Synced folders

Using **synced folders**, Vagrant will automatically sync our files to and from the guest machine. In other words, Vagrant shares our project directory (where the Vagrantfile is in) to the**/vagrant** directory in our guest machine. Run **vagrant up** again and **SSH** into our machine to see:

**$ vagrant up**

**$ vagrant ssh**

**vagrant@vagrant-ubuntu-trusty-32:~$ ls /vagrant**

**Vagrantfile**

The **Vagrantfile** we see inside the virtual machine is actually the same **Vagrantfile** that is on our host machine. Go ahead and touch a file to check it:

**vagrant@vagrant-ubuntu-trusty-32:~$ touch /vagrant/test.txt**

**vagrant@vagrant-ubuntu-trusty-32:~$ ls**

**vagrant@vagrant-ubuntu-trusty-32:~$ exit**

**logout**

**Connection to 127.0.0.1 closed.**

**k@laptop:~/my\_vagrant$ ls**

**test.txt Vagrantfile**

The "test.txt" file is now on our host machine. As we can see, Vagrant kept the folders in sync. With synced folders, we can continue to use our own editor on our host machine and have the files sync into the guest machine.

Provisioning - config.vm.provision

Now, we have a virtual machine running a basic copy of Ubuntu and we can edit files from our machine and have them synced into the virtual machine. Let's now serve those files using a webserver.

Vagrant has built-in support for automated provisioning. Using Vagrant's provisioning feature, Vagrant will automatically install software when we vagrant up so that the guest machine can be repeatable created and ready-to-use.

Apache install via shell provisioner

In this section, we'll setup Apache for our basic project using a shell script, **bootstrap.sh**:

**k@laptop:~/my\_vagrant$ ls**

**bootstrap.sh test.txt Vagrantfile**

The "bootstrap.sh" file looks like this:

**k@laptop:~/my\_vagrant$ ls**

**#!/usr/bin/env bash**

**apt-get update**

**apt-get install -y apache2**

**rm -rf /var/www**

**ln -fs /vagrant /var/www**

Next, we need to configure Vagrant to run this shell script when setting up our machine. We do this by editing the **Vagrantfile**:

The "bootstrap.sh" file looks like this:

**Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|**

**config.vm.box = "ubuntu/trusty32"**

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

**end**

The "provision" line tells Vagrant to use the **shell provisioner** to setup the machine, with the**bootstrap.sh** file. The file **path** is relative to the location of the **project root** (where the**Vagrantfile** is).

After the configuration, we just run **vagrant up** to create our virtual machine via automatic provision by Vagrant. We should see the output from the shell script appear in our terminal.

**k@laptop:~/my\_vagrant$ vagrant up**

**Bringing machine 'default' up with 'virtualbox' provider...**

**==> default: Checking if box 'ubuntu/trusty32' is up to date...**

**==> default: VirtualBox VM is already running.**

If the guest machine is already running from a previous step as in our case, we run **vagrant reload --provision**, which will quickly restart our virtual machine, skipping the initial import step.

**k@laptop:~/my\_vagrant$ vagrant reload --provision**

**==> default: Attempting graceful shutdown of VM...**

**==> default: Checking if box 'ubuntu/trusty32' 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: 22 => 2222 (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: Connection timeout. Retrying...**

**...**

**default: Warning: Remote connection disconnect. Retrying...**

**==> default: Machine booted and ready!**

**==> default: Checking for guest additions in VM...**

**==> default: Mounting shared folders...**

**default: /vagrant => /home/k/my\_vagrant**

**==> default: Running provisioner: shell...**

**...**

**==> default: The following NEW packages will be installed:**

**==> default: apache2 apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3**

**==> default: libaprutil1-ldap ssl-cert**

**==> default: 0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.**

**==> default: Need to get 1,270 kB of archives.**

**==> default: After this operation, 5,050 kB of additional disk space will be used.**

**==> default: Get:1 http://archive.ubuntu.com/ubuntu/ trusty/main libapr1 i386 1.5.0-1 [88.8 kB]**

**==> default: Get:2 http://archive.ubuntu.com/ubuntu/ trusty/main libaprutil1 i386 1.5.3-1 [76.6 kB]**

**==> default: Get:3 http://archive.ubuntu.com/ubuntu/ trusty/main libaprutil1-dbd-sqlite3 i386 1.5.3-1 [10.3 kB]**

**==> default: Get:4 http://archive.ubuntu.com/ubuntu/ trusty/main libaprutil1-ldap i386 1.5.3-1 [8,552 B]**

**==> default: Get:5 http://archive.ubuntu.com/ubuntu/ trusty-updates/main apache2-bin i386 2.4.7-1ubuntu4.1 [821 kB]**

**==> default: Get:6 http://archive.ubuntu.com/ubuntu/ trusty-updates/main apache2-data all 2.4.7-1ubuntu4.1 [160 kB]**

**==> default: Get:7 http://archive.ubuntu.com/ubuntu/ trusty-updates/main apache2 i386 2.4.7-1ubuntu4.1 [87.6 kB]**

**==> default: Get:8 http://archive.ubuntu.com/ubuntu/ trusty/main ssl-cert all 1.0.33 [16.6 kB]**

**==> default: dpkg-preconfigure: unable to re-open stdin: No such file or directory**

**...**

**==> default: Processing triggers for ureadahead (0.100.0-16) ...**

**==> default: Processing triggers for ufw (0.34~rc-0ubuntu2) ...**

**k@laptop:~/my\_vagrant$**

The provision flag on the reload command instructs Vagrant to run the **provisioners**, since usually Vagrant will only do this on the first vagrant up.

After Vagrant completes running, the web server will be up and running. However, we can't see the website from our own browser yet, but we can verify that the provisioning works by loading a file from SSH within the machine:

**k@laptop:~/my\_vagrant$ vagrant ssh**

**Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-39-generic i686)**

**...**

**vagrant@vagrant-ubuntu-trusty-32:~$ wget -qO- 127.0.0.1**

This works because in the shell script above we installed Apache and setup the default**DocumentRoot** of Apache to point to our **/vagrant** directory, which is the default synced folder setup by Vagrant.

We can check if apache is actually running:

**vagrant@vagrant-ubuntu-trusty-32:~$ ps -ef|grep apache2**

**root 2295 1 0 03:44 ? 00:00:00 /usr/sbin/apache2 -k start**

**www-data 2297 2295 0 03:44 ? 00:00:03 /usr/sbin/apache2 -k start**

**www-data 2299 2295 0 03:44 ? 00:00:03 /usr/sbin/apache2 -k start**