Documentation on how to setup caddy and balance the load

Submitted By

Pushpender

Transform iptables into TCP load balancer

Task requirement

To create two new Caddy LB or two nginx and after creating, it needs to forward the 50% of Load to another LB.

iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall.

Environment details

iptable rules will be applicable for all types of OS.

UbuntuNAME="Ubuntu" VERSION="20.04.5

List of tools and technologies

Caddy:- CADDY_VERSION=v2.7.5

I have made two virtual machines so that they can work like a server.

For VM1 caddy installation

VM1 :- 192.168.122.19

Update and upgrade system commands

sudo apt update

sudo apt upgrade

Download and install Caddy, a web server, using the given below commands.

These commands are to be run in the virtual machine that we have created to run the caddy

To install caddy first we have to install the curl package on our server. To do this, run this command.

If you are in a root directory then run this command.

apt install curl -y

Or if you are not in a root directory then run this command.

sudo apt install curl -y

After that install the necessary dependencies to install the caddy web server.

Import the official GPG key for caddy. The keys are imported using these commands.

sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list

Update your system then install the caddy.

sudo apt update

sudo apt install caddy

It automatically starts caddy as a system service name caddy. The status of which you can check with the following command.

systemctl status caddy

Verify Caddy Installation: Confirm the installation of Caddy by checking its version.

caddy version

If everything works for you then it's time to check it by going to the server IP.

Go to the web browser of the same virtual machine in which you have installed caddy and in the url bar type:-

localhost:80

To install vim in a virtual machine.

sudo apt install vim

Then go to the terminal of the virtual machine 1.

cd /usr/share/caddy

ls

sudo vim index.html

Add below content:-

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello rohit</title>
</head>
<body>
        <h1>Hello rohit</h1>
</body>
```

For VM2 caddy installation

VM2:- 192.168.122.103

Update and upgrade system commands.

sudo apt update

sudo apt upgrade

Download and Install Caddy: Download and install Caddy, a web server, using the given below commands.

These commands are to be run in the virtual machine that we have created to run the caddy.

To install caddy first we have to install the curl package on our server. To do this, run this command.

If you are in a root directory then run this command.

apt install curl -y

Or if you are not in a root directory then run this command.

sudo apt install curl -y

-y flag means "yes" to any prompts, so it automatically agrees to the upgrades without asking for confirmation.

After that install the necessary dependencies to install the caddy web server.

Import the official GPG key for caddy. The keys are imported using these commands.

sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list

Update your system then install the caddy.

sudo apt update

sudo apt install caddy

It automatically starts caddy as a system service name caddy. The status of which you can check with the following command.

systemctl status caddy

Verify Caddy Installation: Confirm the installation of Caddy by checking its version.

caddy version

If everything works for you then it's time to check it by going to the server IP.

Go to the web browser of the same virtual machine in which you have installed caddy and in the url bar type:-

localhost:80

To install vim in a virtual machine.

sudo apt install vim

Then go to the terminal of the virtual machine 2.

cd /usr/share/caddy

ls

ls command is used to show a list of files

sudo vim index.html

Vim is a text editor in linux like OS. It is used to create and edit text files.

Add below content:-

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello pushpender</title>
</head>
<body>
        <h1>Hello pushpender</h1>
</body>
</html>
```

Curl on virtual machines

I am able to curl pages from both virtual machines.

curl http://192.168.122.19

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello rohit</title>
</head>
<body>
        <h1>Hello rohit</h1>
</body>
```

curl http://192.168.122.103

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello pushpender</title>
</head>
<body>
        <h1>Hello pushpender</h1>
</body>
</html>
```

Curl on base machine

I am able to curl from Base Machine.

curl http://192.168.122.19

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello rohit</title>
</head>
<body>
        <h1>Hello rohit</h1>
</body>
```

curl

http://192.168.122.103

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello pushpender</title>
</head>
<body>
        <h1>Hello pushpender</h1>
</body>
</html>
```

Commands that to be run on VM1

Now work on Virtual machine 1

a) Enable IP Forwarding:

Activate IP forwarding to facilitate traffic routing between interfaces:

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo vim /etc/sysctl.conf

uncomment in the file below line.

net.ipv4.ip_forward=1

net.ipv4.conf.all.accept_redirects = 0

net.ipv4.conf.all.accept_source_route = 0

net.ipv4.conf.all.log_martians = 1

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo sysctl -p

Modify source IP for forwarding: this command run vm1 source

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo iptables -A POSTROUTING -t nat -p tcp -d 192.168.122.103 --dport 80 -j SNAT --to-source 192.168.122.19

Set default connection drop vm1

This implies that by default, any incoming traffic that doesn't match specific predefined rules will be dropped or rejected. In other words, if the firewall doesn't have a rule that explicitly allows the traffic to pass through, it

will block it as a security measure. This approach follows the principle of allowing only known and authorised traffic, enhancing the network's security.

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo iptables -t filter -P FORWARD DROP

iptables: This is a command-line utility in Linux used to configure the netfilter firewall (which is built into the Linux kernel). It allows you to set up rules to filter or manipulate network packets.

-t filter: Specifies the table to work with. In this case, it's the 'filter' table, which is the default table used by iptables for packet filtering.

-P FORWARD DROP: This part of the command sets the default policy for the FORWARD chain to DROP. The FORWARD chain is responsible for packets that are being forwarded through the system from one network interface to another. When the policy is set to DROP, it means that by default, any packets that don't match any specific rules in the FORWARD chain will be dropped (i.e., not forwarded).

Accept Traffic to the Server:

This involves creating rules that explicitly allow certain types of traffic to reach a designated server. For instance, if you have a server at IP address 192.168.122.44 and it's listening on port 80, you can configure the firewall to accept incoming traffic destined for that server's IP and port. This is done using firewall rules that specify the source, destination, protocol, and port of the allowed traffic. Allow specific traffic from particular sources and to specific destinations (servers) on specific ports (--dport for destination port and --sport for source port). This creates a controlled and secure network environment where only the specified traffic is permitted to traverse the firewall.

This command run on Virtual Machine 1

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo iptables -t filter -A FORWARD -d 192.168.122.103 -p tcp --dport 80 -j ACCEPT

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo iptables -t filter -A FORWARD -s 192.168.122.103 -p tcp --sport 80 -j ACCEPT

iptables: The command-line utility used for managing the netfilter firewall in Linux.

-t filter: Specifies the table to operate on. In this case, it's the 'filter' table, which is the default table for packet filtering.

-A FORWARD: This option appends a rule to the FORWARD chain. The FORWARD chain is responsible for packets being forwarded through the system from one network interface to another.

-d 192.168.122.103: Specifies the destination IP address as 192.168.122.103. This rule will match packets whose destination address is this specific IP.

-p tcp: Specifies the protocol. In this case, it's TCP, which is used for many common types of internet connections.

--dport 80: Specifies the destination port. This rule matches packets whose destination port is 80. Port 80 is commonly used for HTTP web traffic.

-j ACCEPT: This part of the command determines the action to take if a packet matches all the criteria mentioned above. In this case, it's ACCEPT, which means the packets meeting the conditions will be allowed to pass through the firewall.

Command to be run on Virtual Machine 2

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

iptables: The command-line utility used for managing the netfilter firewall in Linux.

-A INPUT: This option appends a rule to the INPUT chain. The INPUT chain is responsible for handling packets destined for the local system itself.

-p tcp: Specifies the protocol. In this case, it's TCP, which is used for many common types of internet connections.

--dport 80: Specifies the destination port. This rule matches packets whose destination port is 80. Port 80 is commonly used for HTTP web traffic.

-j ACCEPT: This part of the command determines the action to take if a packet matches all the criteria mentioned above. In this case, it's ACCEPT, which means the packets meeting the conditions will be allowed to pass through the firewall.

Apply Random Load Balancing and then Round Robin:-

A technique used in network environments to distribute incoming traffic across multiple servers in a random manner. This method is employed to optimise resource utilisation, prevent overload on individual servers, and enhance the overall performance and reliability of a system. In this case, incoming traffic destined for the IP address 192.168.122.123 on port 80 is subject to random distribution between one destinations (192.168.122.13:80) The --mode random option ensures that the traffic is distributed based on a randomised algorithm, with different probabilities assigned to each destination (--probability 0.33 and --probability 0.5 in this case). The goal of applying random load balancing is to distribute traffic unpredictably, achieving a fair distribution of incoming requests among the specified destinations. This way, the servers can collectively handle the load more efficiently, minimising the risk of overloading any single server and contributing to improved performance and fault tolerance.

Command to be run on Virtual Machine 1

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo iptables -A PREROUTING -t nat -p tcp -d 192.168.122.19 --dport 80 \

```
-m statistic --mode random --probability 0.33 \
-j DNAT --to-destination 192.168.122.103:80
```

-A PREROUTING: This option appends a rule to the PREROUTING chain in the nat table. The PREROUTING chain is used to modify packets as soon as they come in, before any routing decisions are made.

-t nat: Specifies the table to operate on. In this case, it's the 'nat' table, used for Network Address Translation.

-p tcp: Specifies the protocol. In this case, it's TCP.

-d 192.168.122.19 --dport 80: Specifies the destination IP address (192.168.122.19) and the destination port (80). This rule will match incoming TCP packets destined for port 80 on the specified IP address.

-m statistic --mode random --probability 0.33: This part of the command uses the 'statistic' module to introduce randomness into the rule. It sets a probability of 33% (0.33) using the '--probability' option. Essentially, this rule will only match packets randomly with a 33% chance.

-j DNAT --to-destination 192.168.122.103:80: If a packet matches the criteria mentioned above (based on the destination IP and port and the probability condition), this part of the command specifies the action to take. It uses Destination Network Address Translation (DNAT) to redirect the packets to a different destination address and port. In this case, it redirects the packets to 192.168.122.103 on port 80.

Capture Specific Port Traffic:

Use tcpdump to monitor network traffic on a specific port:-

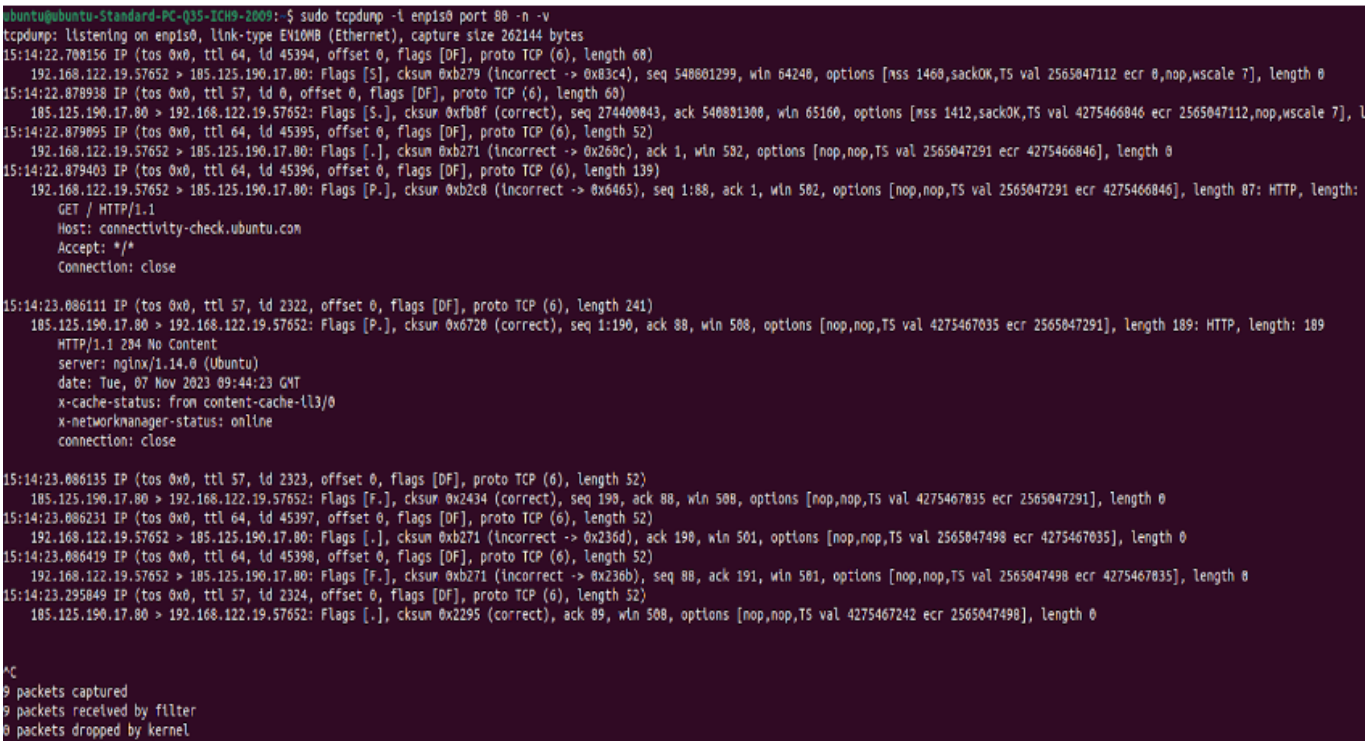
sudo tcpdump -i enp1s0 port 80 -n

Output of TCP Dump

1st curl hit:-

```
ubuntu@ubuntu-Standard-PC-Q35-ICH9-2080:~$ sudo tcpdump -i enp1s0 port 80 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:09:22.714976 IP 192.168.122.19.34278 > 35.232.111.17.80: Flags [S], seq 2470998819, win 64240, options [mss 1460,sackOK,TS val 540937503 ecr 0,nop,wscale 7], length 0
15:09:23.055424 IP 35.232.111.17.80 > 192.168.122.19.34278: Flags [S.], seq 3840651490, ack 2470998820, win 64768, options [mss 1412,sackOK,TS val 4182517993 ecr 540937503,nop,wscale 7], length 0
15:09:23.055527 IP 192.168.122.19.34278 > 35.232.111.17.80: Flags [.], ack 1, win 502, options [nop,nop,TS val 540937843 ecr 4182517993], length 0
15:09:23.055983 IP 192.168.122.19.34278 > 35.232.111.17.80: Flags [P.], seq 1:88, ack 1, win 502, options [nop,nop,TS val 540937844 ecr 4182517993], length 87: HTTP: GET / HTTP/1.1
15:09:23.464786 IP 35.232.111.17.80 > 192.168.122.19.34278: Flags [.], ack 88, win 508, options [nop,nop,TS val 4182518381 ecr 540937844], length 0
15:09:23.464722 IP 35.232.111.17.80 > 192.168.122.19.34278: Flags [P.], seq 1:149, ack 88, win 508, options [nop,nop,TS val 4182518381 ecr 540937844], length 148: HTTP: HTTP/1.1 204 No Content
15:09:23.464723 IP 35.232.111.17.80 > 192.168.122.19.34278: Flags [F.], seq 149, ack 88, win 508, options [nop,nop,TS val 4182518383 ecr 540937844], length 0
15:09:23.464796 IP 192.168.122.19.34278 > 35.232.111.17.80: Flags [.], ack 149, win 501, options [nop,nop,TS val 540938253 ecr 4182518381], length 0
15:09:23.464983 IP 192.168.122.19.34278 > 35.232.111.17.80: Flags [F.], seq 88, ack 150, win 501, options [nop,nop,TS val 540938253 ecr 4182518383], length 0
15:09:23.773254 IP 35.232.111.17.80 > 192.168.122.19.34278: Flags [.], ack 89, win 508, options [nop,nop,TS val 4182518789 ecr 540938253], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

2nd curl Hit:-



```
ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ sudo tcpdump -i enp1s0 port 80 -n -v
tcpdump: listening on enp1s0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:14:22.708156 IP (tos 0x0, ttl 64, id 45394, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.122.19.57652 > 185.125.190.17.80: Flags [S], cksum 0xb279 (incorrect -> 0x83c4), seq 548801299, win 64240, options [mss 1460,sackOK,TS val 2565047112 ecr 0,nop,wscale 7], length 0
15:14:22.878938 IP (tos 0x0, ttl 57, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    185.125.190.17.80 > 192.168.122.19.57652: Flags [S.], cksum 0xfb0f (correct), seq 274400843, ack 548801300, win 65160, options [mss 1412,sackOK,TS val 4275466846 ecr 2565047112,nop,wscale 7], l
15:14:22.879095 IP (tos 0x0, ttl 64, id 45395, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.122.19.57652 > 185.125.190.17.80: Flags [.], cksum 0xb271 (incorrect -> 0x269c), ack 1, win 502, options [nop,nop,TS val 2565047291 ecr 4275466846], length 0
15:14:22.879403 IP (tos 0x0, ttl 64, id 45396, offset 0, flags [DF], proto TCP (6), length 139)
    192.168.122.19.57652 > 185.125.190.17.80: Flags [P.], cksum 0xb2c8 (incorrect -> 0x6465), seq 1:88, ack 1, win 502, options [nop,nop,TS val 2565047291 ecr 4275466846], length 87: HTTP, length:
        GET / HTTP/1.1
        Host: connectivity-check.ubuntu.com
        Accept: */*
        Connection: close

15:14:23.086111 IP (tos 0x0, ttl 57, id 2322, offset 0, flags [DF], proto TCP (6), length 241)
    185.125.190.17.80 > 192.168.122.19.57652: Flags [P.], cksum 0x6720 (correct), seq 1:190, ack 88, win 508, options [nop,nop,TS val 4275467035 ecr 2565047291], length 189: HTTP, length: 189
        HTTP/1.1 204 No Content
        server: nginx/1.14.0 (Ubuntu)
        date: Tue, 07 Nov 2023 09:44:23 GMT
        x-cache-status: from content-cache-ll3/0
        x-networkmanager-status: online
        connection: close

15:14:23.086135 IP (tos 0x0, ttl 57, id 2323, offset 0, flags [DF], proto TCP (6), length 52)
    185.125.190.17.80 > 192.168.122.19.57652: Flags [F.], cksum 0x2434 (correct), seq 190, ack 88, win 508, options [nop,nop,TS val 4275467035 ecr 2565047291], length 0
15:14:23.086231 IP (tos 0x0, ttl 64, id 45397, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.122.19.57652 > 185.125.190.17.80: Flags [.], cksum 0xb271 (incorrect -> 0x236d), ack 190, win 501, options [nop,nop,TS val 2565047498 ecr 4275467035], length 0
15:14:23.086419 IP (tos 0x0, ttl 64, id 45398, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.122.19.57652 > 185.125.190.17.80: Flags [F.], cksum 0xb271 (incorrect -> 0x236b), seq 88, ack 191, win 501, options [nop,nop,TS val 2565047498 ecr 4275467035], length 0
15:14:23.295849 IP (tos 0x0, ttl 57, id 2324, offset 0, flags [DF], proto TCP (6), length 52)
    185.125.190.17.80 > 192.168.122.19.57652: Flags [.], cksum 0x2295 (correct), ack 89, win 508, options [nop,nop,TS val 4275467242 ecr 2565047498], length 0


^C
9 packets captured
9 packets received by filter
0 packets dropped by kernel
```

Load Testing:- (on base machine)

sudo apt install apache2-utils

Install ab command at local for load testing

Install apache2-utils for ab command for Load Testing

run load test using ab command :-

ubuntu@ubuntu-Standard-PC-Q35-ICH9-2009:~$ ab -n 1000 -c 100 http://192.168.122.19:80/

pushpender@pushpender:~$ sudo ab -n 1000 -c 100 http://192.168.122.19:80/

```
pushpender@pushpender-HP:~/Desktop$ sudo ab -n 1000 -c 100 http://192.168.122.19:80/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.122.19 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Caddy
Server Hostname:        192.168.122.19
Server Port:            80

Document Path:          /
Document Length:        234 bytes

Concurrency Level:      100
Time taken for tests:   1.078 seconds
Complete requests:      1000
Failed requests:        638
   (Connect: 0, Receive: 0, Length: 638, Exceptions: 0)
Total transferred:      459018 bytes
HTML transferred:       241018 bytes
Requests per second:    927.81 [#/sec] (mean)
Time per request:       107.781 [ms] (mean)
Time per request:       1.078 [ms] (mean, across all concurrent requests)
Transfer rate:          415.90 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   72 259.1      1    1032
Processing:     0    3   2.3      2      15
Waiting:        0    3   2.3      2      15
Total:          0   75 259.4      4    1034

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      6
  75%      7
  80%      8
  90%     11
  95%   1015
  98%   1026
  99%   1029
 100%   1034 (longest request)
pushpender@pushpender-HP:~/Desktop$
```

curl Output :-

curl http://192.168.122.19

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello rohit</title>
</head>
<body>
        <h1>Hello rohit</h1>
</body>
```

curl http://192.168.122.19

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Hello pushpender</title>
</head>
<body>
        <h1>Hello pushpender</h1>
</body>
</html>
```