CLOUD COMPUTING (INGI2145)
LAB SESSION #1

WALEED REDA, NICOLAS LAURENT, MARCO CANINI

I. BACKGROUND

In this lab session, you'll be introduced to some well-known tools for automated creation and provision of development environments. One of these tools is called Vagrant – which allows us to automatically create VMs with relatively simple commands. Once the VM is up and running, we'll use another tool – called Puppet - with Vagrant to automatically install software and modify configuration. In order to run the created VM, we'll utilize a hypervisor software called VirtualBox. Our goals for this tutorial are two-fold:

1. Getting familiarized with automated provisioning tools for Virtual Machines (VMs)
2. Giving you the means to create a VM using a pre-defined script that you can use for working on your future homework assignments. This VM that you will provision will come preconfigured with all the software that you're going to require for your future lab sessions. This serves two purposes:
    a. The VM will serve as the 'gold standard' for grading; thus, if your solution works in the VM image, you can be sure that it will also work on the graders' machines.
    b. You can easily recreate the original VM if something goes awry with your configuration

II. INTRODUCING VAGRANT AND PUPPET:

Vagrant is a tool that provides an easy way to configure Virtual Machines (VMs). It allows for reproducibility of configuration and enhances productivity when managing a diverse set of VMs. Using Vagrant, development environments can be set-up with relative ease using simple commands. To configure VMs, Vagrant allows automated provisioning of VMs using either shell scripts or third-party tools such as Puppet and Chef. In our tutorial, we'll utilize Puppet as our go-to provisioning tool.

Puppet allows the automated provisioning of development environments by leveraging a declarative style programming language. Using the Puppet language, the system state can be expressed as a set of items - called "resources". These resources can describe a variety of system components, including but not limited to: Installed packages, user groups, running services, etc. The resources are described in files called manifests. Notice that, in this way, Puppet abstracts the implementation details from the users, allowing them to easily describe their system's configuration instead of requiring them to specify "how" to reach said configuration. These features stem from Puppet's Resource Abstraction Layer (RAL) – which is utilized to separate types (high-level models) from providers (platform-specific implementations). By splitting these two, users can specify system states that are largely platform-agnostic.

III. TUTORIAL

Now that we've described our intentions from this tutorial and briefly went over the tools, let's get started!

A. Setup and Installation:

In this sub-section we'll outline how to install our aforementioned tools. The necessary steps are as follows:

1- Head over to https://docs.vagrantup.com/v2/installation/ and install the latest version of Vagrant according to the guidelines in the aforementioned page as per your host operating system
2- Since Vagrant doesn't come pre-installed with Puppet, you'll need to install it separately. Go to https://docs.puppetlabs.com/guides/install_puppet/pre_install.html#check-os-versions-and-system-requirements, then download and install the latest version of puppet
3- Lastly, you'll need to download and install VirtualBox from the following link: https://www.virtualbox.org/wiki/Downloads

B. Test-driving the tools:

Now that we have everything set-up, we're going to show you how to utilize these tools to create your own VM. These next steps outline how to do so:

1- Open your preferred git application and clone the following repository **https://github.com/mcanini/INGI2145-2014/** to your local directory. Alternatively, if you have git installed, you can execute the following command in a terminal:
`git clone` **`https://github.com/mcanini/INGI2145-2014/`**
2- You'll find two main files in this repo:
   a. "**vagrantfile**": This basically holds the configuration for the initial creation of the VM. It comprises things such as VM memory allocation, which provisioning tools to use (in this case Puppet), network interfaces, etc.
   b. "**manifest/base.pp**": This puppet script file contains the provisioning code that describes the configuration and software that are going to be installed on the VM
3- Open up the terminal (or cmd if you're using Windows), and cd to your newly created repo.
4- Run the command `vagrant up` to create your very first VM. Note that this might take a while because this command downloads your box and then proceeds to provision your VM (as described in the puppet script file). If this command fails to run, then you've probably done something wrong with your vagrant installation.
5- A fully provisioned VM is now ready for use. The default password for your vagrant log-in account is also "vagrant".

IV. ADDITIONAL NOTES:

- You can use the following Vagrant commands to clean up your environment:
  - `vagrant suspend`: Will save the current machine's state and stop it. Requires more disk space but allows you to resume work faster
  - `vagrant halt`: Equivalent to a graceful shutdown. Takes more time to start from a cold boot, and the VM still consumes space (albeit lower than the previous command)
  - `vagrant destroy`: Will delete the guest machine from your system but will keep your box. VM itself will consume no diskspace at the cost of having to restart the provisioning process if you want to rebuild your VM.

- You can try to modify the puppet script yourself and see if you want to automatize any other configuration changes to your VM. Take a look at the puppet language tutorial here: https://docs.puppetlabs.com/puppet/latest/reference/lang_summary.html
  - If you're reading the puppet tutorial make sure you familiarize yourself with some of the core concepts of the language. Namely its declarative style, resource execution order as well as the idempotency requirement.
  - In order to reapply any puppet script changes to your VM you can use the command: `vagrant reload --provision`
  - Tip: Take snapshots of your machine using Virtualbox if you're planning on reapplying any configuration changes using puppet in case something goes wrong. You can do this either using VirtualBox, or through a vagrant plugin called https://github.com/dergachev/vagrant-vbox-snapshot