

HAproxy is an open-source and lightweight package that offers high availability and load balancing for TCP and HTTP based programs. It distributes the load among the web and application servers. HAproxy is available for nearly all Linux distributions. It is a widely used load balancer that is popular for its efficiency, reliability, and low memory and CPU footprint. In this post, we will explain how to install and configure HAproxy on a Ubuntu system.

We have setup three machines. We will install HAproxy on one server and the Apache web servers on two servers. Our HAproxy server will then act as a load balancer and will distribute the load among Apache web servers.

Note: The procedure and commands mentioned in this post has been tested on **Ubuntu 20.04 LTS (Focal Fossa)**. The same procedure is also valid for Debian and Mint distributions.

Network Details

We will be using three Ubuntu servers; all on the same network. The details of our servers are as follows:

host [haproxy server]

pub : 3.111.186.151
prv : 172.31.37.244
hostname: Haproxy

.....

slave1

pub: 3.6.38.78
pvr : 172.31.39.22
Hostname: web-server1

.....

slave2

pub: 65.2.146.226
pvr : 172.31.37.241
Hostname: web-server2

```
Hostname: HAproxy, IP address: 172.31.37.244 (Frontend server)
```

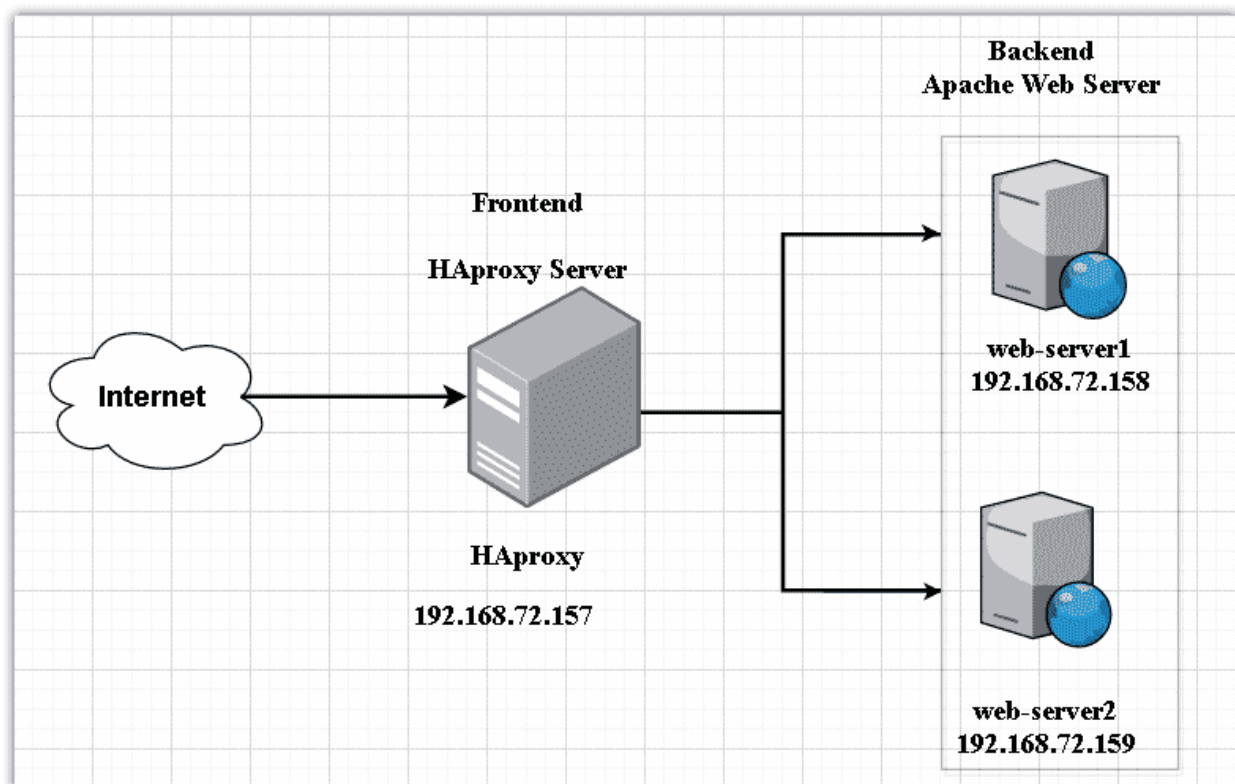
```
Hostname: web-server1, IP address: 172.31.39.22 (Backend servers)
```

```
Hostname: web-server2, IP address: 172.31.37.241 (Backend servers)
```

Note: You must have sudo privileges on all the servers.

We will configure one machine as a load balancer and the other two as web servers. The HAproxy server will be our front-end server that will receive the requests from the users and forward them to the two web servers. The web servers will be our Backend servers that will receive those forwarded requests.

This is how our setup looks like:



Setting up web servers-Backend servers

In this section, we will setup two web servers (**web-server1** and **web-server2**) as our backend servers.

On both web-servers do following things,

edit the `/etc/hosts` file:

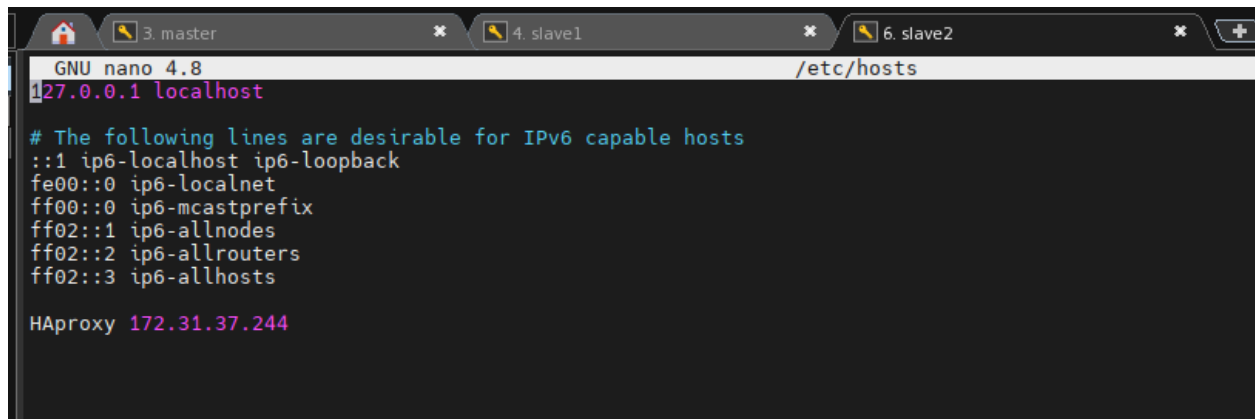
```
$ sudo nano /etc/hosts
```

Then add the hostname entry for HAproxy server as follows:

```
hostname-of-HAproxy IP-address-of-HAproxy-server
```

In our scenario, it would be:

```
HAproxy 172.31.37.244
```



```
GNU nano 4.8 /etc/hosts
1 127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

HAproxy 172.31.37.244
```

2. Setup Apache webserver

Now install Apache web server using the below command in Terminal. You can also visit our post on [How to install the Apache web server on Linux](#).

```
$ sudo apt install apache2 -y
```

Then enable and start the Apache service using the below commands in Terminal:

```
$ sudo systemctl enable apache2  
$ sudo systemctl start apache2
```

Create an index file for web-server1 using the below command in Terminal:

```
$ echo "<H1>Hello! This is webserver1 </H1>" >> /var/www/html/index.html
```

If a firewall is running on your system, you will need to allow Apache traffic through it:

```
$ sudo ufw allow 80/tcp
```

Then reload the firewall configurations:

```
$ ufw reload
```

Now try accessing the site in your web browser by typing http:// followed by either the IP address or the hostname of your web server.

```
http:// hostname-or-public-IP-address
```

```
root@ip-172-31-39-22:~#  
root@ip-172-31-39-22:~# echo "this is webserver1" >> /var/www/html/index.html  
root@ip-172-31-39-22:~# cat /var/www/html/index.html  
this is webserver1  
root@ip-172-31-39-22:~#
```

```
root@ip-172-31-37-241:~#  
root@ip-172-31-37-241:~# echo "this is webserver2" >> /var/www/html/index.html  
root@ip-172-31-37-241:~# cat /var/www/html/index.html  
this is webserver2  
root@ip-172-31-37-241:~#
```

```
this is webserver1
```

```
this is webserver2
```

Setting up HAproxy load balancer-Frontend server

In this section, we will set up an HAproxy load balancer for our **web servers**. This HAproxy server will act as a frontend server and accepts incoming requests from clients.

```
host [ haproxy server ]  
pub : 3.111.186.151  
prv : 172.31.37.244  
hostname: Haproxy
```

On the **HAproxy** server , perform the below steps to setup load balancer.

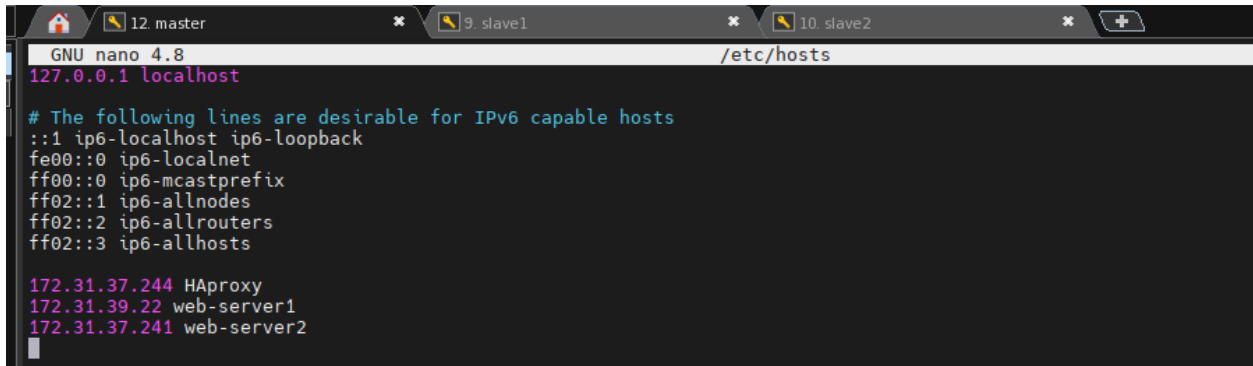
1. Configure hosts file

Edit the `/etc/hosts` file using the below command in Terminal:

```
$ sudo nano /etc/hosts
```

Add the following hostname entries for both **Apache** web servers along with its own hostname:

```
172.31.37.244 HAproxy
172.31.39.22 web-server1
172.31.37.241 web-server2
```



```
GNU nano 4.8 /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

172.31.37.244 HAproxy
172.31.39.22 web-server1
172.31.37.241 web-server2
```

Now save and close the `/etc/hosts` file.

Installing HAproxy load balancer

Now in this step, we will be installing the HAproxy on one of our Ubuntu server (192.168.72.157). To do so, update apt using the following command in Terminal:

```
$ sudo apt-get update -y
```

Then update packages using the below command:

```
$ sudo apt-get upgrade -y
```

Now install HAProxy using the following command in Terminal:

```
$ sudo apt install haproxy -y
```

Configuring HAProxy as a load balancer

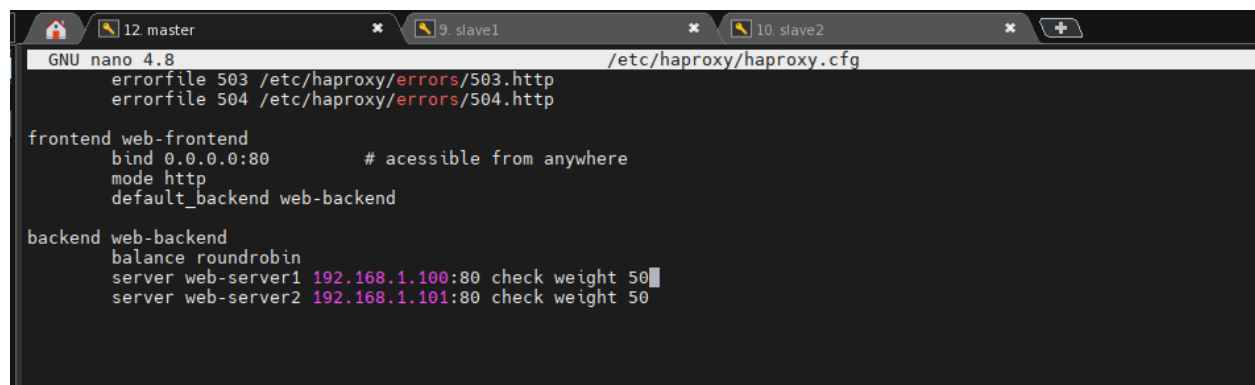
In the following section, we will configure HAProxy as a load balancer. To do so, edit the `/etc/haproxy/haproxy.cfg` file:

```
$ sudo nano /etc/haproxy/haproxy.cfg
```

Append the following lines in the `haproxy.cfg` file replacing the IP addresses with your own IP addresses.

□ The frontend web-frontend in the above configuration lines tells HAProxy to listen to incoming requests on port 80 of 0.0.0.0 and then forward them to backend servers configured under the backend web-backend. While configuring, replace the IP addresses with the relevant IP addresses of your web servers.

```
sudo nano /etc/haproxy/haproxy.cfg
```



```
GNU nano 4.8 /etc/haproxy/haproxy.cfg
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

frontend web-frontend
  bind 0.0.0.0:80      # accessible from anywhere
  mode http
  default_backend web-backend

backend web-backend
  balance roundrobin
  server web-server1 192.168.1.100:80 check weight 50
  server web-server2 192.168.1.101:80 check weight 50
```

```
frontend web-frontend
    bind 0.0.0.0:80      # accessible from anywhere
    mode http
    default_backend web-backend

backend web-backend
    balance roundrobin
    server web-server1 172.31.39.22:80 check weight 50
    server web-server2 172.31.37.241:80 check weight 50
```

The HAProxy configuration file defines a frontend server and a backend server. The frontend server listens on port 80 and forwards all traffic to the backend server. The backend server has two servers defined, web-server1 and web-server2. The IP address and port of each server are defined, as well as the weight of each server. The weight of a server determines how much traffic the server will receive.

The **balance roundrobin** directive tells HAProxy to use the round robin algorithm to load balance traffic to the backend servers. This means that HAProxy will send each new connection to the next server in the list, in a round-robin fashion.

The **check** directive tells HAProxy to check the health of the backend servers. If a server is unhealthy, HAProxy will stop sending traffic to it.

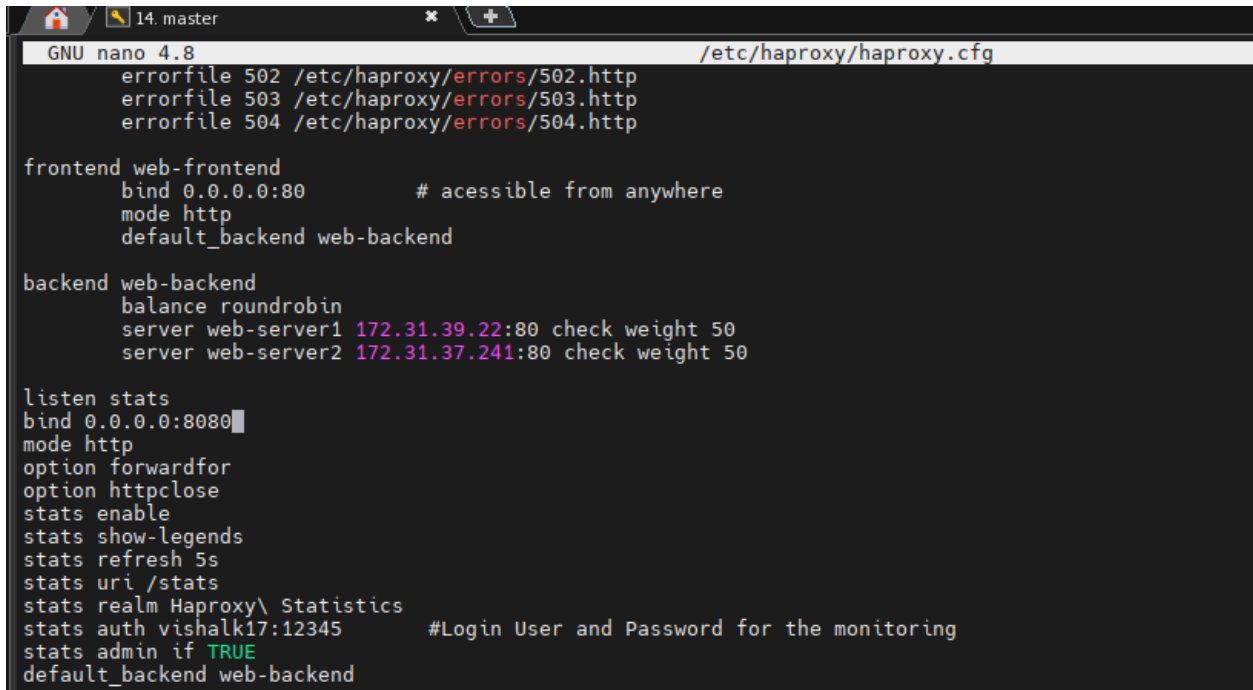
The **weight** directive tells HAProxy how much traffic to send to a backend server. In this case, each server has a weight of 50, which means that they will each receive half of the traffic.

Here is a brief explanation of each line in the HAProxy configuration file:

- **frontend web-frontend** - This line defines the frontend server.
- **bind 0.0.0.0:80** - This line tells HAProxy to listen on port 80 on all network interfaces.
- **mode http** - This line tells HAProxy that the frontend server is an HTTP server.
- **default_backend web-backend** - This line tells HAProxy to forward all traffic to the backend server named **web-backend**.
- **backend web-backend** - This line defines the backend server.
- **balance roundrobin** - This line tells HAProxy to use the round robin algorithm to load balance traffic to the backend servers.
- **server web-server1 172.31.39.22:80 check weight 50** - This line defines the first backend server. The IP address and port of the server are defined, as well as the weight of the server.
- **server web-server2 172.31.37.241:80 check weight 50** - This line defines the second backend server. The IP address and port of the server are defined, as well as the weight of the server.

Configuring HAproxy Monitoring

With HAproxy monitoring, you can view a lot of information including server status, data transferred, uptime, session rate, etc. To configure HAproxy monitoring, append the following lines in the configuration file located at **/etc/haproxy/haproxy.cfg**:



```
GNU nano 4.8 /etc/haproxy/haproxy.cfg
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

frontend web-frontend
    bind 0.0.0.0:80          # accessible from anywhere
    mode http
    default_backend web-backend

backend web-backend
    balance roundrobin
    server web-server1 172.31.39.22:80 check weight 50
    server web-server2 172.31.37.241:80 check weight 50

listen stats
    bind 0.0.0.0:8080
    mode http
    option forwardfor
    option httpclose
    stats enable
    stats show-legends
    stats refresh 5s
    stats uri /stats
    stats realm Haproxy\ Statistics
    stats auth vishalk17:12345      #Login User and Password for the monitoring
    stats admin if TRUE
    default_backend web-backend
```

The above configuration enables the HAproxy “stats” page using the stats directive and secures it with http basic authentication using the username and password defined by the stats auth directive.

Once you are done with the configurations, save and close the haproxy.cfg file.

The HAProxy configuration file defines a stats listener that listens on port 8080. The stats listener allows you to monitor the HAProxy server using a web browser.

The `listen stats` directive tells HAProxy to create a stats listener.

The `bind 0.0.0.0:8080` directive tells HAProxy to listen on port 8080 on all network interfaces.

The `mode http` directive tells HAProxy that the stats listener is an HTTP listener.

The `option forwardfor` directive tells HAProxy to forward the `X-Forwarded-For` header to the backend servers. This header can be used to track the origin of the traffic.

The `option httpclose` directive tells HAProxy to close the TCP connection after the HTTP request has been processed. This can help to improve performance.

The `stats enable` directive tells HAProxy to enable the stats listener.

The `stats show-legends` directive tells HAProxy to show the legends in the stats page.

The `stats refresh 5s` directive tells HAProxy to refresh the stats page every 5 seconds.

The `stats uri /stats` directive tells HAProxy that the stats page is accessible at the URI `/stats`.

The `stats realm Haproxy\ Statistics` directive tells HAProxy that the stats page is protected by a realm named `Haproxy\ Statistics`.

The `stats auth vishalk17:12345` directive tells HAProxy that the stats page can be accessed by users with the username `vishalk17` and the password `12345`.

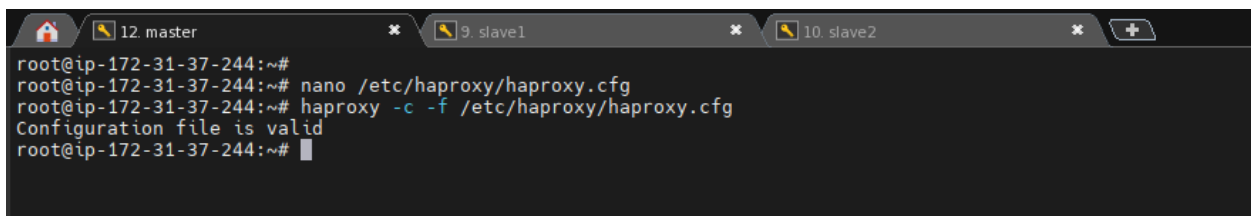
The `stats admin if TRUE` directive tells HAProxy that the stats page is only accessible to administrators.

The `default_backend web-backend` directive tells HAProxy to forward all traffic to the backend server named `web-backend`.

Round robin is a load balancing algorithm that distributes requests to servers in a round-robin fashion. This means that each server will receive an equal number of requests over time one by one.

Once you have created the configuration file, you can start the HAProxy server. To do this, you can use the following command:

```
$ haproxy -c -f /etc/haproxy/haproxy.cfg
```



```
root@ip-172-31-37-244:~#  
root@ip-172-31-37-244:~# nano /etc/haproxy/haproxy.cfg  
root@ip-172-31-37-244:~# haproxy -c -f /etc/haproxy/haproxy.cfg  
Configuration file is valid  
root@ip-172-31-37-244:~#
```

Now to apply the configurations, restart the HAProxy service:

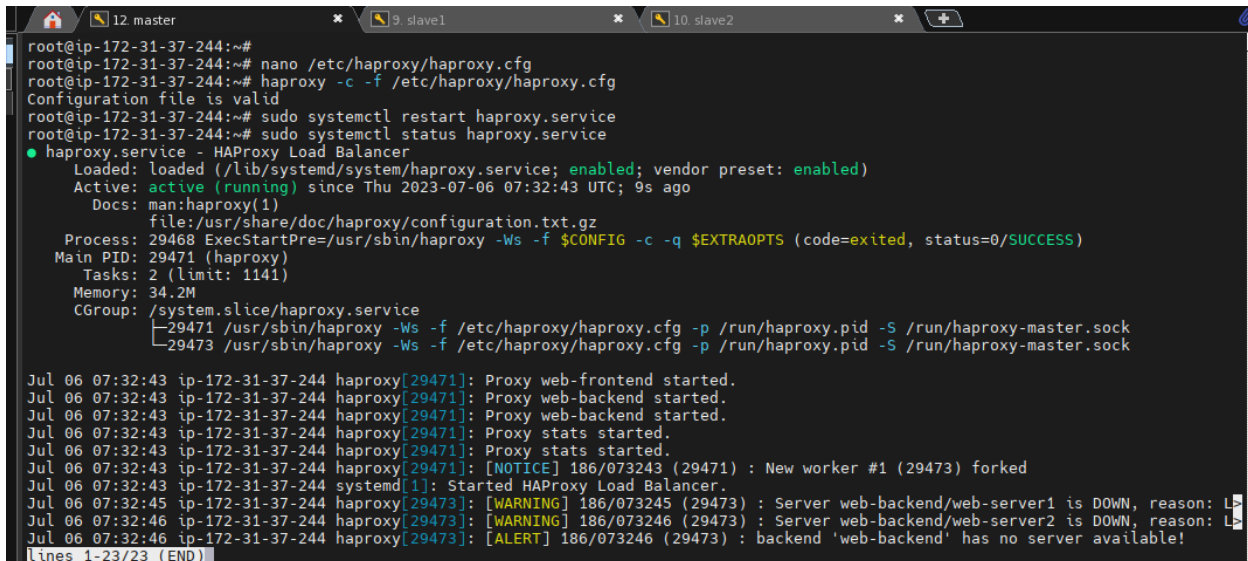
```
$ sudo systemctl restart haproxy.service
```

It will stop and then start the HAProxy service.

To check the status of the HAProxy service, the command would be:

```
$ sudo systemctl status haproxy.service
```

The active (running) status in the following output shows that the HAProxy server is enabled and running fine.



```
root@ip-172-31-37-244:~#
root@ip-172-31-37-244:~# nano /etc/haproxy/haproxy.cfg
root@ip-172-31-37-244:~# haproxy -c -f /etc/haproxy/haproxy.cfg
Configuration file is valid
root@ip-172-31-37-244:~# sudo systemctl restart haproxy.service
root@ip-172-31-37-244:~# sudo systemctl status haproxy.service
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-07-06 07:32:43 UTC; 9s ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
   Process: 29468 ExecStartPre=/usr/sbin/haproxy -Ws -f $CONFIG -c -q $EXTRA_OPTS (code=exited, status=0/SUCCESS)
   Main PID: 29471 (haproxy)
     Tasks: 2 (limit: 1141)
    Memory: 34.2M
   CGroup: /system.slice/haproxy.service
           └─29471 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
             29473 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock

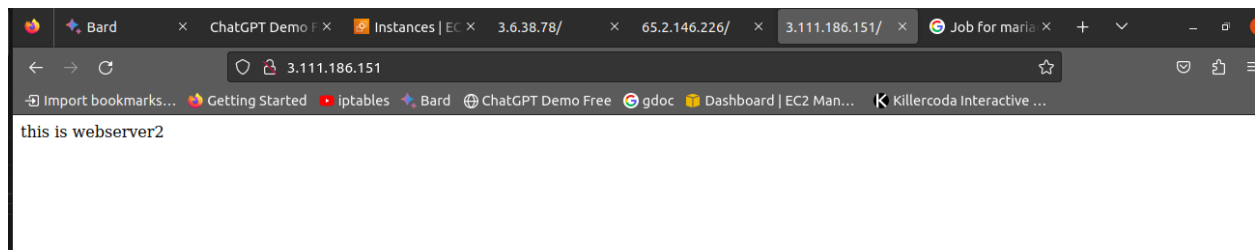
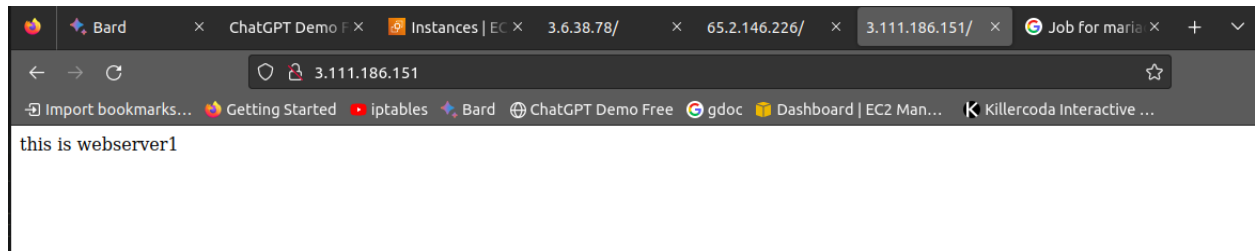
Jul 06 07:32:43 ip-172-31-37-244 haproxy[29471]: Proxy web-frontend started.
Jul 06 07:32:43 ip-172-31-37-244 haproxy[29471]: Proxy web-backend started.
Jul 06 07:32:43 ip-172-31-37-244 haproxy[29471]: Proxy web-backend started.
Jul 06 07:32:43 ip-172-31-37-244 haproxy[29471]: Proxy stats started.
Jul 06 07:32:43 ip-172-31-37-244 haproxy[29471]: Proxy stats started.
Jul 06 07:32:43 ip-172-31-37-244 haproxy[29471]: [NOTICE] 186/073243 (29471) : New worker #1 (29473) forked
Jul 06 07:32:43 ip-172-31-37-244 systemd[1]: Started HAProxy Load Balancer.
Jul 06 07:32:45 ip-172-31-37-244 haproxy[29473]: [WARNING] 186/073245 (29473) : Server web-backend/web-server1 is DOWN, reason: L
Jul 06 07:32:46 ip-172-31-37-244 haproxy[29473]: [WARNING] 186/073246 (29473) : Server web-backend/web-server2 is DOWN, reason: L
Jul 06 07:32:46 ip-172-31-37-244 haproxy[29473]: [ALERT] 186/073246 (29473) : backend 'web-backend' has no server available!
```

Test HA Proxy using a web browser

Now in your HAProxy server, open any web browser and type `http://` followed by the HAProxy server IP address which in our case is `192.168.72.157`.

```
http://public-ip-or-public-dns-of-haproxy-server
```

The HAProxy server will alternatively send the request to both web servers in a round-robin method. You can test this by reloading the webpage a few times.

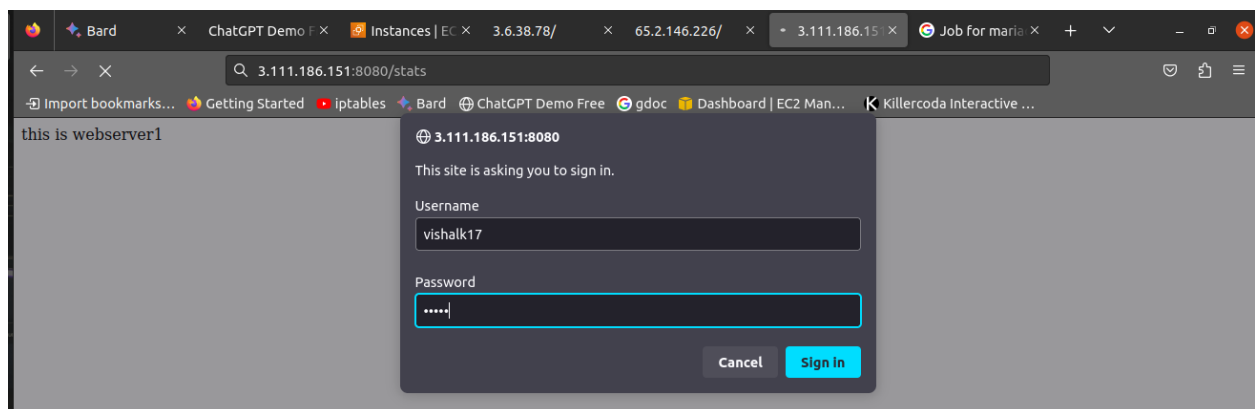


Testing HAproxy Monitoring

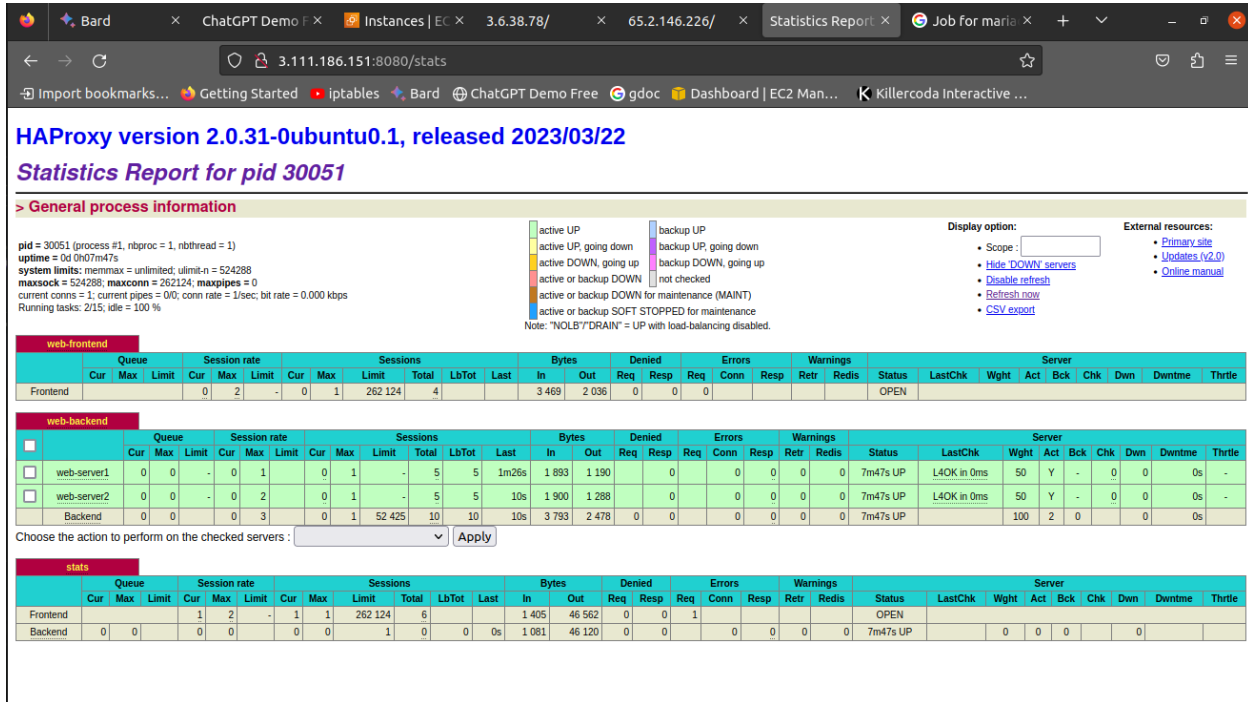
To access the HAproxy monitoring page, type `http://` followed by the IP address/hostname of HAproxy server and port 8080/stats:

`http://public-ip-or-dns-of-ha-proxy:8080/stats`

The following authentication box will appear. Enter the username and password you have configured earlier in the configurations and then press OK.



This is the statistics report for our HAProxy server.



The screenshot shows a web browser displaying the HAProxy statistics report for pid 30051. The browser tabs include Bard, ChatGPT Demo, Instances | EC, 3.6.38.78/, 65.2.146.226/, Statistics Report, and Job for maria. The address bar shows 3.111.186.151:8080/stats. The report title is "HAProxy version 2.0.31-0ubuntu0.1, released 2023/03/22" and "Statistics Report for pid 30051".

> General process information

pid = 30051 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 0h07m47s
system limits: memmax = unlimited; ulimit-n = 524288
maxsock = 524288; maxconn = 262124; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
Running tasks: 2/15; idle = 100 %

Legend:
active UP (green)
active UP, going down (yellow)
active DOWN, going up (orange)
active or backup DOWN (red)
active or backup DOWN for maintenance (MAINT) (purple)
active or backup SOFT STOPPED for maintenance (blue)
backup UP (light blue)
backup UP, going down (light yellow)
backup DOWN, going up (light orange)
not checked (grey)
Note: "NOLB/DRAIN" = UP with load-balancing disabled.

Display option: Scope: []
External resources:
• Primary site
• Updates (v2.0)
• Hide DOWN servers
• Disable refresh
• Refresh now
• CSV export

web-frontend		Queue		Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server												
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend					0	2	-	0	1		262 124	4		3 469	2 036	0	0	0					OPEN									

web-backend		Queue		Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server											
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
<input type="checkbox"/>	web-server1	0	0	-	0	1		0	1		-	5	5	1m26s	1 893	1 190		0	0	0	0	0	7m47s UP	L4OK in 0ms	50	Y	-	0	0	0	-
<input type="checkbox"/>	web-server2	0	0	-	0	2		0	1		-	5	5	10s	1 900	1 288		0	0	0	0	0	7m47s UP	L4OK in 0ms	50	Y	-	0	0	0	-
	Backend	0	0		0	3		0	1		52 425	10	10	10s	3 793	2 478	0	0	0	0	0	0	7m47s UP		100	2	0		0	0	

Choose the action to perform on the checked servers: [] Apply

stats		Queue		Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server												
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend					1	2	-	1	1		262 124	6		1 405	46 562	0	0	1					OPEN									
Backend		0	0		0	0		0	0		1	0	0	0s	1 081	46 120	0	0	0	0	0	0	7m47s UP		0	0	0		0	0		

There you have the installation and configuration of HAProxy load balancer on the Linux system. We have just discussed the basic setup and configuration of HAProxy as a load balancer for Apache web servers. We also looked at some commands for managing the HAProxy server. In the end, we tested the load balancing through the browser. For more information, visit HAProxy [official documentation](#)