Cloud Computing (INGI2145) - Lab Session 1

Waleed Reda, Nicolas Laurent, Marco Canini

1. Background

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.

Then, using the created VM, you will learn how to create Amazon Elastic Cloud Computing (EC2) instance, how to connect to them, and how to manage them using the Amazon Web Services (AWS) Command Line Interface (CLI).

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 Puppet from https://docs.puppetlabs.com/guides/install_puppet/pre_install.html#next-install-puppet
   You don't need Facter or Hiera.

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

4. Clone the course repository by running
   `git clone https://github.com/mcanini/INGI2145-2014/`
   If you don't have git, you can install it from http://git-scm.com/downloads.

The `INGI2145-vm` 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.

5. Run `vagrant up` in the `INGI2145-vm` 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.

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

Note that the `INGI2145-vm` serves as a shared directory between the VM and your machine. On the VM, this directory is mounted under `/vagrant`.

3. Creating and Managing Amazon EC2 Instances

These steps can be followed from the VM created in last section, or from your own machine, in which case you will need to have the AWS CLI installed (http://aws.amazon.com/cli/), which in turns requires Python and pip (a python package manager) to be installed.

- Log in on the AWS console: https://mcanini.signin.aws.amazon.com/console/
  (use this exact URL or it won't work)
  Account: `mcanini`
  User Name: `ingi2145-<AWS username>`
  `AWS username` is the username you provided during the course sign up
  Password: supplied to you during this session

- As shown in the demo, go to Identity and Access Management (IAM) and create/download a new access key for yourself.

- Run `aws configure` and enter your key information, the default region (`us-west-2`) and the output format (`table`). Verify everything is working by running
  `aws ec2 describe-instances`
  The output should be empty or contain your friends' instances, if you're slow.

- Launch an EC2 instance via the console as shown in the demo.
  Things to keep in mind:

    - Create a micro instance.
    - Tag your instance (`key=“user” value = <AWS username>`).
    - Use a magnetic disk.
    - Use the SSH Only INGI2145 security group.
    - After clicking on “Launch”, create and download a new key pair named after your user name (`ingi2145-<AWS username>`).
    - You will need to change the permission of the key pair:
      `chmod 400 ingi2145-<AWS username>`

- Look up your instance’s public IP, then connect to it via:
  `ssh -i <your_key>.pem ec2-user@<ip>`
  e.g., `ssh -i ingi2145-xyz.pem ec2-user@54.69.32.120`
  Then exit (`exit`).

- Create a new EBS volume, either from the AWS console (as shown in the demo), or via the CLI (if you’re a real macho (wo)man).
    - Hint: `aws ec2 create-volume help`
    - Hint: use the `--dry-run` option to verify your command.
    - Set the size to 1GB, and the storage type to magnetic.
    - Your volume must be created in the same availability zone as your instance.

- Attach the volume to the instance:
  ```
  aws ec2 attach-volume --device /dev/sdf
       --instance-id <iid> --volume-id <vid>
  ```
  e.g.,
  ```
  aws ec2 attach-volume --device /dev/sdf
       --instance-id i-705b4c7b --volume-id vol-efa7eaee
  ```
  The device name matters!
  (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-attaching-volume.html)

- Re-connect to VM via SSH, then format the volume and mount it:
  ```
  sudo mkfs -t ext4 /dev/xvdf
  sudo mkdir /mnt/sdf
  sudo mount /dev/xvdf /mnt/sdf
  ```

- Make sure you can read and write the volume via `/mnt/sdf`. Then unmount it and exit the VM.
  ```
  sudo umount /mnt/sdf
  exit
  ```

- Detach the volume from the instance:
  ```
  aws ec2 detach-volume --volume-id <vid>
  ```

  At this point, the volume still exists (as you can tell from the console or from `aws ec2 describe-volumes`). We won't do it, but we could reattach it to the same instance or to another and read what was written on it.

- Instead, delete the volume and terminate your instance (it shouldn't be too hard to figure for a brilliant bunch like you). Take care not to delete your friends' volumes/instances (or they won't stay your friends for long).

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 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, 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