Applied Cloud Computing Lab 2

Submitted By: Yuvarani Masarapu

Task 1:

Part 1

A) Time taken:

```
real    0m25.736s
user    0m25.700s
sys     0m0.020s
```

User+sys is basically the total CPU time used by the process.

Yes, the vm is slower than the physical machine, reason being that the clock speed (the 'real' time as in here) differs for the vm and the physical machine, it is basically shared between the two hence lesser for the vm than the physical machine.

Alternative to vms: Containers

For the containers, though the different 'sub-servers' are isolated and work independently, they have the same kernel backend which makes them so different from a virtual machine. Although, the similarity being they have their independent IP and root access.

Part 2

Task 0: Setting up API access environment

Ans 1) The version of API being used is as shown in the screenshot.

```
root@yuvi-a1:~/Applied-Cloud-Computing# openstack --version
openstack 3.14.2
```

Ans 2) The Openstack Networking service is responsible for having it's component processes mainly the neutron server to interact with the others for taking care of requests.

Each component of the networking service like the neutron server, the plugin agent, the DHCP agent, the L3 agent and the SDN services work together in providing the services that a tenant has requested for. Each component accomplish their tasks through different plugins and make sure that the tenant is able to communicate with the database, by giving indirect virtual access to the database using some advanced protocols like the AMQP, maintaining the Ip configurations etc.

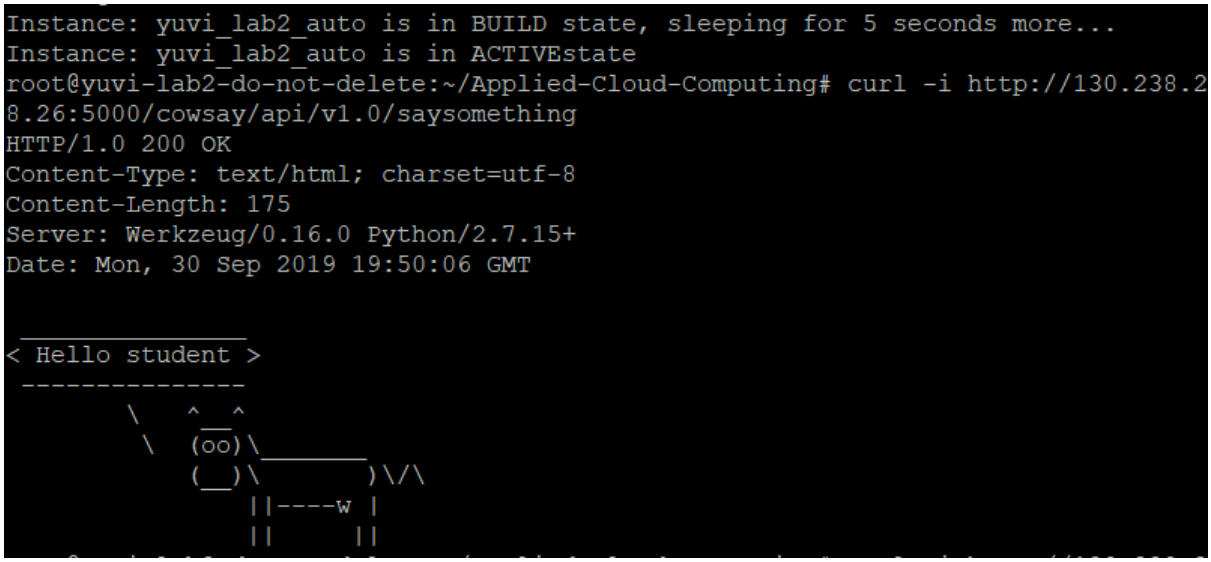
```
root@yuvi-lab2-do-not-delete:~/Applied-Cloud-Computing# openstack network list
+--------------------------------------+--------------------------------------
------------+--------------------------------------+
| ID                                   | Name
            | Subnets                              |
+--------------------------------------+--------------------------------------
------------+--------------------------------------+
| 2f1cb666-ee9d-4dbd-b351-732c5463fde4 | stack_with_init_script-private_network-
6347ifw3m7ha | a95c6d10-66cb-4301-bfe7-38510be2db0f |
| abaac4a6-7c1e-41cd-86ba-bf54219770d2 | SNIC 2019/10-32 Internal IPv4 Network
            | 6b82cc92-e021-4652-9c3c-ecc59a55d844 |
| af006ff3-d68a-4722-a056-0f631c5a0039 | Public External IPv4 network
            | f864d83e-9124-4439-8b53-4a4f2dc77cd5 |
+--------------------------------------+--------------------------------------
------------+--------------------------------------+
```

Ans 3) There is something called EC2 API that has come as replacement to the nova EC2-API service in OpenStack and is perfectly compatible with it and hence made the communication with OpenStack possible.

S3 API is object storage and for openstack to communicate with it, it is possible but some configurations are required to be done. These are listed in the openstack documentation as well. Swift3 middleware is used to enable this object storage communication.

Task 2: Contextualization

Ans 1) After creating an instance using cloudinit configurations, one will be able to execute the curl command in their physical machine. The output is basically same as what received when we installed and ran cowsay in our vms working on lab 1.



The output displays the cowsay api with the message that was input into the cloud-cfg.txt file.

Ans 2) Contextualization is the practical concept of making the requested data (requested to the cloud API by the user) available to the user within an instance of the virtual machine.

Ans 3) Cloudinit configurations are basically written in a data interchangeable type for language format known as the YAML format which is human readable and easy to understand.

Ans 4) Cloudinit package variants depends on it's configuration done by the user data which can be of 8 types: Gzip compressed content, MIME archive, user-data script, include file, cloud config data, upstart job, cloud boothbook and part handler.

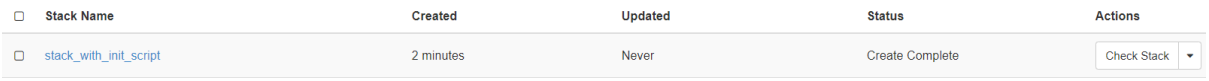
Ans 5) Cloudinit scripts can be run without booting an instance but that is basically for the configurations which will not take effect until the instance boots.

Ans 6) Limitations of cloudinit:

Whenever instance is restarted, the public IP address changes that can create external traffic routing issues. Also, data stored in the data volume of instance is wiped off, backup is required.

Task 4: Orchestration using Heat

```
root@yuvi-lab2-do-not-delete:~/Applied-Cloud-Computing# openstack stack create s
tack_with_init_script -f yaml -t ssc-test-stack.yaml
id: f57935d0-a739-4a1b-ab92-d6e420884cea
stack_name: stack_with_init_script
description: Simple template to deploy a single compute instance
creation_time: '2019-09-30T17:40:52Z'
updated_time: null
stack_status: CREATE_IN_PROGRESS
stack_status_reason: Stack CREATE started
```



Stack Name	Created	Updated	Status	Actions
stack_with_init_script	2 minutes	Never	Create Complete	Check Stack

Ans 1) Heat service uses a template named HOT which is written in human readable YAML.

Ans 2) Advantages of templates over API:

- Version control
- Human readable hence easy to configure
- Assignment of meaningful names to infrastructure resources. Relationships can be defined properly making it easy when a correct order of such resources are required to complete the application.
- Easy and quick modifications can be made to stacks.
- Better integration between hardware and software

Ans 3) The most basic heat template has the ‘heat_template_version’, which is a valid version of HOT, an optional ‘description’ of what the template can do, a ‘resources’ section which has instance definitions (type) and their value parameters ex. Key_name, image, flavor etc.

Key name, image and flavour are to be present obligatory when using a template.

Task 5: Linux Containers Introduction

```
root@yuvi-lab2-do-not-delete:~# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
   Active: active (running) since Mon 2019-09-30 17:56:36 UTC; 55s ago
     Docs: https://docs.docker.com
 Main PID: 5740 (dockerd)
    Tasks: 8
   CGroup: /system.slice/docker.service
           └─5740 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contain

Sep 30 17:56:34 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:34
Sep 30 17:56:34 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:34
Sep 30 17:56:34 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:34
Sep 30 17:56:34 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:34
Sep 30 17:56:35 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:35
Sep 30 17:56:35 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:35
Sep 30 17:56:36 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:36
Sep 30 17:56:36 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:36
Sep 30 17:56:36 yuvi-lab2-do-not-delete systemd[1]: Started Docker Application C
Sep 30 17:56:36 yuvi-lab2-do-not-delete dockerd[5740]: time="2019-09-30T17:56:36
lines 1-19/19 (END)
```

```
root@yuvi-lab2-do-not-delete:~/Applied-Cloud-Computing/container# docker run -i
cowsay
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 181-363-290
```

```
C:\WINDOWS\system32>curl -i http://130.238.28.26:5000/cowsay/api/v1.0/saysomething
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 175
Server: Werkzeug/0.16.0 Python/2.7.15+
Date: Mon, 30 Sep 2019 18:37:46 GMT

 _______________
< Hello student >
 ---------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Ans 1) Containers provide a virtual environment by running on top of the OS/kernel level instead of the hardware below it. There is the benefit of different virtual machines to run independent of each other without any time or space delay.

We can say that these come under **OS virtualization.**

Ans 2) Container frameworks:

1. Warden by Cloud Foundry
2. LXD by Canonical
3. Mesos and Marathon by Twitter
4. Kubernetes by Amazon

Ans 3) Dockerfile contains all the commands that are needed to be called when building that particular image.

It works by running each command or instruction in succession by the docker build.

The first two commands updates and upgrades all the packages so that the system is up to date.

Commands in lines 3,4,5 and 6 installs git, python-pip, flask and upgrades them. These are required before installing the cowsay service. Git is installed because we are cloning the csaas folder from github repository for the cowsay application.

The next commands takes /csaas/cowsay as the workingdirectory which is cloned.

Expose 5000 is done inoredr to open port 5000.

The last two commands basically run the python file app.py found under the path /usr/games/ of the vm.

Ans 4) The command docker build -t cowsay:lastest . builds an image from the available dockerfile (takes the instructions from this file) and the context given in the specified location (in our case . means the current directory)

The docker run -I cowsay command executes commands on the current image and commits the results.

Ans 5) dockerhub is a service that makes possible the management and sharing of containers between members of a team or between different teams.

Team members can login to dockerhub, create a repository with a team name.

They can check image ID under their login, can also tag their images and push them.

Ans 6) The cloudinit script is submitted with this report.