GETTING STARTED WITH VAGRANT

Vagrant is a tool that manages virtual machines for you, using a simplified command line interface. It reads from a configuration file, named “Vagrantfile”, to build the virtual machine initially. It handles networking and port forwarding, setting up shared folders, and has commands to SSH into the box, suspend and resume your VM, etc. Vagrant’s big selling point is that it can create identical virtual machines every time, based off of a simple config file. These VMs are disposable, you can destroy and recreate them as needed.

Since Vagrant’s inception in early 2010, it has grown to be used by thousands of companies worldwide, including Puppet Labs, BBC, Mozilla, Nokia, and many others.

1. Installing Vagrant

The first step for getting started with Vagrant is to install Vagrant and it’s dependency, VirtualBox. Vagrant can work with other providers, such as VMWare Workstation, but it requires additional work and sometimes paid plugins. It’s easiest to get started with VirtualBox, which is free, and has versions for Windows, Mac and Linux.

1. Install Oracle VirtualBox for your platform
2. Download the extension pack (Oracle VM VirtualBox Extension Pack) for all supported platforms, and install it by double clicking on the downloaded file once VirtualBox has been installed.
3. Navigate to the Vagrant website, and click Download. Download the newest version for your operating system.
4. Install Vagrant using the installer you downloaded
5. You should now have access to the `vagrant` command from your command line

2. Create your first VM

If you want to create a new Vagrantfile:

1. Open up the command prompt
2. Run `vagrant init precise32 http://files.vagrantup.com/precise32.box`
3. Now you can run `vagrant up`

If you're working with a project that contains a Vagrantfile:

1. Clone the repository to your local computer using your Git client of choice (assuming the project is in Git or a similar VCS)

2. Open up the folder you put the repository in with the command prompt or Cygwin.
3. Type `vagrant up`, and wait for Vagrant to provision the virtual machine. This may take a while, depending on the provisioning scripts
4. Navigate to `http://localhost:XXXX` in your web browser, where XXXX is the port you have setup to forward to Apache/Nginx/Whatever. Projects should document this, and Vagrant should also tell you which ports it is forwarding.

With Vagrant, `vagrant up` is all you need to work on any project, to install every dependency that project needs, and to setup any networking and synced folders so you can continue working from the comfort of your own machine.

SSH Access

If you require SSH access to your virtual machine, it's as simple as running `vagrant ssh`. You can switch to the root user via `sudo su`. No password is required.

The project source code will be mounted in `/var/www/html`.

Suspending/Resuming Your VM

If you wish to suspend your VM, simply run `vagrant suspend`.

You may later resume your VM using `vagrant resume`.

Cleaning Up Your VM

If you wish to destroy your VM to free up disk space, simply run `vagrant destroy`.

This will **not** delete any files in your repository folder.

You can later run `vagrant up` to create a new virtual machine.

Vagrantfiles

As mentioned previously, Vagrant reads a configuration file from the directory you run vagrant from, called “Vagrantfile”. This file should be commited with your project into version control. Each project should have it’s own Vagrantfile. This may seem like a pain, but Vagrant sets up port forwarding for you, so it’s quite easy to use. Run `vagrant up`, then navigate to localhost:5050 to get to Project XYZ.

The Vagrantfile is defined using a Ruby DSL. You can find in depth documentation over atVagrantUp.com. Here’s a sample file from one of my starter templates, which you can feel free to experiment with or use in your own projects.

```
Vagrant.configure("2") do |config|
  # We use Ubuntu 12.04 LTS (Precise Pangolin) as our box. This was the only
  # stable box that worked in all our environments. We use 13.04 (Raring
  # Ringtail) repository sources in Puppet though.
  config.vm.box = "precise32"
  config.vm.box_url = "http://files.vagrantup.com/precise32.box"

  # Map port 5000 on the host machine to port 80 on the guest (WordPress)
  config.vm.network :forwarded_port, guest: 80, host: 5000

  # Map port 5001 on the host machine to port 8000 on the guest (phpMyAdmin)
  config.vm.network :forwarded_port, guest: 8000, host: 5001

  # Map the repository folder to /var/www/html/wordpress as well as /vagrant
  config.vm.synced_folder "./", "/var/www/html/wordpress"

  # Use Puppet to provision our virtual machine
  config.vm.provision :puppet do |puppet|
    puppet.manifests_path = "vagrant/manifests"
    puppet.manifest_file  = "init.pp"
  end
end
```

You can see that the syntax is very straight forward. Here, we are setting up a Ubuntu VM, forwarding port 5000 on the host machine to port 80 on the VM. We're also going to map the folder with the Vagrantfile in it (our project folder) to /var/www/html/wordpress.

Finally, we're going to run a Puppet manifest, vagrant/manifests/init.pp to "provision"" the vm. Provisioning just means installing and configuring software as needed. Puppet is an automated way of doing this. Vagrant supports plain bash scripts, Puppet, or Chef.

If you'd like to see the accompanying Puppet manifest, check it out on GitHub.

Vagrant Commands

For a full list of commands, please refer to the documentation.

Important commands:

vagrant up

Run this from your project directory to bring up a new virtual machine based on the Vagrantfile in the current directory.

vagrant suspend

Suspend the virtual machine so it's no longer using memory. The VM's state is saved.

vagrant resume

Resume a suspended virtual machine, restoring it's state.

vagrant destroy

Shutdown and destroy a virtual machine. This won't affect your project at all.

vagrant provision

Re-run the provisioning scripts.

vagrant reload

Shutdown and restart the VM to apply any chances made to your Vagrantfile such as new port forwarding configurations. Will also re-run the provisioning scripts.

vagrant ssh

SSH into your VM. No credentials needed. You'll be SSH'd in as the `vagrant` user, but you can switch to root by running `sudo su`.

Source: http://brandonwamboldt.ca/getting-started-with-vagrant-1284/