Load Balancing:
Installation of HA Cluster with Load Balancing on Centos 7 Servers

Kevin Nguyen
knguy523

1 INTRODUCTION

Load balancing is the process of “efficiently distributing incoming network traffic across a group of backend servers.”[1] It plays an important role in maintaining the availability of a service for customers, partners, and end-users. In this report we will discuss a possible solution to this problem.

Multiple servers may be installed to keep up with client requests. However, without a way to distribute requests, a single server may still be overworked. Introducing a load balancer (LB) will help alleviate this issue by managing client requests. Requests are sent to the LB server which will be strategically sent to the backend servers on behalf of the client. However, if this server is offline, users will not be able to access the resource requested. To solve this problem additional LB servers are installed and configured to act as backups. This covers the high availability (HA) of the resource.

The solution involves HAproxy, keepalived, and Ansible. Two frontend servers act as load balancers and three web servers act as the backend. Each server is a VM running Centos 7 on a NAT network. Ansible is used to automate the configuration of the servers. Assessability of the web page on Firefox browser and curl demonstrate the functionality of the setup.

[1] What Is Load Balancing?)

2 RELATED WORKS

A simple implementation of load balancing can be achieved with DNS load balancing. Reliant on the fact that DNS sends a round-robin list of IP addresses, this implementation may cause problems down the line.[2] The round-robin method does not solve the problem of overloading a server- the server may still be overloaded if connection time for each client is high. Additionally, the DNS does not check if the backend is alive, meaning a client may connect to a downed resource.

Other similar works depend on the specifics of the hardware and software. In the past, load balancing was done on hardware but today, there is a shift to software load balancers. Software load balancers include Nginx, HAproxy, and Avi Vantage Software Load Balancer.

The differences between balancers depend on the use cases -some geared towards cloud or applications- and the deployment features. For this project, I chose the web servers as the backend. HAproxy was a good choice because it deals with TCP/HTTP connections and is free.

Similarly, for HA clusters there are many software and hardware configurations. In researching this topic I have come across HA-Linux, Pacemaker, and keepalived.

[2] (What is DNS Load Balancing?)

Many tools exist for automation. It may be as simple as a bash script or more complex and robust with automation tools. For the scope of this project, I found Ansible adequate.

An example setup for this project can be found at upcloud.com.[3] This tutorial goes over the basic setup of HAproxy. The configuration for layer 4 was used as a reference for my project. Apart from that, my project incorporates a HA cluster and automation.

Another example project uses HAproxy but a different HA clustering solution.[4] Concepts are similar apart from the program specific configurations. Again here there is no automation.

3 TECHNICAL DETAILS

The main idea of the project is to have a server that handles requests from clients. Traffic will pass through that server and forward the request to some resource on another server. After looking at descriptions of what LBs are, it was clear that there needs to be a frontend server for client interaction and backends servers.

The frontend server will have the load balancer installed and be configured in a HA cluster. The backend contains a web page the client wants to access.

The servers must be able to communicate with each other, so each machine is placed on the same NAT network.

[3] (Ruostemaa)
[4] (Timme)

Some of the more notable settings of HA proxy are the mode, the port the frontend is bound to, and the list of web servers with IP addresses. The httpchk option ensures HAproxy inquires the HTTP status of the web servers and prevents forwarding to a downed server.

However, the original problem still exists to some degree. If the LB server goes offline, clients will never be able to access their resource they requested. The solution for this is described by some key terms.

While learning about LBs, the keywords “fault tolerance” and “highly available” are often thrown around. It describes a failover system, referring to the “ability to switch automatically and seamlessly to a reliable backup system.”[5] In other words, redundant servers that communicate with each other serve as backups. The LB servers need to be clustered together to achieve fault tolerance and high availability.

More research was required to understand this problem and explore software solutions. Pacemaker is a possible solution; however, I found that it was too complex for the project. Instead, I opted for keepalive for its simplicity. Multiple servers can be assigned as backups but having one backup is sufficient to show the failover.

Notable settings for keepalived are the state of the machine (master or backup), the shared VIP, and the check script that is used to verify that HAproxy is up and running.

Installing and setting up keepalived solves the problem of fault tolerance, but I was still not satisfied. From this course, I learned

[5] (What is failover)

that automation of repetitive tasks is a necessary skill for a system administrator.

Because the problem involves multiple servers, a shell script to automate this process would likely be overly complex. Instead, I searched for more robust automation tools and happened across Ansible.

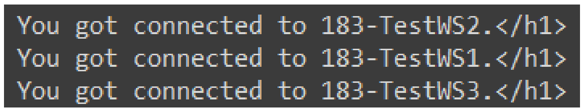
Ansible is an agent-less automation tool that can be installed and used to configure machines through ssh. For this project, I installed Ansible on one server to set itself and the other servers up.

Ansible playbooks exist for configuration management. Tasks include installation of software, the configuration of services, or modification of system settings. I created a playbook and an inventory file containing the IP addresses of the servers on the network. The LBs and web servers must be separated into groups to ensure proper setup when the playbook is run.

4 RESULTS

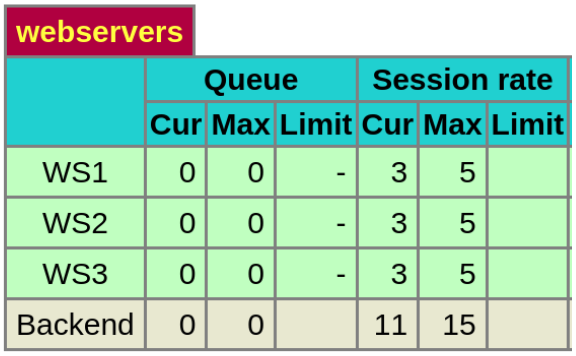
HAproxy

To test the setup, 2 scripts that run 3 curl commands every 0.5 seconds to mimic multiple client requests.



```
You got connected to 183-TestWS2.</h1>
You got connected to 183-TestWS1.</h1>
You got connected to 183-TestWS3.</h1>
```

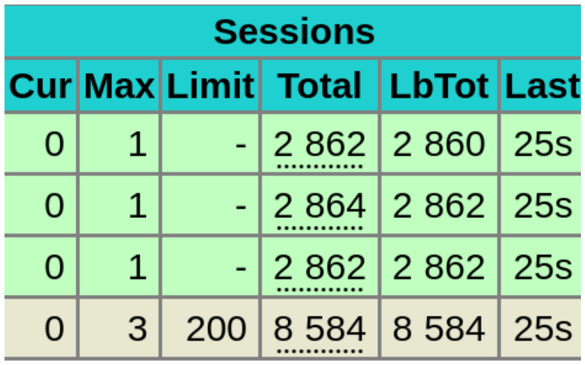
Figure 1: Truncated output of bash script



webservers	Queue			Session rate		
	Cur	Max	Limit	Cur	Max	Limit
WS1	0	0	-	3	5	
WS2	0	0	-	3	5	
WS3	0	0	-	3	5	
Backend	0	0		11	15	

Figure 2: webservers session rate stats

Figure 2 is the HAproxy stat page when HAproy is configured to balance based on least connections.



Sessions					
Cur	Max	Limit	Total	LbTot	Last
0	1	-	2 862	2 860	25s
0	1	-	2 864	2 862	25s
0	1	-	2 862	2 862	25s
0	3	200	8 584	8 584	25s

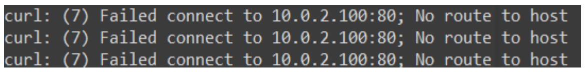
Figure 3: webservers session state

From figure 2 the session rates remain consistent between each server, indicating that the load is indeed balanced between the web servers based on least connections. Although we cannot conclude that the balancing is based on least connections from figure 3, the close number of total sessions indicates load balancing.

Keepalive

During the test, the master keepalive server is brought down by stopping the keepalived service. Almost no disruption was experienced as the backup server quickly began handling client requests.

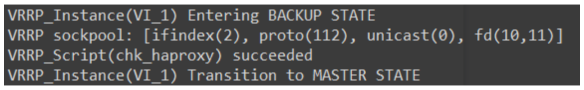
Taking down both servers causes the websites to become accessible as expected.



```
curl: (7) Failed connect to 10.0.2.100:80; No route to host
curl: (7) Failed connect to 10.0.2.100:80; No route to host
curl: (7) Failed connect to 10.0.2.100:80; No route to host
```

Figure 4: Both servers down

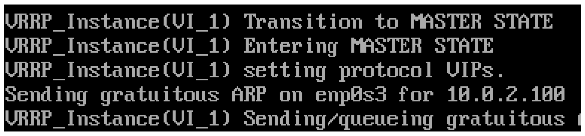
Bringing the backup server back up succuflly promotes it to master, shown in figure 5 below.



```
VRRP_Instance(VI_1) Entering BACKUP STATE
VRRP sockpool: [ifindex(2), proto(112), unicast(0), fd(10,11)]
VRRP_Script(chk_haproxy) succeeded
VRRP_Instance(VI_1) Transition to MASTER STATE
```

Figure 5: Bringing backup LB up

Bringing the original master server back online by starting the service successfully promoted it back to master.



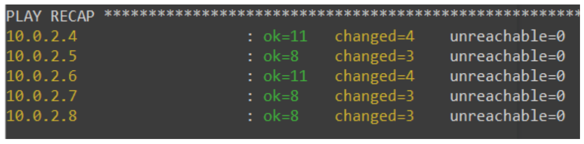
```
VRRP_Instance(VI_1) Transition to MASTER STATE
VRRP_Instance(VI_1) Entering MASTER STATE
VRRP_Instance(VI_1) setting protocol VIPs.
Sending gratuitous ARP on enp0s3 for 10.0.2.100
VRRP_Instance(VI_1) Sending/queueing gratuitous
```

Figure 6: Restarting keepalived on Master

Lastly, taking down the backup server while the master server is still up causes no description.

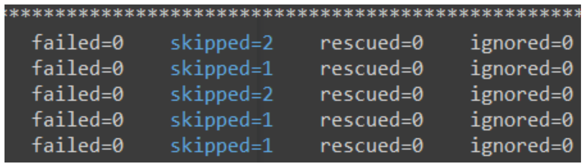
Gratuitous ARP messages shown in figure 6 demonstrate the functionality of VIP. Whichever LB is master owns the VIP and will have traffic routed through it.

Ansible



```
PLAY RECAP *********************************************************
10.0.2.4                   : ok=11   changed=4    unreachable=0
10.0.2.5                   : ok=8    changed=3    unreachable=0
10.0.2.6                   : ok=11   changed=4    unreachable=0
10.0.2.7                   : ok=8    changed=3    unreachable=0
10.0.2.8                   : ok=8    changed=3    unreachable=0
```

Figure 7A: Ansible Playbook Summary



```
****************************************************
  failed=0    skipped=2    rescued=0    ignored=0
  failed=0    skipped=1    rescued=0    ignored=0
  failed=0    skipped=2    rescued=0    ignored=0
  failed=0    skipped=1    rescued=0    ignored=0
  failed=0    skipped=1    rescued=0    ignored=0
```

Figure 7B: Ansible Playbook Summary

Figures 7A and 7B summarize the outcome of the playbook. The playbook checks SELinux configuration, setting it to disabled and restarting if needed. After, the LB servers are configured, installing software and placing the configuration files in the correct play. The firewall is configured to allow connections for keepalive and its service is enabled. Similar steps are followed for the installation of HAproxy.

Web servers are set up Similarly using ansible. Packages are installed and a simple HTML page is created with the hostname of the server. The service is enabled and the firewall configured.

Necessary configuration files need to be in the same directory as the location the playbook is run. Install Ansible on a machine with a network connection to all the machines in the inventory. It can be installed with the following:

```
sudo yum -y install epel-release
&& sudo yum -y install ansible
```

The server inventory file needs to be adjusted to reflect the servers on the network. IP addresses of the load balancer and web servers need to be filled in. Note the first IP address for the load balancers will be set to the master. The targets group is used for pinging the machines with the command:

```
ansible targets -i server-inv.txt
-m ping --ask-pass
```

If the machines are pingable, the following runs the ansible playbook to set up the loadbancers and web servers:

```
ansible-playbook -b -i
server-inv.txt lb-playbook.yml
--ask-pass
```

5 CONCLUSION

In this project, we have shown the basics of setting up a HA load balancer for a backend of web servers in Centos 7. The task can be automated using automation tools if there is a network connection between the servers.

To make this project a step further, the installation of the Centos machines can be automated. A provisioning system with cloud computing management, such as MAAS + Juju/Openstack, would make it increasingly easy to spin up and configure servers. From here, systems like the HA LB setup I explored, can then be easily initialized.

In terms of future work, the software solutions for a HA LB must be considered. Furthermore, configurations for both the HA cluster and LB need to be tailored to the needs and use cases of front and backend servers. This includes connections timeout, more specific HTTP health checks, ACL for more advanced routing, and other forms of authentication.

---

6 REPORT/PROJECT REFERENCES

Critelli, Anthony. “Keepalived and High Availability: Advanced Topics.” *Enable Sysadmin*, Red Hat, Inc., 1 Apr. 2020, https://www.redhat.com/sysadmin/advanced-keepalived.

Critelli, Anthony. “Setting up a Linux Cluster with Keepalived: Basic Configuration.” *Enable Sysadmin*, Red Hat, Inc., 25 Mar. 2020, https://www.redhat.com/sysadmin/keepalived-basics.

“An Introduction to HAProxy and Load Balancing Concepts.” *DigitalOcean*, DigitalOcean, 25 Nov. 2021, https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts.

Lavoie, Chad. “The Four Essential Sections of an HAProxy Configuration.” *HAProxy Technologies*, 15 June 2021, https://www.haproxy.com/blog/the-four-essential-sections-of-an-haproxy-configuration/.

Ruostemaa, Janne. “How to Install Haproxy Load Balancer on Centos.” *UpCloud*, 30 Mar. 2021, https://upcloud.com/community/tutorials/haproxy-load-balancer-centos/.

Timme, Falko. “Setting up a High-Availability Load Balancer (with Failover and Session Support) with HAProxy/Heartbeat on Debian Etch - Page 2 - Page 2.” HowtoForge, Howtoforge, https://www.howtoforge.com/high-availability-load-balancer-haproxy-heartbeat-debian-etch-p2.

“What Is a Failover? Definition and Related Faqs.” *Druva*, 18 June 2021, https://www.druva.com/glossary/what-is-a-failover-definition-and-related-faqs/.

“User Guide.” *User Guide - Ansible Documentation*, 12 Nov. 2021, https://docs.ansible.com/ansible/latest/user_guide/index.html.

“What Is DNS Load Balancing?” *NGINX*, 29 May 2020, https://www.nginx.com/resources/glossary/dns-load-balancing/.

“What Is Load Balancing? How Load Balancers Work.” *NGINX*, 26 Jan. 2021, https://www.nginx.com/resources/glossary/load-balancing/.