OpenStack AutoScale Lab Guide

- Heat, Ceilometer, LBaaS

Document Type	**Lab Guide**
Status	**Draft**
Document Version	**1.4**
Date:	**May 15, 2015**

Table of Content

1. Objectives

Version 0.1, Kilo Stable Release, April 26, 2015

This guide is Open Source under the Apache 2.0 License Agreement and only uses or makes reference to other Free and Open Source Software.

The guide is broken down into the following sections:

- Lab Environment Setup
- Lab Topology
- Lab 1 - Simple Server
- Lab 2 - Load Balancer
- Lab 3 - Autoscale Load Balancer

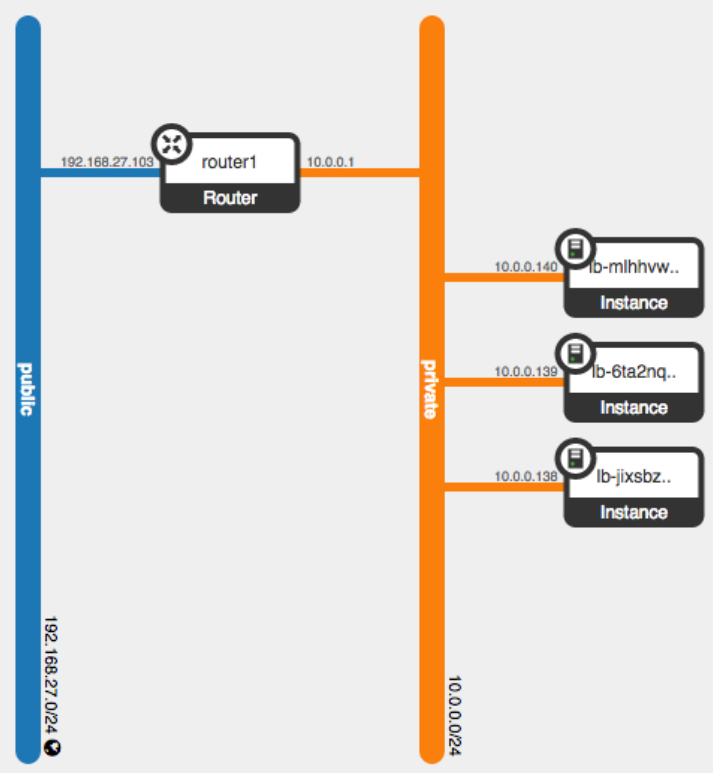
2. Lab Environment Setup

Covered by speaker in the presentation

3. Lab Topology

In the entire lab exercise, we are going to deal with two Neutron networks. One is facing to the external network called "public" and another represents tenant network, which is called "private". Each Neutron network is associated with a single subnet, "public-subnet" and "private-subnet" respectively. The "public-subnet" is mapped to 192.168.27.0/24 and the "private-subnet" is mapped to 10.0.0.0/24. Routing between the public network and the private network is done by Neutron router "router1".

Figure 1: Lab Topology



4. Lab 1: Simple Server

The primary goal of this lab is to make sure the audience become familiar with the Heat Orchestration Template (HOT) template. By the end, we will spawn a simple server by using a hot template. which performs the following actions:

Creates a VM port, assigns a fixed ip from the private network and applies security group rules

Creates a floating ip on the public network and maps it to the fixed ip

Creates security group to allow inbound HTTP request

Inject a ssh key

1. Verify the environment variables
   A. vagrant@devstack:~$ export OS_TENANT_NAME=demo
   B. vagrant@devstack:~$ export OS_USERNAME=admin
   C. vagrant@devstack:~$ export OS_PASSWORD=stack

D. vagrant@devstack:~$ export OS_AUTH_URL=http://192.168.33.2:5000/v2.0

2. Execute the following commands to ensure OpenStack system has an image called “cirros-0.3.4-x86_64-uec”

   A. vagrant@devstack:~$ nova image-list

3. Execute the following commands to ensure OpenStack system has a flavor called “m1.tiny” with vCPU number equal to 1, memory size set to 128MB and disk size set to 1GB

   A. vagrant@devstack:~$ nova flavor-list

4. Run the following command to obtain public network, private network and private subnet information from Neutron

   A. vagrant@devstack:~$ neutron net-list

5. You can find a few hot templates in the following directory:

   A. vagrant@devstack:~$ cd autoscale/hot/

6. Copy those YAML files to home directory

   A. vagrant@devstack:~/autoscale/hot$ cp *.yaml ~/

   B. vagrant@devstack:~/autoscale/hot$ cd

7. Run the command shown below to create a stack called “simple-server” using “simple-server.yaml” file.

   heat stack-create simple-server -f simple-server.yaml \

   -P "key_name=vagrant;\

   node_name=simple-server-1;\

   node_server_flavor=m1.tiny;\

   node_image_name=cirros-0.3.4-x86_64-uec;\

   floating_net_id=<public_net_id>;\

   private_net_id=<private_net_id>;\

   private_subnet_id=<private_subnet_id>"

7. Use “heat stack-list” command to ensure the “simple-stack” is in “CREATE_COMPLETE” state

   A. vagrant@devstack:~$ heat stack-list

8. Verify a VM instance called “simple-server-1” is created as a result of the stack creation and the status of the VM should be “ACTIVE”

   A. vagrant@devstack:~$ nova list

9. Verify the VM instance obtains both fixed IP (in 10.0.0.0/24) and floating IP (in 192.168.27.0/24) addresses from Neutron.

   A. vagrant@devstack:~$ neutron floatingip-list

10. Ping the VM instance’s floating IP address

    A. vagrant@devstack:~$ ping <floating-ip-of-vm>

11. Remove the “simple-server” stack

    A. vagrant@devstack:~$ heat stack-delete simple-server

5. Lab 2: Load Balancer

The objective of this lab is to launch a load balancer by using Heat Template. The load balancer will have a floating IP address on the public network for external access. When incoming HTTP requests hit the floating IP address, load balancer distribute them across multiple web servers sitting on private network.

5.1. Part 1: Load Balancer Without Pool Member

The first part of the lab exercise will create a HAProxy based load balancer pool for HTTP protoocl with VIP and health monitor. Please note that, by the end of Part 1, the load balancer will not have any pool member (i.e. web server) yet.

The HEAT template used in Lab 2 Part 1, load-balancer.yaml, can be summarized as the following major steps:

Creates and configures a load balancer pool with ROUND_ROBIN policy over http

Creates a vip from the internal subnet and assigns it to the loadbalancer pool

Creates a floating ip and associates it with the vip of the loadbalancer pool

Creates a health monitor and associates it with the loadbalancer pool

1. Run the following command to obtain public network, private network and private subnet information from Neutron

   A. vagrant@devstack:~$ neutron net-list

2. Use the command shown below to create a stack called “load-balancer” using “load-balancer.yaml” file.

   heat stack-create load-balancer -f load-balancer.yaml \

   -P "floating_net_id=<public_net_id>;\

   private_subnet_id=<private_subnet_id>"

3. Issue “heat stack-list” to ensure the “load-balancer” stack is in “CREATE_COMPLETE” state

   A. vagrant@devstack:~$ heat stack-list

4. Use the following command to verify the load balancer pool is created and its status is “ACTIVE”

   A. vagrant@devstack:~$ neutron lb-pool-list

Tip: You can see we create a HAProxy based HTTP load balancer. The load balancing method is set to Round Robin.

5. Obtain the VIP of the load balancer pool and make sure its status is “ACTIVE”. The VIP should be in the private subnet range (i.e. 10.0.0.0/24)

   A. vagrant@devstack:~$ neutron lb-vip-list

6. List the matching floating ip in public subnet range (i.e. 192.168.27.0/24)

   A. vagrant@devstack:~$ neutron floatingip-list

7. Find out the member of this load balancer pool

   A. vagrant@devstack:~$ neutron lb-member-list

Tip: Do you see any pool member?

8. Verify the connectivity to the VIP via its floating IP address:

   A. vagrant@devstack:~$ ping <floating-ip-of-lb-vip>

5.2. Part 2: Load Balancer with Pool Members

In Lab 2, Part 2, we will leverage our learnings from Lab 1 "Simple Server" to build several web servers (simulated by a script) running on cirros VMs and add them to the load balancer pool created in Lab 2, Part 1. This should complete the configuration for a typical load balancer without auto-scaling capability.

1. Run the following command to obtain public network, private network and private subnet information from Neutron

   A. vagrant@devstack:~$ neutron net-list

2. Use the following command to obtain the load balancer pool id:

   A. vagrant@devstack:~$ neutron lb-pool-list

3. Use the command shown below to create a stack called “lb-members” using “lb-members.yaml” file.

   heat stack-create lb-members -e environment.yaml -f lb-members.yaml \

   -P "key_name=vagrant;\

   node_name=lb-member;\

   node_server_flavor=m1.tiny;\

   node_image_name=cirros-0.3.4-x86_64-uec;\

   floating_net_id=<floating_net_id>;\

   private_net_id=<private_net_id>;\

   private_subnet_id=<private_subnet_id>;\

pool_id=<lb_pool_id>;\

initial_capacity=2;\

asg_group_min_size=1;\

asg_group_max_size=3"

4. Use “heat stack-list” command to ensure the “lb-members” stack is in “CREATE_COMPLETE” state
   A. vagrant@devstack:~$ heat stack-list
5. Check whether two VM instances are spin up properly. Both should be in “ACTIVE” status
   A. vagrant@devstack:~$ nova list
6. Verify the members of the load balancer pool again
   A. vagrant@devstack:~$ neutron lb-member-list

Tip: This time, you should see two pool members

7. Both load balancer pool members also have floating ip address assigned
   A. vagrant@devstack:~$ neutron floatingip-list
8. Ssh into one of the VM instances by using its floating ip address in public subnet (i.e. 192.168.27.0/24)
   A. vagrant@devstack:~$ ssh cirros@<floating-ip-of-vm>
9. Use the following script to simulate a web server inside the VM instance
   A. $ MYIP=$(ifconfig eth0|grep 'inet addr'|awk -F: '{print $2}'| awk '{print $1}')
   B. $ while true; do echo -e "HTTP/1.0 200 OK\r\n\r\nWelcome to $MYIP" | sudo nc -l -p 80 ; done&
10. Exit from the VM instance
    A. $ exit

11. Repeat the above three steps on another VM instance
12. Send HTTP GET request to load balancer's VIP by using its floating ip address in public subnet (i.e. 192.168.27.0/24)
    A. vagrant@devstack:~$ curl -X GET http://<floating-ip-of-lb-vip>

Tip: You should see HTTP response like "Welcome to x.x.x.x", where x.x.x.x is the IP address assigned to one VM instance as part of load balancer pool

13. Send HTTP GET request to load balancer's VIP by using its floating ip address again
    A. vagrant@devstack:~$ curl -X GET http://<floating-ip-of-lb-vip>

Tip: You should see HTTP response like "Welcome to x.x.x.x", where x.x.x.x is the IP address assigned to another VM instance as part of load balancer pool

14. Remove the "lb-members" by:
    A. vagrant@devstack:~$ heat stack-delete lb-members

6. Lab 3: AutoScale Load Balancer

In Lab 3, we will expand our experience to achieve the final goal of this workshop: building a load balance pool, which can automatically scale up and down based on the alarm fed from Ceilometer. The exercise outlined below primarily focuses on Heat and Ceilometer integration.

1. Run the following command to obtain public network, private network and private subnet information from Neutron
   A. vagrant@devstack:~$ neutron net-list
2. Use the following command to obtain the load balancer pool id:
   A. vagrant@devstack:~$ neutron lb-pool-list
3. Use the command shown below to create a stack called "lb-autoscale-members" using "lb-autoscale-members.yaml" file

   heat stack-create lb-autoscale-members -e environment.yaml -f lb-autoscale-members.yaml \

-P "key_name=vagrant;\

node_name=lb-autoscale-member;\

node_server_flavor=m1.tiny;\

node_image_name=cirros-0.3.4-x86_64-uec;\

floating_net_id=<floating_net_id>;\

private_net_id=<private_net_id>;\

private_subnet_id=<private_subnet_id>;\

pool_id=<lb_pool_id>;\

initial_capacity=1;\

asg_group_min_size=1;\

asg_group_max_size=3;\

cooldown_policy_seconds=60"

4. Use “heat stack-list” command to ensure the “lb-autoscale-members” stack is in “CREATE_COMPLETE” state

   A. vagrant@devstack:~$ heat stack-list

5. Check whether a single VM instance is spin up properly. It should be in “ACTIVE” status

   A. vagrant@devstack:~$ nova list

6. Verify the member of the load balancer pool

   A. vagrant@devstack:~$ neutron lb-member-list

7. The load balancer pool member should also have floating ip address assigned

   A. vagrant@devstack:~$ neutron floatingip-list

8. Ssh into the VM instances by using its floating ip address in public subnet (i.e. 192.168.27.0/24)

   A. vagrant@devstack:~$ ssh cirros@<floating-ip-of-vm>

9. Use the following script to simulate a web server inside the VM instance

   A. $ MYIP=$(ifconfig eth0|grep 'inet addr'|awk -F: '{print $2}'| awk '{print $1}')
   B. $ while true; do echo -e "HTTP/1.0 200 OK\r\n\r\nWelcome to $MYIP" | sudo nc -l -p 80 ; done&

10. Exit from the VM instance
    A. $ exit
11. Send HTTP GET request to load balancer's VIP by using its floating ip address again
    A. vagrant@devstack:~$ curl -X GET http://<floating-ip-of-lb-vip>/
12. List existing Ceilometer alarms
    A. vagrant@devstack:~$ ceilometer alarm-list

Tip: How many ceilometer alarm do you see?

13. List existing load balancer pool and its name
    A. vagrant@devstack:~$ neutron lb-pool-list
14. View statistics for the load balancer pool
    A. vagrant@devstack:~$ neutron lb-pool-stats <lb-pool-name>
15. List Ceilometer meters which is related to the number of total connections for the load balancer pool:
    A. vagrant@devstack:~$ ceilometer meter-list | grep network.services.lb.total.connections
16. Return the most recent 5 samples for the meter "network.services.lb.total.connections.rate"
    A. vagrant@devstack:~$ ceilometer sample-list -m network.services.lb.total.connections.rate -l 5
17. Open another terminal window and log into the devstack VM
    A. vagrant ssh
18. Use ab (Apach Benchmarking) tool to generate HTTP requests to load balancer via its floating ip address

A. ab -n 1000000 http://<floating-ip-of-lb-vip>/

19. Track the value changes for the meter “network.services.lb.total.connections.rate” every minute

    A. vagrant@devstack:~$ ceilometer sample-list -m network.services.lb.total.connections.rate -l 5

20. Check alarm state change every minute

    A. vagrant@devstack:~$ ceilometer alarm-list

21. When a new alarm is raised (i.e. state transition from “ok” to “alarm”), verify whether a new VM instance is spin up after cool-down period

    A. vagrant@devstack:~$ nova list

22. Stop Apache Benchmarking (ab) tool. Don't forget to keep tracking the value changes for the meter “network.services.lb.total.connections.rate”

    A. vagrant@devstack:~$ ceilometer sample-list -m network.services.lb.total.connections.rate -l 5

23. Check alarm state change every minute

    A. vagrant@devstack:~$ ceilometer alarm-list

24. When an existing alarm is off (i.e. state transition from “alarm” to “off”), verify whether a VM instance is removed after cool-down period

    A. vagrant@devstack:~$ nova list

Tip: You may see three VM instances at this moment. However, HEAT will scale down and eventually there is only one VM instance left.

End of OpenStack AutoScale Lab