Install and configure HTTP Load Balancer

Esprit 2024-2025

Goal:

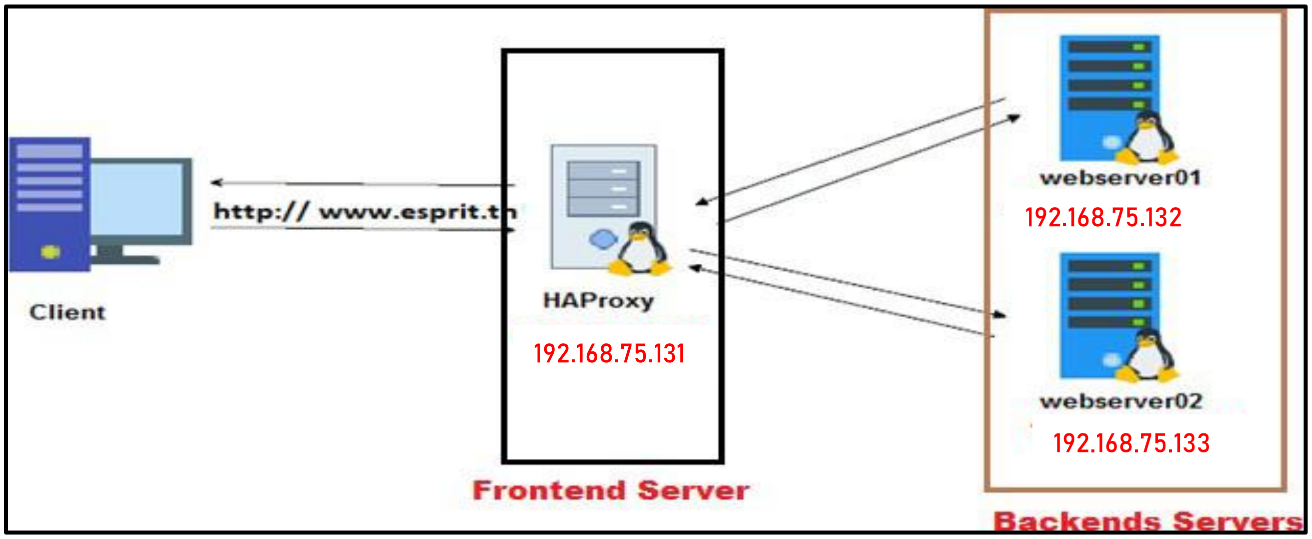
HaProxy, short for High Availability Proxy, is a free and open-source HTTP load balancer and reverse-proxy solution that is widely used to provide high availability to web applications and guarantee maximum possible uptime.

It is a high-performance application that injects performance improvements to your web apps by distributing traffic across multiple endpoints. This way, it ensures that no webserver is overloaded with incoming HTTP requests since the workload is equitably distributed across several nodes.

This document will guide you through deploying it for both simple web applications and large, complex web sites.

This example based on the environment like follows.

Don't forget to make HAProxy Server, webserver01 and webserver02 through the same **LANsegment**



Pre-requisites

To demonstrate HAProxy in action, you need to have at least three Linux systems. One will act as the HAProxy load balancer, while the rest will act as web servers.

(You can use a clone of your virtual machine to create the two **web servers**, and use an **alias** interface with the **ens33** interface of your virtual machine.)

- HAProxy Server @ip 192.168.75.131/24
- Webserver 1(@ip 192.168.75.132/24)
- webserver 2(@ip 192.168.75.133/24)
- Also you should have a website in each one of your backends servers. (**review TP1**)

The HAProxy server will be used as a client for the tests.

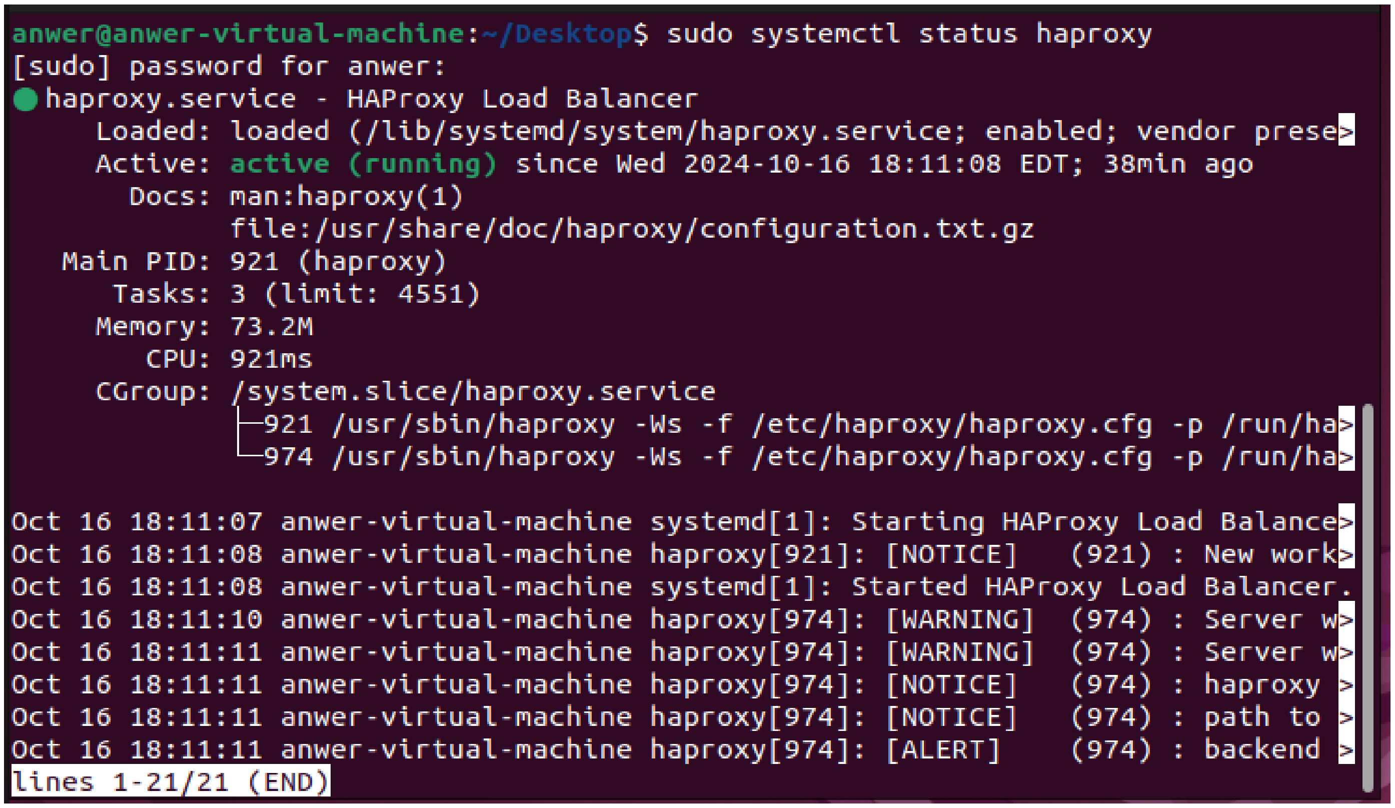
Step 1 – Install HAProxy

HAProxy package is available under default repository of Ubuntu. Use the following command to install HAProxy on your system:

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

Upon installation, the HAProxy service starts by default and listens to TCP port 80. To verify HAProxy is running, run the command

```
$ sudo systemctl status haproxy
```



```
anwer@anwer-virtual-machine:~/Desktop$ sudo systemctl status haproxy
[sudo] password for anwer:
● haproxy.service - HAProxy Load Balancer
     Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor prese>
     Active: active (running) since Wed 2024-10-16 18:11:08 EDT; 38min ago
       Docs: man:haproxy(1)
             file:/usr/share/doc/haproxy/configuration.txt.gz
   Main PID: 921 (haproxy)
      Tasks: 3 (limit: 4551)
     Memory: 73.2M
        CPU: 921ms
     CGroup: /system.slice/haproxy.service
             ├─921 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/ha>
             └─974 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/ha>

Oct 16 18:11:07 anwer-virtual-machine systemd[1]: Starting HAProxy Load Balance>
Oct 16 18:11:08 anwer-virtual-machine haproxy[921]: [NOTICE]   (921) : New work>
Oct 16 18:11:08 anwer-virtual-machine systemd[1]: Started HAProxy Load Balancer.
Oct 16 18:11:10 anwer-virtual-machine haproxy[974]: [WARNING]  (974) : Server w>
Oct 16 18:11:11 anwer-virtual-machine haproxy[974]: [WARNING]  (974) : Server w>
Oct 16 18:11:11 anwer-virtual-machine haproxy[974]: [NOTICE]   (974) : haproxy >
Oct 16 18:11:11 anwer-virtual-machine haproxy[974]: [NOTICE]   (974) : path to >
Oct 16 18:11:11 anwer-virtual-machine haproxy[974]: [ALERT]    (974) : backend >
lines 1-21/21 (END)
```

Step 2 – Configure HAProxy

The next step is to configure HAProxy to distribute traffic evenly between two web servers. The configuration file for haproxy is /etc/haproxy/haproxy.cfg.

Before making any changes to the file, first, make a backup copy.

```
$ sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bk
```

Then open the file using your preferred text editor.

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

Haproxy configuration file is made up of the following sections:

- global: This is the first section that you see at the very top. It contains system-wide settings that handle performance tuning and security.
- defaults: As the name suggests, this section contains settings that should work well without additional customization. These settings include timeout and error reporting configurations.
- frontend and backend: These are the settings that define the frontend and backend settings. For the frontend, we will define the HAProxy server as the front end which will distribute requests to the backend servers which are the webservers. We will also set HAProxy to use round robbin load balancing criteria for distributing traffic.
- listen: This is an optional setting that enables you to enable monitoring of HAProxy statistics.

Now define the frontend and backend settings:

```
frontend linuxtechi
   bind 192.168.75.131:80
   stats uri /haproxy?stats
   default_backend web-servers
   option forwardfor

backend web-servers
    balance roundrobin
    server web1 192.168.75.132:80
    server web2 192.168.75.133:80
```

In the **frontend Site Configuration** section, any request made to the IP address 192.168.75.131 on port 80 will be redirected to port 80 of the servers 192.168.75.132 or 192.168.75.133.

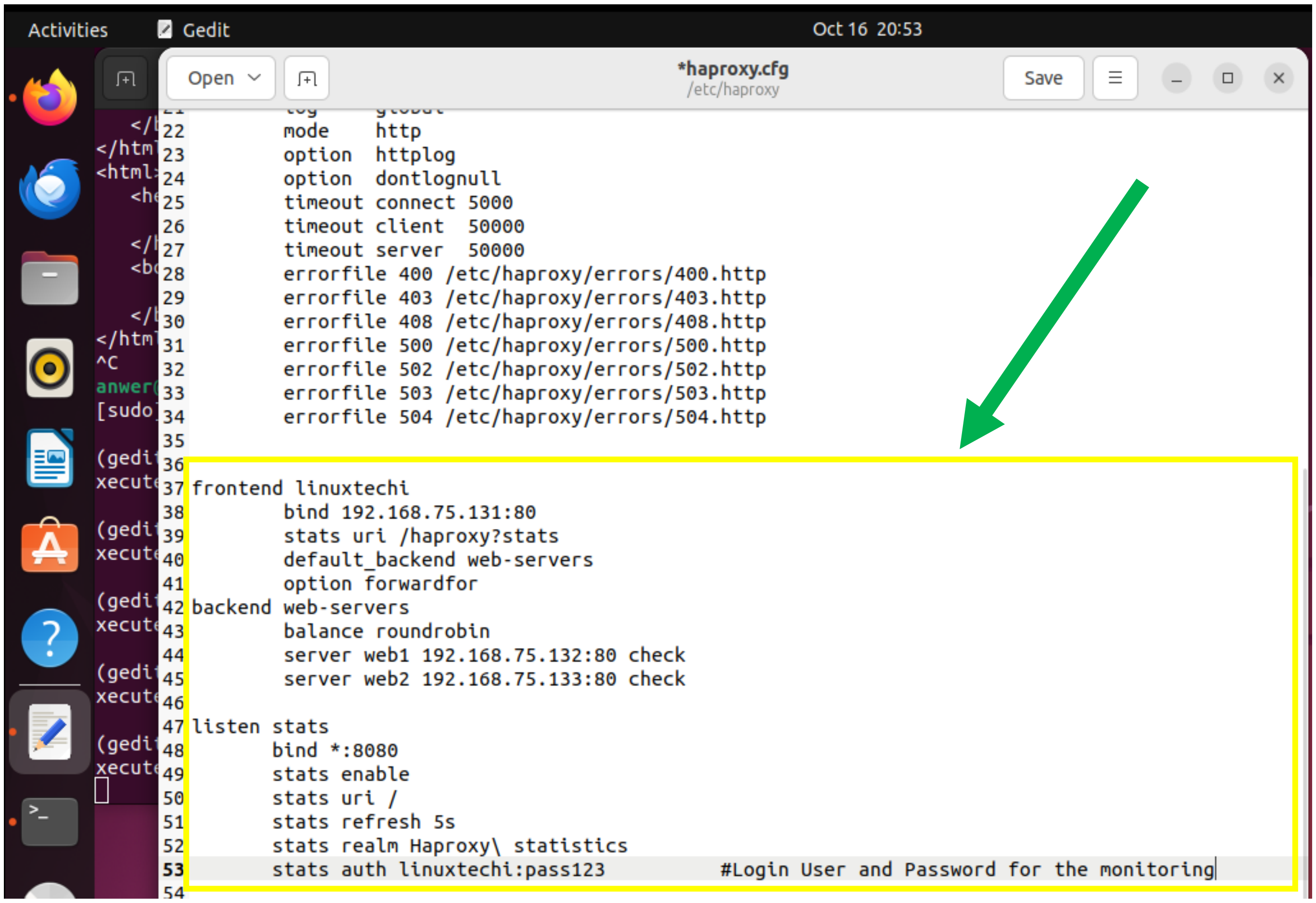
We have configured both the HAProxy server and the web server nodes to listen on port 80. **Make sure to replace the IP addresses for HAProxy and the web servers with your own setup.**

In order to enable viewing the HAProxy statistics from a browser, add the following 'listen' section.

```
listen stats

    bind *:8080
    stats enable
    stats uri /
    stats refresh 5s
    stats realm Haproxy\ Statistics
    stats auth linuxtechi:Pass123      #Login User and Password for the monitoring
```

The stats auth directive specifies the username and password for the login user for viewing statistics on the browser.



```
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http


frontend linuxtechi
        bind 192.168.75.131:80
        stats uri /haproxy?stats
        default_backend web-servers
        option forwardfor
backend web-servers
        balance roundrobin
        server web1 192.168.75.132:80 check
        server web2 192.168.75.133:80 check

listen stats
       bind *:8080
       stats enable
       stats uri /
       stats refresh 5s
       stats realm Haproxy\ statistics
       stats auth linuxtechi:pass123          #Login User and Password for the monitoring
```

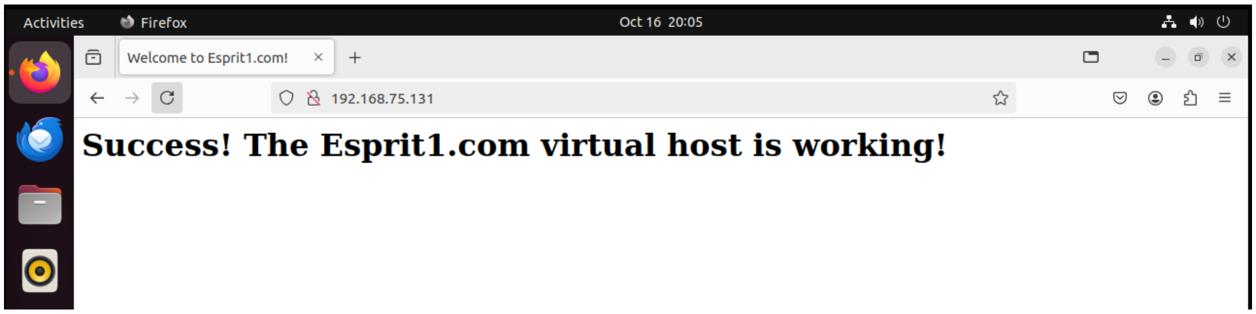
Now save all the changes and exit the configuration file. To reload the new settings, restart haproxy service.

```
$ sudo systemctl restart haproxy
```

Step 3 - Test HAProxy Load Balancing

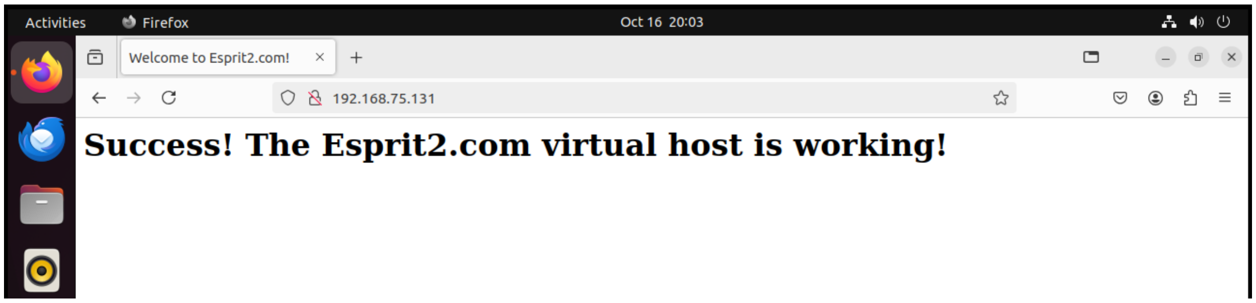
Up until this point, we have successfully configured our HAProxy and both of the back-end web servers. To test if your configuration is working as expected, browse the IP address of the HAProxy server http://192.168.75.131

When you browse for the first time, you should see the web page for the first web server



Success! The Esprit1.com virtual host is working!

Upon refreshing, you should the webpage for the second web server



Success! The Esprit2.com virtual host is working!

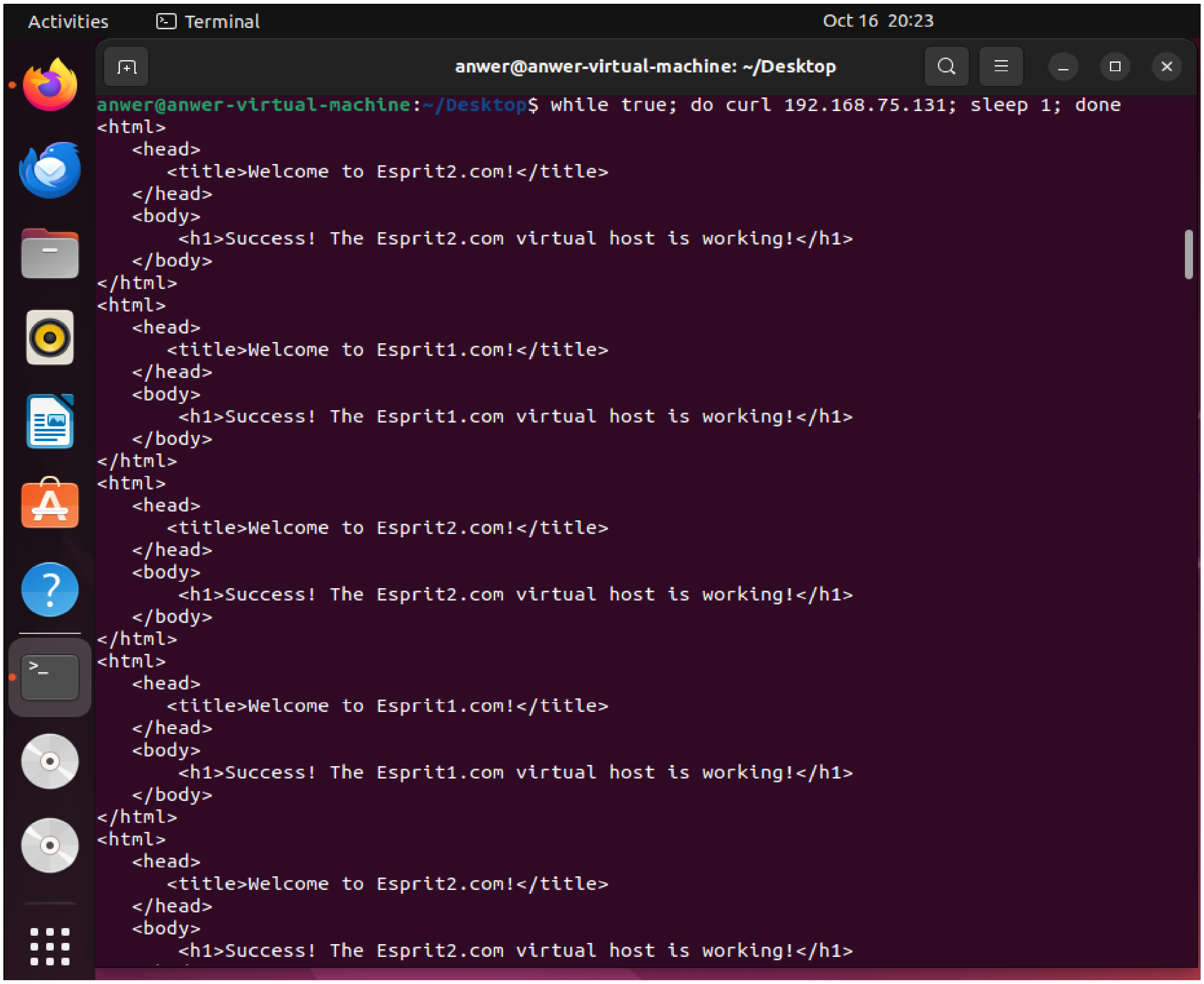
This shows that the HAProxy server is performing its load balancing job spectacularly by distributing incoming web traffic across the two web servers in a Round Robbin algorithm.

Moreover, you can use the following do-while loop with the curl command:

```
$ while true; do curl 192.168.75.131; sleep 1; done
```

This command continuously executes HTTP requests to the IP address 192.168.75.131 every 1 second using the curl utility. The loop will never stop on its own and will continue running until you manually interrupt the process (for example, by pressing Ctrl + C). It is often used to test the availability or responsiveness of a server at regular intervals.

NB: After executing this command, the system may ask you to install an additional tool, such as the 'curl' utility. It is therefore recommended to ensure that all necessary dependencies are installed to guarantee the correct execution of the command.

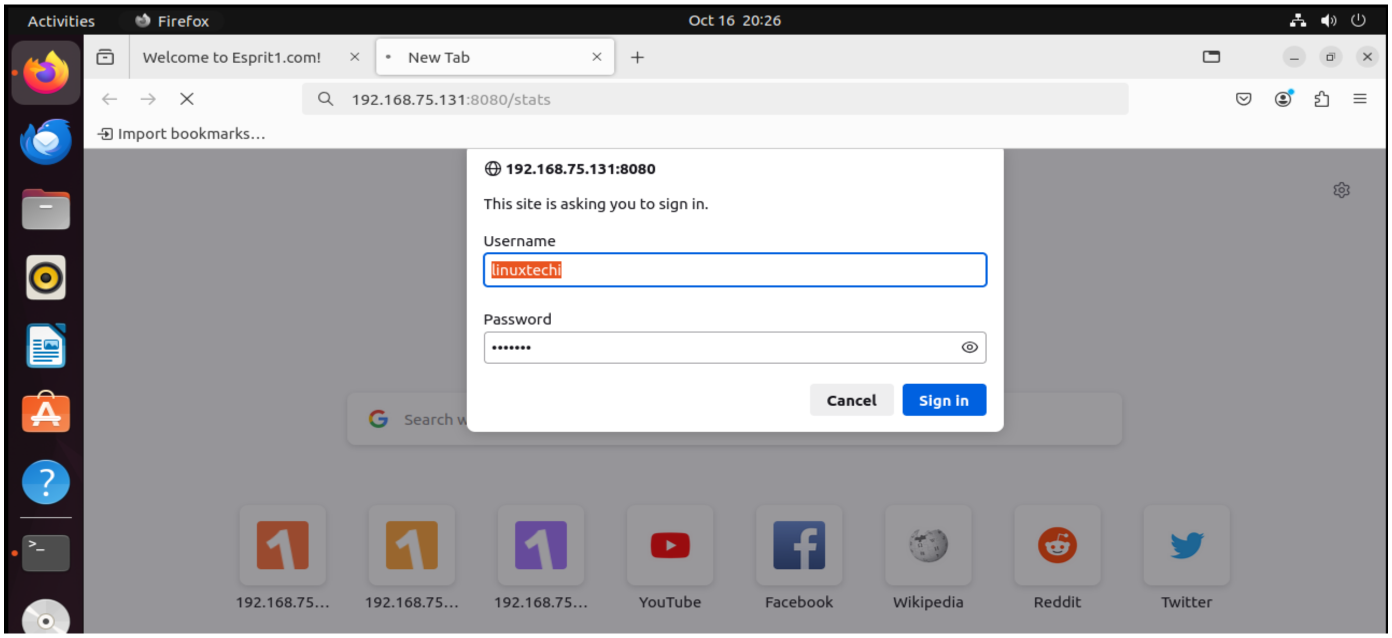


```
anwer@anwer-virtual-machine:~/Desktop$ while true; do curl 192.168.75.131; sleep 1; done
<html>
   <head>
      <title>Welcome to Esprit2.com!</title>
   </head>
   <body>
       <h1>Success! The Esprit2.com virtual host is working!</h1>
   </body>
</html>
<html>
   <head>
      <title>Welcome to Esprit1.com!</title>
   </head>
   <body>
       <h1>Success! The Esprit1.com virtual host is working!</h1>
   </body>
</html>
<html>
   <head>
      <title>Welcome to Esprit2.com!</title>
   </head>
   <body>
       <h1>Success! The Esprit2.com virtual host is working!</h1>
   </body>
</html>
<html>
   <head>
      <title>Welcome to Esprit1.com!</title>
   </head>
   <body>
       <h1>Success! The Esprit1.com virtual host is working!</h1>
   </body>
</html>
<html>
   <head>
      <title>Welcome to Esprit2.com!</title>
   </head>
   <body>
       <h1>Success! The Esprit2.com virtual host is working!</h1>
```

To view monitoring statistics, browse the following URL:

http://192.168.75.131:8080/stats

You will be required to authenticate, so provide your details as specified in Step 2.



192.168.75.131:8080

This site is asking you to sign in.

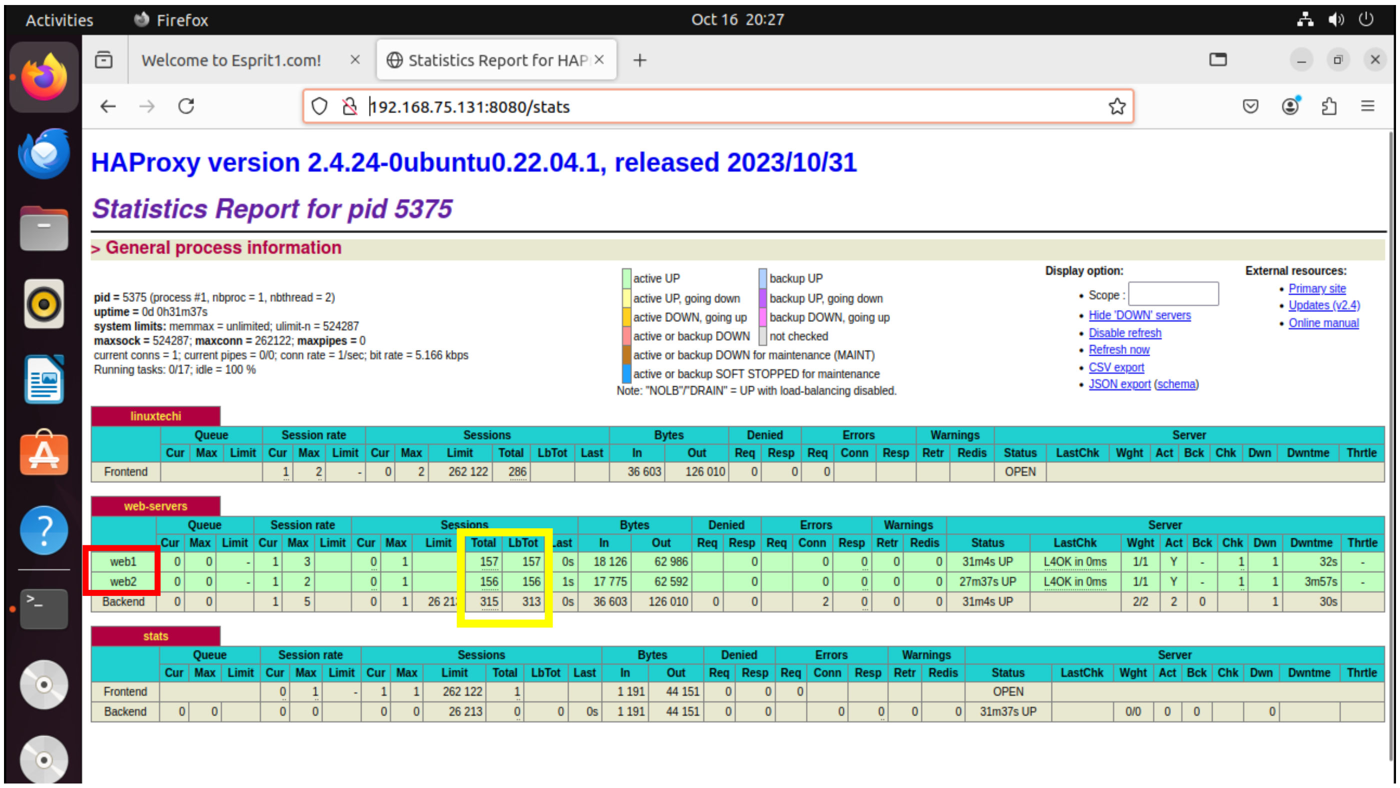
Username: linuxtechi

Password: •••••••

Cancel | Sign in

You will see the following page with statistics on the performance of the HAProxy server.

You can see both web servers in the HAProxy Statistics interface (**framed in red**) and the fields showing the loads distributed across the two servers (**framed in yellow**). Every 5 seconds, the HAProxy Stats interface refreshes, and the loads change according to the incoming traffic.



HAProxy version 2.4.24-0ubuntu0.22.04.1, released 2023/10/31

Statistics Report for pid 5375

> General process information

pid = 5375 (process #1, nbproc = 1, nbthread = 2)
uptime = 0d 0h31m37s
system limits: memmax = unlimited; ulimit-n = 524287
maxsock = 524287; **maxconn =** 262122; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 5.166 kbps
Running tasks: 0/17; idle = 100 %

🟩	active UP	▨	backup UP
🟨	active UP, going down	🟪	backup UP, going down
🟧	active DOWN, going up	🟪	backup DOWN, going up
🟥	active or backup DOWN	⬜	not checked
🟫	active or backup DOWN for maintenance (MAINT)		
🟦	active or backup SOFT STOPPED for maintenance		

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export
- JSON export (schema)

External resources:
- Primary site
- Updates (v2.4)
- Online manual

linuxtechi

	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	-	0	2	262 122	286			36 603	126 010	0	0	0					OPEN								

web-servers

	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
web1	0	0	-	1	3		0	1		157	157	0s	18 126	62 986		0		0	0	0	0	31m4s UP	L4OK in 0ms	1/1	Y	-	1	1	32s	-
web2	0	0	-	1	2		0	1		156	156	1s	17 775	62 592		0		0	0	0	0	27m37s UP	L4OK in 0ms	1/1	Y	-	1	1	3m57s	-
Backend	0	0		1	5		0	1	26 21	315	313	0s	36 603	126 010	0	0		2	0	0	0	31m4s UP		2/2	2	0		1	30s	

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				0	1	-	1	1	262 122	1			1 191	44 151	0	0	0					OPEN								
Backend	0	0		0	0		0	0	26 213	0	0	0s	1 191	44 151	0	0		0	0	0	0	31m37s UP		0/0	0	0		0		

There you have it! We have successfully installed HAProxy on Ubuntu 22.04 and configured it to serve requests across two web servers using the Round Robbin load balancing criteria.