Cloud Computing (INGI2145) - Lab Session 2

Muhammad Bilal and Marco Canini

1. Background

Note: All of these tools (except Docker) are available in INGI systems.

In this lab session, you will use some well-known tools for automatically creating and provisioning development environments, namely Vagrant and Puppet.

Vagrant is used to create and manage Virtual Machines (VMs). Once we have a VM running, we will use another tool – called Puppet – to automatically deploy software and configurations on the VM. A subsequent lab session will cover the use of Vagrant and Puppet more in detail.

Vagrant and Puppet both use configuration scripts to know how to provision the VM. We supply you with pre-defined scripts to generate a VM that will serve as the gold standard for grading the projects.

Lastly, we will take a look at Docker which is used to run and manage containers that serve as environments to build and run applications in production environments. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

Note: You need a 64-bit system to work with Docker (unfortunately). If you have a 32 bit version with virtualization enabled hardware, you can create a 64-bit linux virtual machine and play with docker within it.

2. Provisioning the VM with Vagrant and Puppet

Note: These instructions assume you are working on a machine of your own, where you have permission to install new software. If you can only access the machine in the computer rooms, we can supply you with a pre-made VM to be run there.

1. Install Vagrant from https://www.vagrantup.com/downloads

2. Install VirtualBox from https://www.virtualbox.org/wiki/Downloads

3. Clone the course repository by running
   `git clone https://github.com/mcanini/INGI2145-2015/`
   If you don't have git, you can install it from http://git-scm.com/downloads.

The `Vagrant` directory in the lab 2 directory of the repository contains these files:

- `vagrantfile`: The configuration used by Vagrant to create the VM: memory, network interfaces, provisioning tools – such as puppet – to use, …

- `manifest/base.pp`: The script describing the configuration and software to be installed on the VM by Puppet.

4. Run `vagrant up` in the `Vagrant` directory. This will create the VM, launch it, then run Puppet to configure it and install all required software. Do not attempt to use the VM until the Vagrant returns the shell to you on the host OS.

5. If you receive an error with Vagrant up command "The VirtualBox VM was created with a user that doesn't match the current user …." make sure to delete the `.vagrant` folder in the directory that has vagrantfile in it and then run `vagrant up` again.

6. Log into the VM. The default login and password are both "vagrant" (beware the keyboard layout of the login screen).

Note that the `Vagrant` serves as a shared directory between the VM and your machine. On the VM, this directory is mounted under `/vagrant`.

This VM is for demonstration purposes. You will be provided with full fledge VM that will contain the tools that you would need to work on your project.

3. Introduction to Docker:

Docker allows you to package an application with all of its dependencies into a standardized unit for software development. Docker has slightly different support for Linux and Non-linux systems.

1. If you are using Windows (which I completely don't recommend) or Mac OS X, download the Docker toolbox (https://www.docker.com/toolbox) that has set of tools necessary to get started with Docker.
2. For Linux, use make sure that you have wget utility. If you do, run `wget -qO- https://get.docker.com/ | sh` to download and install docker. On Linux, by default you will only be able to run docker as root user (So make sure to add sudo before all the commands discussed next).

3. For linux, before using docker, start the docker service by `sudo service docker start`. For Mac and Windows open the Docker Quick start terminal application and use it as the command line terminal for all docker commands.
4. After the installation of Docker Toolbox is done, enter `docker run hello-world` in your terminal window, in order to confirm that the installation is working correctly. You will see the output that is a result of running a hello world program in a downloaded Docker container.
5. You can download and run other docker containers based on ubuntu images. Try `docker run -it ubuntu bash` or `docker run docker/whalesay cowsay boo`. These images are available on the docker hub (http://hub.docker.com) along with hundreds of other images created by individuals and companies.
6. Now we will try to create our own docker container (and upload it to docker hub).
7. Open the Dockerfile available in the `lab_sessions/2_vagrant_docker/Docker-container` folder and take a look at it. This is how docker images are created.
8. Create another folder called docker-task and copy the Dockerfile inside it.
9. You will modify this new Dockerfile to accomplish the task.
10. Your task:
    a. Create a docker image based on generic ubuntu image using `FROM ubuntu`
    b. Install docker on it using the RUN command and the necessary bash commands to use are given in Step 2 of Section 3.
    c. Create an account on hub.docker.com
    d. Create a public repository in your account at docker hub
    e. Tag the image with a name and build it using the command shown in Appendix B.
    f. Push your docker image (if you have created an account), using the command in Appendix B.

Appendix A: Additional Notes on Vagrant and Puppet

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 disk space 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, if you want to automate other configuration changes to your VM. Remember however that the original script is what we use for grading!

Take a look at the puppet language tutorial here:
https://docs.puppetlabs.com/puppet/latest/reference/lang_summary.html

A few more advices:

- Make sure to 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 apply any puppet script changes to your VM you can use the command:
  `vagrant reload --provision`

- If you plan to apply Puppet configuration changes to your VM, take a snapshot of the VM via Virtualbox or through a vagrant plugin:
  https://github.com/dergachev/vagrant-vbox-snapshot

Appendix B: Additional Notes on Docker

Docker build (and tag) command:
`docker build -t <hub-username>/<reponame> <Path>`
Path is where the “Dockerfile” exists, from which you want to build an image
-t flag with tag the new image as well.

Docker push command:
`docker push <imagename>`
The image name will be the name with which you have tagged your docker image i.e. <hub-username>/<reponame>

You can see all available images on your system using:
`docker images`