Web Services and Cloud-Based Systems

Assignment 2.1

Tom Peerdeman & Laurens Verspeek
10266186 & 10184465

April 24, 2015

1 Contextualize a VM

In order to contextualize a Virtual Machine (VM) such that it automatically deploys our URL Shortner Web service (developed for Assignment 1.1) at start up, we used OpenNebula. We first had to create a VM. Personally we don't like CentOS, so we decided to use Ubuntu instead. To do this we downloaded a special cloud image of Ubuntu 14.10 from the Ubuntu website (https://cloud-images.ubuntu.com). With the use of the tool *oneimage* we created an image of Ubuntu. To load this image we had to change the driver in the vm template to qcow2. Before we created the VM from this image with the tool *onevm*, we made the image of Ubuntu persistent, so any changes we would make to the VM would be saved in the image of Ubuntu on exit. In order to be able to contextualize the VM, we installed one-context package. This package makes the cloud image compatible with the opennebula contextualization. We register an init script as a image with oneimage and we add this image to our VM template file and it will mount the init script on startup. In the same VM template we specify which (previously mounted) script it needs to run at startup. In the VM template, we also added our own public SSH key so we can easily connect to the VM with SSH.

The init script first installs all the necessary packages for the URL Shortner Web service: apache2, php5 and the sqlite library for php5. The scripts then adds the sqlite extension to the php.ini file in order to enable sqlite. Then the script enables the rewrite module for apache and creates a file with the configuration for the rewrite module (because it needs access to the folder where our service will be placed). Next, the script clones the files needed for the service from our github. To avoid the question about adding github.com to our permanent hosts, the script does that himself before cloning. Now we copy the files needed for the URL shortner service to /var/www/html and give the user www-data access to this folder. Finally, the init script restart apache and the service is up and running!

To test the URL shortener service running in the vm we used the provided ruby test script (https://github.com/SOA-cloud-course/rest). Since the VM uses an internal ip, the service is not accessible from the outside by default. One solution is to run the test script on the das4 head node (fs2.das4.science.uva.nl). However while ruby is available, the scripts dependencies are not. Also we are not allowed to install these dependencies, thus we can't run it on the head node. The second option is to tunnel the port trough an ssh function. Using the command
ssh -L9001:vm_ip:80 clda1507@fs2.das4.science.uva.nl
a listening socket on port 9001 is created locally. Any data sent to this socket is tunneled trough ssh, and arrives at the vm at port 80. Using this we can run the test locally with the url http://localhost:9001. The results are as follows:

```
Run options: --seed 15911

# Running:

........

Finished in 0.217556s, 36.7722 runs/s, 151.6853 assertions/s.

8 runs, 33 assertions, 0 failures, 0 errors, 0 skips
```