

How to Deploy HAProxy on Ubuntu 22.04

Learn how to install and configure HAProxy on Ubuntu 22.04 for load balancing and high availability. Step-by-step guide with configuration examples.

Contents

01	Introduction	3
02	Prerequisites	3
03	Sample HAProxy LoadBalancing Topology	4
04	Install HAProxy on Ubuntu 22.04	5
05	Configure the HAProxy Server	6
06	Configure the Backend Servers	11
07	Access the HAProxy Accelerated Web Application	15
08	Conclusion	17

Introduction

HAProxy (High Availability Proxy) is an open-source proxying application that can serve as a reverse or forward proxy for TCP and HTTP-based applications. You can install HAProxy on Ubuntu 22.04 to efficiently distribute traffic across multiple backend servers, helping to reduce load and improve application performance.

HAProxy acts as the main gateway that handles all external network requests to your backend servers that run web servers, databases, or file management applications. When a client makes a connection request, HAProxy examines the incoming request and forwards it to a backend server based on your routing rules. The backend server processes the request and serves a response to the requesting client. Throughout the process, HAProxy acts as the origin server while distributing the traffic load to the destination servers depending on your configurations.

This article explains how to deploy HAProxy on an Ubuntu 22.04 server and distribute traffic to multiple servers running similar services in a single Vultr VPC 2.0 network to form a cluster. You will configure HAProxy as a load balancer to forward all external requests to backend servers to improve your web application's performance.

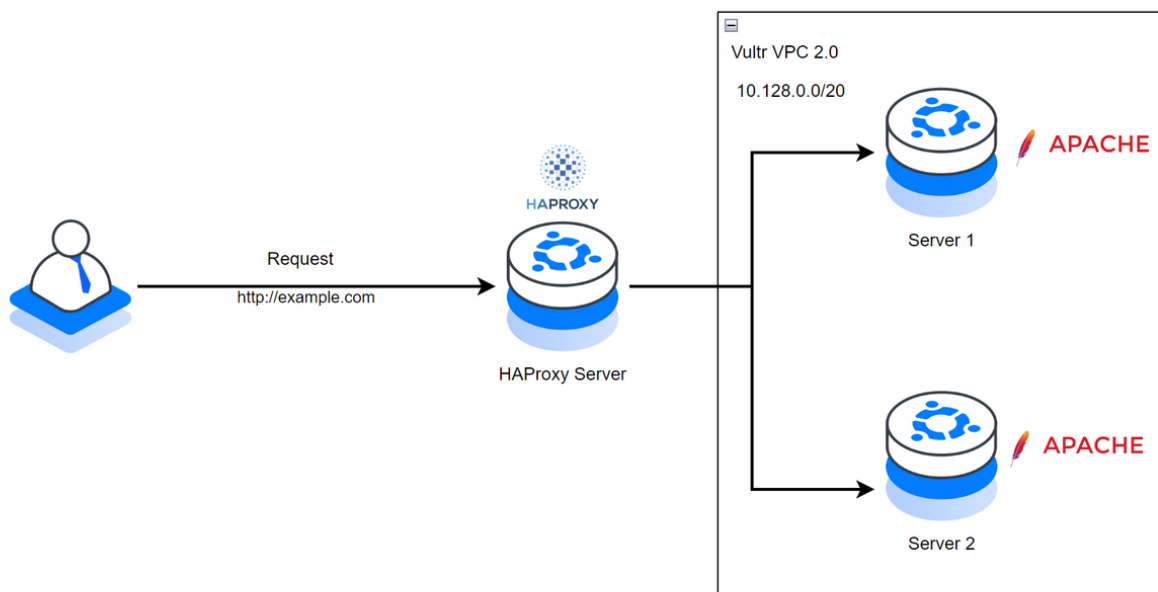
Prerequisites

In a single Vultr location:

- Deploy a [Ubuntu 22.04](#) instance to use as the HAProxy server.
- Deploy at least `2` instances running Ubuntu 22.04 as the backend servers.
- Attach all servers to the same [Vultr VPC 2.0 network](#).
- Set Up a [new domain A record that points to your HAProxy server IP address](#). For example, `haproxy.example.com`.

Sample HAProxy LoadBalancing Topology

This article uses the following example topology to configure HAProxy as a load balancer that distributes traffic to multiple backend servers running the Apache web server application. Depending on your application servers, HAProxy forwards connection requests to a backend server depending on its health status and your load balancing method.



Below are the respective IP address details for each server in the Vultr VPC 2.0 network.

- HA-Proxy Server:
 - **Public Hostname:** `haproxy.example.com`
 - **VPC 2.0 IP:** `10.128.0.2`
- Server 1:
 - **VPC 2.0 IP:** `10.128.0.3`
- Server 2:
 - **VPC 2.0 IP:** `10.128.0.4`

Install HAProxy on Ubuntu 22.04

1. Access the HAProxy server using SSH.

CONSOLE

```
$ ssh root@SERVER-IP
```

2. Create a new non-root user with sudo privileges. For example, `haproxyadmin`.

CONSOLE

```
# adduser haproxyadmin && adduser haproxyadmin sudo
```

3. Switch to the new sudo user account.

CONSOLE

```
# su - haproxyadmin
```

4. Update the server package index.

CONSOLE

```
$ sudo apt update
```

5. Install HAProxy.

CONSOLE

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

6. The latest HAProxy version may not be available in the default APT repositories. Run the following command to install a specific version using the `vybernet` PPA repository on your server.

CONSOLE

```
$ sudo add-apt-repository ppa:vbernat/haproxy-2.8 -y
```

7. Enable HAProxy to start at boot time.

CONSOLE

```
$ sudo systemctl enable haproxy
```

8. View the HAProxy system service status to verify that the application is installed on your server.

CONSOLE

```
$ sudo systemctl status haproxy
```

Output:

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

Configure the HAProxy Server

The HAProxy main configuration file `/etc/haproxy/haproxy.cfg` controls how the application runs and listens for incoming connections on the server. The configuration file includes the following sections by default:

- `global`: Contains settings that define how HAProxy runs on the server. Available options include logging, user/group, system service mode, and SSL configurations.
- `defaults`: Includes the default HAProxy parameters that define the application performance on your server. Available options include timeouts and the application mode. The default value `http` instructs HAProxy to treat all incoming traffic as HTTP while a value such as `tcp` treats traffic as raw TCP connections on the server.

Follow the steps below to modify the configuration file with new sections that define how HAProxy accepts and distributes traffic as a load balancer to backend servers in the Vultr VPC 2.0 network.

1. Back up the original HAProxy configuration file.

CONSOLE

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

2. Open the HAProxy main configuration file using a [text editor like nano](#).

CONSOLE

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

3. Add the following configurations at the end of the file.

```
frontend website-frontend
  bind *:80,*:443
  option httpchk GET /healthcheck
  default_backend servers

backend servers
  balance roundrobin
  server server-1 10.128.0.3:80 weight 1 check
  server server-2 10.128.0.4:80 weight 1 check
```

The above configuration creates two new sections, `frontend` and `backend` with custom labels for identification purposes. The frontend section `website-frontend` defines how HAProxy listens for incoming requests on the server. Within the section:

- `bind`: Instructs HAProxy to listen for incoming connections on the HTTP port `80` and the HTTPS port `443`.
- `option httpchk`: Sets the URL path and request type to use with HAProxy when running health checks on the backend servers.

- `default_backend`: Defines the backend policy and hosts to use with HAProxy. The `servers` value enables the backend group to load balance.

The backend section with the custom label `servers` sets the target hosts to forward traffic and use with HAProxy. Within the section:

- `balance`: Sets the HAProxy load balancing algorithm to use with the backend servers. The value `roundrobin` selects servers in turns where traffic is evenly distributed to all available servers. Other supported modes include `leastconn` and `source` that define how HAProxy load balances traffic between the defined servers.
- `server`: Defines the backend servers to use with HAProxy. For example, the value `server-1 10.128.0.3:80 weight 1 check` creates a new backend server connection with the label `server 1` on the VPC address `10.128.0.3`, and the target port `80`. The server has a weight of `1` which defines its share of the traffic load for uneven distribution while the same weight with other servers such as `server 2` creates an even distribution of traffic. The `check` option enables health checking to stop forwarding traffic to a server in case it's unable to handle requests.

4. Add a new `listen` section at the end of the file with the following configurations to enable monitoring.

```
listen stats
  bind *:8404
  mode http
  stats enable
  stats uri /stats
  stats auth admin:your_password
  stats refresh 10s
```

Save and close the file.

The above configuration enables access to the HAProxy web dashboard that provides real-time traffic monitoring and statistics about the active server processes. The stats page offers insights into various metrics such

as the number of connections, session rates, request rates, response times, and errors. Within the above `listen` configuration:

- `stats`: Sets the listener label to `stats` for identification purposes.
- `bind`: Sets the target host port to listen for incoming connections to the HAProxy statistics page.
- `mode`: Specifies the HAProxy connection mode. The value `http` enables HTTP requests and optimizations on the specified port.
- `stats enable`: Enables the HAProxy statistics page.
- `stats uri`: Specifies the URL path to access the HAProxy stats page.
- `stats auth`: Sets the username and password for accessing the HAProxy statistics page.
- `stats refresh`: Sets the HAProxy statistics refresh interval. The value `10s` specifies a refresh rate of 10 seconds.

5. Restart the HAProxy service to apply your configuration changes.

CONSOLE

```
$ sudo systemctl restart haproxy
```

6. [Uncomplicated Firewall](#) (UFW) is active on Vultr Ubuntu servers and blocks connection requests by default. Allow the HAProxy port `80` through the firewall to enable HTTP connections on the server.

CONSOLE

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

7. Allow the HAProxy port `3804` to enable access to the statistics page.

CONSOLE

```
$ sudo ufw allow 3804/tcp
```

8. Restart the firewall to apply the connection rules.

CONSOLE

```
$ sudo ufw reload
```

9. Test connectivity to your backend server VPC addresses using the Ping utility.

- Server 1

CONSOLE

```
$ ping 10.128.0.3
```

Output:

```
PING 10.50.112.3 (10.50.112.3) 56(84) bytes of data.  
64 bytes from 10.50.112.3: icmp_seq=1 ttl=64 time=0.022 ms  
64 bytes from 10.50.112.3: icmp_seq=2 ttl=64 time=0.041 ms  
  
--- 10.50.112.3 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1021ms  
rtt min/avg/max/mdev = 0.022/0.031/0.041/0.009 ms
```

- Server 2

CONSOLE

```
$ ping 10.128.0.4
```

Output:

```
PING 10.50.112.3 (10.50.112.3) 56(84) bytes of data.  
64 bytes from 10.50.112.3: icmp_seq=1 ttl=64 time=0.022 ms  
64 bytes from 10.50.112.3: icmp_seq=2 ttl=64 time=0.041 ms  
  
--- 10.50.112.3 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1021ms  
rtt min/avg/max/mdev = 0.022/0.031/0.041/0.009 ms
```

Configure the Backend Servers

HAProxy sends health check requests to the backend servers to verify connectivity and handle connection requests. When a server is unavailable, HAProxy does not forward traffic until it passes the health check process again. Follow the steps below to configure each backend server and allow connections to the target port `80` to process HAProxy requests.

1. Access each of your backend servers using the HAProxy server and the associated VPC address. For example, access `server 1`.

CONSOLE

```
$ ssh root@10.128.0.3
```

2. Create a new non-root user with sudo privileges. For example, `sysadmin`.

CONSOLE

```
# adduser sysadmin && adduser sysadmin sudo
```

3. Switch to the non-root user account.

CONSOLE

```
# su - sysadmin
```

4. Update the server package index.

CONSOLE

```
$ sudo apt update
```

5. Install the Apache web server package.

CONSOLE

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

6. Enable Apache to start at boot time.

CONSOLE

```
$ sudo systemctl enable apache2
```

7. Navigate to the Apache web root directory `/var/www/html/`.

CONSOLE

```
$ cd /var/www/html/
```

8. Back up the default Apache `index.html` file as `index.BAK`.

CONSOLE

```
$ sudo mv index.html index.BAK
```

9. Create a new `index.html` file.

CONSOLE

```
$ sudo nano index.html
```

10. Add the following HTML contents to the file on each server.

- **Server-1:**

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Server 1</title>
  </head>
  <body>
    <h1>Hello!</h1>
    <p>This content is served by Server 1.</p>
```

```
</body>  
</html>
```

- **Server-2:**

```
HTML  
  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Server 2</title>  
  </head>  
  <body>  
    <h1>Hello!</h1>  
    <p>This content is served by Server 2.</p>  
  </body>  
</html>
```

Save and close the file.

The above web page configuration displays a `This content is served by Server [number]` message in response to a user request. HAProxy distributes traffic to each of the servers using the round-robin algorithm. In a production environment, all servers should run similar web applications to ensure the same content is served to all clients.

11. Grant the Apache web server ownership privileges to the file.

```
CONSOLE  
  
$ sudo chown -R www-data:www-data /var/www/html/index.html
```

12. Restart Apache to save the changes on each server.

```
CONSOLE  
  
$ sudo systemctl restart apache2
```

13. View your server IP information to verify the Vultr VPC 2.0 interface name.

CONSOLE

```
$ ip a
```

Your output should be similar to the one below.

```
....
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 56:00:04:ef:40:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.21/23 brd 192.168.0.255 scope global dynamic enp1s0
        valid_lft 86266sec preferred_lft 86266sec
    inet6 2a05:f000000:3/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 2591872sec preferred_lft 604672sec
    inet6 fe80::5400:4ff:feef:4023/64 scope link
        valid_lft forever preferred_lft forever
3: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc mq state UP group
default qlen 1000
    link/ether 5a:00:04:ef:40:23 brd ff:ff:ff:ff:ff:ff
    inet 10.128.0.3/20 brd 10.50.127.255 scope global enp8s0
        valid_lft forever preferred_lft forever
    inet6 fe80::5800:4ff:feef:4023/64 scope link
        valid_lft forever preferred_lft forever
```

Based on the above output, the VPC interface name is `enp8s0`.

14. Allow connection requests to the HTTP port `80` from your Vultr VPC 2.0 network interface on each server.

CONSOLE

```
$ sudo ufw allow in on enp8s0 to any port 80
```

15. Restart UFW to apply your firewall changes.

CONSOLE

```
$ sudo ufw reload
```

The HAProxy server listens for external web application requests and distributes traffic evenly to the backend servers. Each server returns its default web page contents as a response to the user request.

Access the HAProxy Accelerated Web Application

HAProxy distributed traffic in rounds to the backend servers within the Vultr VPC 2.0 network. Depending on the backend server health, a single server cannot process multiple requests at a time based on the `round-robin` algorithm that distributes the traffic evenly to all servers. Follow the steps below to access the HAProxy server and verify that your backend servers respond to all user requests.

1. Access your HAProxy server domain URL using a web browser such as Chrome.

```
$ http://haproxy.example.com
```

2. Verify that your web application displays in your browser window. The first request may be forwarded to `Server 1` and the next request to another server in the backend pool depending on the number of backend servers.

Hello!

This content is served by Server 1.

3. Refresh your web browser window and verify that `Server 2` responds to the request.

Hello!

This content is served by Server 2.

4. Refresh the web page again and verify that `Server 1` responds to your HTTP request again due to the `round-robin` algorithm.

Hello!

This content is served by Server 1.

5. Access the HAProxy statistics interface on the monitoring port `8404` to view your HAProxy connection requests and performance.

```
http://haproxy.example.com:8404
```

Enter the administrative user login details you created with the `stats auth` directive in your `listen stats` configuration.

6. Verify that the HAProxy statistics interface displays in your web browser with information about your frontend and backend performance.



VULTR

