What is Vagrant ?

Answer :

- Vagrant creates virtual machines.
- We will provision the virtual machines via shell script or configuration management using tools like Chef and Puppet.
- Vagrant will automatically make modification needed to the Ubuntu configs, Vagrant does this sort of fix on many guest and host setups.
- Managing large amounts of nodes becomes expensive and unmanageable therfore like Vagrant managing your tools typically a system using with virtual box.
- Vagrant handle the entire lifecycle of development machines:
1. It suspend , halt and resume virtual machines.
2. Destroy your virtual machines and delete its metadata.
3. SSH in to your machine.
4. Package up your machines entire state and the re distribute it to other development team members.
- Vagrant is a **must have tool in** development environments in which is allows us to use automation of an environment that will resemble your actual production.

Why should we use Vagrant?

Answer :

- Traditionally we should have to manually install and setup things like, Apache and MySQL locally on our development machines.
- This can be very complex with all amount of moving parts to keep track of .
- Human error is easy to take place when setting up complete local development environments.
- Vagrant is an awesome way of fixing these issues by automating our setup of all these services needed to develop our applications. Each project gets its own virtal machines. Working within your team is snap because we can share our virtual machine images by telling sharing a simple command that will build an exact replica of your environment in minutes.

How does Vagrant Work-/how goes ?

Answer : Developers can now check out versions or repositories from a version control such as GIT and run 'vagrant up'.

- Now the developers can work on their own machines comfortably with their own , editors , browsers and tools.
- We have a consistent and stable development environment now.
- The same scripts or configuration in using vagrant to deploy dev environments is the same used in production therefore the dev environment is as close as possible to the production limiting issues later on.
- If something was to go horribly wrong and we needed to start over from scratch you simply run a "vagant destroy" , that will remove all the traces of that development from your machines and then follow it with another "vagrant up" which will re-create the entire dev environment in moments.
- we are able to keep the dev systems clean by suspending , halting , and destroying our dev environments.

In terms of cost we no longer have to worry about the developer forgetting to shutdown an instance in our cloud at AWS that is costing us by the hour.

- Every project and the knowledge within that project can transfer when the work-flow is followed the same way eliminating issues between machines.

What is Virtualization ?

Answer : There are three types of Virtualization :

1) Desktop Virtualization.
2) Containers
3) Cloud - It is expensive because each cycle of hour you have to pay for it.

Note: For learning Vagrant , we are going to use Desktop-Virtualization like Virtual Box in our OS.

Install Virtual-Box on OSX ?

Answer : I have installed.

Install Vagrant on OSX ?

Answer : www.vagrantup.com Installed. To check version of Installed "vagrant –version".

Is these we can work Vagrant without Virtualbox ?

Answer : You can actually use Vagrant with other "providers" instead of just Virtual Box

- VMware
- AWS EC2
- And just about any other provider out there!

Note : We are primarily using VirtualBox because it is free.

Using Vagrant

Our first vagrant machine ?

Answer : our first Vagrant Up to see how quick and easy it is to deploy machines.

- create the folder and give any name in my case ex: vagrant101.
- **vagrant init precise64 http://files.vagrantup.com/precise64.box.**
- **vagrant up**
- **This will create a fully functional Ubuntu machine of precise64 (Ubuntu 12.04 LTS)**
- **vagrant ssh - will drop to Ubuntu precise image.**
- **ls**
- **ping www.google.com.**
- sudo apt-get update.
- sudo apt-get install vim.
- Exit.
- vagrant destroy.- This will destroy our Ubuntu file.

Our Vagrantfiles ?

Answer : Vagrant is configured on a per project basis.

- Each of these projects has its own 'Vagrant file'
- The Vagrant file is a text file in which vagrant reads that sets up our environment
- There is a description of what OS, how much RAM, and what software to be installed etc.
- You can version control this file so your development team can stay in sync on changes.
- mkdir LinuxAcademy
- cd LinuxAcademy
- **vagrant init precise64 http://files.vagrantup.com/precise64.box.**
- ls --> You will see a brand new file in our directory called Vagrantfile
- Vagrantfile is a ruby file. Please check the path of your ruby file i.e. Vagrantfile.

What are Boxes ?

Answer : Vagrant boxes are just "templates' like in Virtualization

- Boxes contain our base operating systems already setup
- Manage Boxes such as:
  - Vagrant box list
    Vagrant box add
    Vagrant box remove

--> **vagrant box list** :- will show which OS image is available in our box.

--> **vagrant box remove precise64** :- will remove the precise64 image of a file.

--> **vagrant destroy** :- will remove everything and will destroy vagrant images too of OS.

--> **vagrant box remove precise64** :- This will remove the precise64.

--> **vagrant box list** :- It will not show any images because we have deleted the files.

(Vagrant Up) Running Vagrant Machines ?

Answer : In the directory create folder as a LearningVagrant (it is on the the desktop)

- **vagrant init precise64 http://files.vagrantup.com/precise64.box**
- **vagrant up**
- **vagrant status**

SSH to Vagrant Machines ?

Answer : In this lesson we learn how to remote in to our vagrant machines with 'vagrant up'

- **vagrant ssh**
- **Ls**
- **mkdir test**
- **cd test**
- **touch file1**
- **Touch file2**
- **exit**
- **vagrant destroy**
- **say yes**
- **vagrant status**
- **vagrant up**
- **vagrant status**
- **vagrant ssh**
- **ls**

Synched Folders ?

Answer: Folders can be shared between the hosts and the guest virtual machine:

- Use our own editors of choice
- By default this is done by sharing your Project directory to a folder on the guest machine in /vagrant.
- We can actually change this behavior to where and what we want
- **config.vm.synched_folder "src/t," /server/website" end**

Networking Basics ?

Answer : We learn the basics such as port forwarding with vagrant machines :

- Vagrant automatically sets up networking so that developers can start working right away on the virtual machine.
- Forwarding ports
  - A forwarded port exposes a port on the guest machine as a port on a host machine.We will do port 80 for a web server as an example. We would modify our Vagrantfile to look like the following:
- **Config.vm.network"forwarded_port",guest: 80 , host:8080**
- After the Vagrantfile is edited we would run **a vagrant reload**

Commands :

- Go to the directory LinuxAcademy
- vagrant up
- vagrant status
- ls /vagrant/
- exit
- ls
- vim Vagrantfile
- set **config.vm.network"forwarded_port",guest: 80 , host:8080**
- **vagrant up**
- **vagrant reload**
- **vagrant status**
- **vagrant ssh**
- **ls /vagrant**
- **cd /vagrant**
- **ls**
- **sudo python -m SimpleHTTPServer 80**

Environment Management ?

Answer : We learn how to manage our virtual Machine's state with suspending, halting, and vagrant destroys .

- Suspend : It can act as "freezing time" within our virtual machine , it allows us to quickly clean up and quickly resume. After we suspend our virtual machine no more CPU or RAM resources are consumed on the host machine for the Vagrant environment , but disk space continues to be used , **vagrant suspend.**
- **vagrant halt** – Will cleanly shut down your virtual machine
- **vagrant halt –force** – If you need to shut down the server without a clean shut down we can force it.
- **vagrant destroy** - This will completely remove our machine from the environment.

command :

In directory LinuxAcademy

- **vagrant status**
- **vagrant suspend**- It will freeze the command ,then if you type for Vagrant ssh then also it will give us a error.
- Then if you want to resume it type : **vagrant up .**
- **vagrant halt** - It will shutdown VM , to resume you have to type **vagrant up.**
- **vagrant destroy - It will destroy all the file .**

Note : It will just destroy the mounted file on VM and if you want to recall then again you have to type vagrant up and vagrant status.

Automated Provisioning

Provisioners :

Answer : Base Vagrant boxes are bare. Only the minimal kunction such as SSH is installed after a base box install.

- We have the option to install software on our base box but that would have to be done via shell scripts, configuration management systems, or good old manual command-line entry installations.
- Vagrant allows us to do automatic provisioning. This is done when we run a 'vagrant up'.
- Provisioners can be used such as: Shell scripts , Chef, Puppet. There are also Several plug-ins available too.

Installing Apache Manually

Answer : In this lesson we install apache and then setup it up manually. This is a pre lesson to understanding what is involved in apache setup before we start with using provisioners and scripts.

Command

- ***Put in into the LinuxAcademy directory***
- **sudo apt-get update**
- **sudo apt-get install apache2**
- **sudo rm -rf /var/www - this will remove the var/www file.**
- **sudo ln/lan/link or symbolicllink) -fs /vagrant /var/www - this will add the symbolic link for the var/www file for apache.**
- create the index.html file by **echo "Hello" > index.html in directory called LinuxAcademy**
- **ls**
- **vagrant ssh**
- **ls**
- **cd /vagrant/**
- **ls - you will able to see index.html file**
- **exit**
- **Now check on port 8080 , you will able to see index.html on a browser.**

Installing Apache /Shell Script

Answer : In this lesson we will learn that we can use Bash shell scripts to automate the provision of our vagrant environment to already install Apache and have it configured for us without the need to manually do this after a 'vagrant up'.

- Go to the directory LinuxAcademy
- ls
- vagrant status
- vagrant destroy
- vim provision.sh - This is the shell scripting file.
- Provision.sh - The 6 commands should be in one file i.e. provision.sh

```
1. #!/usr/bin/env bash
2. echo "Installing Apache and setting up....please wait"
3. apt-get update >/dev/null 2>&1
4. apt-get install -y apache2 >/dev/null 2>&1
5. rm -rf /var/www
6. ln -fs /vagrant /var/www
```

- vim Vagrantfile - - config.vm.provision "shell", path: "provision.sh"
- vagrant up
- Now you will check your server is listening on port 8080.

Installing Apache /Chef

Answer : In this lesson we automate an apache install using Chef as our configuration management of our vagrant virtual machine.

- vim Vagrantfile

```
1.  config.vm.provision "chef_solo" do |chef|
2.    chef.add_recipe "vagrant_ia"
3.  end
```

- mkdir cookbooks
- cd cookbooks/
- mkdir vagrant_ia
- cd vagrant_ia
- mkdir recipes
- cd recipes/
- vim default.rb
- execute "apt-get update"
- package "apache2"
- service "rm -rf /var/www"
- link "/var/www" do
  to "/vagrant"

```
6.  end
```

- Go to the directory LinuxAcademy
- vagrant destroy
- vagrant up - you will see port 8080 will be running on a browser.

Networking

Answer : In these lesson we learn about Private Networks within our vagrant managed machine environment.

Private Networking

Private addresses that are not accessible from the public internet

DHCP

We can use DHCP to assign these IP addresses within these private networks automatically

```
Vagrant.configure("2") do |config|
  config.vm.network "private_network" , type: "dhcp"
end
```

STATIC IP

- We can use DHCP to assign these IP addresses within these private networks automatically

```
1.  Vagrant.configure("2") do |config|
2.      config.vm.network "private_network" , type: "192.168.70.4"
3.  end
```

Disable Auto-Configuration

```
    Vagrant.configure("2") do |config|
      config.vm.network "private_network" , type: "192.168.70.4"
      auto_config : false
    end
```

Public Networking

DHCP

We can use DHCP to assign these IP addresses within these public networks automatically

```
Vagrant.configure("2") do |config|
  config.vm.network "public_network"
end
```

STATIC IP

```
config.vm.network "public_network" , type: "192.168.0.17"
```

Deafault Interface

```
config.vm.network "public_network" , bridge:'en1:Wi-Fi(AirPort)'
```

Multiple Machines with Vagrant

Managing your multiple machines :

In these lesson we learn how we can setup multi-machine vagrant environments with just a single Vagrantfile.

Answer : In these setup , we make use of many application to be in like Web server , Database and Automate these process using Puppet.

Hands on with multiple machines :

Answer : Setting up our first multi-node vagrant deployment

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "precise64"
  config.vm.box_url = "http://files.vagrantup.com/precise64.box"

  # Setup for Web Server
  config.vm.define "web" do |web|
    web.vm.hostname = "web"
    web.vm.box = "apache"
    web.vm.network "private_network", type: "dhcp"
    web.vm.network "forwarded_port", guest: 80, host: 8081
    web.vm.provision "chef_solo" do |chef|
      chef.add_recipe "vagrant_ia"
    end
  end
  #Setup for MYSQL DB SERVER
  config.vm.define "db" do |db|
    db.vm.hostname = "db"
    db.vm.box = "mysql"
    db.vm.network "private_network", type: "dhcp"
  end
end
```

- vagrant status
- vagrant reload
- vagrant up
- Use cookbooks folder from Chef lesson.
- vagrant ssh web
- ifconfig - 172.28.128.3
- vagrant ssh db
- ifconfig –172.28.128.4
- Now let us communicate with each other
- ping 172.28.128.3 - See the ping statistics move
- ssh 172.28.128.3 - This will communicate and will require password and default password for vagrant machine is always **vagrant - You have communicate to database server to my web server and here full communication between these 2 boxes.**

Vagrant Boxes

Basic Box Management :

- Managed Globally per user not per project
- Boxes are mapped in vagrant to a logical name in which you name
- This name mapping maps to our config.vm.box settings in our vagrant file to actual Box we are building our machine from.
- Boxes are just files
- **Seagrant box**
- Usage: vagrant box <comand> [<args>]
  Available subcommands :
    add
    list
    remove
    repackage
- **Seagrant box add**
- **Seagrant box list**
- **Seagrant box remove**
- **Seagrant box repackage**

How to create boxes from an existing environment:

- **Seagrant status**
- **Seagrant ssh web**
- **Install software now by --> sudo apt-get install mc.**
- **mc**
- **exit**
- **exit**
- **vagrant package web**
- **ls**
- **vagrant box add web package.box**
- **vagrant box list**

Creating your own Boxes:

Building Boxes from Scratch.

- Create a brand new Virtual Machine inside Virtual Box.
- Use dynamically sized drives and set appropriate levels on RAM
- Remove an unnecessary components
- Install from scratch the Opertaing system of choice

Configure the Operating System

- A vagrant user must exist
- SSH server must be installed and configured to run on system boot.
- For Vagrant defaults be sure to use the public vagrant key
- We can customize this however different in the config.ssh.private_key_path to point to our own private key if we wish.