**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. Check if the installation was successful:

List two ways

1] .

2] .

**Project Setup**

1. Vagrant works on per-directory basis. Start by creating a new directory named “DevOps” to store your lab data and create a new repository inside it called as “lab2”. (write the command used below)

2. Initialize the vagrant with the box file name which will generate a Vagrant file.

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

**Vagrant Boxes**

The basic unit in a Vagrant setup is called a Vagrant box. 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.

**In our Lab we will be using "ubuntu/trusty64"**

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

This command downloads the box and stores it locally.

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

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

4.If you would like to remove a box what command will you use:-

**Networking**

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

For example:

**Defining a Forwarded Port**

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

**Private Networks**

Network identifier: private\_network

config.vm.network "private\_network", ip: "192.168.33.10"

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

This creates a forwarded port for the guest system and defines private network. you can also define public network, and other more advanced options.

Also see how we can Configure private network using DHCP

Which are three modes of network access to the guest machine.

1)

2)

3)

**Synced Folders**

Synced folders enable Vagrant to sync a folder on the host machine to the guest machine, allowing you to continue working on your project's files on your host machine, but use the resources in the guest machine to compile or run your project.

Synced folders are configured within your Vagrant file using the config.vm.synced\_folder method. Usage of the configuration directive is very simple:

config.vm.synced\_folder "src/", "/srv/website"

**Provisioning**

As the system are not preloaded with software, we can also automate that. Vagrant supports automatic provisioning through Vagrant file. Provisioning gives you a powerful tool for pre-configuring your virtual environment.

To update and install apache service in the guest OS

e.g

config.vm.provision "shell", inline: <<-SHELL

apt-get update

apt-get install -y apache2

SHELL

Attach all the screenshots of the task done in the lab.

Q) Show the Homepage of your webserver. If you have successfully installed the apache2, you will be able to see the default homepage for apache. This will conclude that your webserver is running on your guest virtual machine.

Q) Deploy Multi-machine topology using vagrant and paste all necessary screenshots.

Q) List all the important Vagrant commands and explain it in brief.  
Q) Explain the stages or the workflow in your own words.  
Q) Push the vagrant file to GitHub. Make a Readme file explaining the vagrant lab.

The **Wh rule????**

1. What did you do in this lab?
2. What problems did you encounter?
3. How did you solve this problem?
4. What aspects of the labs were unclear?