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/

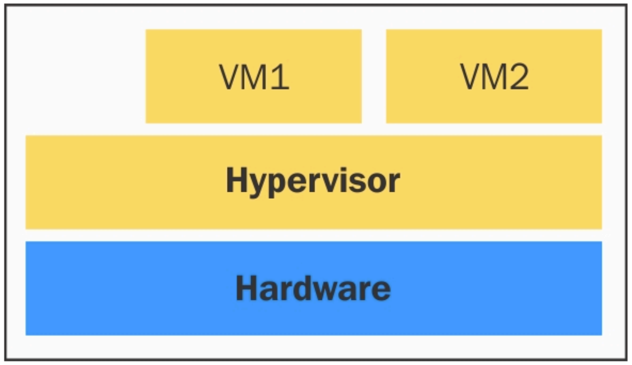
https://www.vagrantup.com/docs/virtualbox/configuration.html

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

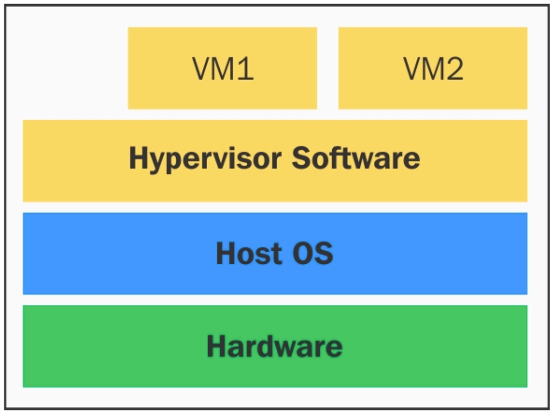
Virtual Machine Review

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

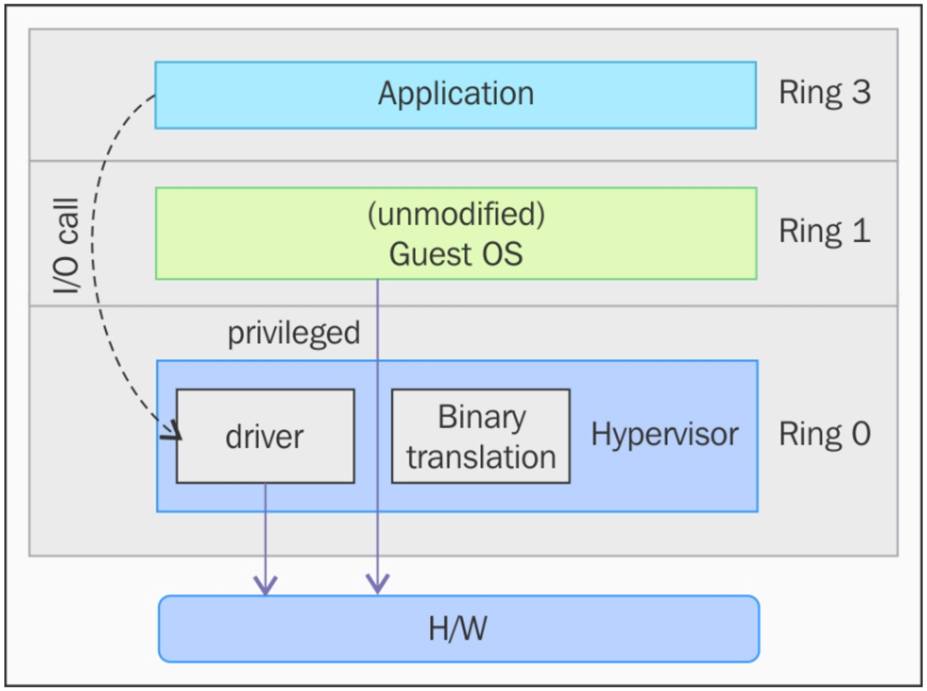
Virtual Machine Review

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

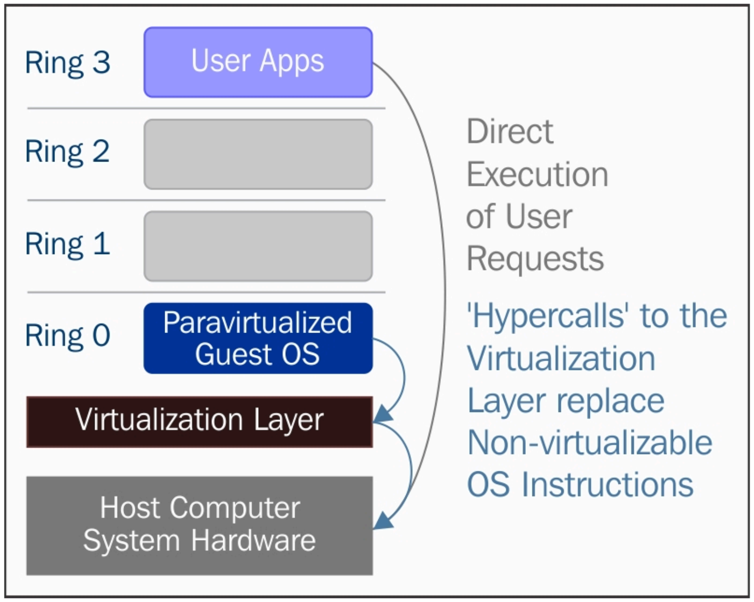
Full virtualization

- 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.


Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Paravirtualized Guest OS
Virtualization Layer
Host Computer System Hardware
Direct Execution of User Requests
'Hypercalls' to the Virtualization Layer replace Non-virtualizable OS Instructions

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

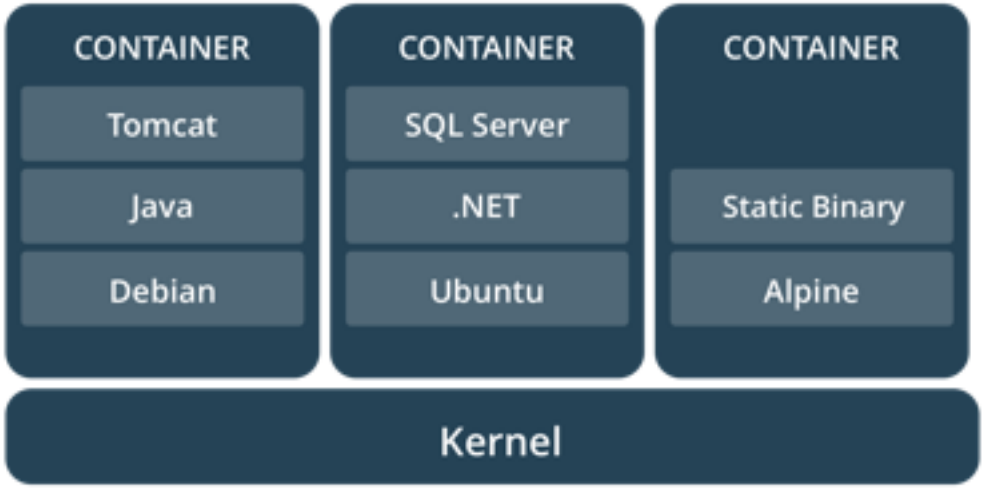
A container does not need a guest OS.

Containers

Basically − a container only contains what it needs to run the applications embedded within it. This typically includes:

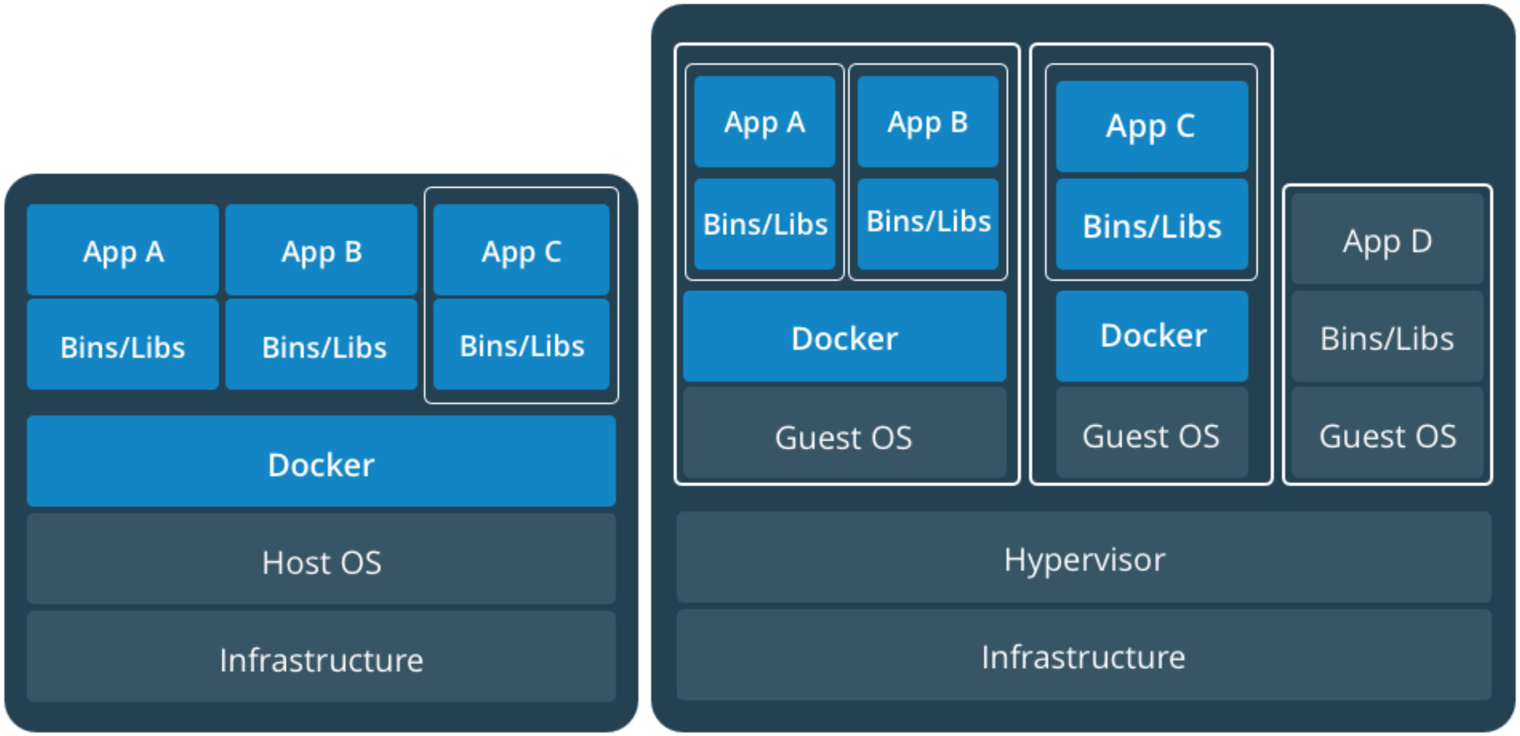
- the applications
- libraries and framework used by the applications
- unlike a VM deployment, a container does NOT need an (guest) OS

Containers



n https://www.docker.com/what-container - each container contains the software it needs to run while zing 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 a future lecture

Docker

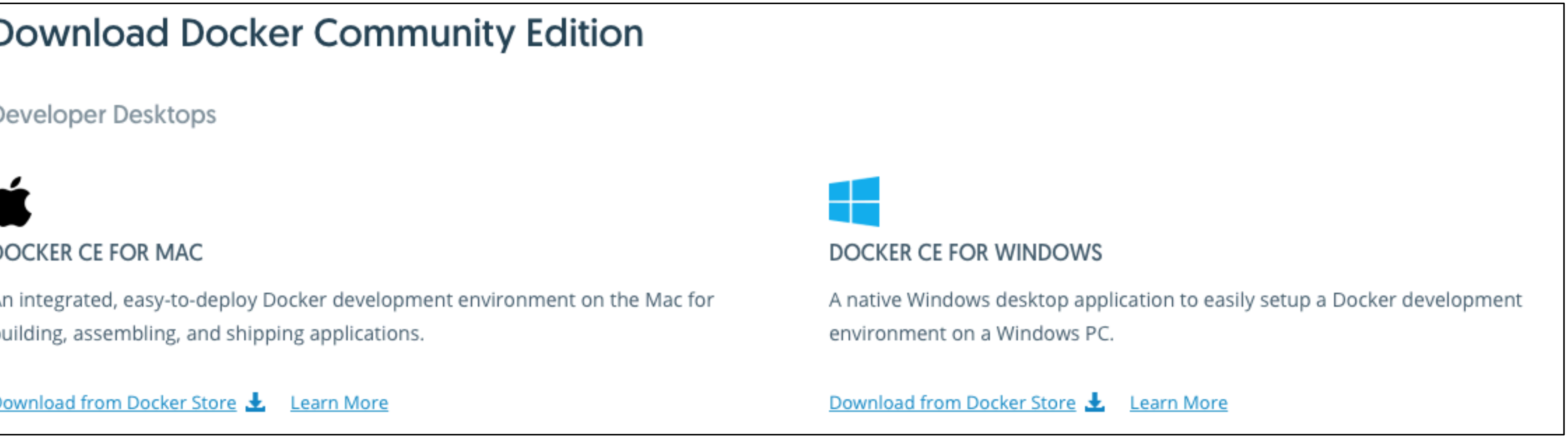
Let's get starting leaning Docker by running looking at some simple examples.

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



Download Docker Community Edition

Developer Desktops

DOCKER CE FOR MAC

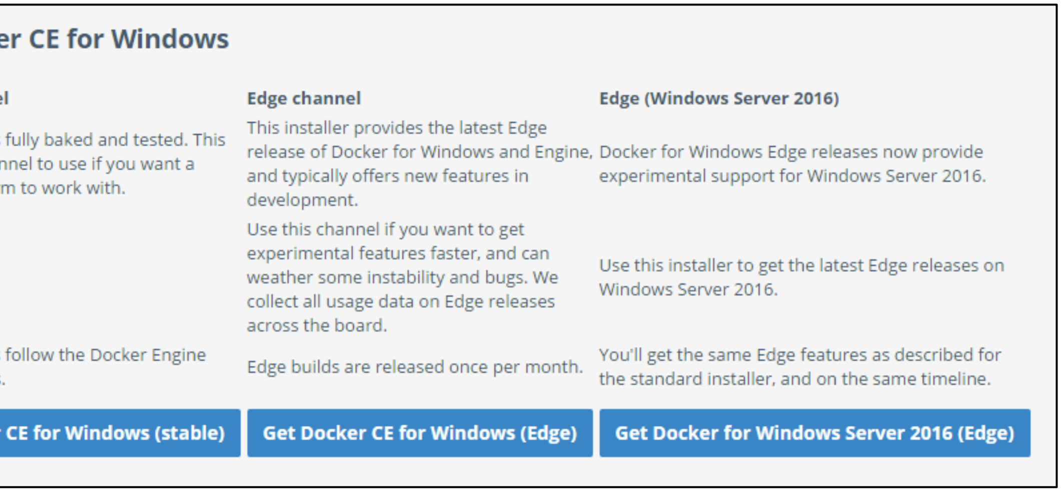
An integrated, easy-to-deploy Docker development environment on the Mac for building, assembling, and shipping applications.

DOCKER CE FOR WINDOWS

A native Windows desktop application to easily setup a Docker development environment on a Windows PC.

Docker also provides version for several Linux distributions and AWS

Docker



er CE for Windows

el

fully baked and tested. This nnel to use if you want a m to work with.

follow the Docker Engine .

Edge channel

This installer provides the latest Edge release of Docker for Windows and Engine, and typically offers new features in development.

Use this channel if you want to get experimental features faster, and can weather some instability and bugs. We collect all usage data on Edge releases across the board.

Edge builds are released once per month.

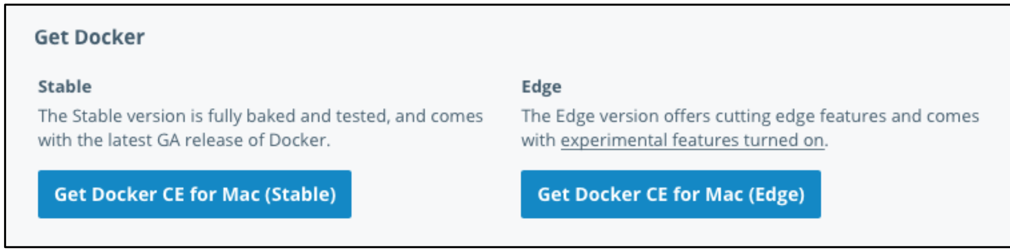
Edge (Windows Server 2016)

Docker for Windows Edge releases now provide experimental support for Windows Server 2016.

Use this installer to get the latest Edge releases on Windows Server 2016.

You'll get the same Edge features as described for the standard installer, and on the same timeline.

CE for Windows (stable)
Get Docker CE for Windows (Edge)
Get Docker for Windows Server 2016 (Edge)



Get Docker

Stable

The Stable version is fully baked and tested, and comes with the latest GA release of Docker.

Edge

The Edge version offers cutting edge features and comes with experimental features turned on.

Get Docker CE for Mac (Stable)
Get Docker CE for Mac (Edge)

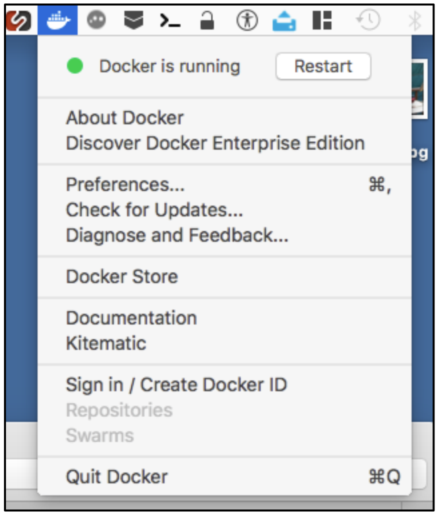
Download the "Stable Channel" version of Docker for your platform. Following the instructions on the site to install Docker

Docker

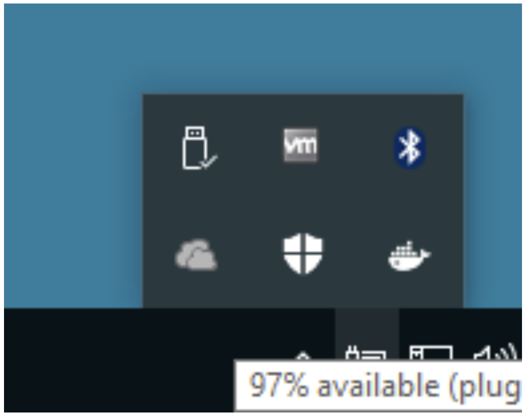
Docker runs as a background service (you may have to start it manually).

When Docker is running you will see the Docker Whale:

Mac: access Docker from the menu bar at the top



Docker in Windows 10



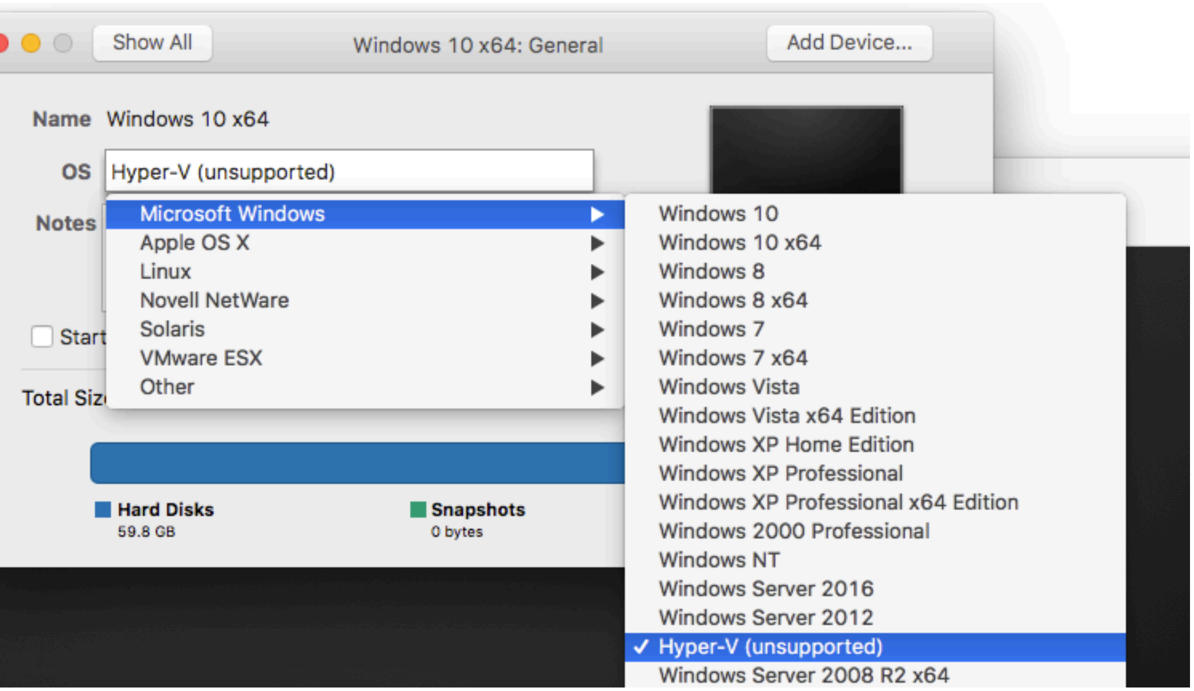
Installing Docker in a Virtual Machine

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



Windows 10 x64: General

Name Windows 10 x64

OS Hyper-V (unsupported)

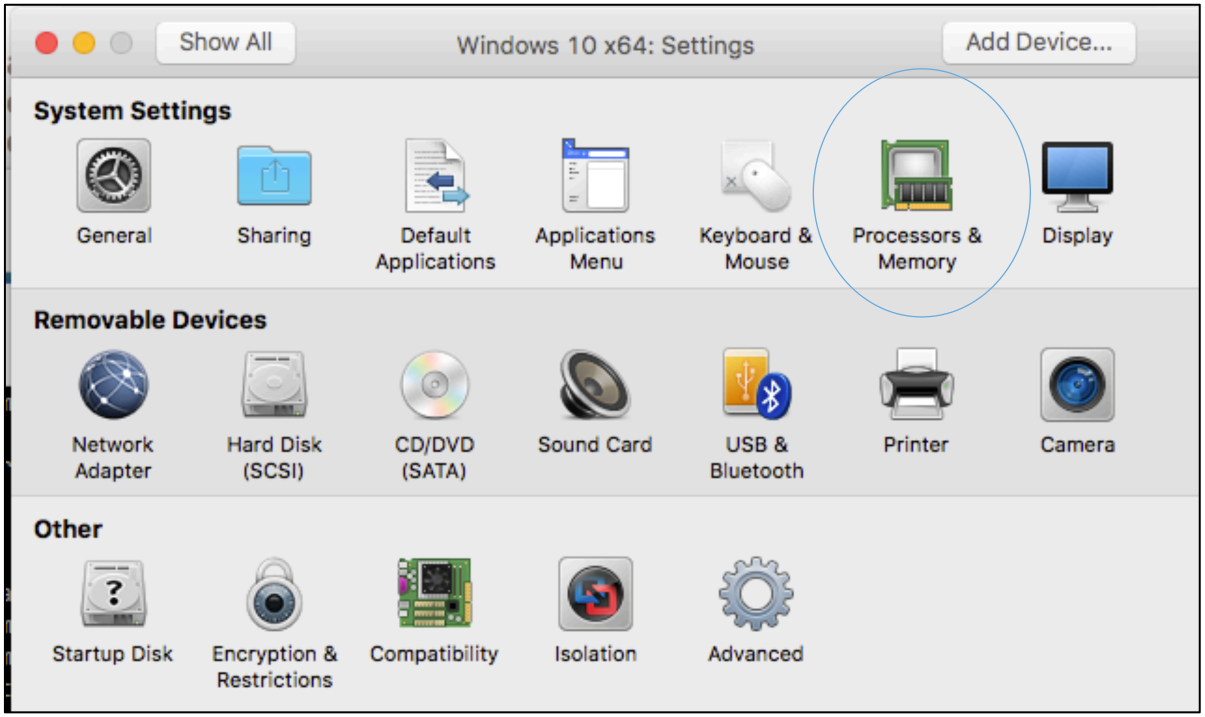
Hard Disks 59.8 GB

Snapshots 0 bytes

For VMWare Fusion;

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

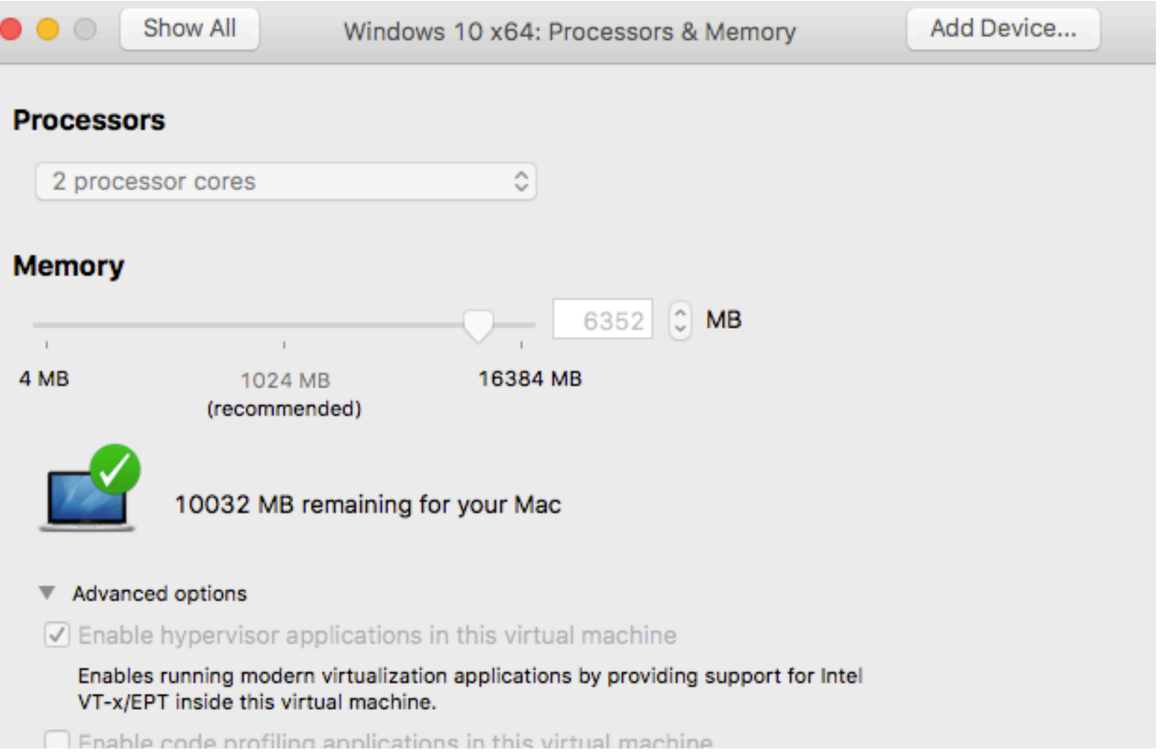
Installing Docker in a Virtual Machine VMWare Fusion



Next;

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

Installing Docker in a Virtual Machine
VMWare Fusion



Processors

2 processor cores

Memory

6352 MB

10032 MB remaining for your Mac

▼ Advanced options

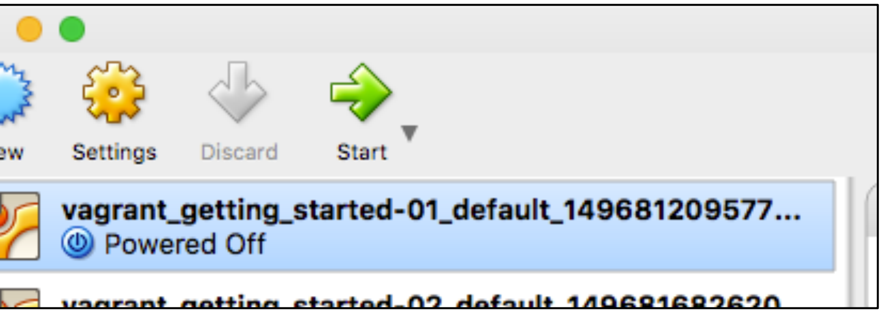
☑ Enable hypervisor applications in this virtual machine

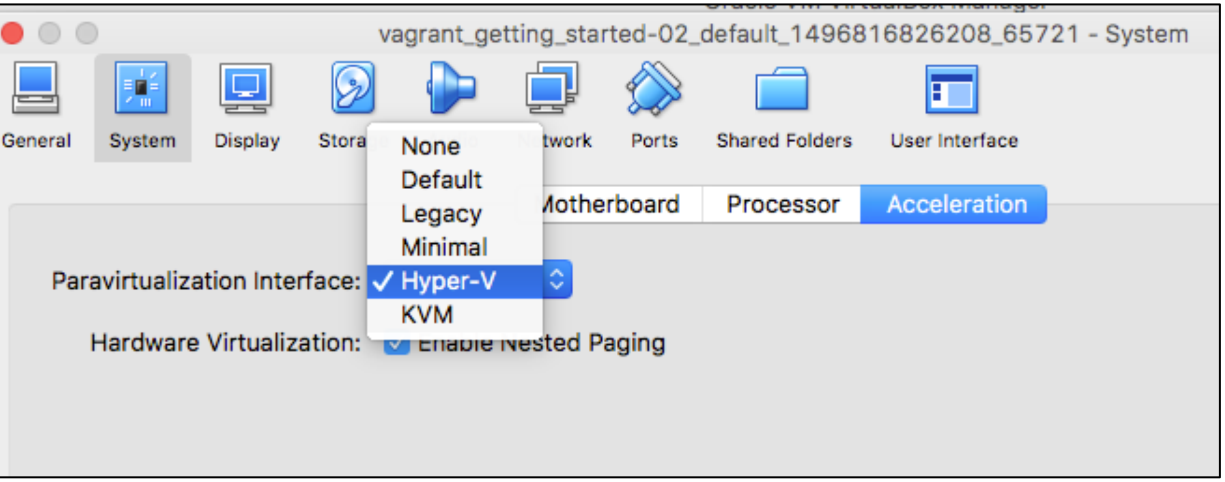
Enables running modern virtualization applications by providing support for Intel VT-x/EPT inside this virtual machine.

☐ Enable code profiling applications in this virtual machine

Under Advanced, check Enable Hypervisor application in this virtual machine

Installing Docker in a Virtual Machine VirtualBox


ew
Settings
Discard
Start
vagrant_getting_started-01_default_149681209577...
Powered Off
vagrant_getting_started-02_default_149681682620



Paravirtualization Interface: ✓ Hyper-V

Hardware Virtualization: ☑ Enable Nested Paging

To enable Hyper-V when using VirutalBo

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

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

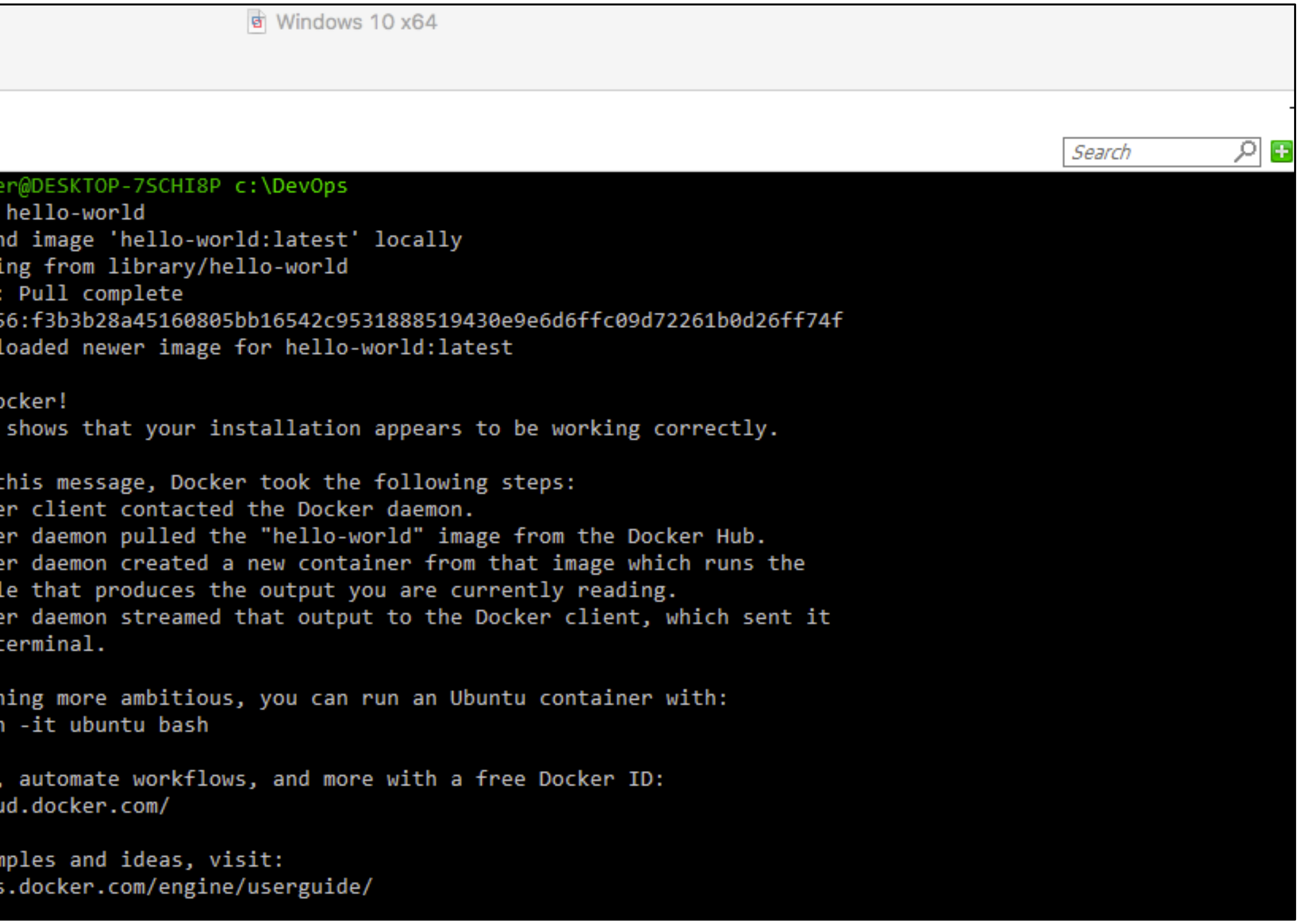
Docker hello-world Example

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

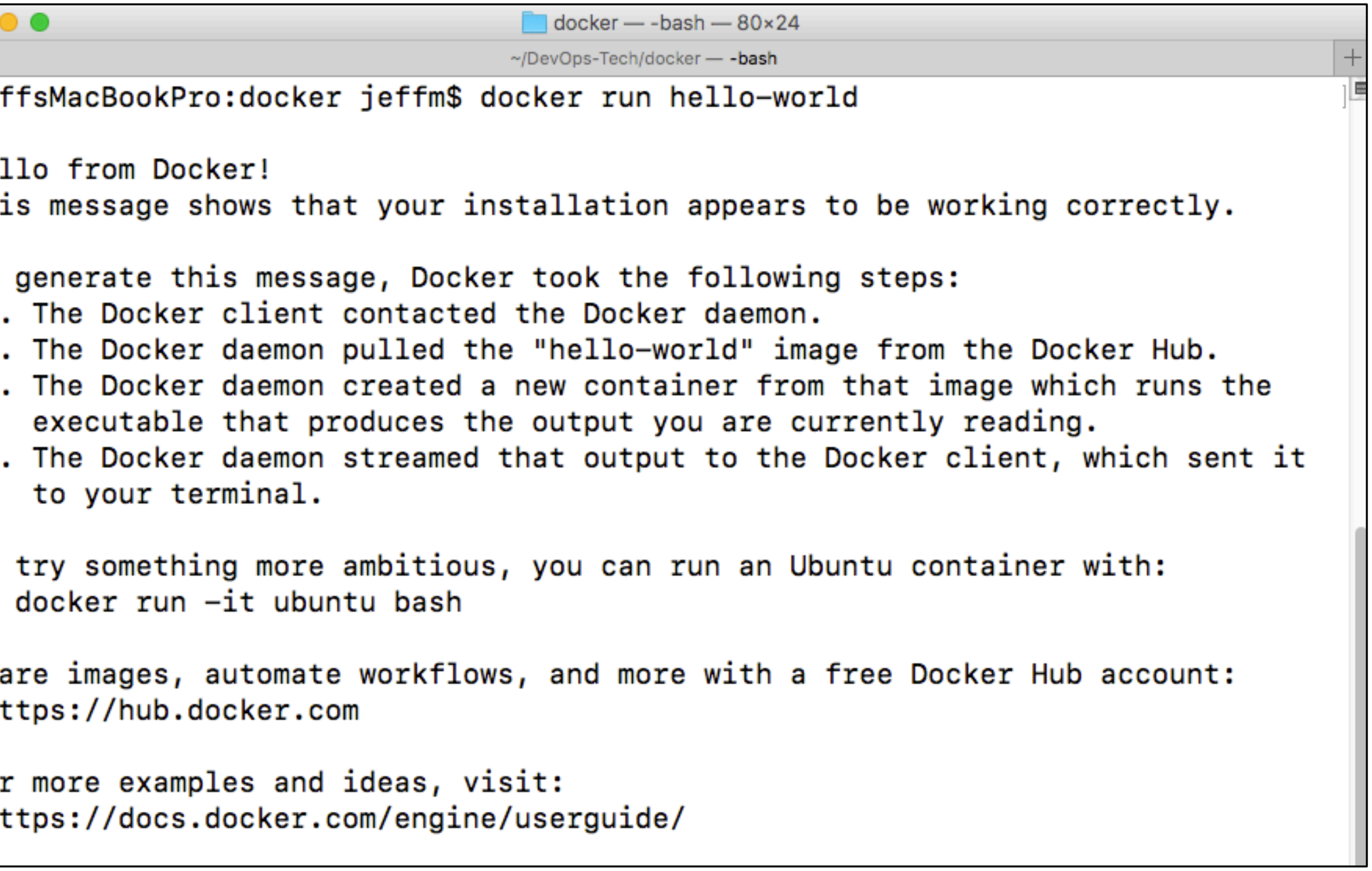


```
er@DESKTOP-7SCHI8P c:\DevOps
 hello-world
nd image 'hello-world:latest' locally
ing from library/hello-world
: Pull complete
56:f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f
loaded newer image for hello-world:latest

ocker!
 shows that your installation appears to be working correctly.

this message, Docker took the following steps:
er client contacted the Docker daemon.
er daemon pulled the "hello-world" image from the Docker Hub.
er daemon created a new container from that image which runs the
le that produces the output you are currently reading.
er daemon streamed that output to the Docker client, which sent it
terminal.

hing more ambitious, you can run an Ubuntu container with:
n -it ubuntu bash

, automate workflows, and more with a free Docker ID:
ud.docker.com/

mples and ideas, visit:
s.docker.com/engine/userguide/
```

In a command prompt:

docker run hello-world

Docker hello-world Example
Mac: Sierra



```
ffsMacBookPro:docker jeffm$ docker run hello-world

llo from Docker!
is message shows that your installation appears to be working correctly.

 generate this message, Docker took the following steps:
. The Docker client contacted the Docker daemon.
. The Docker daemon pulled the "hello-world" image from the Docker Hub.
. The Docker daemon created a new container from that image which runs the
  executable that produces the output you are currently reading.
. The Docker daemon streamed that output to the Docker client, which sent it
  to your terminal.

 try something more ambitious, you can run an Ubuntu container with:
 docker run -it ubuntu bash

are images, automate workflows, and more with a free Docker Hub account:
ttps://hub.docker.com

r more examples and ideas, visit:
ttps://docs.docker.com/engine/userguide/
```

In a terminal:

docker run hello-world

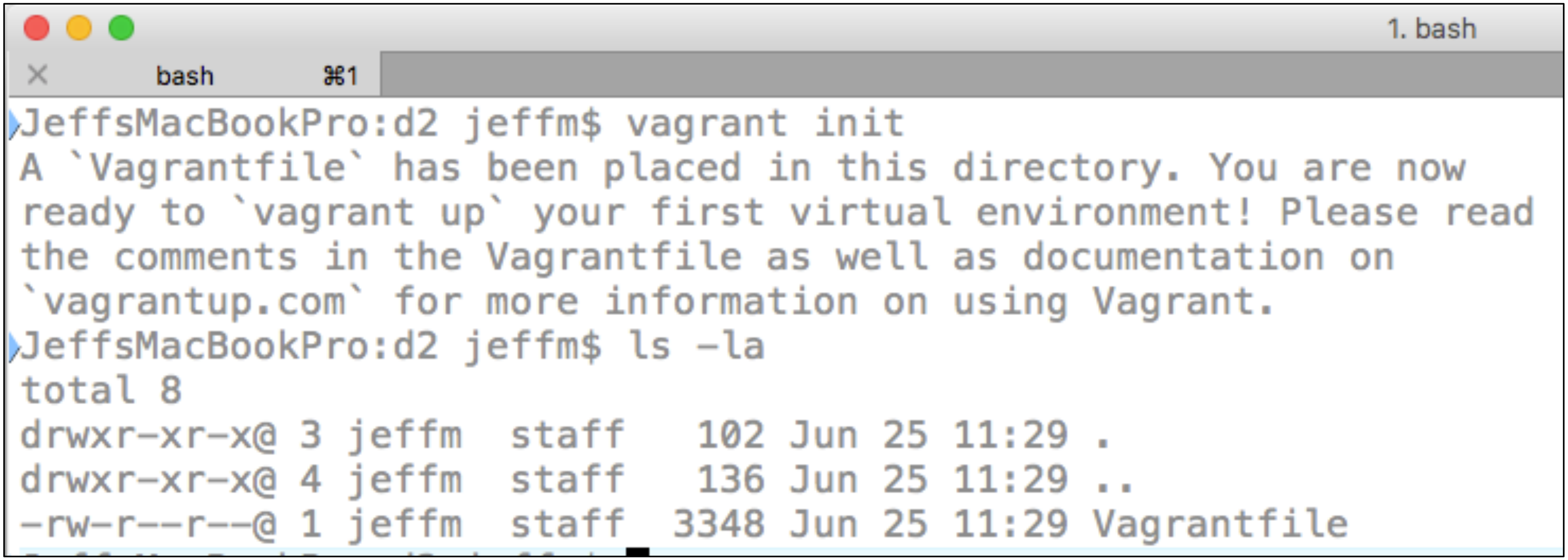
Example 02

In the next example we will:

- (again) look at the Vagrant Registery
- 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**



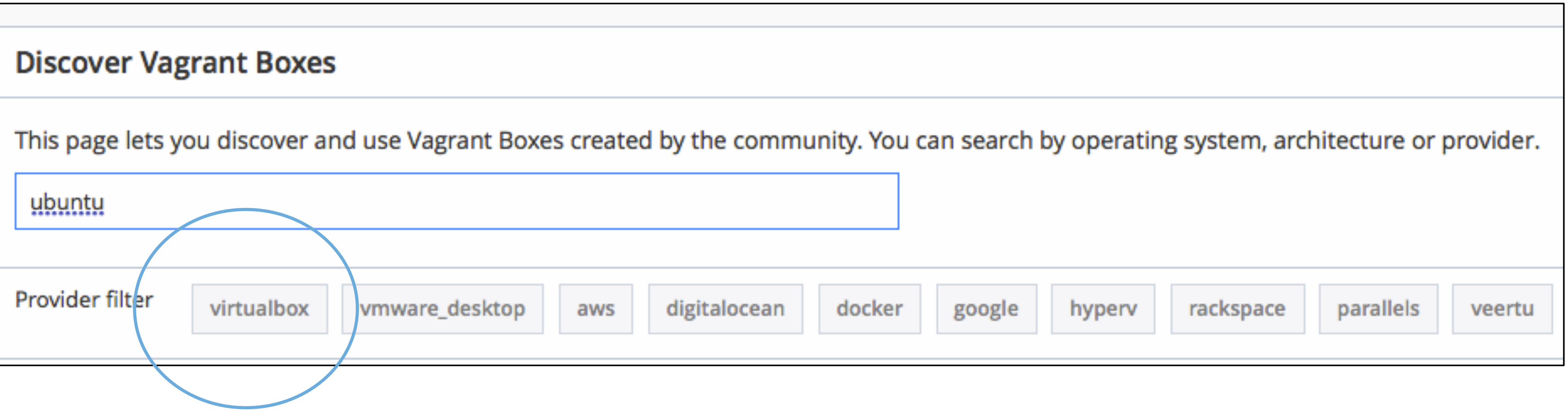
```
JeffsMacBookPro:d2 jeffm$ vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
JeffsMacBookPro:d2 jeffm$ ls -la
total 8
drwxr-xr-x@ 3 jeffm  staff   102 Jun 25 11:29 .
drwxr-xr-x@ 4 jeffm  staff   136 Jun 25 11:29 ..
-rw-r--r--@ 1 jeffm  staff  3348 Jun 25 11:29 Vagrantfile
```

Example 02: Vagrant Registry

Got to Vagrant Cloud (which changed urls on June 27,2017 to):

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

Example 02: Vagrant Registry

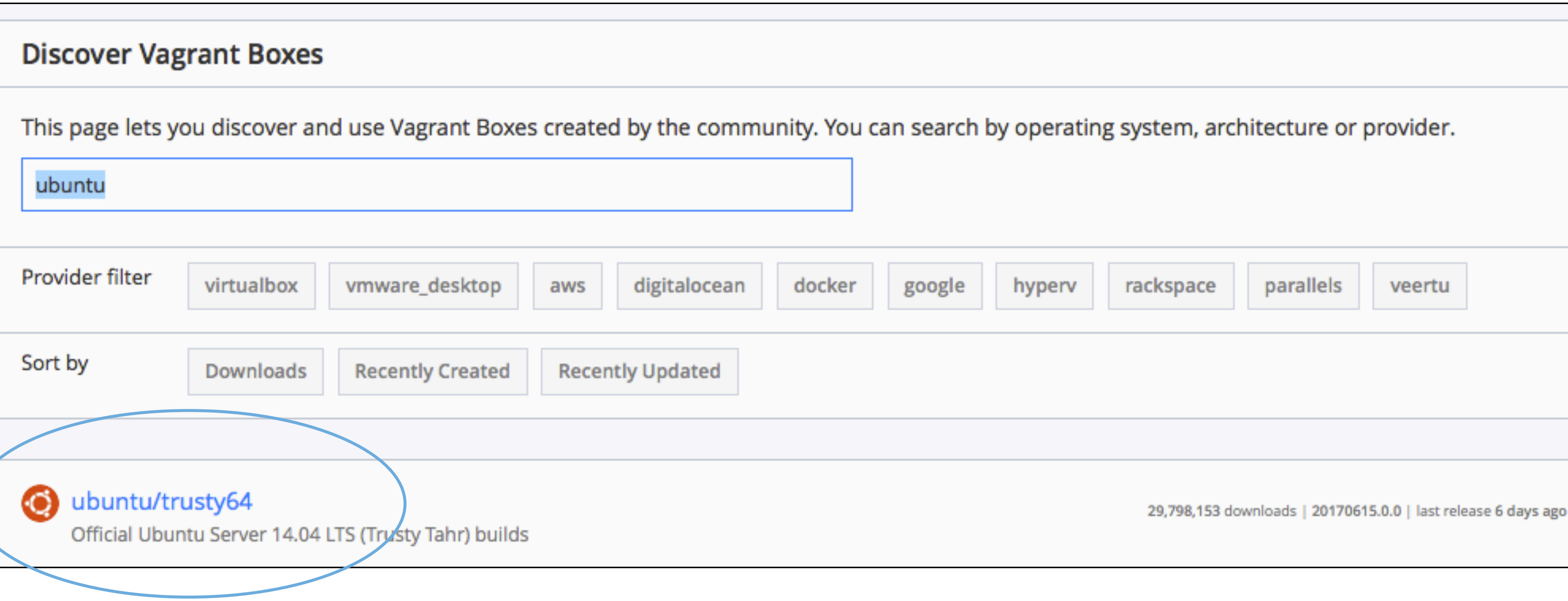


Discover Vagrant Boxes

This page lets you discover and use Vagrant Boxes created by the community. You can search by operating system, architecture or provider.

ubuntu

Example 02: Vagrant Registry



Discover Vagrant Boxes

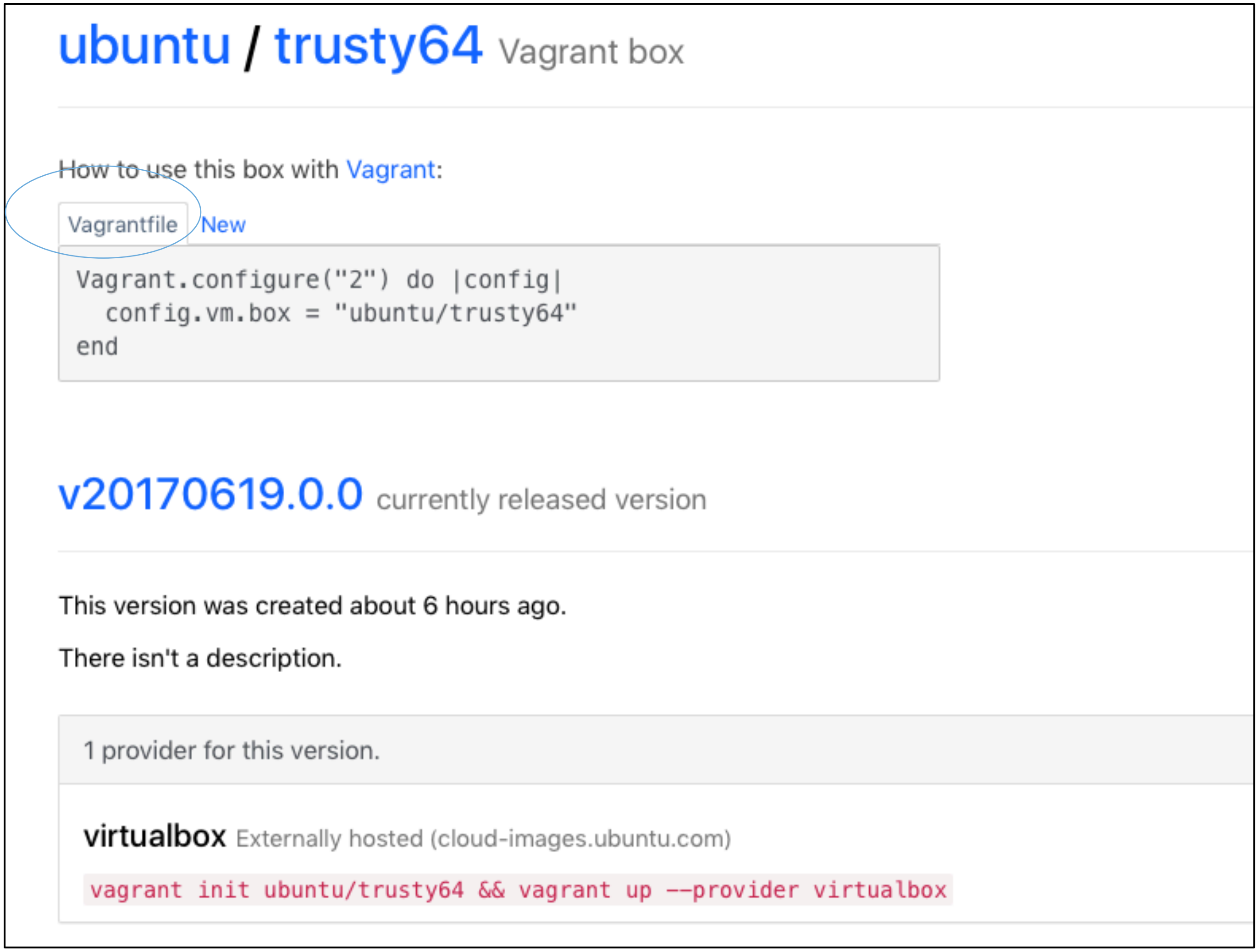
This page lets you discover and use Vagrant Boxes created by the community. You can search by operating system, architecture or provider.

ubuntu

ubuntu/trusty64
Official Ubuntu Server 14.04 LTS (Trusty Tahr) builds

29,798,153 downloads | 20170615.0.0 | last release 6 days ago

Example 02: Vagrant Registry



ubuntu / trusty64 Vagrant box

How to use this box with Vagrant:

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/trusty64"
end
```

v20170619.0.0 currently released version

This version was created about 6 hours ago.

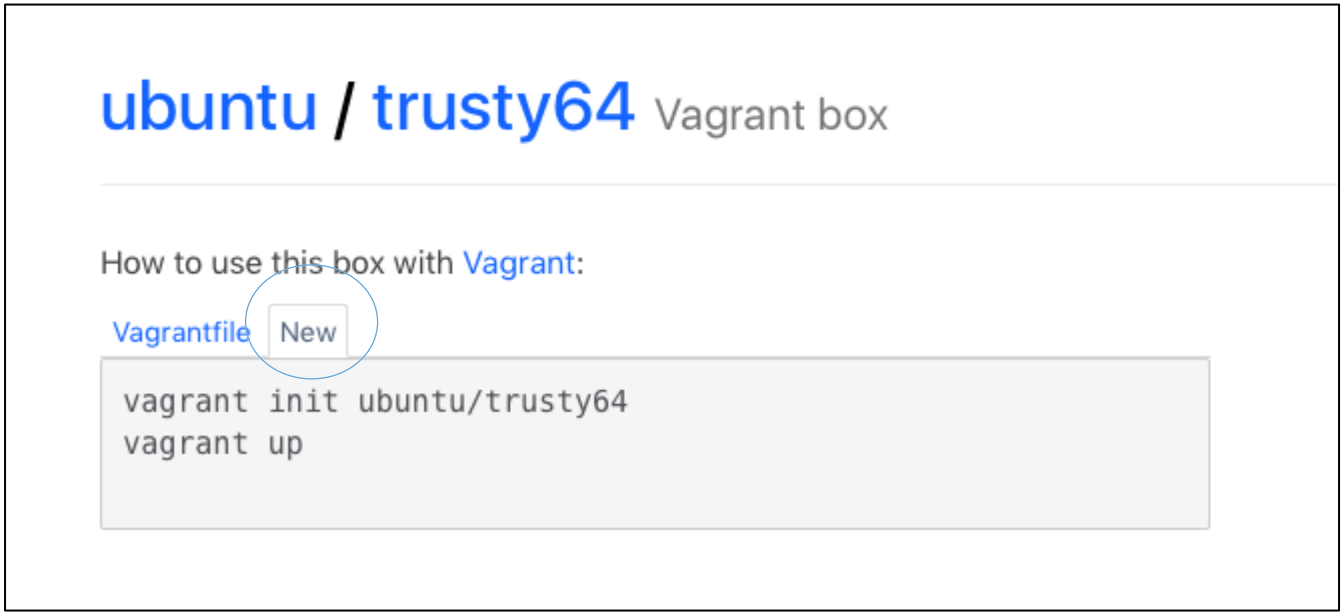
There isn't a description.

1 provider for this version.

virtualbox Externally hosted (cloud-images.ubuntu.com)

```
vagrant init ubuntu/trusty64 && vagrant up --provider virtualbox
```

Example 02: Vagrant Registry



ubuntu / trusty64 Vagrant box

How to use this box with Vagrant:

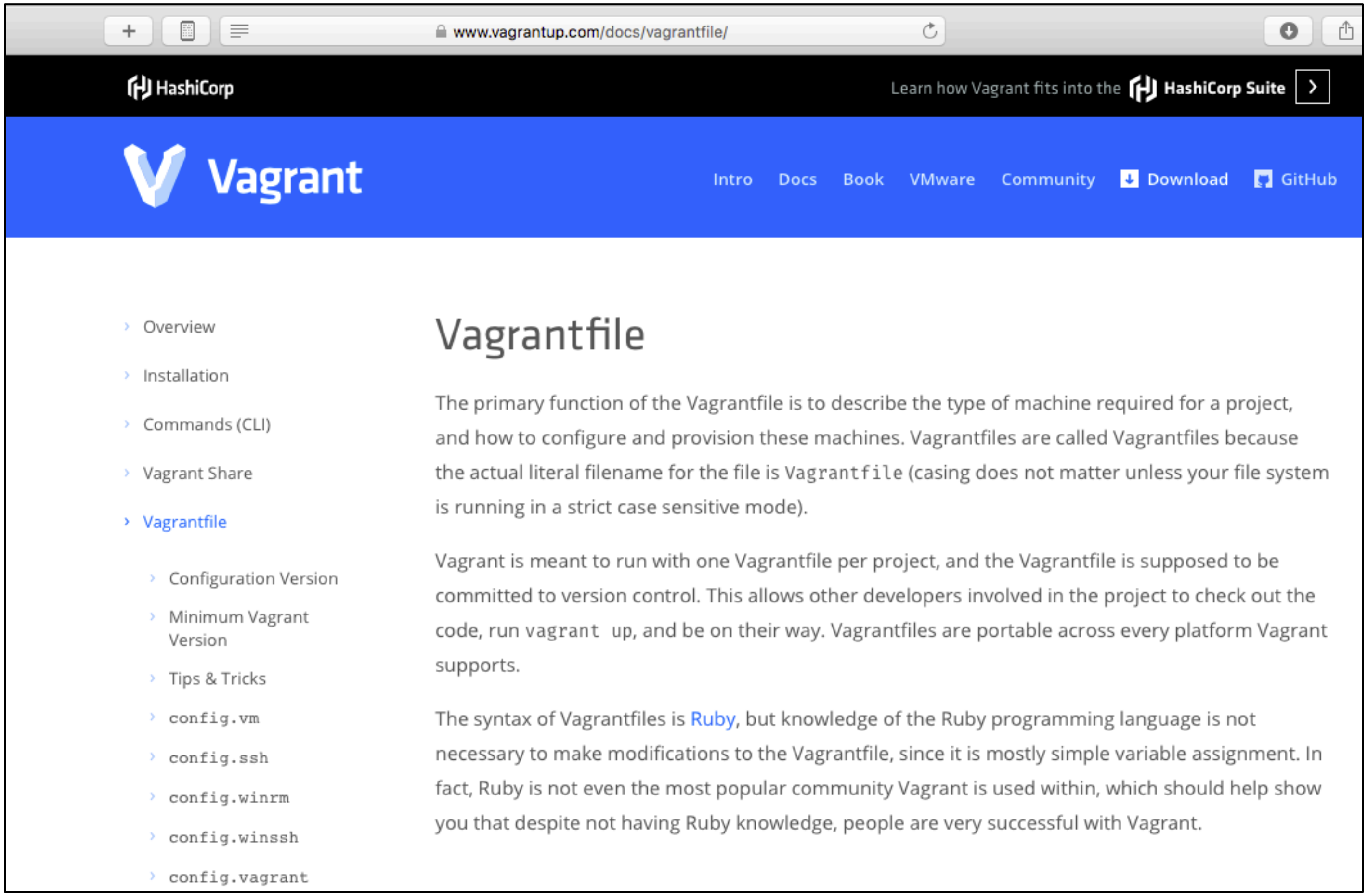
```
vagrant init ubuntu/trusty64
vagrant up
```

Example 02: Vagrantfile Reference

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

documents vagrant files

Example 02: Vagrantfile Reference



Vagrantfile

The primary function of the Vagrantfile is to describe the type of machine required for a project, and how to configure and provision these machines. Vagrantfiles are called Vagrantfiles because the actual literal filename for the file is `Vagrantfile` (casing does not matter unless your file system is running in a strict case sensitive mode).

Vagrant is meant to run with one Vagrantfile per project, and the Vagrantfile is supposed to be committed to version control. This allows other developers involved in the project to check out the code, run `vagrant up`, and be on their way. Vagrantfiles are portable across every platform Vagrant supports.

The syntax of Vagrantfiles is Ruby, but knowledge of the Ruby programming language is not necessary to make modifications to the Vagrantfile, since it is mostly simple variable assignment. In fact, Ruby is not even the most popular community Vagrant is used within, which should help show you that despite not having Ruby knowledge, people are very successful with Vagrant.

Example 02: Vagrantfile

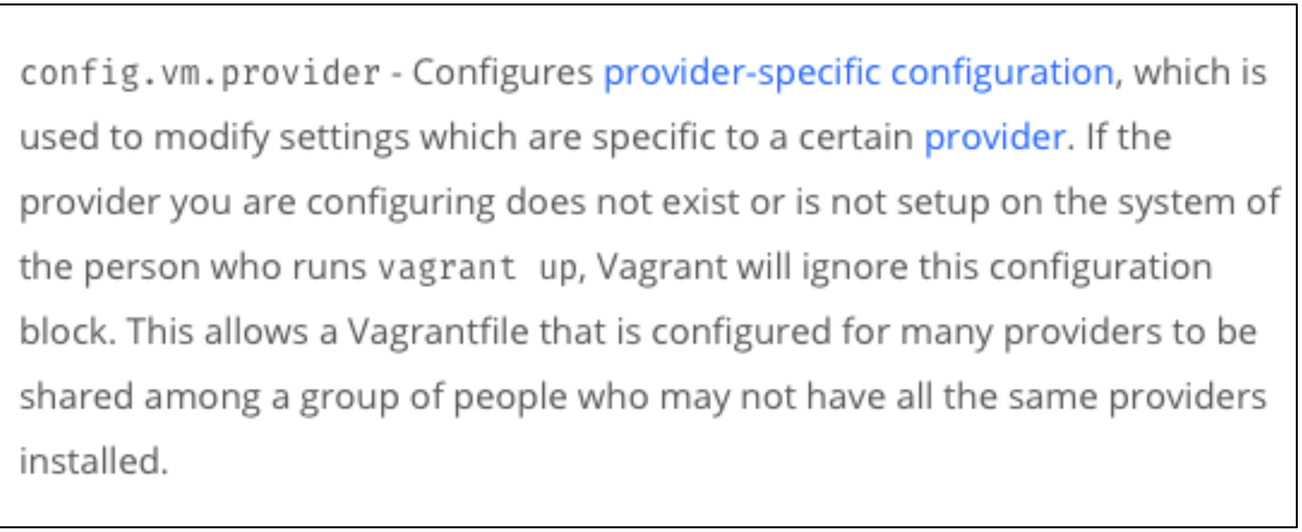
Our first edit in the vagrantfile is to set the box to ubuntu/trusty64

```
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/trusty64"
…
```

Example 02: Vagrantfile

Next, we will add some VirtualBox specific settings

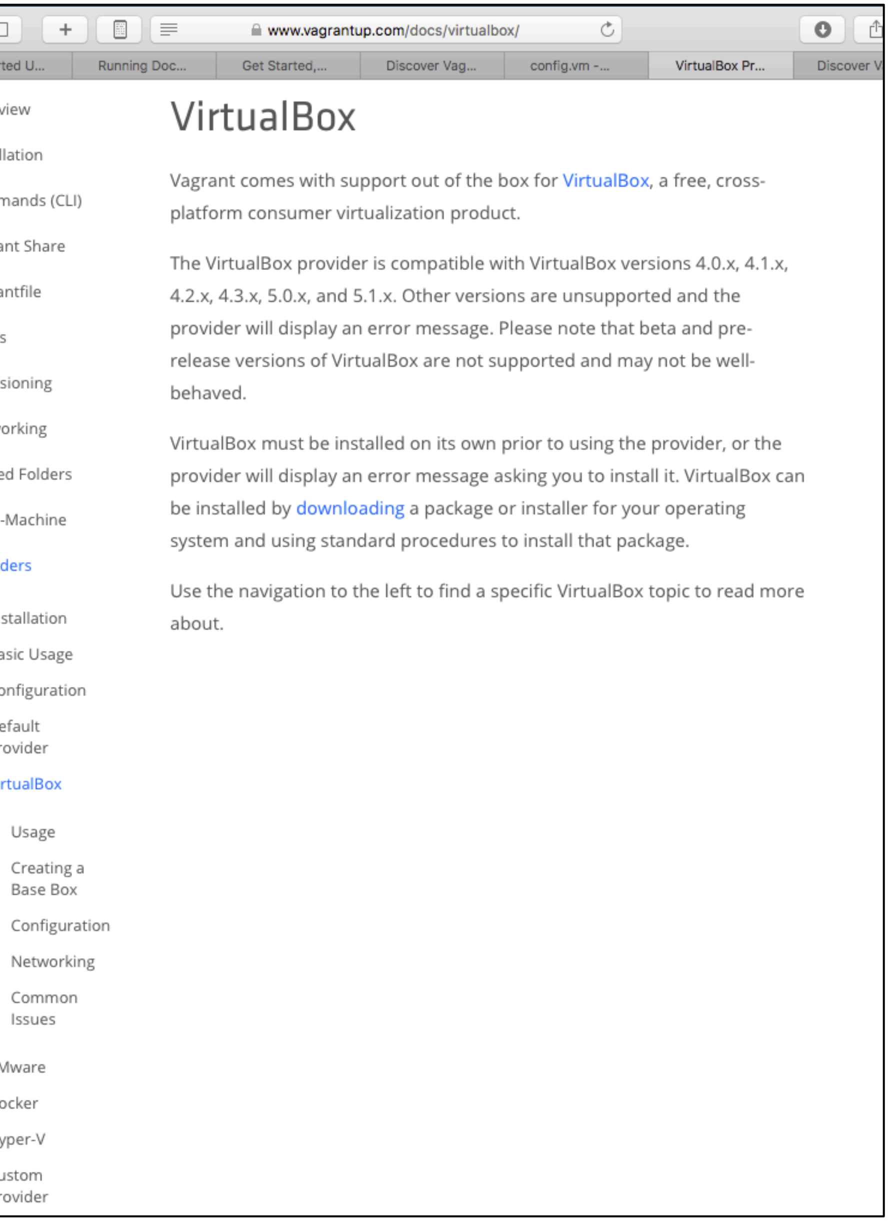
config.vm.provider – sets up provider specific settings:



`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/vagrantfi



VirtualBox

Vagrant comes with support out of the box for VirtualBox, a free, cross-platform consumer virtualization product.

The VirtualBox provider is compatible with VirtualBox versions 4.0.x, 4.1.x, 4.2.x, 4.3.x, 5.0.x, and 5.1.x. Other versions are unsupported and the provider will display an error message. Please note that beta and pre-release versions of VirtualBox are not supported and may not be well-behaved.

VirtualBox must be installed on its own prior to using the provider, or the provider will display an error message asking you to install it. VirtualBox can be installed by downloading a package or installer for your operating system and using standard procedures to install that package.

Use the navigation to the left to find a specific VirtualBox topic to read more about.

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/trusty64"
  ...
 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 = "1024"

    # VM name
    virtbox.name = "d2VM"
 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

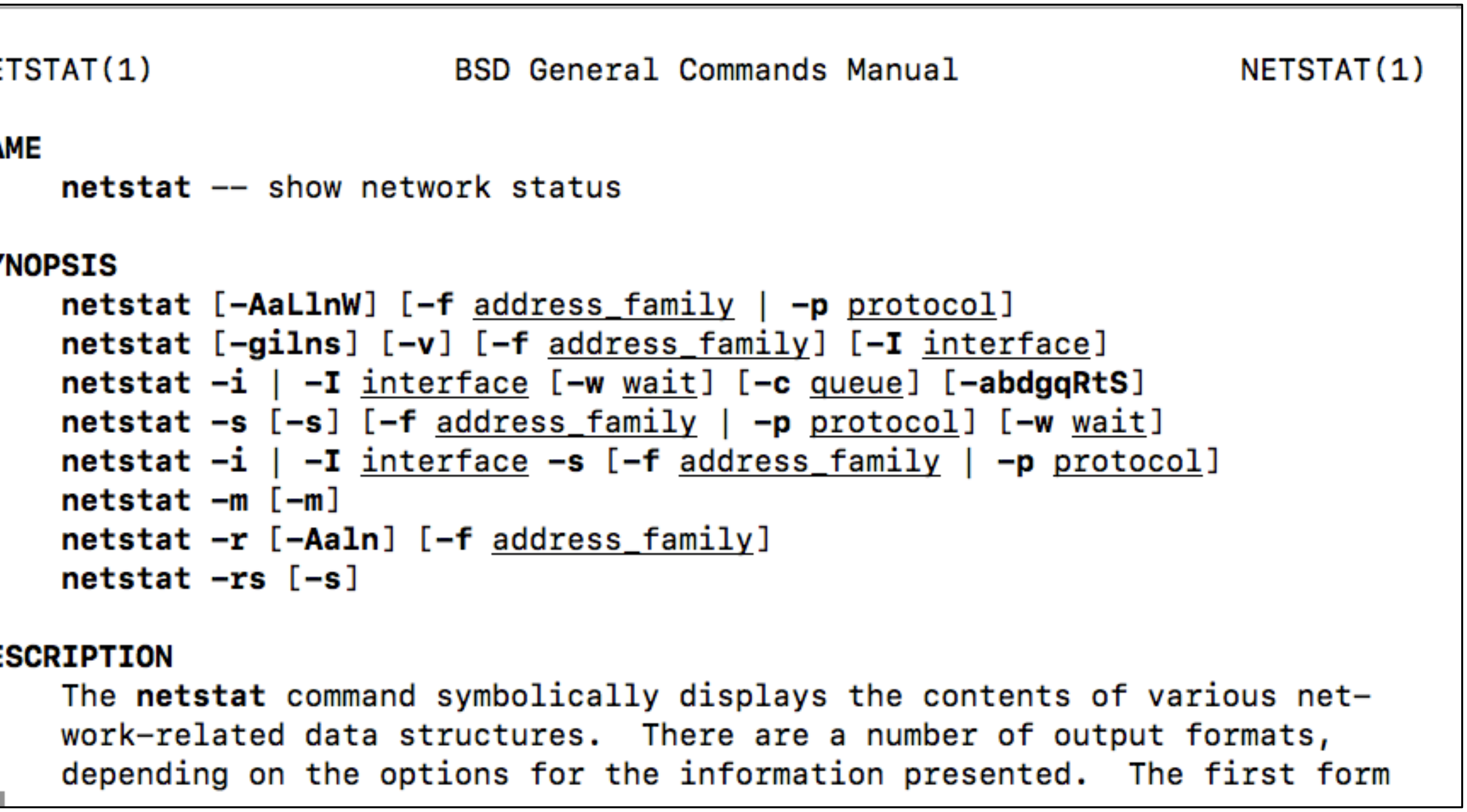
Question − how can we determine which IP port are available on the host machine?

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

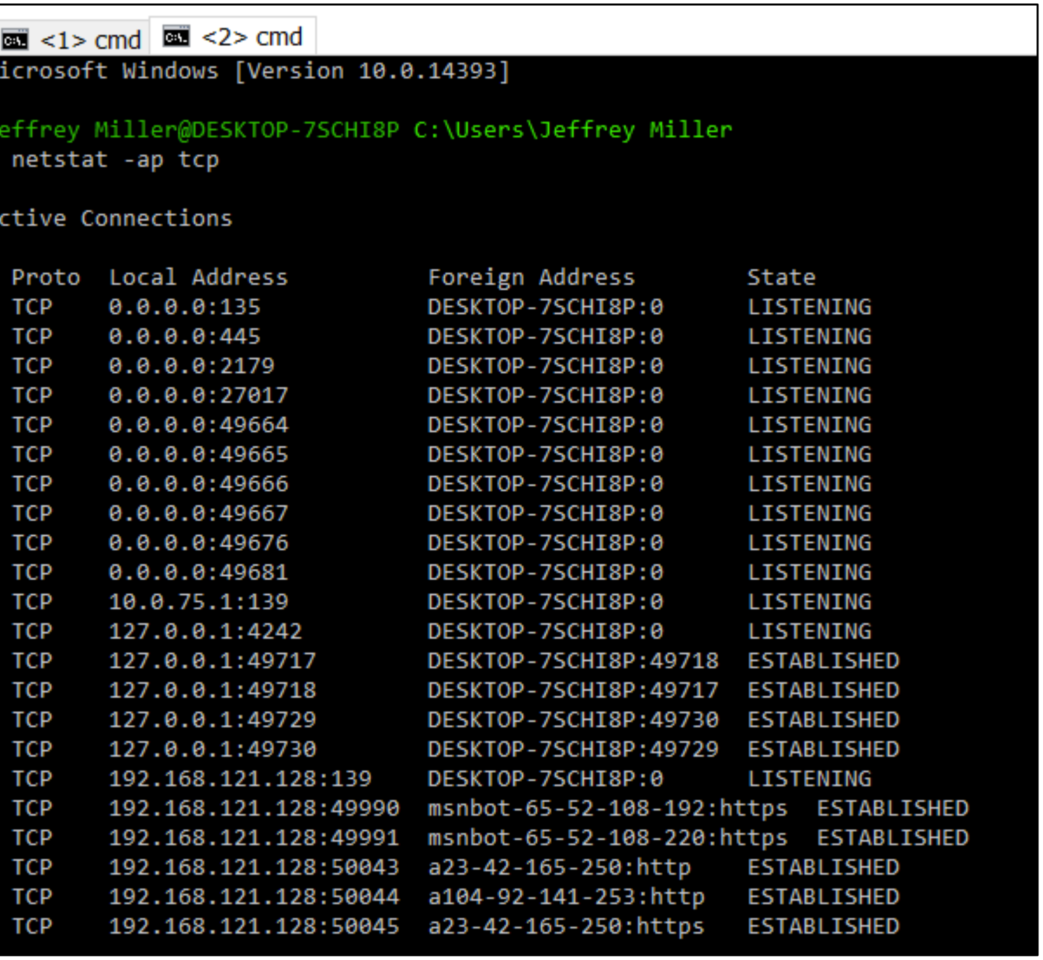
Windows, Cygwin on Windows, Linux, and Mac support the netstat command:



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

AME
   netstat -- show network status

NOPSIS
   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]

SCRIPTION
   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

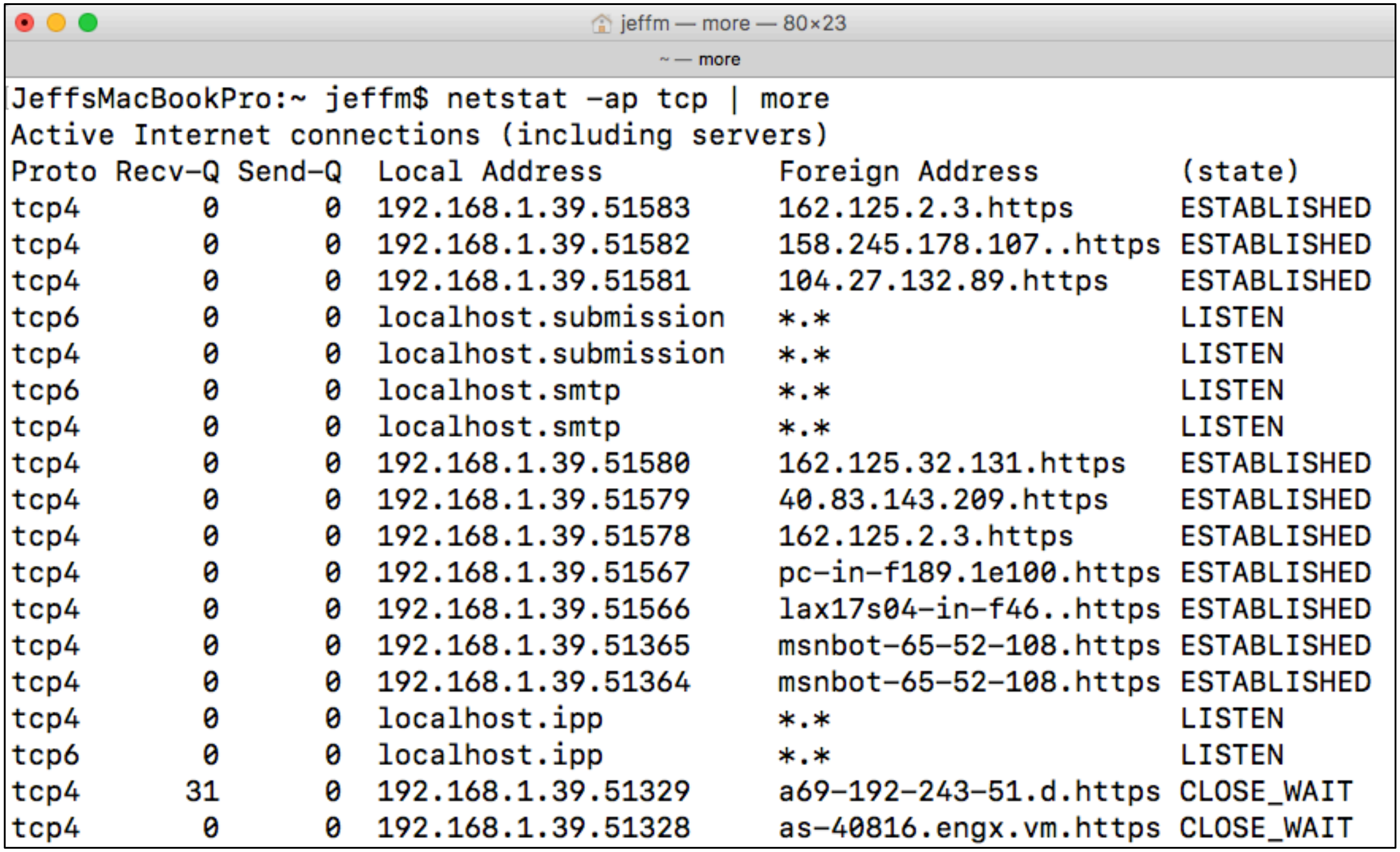


```
icrosoft Windows [Version 10.0.14393]

effrey Miller@DESKTOP-7SCHI8P C:\Users\Jeffrey Miller
 netstat -ap tcp

ctive Connections

 Proto  Local Address          Foreign Address        State
 TCP    0.0.0.0:135            DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:445            DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:2179           DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:27017          DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:49664          DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:49665          DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:49666          DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:49667          DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:49676          DESKTOP-7SCHI8P:0      LISTENING
 TCP    0.0.0.0:49681          DESKTOP-7SCHI8P:0      LISTENING
 TCP    10.0.75.1:139          DESKTOP-7SCHI8P:0      LISTENING
 TCP    127.0.0.1:4242         DESKTOP-7SCHI8P:0      LISTENING
 TCP    127.0.0.1:49717        DESKTOP-7SCHI8P:49718  ESTABLISHED
 TCP    127.0.0.1:49718        DESKTOP-7SCHI8P:49717  ESTABLISHED
 TCP    127.0.0.1:49729        DESKTOP-7SCHI8P:49730  ESTABLISHED
 TCP    127.0.0.1:49730        DESKTOP-7SCHI8P:49729  ESTABLISHED
 TCP    192.168.121.128:139    DESKTOP-7SCHI8P:0      LISTENING
 TCP    192.168.121.128:49990  msnbot-65-52-108-192:https  ESTABLISHED
 TCP    192.168.121.128:49991  msnbot-65-52-108-220:https  ESTABLISHED
 TCP    192.168.121.128:50043  a23-42-165-250:http    ESTABLISHED
 TCP    192.168.121.128:50044  a104-92-141-253:http   ESTABLISHED
 TCP    192.168.121.128:50045  a23-42-165-250:https   ESTABLISHED
```

netstat –ap tcp

on Windows

Example 02: Vagrantfile, IP Ports



```
JeffsMacBookPro:~ jeffm$ netstat -ap tcp | more
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.1.39.51583     162.125.2.3.https      ESTABLISHED
tcp4       0      0  192.168.1.39.51582     158.245.178.107..https ESTABLISHED
tcp4       0      0  192.168.1.39.51581     104.27.132.89.https    ESTABLISHED
tcp6       0      0  localhost.submission   *.*                    LISTEN
tcp4       0      0  localhost.submission   *.*                    LISTEN
tcp6       0      0  localhost.smtp         *.*                    LISTEN
tcp4       0      0  localhost.smtp         *.*                    LISTEN
tcp4       0      0  192.168.1.39.51580     162.125.32.131.https   ESTABLISHED
tcp4       0      0  192.168.1.39.51579     40.83.143.209.https    ESTABLISHED
tcp4       0      0  192.168.1.39.51578     162.125.2.3.https      ESTABLISHED
tcp4       0      0  192.168.1.39.51567     pc-in-f189.1e100.https ESTABLISHED
tcp4       0      0  192.168.1.39.51566     lax17s04-in-f46..https ESTABLISHED
tcp4       0      0  192.168.1.39.51365     msnbot-65-52-108.https ESTABLISHED
tcp4       0      0  192.168.1.39.51364     msnbot-65-52-108.https ESTABLISHED
tcp4       0      0  localhost.ipp          *.*                    LISTEN
tcp6       0      0  localhost.ipp          *.*                    LISTEN
tcp4      31      0  192.168.1.39.51329     a69-192-243-51.d.https CLOSE_WAIT
tcp4       0      0  192.168.1.39.51328     as-40816.engx.vm.https CLOSE_WAIT
```

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 28017 ports in the VM into host port 37017 and 38017

NOTE: you may have to use different ports on your host machine

Example 02: Vagrantfile, IP Ports

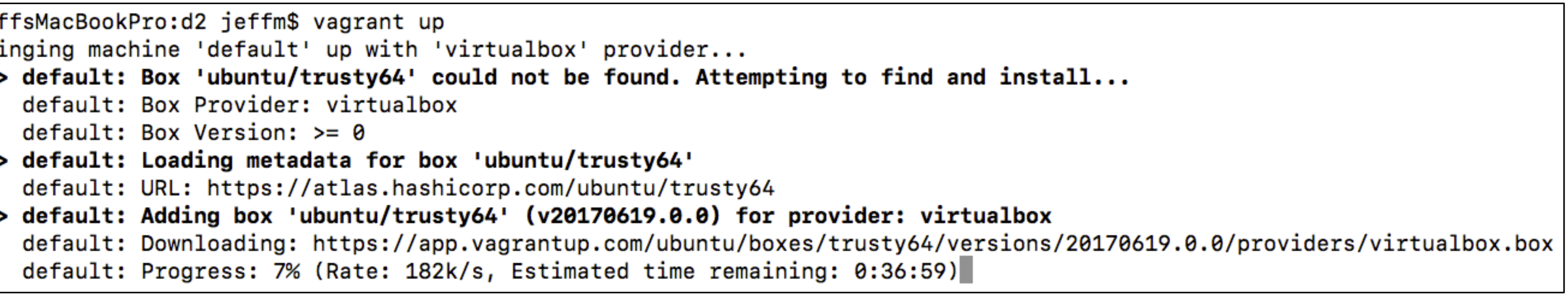
Create a forwarded port mapping which allows access to a specific port
within the machine from a port on the host machine and only allow access
via 127.0.0.1 to disable public access

```
config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"
onfig.vm.network "forwarded_port", guest: 27017, host: 37017, host_ip: "127.0.0.1"
onfig.vm.network "forwarded_port", guest: 28017, host: 38017, host_ip: "127.0.0.1"
```

Example 02: Vagrantfile

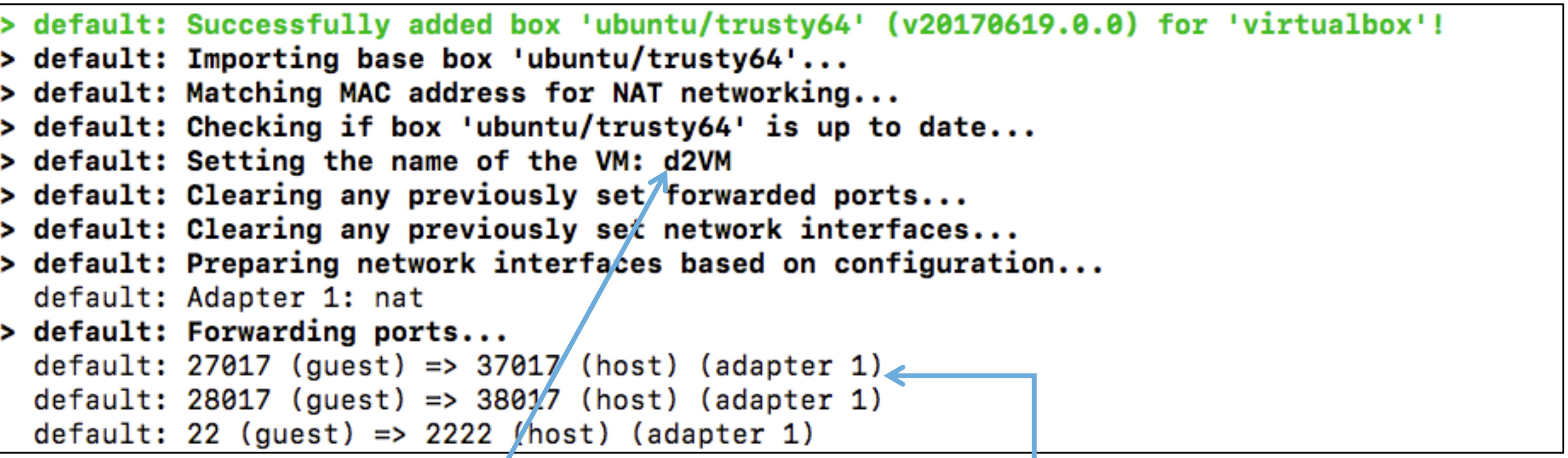
Now, lets test our Vagrant file to make sure it works:

Example 02: Vagrantfile



```
ffsMacBookPro:d2 jeffm$ vagrant up
inging machine 'default' up with 'virtualbox' provider...
> default: Box 'ubuntu/trusty64' could not be found. Attempting to find and install...
  default: Box Provider: virtualbox
  default: Box Version: >= 0
> default: Loading metadata for box 'ubuntu/trusty64'
  default: URL: https://atlas.hashicorp.com/ubuntu/trusty64
> default: Adding box 'ubuntu/trusty64' (v20170619.0.0) for provider: virtualbox
  default: Downloading: https://app.vagrantup.com/ubuntu/boxes/trusty64/versions/20170619.0.0/providers/virtualbox.box
  default: Progress: 7% (Rate: 182k/s, Estimated time remaining: 0:36:59)
```

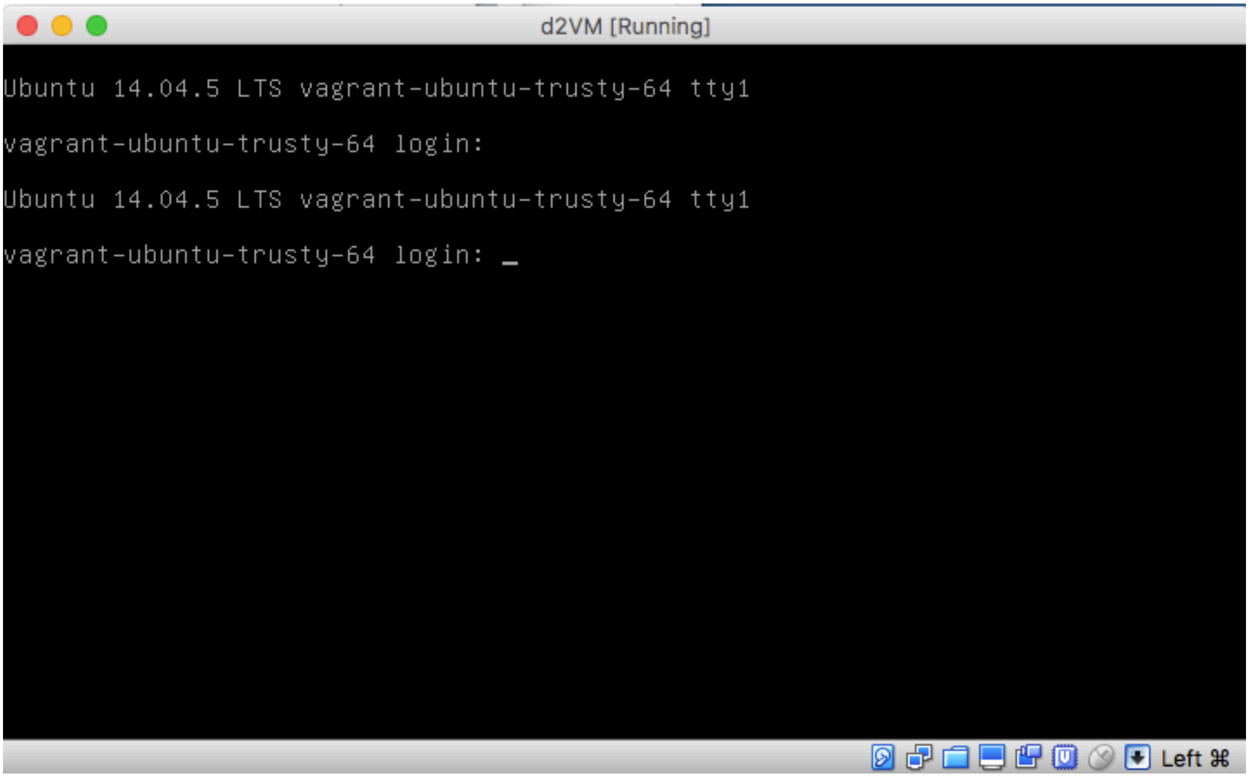
Vagrant automatically downloads ubuntu/trusty64 if it cannot find it on the local machine



```
> default: Successfully added box 'ubuntu/trusty64' (v20170619.0.0) for 'virtualbox'!
> default: Importing base box 'ubuntu/trusty64'...
> default: Matching MAC address for NAT networking...
> default: Checking if box 'ubuntu/trusty64' is up to date...
> default: Setting the name of the VM: d2VM
> default: Clearing any previously set forwarded ports...
> 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: 28017 (guest) => 38017 (host) (adapter 1)
  default: 22 (guest) => 2222 (host) (adapter 1)
```

```
agrant.configure("2") do |config|
config.vm.box = "ubuntu/trusty64"

config.vm.network "forwarded_port", guest: 27017, host: 37017, host_ip: "127.0.0.1"
config.vm.network "forwarded_port", guest: 28017, host: 38017, host_ip: "127.0.0.1"

onfig.vm.provider "virtualbox" do |virtbox|
virtbox.gui = true
virtbox.memory = "1024"
virtbox.name = "d2VM"
end
```

Example 02: Vagrantfile



```
Ubuntu 14.04.5 LTS vagrant-ubuntu-trusty-64 tty1

vagrant-ubuntu-trusty-64 login:

Ubuntu 14.04.5 LTS vagrant-ubuntu-trusty-64 tty1

vagrant-ubuntu-trusty-64 login: _
```

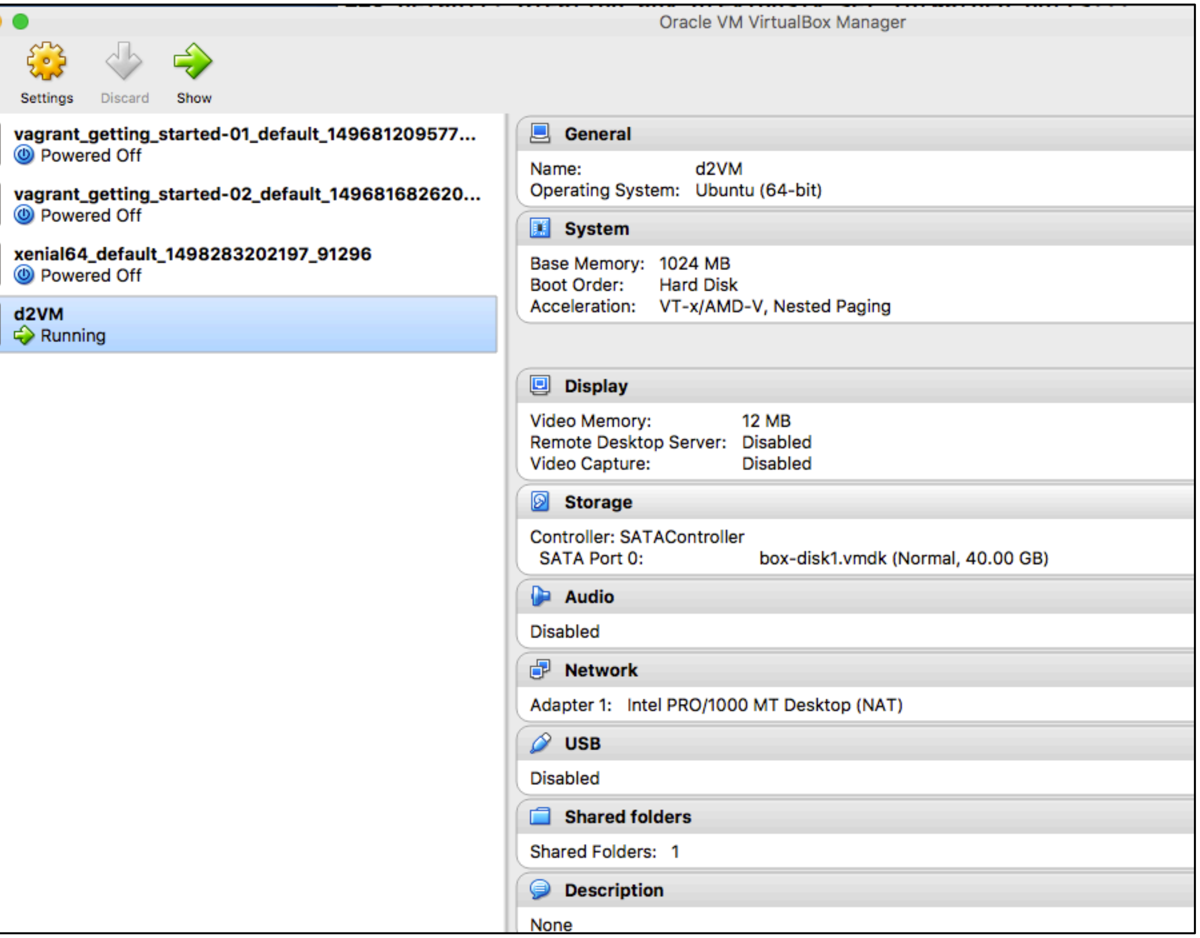
we set gui to true in the Vagrant file – s VirtualBox brought up a terminal to log into.

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 interacti UI. Instead automated ssh access, whic we will cover later on, is used.

Example 02: Vagrantfile



vagrant_getting_started-01_default_149681209577...
Powered Off

vagrant_getting_started-02_default_149681682620...
Powered Off

xenial64_default_1498283202197_91296
Powered Off

d2VM
Running

General

Name: d2VM
Operating System: Ubuntu (64-bit)

System

Base Memory: 1024 MB
Boot Order: Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

Display

Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

Controller: SATAController
SATA Port 0: box-disk1.vmdk (Normal, 40.00 GB)

Audio

Disabled

Network

Adapter 1: Intel PRO/1000 MT Desktop (NAT)

USB

Disabled

Shared folders

Shared Folders: 1

Description

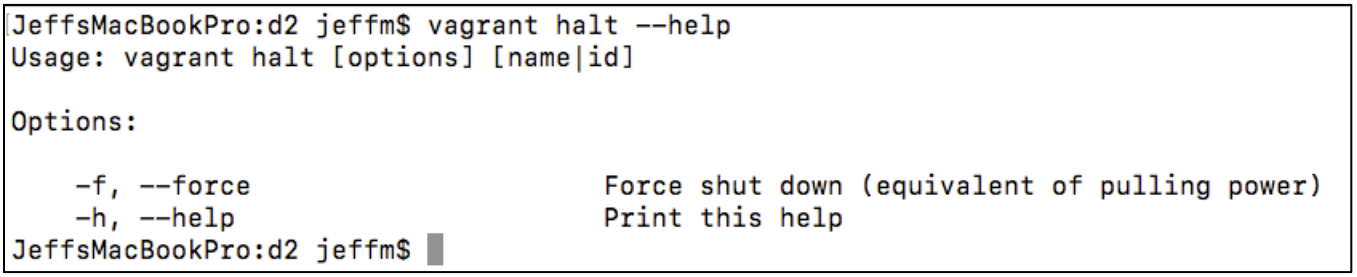
None

Here is the VirtualBox UI – showing our Vagrant provisioned VM

Example 02: Vagrantfile

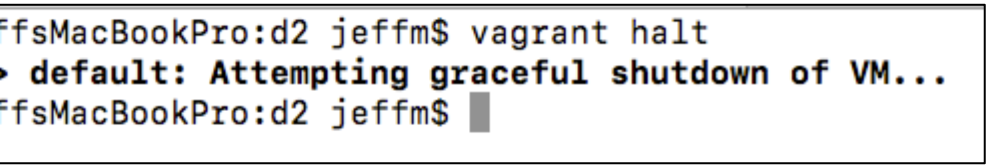
Let's halt the VM and add a Vagrant previsioning script to install Docker (ce) in the VM.

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

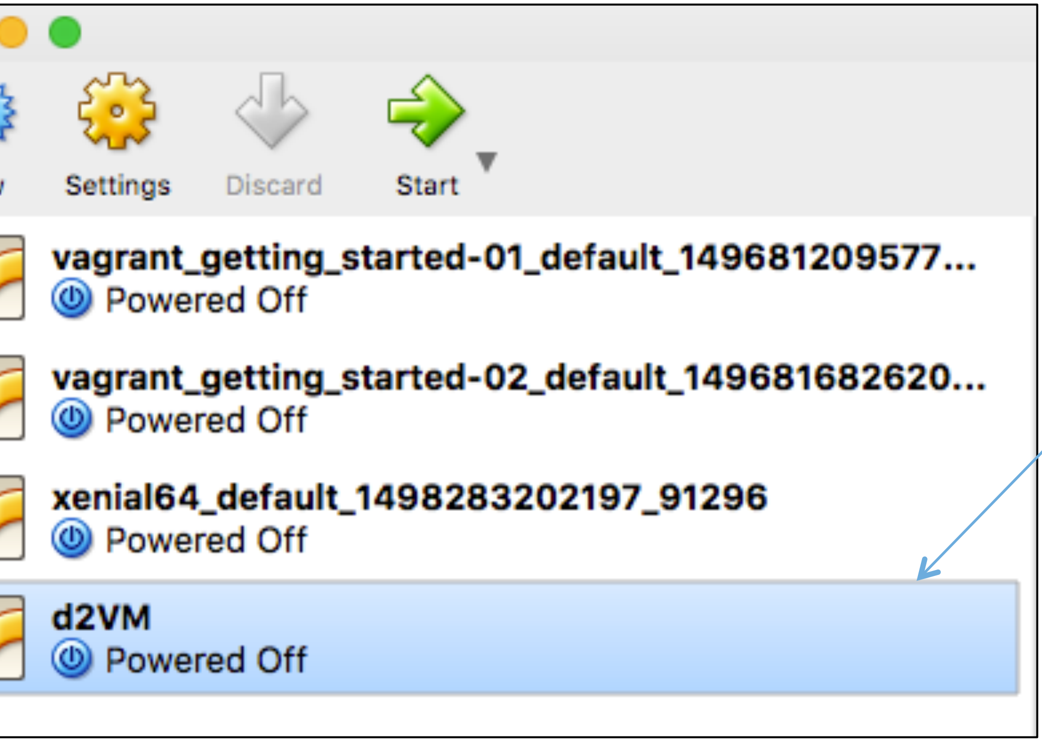


```
JeffsMacBookPro:d2 jeffm$ vagrant halt --help
Usage: vagrant halt [options] [name|id]

Options:

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

Example 02: Vagrantfile



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



vagrant_getting_started-01_default_149681209577...
Powered Off

vagrant_getting_started-02_default_149681682620...
Powered Off

xenial64_default_1498283202197_91296
Powered Off

d2VM
Powered Off

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 intc our Ubuntu running in VirtualBox

Looking at URL:

https://docs.docker.com/engine/installation/linux/ubuntu

Example 02: Provisioning Script to install Docker

For Trusty the following installation steps are documented:

```
sudo apt-get update

# the \ is  a line continuation character
sudo apt-get install  \
linux-image-extra-$(uname -r) \
   linux-image-extra-virtual

#we will skip repository installation step for now
#NOTE: do NOT use command: sudo apt-get install docker-ce
sudo curl -sSL https://get.docker.com/ | sh
```

Example 02: Provisioning Script to install Docker

We will put the commands to install docker into a Bash shell script called - install-docker.sh − and make this script a Vagrant provisioning script

Suggestion: before running the script as a Vagrant provisioning script

- put all of the commands into script file **install-docker.sh** and run the script file in a ssh terminal connected to the Trusty VM
- delete docker using:

  ```
  sudo apt-get −purge remove docker-ce
  ```
- Once you know the script works, use the script file as a Vagrant provisioning file

Example 02: Provisioning Script to install Docker

```
#!/usr/bin/env bash

sudo apt-get update
sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual
sudo curl -sSL https://get.docker.com/ | sh
```

install-docker.sh

The sudo curl -sSL https://get.docker.com/ | sh command:

- download the get-docker script from get.docker.com
- pipes the output of the curl command (i.e. the script) in a shell to execute it.
- you can type https://get.docker.com into your browser to see this script

NOTE: do NOT use command: sudo apt-get install docker-ce

Example 02: Provisioning Script to install Docker

```
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/trusty64"

  config.vm.network "forwarded_port", guest: 27017, host: 37017, host_ip: "127.0.0.1"
  config.vm.network "forwarded_port", guest: 28017, host: 38017, host_ip: "127.0.0.1"


  config.vm.provider "virtualbox" do |virtbox|
    virtbox.gui = true
    virtbox.memory = "1024"
    virtbox.name = "d2VM"
  end

  config.vm.provision :shell, path: "install-docker.sh"

end
```

Vagrant file including the command to run install-docker.sh

Example 02: Updating after revising Vagrant file

When changes are made to a Vagrant file you need to:

- halt the VM − vagrant halt
- followed by - vagrant up

vagrant reload − halts the virtual machine and restarts it by calling halt and up

<u>However</u> − if you update a provisioner − as we did - …

Example 02: Updates after revising Vagrant file

<u>However</u> – if you update a provisioner – as we did - …

- if the VM is NOT running using

```
vagrant up --provision
#or
vagrant up --provision-with shell #since we added a shell provisioning script
```

- if the VM is running:

```
vagrant reload --provision
#or
vagrant reload --provision-with shell
```

Example 02: Updates after revising Vagrant file

Recall from previous sections:

Provisioners in Vagrant allow you to automatically install software, alter configurations, and mor 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).

Example 02: Updates after revising Vagrant file

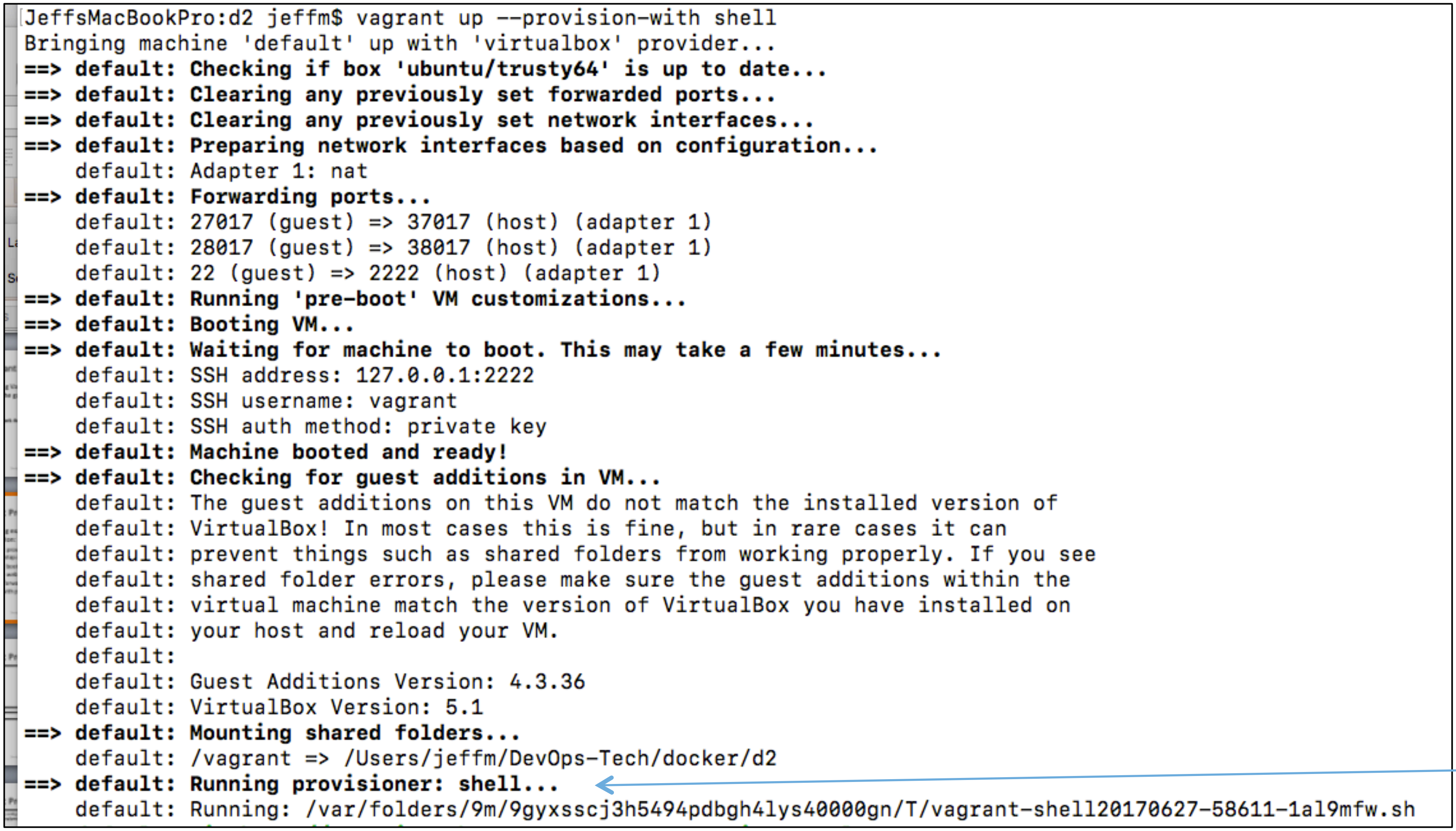
Since the VM is halted command - **vagrant up --provision** − will be used.

The next several slides will show the output from running:

vagrant up --provision

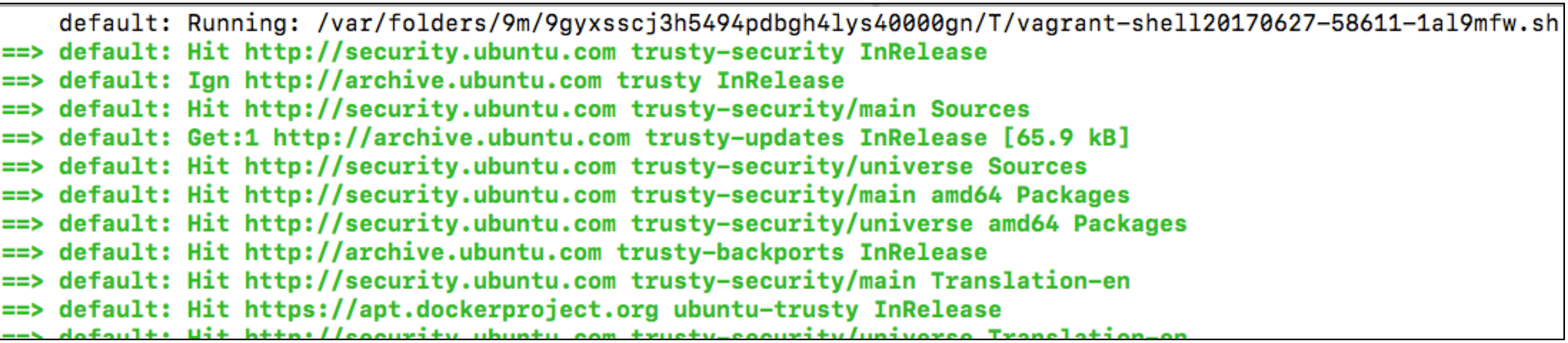
NOTE: only selected portions of the output are presented

Example 02: Updates after revising Vagrant file



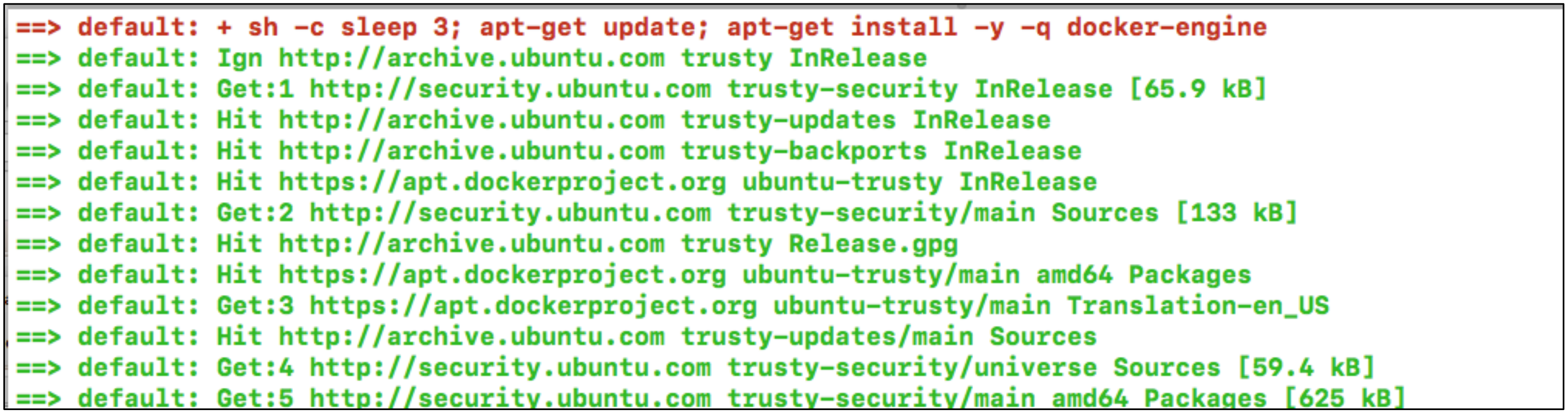
```
JeffsMacBookPro:d2 jeffm$ vagrant up --provision-with shell
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/trusty64' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 27017 (guest) => 37017 (host) (adapter 1)
    default: 28017 (guest) => 38017 (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: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 4.3.36
    default: VirtualBox Version: 5.1
==> default: Mounting shared folders...
    default: /vagrant => /Users/jeffm/DevOps-Tech/docker/d2
==> default: Running provisioner: shell...
    default: Running: /var/folders/9m/9gyxsscj3h5494pdbgh4lys40000gn/T/vagrant-shell20170627-58611-1al9mfw.sh
```

Example 02: Updates after revising Vagrant file



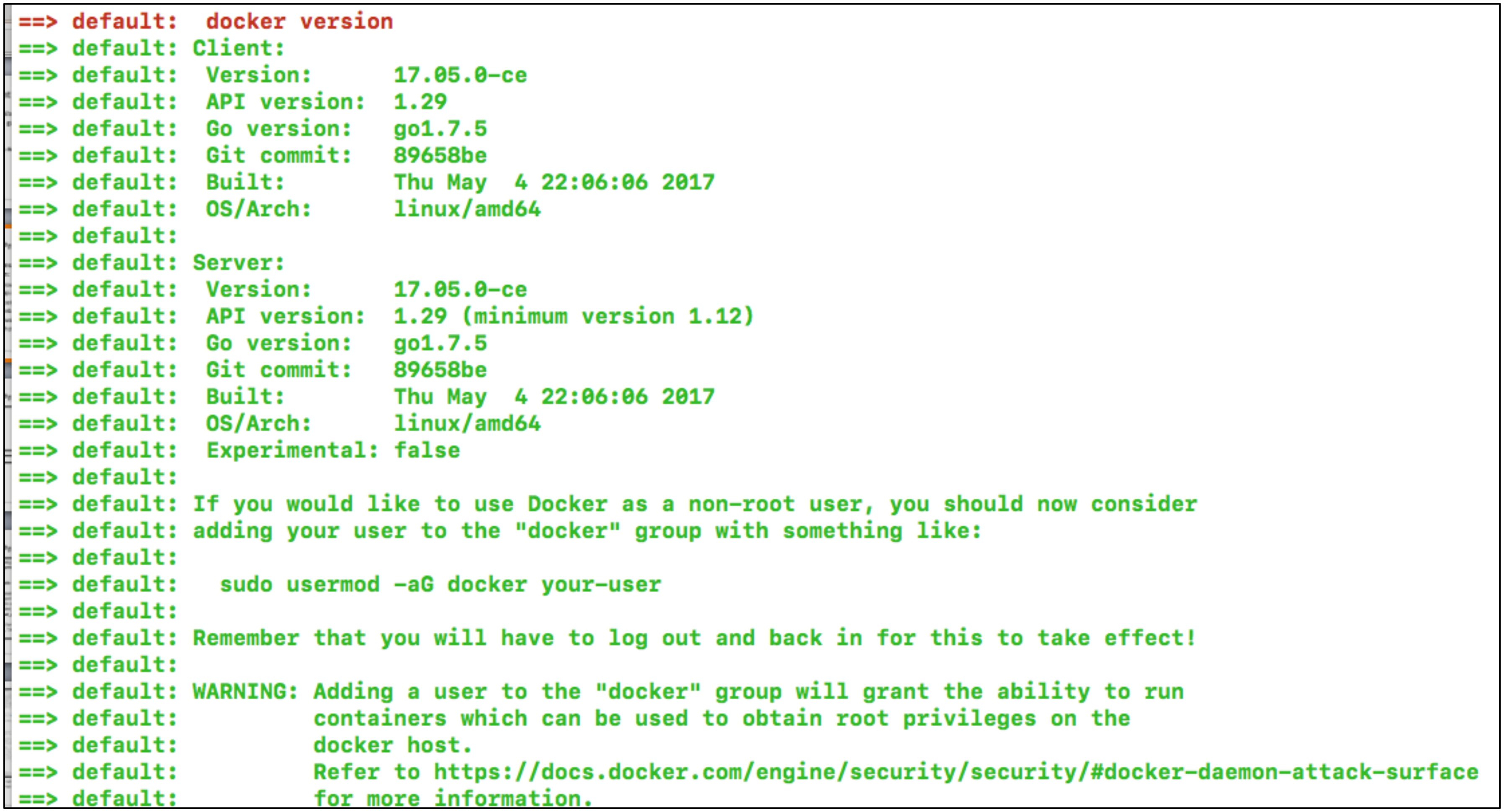
```
    default: Running: /var/folders/9m/9gyxsscj3h5494pdbgh4lys40000gn/T/vagrant-shell20170627-58611-1al9mfw.sh
==> default: Hit http://security.ubuntu.com trusty-security InRelease
==> default: Ign http://archive.ubuntu.com trusty InRelease
==> default: Hit http://security.ubuntu.com trusty-security/main Sources
==> default: Get:1 http://archive.ubuntu.com trusty-updates InRelease [65.9 kB]
==> default: Hit http://security.ubuntu.com trusty-security/universe Sources
==> default: Hit http://security.ubuntu.com trusty-security/main amd64 Packages
==> default: Hit http://security.ubuntu.com trusty-security/universe amd64 Packages
==> default: Hit http://archive.ubuntu.com trusty-backports InRelease
==> default: Hit http://security.ubuntu.com trusty-security/main Translation-en
==> default: Hit https://apt.dockerproject.org ubuntu-trusty InRelease
==> default: Hit http://security.ubuntu.com trusty-security/universe Translation-en
```

Example 02: Updates after revising Vagrant file



```
==> default: + sh -c sleep 3; apt-get update; apt-get install -y -q docker-engine
==> default: Ign http://archive.ubuntu.com trusty InRelease
==> default: Get:1 http://security.ubuntu.com trusty-security InRelease [65.9 kB]
==> default: Hit http://archive.ubuntu.com trusty-updates InRelease
==> default: Hit http://archive.ubuntu.com trusty-backports InRelease
==> default: Hit https://apt.dockerproject.org ubuntu-trusty InRelease
==> default: Get:2 http://security.ubuntu.com trusty-security/main Sources [133 kB]
==> default: Hit http://archive.ubuntu.com trusty Release.gpg
==> default: Hit https://apt.dockerproject.org ubuntu-trusty/main amd64 Packages
==> default: Get:3 https://apt.dockerproject.org ubuntu-trusty/main Translation-en_US
==> default: Hit http://archive.ubuntu.com trusty-updates/main Sources
==> default: Get:4 http://security.ubuntu.com trusty-security/universe Sources [59.4 kB]
==> default: Get:5 http://security.ubuntu.com trusty-security/main amd64 Packages [625 kB]
```

Example 02: Updates after revising Vagrant file



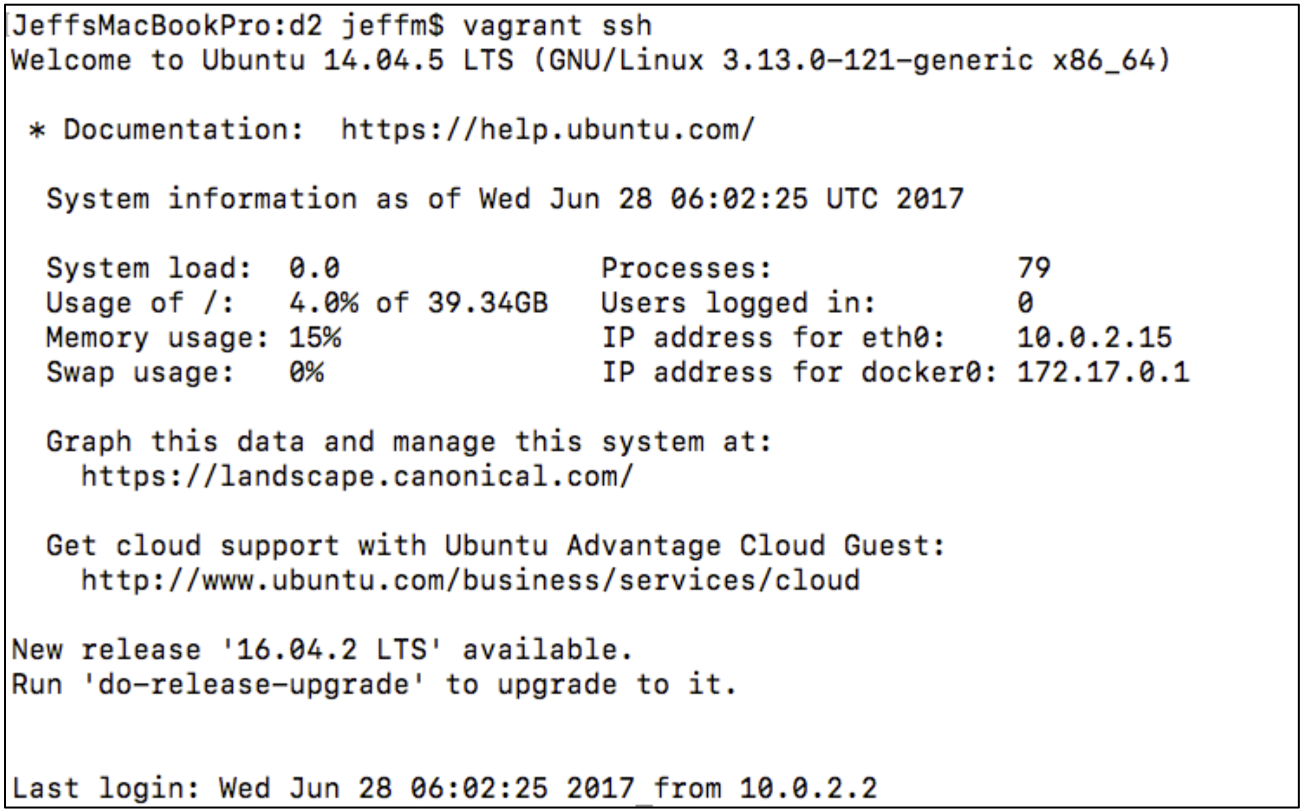
```
==> default:  docker version
==> default: Client:
==> default:  Version:      17.05.0-ce
==> default:  API version:  1.29
==> default:  Go version:   go1.7.5
==> default:  Git commit:   89658be
==> default:  Built:        Thu May  4 22:06:06 2017
==> default:  OS/Arch:      linux/amd64
==> default:
==> default: Server:
==> default:  Version:      17.05.0-ce
==> default:  API version:  1.29 (minimum version 1.12)
==> default:  Go version:   go1.7.5
==> default:  Git commit:   89658be
==> default:  Built:        Thu May  4 22:06:06 2017
==> default:  OS/Arch:      linux/amd64
==> default:  Experimental: false
==> default:
==> default: If you would like to use Docker as a non-root user, you should now consider
==> default: adding your user to the "docker" group with something like:
==> default:
==> default:   sudo usermod -aG docker your-user
==> default:
==> default: Remember that you will have to log out and back in for this to take effect!
==> default:
==> default: WARNING: Adding a user to the "docker" group will grant the ability to run
==> default:          containers which can be used to obtain root privileges on the
==> default:          docker host.
==> default:          Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
==> default:          for more information.
```

Example 02: Running Docker in the VM

OK, let's run "vagrant ssh" to get into the VM

Example 02: Running Docker in the VM

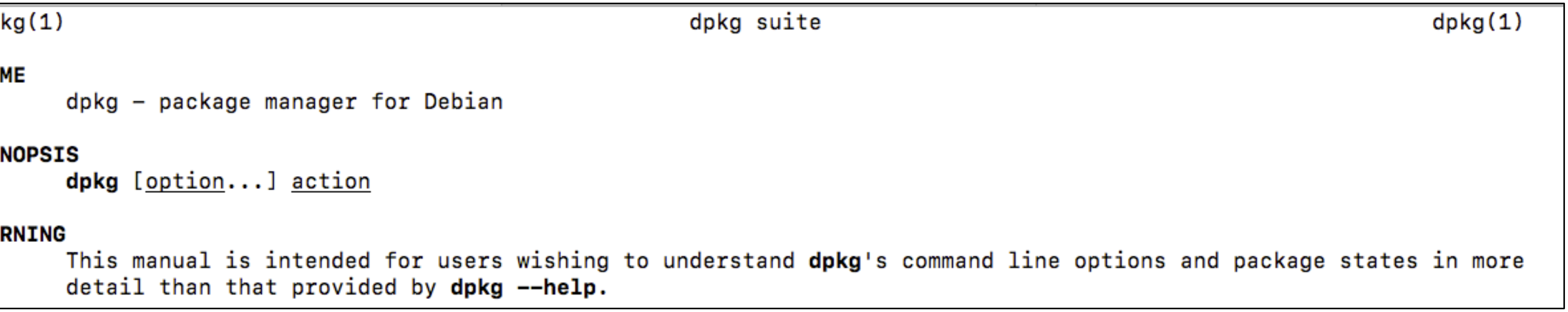
OK, let's run "vagrant ssh" to get into the VM



```
JeffsMacBookPro:d2 jeffm$ vagrant ssh
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-121-generic x86_64)

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

  System information as of Wed Jun 28 06:02:25 UTC 2017

  System load:  0.0               Processes:              79
  Usage of /:   4.0% of 39.34GB   Users logged in:        0
  Memory usage: 15%               IP address for eth0:    10.0.2.15
  Swap usage:   0%                IP address for docker0: 172.17.0.1

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

New release '16.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Wed Jun 28 06:02:25 2017 from 10.0.2.2
```

Example 02: Running Docker in the VM

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

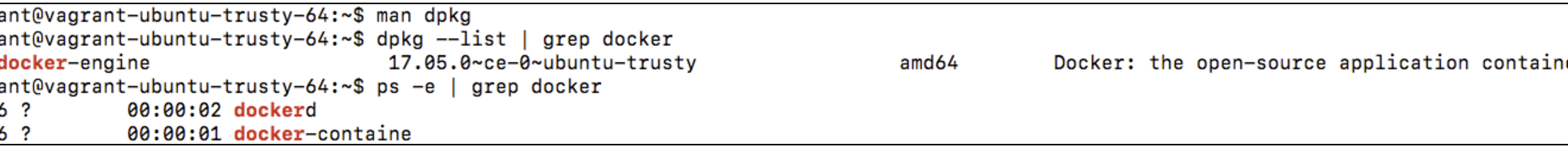


```
kg(1)                                    dpkg suite                                    dpkg(1)

ME
     dpkg - package manager for Debian

NOPSIS
     dpkg [option...] action

RNING
     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

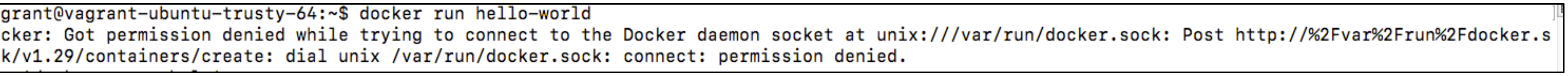


```
ant@vagrant-ubuntu-trusty-64:~$ man dpkg
ant@vagrant-ubuntu-trusty-64:~$ dpkg --list | grep docker
docker-engine                    17.05.0~ce-0~ubuntu-trusty                amd64        Docker: the open-source application contain
ant@vagrant-ubuntu-trusty-64:~$ ps -e | grep docker
6 ?        00:00:02 dockerd
6 ?        00:00:01 docker-containe
```

Example 02: Running Docker in the VM

Let's run docker hello-world in our Vagrant provisioned VM:

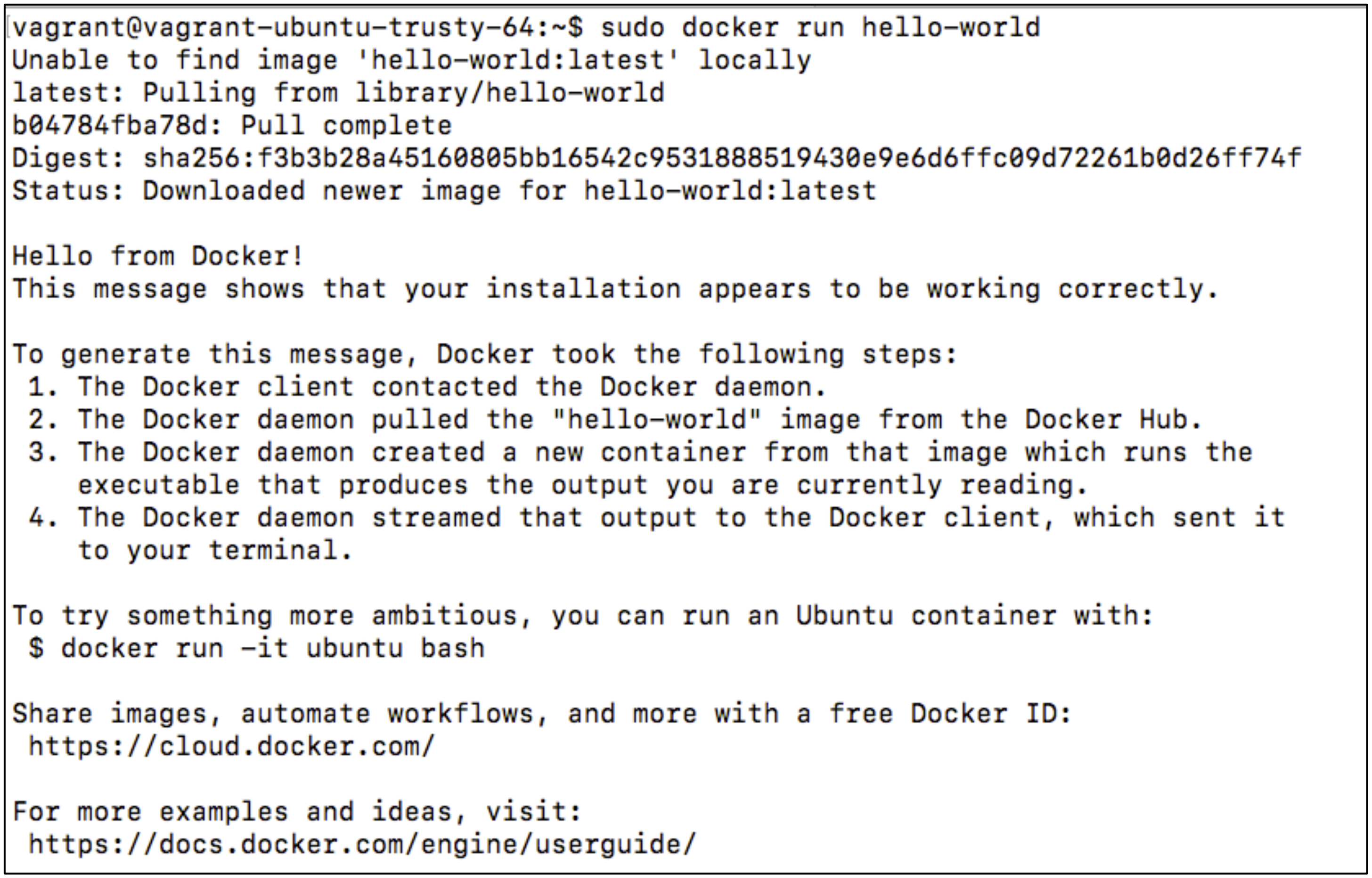
docker run hello-world



```
grant@vagrant-ubuntu-trusty-64:~$ docker run hello-world
cker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.s
k/v1.29/containers/create: dial unix /var/run/docker.sock: connect: permission denied.
```

- Notice we had a permissions error.
- Let's trying again using sudo

Example 02: Running Docker in the VM



```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b04784fba78d: Pull complete
Digest: sha256:f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

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,
    set gui=true,
    mapped VM port to host machine ports, used netstat to find open ports
Ran our first test with the Vagrant file to get ubuntu/trusty64

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

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 Chef, Puppet – which we will study later on

What's Next?

We will look at the docker registry

We will look at the docker file

We will look at more Docker examples