Load Balancer in OpenStack

Introduction of Load Balancer

Load Balancing distributes work loads on a set of computing resources. It aims to optimize resource utilization, maximize throughput, minimize response time, and avoid overload of any single resource

Here, to demonstrate how Load Balancer works, we focus on network load balancing. It distributes client's HTTP request across a set of servers in the load balancer pool. The response from one of the servers includes the server's hostname.

OpenStack provides Load Balance as a Service (LBaas), which is an advanced service of neutron. Lbaas allows for proprietary and open-source load balancing technologies to drive the actual load balancing of requests.

Here, Haproxy is used in OpenStack to implementing load balancing.

In OpenStack, the components included in Load Balancer are as follows:

- Load Balancer pool
- Members of the pool
- The Virtual IP (VIP) of the pool
- The Floating IP associated with VIP
- Monitor

Requirements for the Members of the pool

The members of the Load Balancer pool must be servers that supports HTTP or HTTPS protocol to respond with its hostname.

To support HTTP protocol

There are two ways to set up a HTTP server which can respond with its hostname.

1. Using Shell command.

Before running the Shell command, make sure that Apache service has been stopped because Apache also listens to port 80.

To stop it, type "service Apache2 stop"

Run the following command:

While true; do echo –e "HTTP/1.1 200 OK \r\n\r\n Welcome to `hostname`" | sudo nc –l –p 80; done

2. Using Apache

The VM should boot from Linux with Apache and PHP installed.

1) Start Apache service
   Before starting Apache, make sure no Shell scripts are running to listen port 80.
2) /etc/apache2/sites-enabled/000-default.conf:
   To configure the default webpage and the default directory for that.

The default webpage for Apache is "index.html", which resides "/var/www/html/". It's not necessary to change this.

3) Provide the files which is needed to show the hostname
   All the files are in "/var/www/html/loadbalancer/", they are as follows:
   - http.html, http.js,
     Note: In http.js, the floating IP should match with the one of the pool
   - hostname.php

To support HTTPS protocol

Apache can be used to set up a server supporting HTTPS protocol.
The environment requirements for such a server is Linux with Apache and PHP. It can be implemented in the following steps:

1) /etc/apache2/sites-enabled/000-default-ssl.conf
   To make the server support HTTPS, /etc/apache2/sites-available/default-ssl.conf is copied to /etc/apache2/sites-enabled/

   Cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-enabled/000-default-ssl.conf

   It's the configuration file for ssl. Here, no change is made.
2) Provide the files which is needed to show the hostname
   This is the same as in setting up a HTTP server.
   All the files are in "/var/www/html/loadbalancer/", they are as follows:
   - https.html, https.js
     Note: In https.js, the floating IP should match with the one of the pool.
   - hostname.php
3) Enable SSL module
   In CLI window, type in "a2enmod ssl" to enable SSL module.
   "a2dismod ssl" can be used to disable SSL.
   After the execution of this command, restarting Apache service is required, so type in "service Apache2 restarted"

A Snapshot in which HTTP/HTTPS is available

A snapshot has been created which has implemented both services, which is "Loadbalancer".
Two credentials can be used to log in:

a) Username: root, password: root
b) Username: chuck, password: chuck

There are some Shell script files in the home folder to test Load Balancer under the account of "chuck".

Implementation of Load Balancer

The servers that will be balanced by Load Balancer have been created. They can be called HOST1, HOST2, HOST3…. Next, the Load Balancer can be implemented by Horizon.

1. Create a pool
2. Assign a Virtual IP (VIP) for the pool
3. Associate the VIP with a Floating IP
4. Add the servers into the pool
5. Add a monitor
   In Juno, only CLI can complete the association.
   Neutron lb-healthmonitor-associate is the command
6. Associate the monitor with the pool

After all the above steps, a Load Balancer has been implemented. You can check the status of all the servers in the pool, which could be "active" or "inactive".

Note: Servers can support HTTP/HTTPS concurrently, while a pool and a monitor which have been associated with each other can only work under one protocol. Therefore, to test both HTTP/HTTPS protocols, two pools and two monitors should be created to support both protocols respectively.

Test of Load Balancer

Let "10.0.0.xxx" be the floating IP of the pool which works under HTTP protocol.
Let "192.168.xxx.xxx" be the VIP of the pool which works under HTTP protocol.

Let "10.0.0.yyy" be the floating IP of the pool which works under HTTPS protocol.
Let "192.168.yyy.yyy" be the VIP of the pool which works under HTTPS protocol.
Before the test, make sure the Floating IP is the same as the one in http.js and https.js

For servers which are implemented with Apache, two methods can be used to test Load Balancer:

1. CLI for testing HTTP implemented with Shell command
   Use curl to send HTTP GET request:
   Curl –s http://10.0.0.xxx

   Or request in a recursive way:
   While true; do curl -s http://10.0.0.xxx; done

2. CLI for testing HTTP implemented with Apache
   Use curl to send HTTP GET request:
   Curl –s http://10.0.0.xxx/loadbalancer/hostname.php

   Or request in a recursive way:
   While true; do curl -s http://10.0.0.xxx/loadbalancer/hostname.php; done

3. Web browser for testing HTTP
   - From login server, start Chrome or Firefox and type in http://10.0.0.xxx/loadbalancer/http.html to test HTTP protocol.
   - In the webpage opened, press "Start Test".
     Then you can see the names of the responding server are displayed on the screen.

     Note: for browser, after one session is created, always one server will respond to it. So you can't see hostnames of different servers alternates on the screen. But if you open another tab and test the Load Balancer from the start, different hostnames can be seen.

4. CLI for testing HTTPS
   - Use curl to send HTTP GET request:

   Curl –s -k https://10.0.0.yyy/loadbalancer/hostname.php

   - Or request in a recursive way:

   While true; do curl –s -k https://10.0.0.yyy/loadbalancer/hostname.php; done

5. Web browser for testing HTTPS
   - From login server, start Chrome or Firefox and type in http://10.0.0.yyy/loadbalancer/http.html to test HTTP protocol.
   - In the webpage opened, press "Start Test".
     Then you can see the names of the responding server are displayed on the screen.

     Note: for browser, after one session is created, always one server will respond to it. So you can't see hostnames of different servers alternates on the screen. But if you open another tab and test the Load Balancer from the start, different hostnames can be seen.