Vagrant Part 02

References

1. Vagrant: Up and Running, by Mitchell Hashimoto. Publisher: O'Reilly Media, Inc.
2. https://www.vagrantup.com
3. https://www.vagrantup.com/docs/provisioning/

Synced Folders

- Vagrant maps the root Vagrant project directory into a folder in the Vagrant managed VM as /vagrant

- You can synch other folders between your host machine and the Vagrant VM in the Vagrantfile of your project − relative paths on the host machine as relative to the root directory of the Vagrant project

Synced Folders

- The following line when added to Vagrantfile maps the parent directory into a folder called vagrant_scripts off of the root in the Vagrant VM:

config.vm.synced_folder ".." , "/vagrant_scripts"

Synced Folders

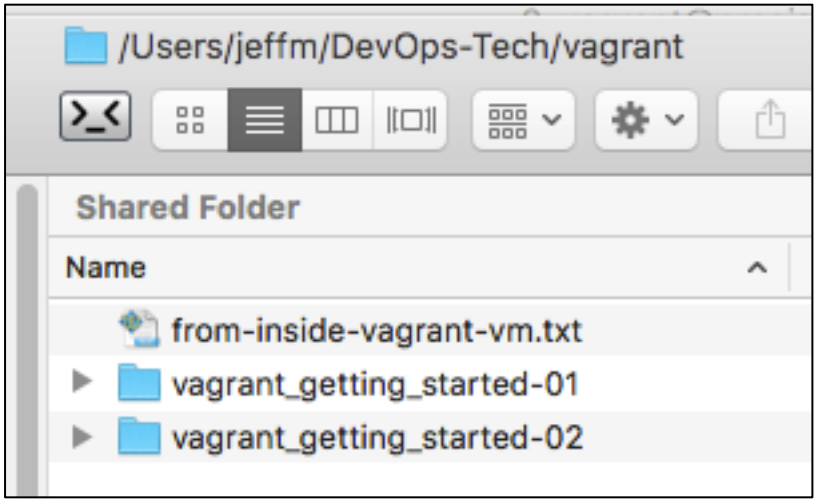
```
JeffsMacBookPro:vagrant_getting_started-02 jeffm$ vagrant ssh
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Wed Jun  7 05:19:34 2017 from 10.0.2.2
vagrant@precise64:~$ cd /vagrant_scripts
vagrant@precise64:/vagrant_scripts$ ls
vagrant_getting_started-01  vagrant_getting_started-02
vagrant@precise64:/vagrant_scripts$ ls > from-inside-vagrant-vm.txt
vagrant@precise64:/vagrant_scripts$ ls
from-inside-vagrant-vm.txt  vagrant_getting_started-01  vagrant_getting_started-02
vagrant@precise64:/vagrant_scripts$
```

From "inside" the Vagrant managed VM using ssh

Synced Folders



From the host machine

Vagrant Provisioning

- Provisioners in Vagrant allow you to:
  - automatically install software,
  - alter configurations,
  - and perform various tasks

  as part of the **vagrant up** process.

Vagrant Provisioning

- You can use Bash Shell scripts, Chef, Puppet, and others as a provisioner.

- We will use Files and Bash Scripts in this section.

Vagrant Provisioning

- Provisioning can take place:
  - on the first use of "vagrant up"
  - subsequent uses of "vagrant up" will not provision the VM unless you use command line flag:

    --provision

Vagrant Provisioning

- By default, provisioners are only run once, **during the first vagrant up since the last vagrant destroy**, unless the --provision flag is set

Vagrant Provisioning

- For example, if you make a change in the Vagrant file, or want to have Vagrant run a Bash script to provision:

  – if the VM is running stop it with: **vagrant halt** (no needed if the VM is not running)

  – next bring the VM back up with:

  vagrant up --provision

Vagrant Provisioning

- Command **vagrant reload --provision**

  will run **vagrant halt** followed by

  vagrant up --provision

- **vagrant up --no-provision** prevent provisioning

Vagrant Provisioning

- You can also run

  vagrant provision

  on a running VM

Vagrant Provisioning

- To automatically upload a files and/or directories into a Vagrant VM , you can add line:

config.vm.provision "file", source: "path-on-host", destination: "path-in-vagrant-vm"

For example:

config.vm.provision "file", source: "~/myfiles", destination: "/myfileszz"

copies the contents of file or directory myfiles in your host's home directory into file or directory /myfileszz under the root of the VagrantVM

Vagrant Provisioning

- Using the shell provisioner you can load and execute a script file within the guest VM
- For Macs and Linux VMs you can upload Bash Shell Scripts
- For Windows VM, you can upload and execute Powershell or Batch files

Vagrant Provisioning

- The following Vagrantfile command uploads a Bash shell script named bootstrap.sh into a Linux guest − and executes it

config.vm.provision :shell, path: "bootstrap.sh"

Vagrant Port Forwarding

- A common practice is to put a web server or web site that is accessed over the network/ internet.
- To do this – you can use **port forwarding**
- port forwarding associates an IP port in the VM with a port on the host machine

Vagrant Port Forwarding

- The following Vagrantfile command associates port 80 on the guest with port 4567 on the host:

  config.vm.network :forwarded_port, guest: 80, host: 4567

Vagrant Provisioning Example

- The following example, from the Vagrant documentation:
  - using script provisioning, uploads a bash script called bootstap.sh intp the guest
  - the code in bootstrap.sh downloads and installs the Apache web server
  - using port forwarding, associated port 80 on the guest VM with port 4567 on the host machine.

Vagrant Provisioning Example

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

apt-get update
apt-get install -y apache2
if ! [ -L /var/www ]; then
  rm -rf /var/www
  ln -fs /vagrant /var/www
fi
```

bootstrap.sh

Vagrant Provisioning Example

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/precise64"

  # have vagrant run a script
  config.vm.provision :shell, path: "bootstrap.sh"

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine and only allow access
  # via 127.0.0.1 to disable public access
  # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"
  config.vm.network :forwarded_port, guest: 80, host: 4567

  #synch local folder ../data to /variant_data in VM
  config.vm.synced_folder "..", "/vagrant_data"
end
```

Vagrantfile

```
JeffsMacBookPro:vagrant_getting_started-02 jeffm$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'hashicorp/precise64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'hashicorp/precise64' is up to date...
==> default: Setting the name of the VM: vagrant_getting_started-02_default_1496816826208_65721
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 80 (guest) => 4567 (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:
    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!
==> 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: 4.2.0
    default: VirtualBox Version: 5.1
==> default: Mounting shared folders...
    default: /vagrant => /Users/jeffm/DevOps-Tech/vagrant/vagrant_getting_started-02
    default: /vagrant_data => /Users/jeffm/DevOps-Tech/vagrant
```

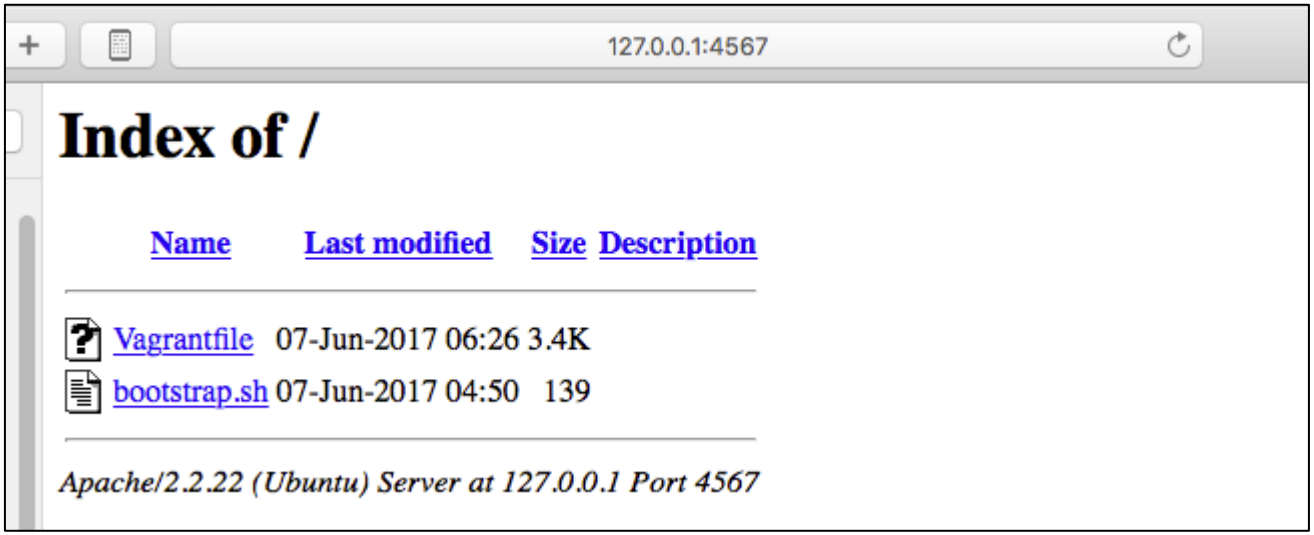
- create a new directory and run "vagrant init" for each new VM
- write the Vagrant file, next run -
- vagrant up

```
JeffsMacBookPro:vagrant_getting_started-02 jeffm$ vagrant ssh
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2
vagrant@precise64:~$ ls /vagrant_data
from-inside-vagrant-vm.txt  vagrant_getting_started-01  vagrant_getting_started-02
vagrant@precise64:~$
```

vagrant ssh



http://127.0.0.1:4567

Vagrant

- Command **vagrant halt** shutdowns the VM
- The VM can be brought backup without provisioning using: **vagrant up**

Vagrant

```
JeffsMacBookPro:vagrant_getting_started-02 jeffm$ vagrant halt
==> default: Attempting graceful shutdown of VM...
```

vagrant halt

Vagrant

```
JeffsMacBookPro:vagrant_getting_started-02 jeffm$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'hashicorp/precise64' 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: 80 (guest) => 4567 (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: 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: 4.2.0
    default: VirtualBox Version: 5.1
==> default: Mounting shared folders...
    default: /vagrant => /Users/jeffm/DevOps-Tech/vagrant/vagrant_getting_started-02
    default: /vagrant_data => /Users/jeffm/DevOps-Tech/vagrant
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
```

Vagrant

- Besides uploading and running Bash Shell scripts, you can also place Bash Shell Scripts "inline" within the Vagrantfile , for example

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

- You can also place Ruby scripts within Vagrantfile