

LOAD BALANCER

Today we shall configure load balancer and Haproxy using ansible. Before moving on to the steps let us understand both the terms:

The program running to serve on behalf of another is SERVER PROXY. Here we are using HA PROXY which is behaving as a frontend. It has a feature Load balancer which balances the load and distributes the load to multiple same webserver hosted. The process by which is distributes the load is known as ROUND ROBIN. Reverse proxy is the process when the hosted server gives the response back to frontend (haproxy) , it prevents the direct interaction between client and the server using haproxy

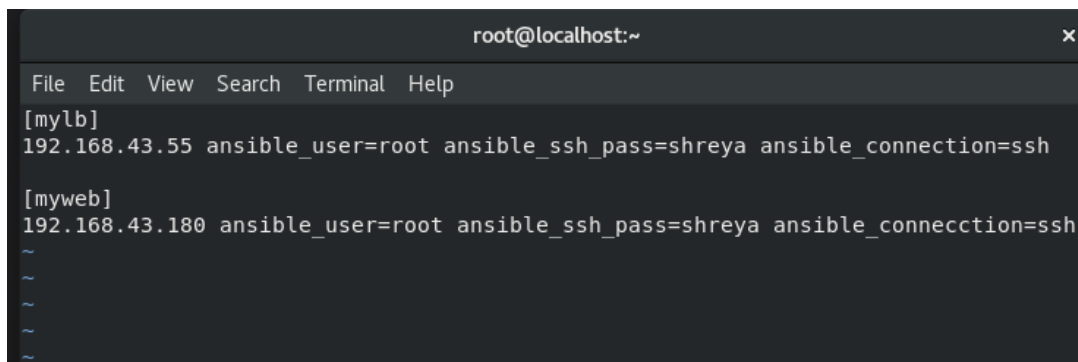
Advantages to using Load Balancer are that:

1. As we don't let the client to directly interact with the server, this concept is highly secure
2. If any of the back-end server fails, client won't have to go to the bad experience of application crash. This is because, we have just provided a front-end ip, and we have a lot of back-end servers running in the background. If any one of the backend server fails, user will get automatically to another server without any lagging.
This is very good for the reputation of the company.

Now, let us move to configure the load balancer:

In the root

Vim ip.txt



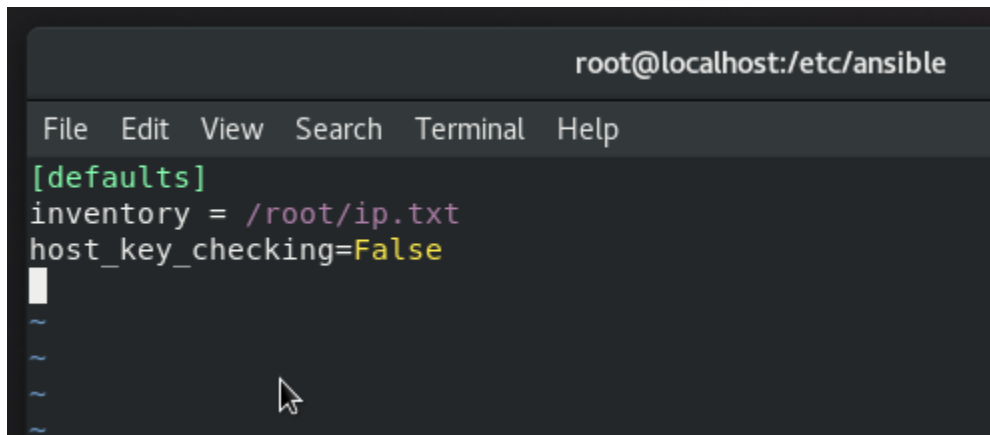
```
root@localhost:~  
File Edit View Search Terminal Help  
[mylb]  
192.168.43.55 ansible_user=root ansible_ssh_pass=shreya ansible_connection=ssh  
  
[myweb]  
192.168.43.180 ansible_user=root ansible_ssh_pass=shreya ansible_conneccction=ssh  
~  
~  
~  
~
```

After assigning IPs to the LB and the webserver, we now edit the configuration file.

```
cd /etc/ansible
```

```
vim ansible.cfg
```

Under defaults, give the location of the inventory. Assign `host_key_checking=False`. It will enable the remote connection.



```
root@localhost:/etc/ansible
File Edit View Search Terminal Help
[defaults]
inventory = /root/ip.txt
host_key_checking=False
~
~
~
~
```

Next step is to check the connectivity. For that we use the command **ansible all -m ping**.



```
[root@localhost ansible]# ansible all -m ping
192.168.43.180 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
192.168.43.55 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
```

Both the target nodes are newly installed, so we don't have yum configured. I have a playbook named yum.yml. I shall run it using **ansible-playbook yum.yml** and hence, both the target node will get yum configured.

vim yum.yml

```
File Edit View Search Terminal Help
- hosts: all
  tasks:
    - file:
        state: directory
        path: "/dvd1"
    - mount:
        src: "/dev/cdrom"
        path: "/dvd1"
        state: mounted
        fstype: "iso9660"
    - yum_repository:
        baseurl: "/dvd1/AppStream"
        name: "mydvd1"
        description: "my yum is configured for repo AppStream"
        gpgcheck: no
    - yum_repository:
        baseurl: "/dvd1/BaseOS"
        name: "mydvd2"
        description: "my yum is configured for repo BaseOS"
        gpgcheck: no
```

```
File Edit View Search Terminal Help
PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.43.180]
ok: [192.168.43.55]

TASK [file] *****
changed: [192.168.43.55]
changed: [192.168.43.180]

TASK [mount] *****
changed: [192.168.43.55]
changed: [192.168.43.180]

TASK [yum_repository] *****
changed: [192.168.43.55]
changed: [192.168.43.180]

TASK [yum_repository] *****
changed: [192.168.43.55]
changed: [192.168.43.180]

PLAY RECAP *****
192.168.43.180 : ok=5  changed=4  unreachable=0  failed=0  s
Enterprise Linux
```

Yum has been successfully configured.

After this, we write a new playbook for installing haproxy. One of the IP will act as the webserver and the other will act as the load balancer.

```
root@localhost:/haproxy
File Edit View Search Terminal Help

- hosts: myweb
  tasks:
    - name: "Install httpd apache server"
      package:
        name: "httpd"
    - copy:
        dest: "/var/www/html/web.html"
        content: "Testing Load Balancer"
    - service:
        name: "httpd"
        state: restarted
- hosts: mylb
  tasks:
    - name: "install LB software"
      package:
        name: "haproxy"
```

When we run it, haproxy will get installed in the load balancer target node, and https is successfully activated in the webserver IP.

```
[root@localhost haproxy]# ansible-playbook load.yml

PLAY [myweb] *****
TASK [Gathering Facts] *****
ok: [192.168.43.180]
TASK [Install httpd apache server] *****
changed: [192.168.43.180]
TASK [copy] *****
changed: [192.168.43.180]
TASK [service] *****
changed: [192.168.43.180]
PLAY [mylb] *****
TASK [Gathering Facts] *****
ok: [192.168.43.55]
TASK [install LB software] *****
changed: [192.168.43.55]
```

Now, we go to the target 1, that is our load balancer, and copy the configuration file of haproxy to the control node.

First, check if haproxy has been installed using **rpm -q haproxy**

Command: **scp /etc/haproxy/haproxy.cfg 192.168.43.105:/haproxy**

This command will copy the haproxy.cfg file from the target node to the control node, whose IP is 192.168.43.105. It will get saved in the directory named haproxy.

```
Valid_HTT forever preferred_HTT forever
[root@localhost ~]# rpm -q haproxy
haproxy-1.8.15-5.el8.x86_64
[root@localhost ~]# scp /etc/haproxy/haproxy.cfg 192.168.43.105:/haproxy
The authenticity of host '192.168.43.105 (192.168.43.105)' can't be established.
ECDSA key fingerprint is SHA256:j7qGB/yHNZ9721iTDfu6SeK07I+3QHvBnUgqUGUF4Zk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.105' (ECDSA) to the list of known hosts.
root@192.168.43.105's password:
haproxy.cfg                                100% 3284      2.8MB/s   00:00
[root@localhost ~]# _
```

Come back to the control node and check if the file has reached there.

```
[root@localhost haproxy]# ls
haproxy.cfg  load.yml  yum.yml
[root@localhost haproxy]#
```

Edit to add the IP of the backend.

vim haproxy.cfg

Do the following changes.

```
#-----
# round robin balancing between the various backends
#-----
backend app
    balance      roundrobin
    server  app1 192.168.43.180:80 check
~
~
~
```

Copy the configuration file from the control node to the target node and start the service of haproxy.

```
root@localhost:/haproxy
File Edit View Search Terminal Help
package:
  name: "httpd"
- copy:
  dest: "/var/www/html/web.html"
  content: "Testing Load Balancer"
- service:
  name: "httpd"
  state: restarted
- hosts: mylb
  tasks:
  - name: "install LB software"
    package:
      name: "haproxy"
  - template:
    dest: "/etc/haproxy/haproxy.cfg"
    src: "haproxy.cfg"
  - service:
    name: "haproxy"
    state: restarted
```

Run the playbook.

```
root@localhost:/haproxy
File Edit View Search Terminal Help
TASK [service] *****
changed: [192.168.43.180]

PLAY [mylb] *****

TASK [Gathering Facts] *****
ok: [192.168.43.55]

TASK [install LB software] *****
ok: [192.168.43.55]

TASK [template] *****
ok: [192.168.43.55]

TASK [service] *****
changed: [192.168.43.55]

PLAY RECAP *****
192.168.43.180      : ok=4    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.43.55      : ok=4    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

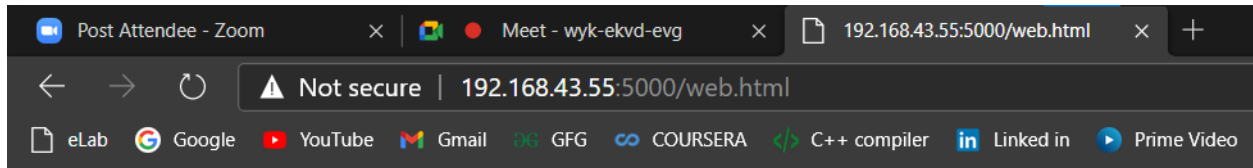
[root@localhost haproxy]#
```

Enterprise Linux

As we know that, webserver was configured on IP 192.168.43.180, but we don't want to expose that IP to the public. Hence we have used the concept of load balancer.

In the browser, we write the IP of the load balancer's target node. Along with that we give the port number of the Haproxy.

The content that we have written on the webpage will appear successfully!



Testing Load Balancer