Description

Initially, we'd like you to set up a private cloud using OpenStack-Ansible All-In-One, described here. You need to deploy OpenStack on its latest stable release (currently, *Victoria*) and your deployment must include OpenStack base services plus Octavia and Heat. After a successful deployment, try to log in with the *admin* credentials (password is provided in the *keystone_auth_admin_password* field of *user_secrets.yml* file) and check *System Information* for the status of OpenStack services.

You own a *private* cloud and clouds are limitless in nature, but obviously that doesn't mean you have actual infinite resources! So, to protect your *Quality of Service*, make sure that each project in your cloud cannot consume more resources than the following thresholds: *2 Instances, 4 CPU Cores* and *4 GBs of RAM*.

Now, you need to create a project called cloud and a user with the combination of your names and set the *cloud* project as its Primary Project with the role *member*. Before logging out from the *admin* user, specify the custom quota for this project as follows: *6 Instances, 8 Cores of CPU* and *12 GBs of RAM*.

Our *ultimate* goal is to orchestrate the deployment of two applications in our cloud. We want to deploy them into a non-containerized three-node cluster with a dedicated node for the database. We will expose our apps to the outside through a load-balancer. The details are specified in the following sections.

Networks

Start off by logging in to your new user account; create a *nonshared* network called auth-app with the CIDR of 192.168.65.0/26. Then, create a *router* called cloud-router that connects the *auth-app* network to the outside world. Note that our instances should not be accessible to the public at all. The load balancer and the instances need to be on the same network to communicate; however, the load-balancer must also serve our apps to the outside world.

General Specifications

You are free to use your choice of Linux distro for either of the apps or database nodes. If you are new to the Linux, we recommend the latest stable release of Ubuntu Server (Currently, *Ubuntu Server 20.04*). Each app node should have *1 CPU Core, 1 GB of RAM and 5 GBs of Storage*. The database node has the same configurations except a *Storage of 20 GBs*.

On the creation of the instances, add the OpenStack host *root* SSH key and set password for the *root* user. The root password could be used when you cannot access the instance via the

network; in that case, you may log in via the console. Make sure that the SSH service on instances allows incoming requests only through SSH key.

Database

One of the two services that you are going to deploy needs a Mongo database. Create an instance, install and secure the Mongo database. It needs to start on boot and bind to its IP address on the *auth-app* network. For reasons you'll find out later, this instance needs to have a static IP address. Finally, make sure that only requests from the *auth-app* network reaches your Mongo database (*do not set rules inside the machine*).

Applications

The two applications that you will deploy are Authentiq and Basic System Info. The first is a *JWT-based* authentication service that depends on a Mongo database and the second displays a little information about the host that it is running on (it comes handy when dealing with the load-balancer). Both of these services are written in node.js and it is a good idea to start by installing node and npm. The Authentiq app requires some configurations which are explained thoroughly in its *ReadMe* page. You need to configure them to start on boot and they should bind to 127.0.0.1 only. Next, expose the apps only to the load-balancer via domain names by using a reverse-proxy software. If you are starter, we recommend Nginx. Publish the *System Info* app on the http://fumcloud.pro domain and the *Authentiq* app on http://api.fumcloud.pro subdomain. Set the Authentiq virtual server as default (figure out why this is necessary). Finally, note that we need three replicas of this node, configured in the same way.

Load-balancer

Create a load-balancer that listens on port 80 for *HTTP requests* and forwards them towards our cluster using round-robin algorithm with equal weights. The load-balancer should monitor the state of the cluster using the Heartbeat API of the *Authentiq* app.

Orchestration

So far, you have done everything manually. Now, it is time to use the OpenStack's orchestration engine, AKA Heat, and deploy your app into your cloud from scratch with the push of a button. Your main task in this project is writing a Heat template that satisfies every requirement mentioned earlier. Your template should include the creation and configuration of everything except the parts before the *Network* section and uploading a raw OS. It is very important to write your template as modular as possible. Please note that although the Heat graphical *Template Generator* is a good point to start, it has many limitations and therefore you should

use the OpenStack CLI to deploy your template. To use the OpenStack CLI, prepare and *RC* file for your user and connect to the *utility container*. To learn about the Resource Types in a Heat templates refer to this document.

Access the Cluster

In a production environment, the IP address of the load-balancer is a *valid* IP address published on the internet and you can easily access your apps. However, in this lab deployment, the public network IP addresses are not really public and can only be accessed on the OpenStack host machine. You need to figure out a way to access your apps with their *domain names* from your computer. There are a lot of solutions to this requirement, but if you do not have a preference, I'd recommend SSH Dynamic Port Forwarding and a static DNS record on *OpenStack Host* machine.