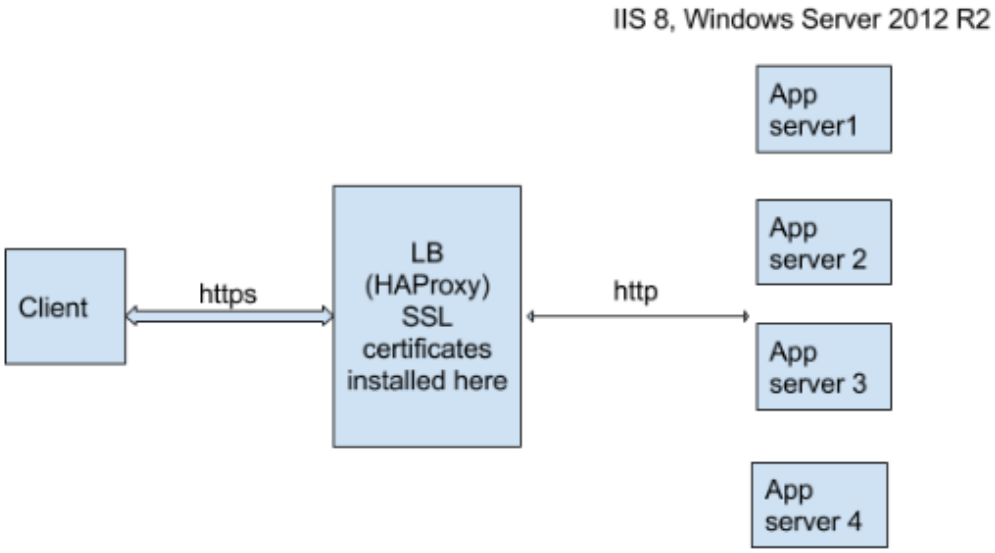
How can I enable HTTP/2 on HAProxy?

We have recently shifted from HTTP to HTTPS. As we have already moved to HTTPS, we are thinking of moving to HTTP/2 to get performance benefits.



As explained above that requests between browser and LB are secured (HTTPS) while communication between LB and app server still using HTTP

What is the possibility of enabling HTTP /2 with the current setup? Can we enable HTTP/2 between browser and LB while communication between LB and app servers remain on HTTP?

http https haproxy http2

edited Jun 21 '17 at 13:40

mikemaccana
39.4k ● 43 ● 220 ● 286

asked Nov 17 '16 at 13:38

ssharma
90 ● 1 ● 1 ● 10

3 Answers

HAProxy 1.8 supports HTTP/2

From the 1.8 announcement:

> HAProxy 1.8 now supports HTTP/2 on the client side (in the frontend sections) and can act as a gateway between HTTP/2 clients and your HTTP/1.1 and HTTP/1.0 applications.

You'll need the `h2` directive in your `haproxy.conf`. From **CertSimple's HAProxy HTTP/2 and dynamic load balancing guide**:

```
frontend myapp
  bind :443 ssl crt /path/to/cert.crt alpn h2,http/1.1
  mode http
```

Older versions of HAProxy

Older versions of HAProxy like 1.6 and 1.7 only support pass-through HTTP/2 - ie, directing traffic onto a seperate app server that supports HTTP/2. This is significantly more complicated - see other answers on how to do this. To terminate HTTP/2 and read the traffic on HAProxy, you'll need HAProxy 1.8.

edited May 30 at 9:16

answered Feb 15 '17 at 14:09

mikemaccana
39.4k ● 43 ● 220 ● 286

2

Quick note here: alpn http/1.1,h2 - the order of alpn options is "preference". So if you want users to be using HTTP2 first then use: alpn h2,http/1.1 – Paul Lemke Dec 1 '17 at 15:34

Fixed, thanks @PaulLemke! 😀 – mikemaccana Dec 4 '17 at 15:50

Just a quick note. The conversion from HTTP2 to HTTP1.1 still has some issues. For our HTTPS GZip POST, the Transfer-Encoding (not included in HTTP2) was not auto converted to HTTP1.1 And thus the body not read. If anyone has the same problem, try using only http 1.1 only. There the protocol requires Transfer-Encoding = chunked (even though it's default on HTTP2). Thus if done manually on HAProxy, this works. Letting the system convert automatically does not add this header. – Tobias Reich Aug 9 at 11:57

@TobiasReich Add a link to the HAProxy bug you reported – mikemaccana Aug 10 at 13:11

I admit, I haven't filed a bug report right now. (To be honest I haven't found their bugtracker page. The Github page also doesn't offer issues...) – Tobias Reich Aug 13 at 11:44

haproxy doesn't really support http/2 yet

The only support it does have, is to detect a http/2 request, and pass the https / tcp443 tcp stream to a server that does support https and http/2

here is someone else's guide http://m12.io/blog/http-2-with-haproxy-and-nginx-guide

answered Nov 18 '16 at 21:52

Scott Farrell
47 ● 3

Following should work on your load balancer if you are able to run some `NginX` alongside with `HaProxy`. `NginX` is (ab)used as a pure SSL terminator, not as a full featured web server, so no content is served by this `NginX`.

> Warning: This was done in a hurry, so nothing is verified that this really works. Some examples are missing, so sorry for the links.

I call this idea after the famous picture of *Munchhausen, pulling himself and the horse out of a mire*:

The Munchhausen Method

First, do a H2 setup in HaProxy like in the answer of Scott Farrell with following tweaks:

```
frontend http-in
    mode http
    bind *:80
    option forwardfor
    default_backend nodes-http

frontend https-in
    mode tcp
    bind *:443 ssl crt /etc/ssl/dummy.pem alpn h2,http/1.1
    use_backend nodes-http2 if { ssl_fc_alpn -i h2 }
    default_backend nodes-http

frontend http-lo
    mode http
    bind 127.0.0.1:82
    #http-request set-header X-Forwarded-For req.hdr_ip([X-Forwarded-For])
    default_backend nodes-http

backend nodes-http
    mode http
    server node1 web.server:80 check

backend nodes-http2
    mode tcp
    server loadbalancer 127.0.0.1:81 check send-proxy
```

This loops the `HTTP/2` connection back to your loadbalancer machine and accepts the decoded requests to enter loadbalancing again via `http-lo`.

Now on the LB itself, start `NginX` to listen on Port `81` as in the config instance to terminate the `HTTP/2` connection and proxy it back to your loadbalancer again.

In NginX be sure to:

- Use send-proxy-protocol in NginX
- Terminate the SSL using `HTTP/2` in `NginX`
- Proxy everything transparently (aka. dumb) back to `HaProxy` port `82`

  ```
  # Sorry, example `NginX`-config is missing here,
  # but it includes something like:
  proxy_pass http://127.0.0.1:82;
  ```

- Do not forget to include the Client-IP via `X-Forwarded-For` header in the proxy request (I do not know how to configure NginX to use the "Send Proxy" Protocol on outgoing proxy requests).

Note that this setup is mostly static. The changing part is about all those domains and their TLS-certs.

ASCII picture of `HTTP/2` request flow

```
     Browser
        |  HTTP/2
        V
     Loadbalancer HaProxy *:443
        |  frontend https-in
        |  backend nodes-http2
        |  send-proxy
        |  TCP (transparent, HTTP/2)
        V
     Loadbalancer NginX 127.0.0.1:81
        |  HTTP/2 termination
        |  proxy_protocol
        |  proxy_pass 127.0.0.1:82
        |  Add header X-Forwarded-For
```

```
        |  HTTP
        V
    Loadbalancer HaProxy 127.0.0.1:82
        |  frontend https-lo
        |  Forward Header X-Forwarded-For
        |  backend nodes-http
        |  # DO YOUR LOADBALANCING HERE
        |  HTTP
        V
     web.server:80
```

Yes, it loops 2 times through HaProxy, but thanks to how fast HaProxy works this works lightning fast.

The real inefficient part is when it comes to uncompress the `HTTP/2` headers into plain `HTTP` headers ..

answered Oct 27 '17 at 13:24

Tino

3,578 ● 1 ● 27 ● 37