**LAB2: -Vagrant**

**Introduction to Vagrant**

Vagrant is an open-source software product for building and maintaining portable virtual software development environments, e.g. for VirtualBox, KVM, Hyper-V, Docker containers, VMware, and AWS. It tries to simplify software configuration management of virtualizations in order to increase development productivity.

Vagrant is a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.

**Get started with Vagrant?**

The best way to get started with Vagrant is to install it and try it out yourself. Aside from that, the [official documentation](https://www.vagrantup.com/docs/) are invaluable, and provide great directions for taking your first steps. It also helps to know some of the basic terminology used by Vagrant.

* **Box**: A box is a packaged Vagrant environment, typically a virtual machine.
* **Provider**: A provider is the location in which the virtual environment runs. It can be local (the default is to use VirtualBox), remote, or even a special case like a [Docker](https://opensource.com/resources/what-docker) container.
* **Provisioner**: A provisioner is a tool to set up the virtual environment, and can be as simple as a shell script, but alternatively a more advanced tool like Chef, Puppet, or Ansible can be used.

**Installation**

1. Before you start, make sure you already have a virtual provider on your system. Providers that work with Vagrant include VirtualBox, VMware, Docker, Hyper-V, and cloud service providers like Azure.

2. To find the latest version of Vagrant, use a web browser to navigate to its official webpage:

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

You will see a list of all the different supported operating systems, with a 32-bit and 64-bit package for each. Download the appropriate file for your operating system.

3. There are two ways to check if the installation was successful:

vagrant -v

which should show the version number running on your computer.

other way is to go to terminal and type command:

vagrant

which will list of frequently used commands if the tool were installed correctly.

**Project Setup**

1. Start by creating a directory to store your Vagrant file:

sudo mkdir vagrant-test

cd vagrant-test

2. Download the Ubuntu Trusty Tahr distribution from a common library and create a basic Vagrantfile with:

vagrant init ubuntu/trusty64

there are many different virtual boxes. You can browse them on the following link <https://app.vagrantup.com/boxes/search>

When you run the **init** command, Vagrant installs the box to the current directory. The Vagrantfile is placed in the same directory and can be edited or copied.

**Vagrant Boxes**

The basic unit in a Vagrant setup is called a “box” or a “Vagrantbox.” This is a complete, self-contained image of an operating system environment.

A Vagrant Box is a clone of a base operating system image. Using a clone speeds up the launching and provisioning process.

1. Instead of using the init command above, you can simply download and add a box with the command:

vagrant box add ubuntu/trusty64

This downloads the box and stores it locally.

2. Next, you need to configure the Vagrantfile for the virtual box it will serve. Open the Vagrantfile with the command:

sudo vi vagrantfile

3. Once the Vagrantfile is open, change the config.vm.box string from “base” to “ubuntu/trusty64”.

config.vm.box = "ubuntu/trusty64"

1. You can add another line above the end command to specify a box version:

config.vm.box\_version = “1.0.1”

Or you can specify a URL to link directly to the box:

config.vm.box\_url = “https://vagrantcloud.com/ubuntu/trusty64”

4.If you’d like to remove a box, use the following:

vagrant box remove ubuntu/trusty64

**Provisioning**

As the systems are not preloaded with software, we can also automate that. Vagrant supports automatic provisioning through a bootstrap.sh file saved in the same directory as the Vagrantfile. Provisioning gives you a powerful tool for pre-configuring your virtual environment

To update and install basic httpd service and apache,in the guest OS use the command:

sudo vi bootstrap.sh

In that file, enter the following:

#!/usr/bin/env bash

apt-get update

apt-get install -y httpd

sudo apt-get install apache2

fi

Save the file and exit.

Next, edit the Vagrantfile and add the provisioning line. It should look as follows:

Vagrant.configure("2") do |config|

config.vm.box = "ubuntu/trusty64"

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

end

When Vagrant reads the Vagrantfile, it is redirected to read the **bootstrap.sh** file we just created. That bootstrap file will update the package manager, then update and install httpd service.

**Networking**

Vagrant includes options to place your virtual machine on a network. At the end of your Vagrantfile, just before the end command, use the config.vm.network command to specify network settings.

For example:

config.vm.network “forwarded\_port”, guest: 80, host: 8080

For the changes to take place, save and reload Vagrant with the command:

vagrant reload

This creates a forwarded port for the guest system. You may also define private networks, public networks, and other more advanced options.

**Launching and Connecting**

1. Vagrant Up

The main command to launch your new virtual environment is:

vagrant up

This will run the software and start a virtual Ubuntu environment quickly. However, even though the virtual machine is running, you will not see any output. Vagrant does not give any kind of user interface.

2. Vagrant SSH

You can connect to your virtual machine (and verify that it is running) by using an SSH connection:

vagrant ssh

This opens a secure shell connection to the new virtual machine. Your command prompt will change to vagrant@trusty64 to indicate that you are logged into the virtual machine.

Once you are done exploring the virtual machine, you can exit the session with **CTRL-D**. The virtual machine will still be running in the background, but the SSH connection will be closed.

To stop the virtual machine from running, enter:

vagrant destroy

The file you downloaded will remain, but anything that was running inside the virtual machine is will be gone.

Once you are done working on your guest system, you have a few options how to end the session.

1. To stop the machine and save its current state run:

vagrant suspend

You can resume by running vagrant up again. This is much like putting the machine in sleep mode.

2. To shut down the virtual machine use the command:

vagrant halt

Again, vagrant up will reboot the same virtual machine, and you can resume where you left off. This is much like shutting down a regular machine.

3. To remove all traces of the virtual machine from your system type in the following:

vagrant destroy

Anything you have saved in the virtual machine will be removed. This frees up the system resources used by Vagrant. The next time you vagrant up, the machine will have to be re-imported and re-provisioned. This is much like formatting the hard drive on a system, then reloading a fresh image.