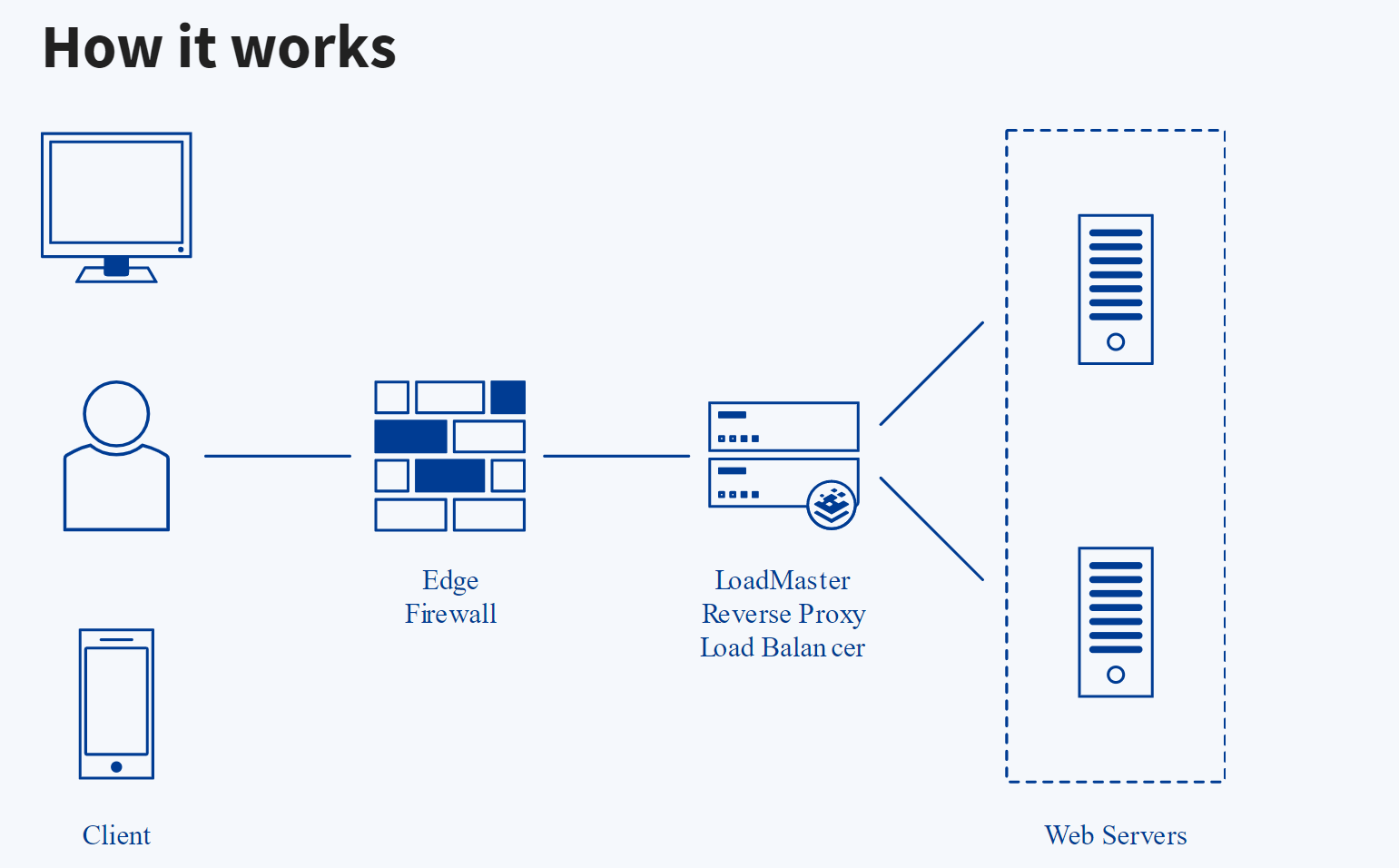
**Load Balancer Solution With Apache**

# 

The objective of this project is to deploy and configure an Apache Load Balancer for Tooling Website solution on a separate Ubuntu EC2 instance, ensuring that users can be served by Web servers through the Load Balancer.

A load balancer intelligently distributes traffic from clients across multiple servers without the clients having to understand how many servers are in use or how they are configured. Because the load balancer sits between the clients and the servers it can enhance the user experience by providing additional security, performance, resilience and simply scaling your website.

**How it works:**



When we access a website in the Internet we use an [URL](https://en.wikipedia.org/wiki/URL) and we do not really know how many servers are out there serving our requests, this complexity is hidden from a regular user, but in case of websites that are being visited by millions of users per day (like Google or Reddit) it is impossible to serve all the users from a single Web Server (it is also applicable to databases, but for now we will not focus on distributed DBs).

Each URL contains a [domain name](https://en.wikipedia.org/wiki/Domain_name) part, which is translated (resolved) to the IP address of a target server that will serve requests when opening a website on the Internet. Translation (resolution) of domain names is performed by [DNS servers](https://en.wikipedia.org/wiki/Domain_Name_System), the most commonly used one has a public IP address 8.8.8.8 and belongs to Google. You can try to query it with [nslookup](https://en.wikipedia.org/wiki/Nslookup) command:

nslookup 8.8.8.8; Server: UnKnown

Address: 103.86.99.99; Name: dns.google; Address: 8.8.8.8

When you have just one Web server and load increases - you want to serve more and more customers, you can add more CPU and RAM or completely replace the server with a more powerful one - this is called “vertical scaling”. This approach has limitations - at some point you reach the maximum capacity of CPU and RAM that can be installed into your server.

Another approach used to cater for increased traffic is “horizontal scaling” - distributing load across multiple Web servers. This approach is much more common and can be applied almost seamlessly and almost infinitely (you can imagine how many server Google has to serve billions of search requests).

Horizontal scaling allows us to adapt to current load by adding (scale out) or removing (scale in) Web servers. Adjustment of number of servers can be done manually or automatically (for example, based on some monitored metrics like CPU and Memory load).

Property of a system (in our case it is Web tier) to be able to handle growing load by adding resources, is called [“Scalability”](https://en.wikipedia.org/wiki/Scalability).

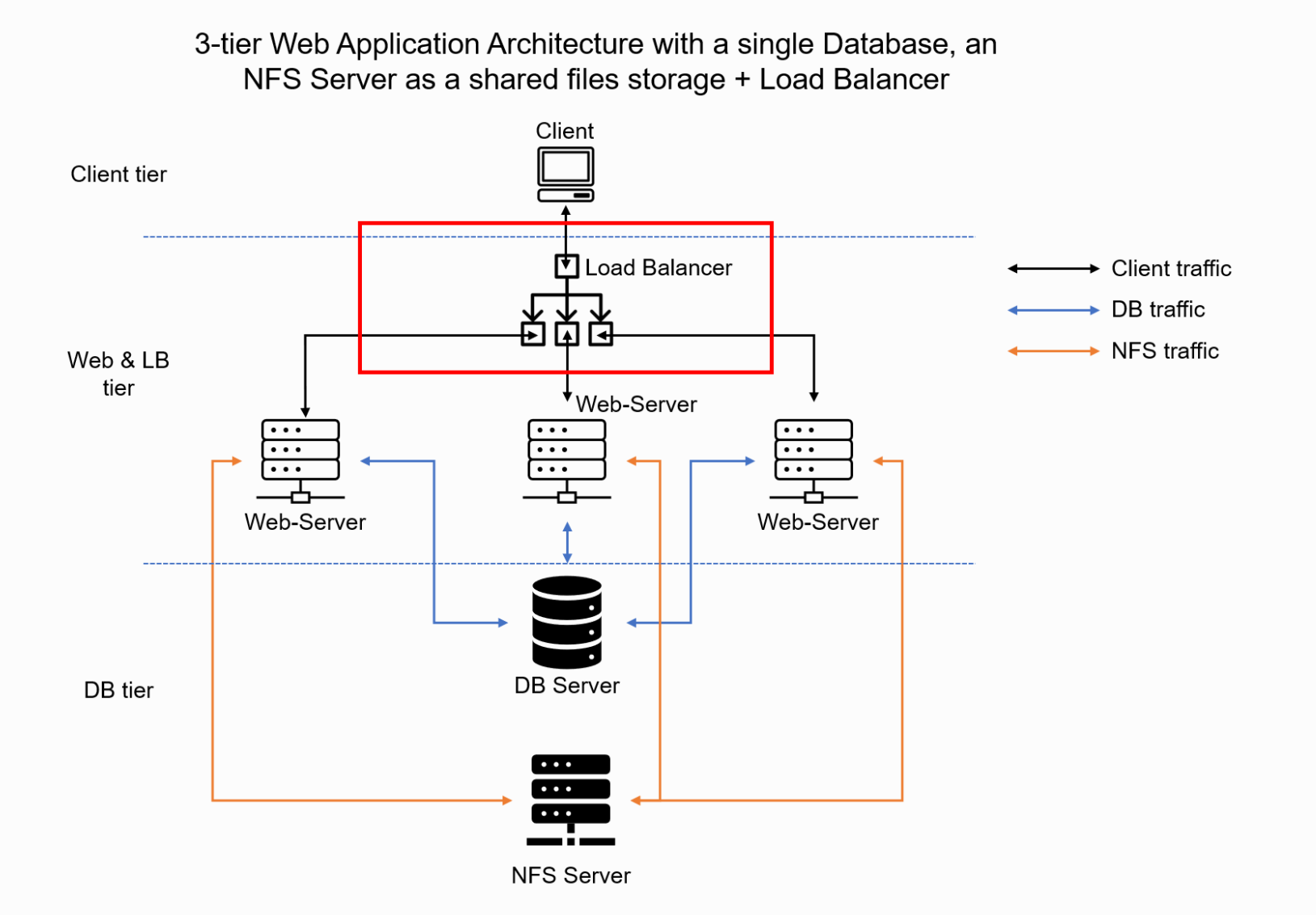
In order to hide all this complexity and to have a single point of access with a single public IP address/name, a [Load Balancer](https://en.wikipedia.org/wiki/Load_balancing_(computing)) can be used. A Load Balancer (LB) distributes clients’ requests among underlying Web Servers/ [Google servers](https://en.wikipedia.org/wiki/Google_data_centers).and makes sure that the load is distributed in an optimal way.

Let us take a look at the updated solution architecture with an LB added on top of Web Servers (for simplicity let us assume it is a software L7 Application LB, for example - [Apache](https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html), [NGINX](https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/) or [HAProxy](http://www.haproxy.org/))

In this project we will enhance our Tooling Website solution by adding a Load Balancer to distribute traffic between Web Servers and allow users to access our website using a single URL.

## Task

Deploy and configure an Apache Load Balancer for Tooling Website solution on a separate Ubuntu EC2 intance. Make sure that users can be served by Web servers through the Load Balancer.We will implement this solution with 2 Web Servers, the approach will be the same for 3 and more Web Servers.

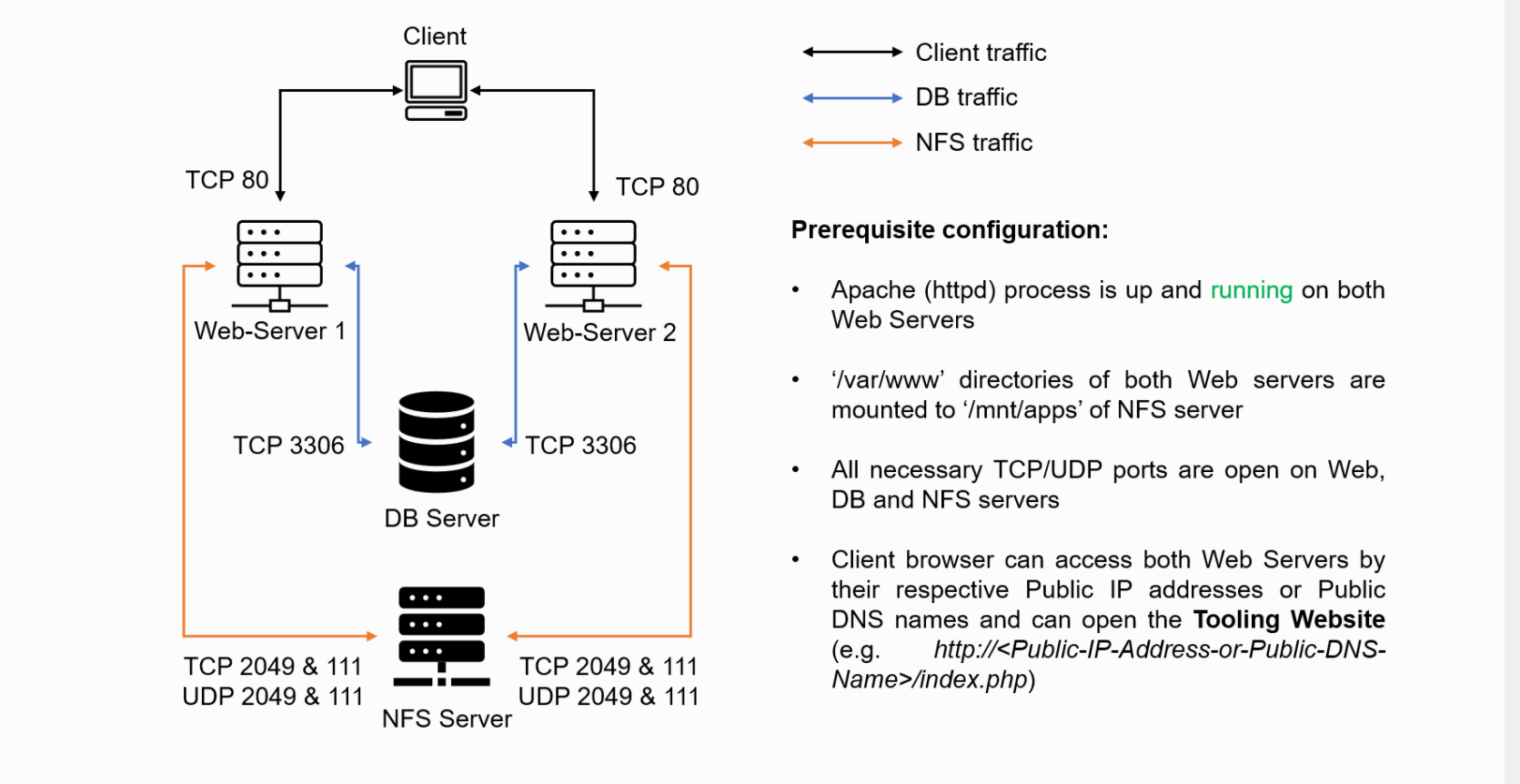


## Prerequisites

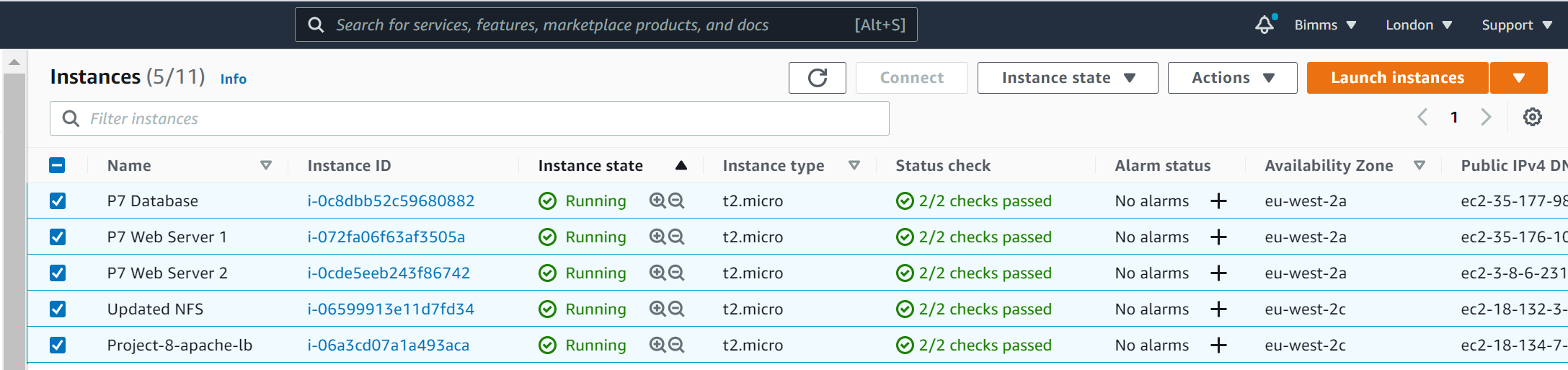
Ensure that the instances below are created on AWS:

1. Two RHEL8 Web Servers:(Web servers 1&2 as configured in Project 7)
2. One MySQL DB Server for Apache LB (based on Ubuntu 20.04)
3. One RHEL8 NFS server

**Target Architecture:**



AWS Instances



**STEPS:**

## 

## Configure Apache As A Load Balancer

1. Create an Ubuntu Server 20.04 EC2 instance and name it Project-8-apache-lb.
2. Create a Security Group and open the TCP port 80 by adjusting the inbound rule.
3. Install Apache Load Balancer on Project-8-apache-lb server and configure it to point traffic coming to LB to both Web Servers:

*#Install apache2*

sudo apt update

sudo apt install apache2 -y

sudo apt-get install libxml2-dev

*#Enable following modules:*

sudo a2enmod rewrite

sudo a2enmod proxy

sudo a2enmod proxy\_balancer

sudo a2enmod proxy\_http

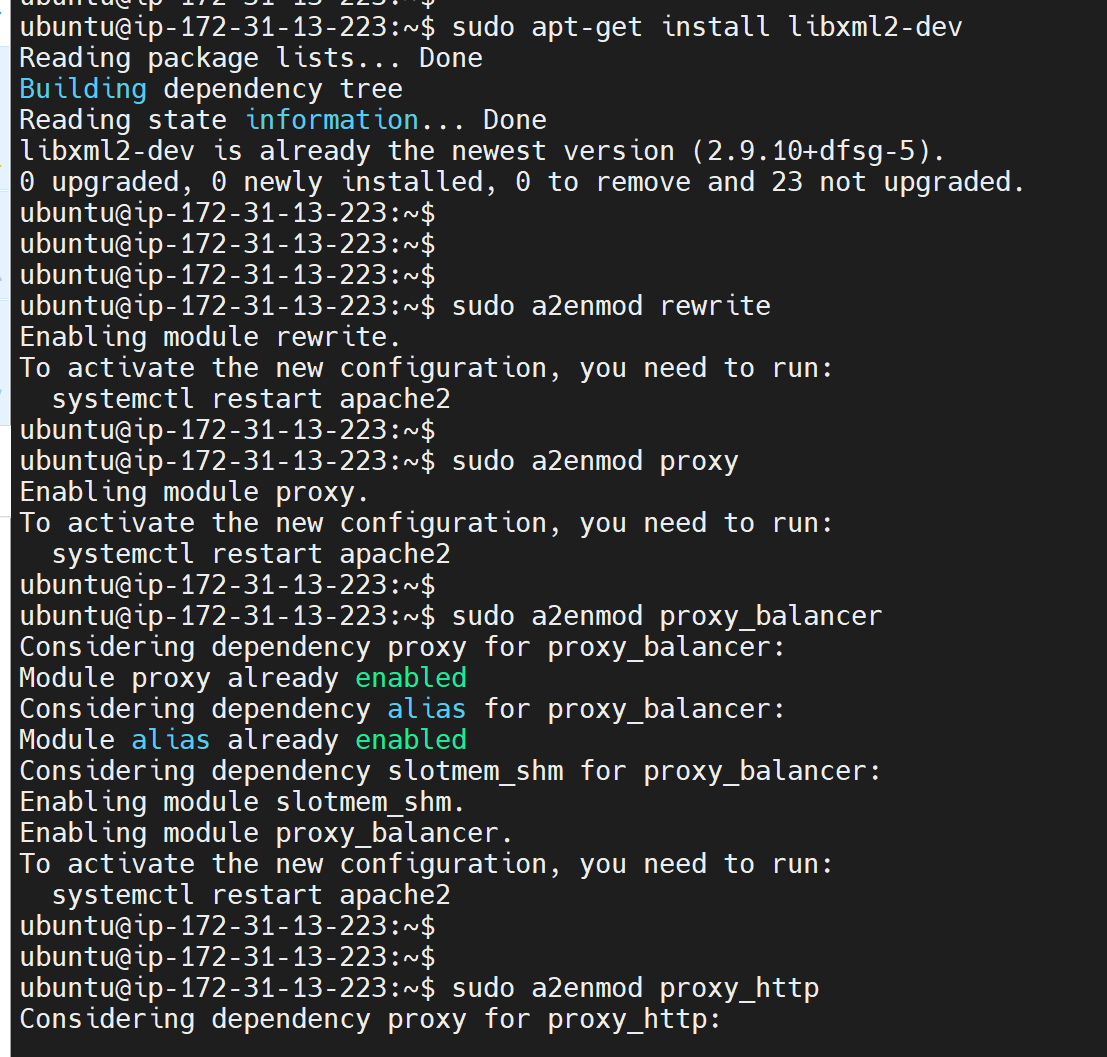
sudo a2enmod headers

sudo a2enmod lbmethod\_bytraffic

*#Restart apache2 service*

sudo systemctl restart apache2

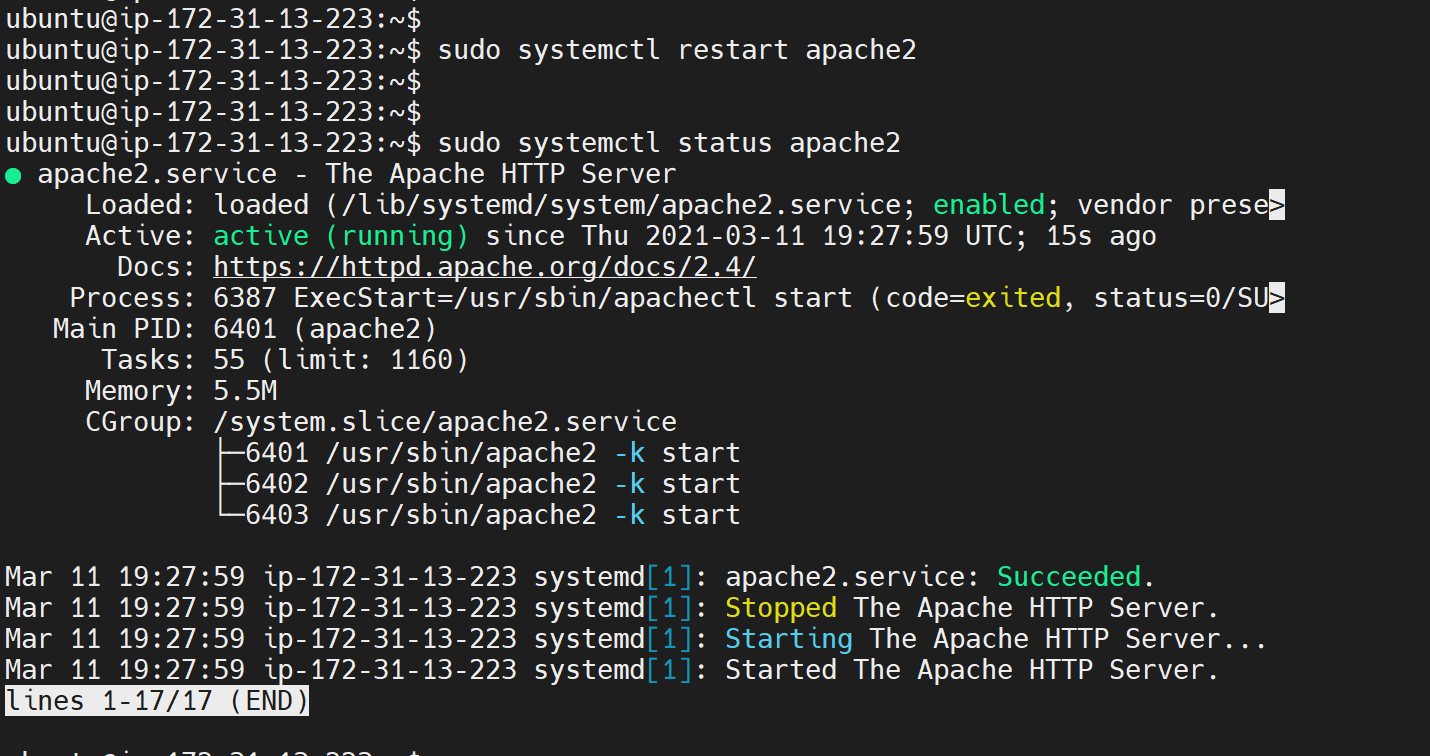
**Output**



d. Verify that apache2 is up and running:

sudo systemctl status apache2

Output



**2. Configure load balancing**

Go into the load balancing configuration file and add the below. Ensure that the web servers’ IP addresses are replaced.

sudo vi /etc/apache2/sites-available/000-default.conf

*#Add this configuration into this section <VirtualHost \*:80> </VirtualHost>*

<Proxy "balancer://mycluster">

BalancerMember http://<WebServer1-Private-IP-Address>:80 loadfactor=5 timeout=1

BalancerMember http://<WebServer2-Private-IP-Address>:80 loadfactor=5 timeout=1

ProxySet lbmethod=bytraffic

*# ProxySet lbmethod=byrequests*

</Proxy>

ProxyPreserveHost On

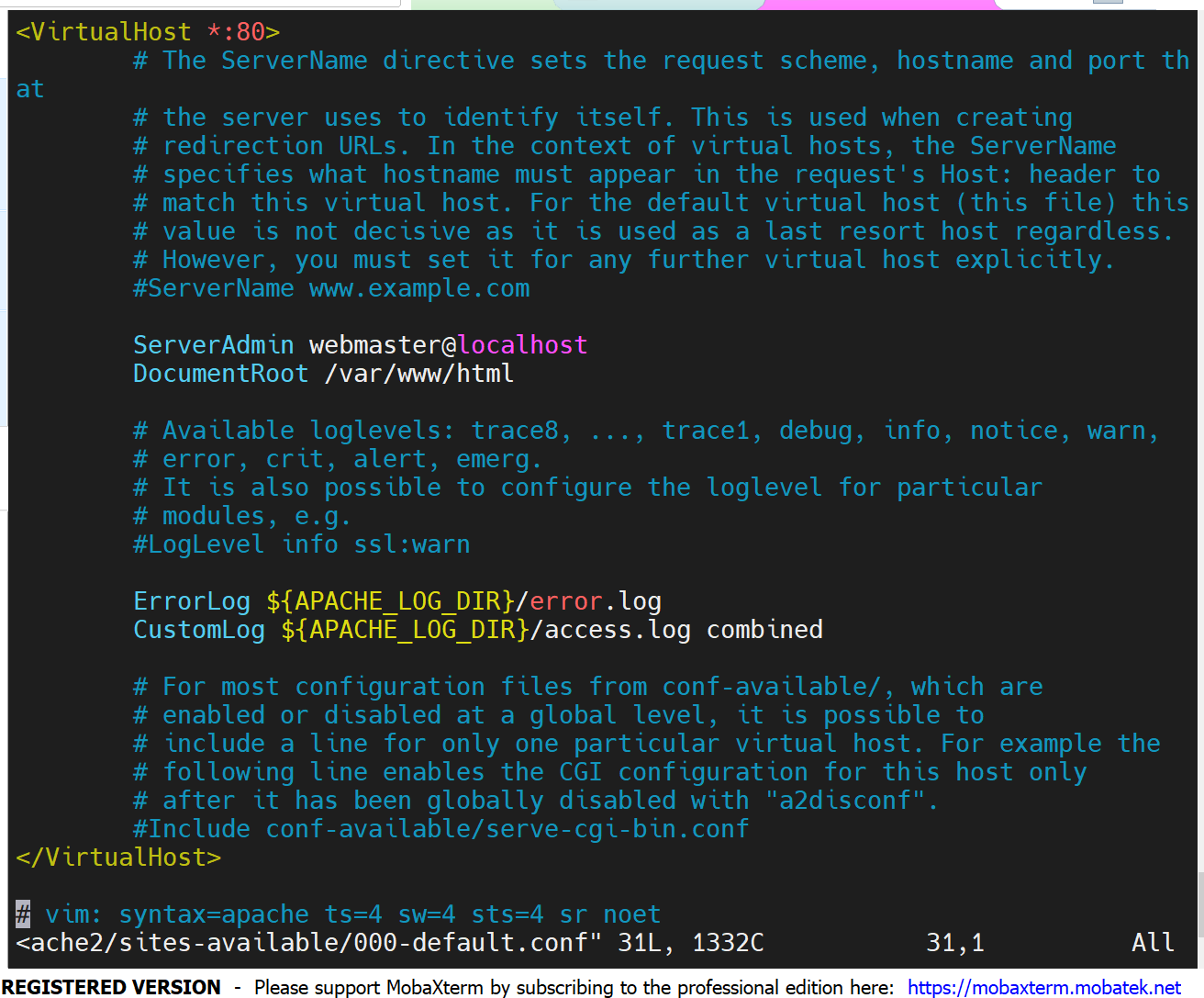
ProxyPass / balancer://mycluster/

ProxyPassReverse / balancer://mycluster/

*#Restart apache server*

#sudo systemctl restart apache2

Pre Configuration:



Post Configuration:

Add the below into the load balancer configuration file:

BalancerMember http://<WebServer1-Private-IP-Address>:80 loadfactor=5 timeout=1

BalancerMember http://<WebServer2-Private-IP-Address>:80 loadfactor=5 timeout=1

BalancerMember http://172.31.16.162:80 loadfactor=5 timeout=1

BalancerMember http://172.31.16.41:80 loadfactor=5 timeout=1

#Add this configuration into this section <VirtualHost \*:80> </VirtualHost>

<Proxy "balancer://mycluster">

BalancerMember http://172.31.16.162:80 loadfactor=5 timeout=1

BalancerMember http://172.31.16.41:80 loadfactor=5 timeout=1

ProxySet lbmethod=bytraffic

# ProxySet lbmethod=byrequests

</Proxy>

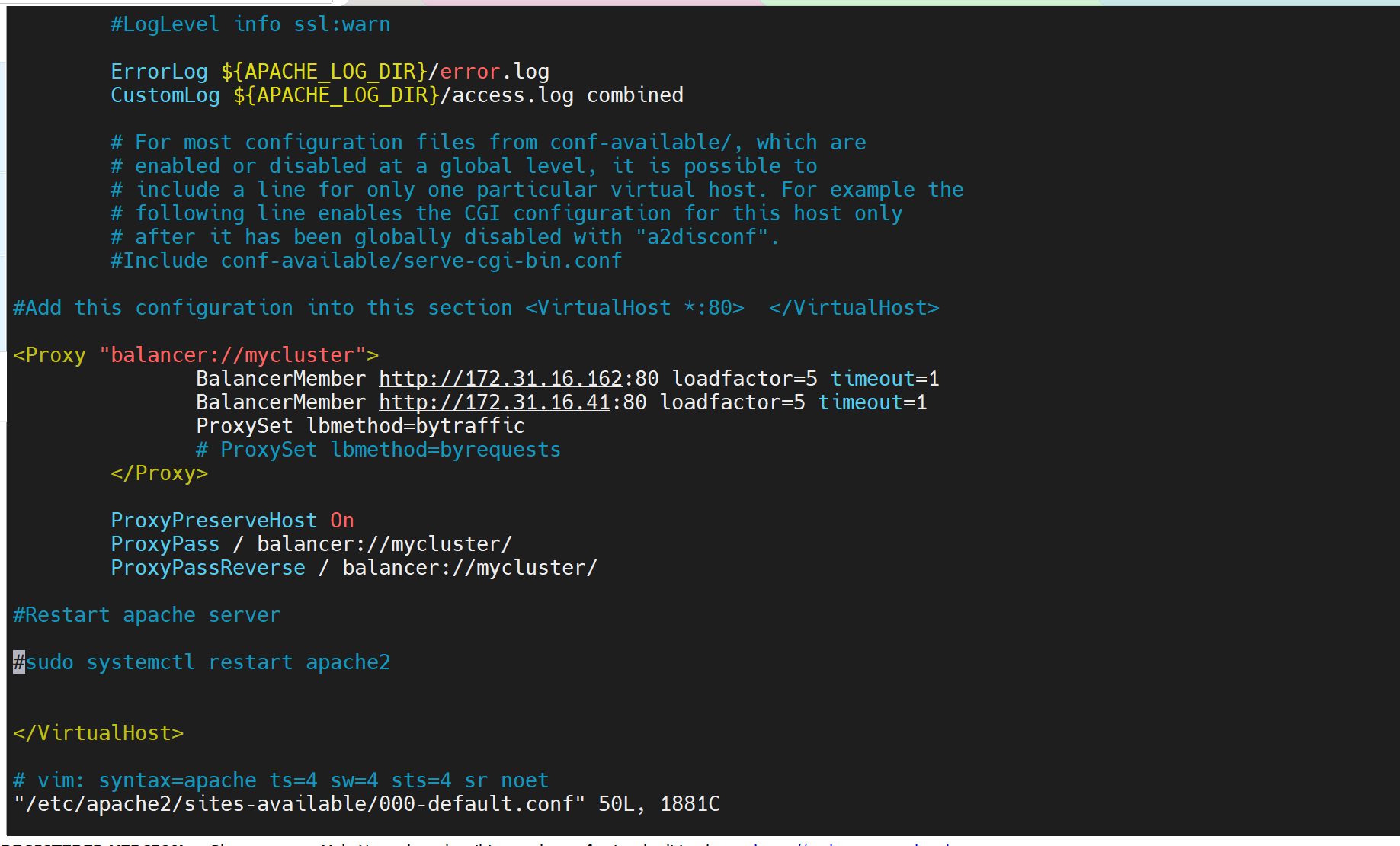
ProxyPreserveHost On

ProxyPass / balancer://mycluster/

ProxyPassReverse / balancer://mycluster/

#Restart apache server

#sudo systemctl restart apache2



Terms:

b[ytraffic](https://httpd.apache.org/docs/2.4/mod/mod_lbmethod_bytraffic.html): balancing method will distribute incoming load between your Web Servers according to current traffic load.

loadfactor: We can control in which proportion the traffic must be distributed by loadfactor parameter.

You can also study and try other methods, like: [bybusyness](https://httpd.apache.org/docs/2.4/mod/mod_lbmethod_bybusyness.html), [byrequests](https://httpd.apache.org/docs/2.4/mod/mod_lbmethod_byrequests.html), [heartbeat](https://httpd.apache.org/docs/2.4/mod/mod_lbmethod_heartbeat.html)

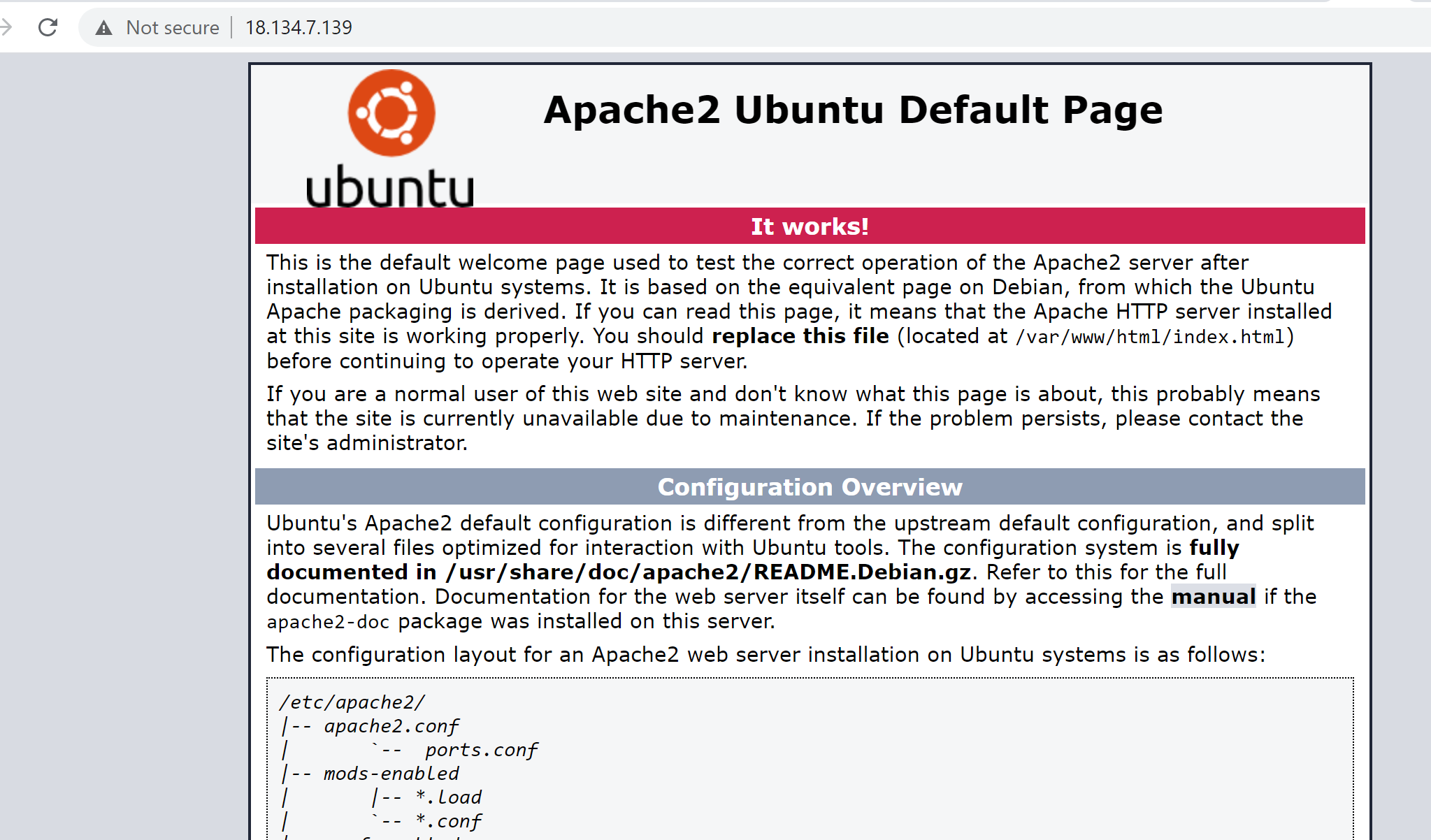
**3. Confirm that the configuration is working**

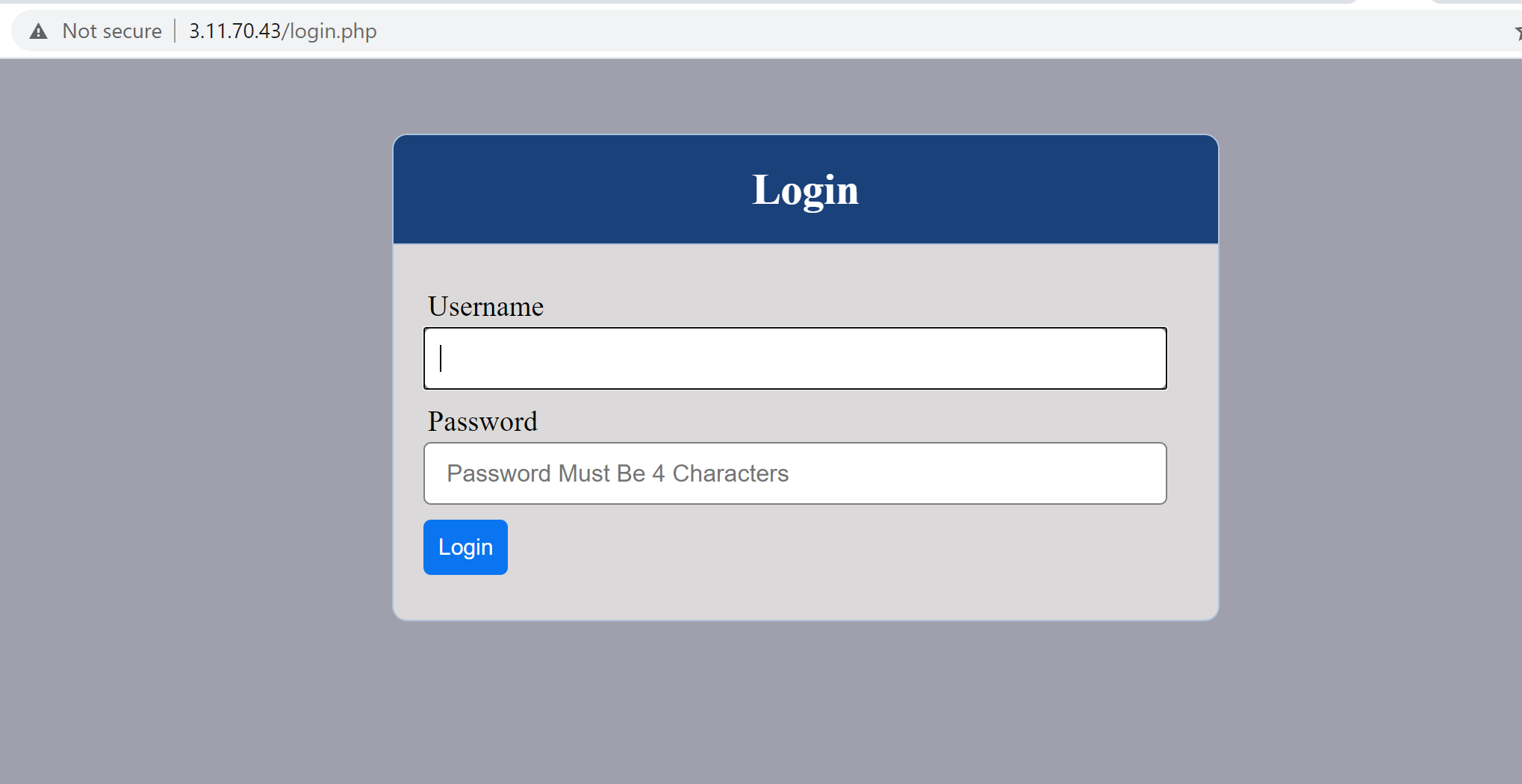
Try to access your LB’s public IP address or Public DNS name from your browser:

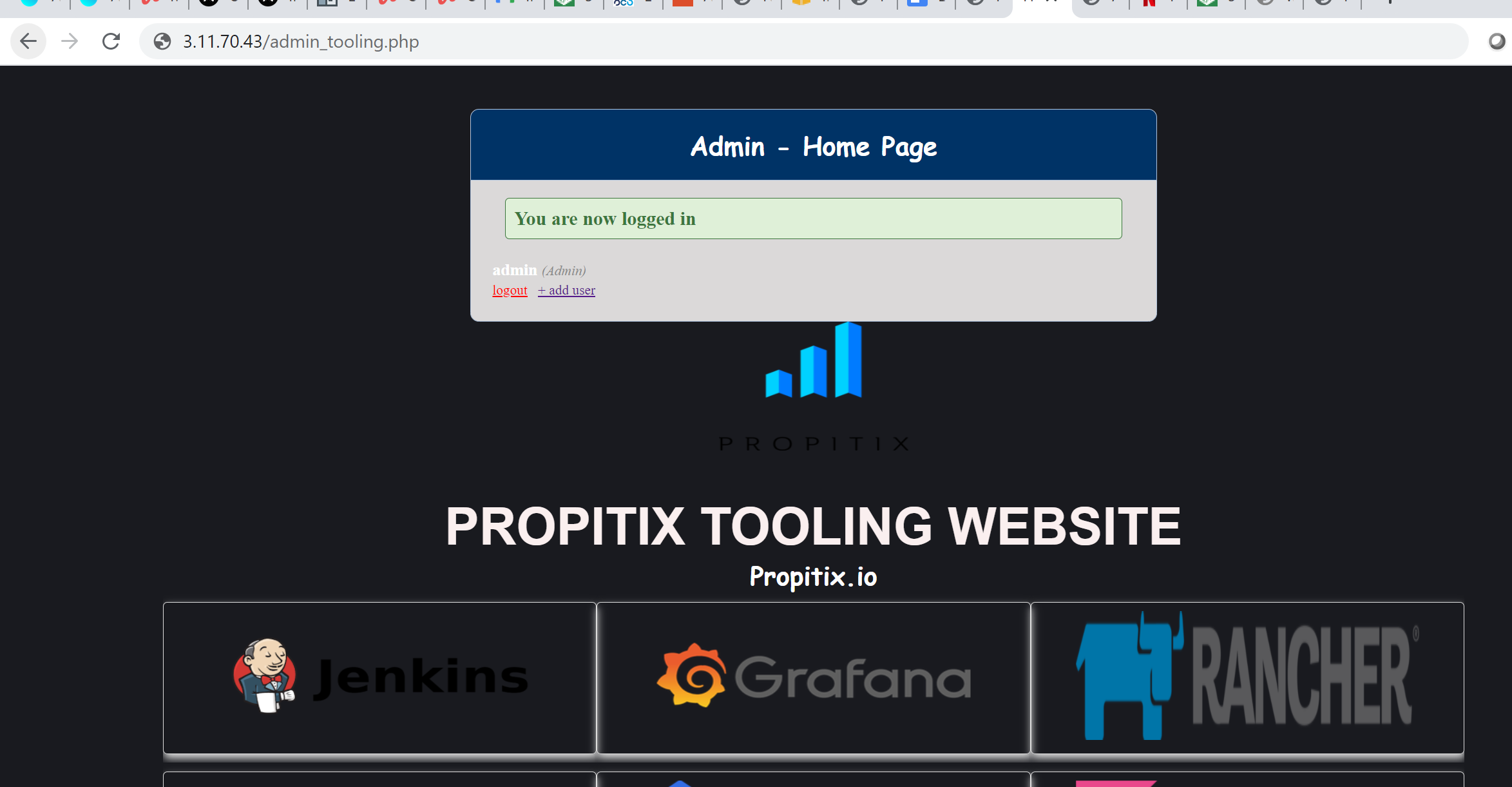
http://<Load-Balancer-Public-IP-Address-**or**-Public-DNS-Name>/index.php

http://18.134.7.139 /index.php

http://3.11.70.43 /index.php







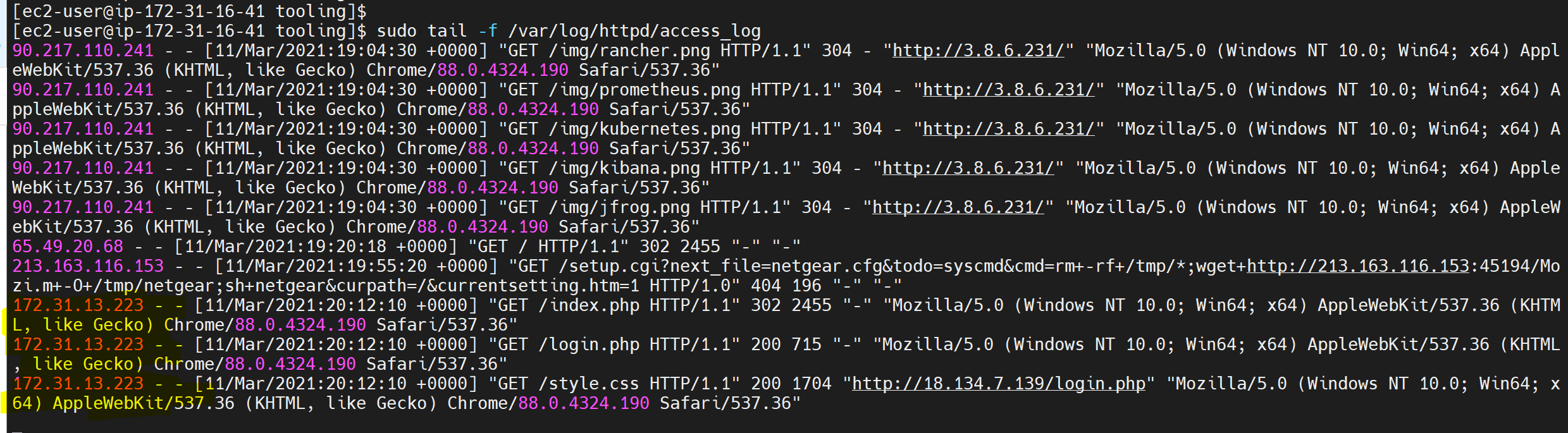
Note: If in the Project-7 you mounted /var/log/httpd/ from your Web Servers to the NFS server - unmount them and make sure that each Web Server has its own log directory.

Open two ssh/Putty consoles for both Web Servers and run following command:

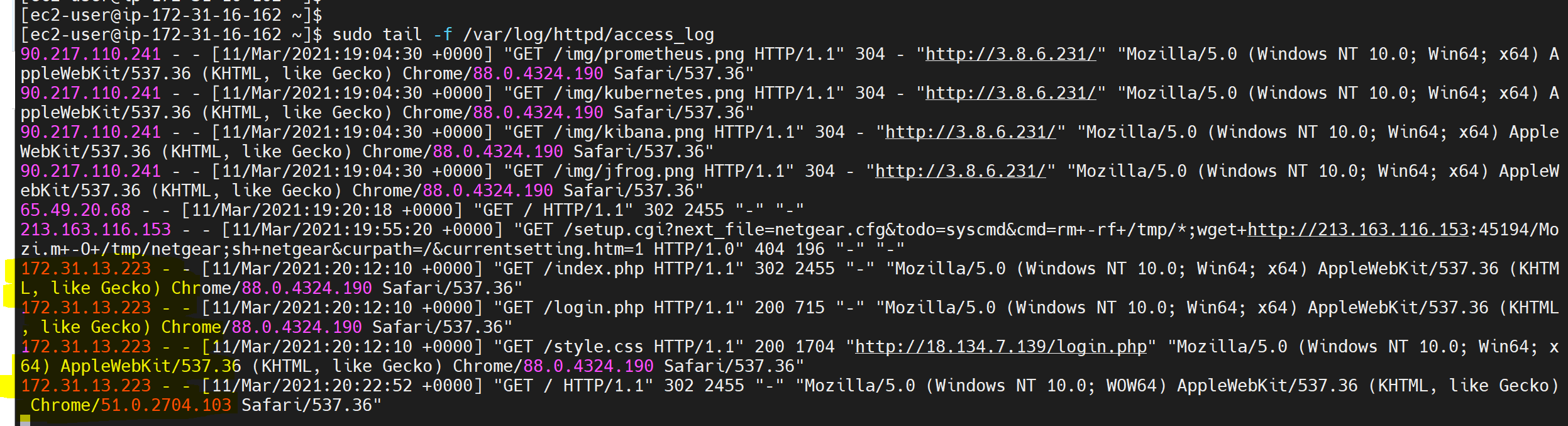
sudo tail -f /var/log/httpd/access\_log

Output:

Web Server 2



Web Server 1



Try to refresh the browser page http://<Load-Balancer-Public-IP-Address-or-Public-DNS-Name>/index.php several times and make sure that both servers receive HTTP GET requests from your LB - new records must appear in each server’s log file. The number of requests to each server will be approximately the same since we set loadfactor to the same value for both servers - it means that traffic will be disctributed evenly between them.

If you have configured everything correctly - your users will not even notice that their requests are served by more than one server.

## **4. Configure Local DNS Names Resolution**

Sometimes it is tedious to remember and switch between IP addresses, especially if you have a lot of servers under your management. What we can do, is to configure local domain name resolution. The easiest way is to use /etc/hosts file, although this approach is not very scalable, but it is very easy to configure and shows the concept well. So let us configure IP address to domain name mapping for our LB.

STEPS:

Open this file on your LB server

sudo vi /etc/hosts

Add *2 records into this file with Local IP address and arbitrary name for both of your Web Servers*

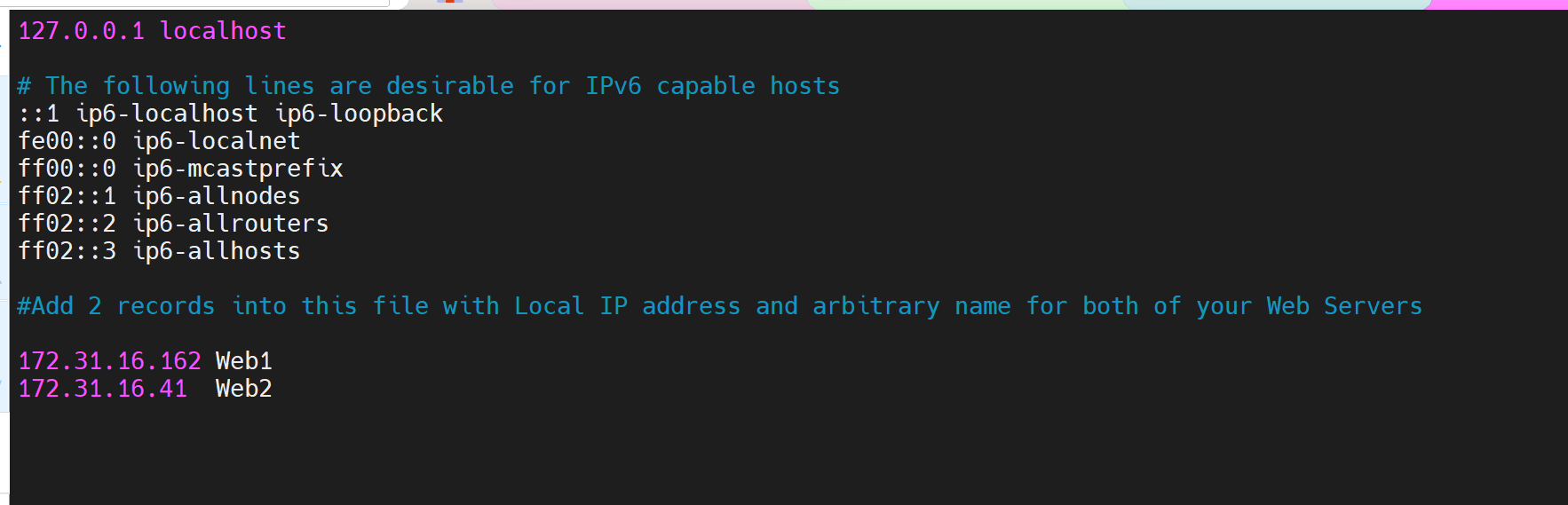
<WebServer1-Private-IP-Address> Web1

<WebServer2-Private-IP-Address> Web2

172.31.16.162 Web1

172.31.16.41 Web2

Output



Now let’s update the LB config file with those names instead of IP addresses.

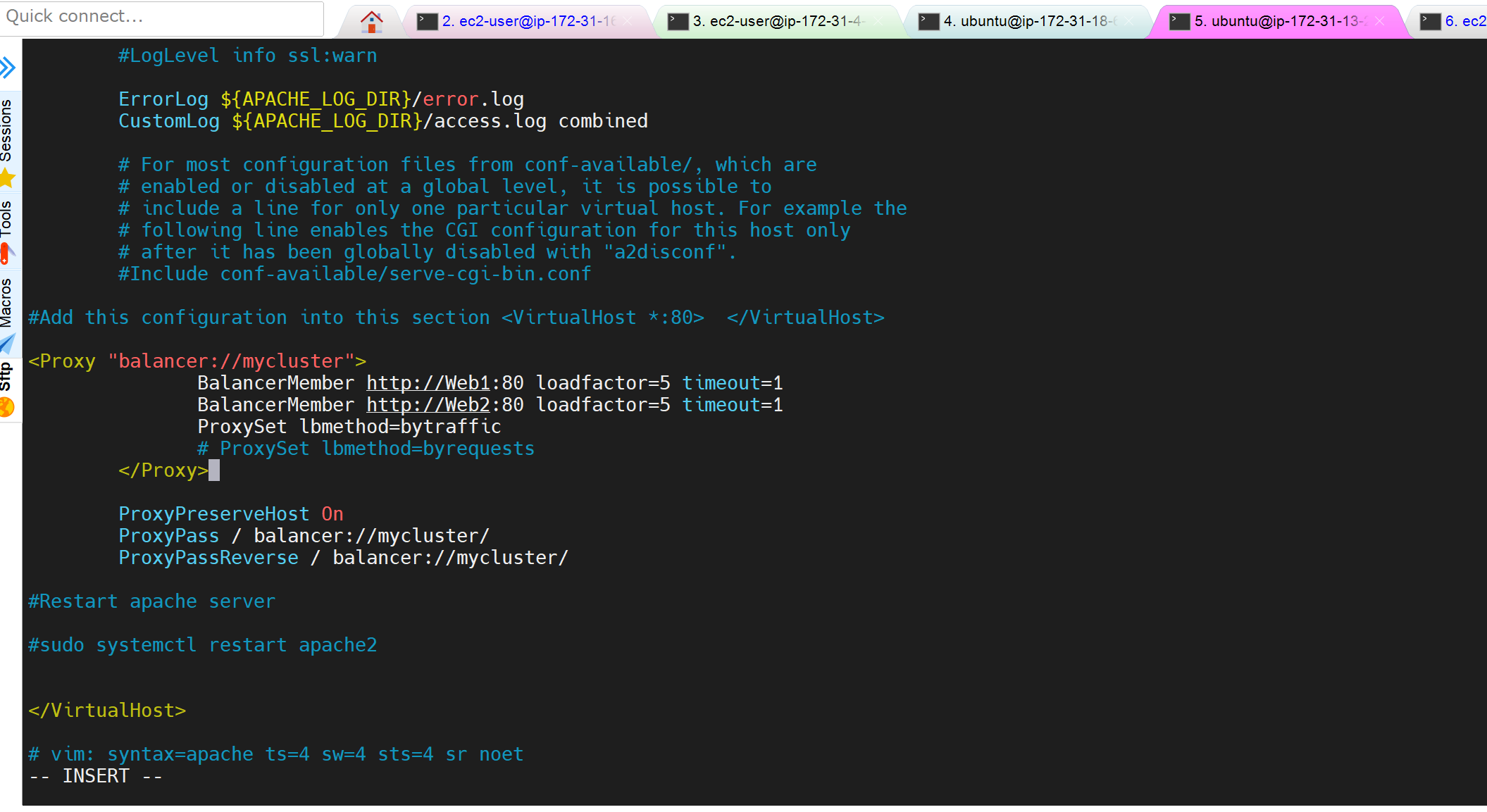
BalancerMember http://Web1:80 loadfactor=5 timeout=1

BalancerMember http://Web2:80 loadfactor=5 timeout=1

BalancerMember http://Web1:80 loadfactor=5 timeout=1

BalancerMember http://Web2:80 loadfactor=5 timeout=1

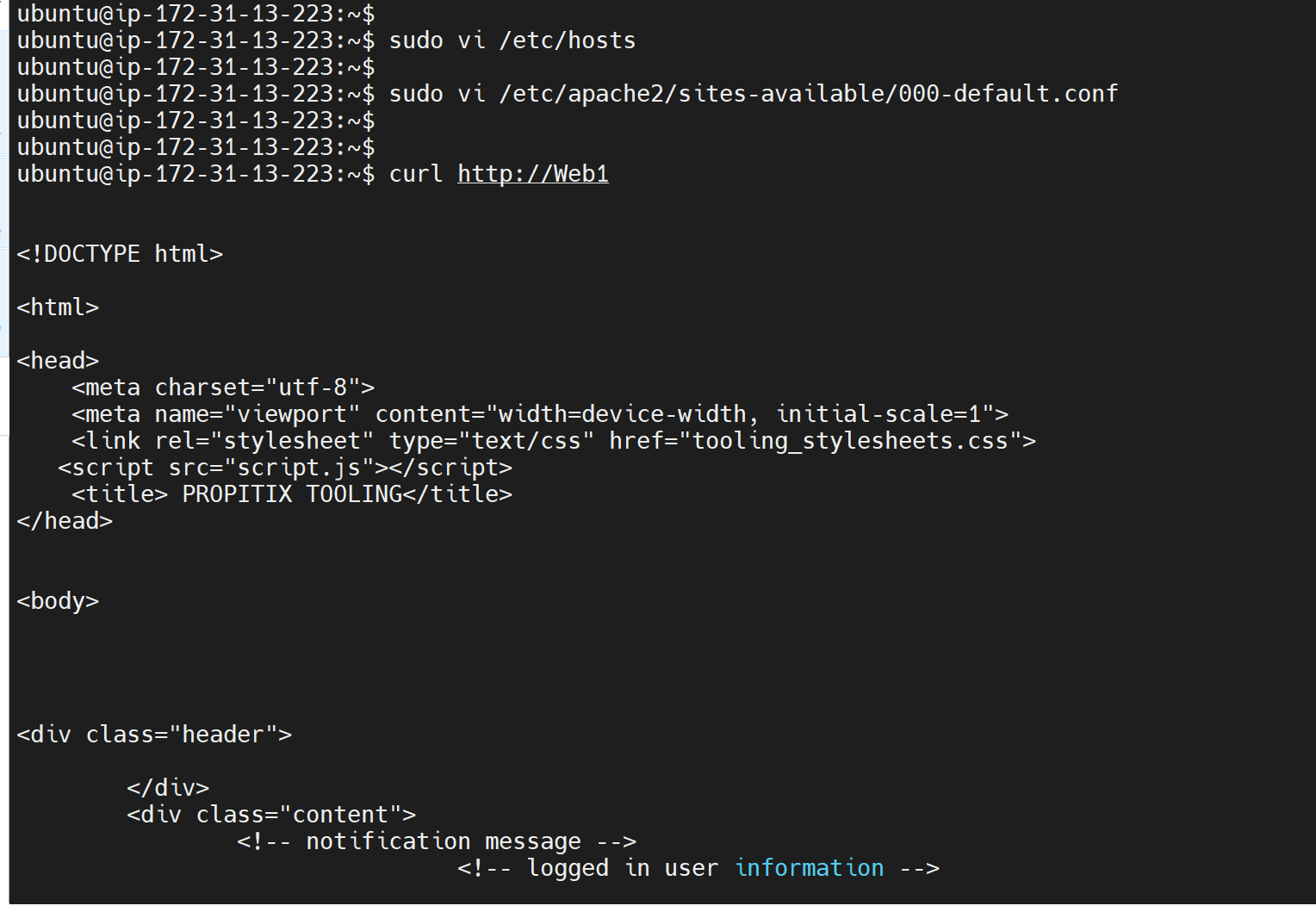
Output

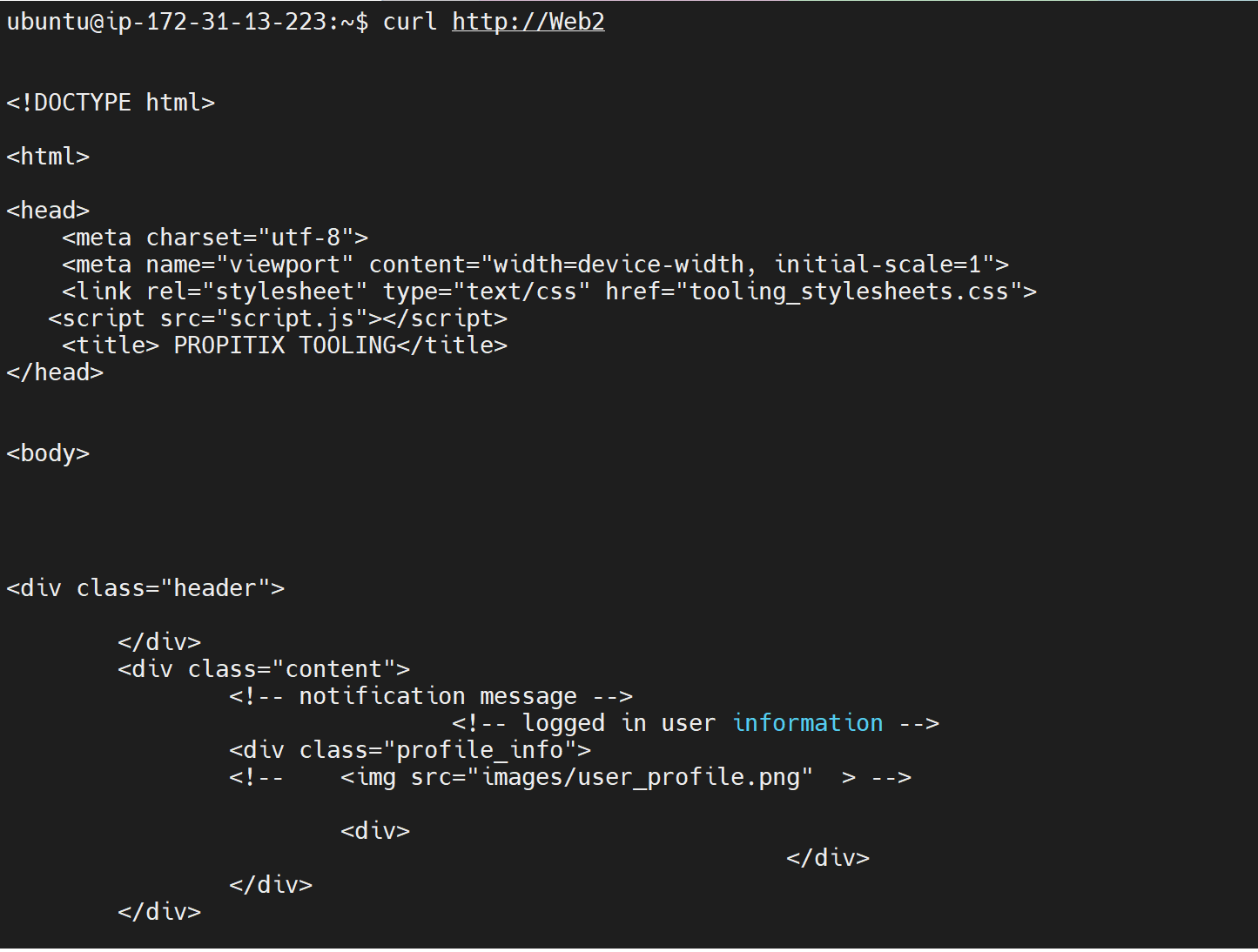


Now, curl both Web Servers from LB locally

curl http://Web1

curl http://Web2





This is only internal configuration and it is also local to the LB server, these names: web 1 and web 2, will neither be ‘resolvable’ from other servers internally nor from the Internet.

**Credits;**

darey.io

Load Balancer: <https://www.nginx.com/resources/glossary/load-balancing/>

Layer 4 Network Load balancing:

<https://www.nginx.com/resources/glossary/layer-4-load-balancing/>

Layer 7 Application Load Balancer:

<https://www.nginx.com/resources/glossary/layer-7-load-balancing/>

[Apache mod\_proxy\_balancer module](https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html) and Sticky session:

<https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html>

<https://kemptechnologies.com/uk/load-balancing/http-load-balancer/?gclid=CjwKCAiA4rGCBhAQEiwAelVti2T-lpA97ZssiqG1PHROrbfqCPTN0yz_LgoMQDxXleJ09WoDa619FRoC-XAQAvD_BwE>