Web Services and Cloud Based Systems - Assignment 2

Arumoy Shome
VU ID: 2636393

Shruti Rao
VU ID: 2631481

Group 13

Assignment 2.1

Contextualization:

For contextualization (init.sh), we first updated the Ubuntu package manager - `apt`. As the url shortner application is written in Python, the Python package manager `pip3` was installed. `pip3` was then used to install the required dependencies - `flask` and `short-url`. Variables required by `flask` were exported as `bash` variables. The url shortner application was copied to the home directory. Finally, the database was initialized and the service was started.

As the Network Template could not be generated due to permission issues, we are giving an overview of what it would be. Typically, the Virtual Network template would be used to instantiate the Virtual Network for the Virtual Machine. It would be assigned a name (`NAME`) to be referenced by, and a driver to implement it (`VN_MAD`). Next, Address Range available for virtual network (`AR TYPE`) would have to be defined and set to IPv4 and have a range of 10 networks (`AR SIZE`). Finally, `CLUSTERs` would be set to 0 as we don't require the network to deploy to multiple clusters.

```
NAME=apptemplate
VN_MAD="bridge"
AR=[
      IP="10.0.0.1",
      SIZE="10",
      TYPE="IP4" ]
CLUSTERS=0"
```

Problems with Debugging:

Overall, debugging the `init.sh` shell script was hard because there was no clear way to see the errors. Scripts were executed in the background with no visual feedback of error or success. Moreover, we initially had shell commands in our script such as `mv`, that were forbidden. However, without any error messages or user feedback, detecting them were not easy. Additionally, installing large packages took a while. Without any feedback of installation, we assumed them to have failed. It took some time to realize that we simply had to wait a while for them to finish. To combat some of these problems, we devised a simple logging system where we recorded the success of each command being executed. This made the process a bit more easier.

Experience with OpenNebula and Virtual Machines:

We had to modify our initial methodology to make the service run. Our initial process involved installing `pip3` and python virtual environment - `pipenv`. Out initial `init.sh` script tried to clone the github repository, `cd` into the required folder and start a shell withing the `virtualenv`. This did not work and we were unable to understand why. Thus, we had to resort to passing the url app via the template.

We found the OpenNebula documentation extensive and helpful. The API was intuitive and easy to work with. As OpenNebula was already set up on the cluster, getting started was very easy. We noticed that modifying the template required us to delete and instantiate the virtual machine each time. This was cumbersome.