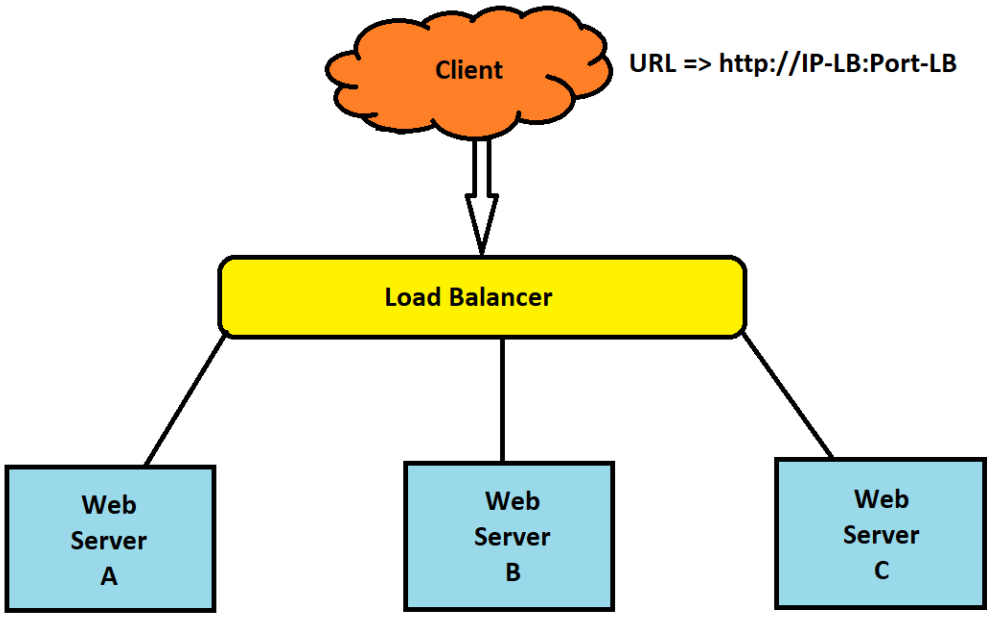
Ansible

Session 5 - 24th Dec 2022 Summary

- When clients go to the web server then there is a limit of clients for the web server means for example only 100 clients can connect to the web server concurrently.

- For solving the above problem we can launch multiple web servers and provide the ip's to the client.

- But providing multiple ip's to the client for one service is not a good user experience. Therefore we can launch one system between client and web server and we can provide ip of the middle system to the client.Client can use that ip for connecting to the web server. And that middle system is known as **Load Balancer**.



- Client will connect to the web server through load balancer one by one means when client enters the ip of load balancer then first they connect to

Web Server A then next time Web Server B then C then again go to A and this type of algorithm is known as **Round Robin Algorithm**.

➢ The load balancer distributes incoming application traffic across multiple targets. Load Balancer is also known as **frontend or reverse proxy**.

➢ Web server is also known as backend.

➢ Adding or removing web servers this concept is known as **horizontal scaling**. Adding a web server is known as **scale-out** and removing a web server is known as **scale-in**.

➢ There are a lot of products for load balancer and one of product is **haproxy**.

➢ **Manual setup for load balancer and their target ie. web server.**

1) **Web Server :** With a web server we need a php package for running php code.

```
Setup for the web server:-
--------------------------
1) # yum install httpd php -y

2) # vim /var/www/html/index.php
   Add this code ->
   <pre>
   <?php
          print_r($_SERVER);
   ?>
   </pre>

3) Start web service : # systemctl start httpd
```

2) **Load Balancer :** Setup load balancer with 8080 port number and round robin algorithm.

```
Setup for the load balancer:-
-----------------------------
1) # yum install haproxy -y

2) # vim /etc/haproxy/haproxy.cfg
   Add this content ->
   frontend lb
        bind *:8080
        timeout client 10s
        default_backend myweb

   backend myweb
        balance roundrobin
        timeout connect 10s
        timeout server 10s
        server  app1 IP-Web-A:80
        server  app2 IP-Web-B:80

3) Start web service : # systemctl start haproxy
```

➢ Ansible uses **Jinja2 templating** to enable dynamic expressions and access to variables and use many more programming concepts. Eg. conditions, loops, etc.

1) **Variables** : If we want to use variable in template file then,
   Syntax : {{ variable_name }}

   When Playbooks are executed, these variables get replaced by actual values defined in Ansible Playbooks.

2) **Comments** : If we want to use comment in template file then,
   Syntax : {# this is comment #}

3) **Conditions** : If we want to use if condition in template file then,
   Syntax : {% if <condition> %}
   - - - - block - - - -

{% else %}
- - - -block- - - - -
{% endif %}

4) **Loop** : If we want to use loop in template file then,
Syntax : {% for i in name %}
{{ i }}
{% endfor %}

➢ When it comes to transferring files to a remote system with Ansible, the **copy and template** modules are used. Ansible's copy module does exactly what you expect. It copies a file from the local host to a remote server. The template module also copies a file to a remote server, but it allows you to use Jinja2 template.

➢ When we use Jinja2 in any file then that file is known as **Template file** & give **".j2"** extension for template file.

➢ **Setup for load balancer and their target ie. web server using Ansible :**

1) Create inventory-file in ansible manager node for a total 4 target nodes. In that inventory create 2 host groups such as lb and web.

 # vim /etc/ansible/hosts

```
[lb]
13.233.126.223 ansible_user=root ansible_password=saurabh

[web]
52.66.244.45 ansible_user=root ansible_password=saurabh
65.2.73.88 ansible_user=root ansible_password=saurabh
```

2) Check all the target nodes are pingable or not.

```
[root@ip-172-31-1-104 ~]# ansible lb -m ping
13.233.126.223 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[root@ip-172-31-1-104 ~]# ansible web -m ping
65.2.73.88 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
52.66.244.45 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

3) Create index.php code. This code simply prints the internal information of the OS such as ip, port, method, etc.

```
<pre>
<?php
        print_r($_SERVER);
?>
</pre>
~
```

4) Create a web.yml playbook for configuring a web server.

 # vim web.yml

```
- hosts: web
  tasks:
  - name: "Installing web server"
    package:
      name: "httpd"
      state: present

  - name: "Installing php package"
    package:
      name: "php"
      state: present

  - name: "Deploy web page"
    copy:
      src: "index.php"
      dest: "/var/www/html/"

  - name: "Start web service"
    service:
      name: "httpd"
      state: started
~
```

5) Register web server ip to load balancer with port no. 8080 of load balancer, round robin algorithm. For this, create a local configuration file with ".j2" extension.

 # vim local.cfg.j2

```
frontend lb
        bind *:8080
        timeout client 10s
        default_backend myweb

backend myweb
        balance roundrobin
        timeout connect 10s
        timeout server 10s
        server  app1 52.66.244.45:80
        server  app2 65.2.73.88:80
~
```

6) Create a playbook for configuring load balancer.

 # vim lb.yml

```
- hosts: lb
  tasks:
  - name: "Installing load balancer"
    package:
      name: "haproxy"
      state: present

  - name: "Registration web server to lb"
    template:
      src: "local.cfg.j2"
      dest: "/etc/haproxy/haproxy.cfg"

  - name: "Start the service"
    service:
      name: "haproxy"
      state: restarted
~
```

7) After doing this, run the playbooks.

 # ansible-playbook web.yml

 # ansible-playbook lb.yml

Check setup : *# curl http://IP-LB:8080*

➢ But if we want to add one more web server to load balancer then we have to open the local registration file(local.cfg.j2) and add the ip there. But this is a manual process. But if we want, as soon as we configure the web server our

local registration conf file(local.cfg.j2) automatically fetches the ip from inventory. For that we have to use some jinja with magic variables.

In the code below we use “groups” magic variable which return all groups in the inventory file.

```
frontend lb
        bind *:8080
        timeout client 10s
        default_backend myweb

backend myweb
        balance roundrobin
        timeout connect 10s
        timeout server 10s
{% for ip in groups['web']_%}
        server  app{{ loop.index }} {{ ip }}:80
{% endfor  %}
~
```