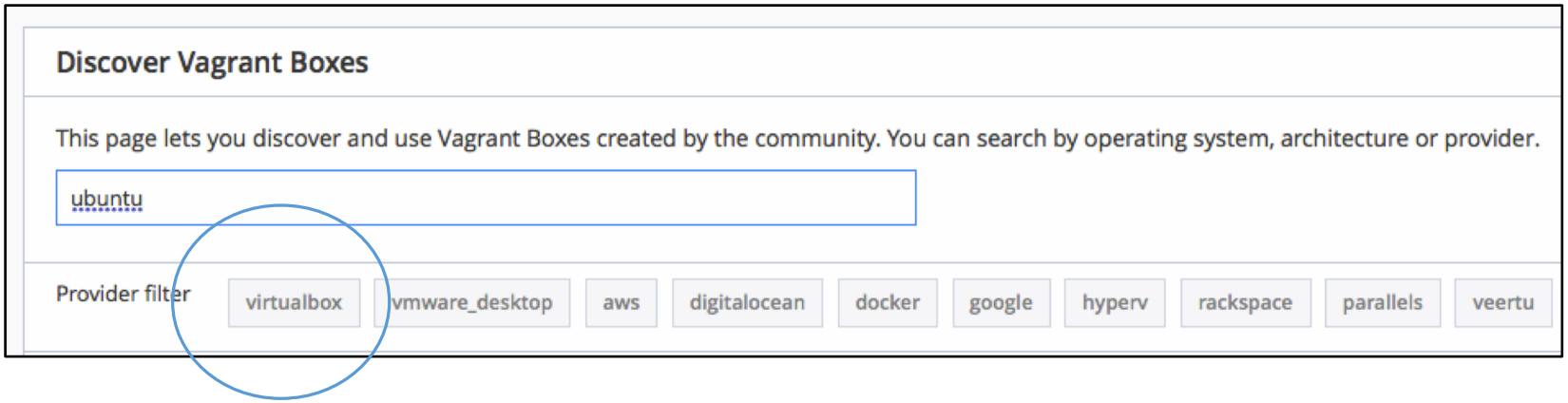


Example 02: Vagrant Registry

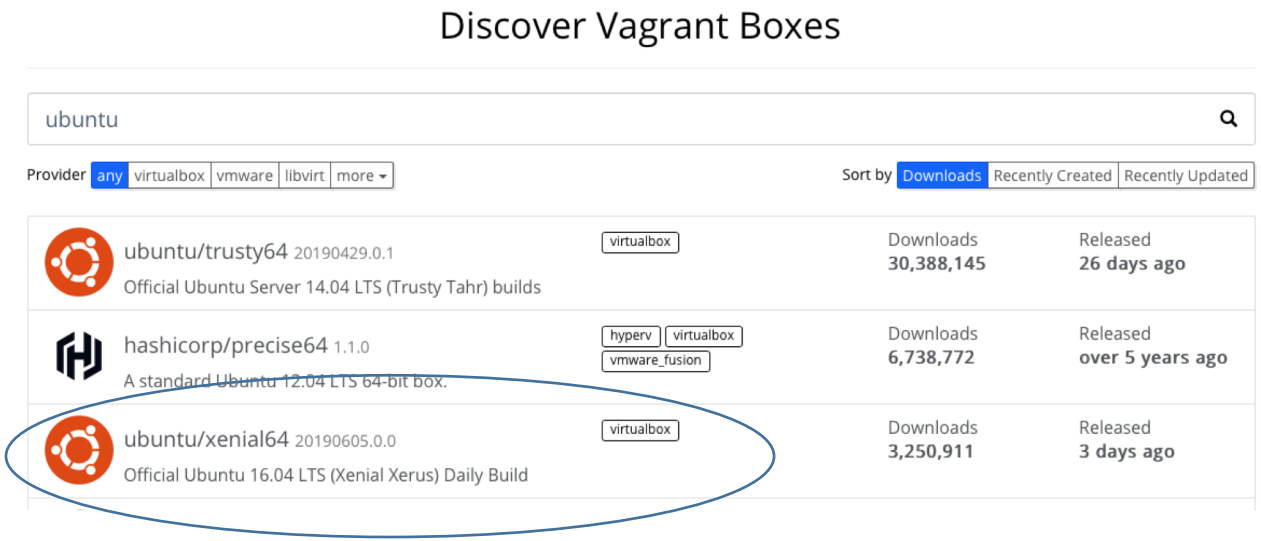
- Got to Vagrant Cloud:

  https://app.vagrantup.com/boxes/search:

Example 02: Vagrant Registry



Example 02: Vagrant Registry



Get xenial64 – Ubuntu 16.04

Example 02: Vagrant Registry

ubuntu / xenial64 Vagrant box

How to use this box with Vagrant:

Vagrantfile New

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/xenial64"
end
```

v20190605.0.0 currently released version

Example 02: Vagrant Registry

ubuntu / xenial64 Vagrant box

How to use this box with Vagrant:

Vagrantfile | New

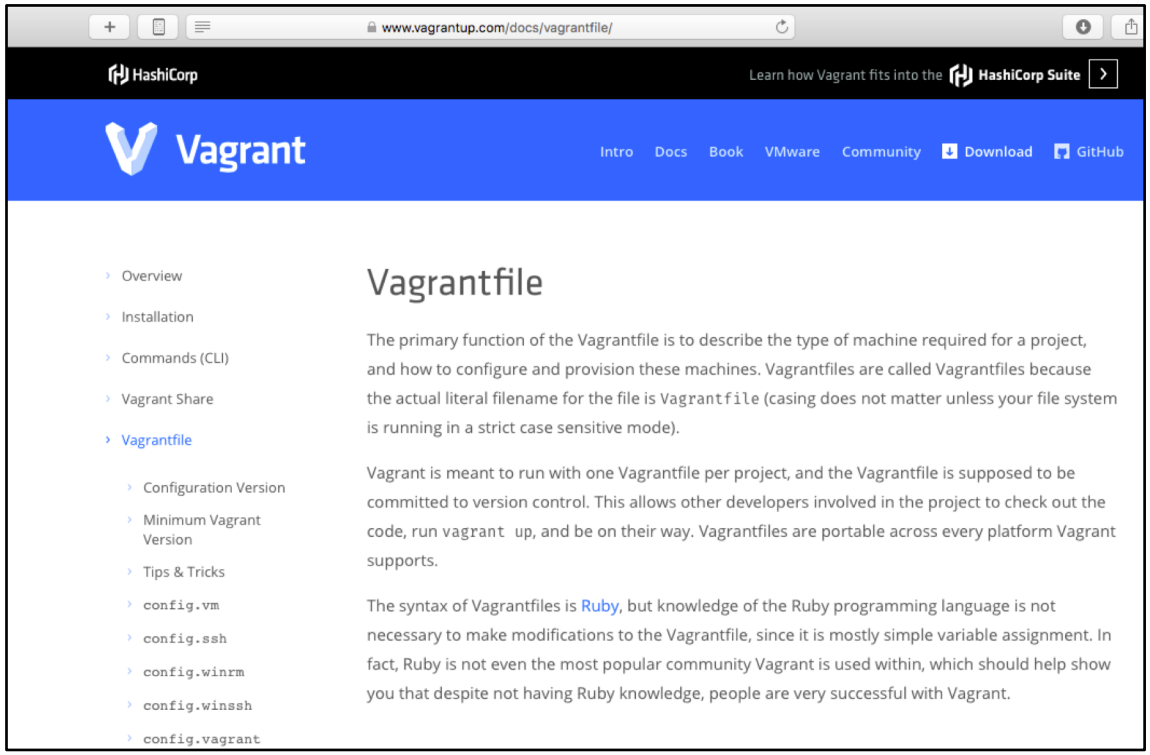
```
vagrant init ubuntu/xenial64
vagrant up
```

Example 02: Vagrantfile Reference

- https://www.vagrantup.com/docs/vagrantfile/

  documents vagrant files

Example 02: Vagrantfile Reference



Example 02: Vagrantfile

- Our first edit in the vagrantfile is to set the box to ubuntu/trusty64

```
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://atlas.hashicorp.com/search.
  config.vm.box = "ubuntu/xenial64"
…
```

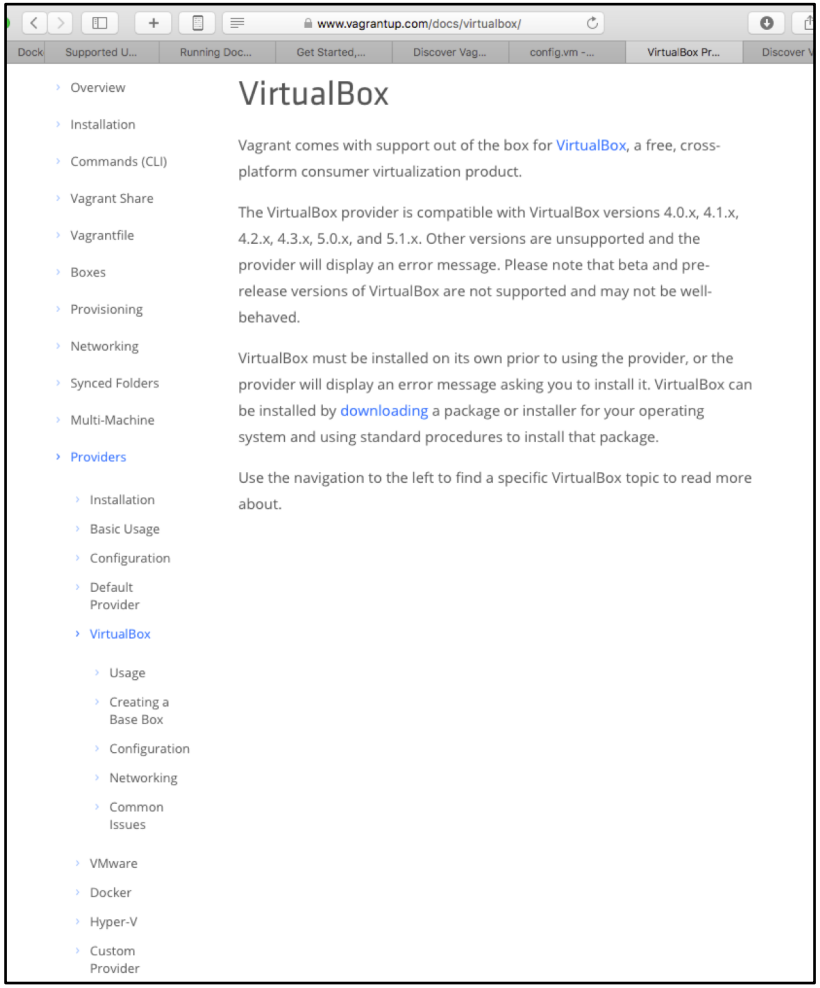
Example 02: Vagrantfile

- Next, we will add some VirtualBox specific settings
- config.vm.provider – sets up provider specific settings:

> `config.vm.provider` - Configures provider-specific configuration, which is used to modify settings which are specific to a certain provider. If the provider you are configuring does not exist or is not setup on the system of the person who runs `vagrant up`, Vagrant will ignore this configuration block. This allows a Vagrantfile that is configured for many providers to be shared among a group of people who may not have all the same providers installed.

from:

https://www.vagrantup.com/docs/vagrantfile



Vagrant supports:

- VirtualBox
- VMWare
- Docker
- Hyper-V
- and Custom Providers

- In this example we will use VirtualBox

Example 02: Vagrantfile

- We will add configuration for the following VirtualBox settings:
  - we will add a GUI (please see all disclaimers about adding a GUI on the following slides)
  - we will name the VM
  - we will specify how much memory the VM has
  - we will specify how many CPUs the VM has

Example 02: Vagrantfile

- VirtualBox specific vagrant file settings:

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/xenial64"
  …
 config.vm.provider "virtualbox" do |virtbox|
    # Display the VirtualBox GUI when booting the machine
    virtbox.gui = true

    # Customize the amount of memory on the VM:
    virtbox.memory = "2048"

    # VM name
    virtbox.name = "d2VM-01"
 end
 …
```

Example 02: Vagrantfile

- Next, since we know we will be installing MongoDB using Docker within our VM, let's make sure the default ports MongoDB needs are open in the VM.

- https://docs.mongodb.com/manual/reference/default-mongodb-port/

  lists the default ports.

- For our basic usage of MongoDB we need to make sure ports: 27017 and 28017 are open in the VM

- We will map these port to different ports on our host machine

Example 02: Vagrantfile, IP Ports

- Question – how can we determine which IP port are available on the host machine?
- The following slides will show how to do this in MS-Windows and Mac
- NOTE: there are many different ways to do this. However, when we work in DevOps, because we are interested in automating everything we do, we will use terminal/command-line methods instead of GUI-based methods

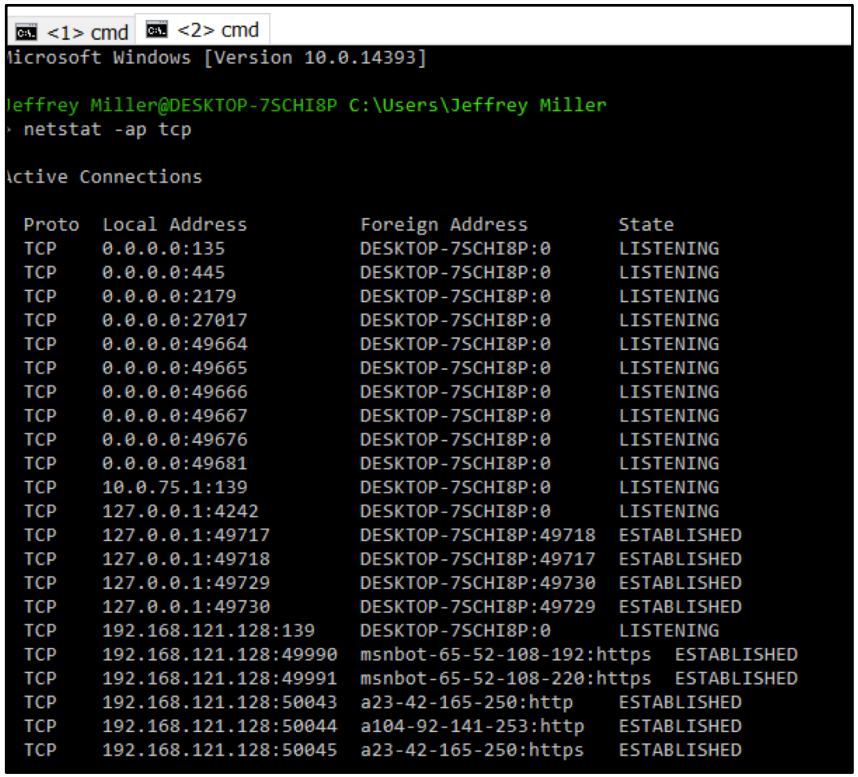
Example 02: Vagrantfile,IP Ports

- Windows, Cygwin on Windows, Linux, and Mac support the netstat command:

```
NETSTAT(1)                BSD General Commands Manual               NETSTAT(1)

NAME
     netstat -- show network status

SYNOPSIS
     netstat [-AaLlnW] [-f address_family | -p protocol]
     netstat [-gilns] [-v] [-f address_family] [-I interface]
     netstat -i | -I interface [-w wait] [-c queue] [-abdgqRtS]
     netstat -s [-s] [-f address_family | -p protocol] [-w wait]
     netstat -i | -I interface -s [-f address_family | -p protocol]
     netstat -m [-m]
     netstat -r [-Aaln] [-f address_family]
     netstat -rs [-s]

DESCRIPTION
     The netstat command symbolically displays the contents of various net-
     work-related data structures.  There are a number of output formats,
     depending on the options for the information presented.  The first form
```

man netstat

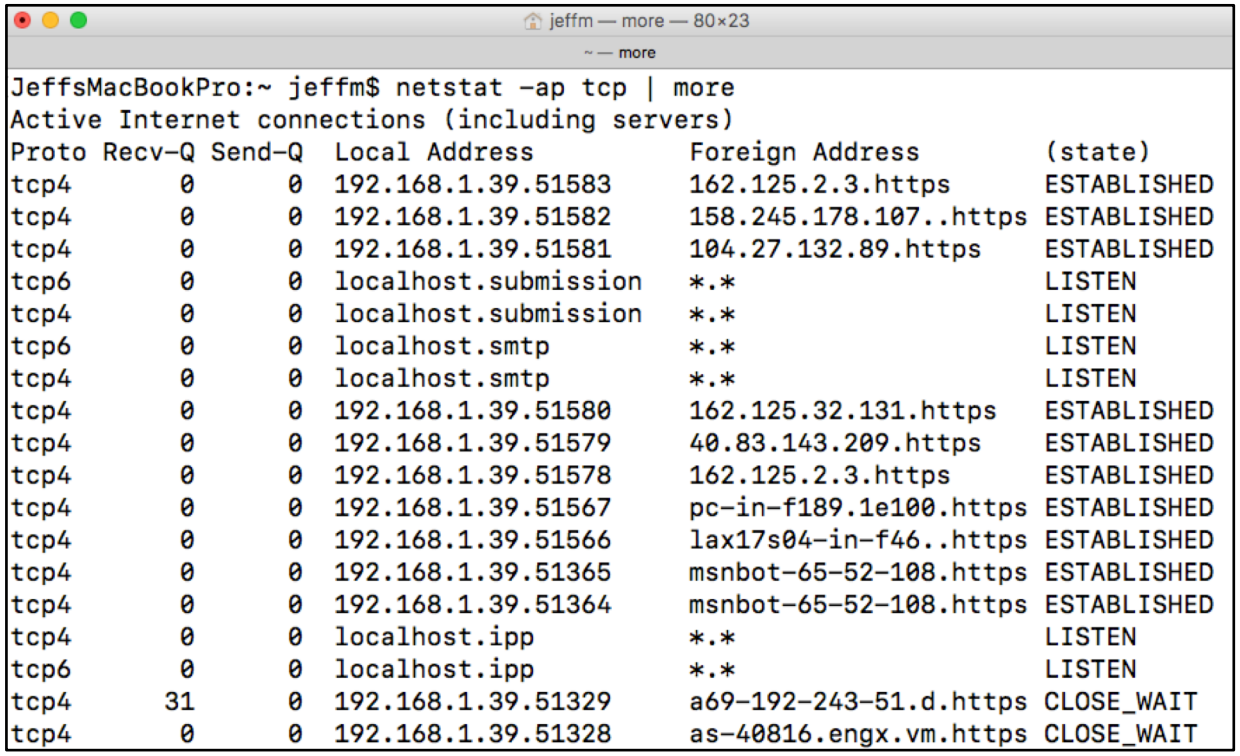
Example 02: Vagrantfile, IP Ports



netstat –ap tcp

on Windows

Example 02: Vagrantfile, IP Ports



netstat –ap tcp

on a Mac

Example 02: Vagrantfile, IP Ports

- **We can use any port on our machine that is NOT in the list**

- For this example, I will map the MongoDB 27017 and 27018 ports in the VM into host port 37017 and 37018
- NOTE: you may have to use different ports on your host machine

Example 02: Vagrantfile, IP Ports

```
# Create a forwarded port mapping which allows access to a specific port
 # within the machine from a port on the host machine and only allow access
 # via 127.0.0.1 to disable public access
 # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"
 config.vm.network "forwarded_port", guest: 27017, host: 37017, host_ip: "127.0.0.1"
 config.vm.network "forwarded_port", guest: 27018, host: 37018, host_ip: "127.0.0.1"
```

Example 02: Vagrantfile

- Now, lets test our Vagrant file to make sure it works:

Example 02: Vagrantfile

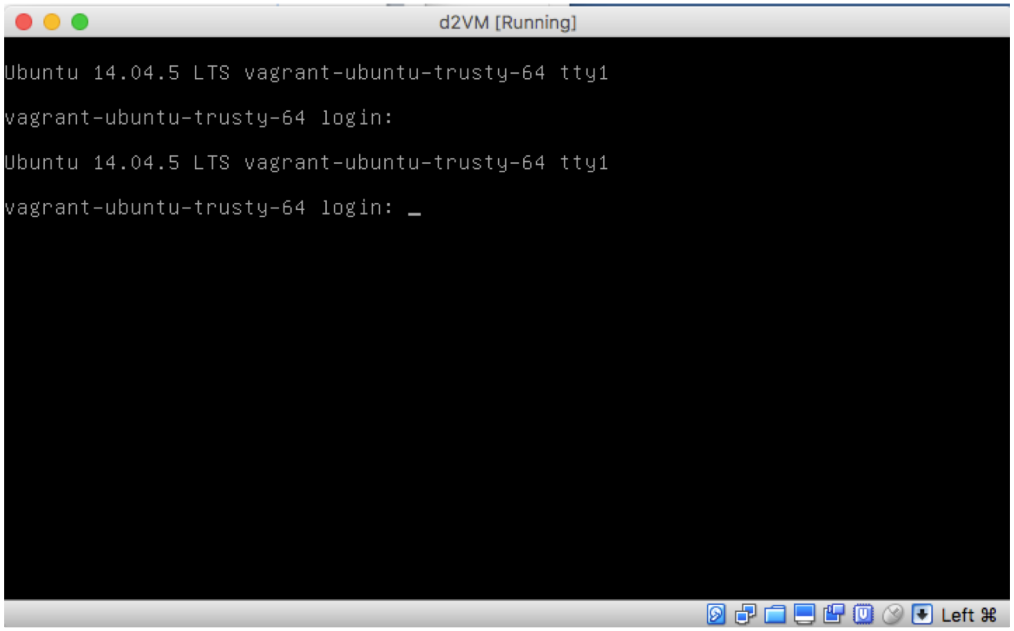
```
d2>vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/xenial64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/xenial64' is up to date...
==> default: A newer version of the box 'ubuntu/xenial64' for provider 'virtualbox' is
==> default: available! You currently have version '20170815.1.0'. The latest is version
==> default: '20190605.0.0'. Run `vagrant box update` to update.
==> default: Setting the name of the VM: d2VM-01
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 27017 (guest) => 37017 (host) (adapter 1)
    default: 27018 (guest) => 37018 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: ubuntu
    default: SSH auth method: password
```

Vagrant automatically downloads ubuntu/xenial64 if it cannot find it on the local machine

```
d2>vagrant port
The forwarded ports for the machine are listed below. Please note that
these values may differ from values configured in the Vagrantfile if the
provider supports automatic port collision detection and resolution.

 27017 (guest) => 37017 (host)
 27018 (guest) => 37018 (host)
    22 (guest) => 2222 (host)
```

Check to port mappings using: vagrant port

Example 02: Vagrantfile



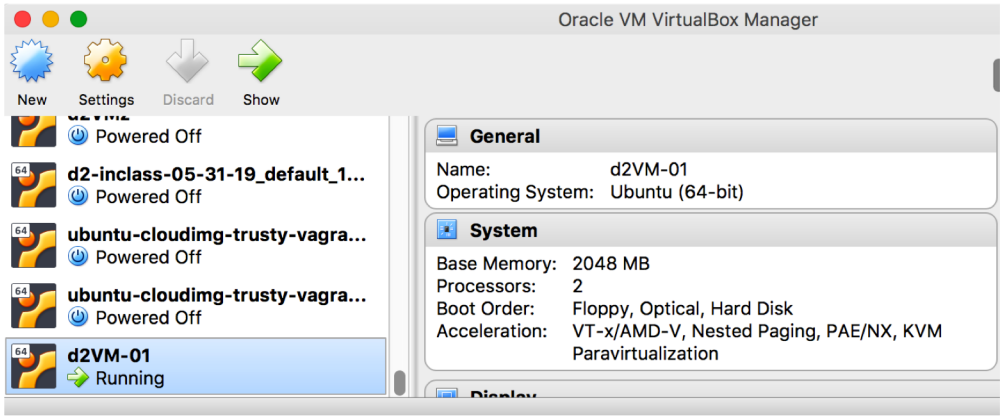
we set gui to true in the Vagrant file – so VirtualBox brought up a terminal to log into.

To log in using the VirtualBox provided terminal – the default user-name and password is:

vagrant
vagrant

However, the majority of the work done in DevOps will not bring up an interactive UI. Instead automated ssh access, which we will cover later on, is used.

Example 02: Vagrantfile



Here is the VirtualBox UI – showing our Vagrant provisioned VM

Example 02: Vagrantfile

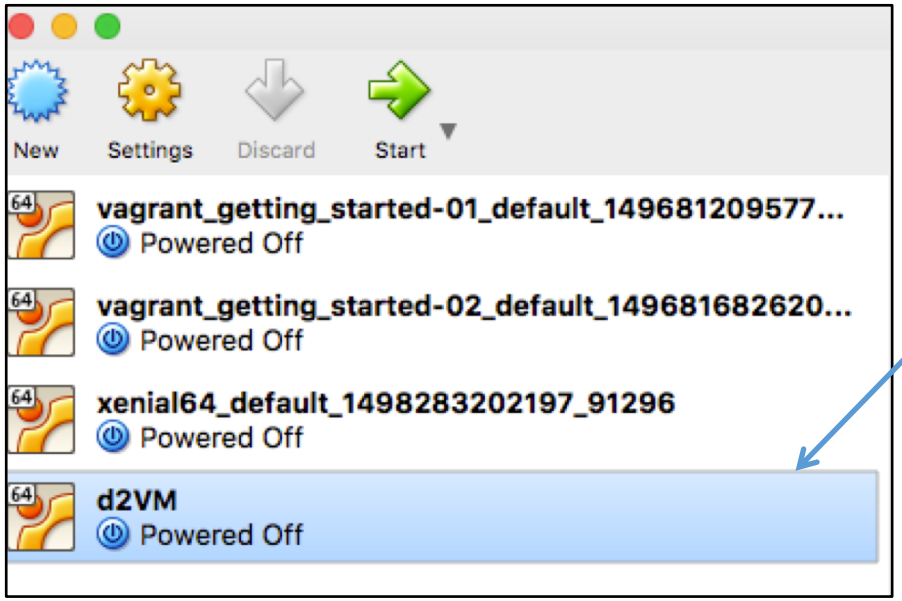
Let's halt the VM and add a Vagrant provisioning script to install Docker (ce) in the VM.

In a terminal, "cd" into the directory that has our Vagrant file, run vagrant halt

```
JeffsMacBookPro:d2 jeffm$ vagrant halt --help
Usage: vagrant halt [options] [name|id]

Options:

    -f, --force                      Force shut down (equivalent of pulling power)
    -h, --help                       Print this help
JeffsMacBookPro:d2 jeffm$
```

Example 02: Vagrantfile

```
JeffsMacBookPro:d2 jeffm$ vagrant halt
==> default: Attempting graceful shutdown of VM...
JeffsMacBookPro:d2 jeffm$
```



VirtualBox was running when "vagrant halt" was called.

Notice how the VirtualBox GUI "sees" that the VM is shutdown

Example 02: Provisioning Script to install Docker

- Next, we want to add a provisioning script that will install Docker into our Ubuntu running in VirtualBox

- The docker docs site can help us get the information we need

  https://docs.docker.com/install/linux/docker-ce/ubuntu/

  We will add two provisioning scripts:
  - One to add a docker repository into the list of repos apt/apt-get use
  - And another to install docker

Example 02: Provisioning Script to install Docker

- Docker CE support the Xenial 16.04 release, this is why we switched from trusty64 to xenial64

- The docker install page also mentions "storage drivers": overlay2, aufs, btrfs

  In brief, a Docker container has read-only portions that persist when the container is not running.

  You can also write data into Docker containers. For example when you run a script in Docker that downloads and install something (like Python), that "data" will not be persisted (saved) within the original Docker images (unless you create a new Docker image that includes what you installed).

  Docker page: https://docs.docker.com/storage/storagedriver/ - describes the major difference between images and containers a being is – an image has a top "layer" that is writable. Once a container is no longer running, the data written into the top layer no longer exists.

  Docker uses "storage drivers" to manage the contents of images and the writable part of a container.

Example 02: Provisioning Script to install Docker

- The docker "storage drivers": overlay2, aufs, btrfs continued (https://docs.docker.com/storage/):

  When something (a program, a service) writes data, by default, the data is written into the writable container layer – which mean the data is not persisted.

  Writing into the container writable layer requires a storage driver that manages the file system.

  The storage drivers implement some called a "union filesystem". Link:

  https://www.datalight.com/blog/2016/01/27/explaining-overlayfs-—-what-it-does-and-how-it-works/

  has a good definition of union file systems:

  Union file systems are a creative solution to allow a virtual merge of multiple folders, while keeping their actual contents separate.

Example 02: Provisioning Script to install Docker

- The docker "storage drivers": overlay2, aufs, btrfs continued (https://docs.docker.com/storage/):

  Union file systems operator in layers. Typically, only the top layer is writeable.

  When a union file system is mounted (i.e. the file system within a Docker container), all of the file systems appear to have the same directory structure.

- Overlay2 is the recommended (and default) storage driver for Docker.

Example 02: Provisioning Script to install Docker

- Side point: so if data written into to top-layer of a Docker container is NOT saved, what do we need to do to save the contents of file operations when a container is halted?

  Volumes and **bind** mounts (https://docs.docker.com/storage/).

  Volumes are stored within the host's file system and managed by Docker in folder /var/lib/docker/volumes).

  Bind mounts can be anywhere within the host system.

Example 02: Provisioning Script to install Docker

- Now let's consider how we can persist data when a Docker container running in a VM (via Vagrant) hosts Docker.
  - If the programs we run in the Docker container just write normally into the (union mount/overlay2) file system, the data will not be saved when the container stops running.
  - We can persist our data created within the Docker container by writing into a Docker Volume (/var/lib/docker/volumes) or a bind mount into another local in the virtual machine.
  - But what happens when the VM that hosted the Docker image terminates?

    By default, that data will not be saved, unless it was written into:
    - A persistent file system else where on the network
    - Or, a folder shared between the VM and the host OS

Example 02: Provisioning Script to install Docker

- One more note: The contents of: https://docs.docker.com/install/linux/docker-ce/ubuntu/

  states that the modules listed below are needed to install Docker in Ubuntu. This was not needed by your instructor to install docker-ce (community edition) within Ubuntu xenial64 (16.04):

  - docker-ce-cli (command line interface)
  - containerd.io

    used by Docker to abstract calls into the underlying host OS

```
ubuntu@ubuntu-xenial:~$ grep ^[^#] /etc/apt/sources.list /etc/apt/sources.list.d/*
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial main restricted
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial main restricted
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates main restricted
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates main restricted
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial universe
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial universe
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates universe
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates universe
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial multiverse
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates multiverse
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-backports main restricted univer
se
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-backports main restricted un
iverse
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security main restricted
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security main restricted
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security universe
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security universe
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security multiverse
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security multiverse
/etc/apt/sources.list:deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
```

Use vagrant ssh to get into the running VM. Notice there are no sites with "docker"

```
ubuntu@ubuntu-xenial:~$ grep ^[^#] /etc/apt/sources.list /etc/apt/sources.list.d/*
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial main restricted
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial main restricted
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates main restricted
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates main restricted
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial universe
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial universe
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates universe
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates universe
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial multiverse
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates multiverse
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-backports main restricted universe multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-backports main restricted universe multiverse
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security main restricted
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security main restricted
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security universe
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security universe
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security multiverse
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security multiverse
grep: /etc/apt/sources.list.d/*: No such file or directory
```

Example 02: Provisioning Script to install Docker

Provisioning script - **setup-docker-repo** – from Vagrantfile

```
# from: https://docs.docker.com/install/linux/docker-ce/ubuntu/
 config.vm.provision "setup-docker-repo",type:"shell", inline: <<-SHELL
   apt-get update

   sudo apt-get install \
            apt-transport-https \
            ca-certificates \
            curl \
            gnupg-agent \
            software-properties-common

   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
            sudo apt-key fingerprint 0EBFCD88

            sudo add-apt-repository \
                     "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
 SHELL
```

From the host system run: **vagrant provision --provision-with setup-docker-repo**

```
d2>vagrant provision --provision-with setup-docker-repo
==> default: Running provisioner: setup-docker-repo (shell)...
    default: Running: inline script
    default: Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
    default: Get:2 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
```

...

```
    default: Reading state information...
    default: The following additional packages will be installed:
    default:   libassuan0 libcurl3-gnutls libnpth0 pinentry-curses
    default:   python3-software-properties
    default: Suggested packages:
    default:   pinentry-doc
    default: The following NEW packages will be installed:
    default:   gnupg-agent libassuan0 libnpth0 pinentry-curses
    default: The following packages will be upgraded:
    default:   apt-transport-https ca-certificates curl libcurl3-gnutls
    default:   python3-software-properties software-properties-common
    default: 6 upgraded, 4 newly installed, 0 to remove and 205 not upgraded.
    default: Need to get 860 kB of archives.
    default: After this operation, 1,145 kB of additional disk space will be used.
    default: Do you want to continue? [Y/n]
    default: Abort.
    default: OK
    default: pub   4096R/0EBFCD88 2017-02-22
    default:       Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
    default: uid                  Docker Release (CE deb) <docker@docker.com>
    default: sub   4096R/F273FCD8 2017-02-22
```

Now, back in the VM, lets list the repos used by apt/apt-get again:

```
ubuntu@ubuntu-xenial:~$ grep ^[^#] /etc/apt/sources.list /etc/apt/sources.list.d/*
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial main restricted
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial main restricted
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates main restricted
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates main restricted
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial universe
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial universe
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates universe
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates universe
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial multiverse
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-updates multiverse
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-updates multiverse
/etc/apt/sources.list:deb http://archive.ubuntu.com/ubuntu xenial-backports main restricted unive
se
/etc/apt/sources.list:deb-src http://archive.ubuntu.com/ubuntu xenial-backports main restricted u
iverse
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security main restricted
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security main restricted
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security universe
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security universe
/etc/apt/sources.list:deb http://security.ubuntu.com/ubuntu xenial-security multiverse
/etc/apt/sources.list:deb-src http://security.ubuntu.com/ubuntu xenial-security multiverse
/etc/apt/sources.list:deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
```

Example 02: Provisioning Script to install Docker

Now, let's look at our second Vagrant provisioning script: "install-docker":

```
config.vm.provision "install-docker",type:"shell", inline: <<-SHELL
  apt-get update
  #did not need this, docker-cd-cli, and containerd.io are installed with docker-ce
  #sudo apt-get install docker-ce docker-ce-cli containerd.io
   sudo apt-get -y install docker-ce
  SHELL
```

From the host system run: **vagrant provision --provision-with install-docker**

```
d2>vagrant provision --provision-with install-docker
==> default: Running provisioner: install-docker (shell)...
    default: Running: inline script
    default: Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
    default: Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
    default: Get:3 https://download.docker.com/linux/ubuntu xenial InRelease [66.2 kB]
    default: Hit:4 http://archive.ubuntu.com/ubuntu xenial-updates InRelease
    default: Get:5 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages [8,482 B]
```

...

```
    default: The following additional packages will be installed:
    default:   aufs-tools cgroupfs-mount containerd.io docker-ce-cli libltdl7 libseccomp2
```

...

```
    default: Setting up aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
    default: Setting up cgroupfs-mount (1.2) ...
    default: Setting up containerd.io (1.2.5-1) ...
    default: Setting up docker-ce-cli (5:18.09.6~3-0~ubuntu-xenial) ...
    default: Setting up docker-ce (5:18.09.6~3-0~ubuntu-xenial) ...
```

Example 02: Running Docker in the VM

- Let's use dpkg (Debian's native package manager) to see if docker is installed:

```
dpkg(1)                                    dpkg suite                                    dpkg(1)

NAME
       dpkg - package manager for Debian

SYNOPSIS
       dpkg [option...] action

WARNING
       This manual is intended for users wishing to understand dpkg's command line options and package states in more
       detail than that provided by dpkg --help.
```

Example 02: Running Docker in the VM

- The shell below shows the output of:
  - dpkg --list | grep docker
  - ps -e | grep docker

```
ubuntu@ubuntu-xenial:~$ dpkg --list | grep docker
ii  docker-ce                          5:18.09.6~3-0~ubuntu-xenial
ii  docker-ce-cli                      5:18.09.6~3-0~ubuntu-xenial
ubuntu@ubuntu-xenial:~$ ps -e | grep docker
13769 ?        00:00:01 dockerd
```

Now, "vagrant ssh" back into the VM:

```
ubuntu@ubuntu-xenial:~$ docker --version
Docker version 18.09.6, build 481bc77
ubuntu@ubuntu-xenial:~$ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker
ar%2Frun%2Fdocker.sock/v1.39/containers/create: dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
ubuntu@ubuntu-xenial:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:0e11c388b664df8a27a901dce21eb89f11d8292f7fca1b3e3c4321bf7897bffe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Notice, we still need to run docker using "sudo"

See next slide …

Site: https://docs.docker.com/install/linux/linux-postinstall/ documents post install steps for Docker.

By default, the Docker daemon always runs a root user. This means all docker commands you run that access the Docker daemon must used "sudo". To get around this you can do the following:

see if the docker group is present in the VM hosting Docker:

```
ubuntu@ubuntu-xenial:~$ cat /etc/group | grep docker
docker:x:999:
ubuntu@ubuntu-xenial:~$
```

If the docker group is not present create it:
sudo groupadd docker

Add the current user you are logged into the VM into the Docker group:
sudo usermod -a -G docker $USER

Log out of the VM and log back in using vagrant ssh:

```
ubuntu@ubuntu-xenial:~$ cat /etc/group | grep docker
docker:x:999:ubuntu
ubuntu@ubuntu-xenial:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Example 02: Updates after revising Vagrant file

- Recall from previous sections:

  Provisioners in Vagrant allow you to automatically install software, alter configurations, and more on the machine as part of the vagrant up process.

- Some of the built-in provisioners are:
  - **File** – allows you to upload a file into the VM
  - **Shell** – execute Bash scripts in the VM
  - **Several automation and deployments tools** including: Ansible, CFEngine, Chef, Puppet, Salt, and **Docker** (time permitting we will cover Docker Vagrant provisioning later in the course).

Summary

- Wow – we have done a lot!
- The following slides will provide a summary of what we have covered in this section

Summary

- Review Virtual Machines, type 1, type 2, full and paravirtualization
- discussed how containers provide a lighter weight deployment unit (when compared to VMs)
- installing docker on your physical machine
- we ran docker hello-world on our physical machine
- installed Docker on a VM - nested virtualization

Example 02 Summary

- Vagrant Registry
- Vagrantfile basics
- our Vagrantfile for Example-02:
  - named the VM,
  - set memory for VM,
  - mapped VM port to host machine ports, used netstat to find open ports
- Ran our first test with the Vagrant file to get ubuntu/xenial64

Example 02 Summary

- The default login for Vagrant VM is: vagrant , vagrant
- vagrant halt
- wrote a Vagrant provisioning script to get and install docker in the VM
- vagrant up --provision , vagrant up --provision-with shell
- vagrant reload --provision
- vagrant reload --provision-with shell
- vagrant ssh
- Apt, apt-get, dpkg, ps -e, grep
- ran docker hello-world in our vagrant provisioned VM

DevOp's is all about **Automation**

- While we did perform some manual steps – please keep in mind –

  DevOps is all about automating the configuration and deployment of resources like VM's , Docker, and the applications/servers/DBs/etc., that run in VMs and Containers

  .
- A good methodology is to:
  - do manual setup/configuration from the command line until you have your deployment in place
  - followed by putting your setup/configuration into Bash scripts
  - Using automated ssh by writing code (e.g. in Bash , Python)
  - Using automated deployment tools like Terraform, Ansible