CHECKLIST

Installation I

Installing Ruby and Ruby on Rails is something that all developers in any operating system struggle w workings of each OS, there are many tools for Linux, Mac OS X, and Windows that people have create of RoR. However, configurations and dependencies are always going to be different depending on the developer in one OS will get an error that others in another OS would not. These types of inconsistenc developing in any language or framework.

Enter Vagrant

Why Vagrant? (https://www.vagrantup.com/docs/why-vagrant) Vagrant is a tool to create configurabl development environments.

> *... so whether you are working on Linux, Mac OS X, or Windows, all you running code in the same environment, against the same dependencies same way. Say goodbye to "works on my machine" bugs. - Vagrant*

Doesn't this sound awesome? Everyone in this track will be running the same exact environment. Ther configuration or dependency error, the solution can be quickly shared to everyone! This means that al caused by the developer themselves, instead of obscure environment errors. In this chapter, we will se virtual machines, and set up a static IP address for us to interact with our RoR application through a b

Set Up Vagrant

Vagrant allows users to easily create virtual machines. For those unfamiliar with VMs, they are simply a system. For example, you can have a Mac OS X computer running a Windows virtual machine. To spin uses VirtualBox, a free virtualization software.

- Download (https://www.virtualbox.org/wiki/Downloads) VirtualBox for your operating system.
- Download (https://www.vagrantup.com/downloads.html) Vagrant for your operating system.

Getting Started

The first vagrant command that we are going to learn is `vagrant init` . This built-in command will create *Vagrantfile*. This file is what Vagrant uses to configure our virtual machines. Let's create a directory an

```
$ mkdir first_vagrant_box
$ cd first_vagrant_box
$ vagrant init
```

Now you should have a Vagrantfile in your directory. Feel free to open it in your text editor but do not

Boxes

Instead of installing virtual machines from scratch every time, Vagrant uses a base image system to qu virtual machine. These images are called Vagrant "boxes". Vagrant has a whole catalog of public boxes can visit at HashiCorp's Atlas catalog (https://atlas.hashicorp.com/boxes/search). Let's add the ubuntu computer.

```
$ vagrant box add ubuntu/trusty64
```

This command downloads that box from the catalog and saves it globally. To see a list of your boxes y

```
$ vagrant box list
```

Next, we will configure our Vagrantfile to use the box that we just downloaded. Open the Vagrantfile a machine to boot up with our ubuntu box.

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

Vagrant Up

So far, we have done a couple things.

- Downloaded a ubuntu image.

- Initialized a directory with a Vagrantfile that uses that image.

Next, we have to actually spin up that image into a running virtual machine. In your console, run:

```
$ vagrant up
```

Once the VM is up and running, we can ssh into it.

```
$ vagrant ssh
```

This command allows you to ssh into your virtual machine and interact with it. Soon, we will install Rub
machines!

Synced Folder

Everything is great so far, but how do we actually develop with our VM. Unless you are an expert in ter
Nano, writing code in your VMs does not seem like fun. So, Vagrant comes with a synced folder functi
into your VM, run:

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /vagrant
vagrant@vagrant-ubuntu-trusty-64:~$ ls
Vagrantfile
```

This */vagrant* directory in your VM is automatically synced with the host machine's directory where va
means, if we open another terminal window and create a new file in host's *first_vagrant_box* directory
/vagrant directory

```
$ cd first_vagrant_box
$ touch is_this_file_synced
```

Moment of truth.

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /vagrant
vagrant@vagrant-ubuntu-trusty-64:~$ ls
is_this_file_synced  Vagrantfile
```