Making Developers More Productive with Vagrant, VirtualBox, and Docker

John J. Rofrano

Senior Technical Staff Member, IBM T.J. Watson Research

usenix LISA16

December 4–9, 2016 | Boston, MA

www.usenix.org/lisa16 #lisa16

DevOps Principle of the Day:

“**AUTOMATE EVERYTHING!**”

Even the Development Team !!!

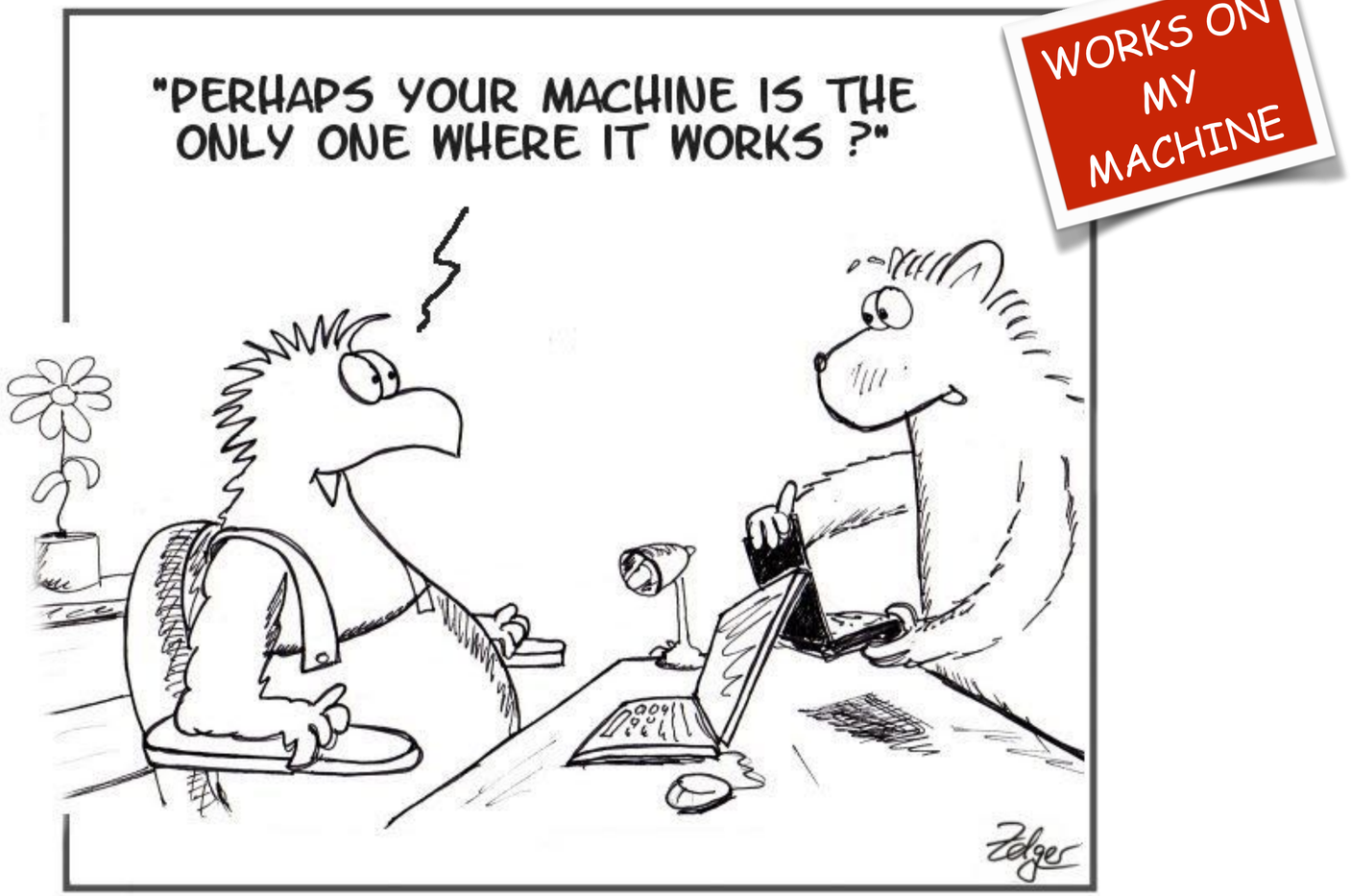
What We Will Cover?

- How to create customized development environments with Vagrant and VirtualBox that can be created and destroyed and created again in a consistent fashion
- How to leverage Docker containers for middleware deployment
- How to make your DevOps team more productive with just a single command: `> vagrant up`

The Problem

- Manually creating local environments for developers to work in is:
  - Time consuming
  - Error prone
  - Inconsistent at best
  - Unreproducible at worst!

The Goal is to Avoid This



It works on my machine

The Wrong Solution

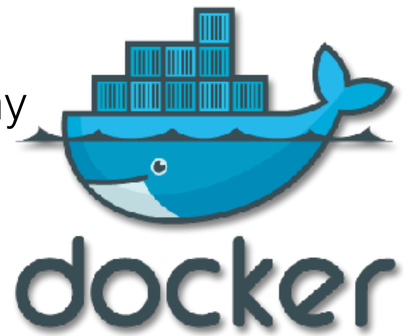
- Carefully and painstakingly create documentation in the form of “runbooks” that give step-by-step instructions on how to recreate the development environment via cut-n-paste
- NOT VERY AGILE !!!
  - *We value: Working software over comprehensive documentation*

The Right Solution

- **Automate the creation of local development environments right on your laptop or desktop**

Use Vagrant to quickly provision complex configurations consistently with repeatability

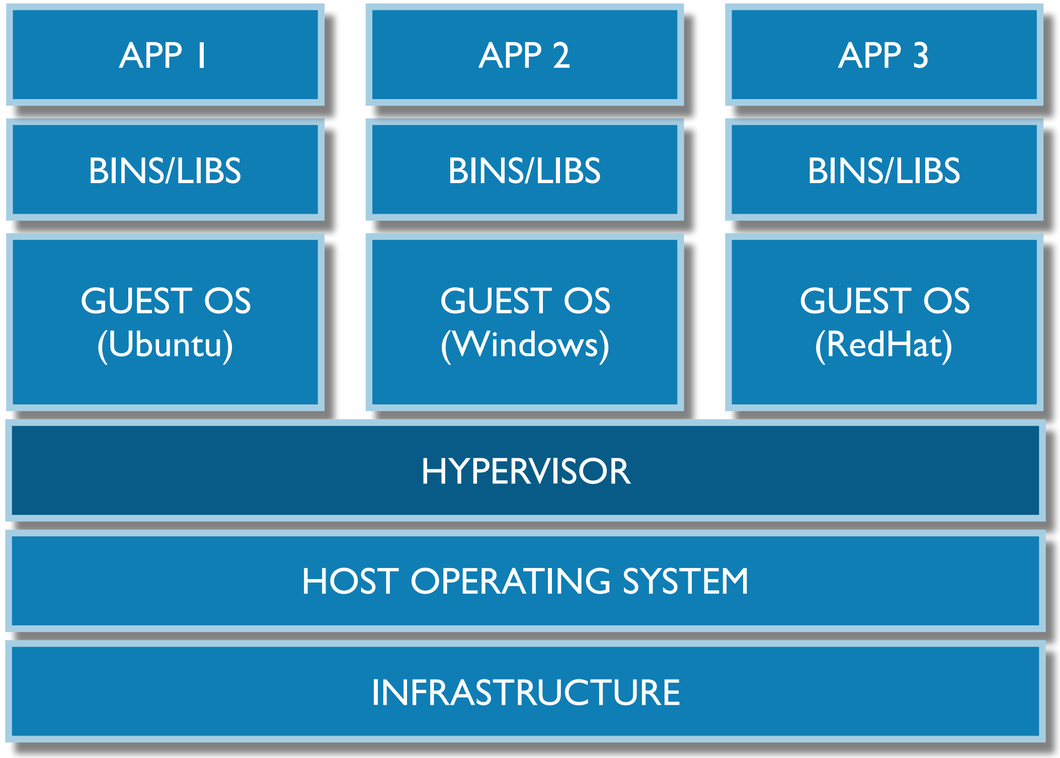
Use Docker to handle middleware without any installation



Use VirtualBox to create Virtual Machines to develop in

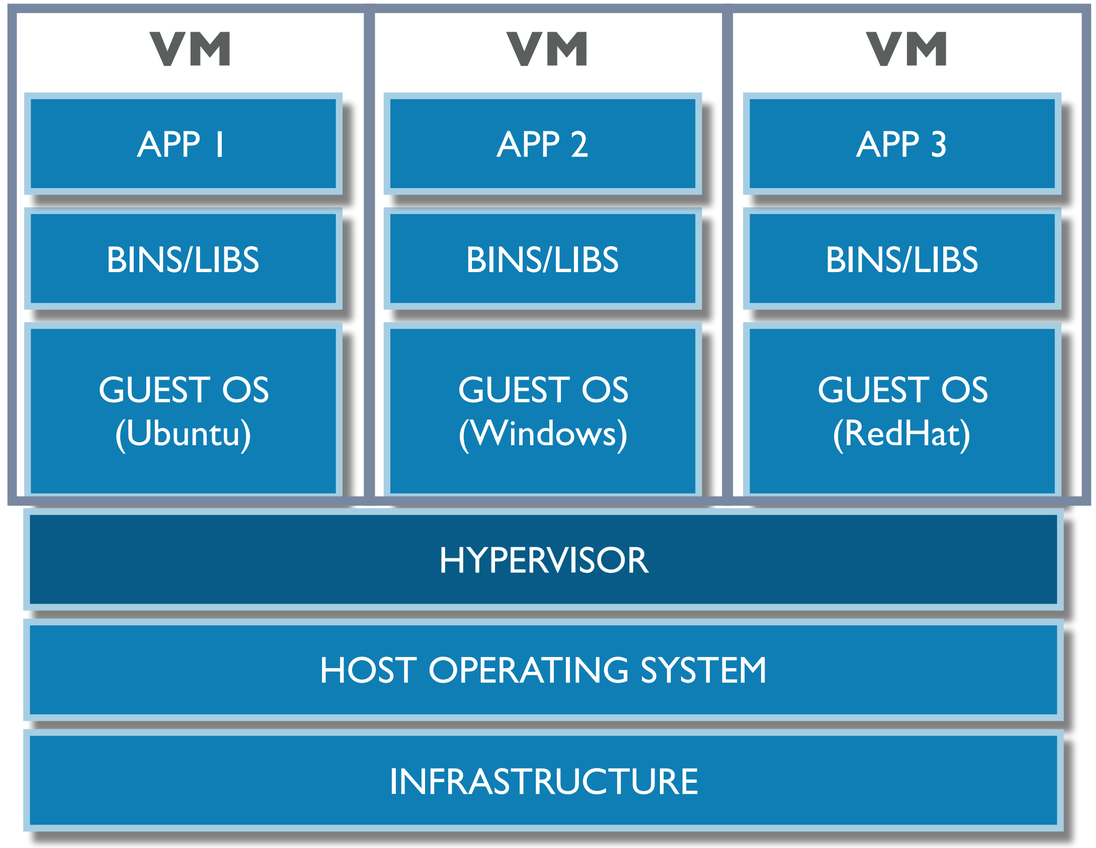
Virtual Machines

- **Virtual Machines** (VM) are created by emulating computer hardware in software
- The emulation is provided by software called a **Hypervisor**
- Each **Guest OS** thinks it's talking to dedicated computer hardware but it is really talking to the hypervisor that is sharing a much larger system



Virtual Machines

- **Virtual Machines** (VM) are created by emulating computer hardware in software
- The emulation is provided by software called a **Hypervisor**
- Each **Guest OS** thinks it's talking to dedicated computer hardware but it is really talking to the hypervisor that is sharing a much larger system



What is VirtualBox?

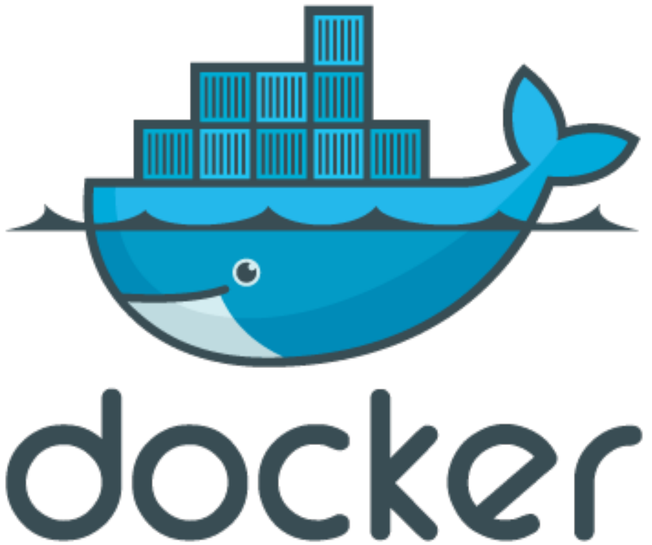
- VirtualBox is a free Hypervisor that runs on OS X, Windows, and Linux
- Similar to VMware Workstation on a PC, or VMware Fusions and Parallels Desktop on a Mac
- Allows you to run your code in a Virtual Machine

What is Vagrant?

- Vagrant is a developers tool for creating lightweight, reproducible and portable virtual environments via command-line
- It supports VirtualBox, VMware, SoftLayer, Amazon AWS and Digital Ocean
- It supports configuration management utilities like Puppet, Chef, etc.
- It supports Docker natively which makes is easy to use Docker containers in VMs

What is Docker?

- Docker packages applications into Containers that include all of their dependencies, but share the kernel with other containers
- Containers run as an isolated process in userspace on the host operating system
- Containers are not tied to any specific infrastructure – Docker containers run on any computer, on any infrastructure and in any cloud.
- This guarantees that it will always run the same, regardless of the environment it is running in.



Baked vs. Fried

- Do you Bake software into the image?
- Or do you Fry it up fresh when you need it?
- I like to Fry!

Today's Focus

- Installing Vagrant and VirtualBox
- Create a simply Vagrantfile that spins up a Web Server in a VM
- Create a more complex Vagrantfile that leverages Docker

Hands-On

“live session”

Some Assembly Required

- Tools you will need to complete this lab:
  - Computer running OS X, Linux, or Windows*
  - Internet Access to download boxes
  - Text Editor (…i like Atom)
  - GitHub Account
  - PC users must have "*VT-x/AMD-V hardware acceleration*" turned on in your BIOS for VirtualBox to work.

* Windows users may need an ssh client

Installing VirtualBox

https://www.virtualbox.org/wiki/Downloads

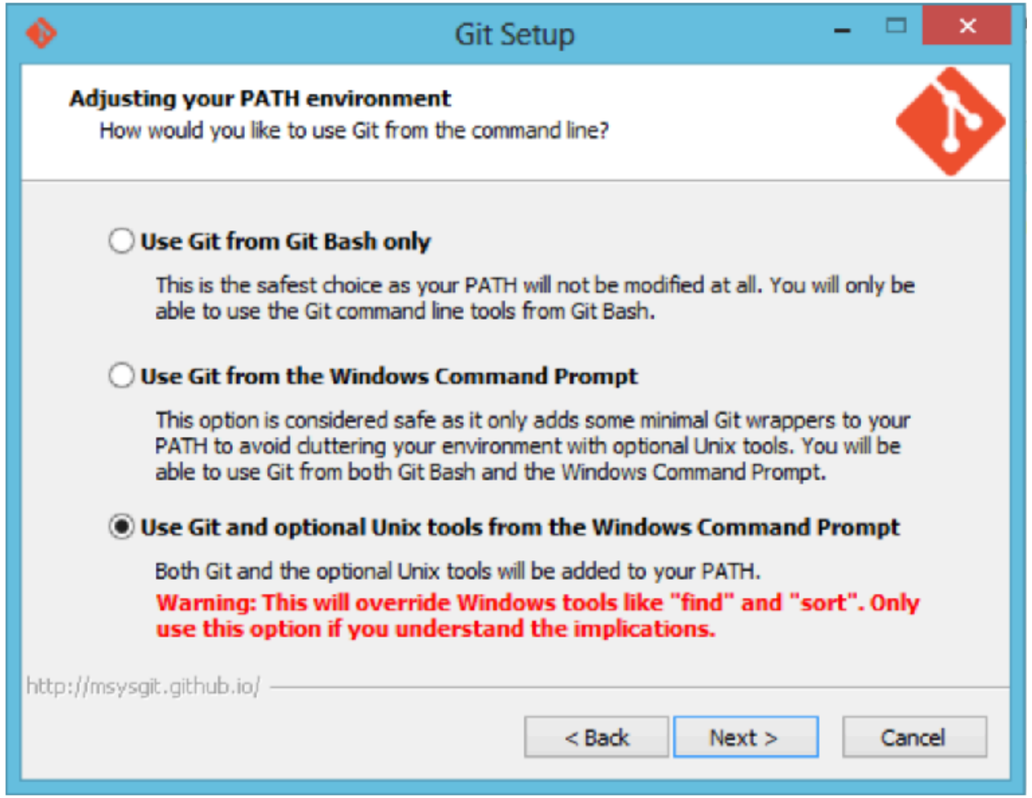
Installing Vagrant

https://www.vagrantup.com/downloads.html

Windows SSH

- Vagrant requires an SSH client which Windows doesn't have by default
- The easiest way to solve this is to install git with the optional unix tools
- Here is a tutorial:

http://tech.osteel.me/posts/2015/01/25/how-to-use-vagrant-on-windows.html



Create Your First Vagrant VM

- Create a folder to work in:

```
$ mkdir lab-vagrant-demo
$ cd lab-vagrant-demo
```

- Go from Zero to running Ubuntu Trusty 64 with these 2 commands:

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

- That's it! 😀

What Did "`init`" do?

- It created a default `Vagrantfile` that you can customize
- The `Vagrantfile` controls how the VM will be provisioned
- By editing the `Vagrantfile` we can change the VM's behavior
- The `Vagrantfile` will be what you check-in to GitHub

Default Vagrantfile

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

  config.vm.box = "ubuntu/trusty64"

  # config.vm.box_check_update = false

  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # config.vm.network "private_network", ip: "192.168.33.10"

  # config.vm.network "public_network"

  # config.vm.synced_folder "../data", "/vagrant_data"

  # config.vm.provider "virtualbox" do |vb|
  #   vb.gui = true
  #   vb.memory = "1024"
  # end

  # config.push.define "atlas" do |push|
  #   push.app = "YOUR_ATLAS_USERNAME/YOUR_APPLICATION_NAME"
  # end

  # config.vm.provision "shell", inline: <<-SHELL
  #   sudo apt-get update
  #   sudo apt-get install -y apache2
  # SHELL
end
```

Default Vagrantfile

This is a Ruby file that uses Ruby syntax

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

  config.vm.box = "ubuntu/trusty64"

  # config.vm.box_check_update = false

  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # config.vm.network "private_network", ip: "192.168.33.10"

  # config.vm.network "public_network"

  # config.vm.synced_folder "../data", "/vagrant_data"

  # config.vm.provider "virtualbox" do |vb|
  #   vb.gui = true
  #   vb.memory = "1024"
  # end

  # config.push.define "atlas" do |push|
  #   push.app = "YOUR_ATLAS_USERNAME/YOUR_APPLICATION_NAME"
  # end

  # config.vm.provision "shell", inline: <<-SHELL
  #   sudo apt-get update
  #   sudo apt-get install -y apache2
  # SHELL
end
```

Default Vagrantfile

This is a Ruby file that uses Ruby syntax

Selects the box to use as a base

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

  # config.vm.box_check_update = false

  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # config.vm.network "private_network", ip: "192.168.33.10"

  # config.vm.network "public_network"

  # config.vm.synced_folder "../data", "/vagrant_data"

  # config.vm.provider "virtualbox" do |vb|
  #   vb.gui = true
  #   vb.memory = "1024"
  # end

  # config.push.define "atlas" do |push|
  #   push.app = "YOUR_ATLAS_USERNAME/YOUR_APPLICATION_NAME"
  # end

  # config.vm.provision "shell", inline: <<-SHELL
  #   sudo apt-get update
  #   sudo apt-get install -y apache2
  # SHELL
end
```

Default Vagrantfile

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

  # config.vm.box_check_update = false

  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # config.vm.network "private_network", ip: "192.168.33.10"

  # config.vm.network "public_network"

  # config.vm.synced_folder "../data", "/vagrant_data"

  # config.vm.provider "virtualbox" do |vb|
  #   vb.gui = true
  #   vb.memory = "1024"
  # end

  # config.push.define "atlas" do |push|
  #   push.app = "YOUR_ATLAS_USERNAME/YOUR_APPLICATION_NAME"
  # end

  # config.vm.provision "shell", inline: <<-SHELL
  #   sudo apt-get update
  #   sudo apt-get install -y apache2
  # SHELL
end
```

This is a Ruby file that uses Ruby syntax

Selects the box to use as a base

Everything else is commented out

What Did "`up`" Do?

- Vagrant downloaded the ubuntu/trusty64 box from atlas.hashicorp.com
- VirtualBox was called to create a VM from that box
- Installed VirtualBox tools and setup the network and shared your current folder as /vagrant
- Ran provisioning scripts to prepare the VM (more on that later)
- Set up SSH keys to logon to the VM

Initial Vagrant Run

```
iotia:lab-vagrant-demo rofrano$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> 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: lab-vagrant-demo_default_1466018224850_26496
==> default: Clearing any previously set forwarded ports...
==> default: Couldn't find Cheffile at ./Cheffile.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> 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: Warning: Remote connection disconnect. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
```

Initial Vagrant Run

```
iotia:lab-vagrant-demo rofrano$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> 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: lab-vagrant-demo_default_1466018224850_26496
==> default: Clearing any previously set forwarded ports...
==> default: Couldn't find Cheffile at ./Cheffile.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> 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: Warning: Remote connection disconnect. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
```

Creates the VM

Initial Vagrant Run

```
iotia:lab-vagrant-demo rofrano$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> 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: lab-vagrant-demo_default_1466018224850_26496
==> default: Clearing any previously set forwarded ports...
==> default: Couldn't find Cheffile at ./Cheffile.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> 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: Warning: Remote connection disconnect. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
```

Creates the VM

Setup Network/Port Forwarding

Initial Vagrant Run

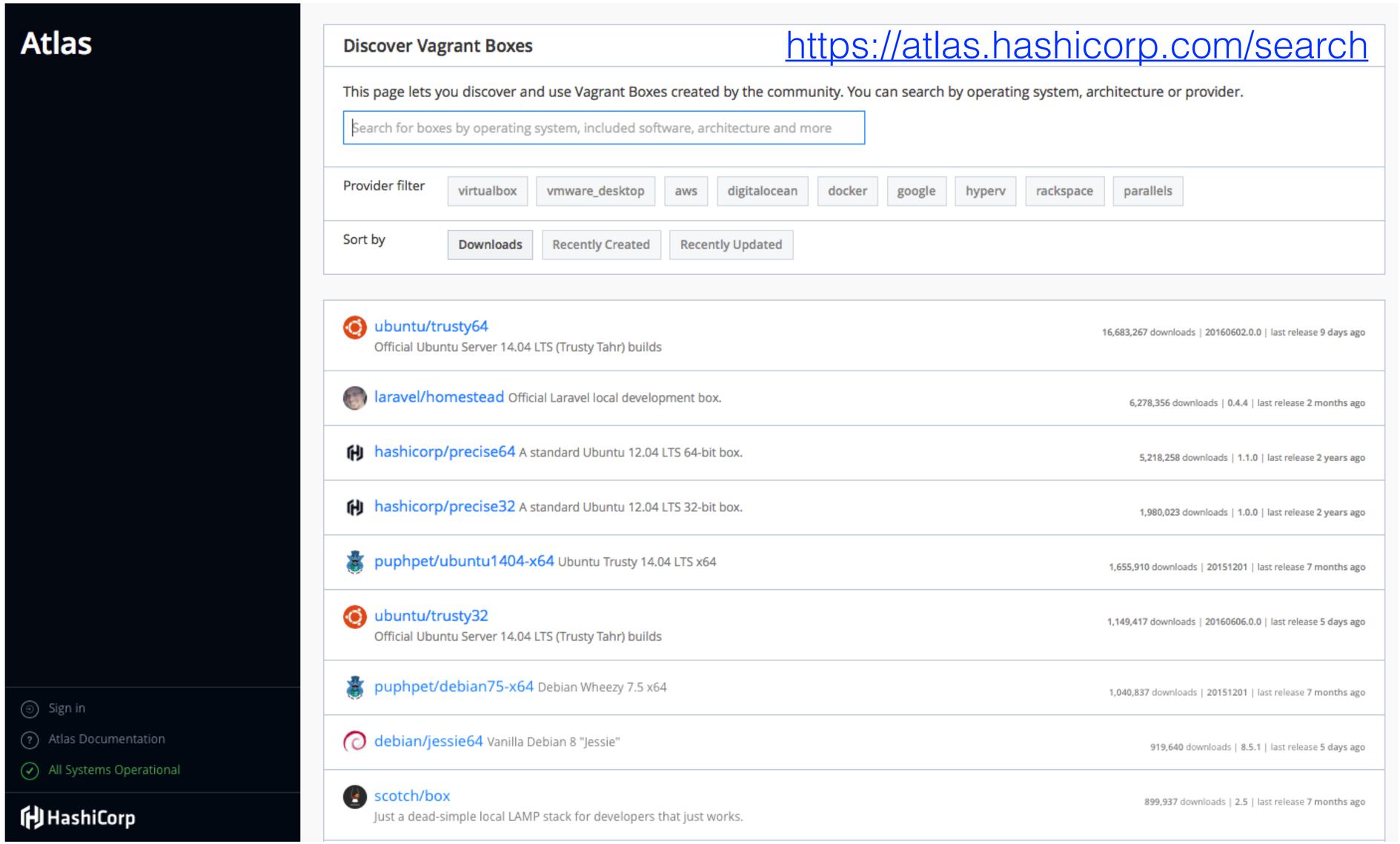
```
iotia:lab-vagrant-demo rofrano$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/trusty64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/trusty64' is up to date...
==> default: Setting the name of the VM: lab-vagrant-demo_default_1466018224850_26496
==> default: Clearing any previously set forwarded ports...
==> default: Couldn't find Cheffile at ./Cheffile.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> 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: Warning: Remote connection disconnect. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
```

Creates the VM

Setup Network/Port Forwarding

Setup SSH keys

Where did Ubuntu/Trusty64 Come From?



What Other Boxes Are There?

http://www.vagrantbox.es/

Vagrantbox.es

Fork me on GitHub

Vagrant is an amazing tool for managing virtual machines via a simple to use command line interface. With a simple `vagrant up` you can be working in a clean environment based on a standard template.

These standard templates are called base boxes, and this website is simply a list of boxes people have been nice enough to make publicly available.

Suggest a Box

Do you know of another base box? Send a pull request and we'll add it to the list below.

Available Boxes

To use the available boxes just replace {title} and {url} with the information in the table below.

```
$ vagrant box add {title} {url}
$ vagrant init {title}
$ vagrant up
```

Search:

Name	Provider	URL	Size (MB)
Virtuozzo 7 x64 (Guest Additions 4.3.26)	VirtualBox	Copy https://atlas.hashicorp.com/OpenVZ/boxes/Virtuozzo-7b2	912
Debian Jessie 8.1.0 Release x64 (Minimal, Guest Additions 4.3.26)	VirtualBox	Copy https://atlas.hashicorp.com/ARTACK/boxes/debian-jessie	692
CentOS 7.0 x64 (Minimal, VirtualBox Guest Additions 4.3.28, Puppet 3.8.1 - see here for more infos)	VirtualBox	Copy https://github.com/tommy-muehle/puppet-vagrant-boxes/releases/download/1.1.0/centos-7.0-x86_64.box	475
CentOS 6.6 x64 (Minimal, VirtualBox Guest Additions, Puppet 3.7.5 - see here for more infos)	VirtualBox	Copy https://github.com/tommy-muehle/puppet-vagrant-boxes/releases/download/1.0.0/centos-6.6-x86_64.box	348
CentOS 7 x64 (Minimal, Shrinked, Guest Additions 4.3.26) (Monthly updates)	VirtualBox	Copy https://github.com/holms/vagrant-centos7-box/releases/download/7.1.1503.001/CentOS-7.1.1503-x86_64-netboot.box	437
CentOS 7.1 x64 (Minimal, Puppet 4.2.3, Guest Additions 4.3.30)[notes]	VirtualBox	Copy https://github.com/CommanderK5/packer-centos-template/releases/download/0.7.1/vagrant-centos-7.1.box	576
CentOS 6.7 x64 (Minimal, Puppet 4.2.3, Guest Additions 4.3.30)[notes]	VirtualBox	Copy https://github.com/CommanderK5/packer-centos-template/releases/download/0.6.7/vagrant-centos-6.7.box	577
Debian Jessie 8.0 Release x64 (Minimal, Shrinked, Guest Additions 4.3.26)	VirtualBox	Copy https://github.com/holms/vagrant-jessie-box/releases/download/Jessie-v0.1/Debian-jessie-amd64-netboot.box	437
Debian 8.1 x64 (LXC, Puppet 3.7.2, Guest Additions 4.3.28)	VirtualBox	Copy https://dl.dropboxusercontent.com/u/3523744/boxes/debian-8.1-amd64-lxc-puppet/debian-8.1-lxc-puppet.box	594
Debian Jessie 8.0 RC2 64-bit (Mininimal, Shrinked + Guest Additions 4.3.24 + Puppet 3.7.2)	VirtualBox	Copy http://static.gender-api.com/debian-8-jessie-rc2-x64-slim.box	328

What Other Boxes Are There?

http://www.vagrantbox.es/

Vagrantbox.es

Vagrant is an amazing tool for managing virtual machines via a simple to use command line interface. With a simple `vagrant up` you can be working in a clean environment based on a standard template.

These standard templates are called base boxes, and this website is simply a list of boxes people have been nice enough to make publicly available.

Suggest a Box

Do you know of another base box? Send a pull request and we'll add it to the list below.

Available Boxes

To use the available boxes just replace {title} and {url} with the information in th

```
$ vagrant box add {title} {url}
$ vagrant init {title}
$ vagrant up
```

This is one way to add these boxes

Search:

Name	Provider	URL	Size (MB)
Virtuozzo 7 x64 (Guest Additions 4.3.26)	VirtualBox	Copy https://atlas.hashicorp.com/OpenVZ/boxes/Virtuozzo-7b2	912
Debian Jessie 8.1.0 Release x64 (Minimal, Guest Additions 4.3.26)	VirtualBox	Copy https://atlas.hashicorp.com/ARTACK/boxes/debian-jessie	692
CentOS 7.0 x64 (Minimal, VirtualBox Guest Additions 4.3.28, Puppet 3.8.1 - see here for more infos)	VirtualBox	Copy https://github.com/tommy-muehle/puppet-vagrant-boxes/releases/download/1.1.0/centos-7.0-x86_64.box	475
CentOS 6.6 x64 (Minimal, VirtualBox Guest Additions, Puppet 3.7.5 - see here for more infos)	VirtualBox	Copy https://github.com/tommy-muehle/puppet-vagrant-boxes/releases/download/1.0.0/centos-6.6-x86_64.box	348
CentOS 7 x64 (Minimal, Shrinked, Guest Additions 4.3.26) (Monthly updates)	VirtualBox	Copy https://github.com/holms/vagrant-centos7-box/releases/download/7.1.1503.001/CentOS-7.1.1503-x86_64-netboot.box	437
CentOS 7.1 x64 (Minimal, Puppet 4.2.3, Guest Additions 4.3.30)[notes]	VirtualBox	Copy https://github.com/CommanderK5/packer-centos-template/releases/download/0.7.1/vagrant-centos-7.1.box	576
CentOS 6.7 x64 (Minimal, Puppet 4.2.3, Guest Additions 4.3.30)[notes]	VirtualBox	Copy https://github.com/CommanderK5/packer-centos-template/releases/download/0.6.7/vagrant-centos-6.7.box	577
Debian Jessie 8.0 Release x64 (Minimal, Shrinked, Guest Additions 4.3.26)	VirtualBox	Copy https://github.com/holms/vagrant-jessie-box/releases/download/Jessie-v0.1/Debian-jessie-amd64-netboot.box	437
Debian 8.1 x64 (LXC, Puppet 3.7.2, Guest Additions 4.3.28)	VirtualBox	Copy https://dl.dropboxusercontent.com/u/3523744/boxes/debian-8.1-amd64-lxc-puppet/debian-8.1-lxc-puppet.box	594
Debian Jessie 8.0 RC2 64-bit (Mininimal, Shrinked + Guest Additions 4.3.24 + Puppet 3.7.2)	VirtualBox	Copy http://static.gender-api.com/debian-8-jessie-rc2-x64-slim.box	328

How To Use A Different Box?

- To use a different box, specify a name in `config.vm.box` and the url in `config.vm.box_url`

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

 config.vm.box = "centos64-x86_64"
 config.vm.box_url = "https://github.com/2creatives/vagrant-centos/releases/
download/v6.4.2/centos64-x86_64-20140116.box"

end
```

Accessing Your VM

- Use SSH client to access your VM

```
$ vagrant ssh
```

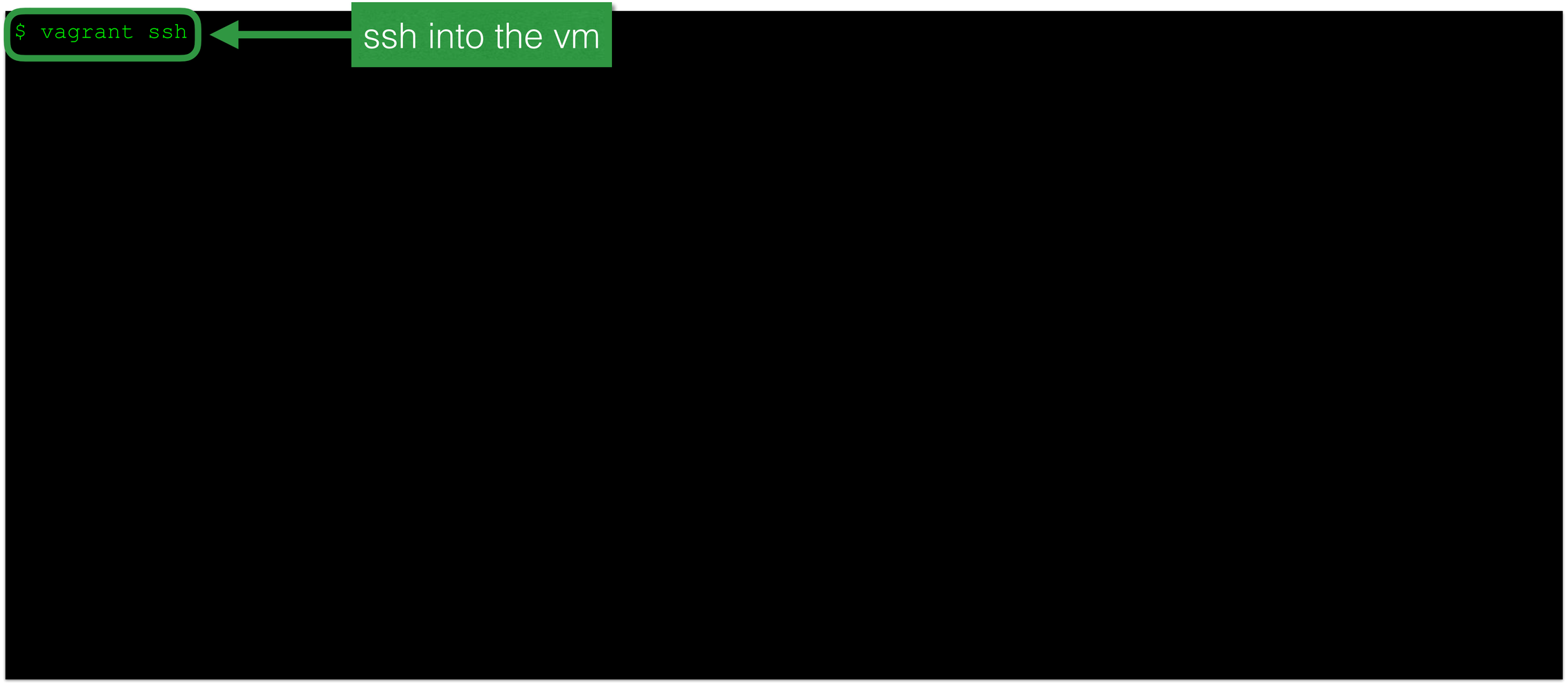
- `/vagrant` folder holds your current directory on the host OS

```
$ cd /vagrant
```

Vagrant SSH

```
$ vagrant ssh
```

Vagrant SSH



Vagrant SSH

```
$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

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

  System information as of Wed Jun 15 19:17:28 UTC 2016

  System load:  0.91              Processes:           81
  Usage of /:   3.5% of 39.34GB   Users logged in:     0
  Memory usage: 25%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  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

0 packages can be updated.
0 updates are security updates.

$
```

Vagrant SSH

```
$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

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

  System information as of Wed Jun 15 19:17:28 UTC 2016

  System load:  0.91              Processes:           81
  Usage of /:   3.5% of 39.34GB   Users logged in:     0
  Memory usage: 25%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  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

0 packages can be updated.
0 updates are security updates.

$ cd /vagrant
$ ls -l
```

Vagrant SSH

```
$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

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

  System information as of Wed Jun 15 19:17:28 UTC 2016

  System load:  0.91              Processes:           81
  Usage of /:   3.5% of 39.34GB   Users logged in:     0
  Memory usage: 25%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  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

0 packages can be updated.
0 updates are security updat

$ cd /vagrant
$ ls -l
```

access to local filesystem via the `/vagrant` folder

Vagrant SSH

```
$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

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

  System information as of Wed Jun 15 19:17:28 UTC 2016

  System load:  0.91              Processes:           81
  Usage of /:   3.5% of 39.34GB   Users logged in:     0
  Memory usage: 25%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  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

0 packages can be updated.
0 updates are security updates.

$ cd /vagrant
$ ls -l
total 4
-rw-r--r-- 1 vagrant vagrant 1928 Jun 15 19:12 Vagrantfile
$
```

Vagrant SSH

```
$ vagrant ssh
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

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

  System information as of Wed Jun 15 19:17:28 UTC 2016

  System load:  0.91              Processes:           81
  Usage of /:   3.5% of 39.34GB   Users logged in:     0
  Memory usage: 25%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  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

0 packages can be updated.
0 updates are security updates.

$ cd /vagrant
$ ls -l
total 4
-rw-r--r-- 1 vagrant vagrant 1928 Jun 15 19:12 Vagrantfile
$
```

This is your `Vagrantfile` from inside the VM

Getting In and Out of your VM

- Just like any remote server that you ssh into, you use exit to leave

```
$ exit
iotia:lab-vagrant rofrano$
```

- To get back in you just ssh again

```
iotia:lab-vagrant rofrano$ vagrant ssh
$
```

Vagrant file sharing

- Vagrant is sharing the host folder it was invoked from within the vm as `/vagrant`
- This means that you and edit files with your native Mac or Windows editor
- …and still be able to run the code on your Linux VM

Install Apache

- We can install software using Chef, Ansible, Puppet, Salt, Shell, etc.
- Let's use the "shell" provisioner to add apache2 *(just uncomment the code)*

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

  config.vm.box = "ubuntu/trusty64"

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
  SHELL

end
```

Install Apache

- We can install software using Chef, Ansible, Puppet, Salt, Shell, etc.
- Let's use the "shell" provisioner to add apache2 *(just uncomment the code)*

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

  config.vm.box = "ubuntu/trusty64"

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
  SHELL

end
```

Uncomment this code
from the bottom of the `Vagrantfile`
It uses shell syntax for the
OS (Ubuntu in this case)

Re-provision the VM after Updates

- After making changes to a `Vagrantfile` in any of the provisioning sections, you must re-provision the VM

```
$ exit

$ vagrant reload --provision
```

Re-provision the VM after Updates

- After making changes to a `Vagrantfile` in any of the provisioning sections, you must re-provision the VM

```
$ exit

$ vagrant reload --provision
```

Reload alone will only reboot the VM and run non-provisioning blocks but the --provision parameter forces the provision blocks to be re-evaluated and run

Testing the Web Server

- Use SSH client to access your VM and CURL to access Apache

```
$ vagrant ssh
$ curl 127.0.0.1
```

- You should see HTML returned
- But if you access it from your browser it will fail! Why?

Let's Forward The Web Port

- We must forward the ports that we want to access from outside the VM. Let's add a private IP Address for direct access too:

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

  config.vm.box = "ubuntu/trusty64"

  # accessing "localhost:8080" will access port 80 on the guest machine.
  config.vm.network "forwarded_port", guest: 80, host: 8080

  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
  SHELL

end
```

Let's Forward The Web Port

- We must forward the ports that we want to access from outside the VM. Let's add a private IP Address for direct access too:

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

  config.vm.box = "ubuntu/trusty64"

  # accessing "localhost:8080" will access port 80 on the guest machine.
  config.vm.network "forwarded_port", guest: 80, host: 8080

  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
  SHELL

end
```

We should use ports higher than 1024 if we are not root

Reload the VM after Updates

- Since we didn't change any of the provisioning blocks in the `Vagrantfile`, we can just reload the VM and the port forwarding will be picked up

```
$ exit
$ vagrant reload
```

New Vagrant Run

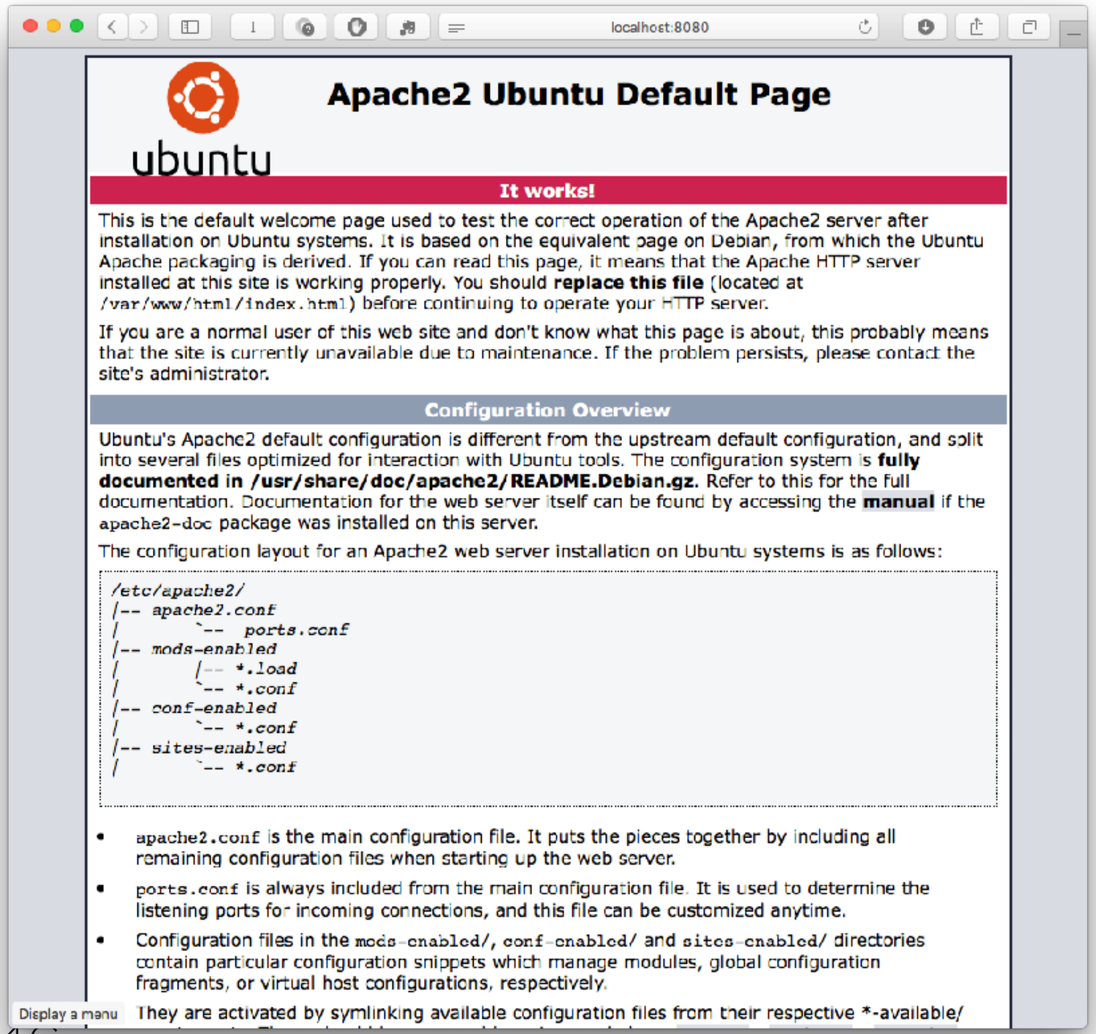
```
iotia:lab-vagrant-demo rofrano$ vagrant reload
==> default: Attempting graceful shutdown of VM...
==> default: Checking if box 'ubuntu/trusty64' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Couldn't find Cheffile at ./Cheffile.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 80 (guest) => 8080 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> 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: Warning: Remote connection disconnect. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
==> default: Machine booted and ready!
GuestAdditions 5.0.20 running --- OK.
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => /Users/rofrano/github/Demo/lab-vagrant-demo
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
iotia:lab-vagrant-demo rofrano$
```

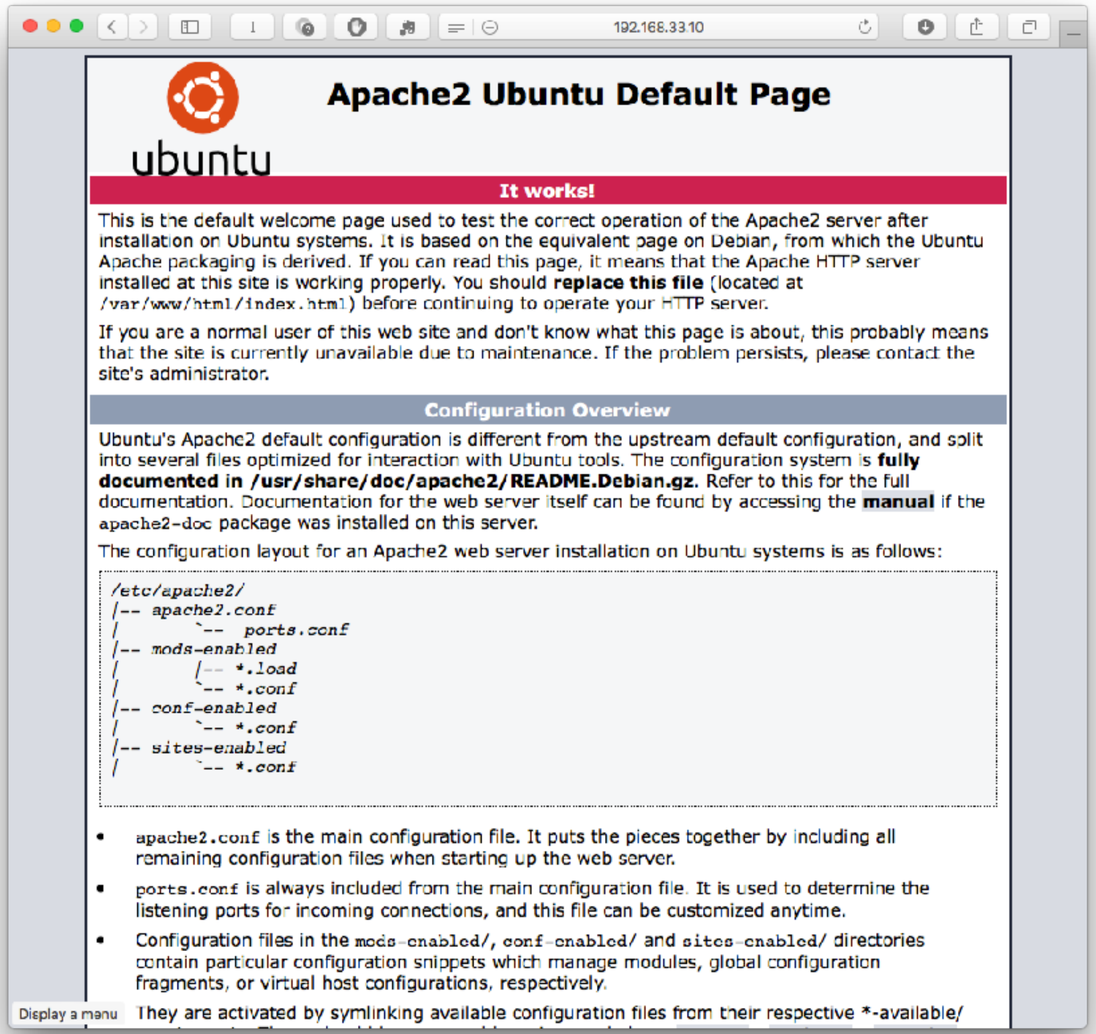
New Vagrant Run

```
iotia:lab-vagrant-demo rofrano$ vagrant reload
==> default: Attempting graceful shutdown of VM...
==> default: Checking if box 'ubuntu/trusty64' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Couldn't find Cheffile at ./Cheffile.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 80 (guest) => 8080 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> 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: Warning: Remote connection disconnect. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
==> default: Machine booted and ready!
GuestAdditions 5.0.20 running --- OK.
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => /Users/rofrano/github/Demo/lab-vagrant-demo
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
iotia:lab-vagrant-demo rofrano$
```

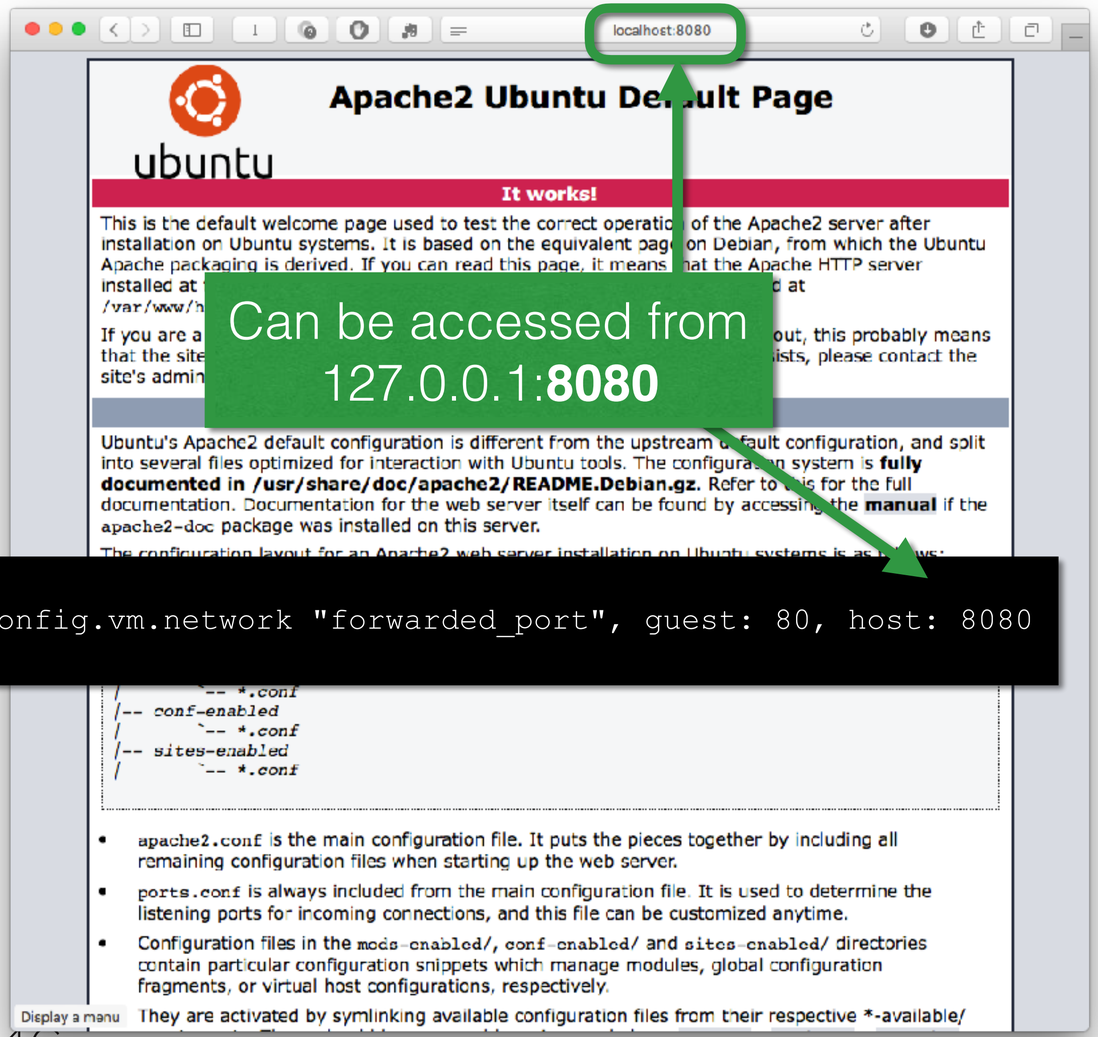
Our port got forwarded from 80 inside the VM to 8080 outside the VM (on the host)

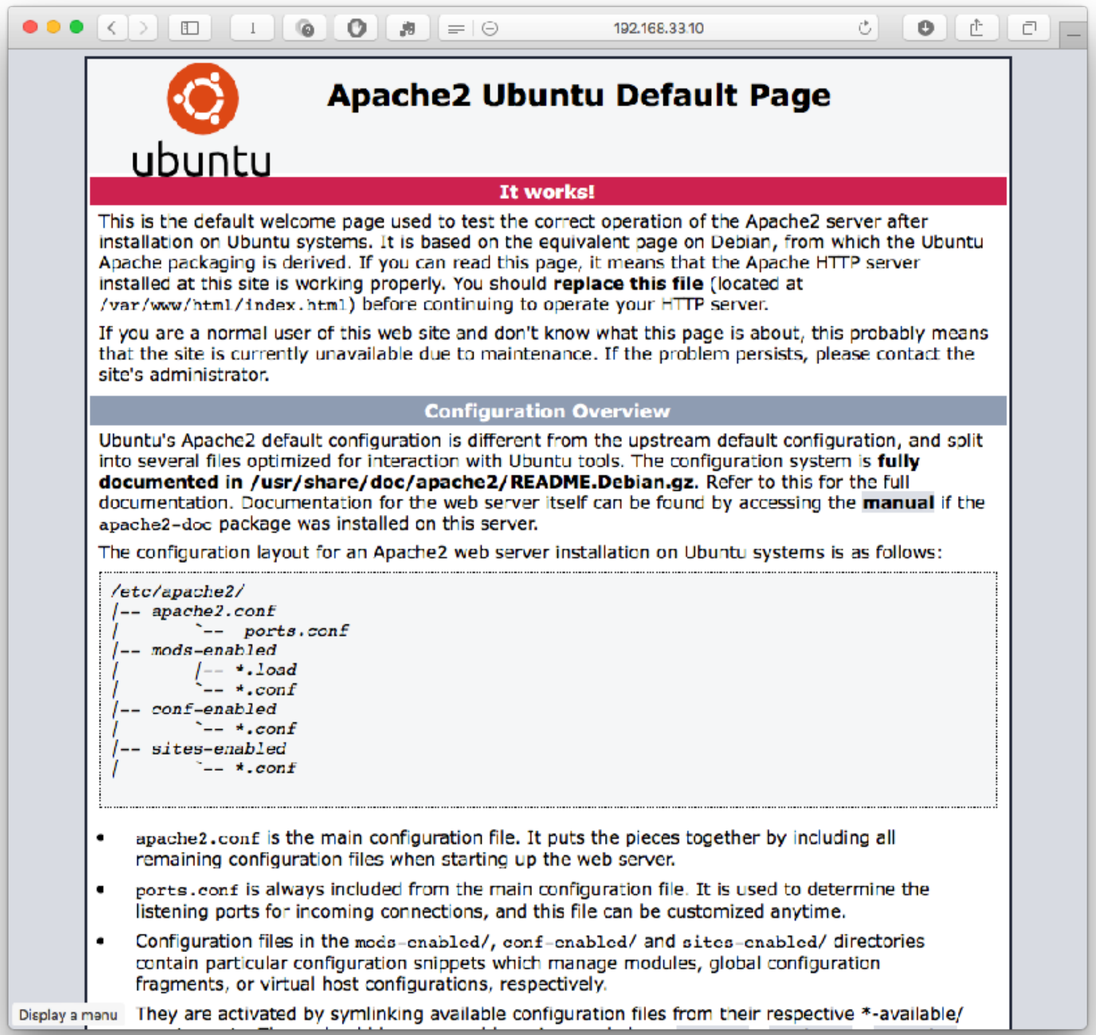
Test It In Your Browser



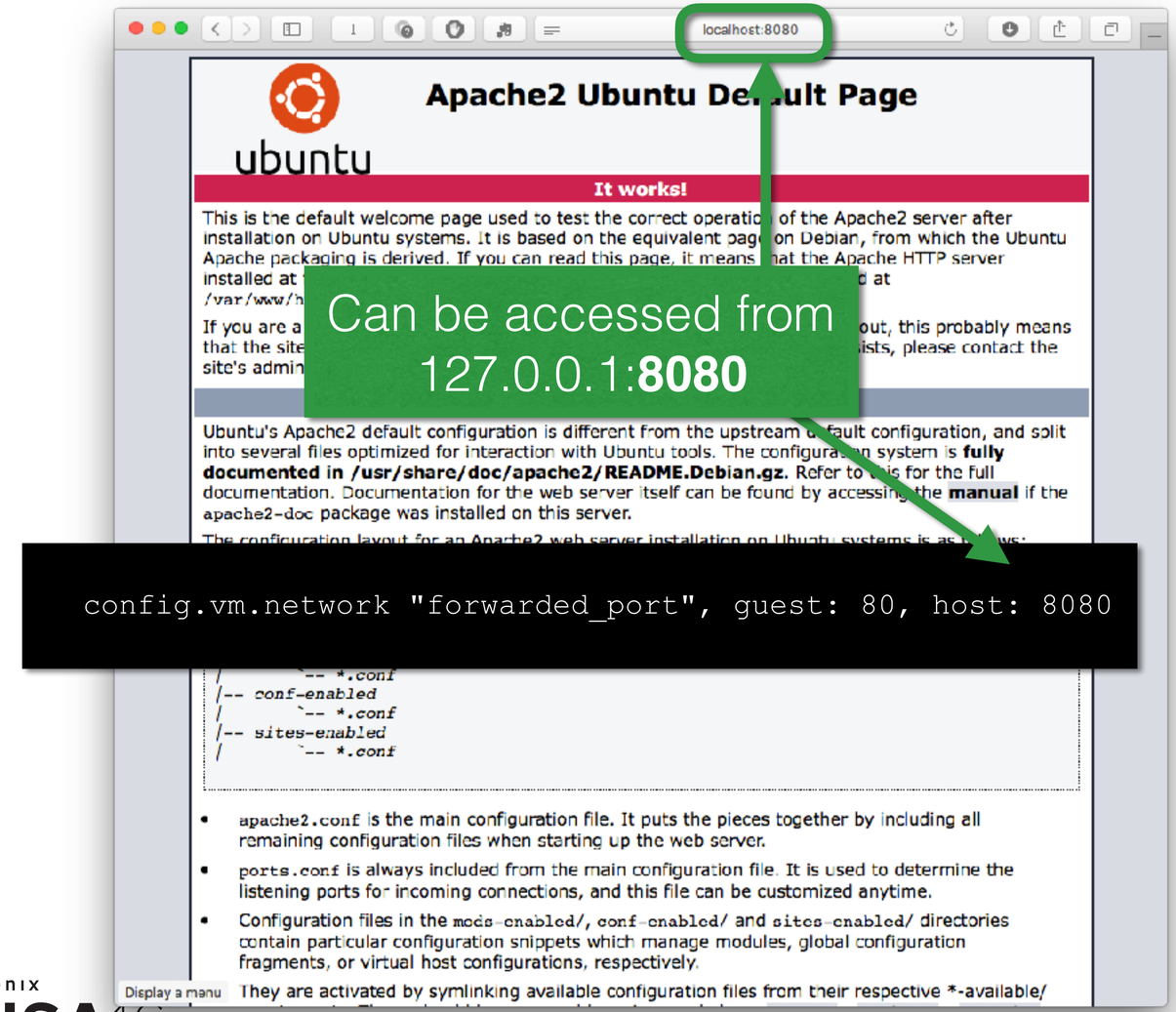


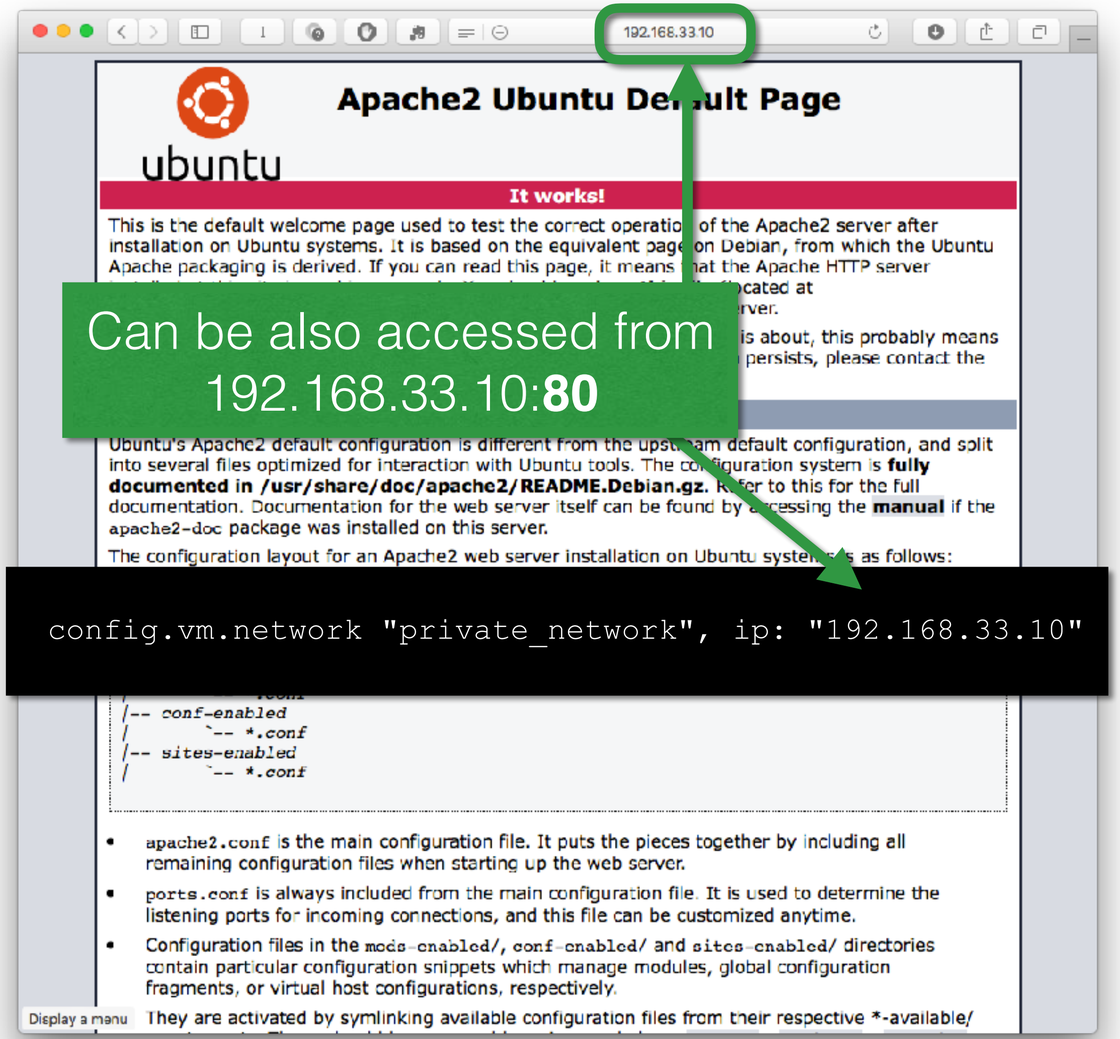
Test It In Your Browser





Test It In Your Browser





More Port Forwarding Examples

```
# Forward RAILS port
config.vm.network :forwarded_port, guest: 3000, host: 3000, auto_correct: true

# Forward Python Flask port
config.vm.network :forwarded_port, guest: 5000, host: 5000, auto_correct: true

# Forward PostgreSQL port
config.vm.network :forwarded_port, guest: 5432, host: 5432, auto_correct: true

# Forward MySQL port
config.vm.network :forwarded_port, guest: 3306, host: 3306, auto_correct: true

# Forward Redis port
config.vm.network :forwarded_port, guest: 6379, host: 6379, auto_correct: true

# Forward RabbitMQ Ports
config.vm.network :forwarded_port, guest: 15672, host: 15672, auto_correct: true
config.vm.network :forwarded_port, guest: 4369,  host: 4369,  auto_correct: true
config.vm.network :forwarded_port, guest: 5672,  host: 5672,  auto_correct: true
```

Adding Your Own Content

- We can link a local folder to the html root to serve up our own content (remember our local folder is shared with the VM!)
- First lets create a new `./html` folder and add `index.html`:

```
$ mkdir html
$ cd html
$ echo '<html><h1>Hello World!</h1></html>' > index.html
```

Link Our HTML Folder

- We can point `/var/www/html` to our host folder `/vagrant/html`

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

  config.vm.box = “ubuntu/trusty64"

  # accessing "localhost:8080" will access port 80 on the guest machine.
  config.vm.network "forwarded_port", guest: 80, host: 8080

  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
    sudo rm -rf /var/www/html
    sudo ln -s /vagrant/html /var/www/html
  SHELL

end
```

Link Our HTML Folder

- We can point `/var/www/html` to our host folder `/vagrant/html`

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

  config.vm.box = “ubuntu/trusty64"

  # accessing "localhost:8080" will access port 80 on the guest machine.
  config.vm.network "forwarded_port", guest: 80, host: 8080

  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
    sudo rm -rf /var/www/html
    sudo ln -s /vagrant/html /var/www/html
  SHELL

end
```

Remove the /var/www/html folder and link to our host’s ./html folder

Add 'rm' and 'ln' Commands

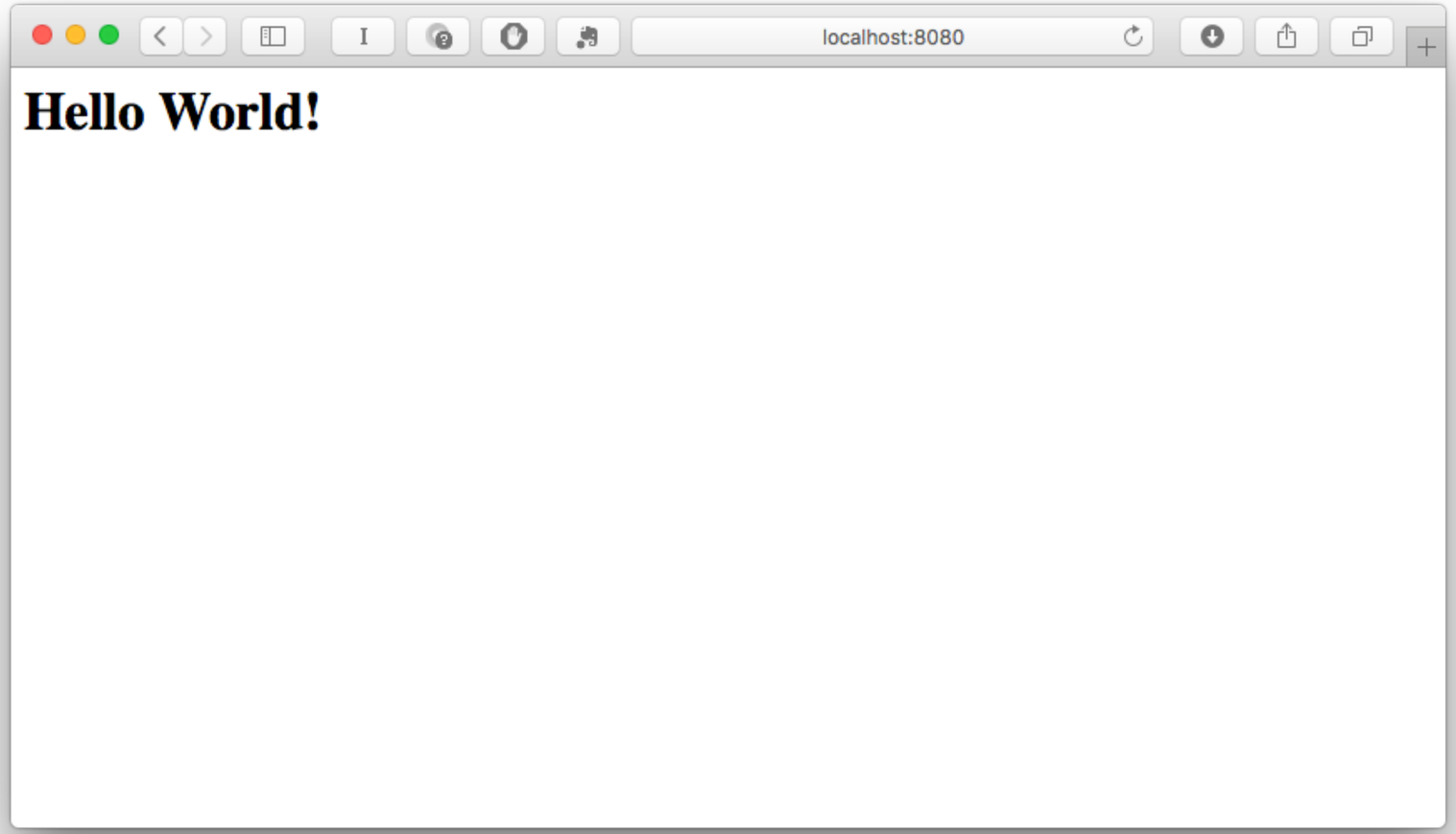
```
config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
    sudo rm -rf /var/www/html
    sudo ln -s /vagrant/html /var/www/html
SHELL
```

Re-provision the VM after Updates

- Since we changed a provision block in the `Vagrantfile` we must reload and provision

```
$ vagrant reload --provision
```

Apache is Serving Our Page



What Just Happened?

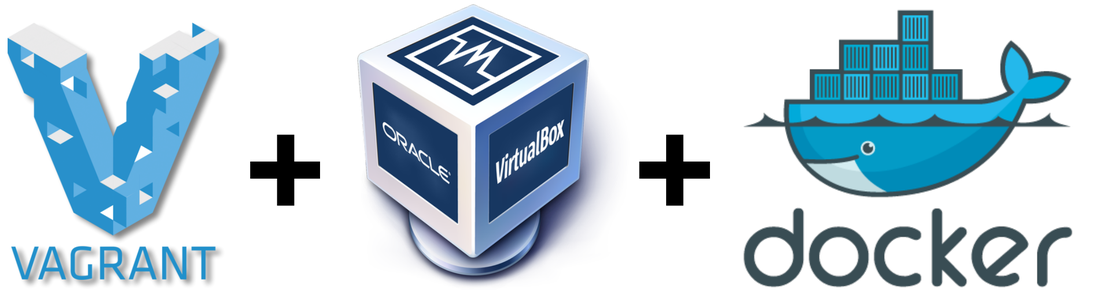
- We are serving up local files from our laptop inside of the VM
- That means we can develop locally with our favorite IDE tools
- And we can run the code in a Linux VM just like a production server!
- Very Cool ! ! ! 😎

Removing a VM Permanently

- When you no longer need a VM you can delete it from your computer.
- We are finished with this one and will create a new one in the next section
- This is accomplished with the `destroy` command

```
$ vagrant destroy

    default: Are you sure you want to destroy the 'default' VM? [y/N]
==> default: Destroying VM and associated drives...
```

Lets Add Docker To The Mix



- We want to create an environment to run a Python / Flask microservice that uses a Redis database to store a hit counter
- We will use Docker to run Redis without having to install anything
- Vagrant supports a Docker provisioner that we can leverage

Clone the Project from GitHub

- You are a new developer to the project and you want to get up and running quickly so that you can start developing:

```
$ git clone git@github.com:rofrano/lab-vagrant.git
$ cd lab-vagrant
$ vagrant up
```

Complete Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 5000, host: 5000
  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y git python-pip python-dev build-essential
    sudo apt-get -y autoremove
    # Install app dependencies
    cd /vagrant
    sudo pip install -r requirements.txt
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
    # Make vi look nice
    echo "colorscheme desert" > ~/.vimrc
  SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

Complete Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 5000, host: 5000
  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y git python-pip python-dev build-essential
    sudo apt-get -y autoremove
    # Install app dependencies
    cd /vagrant
    sudo pip install -r requirements.txt
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
    # Make vi look nice
    echo "colorscheme desert" > ~/.vimrc
  SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

Create an Ubuntu VM and Forward the ports you need

Complete Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 5000, host: 5000
  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y git python-pip python-dev build-essential
    sudo apt-get -y autoremove
    # Install app dependencies
    cd /vagrant
    sudo pip install -r requirements.txt
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
    # Make vi look nice
    echo "colorscheme desert" > ~/.vimrc
  SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

Create an Ubuntu VM and Forward the ports you need

Allocate the VM resources

Complete Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 5000, host: 5000
  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y git python-pip python-dev build-essential
    sudo apt-get -y autoremove
    # Install app dependencies
    cd /vagrant
    sudo pip install -r requirements.txt
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
    # Make vi look nice
    echo "colorscheme desert" > ~/.vimrc
  SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

Create an Ubuntu VM and Forward the ports you need

Allocate the VM resources

Provision the development environment

Complete Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 5000, host: 5000
  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y git python-pip python-dev build-essential
    sudo apt-get -y autoremove
    # Install app dependencies
    cd /vagrant
    sudo pip install -r requirements.txt
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
    # Make vi look nice
    echo "colorscheme desert" > ~/.vimrc
  SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

Create an Ubuntu VM and Forward the ports you need

Allocate the VM resources

Provision the development environment

Add Docker Containers for middleware

Application Code

app.py

```python
import os
import redis
from flask import Flask, jsonify

port = os.getenv('PORT', '5000')
hostname = os.getenv('HOSTNAME', '127.0.0.1')
redis_port = os.getenv('REDIS_PORT', '6379')

app = Flask(__name__)

# Application Routes
@app.route('/')
def index():
    return jsonify(name='Hit Me Service', version='1.0', url='/hits'), 200

@app.route('/hits')
def hits():
    redis_server.incr('hit_counter')
    count = redis_server.get('hit_counter')
    return jsonify(hits=count), 200


if __name__ == "__main__":

    redis_server = redis.Redis(host=hostname, port=int(redis_port))
    app.run(host='0.0.0.0', port=int(port))

```

Application Code

app.py

```python
import os
import redis
from flask import Flask, jsonify

port = os.getenv('PORT', '5000')
hostname = os.getenv('HOSTNAME', '127.0.0.1')
redis_port = os.getenv('REDIS_PORT', '6379')

app = Flask(__name__)

# Application Routes
@app.route('/')
def index():
    return jsonify(name='Hit Me Service', version='1.0', url='/hits'), 200

@app.route('/hits')
def hits():
    redis_server.incr('hit_counter')
    count = redis_server.get('hit_counter')
    return jsonify(hits=count), 200


if __name__ == "__main__":

    redis_server = redis.Redis(host=hostname, port=int(redis_port))
    app.run(host='0.0.0.0', port=int(port))

```

Pull runtime data from the environment

Application Code

app.py

```python
import os
import redis
from flask import Flask, jsonify

port = os.getenv('PORT', '5000')
hostname = os.getenv('HOSTNAME', '127.0.0.1')
redis_port = os.getenv('REDIS_PORT', '6379')

app = Flask(__name__)

# Application Routes
@app.route('/')
def index():
    return jsonify(name='Hit Me Service', version='1.0', url='/hits'), 200

@app.route('/hits')
def hits():
    redis_server.incr('hit_counter')
    count = redis_server.get('hit_counter')
    return jsonify(hits=count), 200


if __name__ == "__main__":

    redis_server = redis.Redis(host=hostname, port=int(redis_port))
    app.run(host='0.0.0.0', port=int(port))

```

Pull runtime data from the environment

Connect to **Redis** and **run** Flask Server

Application Code

app.py

```python
import os
import redis
from flask import Flask, jsonify

port = os.getenv('PORT', '5000')
hostname = os.getenv('HOSTNAME', '127.0.0.1')
redis_port = os.getenv('REDIS_PORT', '6379')

app = Flask(__name__)

# Application Routes
@app.route('/')
def index():
    return jsonify(name='Hit Me Service', version='1.0', url='/hits'), 200

@app.route('/hits')
def hits():
    redis_server.incr('hit_counter')
    count = redis_server.get('hit_counter')
    return jsonify(hits=count), 200


if __name__ == "__main__":

    redis_server = redis.Redis(host=hostname, port=int(redis_port))
    app.run(host='0.0.0.0', port=int(port))

```

Pull runtime data from the environment

/ returns info

Connect to **Redis** and **run** Flask Server

Application Code

app.py

```python
import os
import redis
from flask import Flask, jsonify

port = os.getenv('PORT', '5000')
hostname = os.getenv('HOSTNAME', '127.0.0.1')
redis_port = os.getenv('REDIS_PORT', '6379')

app = Flask(__name__)

# Application Routes
@app.route('/')
def index():
    return jsonify(name='Hit Me Service', version='1.0', url='/hits'), 200

@app.route('/hits')
def hits():
    redis_server.incr('hit_counter')
    count = redis_server.get('hit_counter')
    return jsonify(hits=count), 200

if __name__ == "__main__":

    redis_server = redis.Redis(host=hostname, port=int(redis_port))
    app.run(host='0.0.0.0', port=int(port))
```

Pull runtime data from the environment

`/` returns info

`/hits` calls Redis to increment counter and return it

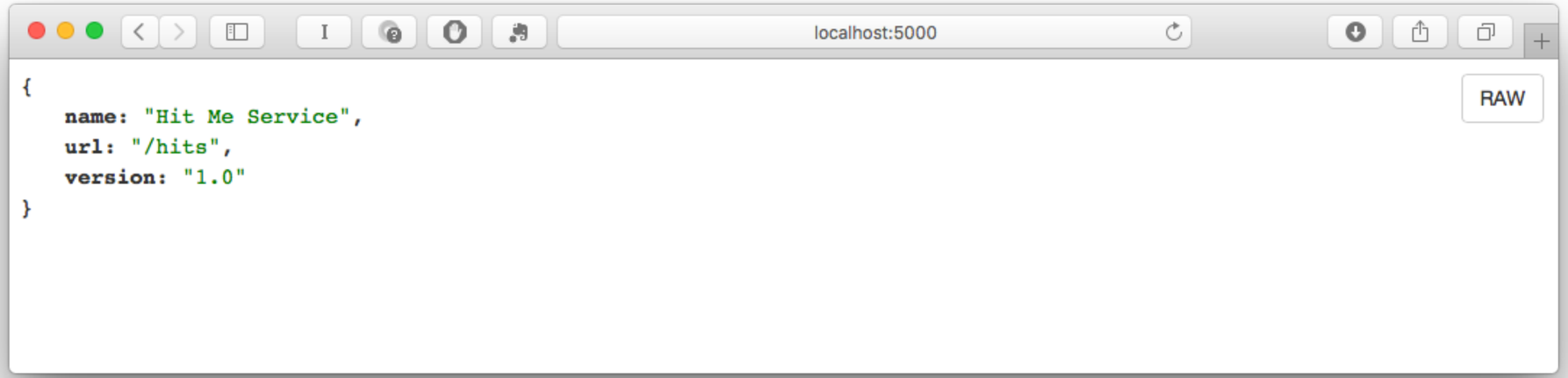
Connect to **Redis** and **run** Flask Server

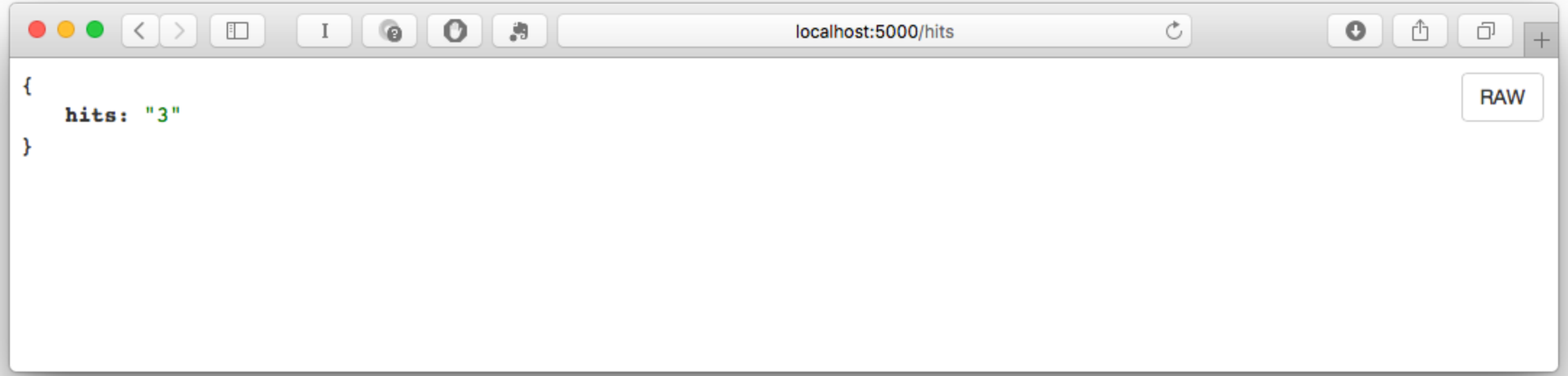
Start the Application

- Once the Vagrant VM has been provisioned we can `ssh` into it and start the Python Flask application

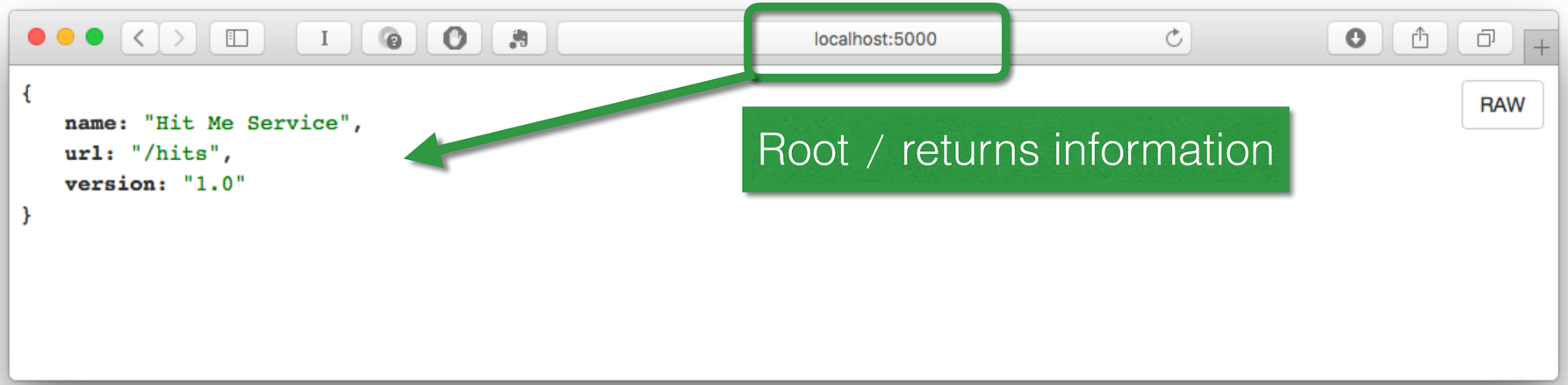
```
$ vagrant ssh
$ cd /vagrant
$ python app.py

* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

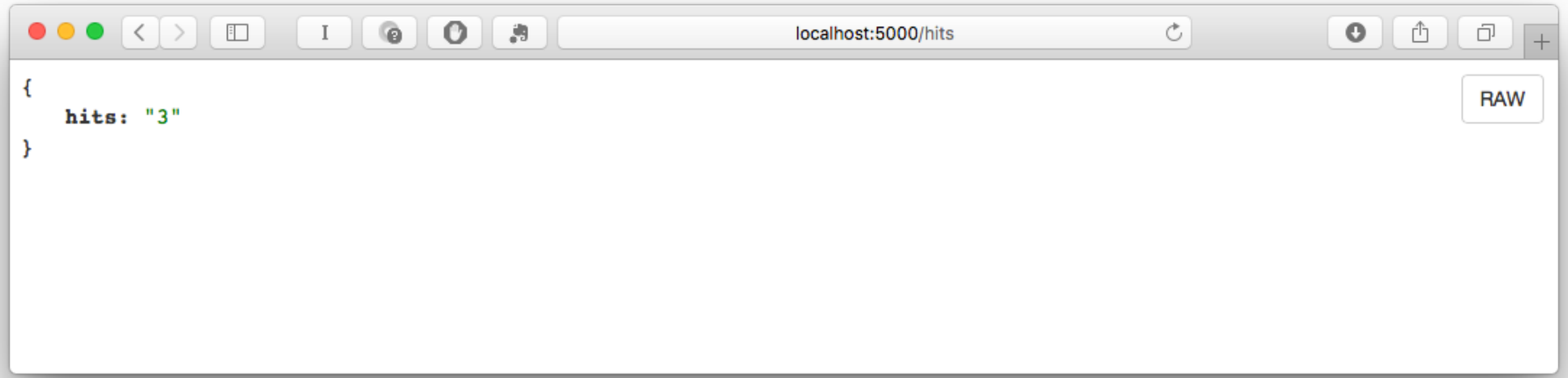
Flask is Serving Our Application



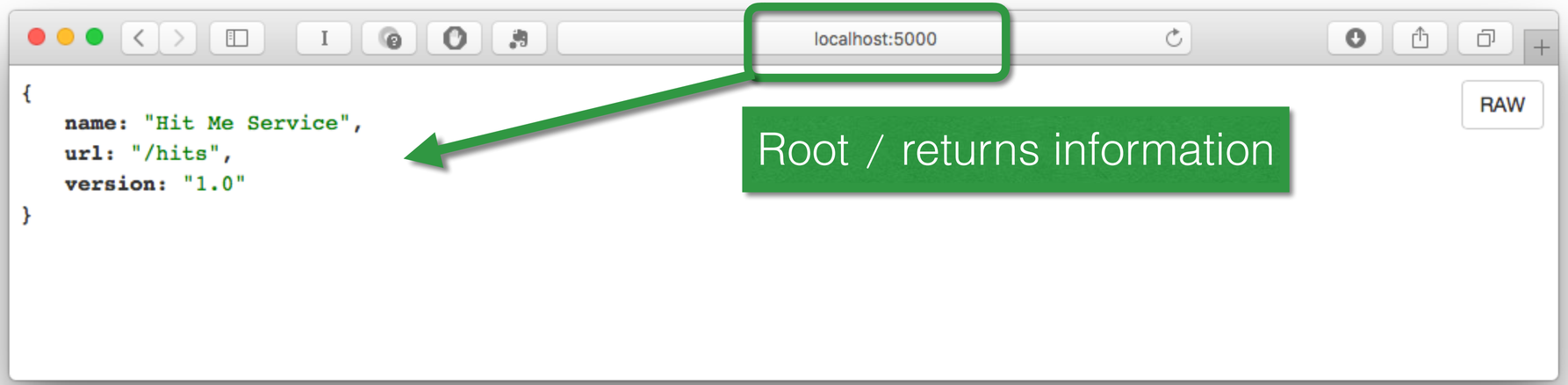


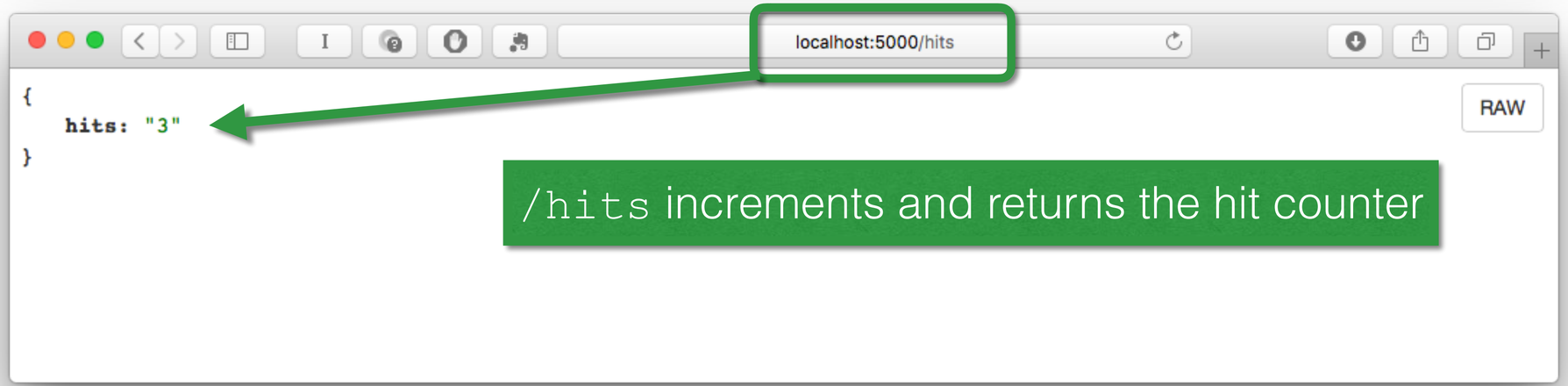
Flask is Serving Our Application





Flask is Serving Our Application





What Just Happened?

- A developer (you) who possibly never worked with Python, Flask, or Redis before
- Just cloned a git repo and provisioned a working development environment
- With one command: `vagrant up`

Vagrant Python Environment

```
# Forward Python Flask port
config.vm.network :forwarded_port, guest: 5000, host: 5000, auto_correct: true

# Setup a Python development environment
config.vm.provision "shell", inline: <<-SHELL
  sudo apt-get update
  sudo apt-get install -y git python-pip python-dev build-essential
  sudo apt-get -y autoremove
  # Install app dependencies
  cd /vagrant
  sudo pip install -r requirements.txt
SHELL
```

Vagrant Ruby on Rails Environment

```
# Forward RAILS port
  config.vm.network :forwarded_port, guest: 3000, host: 3000, auto_correct: true

# Setup a Ruby on Rails development environment
  config.vm.provision :shell, privileged: false, inline: <<-SHELL
    sudo apt-get update
    sudo apt-get -y install git libsqlite3-dev libpq-dev nodejs
    sudo apt-get -y autoremove

    # disable docs during gem install
    echo "gem: --no-document" >> ~/.gemrc
    gem update --no-rdoc --no-ri
    gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
    # use RVM to install Ruby and Rails
    curl -sSL https://get.rvm.io | bash -s stable --rails

    # Add Heroku toolbelt
    sudo wget -O- https://toolbelt.heroku.com/install-ubuntu.sh | sh
  SHELL
```

Working With Docker In The VM

- We can run any Docker commands inside the VM to get the current status of any Docker containers

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis               alpine              8a8bdae38335        7 days ago          29.07 MB
```

Working With Docker In The VM

- We can run any Docker commands inside the VM to get the current status of any Docker containers

List Image Inventory

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis               alpine              8a8bdae38335        7 days ago          29.07 MB
```

Working With Docker In The VM

- We can run any Docker commands inside the VM to get the current status of any Docker containers

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis               alpine              8a8bdae38335        7 days ago          29.07 MB
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
8ef040bad484        redis:alpine        "docker-entrypoint.sh"   25 hours ago        Up 16 minutes       0.0.0.0:6379->6379/tcp   redis
```

Working With Docker In The VM

- We can run any Docker commands inside the VM to get the current status of any Docker containers

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis               alpine              8a8bdae38335        7 days ago          29.07 MB
```

List Running Containers

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
8ef040bad484        redis:alpine        "docker-entrypoint.sh"   25 hours ago        Up 16 minutes       0.0.0.0:6379->6379/tcp   redis
```

Vagrantfile to Docker Mapping

- This is the mapping from the Vagrantfile to the Docker containers:

`Vagrantfile`

```
config.vm.provision "docker" do |d|
   d.pull_images "redis:alpine"
   d.run "redis:alpine",
     args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
 end
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED         STATUS              PORTS                    NAMES
8ef040bad484        redis:alpine        "docker-entrypoint.sh"   25 hours ago    Up 16 minutes       0.0.0.0:6379->6379/tcp   redis
```

Vagrantfile to Docker Mapping

- This is the mapping from the Vagrantfile to the Docker containers:

Vagrantfile

```
config.vm.provision "docker" do |d|
   d.pull_images "redis:alpine"
   d.run "redis:alpine",
     args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
 end
```

Image to run

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
8ef040bad484        redis:alpine        "docker-entrypoint.sh"   25 hours ago        Up 16 minutes       0.0.0.0:6379->6379/tcp   redis
```

Vagrantfile to Docker Mapping

- This is the mapping from the Vagrantfile to the Docker containers:

Vagrantfile

```
config.vm.provision "docker" do |d|
   d.pull_images "redis:alpine"
   d.run "redis:alpine",
     args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
 end
```

Image to run

Port to listen on

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
8ef040bad484        redis:alpine        "docker-entrypoint.sh"   25 hours ago        Up 16 minutes       0.0.0.0:6379->6379/tcp   redis
```

Vagrantfile to Docker Mapping

- This is the mapping from the Vagrantfile to the Docker containers:

Vagrantfile

```
config.vm.provision "docker" do |d|
   d.pull_images "redis:alpine"
   d.run "redis:alpine",
     args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
 end
```

Image to run

Port to listen on

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
8ef040bad484        redis:alpine        "docker-entrypoint.sh"   25 hours ago        Up 16 minutes       0.0.0.0:6379->6379/tcp   redis
```

name of container

What about data persistence?

- Map a Volume in the VM to persist data cross Docker containers

```
Vagrant.configure(2) do |config|
 config.vm.provision "shell", inline: <<-SHELL
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
 SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

What about data persistence?

- Map a Volume in the VM to persist data cross Docker containers

```
Vagrant.configure(2) do |config|
 config.vm.provision "shell", inline: <<-SHELL
    # Prepare Redis data share
    sudo mkdir -p /var/lib/redis/data
    sudo chown vagrant:vagrant /var/lib/redis/data
 SHELL

  # Add Redis docker container
  config.vm.provision "docker" do |d|
    d.pull_images "redis:alpine"
    d.run "redis:alpine",
      args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
  end
end
```

Persist container data in the VM

Data Volumes for PostgreSQL

```
##############################################################
# Add PostgreSQL docker container
##############################################################
config.vm.provision "shell", inline: <<-SHELL
  # Prepare PostgreSQL data share
  sudo mkdir -p /var/lib/postgresql/data
  sudo chown vagrant:vagrant /var/lib/postgresql/data
SHELL

# Add PostgreSQL docker container
config.vm.provision "docker" do |d|
  d.pull_images "postgres"
  d.run "postgres",
    args: "-d --name postgres -p 5432:5432 -v /var/lib/postgresql/data:/var/lib/postgresql/data"
end
```

Data Volumes for PostgreSQL

```
##########################################################
# Add PostgreSQL docker container
##########################################################
config.vm.provision "shell", inline: <<-SHELL
  # Prepare PostgreSQL data share
  sudo mkdir -p /var/lib/postgresql/data
  sudo chown vagrant:vagrant /var/lib/postgresql/data
SHELL

# Add PostgreSQL docker container
config.vm.provision "docker" do |d|
  d.pull_images "postgres"
  d.run "postgres",
    args: "-d --name postgres -p 5432:5432 -v /var/lib/postgresql/data:/var/lib/postgresql/data"
end
```

volume on the host

Data Volumes for PostgreSQL

```
##############################################################
# Add PostgreSQL docker container
##############################################################
config.vm.provision "shell", inline: <<-SHELL
  # Prepare PostgreSQL data share
  sudo mkdir -p /var/lib/postgresql/data
  sudo chown vagrant:vagrant /var/lib/postgresql/data
SHELL

# Add PostgreSQL docker container
config.vm.provision "docker" do |d|
  d.pull_images "postgres"
  d.run "postgres",
    args: "-d --name postgres -p 5432:5432 -v /var/lib/postgresql/data:/var/lib/postgresql/data"
end
```

volume on the host

Volume in container

Data Volumes for MySQL

```
##############################################################
# Add MySQL docker container
##############################################################
config.vm.provision "shell", inline: <<-SHELL
  # Prepare MySQL data share
  sudo mkdir -p /var/lib/mysql
  sudo chown vagrant:vagrant /var/lib/mysql
SHELL

# Add MySQL docker container
config.vm.provision "docker" do |d|
  d.pull_images "mariadb"
  d.run "mariadb",
    args: "--restart=always -d --name mariadb -p 3306:3306 -v /var/lib/mysql:/var/lib/mysql  -e
MYSQL_ROOT_PASSWORD=passw0rd"
end
```

Data Volumes for Redis

```
##############################################################
# Add Redis docker container
##############################################################
config.vm.provision "shell", inline: <<-SHELL
  # Prepare Redis data share
  sudo mkdir -p /var/lib/redis/data
  sudo chown vagrant:vagrant /var/lib/redis/data
SHELL

# Add Redis docker container
config.vm.provision "docker" do |d|
  d.pull_images "redis:alpine"
  d.run "redis:alpine",
    args: "--restart=always -d --name redis -h redis -p 6379:6379 -v /var/lib/redis/data:/data"
end
```

Data Volumes for CouchDB

```
##############################################################
# Add CouchDB docker container
##############################################################
config.vm.provision "shell", inline: <<-SHELL
  # Prepare CouchDB data share
  sudo mkdir -p /usr/local/var/lib/couchdb
  sudo chown vagrant:vagrant /usr/local/var/lib/couchdb
SHELL

# Add CouchDB docker container
config.vm.provision "docker" do |d|
  d.pull_images "couchdb"
  d.run "couchdb",
    args: "--restart=always -d --name couchdb -p 5984:5984 -v /usr/local/var/lib/couchdb:/usr/local/
var/lib/couchdb"
end
```

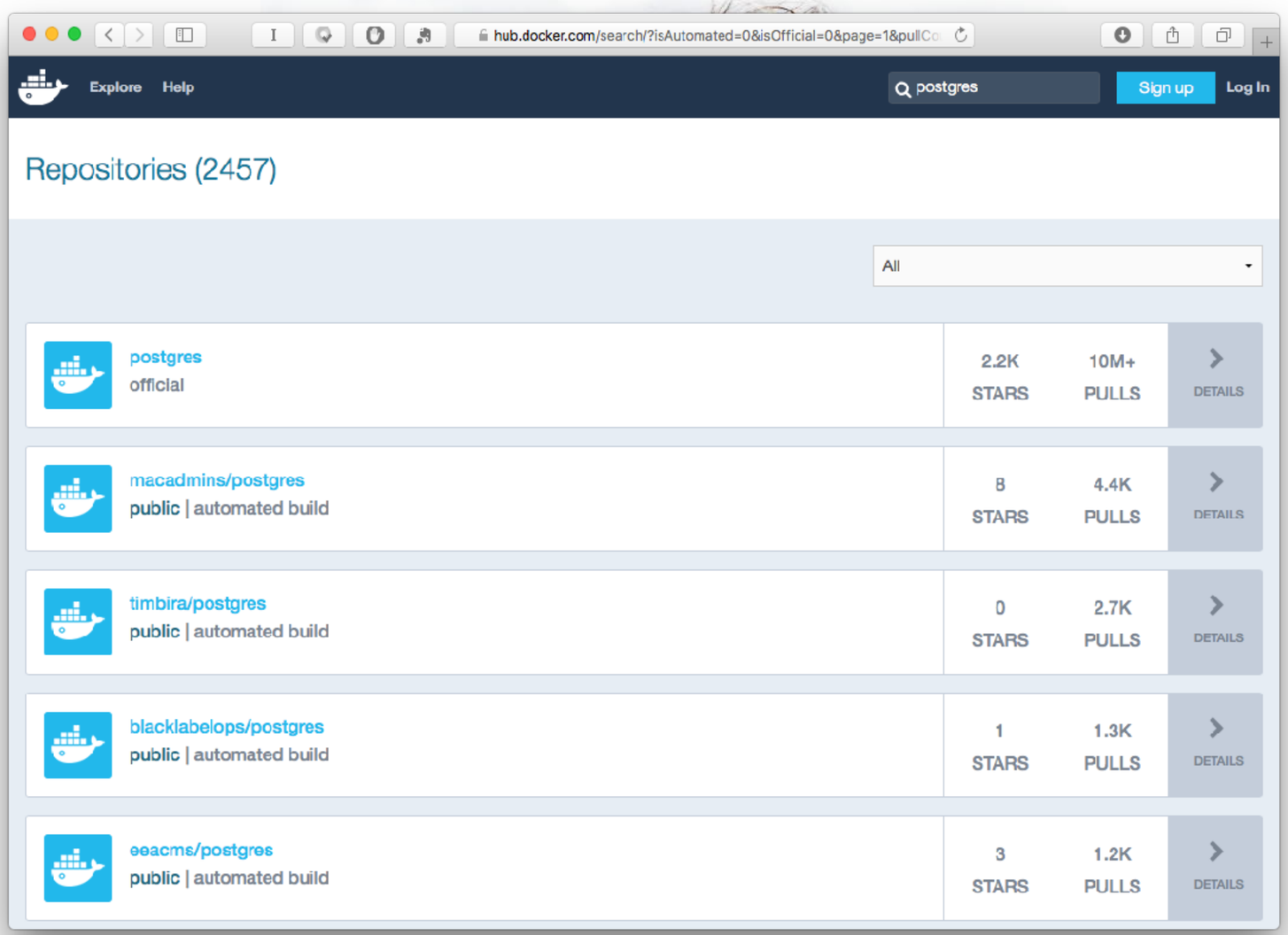
Handy Docker Startup Args

- `--restart=always` <— if it stops, restart it
- `-d` <— run as a daemon process (background)
- `--name <name>` <— name of running container in docker ps
- `-p <host>:<cont>` <— forward host port to container port
- `-v </host/folder>:<container/folder>` <— map volume
- `-e MY_ENV_VAR=<value>` <— inject environment variable

Where To Get More Containers?

hub.docker.com

- Contains 1000's of Docker images with almost every software you can imagine
- Usually "official" images come from the creator of the software
- Most have documentation on how to best use them



Ending Your Vagrant Session

- When you are done for the day, it's a good idea to shutdown your VM.
- This is accomplished with the `halt` command

```
$ exit
iotia:lab-vagrant rofrano$ vagrant halt
==> default: Attempting graceful shutdown of VM...
```

Removing a VM Permanently

- When you no longer need a VM you can delete it from your computer
- This is accomplished with the `destroy` command

```
$ vagrant destroy

    default: Are you sure you want to destroy the 'default' VM? [y/N]
==> default: Destroying VM and associated drives...
```

Handy Vagrant Commands

- `vagrant up` — bring up the vm
- `vagrant ssh` — open a shell inside the vm
- `exit` — exit out of the vm shell back to your host computer
- `vagrant halt` — shutdown the vm to return later with vagrant up
- `vagrant suspend | resume` — suspend vm and resume it later
- `vagrant reload --provision` — restarts vagrant machine, loads new Vagrantfile configuration and reprovisions shell scripts
- `vagrant destroy` — delete the vm for good
- `vagrant status` — see the status of all the current Vagrantfile's VM
- `vagrant global-status` — see the status of all Vagrant VM's

Summary

- You just deployed your first Vagrant environment
- You learned how to automatically install additional software
- You added Docker containers to supply needed middleware
- You can now check your `Vagrantfile` into github so that others can create the perfect development environment when working on your project.

QUESTIONS?

THANK YOU!