Configure HAProxy

Wednesday, June 14, 2017

9:24 AM

Configuring Simple Load Balancing Using HAProxy

The following example uses HAProxy to implement a front-end server that balances incoming requests between two back-end web servers, and which is also able to handle service outages on the back-end servers.

Figure shows an HAProxy server (10.0.0.10), which is connected to an externally facing network (10.0.0/24) and to an internal network (192.168.1/24). Two web servers, websvr1 (192.168.1.71) and websvr2 (192.168.1.72), are accessible on the internal network. The IP address 10.0.0.10 is in the private address range 10.0.0/24, which cannot be routed on the Internet. An upstream network address translation (NAT) gateway or a proxy server provides access to and from the Internet.

Example HAProxy Configuration for Load Balancing

Machine generated alternative text: External Network
10.0.0/24
HAProxy Server
10.0.0.10
192.168.1.10
websvrl
192.168.1.71
Internal Network
192.168.1/24
NAT Gateway or Proxy
websvr2
192.168.1.72

The diagram shows an HAProxy server (10.0.0.10), which is connected to an externally facing network (10.0.0/24) and to an internal network (192.168.1/24). Two web servers, websvr1 (192.168.1.71) and websvr2 (192.168.1.72), are accessible on the internal network. The IP address 10.0.0.10 is in the private address range 10.0.0/24, which cannot be routed on the Internet. An upstream network address translation (NAT) gateway or a proxy server provides access to and from the Internet.

You might use the following configuration in /etc/haproxy/haproxy.cfg on the server:

global

daemon

log 127.0.0.1 local0 debug

maxconn 50000

nbproc 1

defaults

mode http

timeout connect 5s

timeout client 25s

timeout server 25s

timeout queue 10s

# Handle Incoming HTTP Connection Requests

listen http-incoming

mode http

bind 10.0.0.10:80

# Use each server in turn, according to its weight value

balance roundrobin

# Verify that service is available

option httpchk OPTIONS \* HTTP/1.1\r\nHost:\ www

# Insert X-Forwarded-For header

option forwardfor

# Define the back-end servers, which can handle up to 512 concurrent connections each

server websvr1 192.168.1.71:80 weight 1 maxconn 512 check

server websvr2 192.168.1.72:80 weight 1 maxconn 512 check

This configuration balances HTTP traffic between the two back-end web servers websvr1 and websvr2, whose firewalls are configured to accept incoming TCP requests on port 80.

After implementing simple /var/www/html/index.html files on the web servers and using curl to test connectivity, the following output demonstrate how HAProxy balances the traffic between the servers and how it handles the httpd service stopping on websvr1:

$ while true; do curl <http://10.0.0.10>; sleep 1; done

This is HTTP server websvr1 (192.168.1.71).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr1 (192.168.1.71).

This is HTTP server websvr2 (192.168.1.72).

...

This is HTTP server websvr2 (192.168.1.72).

<html><body><h1>503 Service Unavailable</h1>

No server is available to handle this request.

</body></html>

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr2 (192.168.1.72).

...

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr1 (192.168.1.71).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr1 (192.168.1.71).

...

^C

$

In this example, HAProxy detected that the httpd service had restarted on websvr1 and resumed using that server in addition to websvr2.

By combining the load balancing capability of HAProxy with the high availability capability of Keepalived or Oracle Clusterware, you can configure a backup load balancer that ensures continuity of service in the event that the master load balancer fails.

**Configuring HAProxy for Session Persistence**

Many web-based application require that a user session is persistently served by the same web server.

If you want web sessions to have persistent connections to the same server, you can use a balance algorithm such as hdr, rdp-cookie, source, uri, or url\_param.

If your implementation requires the use of the leastconn, roundrobin, or static-rr algorithm, you can implement session persistence by using server-dependent cookies.

To enable session persistence for all pages on a web server, use the cookie directive to define the name of the cookie to be inserted and add the cookie option and server name to the server lines, for example:

cookie WEBSVR insert

server websvr1 192.168.1.71:80 weight 1 maxconn 512 cookie 1 check

server websvr2 192.168.1.72:80 weight 1 maxconn 512 cookie 2 check

HAProxy includes an additional Set-Cookie: header that identifies the web server in its response to the client, for example: Set-Cookie: WEBSVR=N; path=page\_path. If a client subsequently specifies the WEBSVR cookie in a request, HAProxy forwards the request to the web server whose server cookievalue matches the value of WEBSVR.

The following example demonstrates how an inserted cookie ensures session persistence:

$ while true; do curl http://10.0.0.10; sleep 1; done

This is HTTP server websvr1 (192.168.1.71).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr1 (192.168.1.71).

^C

$ curl http://10.0.0.10 -D /dev/stdout

HTTP/1.1 200 OK

Date: ...

Server: Apache/2.4.6 ()

Last-Modified: ...

ETag: "26-5125afd089491"

Accept-Ranges: bytes

Content-Length: 38

Content-Type: text/html; charset=UTF-8

Set-Cookie: WEBSVR=2; path=/

This is HTTP server svr2 (192.168.1.72).

$ while true; do curl http://10.0.0.10 --cookie "WEBSVR=2;"; sleep 1; done

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr2 (192.168.1.72).

This is HTTP server websvr2 (192.168.1.72).

^C

To enable persistence selectively on a web server, use the cookie directive to specify that HAProxy should expect the specified cookie, usually a session ID cookie or other existing cookie, to be prefixed with the server cookie value and a ~ delimiter, for example:

cookie SESSIONID prefix

server websvr1 192.168.1.71:80 weight 1 maxconn 512 cookie 1 check

server websvr2 192.168.1.72:80 weight 1 maxconn 512 cookie 2 check

If the value of SESSIONID is prefixed with a server cookie value, for example: Set-Cookie: SESSIONID=N~Session\_ID;, HAProxy strips the prefix and delimiter from the SESSIONID cookie before forwarding the request to the web server whose server cookie value matches the prefix.

The following example demonstrates how using a prefixed cookie enables session persistence:

$ while true; do curl http://10.0.0.10 --cookie "SESSIONID=1~1234;"; sleep 1; done

This is HTTP server websvr1 (192.168.1.71).

This is HTTP server websvr1 (192.168.1.71).

This is HTTP server websvr1 (192.168.1.71).

^C

A real web application would usually set the session ID on the server side, in which case the first HAProxy response would include the prefixed cookie in the Set-Cookie: header.