By Falko Timme

Published: 2007-11-06 13:50

Setting Up A High-Availability Load Balancer (With Failover and Session Support) With HAProxy/Heartbeat On Debian Etch

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 10/27/2007

This article explains how to set up a two-node load balancer in an active/passive configuration with **HAProxy** and heartbeat on Debian Etch. The load balancer sits between the user and two (or more) backend Apache web servers that hold the same content. Not only does the load balancer distribute the requests to the two backend Apache servers, it also checks the health of the backend servers. If one of them is down, all requests will automatically be redirected to the remaining backend server. In addition to that, the two load balancer nodes monitor each other using heartbeat, and if the master fails, the slave becomes the master, which means the users will not notice any disruption of the service. HAProxy is session-aware, which means you can use it with any web application that makes use of sessions (such as forums, shopping carts, etc.).

From the HAProxy web site: ***"HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for web sites crawling under very high loads while needing persistence or Layer7 processing. Supporting tens of thousands of connections is clearly realistic with todays hardware. Its mode of operation makes its integration into existing architectures very easy and riskless, while still offering the possibility not to expose fragile web servers to the Net."***

I do not issue any guarantee that this will work for you!

1 Preliminary Note

In this tutorial I will use the following hosts:

- Load Balancer 1: `lb1.example.com`, IP address: `192.168.0.100`
- Load Balancer 2: `lb2.example.com`, IP address: `192.168.0.101`
- Web Server 1: `http1.example.com`, IP address: `192.168.0.102`

- Web Server 2: `http2.example.com`, IP address: `192.168.0.103`
- We also need a virtual IP address that floats between `lb1` and `lb2`: `192.168.0.99`

Here's a little diagram that shows our setup:

```
    shared IP=192.168.0.99
 192.168.0.100  192.168.0.101 192.168.0.102 192.168.0.103
 -------+------------+--------------+-----------+----------
        |            |              |           |
     +--+--+      +--+--+      +----+----+ +----+----+
     | lb1 |      | lb2 |      |  http1  | |  http2  |
     +-----+      +-----+      +---------+ +---------+
     haproxy      haproxy      2 web servers (Apache)
     heartbeat    heartbeat
```

The shared (virtual) IP address is no problem as long as you're in your own LAN where you can assign IP addresses as you like. However, if you want to use this setup with public IP addresses, you need to find a hoster where you can rent two servers (the load balancer nodes) in the same subnet; you can then use a free IP address in this subnet for the virtual IP address. Here in Germany, ***Hetzner*** is a hoster that allows you to do this - just talk to them.

`http1` and `http2` are standard Debian Etch Apache setups with the document root `/var/www` (the configuration of this default vhost is stored in `/etc/apache2/sites-available/default`). If your document root differs, you might have to adjust this guide a bit.

To demonstrate the session-awareness of HAProxy, I'm assuming that the web application that is installed on `http1` and `http2` uses the session id `JSESSIONID`.

2 Preparing The Backend Web Servers

We will configure HAProxy as a transparent proxy, i.e., it will pass on the original user's IP address in a field called `X-Forwarded-For` to the backend web servers. Of course, the backend web servers should log the original user's IP address in their access logs instead of the IP addresses of our load balancers. Therefore we must modify the `LogFormat` line in `/etc/apache2/apache2.conf` and replace `%h` with `%{X-Forwarded-For}i`:

http1/http2:

```
vi /etc/apache2/apache2.conf
```

```
[...]
#LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
[...]
```

Also, we will configure HAProxy to check the backend servers' health by continuously requesting the file *check.txt* (translates to */var/www/check.txt* if */var/www* is your document root) from the backend servers. Of course, these requests would totally bloat the access logs and mess up your page view statistics (if you use a tool like Webalizer or AWstats that generates statistics based on the access logs).

Therefore we open our vhost configuration (in this example it's in */etc/apache2/sites-available/default*) and put these two lines into it (comment out all other *CustomLog* directives in your vhost configuration):

```
vi /etc/apache2/sites-available/default
```

```
[...]
SetEnvIf Request_URI "^/check\.txt$" dontlog
CustomLog /var/log/apache2/access.log combined env=!dontlog
[...]
```

This configuration prevents that requests to *check.txt* get logged in Apache's access log.

Afterwards we restart Apache:

```
/etc/init.d/apache2 restart
```

... and create the file *check.txt* (this can be an empty file):

```
touch /var/www/check.txt
```

We are finished already with the backend servers; the rest of the configuration happens on the two load balancer nodes.

3 Installing HAProxy

Unfortunately HAProxy is available as a Debian package for Debian Lenny (testing) and Sid (unstable), but not for Etch. Therefore we will install the HAProxy package from Lenny. To do this, open */etc/apt/sources.list* and add the line *deb http://ftp2.de.debian.org/debian/ lenny main*; your */etc/apt/sources.list* could then look like this:

lb1/lb2:

```
vi /etc/apt/sources.list
```

```
deb http://ftp2.de.debian.org/debian/ etch main
deb-src http://ftp2.de.debian.org/debian/ etch main

deb http://ftp2.de.debian.org/debian/ lenny main

deb http://security.debian.org/ etch/updates main contrib
deb-src http://security.debian.org/ etch/updates main contrib
```

Of course (in order not to mess up our system), we want to install packages from Lenny only if there's no appropriate package from Etch - if there are packages from Etch and Lenny, we want to install the one from Etch. To do this, we give packages from Etch a higher priority in */etc/apt/preferences*:

```
vi /etc/apt/preferences
```

```
Package: *
Pin: release a=etch
Pin-Priority: 700

Package: *
Pin: release a=lenny
Pin-Priority: 650
```

(The terms `etch` and `lenny` refer to the appropriate terms in `/etc/apt/sources.list`; if you're using `stable` and `testing` there, you must use `stable` and `testing` instead of `etch` and `lenny` in `/etc/apt/preferences` as well.)

Afterwards, we update our packages database:

```
apt-get update
```

... upgrade the installed packages:

```
apt-get upgrade
```

... and install HAProxy:

```
apt-get install haproxy
```

4 Configuring The Load Balancers

The HAProxy configuration is stored in `/etc/haproxy.cfg` and is pretty straight-forward. I won't explain all the directives here; to learn more about all options, please read ***http://haproxy.1wt.eu/download/1.3/doc/haproxy-en.txt*** and ***http://haproxy.1wt.eu/download/1.2/doc/architecture.txt***.

We back up the original `/etc/haproxy.cfg` and create a new one like this:

lb1/lb2:

```
cp /etc/haproxy.cfg /etc/haproxy.cfg_orig

cat /dev/null > /etc/haproxy.cfg

vi /etc/haproxy.cfg
```

```
global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        #log loghost    local0 info
        maxconn 4096
        #debug
        #quiet
        user haproxy
        group haproxy

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        retries 3
        redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000
```

```
listen webfarm 192.168.0.99:80
       mode http
       stats enable
       stats auth someuser:somepassword
       balance roundrobin
       cookie JSESSIONID prefix
       option httpclose
       option forwardfor
       option httpchk HEAD /check.txt HTTP/1.0
       server webA 192.168.0.102:80 cookie A check
       server webB 192.168.0.103:80 cookie B check
```

Afterwards, we set *ENABLED* to *1* in */etc/default/haproxy*:

```
vi /etc/default/haproxy
```

```
# Set ENABLED to 1 if you want the init script to start haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

5 Setting Up Heartbeat

We've just configured HAProxy to listen on the virtual IP address *192.168.0.99*, but someone has to tell *lb1* and *lb2* that they should listen on that IP address. This is done by heartbeat which we install like this:

lb1/lb2:

```
apt-get install heartbeat
```

To allow HAProxy to bind to the shared IP address, we add the following line to */etc/sysctl.conf*:

```
vi /etc/sysctl.conf
```

```
[...]
net.ipv4.ip_nonlocal_bind=1
```

... and run:

```
sysctl -p
```

Now we have to create three configuration files for heartbeat, */etc/ha.d/authkeys*, */etc/ha.d/ha.cf*, and */etc/ha.d/haresources*. */etc/ha.d/authkeys* and */etc/ha.d/haresources* must be identical on *lb1* and *lb2*, and */etc/ha.d/ha.cf* differs by just one line!

lb1/lb2:

```
vi /etc/ha.d/authkeys
```

```
auth 3
3 md5 somerandomstring
```

somerandomstring is a password which the two heartbeat daemons on *lb1* and *lb2* use to authenticate against each other. Use your own string here. You have the choice between three authentication mechanisms. I use *md5* as it is the most secure one.

`/etc/ha.d/authkeys` should be readable by root only, therefore we do this:

lb1/lb2:

```
chmod 600 /etc/ha.d/authkeys
```

lb1:

```
vi /etc/ha.d/ha.cf
```

```
#
#       keepalive: how many seconds between heartbeats
#
keepalive 2
#
#       deadtime: seconds-to-declare-host-dead
#
deadtime 10
#
#       What UDP port to use for udp or ppp-udp communication?
#
udpport         694
bcast  eth0
mcast eth0 225.0.0.1 694 1 0
ucast eth0 192.168.0.101
#       What interfaces to heartbeat over?
udp     eth0
#
#       Facility to use for syslog()/logger (alternative to log/debugfile)
#
```

```
logfacility     local0
#
#       Tell what machines are in the cluster
#       node    nodename ...    -- must match uname -n
node    lb1.example.com
node    lb2.example.com
```

Important: As nodenames we must use the output of

```
uname -n
```

on `lb1` and `lb2`.

The `udpport`, `bcast`, `mcast`, and `ucast` options specify how the two heartbeat nodes communicate with each other to find out if the other node is still alive. You can leave the `udpport`, `bcast`, and `mcast` lines as shown above, but in the `ucast` line it's important that you specify the IP address of the other heartbeat node; in this case it's `192.168.0.101` (`lb2.example.com`).

On `lb2` the file looks pretty much the same, except that the `ucast` line holds the IP address of `lb1`:

lb2:

```
vi /etc/ha.d/ha.cf
```

```
#
#       keepalive: how many seconds between heartbeats
#
keepalive 2
#
#       deadtime: seconds-to-declare-host-dead
```

```
#
deadtime 10
#
#       What UDP port to use for udp or ppp-udp communication?
#
udpport         694
bcast  eth0
mcast eth0 225.0.0.1 694 1 0
ucast eth0 192.168.0.100
#       What interfaces to heartbeat over?
udp     eth0
#
#       Facility to use for syslog()/logger (alternative to log/debugfile)
#
logfacility     local0
#
#       Tell what machines are in the cluster
#       node    nodename ...    -- must match uname -n
node    lb1.example.com
node    lb2.example.com
```

lb1/lb2:

```
vi /etc/ha.d/haresources
```

```
lb1.example.com 192.168.0.99
```

The first word is the output of

```
uname -n
```

on *lb1*, no matter if you create the file on *lb1* or *lb2*! It is followed by our virtual IP address (*192.168.0.99* in our example).

Finally we start heartbeat on both load balancers:

lb1/lb2:

```
/etc/init.d/heartbeat start
```

Then run:

lb1:

```
ip addr sh eth0
```

... and you should find that *lb1* is now listening on the shared IP address, too:

```
lb1:~# ip addr sh eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:a5:5b:93 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.100/24 brd 192.168.0.255 scope global eth0
    inet 192.168.0.99/24 brd 192.168.0.255 scope global secondary eth0:0
    inet6 fe80::20c:29ff:fea5:5b93/64 scope link
       valid_lft forever preferred_lft forever
lb1:~#
```

You can check this again by running:

```
ifconfig
```

```
lb1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:A5:5B:93
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fea5:5b93/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:63983 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:92604963 (88.3 MiB)  TX bytes:2689903 (2.5 MiB)
          Interrupt:177 Base address:0x1400

eth0:0    Link encap:Ethernet  HWaddr 00:0C:29:A5:5B:93
          inet addr:192.168.0.99  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:177 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:56 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3888 (3.7 KiB)  TX bytes:3888 (3.7 KiB)

lb1:~#
```

As `lb2` is the passive load balancer, it should not be listening on the virtual IP address as long as `lb1` is up. We can check that with:

lb2:

```
ip addr sh eth0
```

The output should look like this:

```
lb2:~# ip addr sh eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:e0:78:92 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::20c:29ff:fee0:7892/64 scope link
       valid_lft forever preferred_lft forever
lb2:~#
```

The output of

```
ifconfig
```

shouldn't display the virtual IP address either:

```
lb2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:E0:78:92
          inet addr:192.168.0.101  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee0:7892/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:75127 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42144 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109669197 (104.5 MiB)  TX bytes:3393369 (3.2 MiB)
          Interrupt:169 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:56 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3888 (3.7 KiB)  TX bytes:3888 (3.7 KiB)

lb2:~#
```

6 Starting HAProxy

Now we can start HAProxy:

lb1/lb2:

```
/etc/init.d/haproxy start
```

7 Testing

Our high-availability load balancer is now up and running.

You can now make HTTP requests to the virtual IP address `192.168.0.99` (or to any domain/hostname that is pointing to the virtual IP address), and you should get content from the backend web servers.

You can test its high-availability/failover capabilities by switching off one backend web server - the load balancer should then redirect all requests to the remaining backend web server. Afterwards, switch off the active load balancer (`lb1`) - `lb2` should take over immediately. You can check that by running:

lb2:

```
ip addr sh eth0
```

You should now see the virtual IP address in the output on `lb2`:

```
lb2:~# ip addr sh eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:e0:78:92 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.0.255 scope global eth0
    inet 192.168.0.99/24 brd 192.168.0.255 scope global secondary eth0:0
    inet6 fe80::20c:29ff:fee0:7892/64 scope link
       valid_lft forever preferred_lft forever
lb2:~#
```

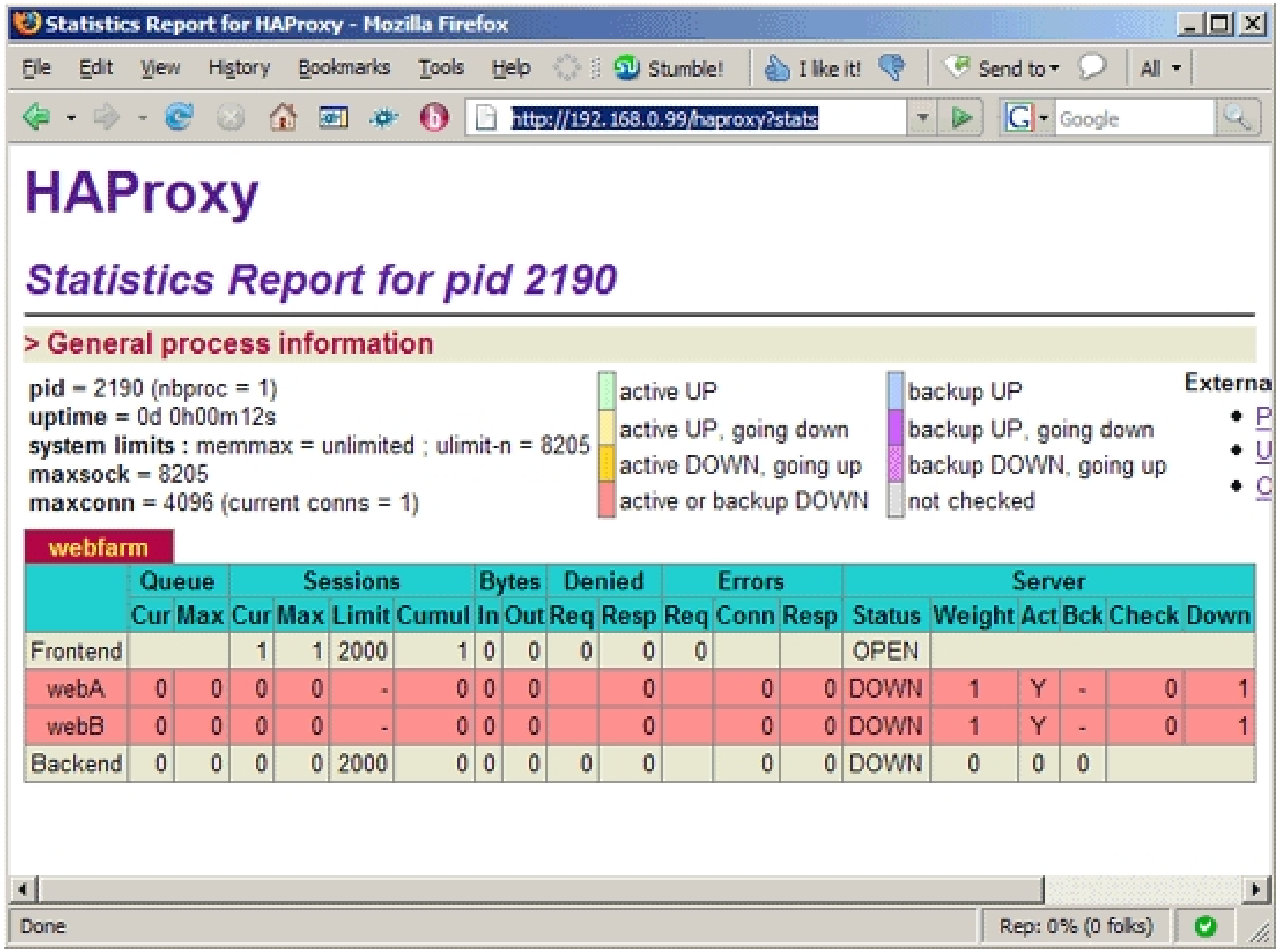
The same goes for the output of

```
ifconfig
```

When `lb1` comes up again, it will take over the master role again.

8 HAProxy Statistics

You might have noticed that we have used the options `stats enable` and `stats auth someuser:somepassword` in the HAProxy configuration in chapter 4. This allow us to access (password-protected) HAProxy statistics under the URL `http://192.168.0.99/haproxy?stats`. This is how it looks:


Statistics Report for HAProxy - Mozilla Firefox
File Edit View History Bookmarks Tools Help Stumble! I like it! Send to All
http://192.168.0.99/haproxy?stats
Google
Done
Rep: 0% (0 folks)

HAProxy

Statistics Report for pid 2190

> General process information

pid = 2190 (nbproc = 1)
uptime = 0d 0h00m12s
system limits : memmax = unlimited ; ulimit-n = 8205
maxsock = 8205
maxconn = 4096 (current conns = 1)

🟩	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

Externa
- P
- U
- C

webfarm

	Queue		Sessions				Bytes		Denied		Errors			Server					
	Cur	Max	Cur	Max	Limit	Cumul	In	Out	Req	Resp	Req	Conn	Resp	Status	Weight	Act	Bck	Check	Down
Frontend			1	1	2000	1	0	0	0	0	0			OPEN					
webA	0	0	0	0	-	0	0	0		0		0	0	DOWN	1	Y	-	0	1
webB	0	0	0	0	-	0	0	0		0		0	0	DOWN	1	Y	-	0	1
Backend	0	0	0	0	2000	0	0	0	0	0		0	0	DOWN	0	0	0		

If you don't need the statistics, just comment out or remove the `stats` lines from the HAProxy configuration.

9 Links

- HAProxy: ***http://haproxy.1wt.eu***
- Heartbeat: ***http://www.linux-ha.org/Heartbeat***
- Debian: ***http://www.debian.org***