

How to Install HAProxy on Ubuntu 20.04

Learn how to install and configure HAProxy on Ubuntu 20.04 with our step-by-step guide. Set up load balancing for your web servers quickly and efficiently.

Contents

01	Introduction	3
02	Prerequisites	3
03	1. Install the HAProxy Server on the Main Server	4
04	2. Configure the HAProxy Server	5
05	3. Configure Apache Listening Port On the Backend Servers	8
06	4. Create Web Content on the Backend Servers	9
07	5. Test the HAProxy Load Balancing Algorithm	11
08	Conclusion	12

Introduction

HAProxy stands for **H**igh **A**vailability **P**roxy. It's an open-source load balancer that you can use to distribute HTTP traffic to multiple backend applications, websites, or databases to create a highly available system. It is the fastest and most superior load balancer that offers massive scalability in any environment, making it the number one choice for high-profile websites like Twitter, GitHub, and Amazon Web Service.

In this age of an internet-connected world, your application might be serving millions of users around the globe (For instance, a social media application). In such a mission-critical system, any downtime may lead to financial loss and an unpleasant customer experience. To ensure that your application works even when some of its components fail, you should create multiple instances of your computing infrastructure and put a customer-facing load balancer to route traffic to the clustered environment.

HAProxy allows your system to tolerate interruptions with no downtime to users. To use the technology, you should design your system with **redundancy** in mind. That is, you should run multiple instances of computer components that are very likely to fail or require periodic maintenance. Also, HAProxy **failover mechanism** uses different performance metrics to **monitor** the availability and health of the redundant servers working as a group to switch traffic only to active components. Therefore, HAProxy meets all the basic elements of high availability. That is **redundancy**, **failover**, and **monitoring**.

In this guide, you'll install and configure the HAProxy load balancer on Ubuntu 20.04 server to distribute web traffic to two different servers.

Prerequisites

To follow along with this HAProxy tutorial, you require the following:

- A set of **3** [Ubuntu 20.04 servers](#) configured with private networking. Please refer to the guide on [How to Create a Private Network](#) to set up private

networking for your Vultr servers. Your servers should be in the same location.

- A sudo user.

This guide uses the following public/private IP addresses and hostnames for the servers. Make sure to change the values accordingly.

Hostname: **main-server**

- Public IP: **192.0.2.10**
- Private IP: **10.0.0.10**

Hostname: **server-1**

- Public IP: **192.0.2.11**
- Private IP: **10.0.0.11**

Hostname: **server-2**

- Public IP: **192.0.2.12**
- Private IP: **10.0.0.12**

Ensure [Apache webserver](#) is only installed on **server-1** and **server-2**. Don't install Apache on the **main-server** because there will be a port conflict. This is because Apache listens on the same port (**80**) as HAProxy.

1. Install the HAProxy Server on the Main Server

In this step, you'll pull the HAProxy package from the official Ubuntu software repository. SSH to your server and update the package information index by running the command below.

```
$ sudo apt update
```

Next, issue the following command to install the `haproxy` package.

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

Check the status of the HAProxy server.

```
$ sudo systemctl status haproxy
```

Ensure the HAProxy service is running by confirming the following output.

```
...  
Active: active (running)  
...
```

To check your HAProxy version, run the following command.

```
$ haproxy -v
```

Ensure you get the version number displayed as shown below.

```
HA-Proxy version 2.0.13-2ubuntu0.2 2021/08/16 - https://haproxy.org/
```

After installing the HAProxy package, you'll configure it next to distribute traffic to the rest of your servers.

2. Configure the HAProxy Server

HAProxy maintains a configuration file at the following location.

```
/etc/haproxy/haproxy.cfg
```

First, back up the file using the Linux `cp` command to ensure you can revert to the default settings if you make any mistakes.

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

Next, use the `nano` text editor to open the file for editing purposes.

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

Once opened, the HAProxy configuration file is divided into different sections, which play a vital role, as explained below.

- **global:** At the top, HAProxy contains system-wide settings that mainly deal with security and performance tuning. For now, don't change these values. Here are some of the **global** settings.

```
global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd
listeners
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

    # Default SSL material locations
    ca-base /etc/ssl/certs
    crt-base /etc/ssl/private
    ...
```

- **defaults:** The **defaults** holds some settings that you would probably use without further customizations, including error reporting and timeout configurations. The settings may be similar to the following output. Don't touch these settings; you'll run HAProxy with the default values for now.

```
defaults
    log      global
    mode     http
    ...
    timeout server 50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    ...
    errorfile 408 /etc/haproxy/errors/408.http
```

- **frontend** and **backend**: In addition to the **global** and **defaults** settings HAProxy allows you to define some **frontend** and **backend** settings. In this guide, you're using HAProxy as a reverse proxy in front of two backend servers. Therefore, you should define both the **frontend** and **backend** settings as shown below.

In the frontend, instruct HAProxy to bind to port **80** (`bind *:80`) and forward traffic to a **backend** section which you've named `web_servers`. In the backend section, set HAProxy to use the `roundrobin` algorithm to select every backend server cyclically without any preference. You may also use the `leastconn` algorithm to prioritize the server with the least active connections. Next, define the IP address and ports of the servers where you want the traffic to be distributed. Remember to replace the IP addresses `10.0.0.11` and `10.0.0.12` with the appropriate **private IP addresses** of your nominated backend servers. The `check` parameter validates the health of the server before routing any clients to it.

```
frontend example_front_end
    bind *:80
    option forwardfor
    default_backend web_servers

backend web_servers
    balance roundrobin
    server server-1 10.0.0.11:8080 check
    server server-2 10.0.0.12:8080 check
```

- **listen**: Optionally, define some statistical settings on a new **listen** section as shown below. The settings will help you log in to the HAProxy server on a web browser and view the performance of your servers. Replace `ha_proxy_admin` with an appropriate username. Change `EXAMPLE_PASSWORD` with a strong value.

```
listen stats
    bind :32600
    stats enable
    stats uri /
```

```
stats hide-version
stats auth ha_proxy_admin:EXAMPLE_PASSWORD
```

Save and close the `/etc/haproxy/haproxy.cfg` when you're through with editing. In the above settings, you've instructed HAProxy to listen on port **80**. In case you've Apache web server installed on the **main-server**, it will still listen on the same port(**80**), and there will be a conflict, and HaProxy won't be able to load. Therefore, stop the Apache webserver using the following command in case you've it.

```
$ sudo systemctl stop apache2
```

You can now gracefully restart HAProxy to load the new configuration settings using the following command.

```
$ sudo systemctl restart haproxy
```

The HAProxy server is now ready to listen for HTTP traffic and route it to the appropriate backend servers. In the next step, you'll configure the Apache webserver on the backend servers.

3. Configure Apache Listening Port On the Backend Servers

By default, the Apache web server listens for HTTP traffic on port **80**. In the previous section, you've configured your HAProxy server to direct traffic to port **8080** on the backend servers. In this step, you'll configure the same on these two servers.

Open the `/etc/apache2/ports.conf` file on both `server-1` and `server-2` on different terminals using `nano` text editor.

```
$ sudo nano /etc/apache2/ports.conf
```

Locate the following `Listen` directive.

```
Listen 80
```

Then, change the port from `80` to `8080` as shown below on both backend servers.

```
Listen 8080
```

Still, on both the backend servers, open the virtual host file below.

```
$ sudo nano /etc/apache2/sites-available/000-default.conf
```

Locate the line below.

```
<VirtualHost *:80>  
...
```

Change the value of `*:80` to `*:8080`

```
<VirtualHost *:8080>  
...
```

Restart Apache on both servers to load the new changes.

```
$ sudo systemctl restart apache2
```

Apache webserver should now listen for incoming traffic on port `8080` on both the backend servers as soon as it's forwarded by the HAProxy server. In the next step, you'll create some web content on the backend servers.

4. Create Web Content on the Backend Servers

Clients will send HTTP requests to the main server where you've installed HAProxy. Then, HAProxy will route the traffic in a balanced way to the backend

servers. Therefore, you should create some web content on the public root directories of **server-1** and **server-2**.

First, delete the default `/var/www/html/index.html` files on both the backend servers.

- **server-1:**

```
$ sudo rm /var/www/html/index.html
```

- **server-2:**

```
$ sudo rm /var/www/html/index.html
```

Next, open a new HTML file on **server-1**.

```
$ sudo nano /var/www/html/index.html
```

Then, enter the content below into the `/var/www/html/index.html` on **server-1**.

```
<html>
  <head>
    <title>Backend Server 1</title>
  </head>
  <body>
    <h1>Server 1 is working.</h1>
  </body>
</html>
```

Save and close the file. Next, create a new `/var/www/html/index.html` on **server-2**.

```
$ sudo nano /var/www/html/index.html
```

Then enter the information below into the `/var/www/html/index.html` file on **server-2**.

```
<html>
  <head>
    <title>Backend Server 2</title>
```

```
</head>
<body>
  <h1>Server 2 is working.</h1>
</body>
</html>
```

Save and close the file. You've now created different web content in the backend servers. In the next step, you'll test whether HAProxy can route and distribute traffic to your backend servers.

5. Test the HAProxy Load Balancing Algorithm

In your web browser, enter the public IP address or domain name of the **main-server** where you installed HAProxy.

- <http://192.0.2.10>

You should receive the response below when you visit the page for the first time showing you have connected to backend **server-1**.

Server 1 is working.

Refresh the page, and this time around, you should view the content of backend **server-2**.

Server 2 is working.

To view general statistical information about your frontend and backend servers, visit the HAProxy's server public IP address on port `32600`.

- <http://192.0.2.10:32600>

Enter the login details that you defined on the stats sections in the HAProxy configuration file. For instance `username:ha_proxy_admin, password:EXAMPLE_PASSWORD` to log in.

```
...
listen stats
...
stats auth ha_proxy_admin:EXAMPLE_PASSWORD
...
```

You should see a dashboard with a lot of HAProxy information.

HAProxy

Statistics Report for pid 1715

> General process information

pid = 1715 (process #1, nproc = 1, nbthread = 1)
 uptime = 0d 0h 1m 52s
 system limits: memmax = unlimited; ulimit-n = 1024
 maxsock = 1024; maxconn = 492; maxpipes = 0
 current conns = 1; current pipes = 0; conn rate = 1/sec; bit rate = 2.991 kbps
 Running tasks: 1/15; 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: External resources:
 • [Evince.sde](#)
 • [Upgrades \(v2.0\)](#)
 • [Hide DOWN servers](#)
 • [Refresh now](#)
 • [CSV export](#)
 • [Online manual](#)

example.front_end		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	0	1	0	0	-	0	0	1	492	1		0	221	0	0	0	0	0	0	0	0	1	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	
server-1	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	1m8s UP	L40K in 1ms	1	Y	-	0	0	0s	-
server-2	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	1m8s UP	L40K in 0ms	1	Y	-	0	0	0s	-
Backend	0	0	-	0	0	-	0	0	50	0	0	?	0	0	0	0	0	0	0	0	0	0	1m8s UP		2	2	0	0	0	0s	-

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	2	-	492	3					418	504	0	0	1	0	0	0	0	0	OPEN								
Backend	0	0	-	0	0	-	50	0					418	504	0	0	0	0	0	0	0	0	1m8s UP		0	0	0	0	0	0s	-

Your HAProxy server is working as expected, and indeed, you're redistributing the load from the main server to the backend servers as you had expected. In this guide, you've created two different web pages with varying content to prove the load-distribution concept. In a production environment, the contents on the redundant servers should be similar, or if you're fetching data from a database, the backend servers should display the same data. For instance, the backends should connect to a MySQL group replicated database. This helps your app to tolerate downtimes in case some of the backend servers fail.

Conclusion

In this tutorial, you've installed HAProxy on Ubuntu 20.04 and successfully distributed HTTP traffic to two different backend servers. You can extend the

logic in this guide and even set up more redundant servers depending on your application load. HAProxy allows you to distribute traffic on your clustered environment to avoid overwhelming a single server when thousands of clients are connected to your application.



VULTR

