**NGINX Load balancer and HA proxy in AWS**

**NGINX:**

NGINX is a free, open-source web server that can also be used as a load balancer, reverse proxy, and more. It's known for its high performance and low resource utilization.

**Nginx load balancer:**

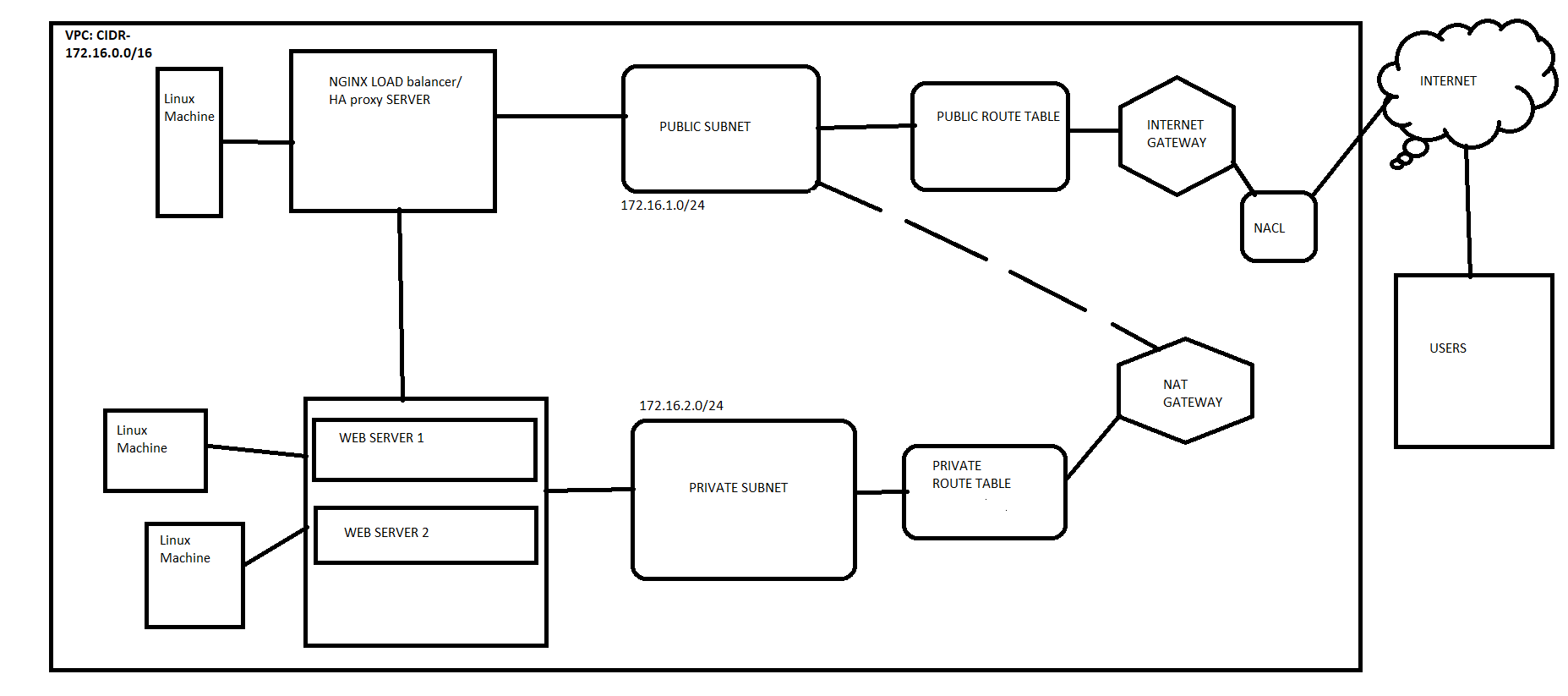
In load balancing setup, NGINX distributes incoming traffic among multiple backend servers. This is done to improve scalability, availability, and performance by spreading the requests evenly.

**HA(high availability) proxy**:

* + HAProxy is specifically designed as a high-performance load balancer, proxy server and it do health checks.
  + It is often used in environments where the primary focus is load balancing and high availability for HTTP/HTTPS, TCP, and SSL/TLS traffic.
  + HAProxy is often favored in enterprise environments or more complex architectures, as it offers very fine-grained control over load balancing algorithms and health checks.

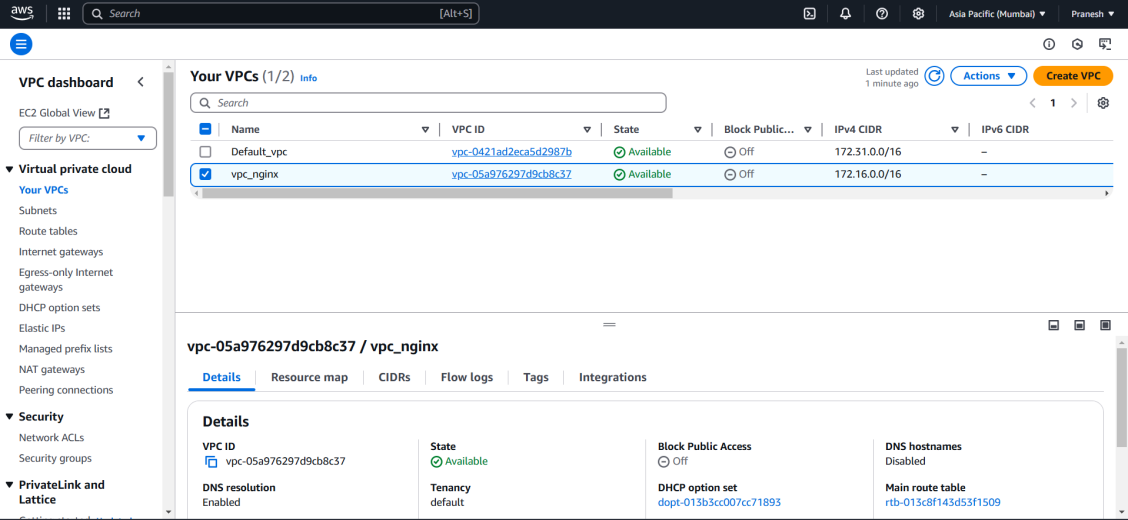
|  |  |  |
| --- | --- | --- |
| Feature | NGINX Load Balancing | HAProxy Load Balancing |
| Configuration Structure | Uses upstream block for backend servers and proxy\_pass. | Uses frontend and backend sections for traffic routing. |
| Load Balancing Algorithms | Round-robin (default), least connections, IP hash, etc. | Round-robin (default), leastconn, source (and many more). |
| Health Checks | No built-in health checks; requires external modules. | Built-in health checks for backend servers using check. |
| Session Persistence | Can use ip\_hash for session persistence. | Uses stick-table for session persistence, or source. |
| Use Case | Web server + reverse proxy + load balancer. | Specialized **high-performance** load balancer. |

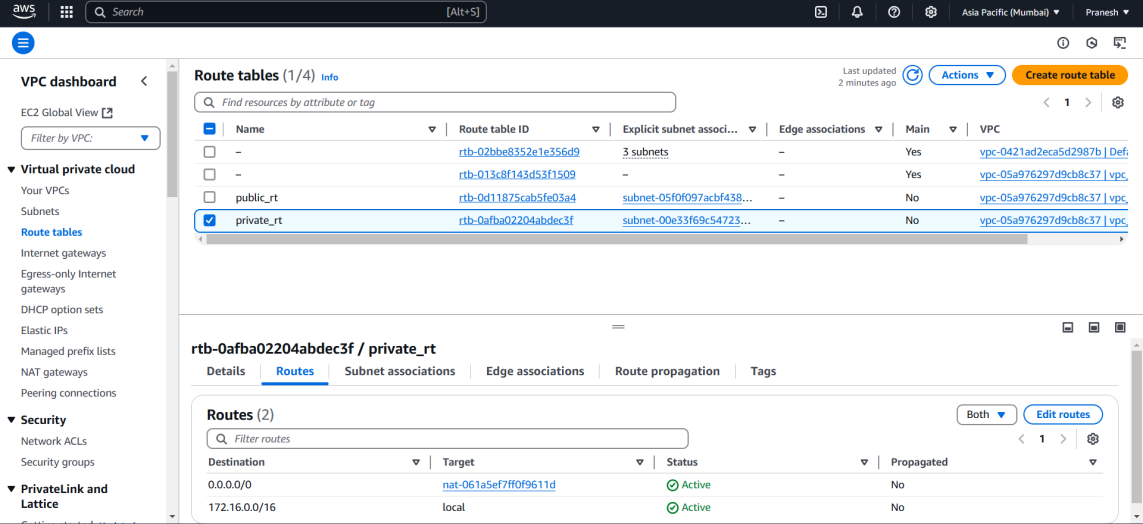
**Architecture:**



In AWS create a VPC :

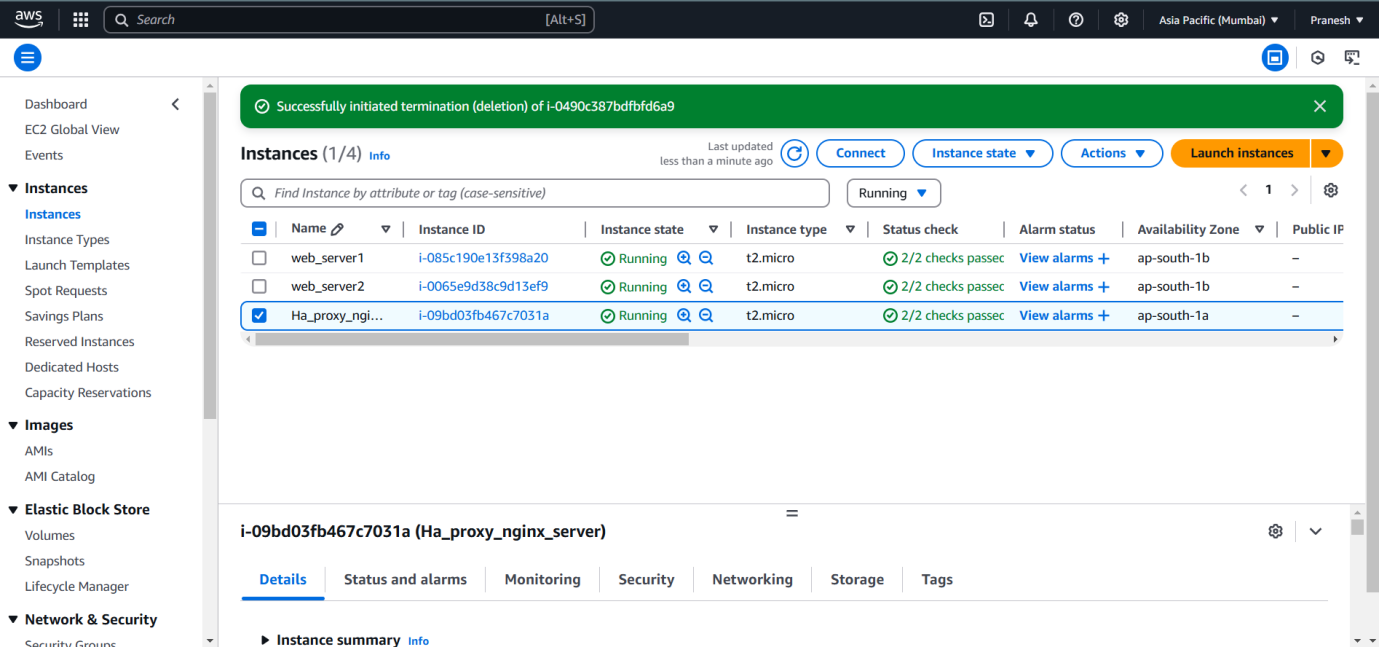
1. Creating VPC with CIDR block : 172.16.0.0/16 in one region.
2. Create two subnet public and private under the VPC.
3. Create Internet gateway for the resources to communicate public internet and attach to the vpc.
4. Create two route table public and private.
5. Associate route of IGW in public route table and associate public subnet in public RT.
6. Associate private subnet in private RT.
7. Create a security group(with inbound rules of SSH(22),HTTP(80)for public and for private(custom under public ip) under createdVPC.
8. Now for the private server to get secured access to the internet (for package installation NGINX)configure NAT gateway.
9. NAT gateway: it establish secure access to the outbound internet for the VM deployed under private subnet.
10. Create NAT gateway connect to public subnet and define the route and private subnet assosciation to the NAT.





NOW create Three instance (Linux Machine):

One (NGINX load balancer/HA proxy) is under public subnet and another Two(web server) is under Private subnet.



Now install NGINX in the machine and start the service:

Yum install nginx

Systemctl start nginx

Systemctl enable nginx in all the three machine

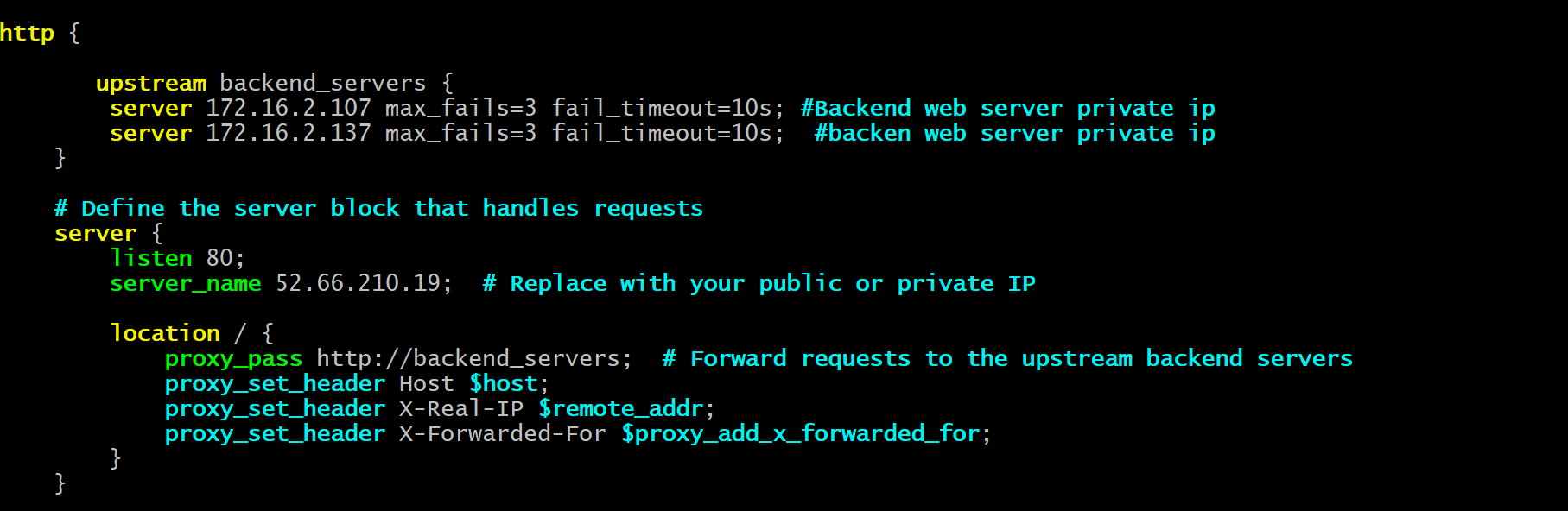
After starting the service go the Nginx conf file /etc/nginx/nginx.conf and edit the conf for the server to act as Load balancer and add the Backend server(web server) IP :

**For load balancer: (I have given with round robin) with failover concept.**

**Round Robin** (default): Automatically distributes requests evenly.

**Least Connections**: Sends requests to the server with the least active connections.

**IP Hash**: Routes requests from the same IP address to the same backend server



We have always insert the conf under the http block.

Test the conf file by running cmd sudo ngnix -t – it should pass

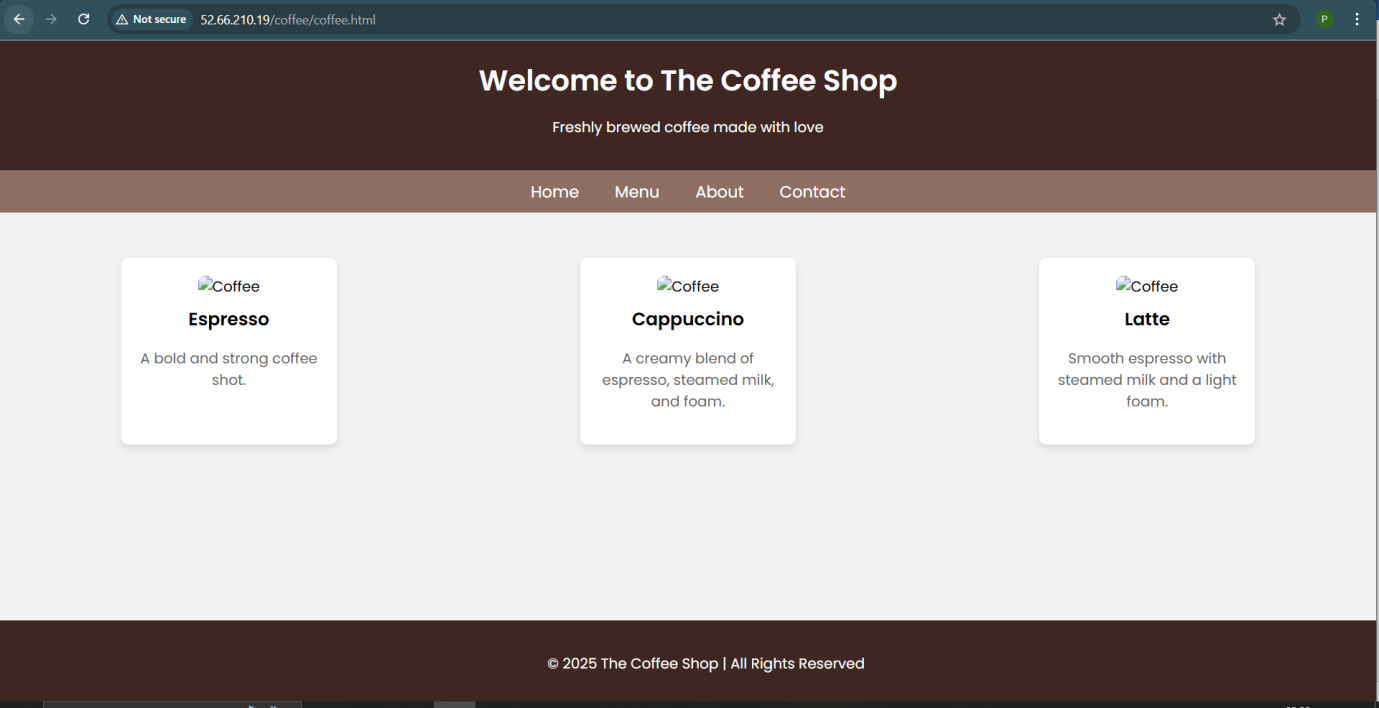
And reload the service—systemctl reload ngnix.

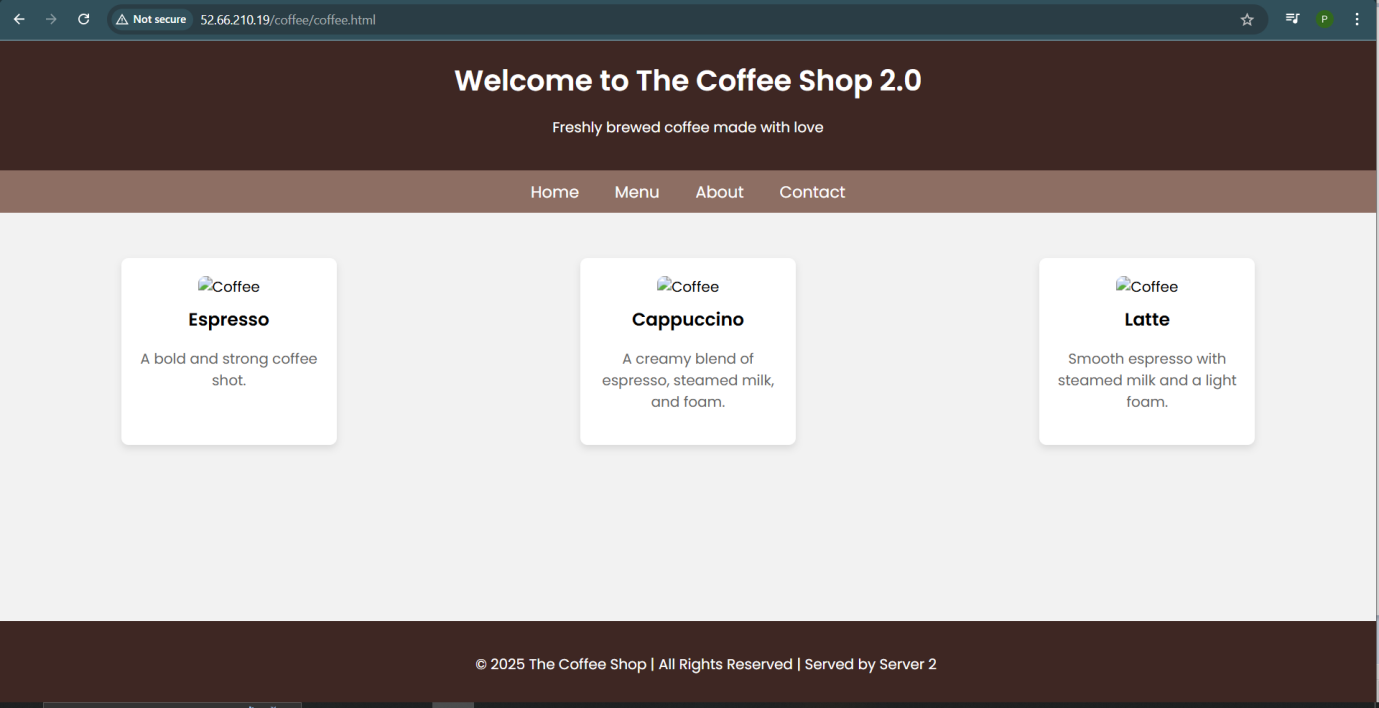
Now in the private subnet we have two machine for web server log in to the machine and install the Nginx package and put the code file in /usr/share/nginx/html.

And set the proper permissions.

And set access for SElinux by running cmnd chcon -R -t httpd\_sys\_content\_t /usr/share/ngnix/html/filename.

For the web page access from the Local machine try to access the web page by giving Proxy server public ip with path.





You can see for every hit it changes the load to both machine one after another.

If the One server down(stop one instance in AWS) we can get the access to the web page by the second server.

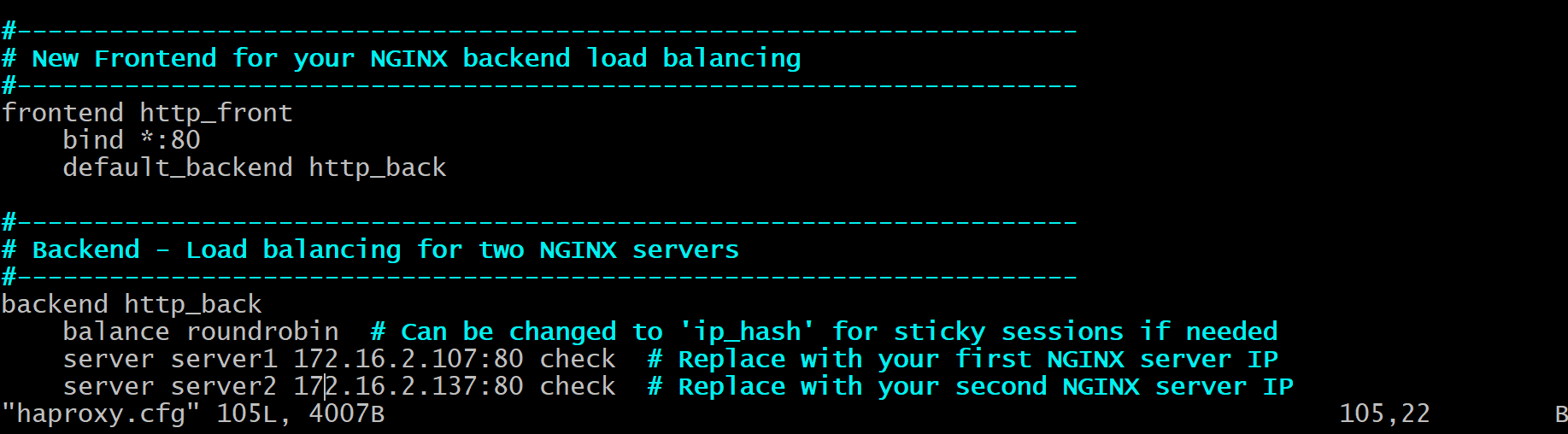
**For HA Proxy: (Consider the web page is already running in ngnix web server)**

Yum install haproxy

Systemctl start haproxy

Systemctl enable haproxy

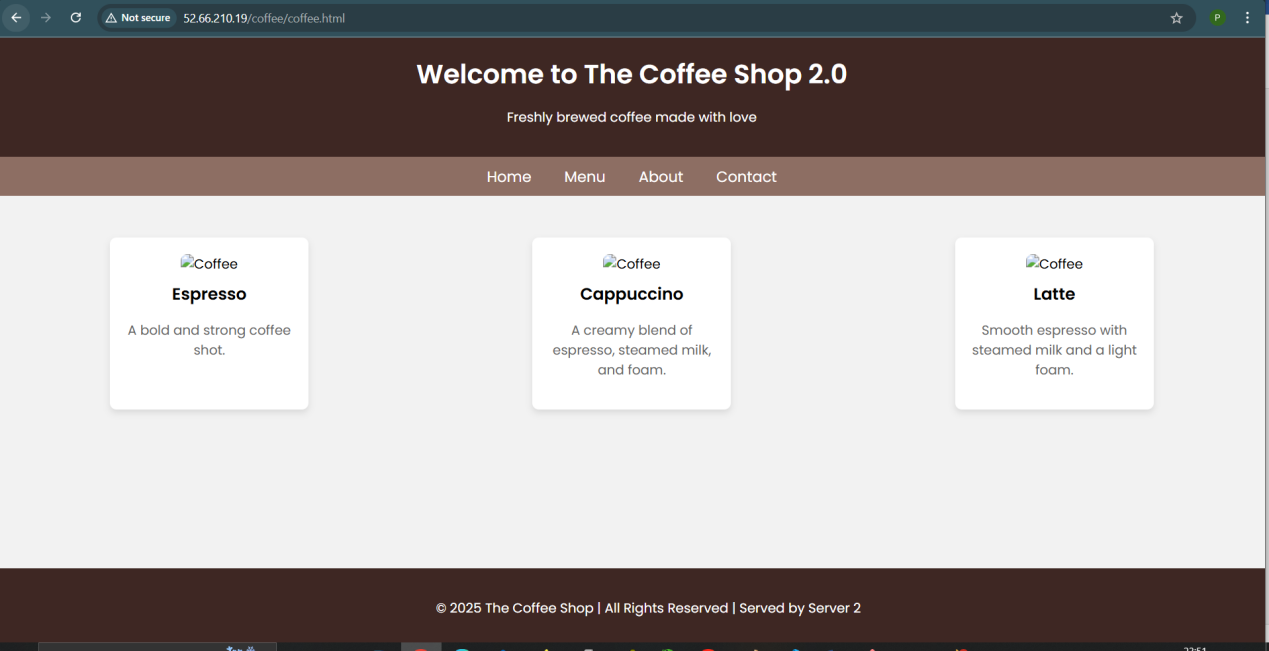
Go the Haproxy conf file /etc/haproxy/haproxy.cfg and edit the conf for the server to act as Haproxy and add the Backend server (web server) IP :



Test the conf file by running cmd sudo haproxy -f /etc/haproxy/haproxy.cfg-c

Once valid run systemctl reload haproxy.

Now check in the search engine by running public ip of haproxy server with file path.



You can see for every hit it changes the load to both machine one after another.