VAGRANT

Agenda

- What is Vagrant
- Why do we need Vagrant
- Installing Vagrant
- Some important commands
- Vagrant Networking
- Shared folders

What is Vagrant

Vagrant is a tool for building complete development environments, sandboxed in a virtual machine. Vagrant lowers development environment setup time, increases development /production parity, and brings the idea of disposable compute resources down to the desktop.

Why Vagrant

- Creates a virtual machine for you based on an operating system of your choice.
- Modifies the physical properties of this virtual machine (e.g., RAM, number of CPUs, etc.).
- Establishes network interfaces so that you can access your virtual machine from your own computer, another device on the same network, or even from another virtual machine.
- Sets up shared folders so that you can continue editing files on your own machine and have those modifications mirror over to the guest machine.
- Boots the virtual machine so that it is running.
- Sets the hostname of the machine, since a lot of software depends on this being properly set.
- Provisions software on the machine via a shell script or configuration management solution such as Chef, Puppet, or a custom solution.

Installing Vagrant

- Download Oracle VM VirtualBox and VirtualBox Extension Pack

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

•Download Vagrant

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

•Download GIT/GIT Bash

https://git-scm.com/download/win

Vagrant Terminology

Vagrant base box - a stored VirtualBox machine packaged into a single file. Think of this as the template for your Vagrant box.

Vagrant box - an instance of a VirtualBox VM that has been provisioned and started using

Provision - the configuration step that comes after the Vagrant box loads.

Vagrantfile - a single file that defines what a particular Vagrant box is, including the base box, network settings, and provisioning.

Vagrant File

MINGW64:/c/Users/VS1645/vag/test1

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

Forwarding port

MINGW64:/c/Users/VS1645/vag/test1

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
end
```

Network interface

MINGW64:/c/Users/VS1645/vag/test1

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.network "private_network", ip: "192.168.3.100"
end
```

Shared Folders

MINGW64:/c/Users/VS1645/vag/test1

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.network "private_network", ip: "192.168.3.100"
  config.vm.synced_folder "data", "/vagrant_data"
end
```

Shell Provisioning

MINGW64:/c/Users/VS1645/vag/test1

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.network "private_network", ip: "192.168.3.100"
  config.vm.synced_folder "data", "/vagrant_data"
  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y apache2
  SHELL
end
```

Multi Machine

MINGW64:/c/Users/VS1645/vag/multi

```
Vagrant.configure(2) do |config|
  config.vm.define :server1 do |server1|
        server1.vm.box = "ubuntu/trusty64"
  end

  config.vm.define :server2 do |server2|
        server2.vm.box = "ubuntu/trusty64"
  end
end
```

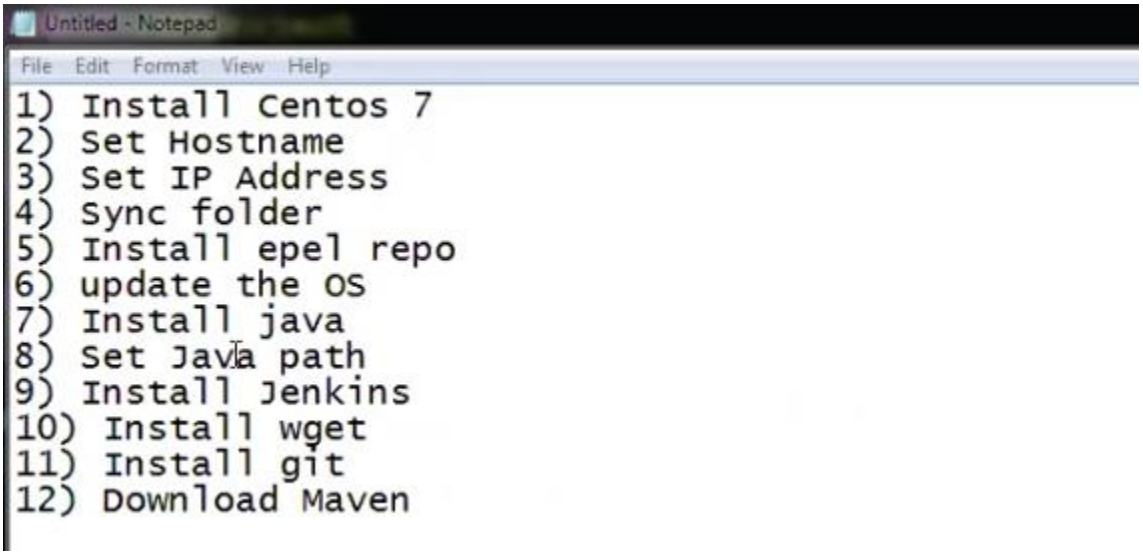
Vagrant commands

vagrant –version - Display version of Vagrant installed
vagrant box add – Add a new base box
vagrant box list – List all the boxes
vagrant box remove– List all the boxes
vagrant init - initialize a new vagrant box in the current directory
vagrant up - start an existing vagrant environment (box) and provision it
vagrant ssh - shell into a running vagrant box
vagrant halt - stop a running vagrant box (shut down the computer)

Vagrant commands

vagrant reload – restartes a virtual machine
vagrant suspend -- Suspends a virtual machine (remembers state)
vagrant resume - Resume a suspended machine (vagrant up works just fine for this as well)
vagrant destroy - completely destroy a vagrant box (delete all the things).
vagrant global-status – display the status of the all the vagrant boxes
vagrant box outdated –checks for updates for vagrant boxes
vagrant box update–updates the vagrant boxes

Use Case :



Code in vagrant file :

```
Vagrant.configure("2") do |config|
        config.vm.define "master" do |master|
        master.vm.provider :virtualbox
        #Jenkins Master node running on Centos 7
        master.vm.box = "centos/7"
        master.vm.hostname = "master.mylab.local"
        master.vm.network :private_network, ip:"10.0.0.3"
        #master.hostmanager.aliases = %w(master1)
        config.vm.synced_folder ".", "/vagrant", type: "virtualbox"
        master.vm.provision "shell", inline: <<-SHELL
                sudo systemctl reload  sshd
                sudo yum install epel-release -y
                sudo yum update -y
                #Java installtion and path set
```

```
sudo yum install java-1.8.0* -y

java -version

sudo cp /etc/profile /etc/profile_backup

echo 'export JAVA_HOME=/usr/lib/jvm/java-openjdk' | sudo tee -a /etc/profile

echo 'export JRE_HOME=/usr/lib/jvm/jre' | sudo tee -a /etc/profile_backup

source /etc/profile

#cross check Java path and home dir

echo $JAVA_HOME

echo $JRE_HOME

#Installation of wget

sudo yum install wget -y

#Installation of Jenkins

cd ~

sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo

sudo rpm --import http://pkg.jenkins-ci.org/redhat-stable/jenkins-ci.org.key

sudo yum install jenkins -y

#Start the Jenkin service

sudo systemctl start jenkins.service

sudo systemctl enable jenkins.service

#Allow Jenkins port on firewall

sudo firewall-cmd --zone=public --permanent --add-port=8080/tcp

sudo firewall-cmd --reload

#Installation of git

sudo yum install git -y

#Download The Maven
```

```
                cd /opt
                sudo wget http://www-eu.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz
                sudo tar -xvf apache-maven-3.5.2-bin.tar.gz
                sudo sed -i '/PasswordAuthentication/d' /etc/ssh/sshd_config
                sudo echo 'PasswordAuthentication yes' >> /etc/ssh/sshd_config
                sudo systemctl reload  sshd
                SHELL
        end
end
```