

HAProxyConf 2022 Paris, France

# Boost your web-app with HAProxy and Varnish



**HAPROXY**  
**Conf 2022**

8-9 Nov, Paris

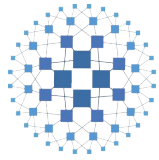
Jérémy Lecour  
@jlecour – CTO

**evolix**



<https://gitea.evolix.org/evolix/haproxyconf-2022>





# HAPROXY

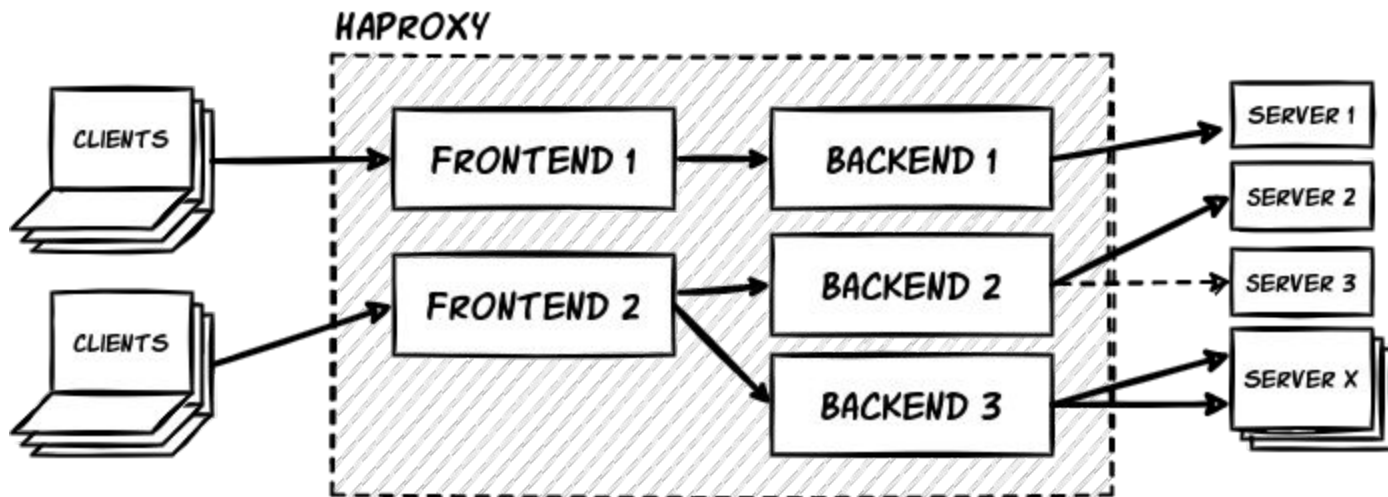
&



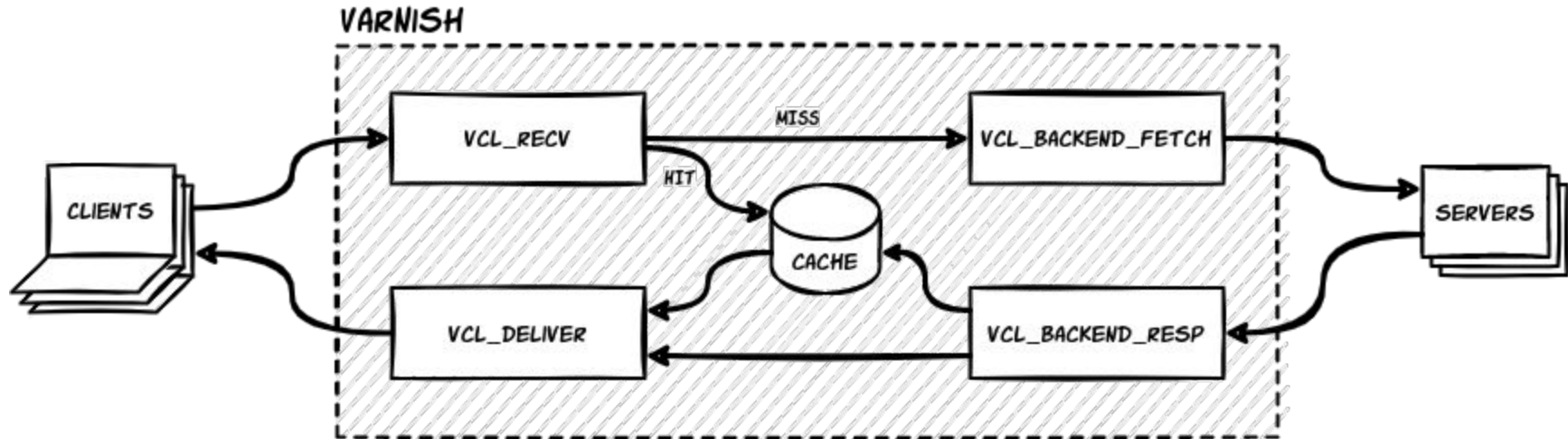
# VARNISH CACHE

- Free Libre Open Source Software
- Run on many platforms
- Resource efficient and stable
- Great documentation and communities
- Commercial support available

# HAProxy – overview

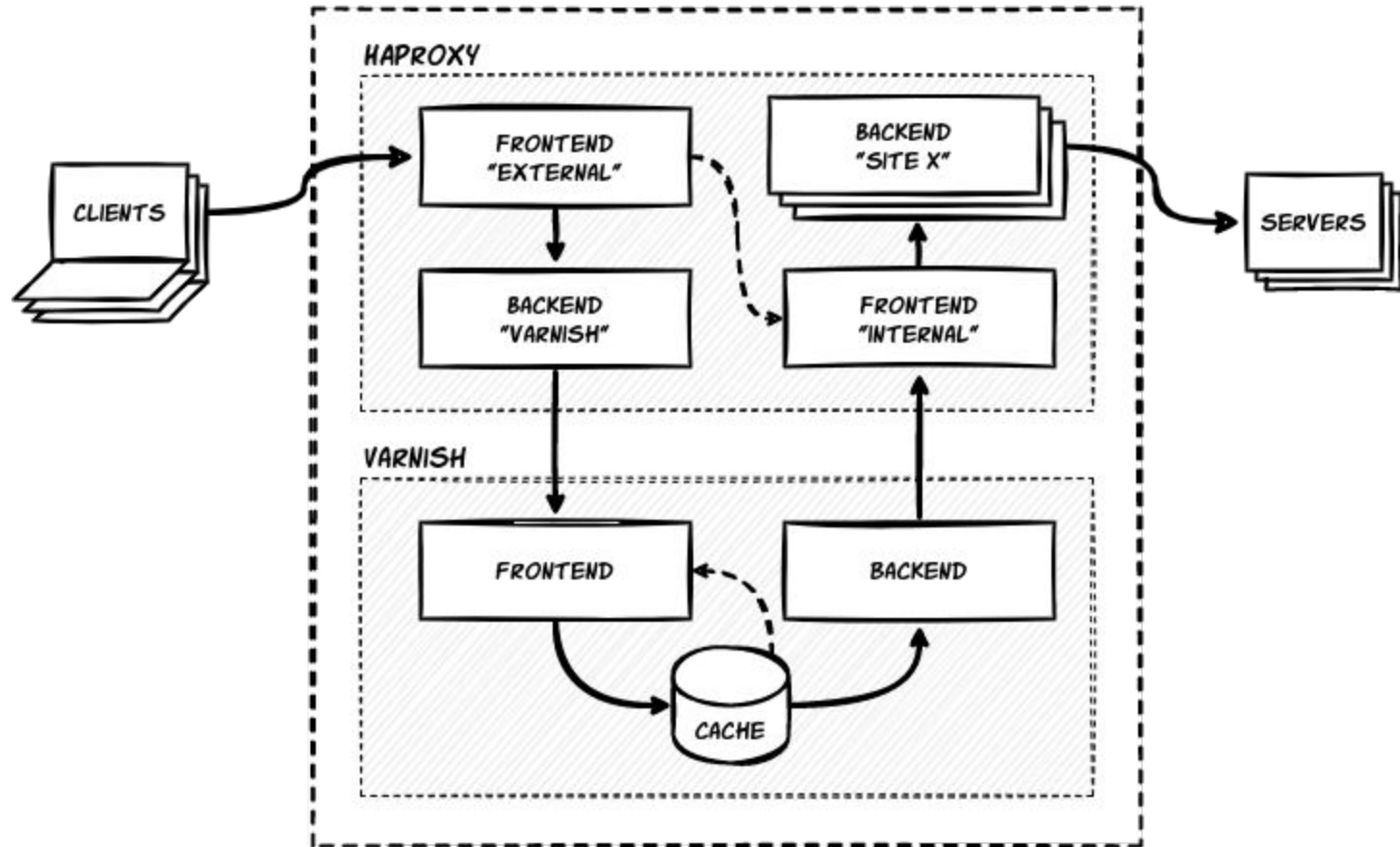


# Varnish – overview



**HAProxy + Varnish = “Boost”** – a CDN-like service

# BOOST PROXY



**Let's zoom in**

# Multiple sites or apps

```
frontend external
  acl example_com_domains hdr(host) -i example.com
  acl foo_bar_domains     hdr(host) -i foo-bar.com foo-bar.org
  [...]
  use_backend example_com if example_com_domains
  use_backend foo_bar     if foo_bar_domains

backend example_com
  [...]
  server srv10-1 192.0.10.1:80

backend foo_bar
  [...]
  server srv20-1 192.0.20.1:80
  server srv20-2 192.0.20.2:80
```

haproxy.cfg



# Pass the request to Varnish – cache or no cache, there is no 503

```
frontend external
  # Is Varnish available?
  acl varnish_available nbsrv(varnish) gt 0
  # Pass the request to Varnish
  use_backend varnish if varnish_available
  # ... or fall back to a direct backend
  default_backend direct_backend
```

haproxy.cfg

```
backend varnish
  option httpchk HEAD /varnishcheck
  server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2
```

```
sub vcl_recv {
  if (req.url == "/varnishcheck") {
    return(synth(200, "Hi HAProxy, I'm fine!"));
  }
  [...]
}
```

varnish.vcl

# Pass the request to Varnish – should I stay or should I go?

```
frontend external
  acl example_com_domains hdr(host) -i example.com
  [...]
  use_backend varnish if example_com_domains
```

haproxy.cfg

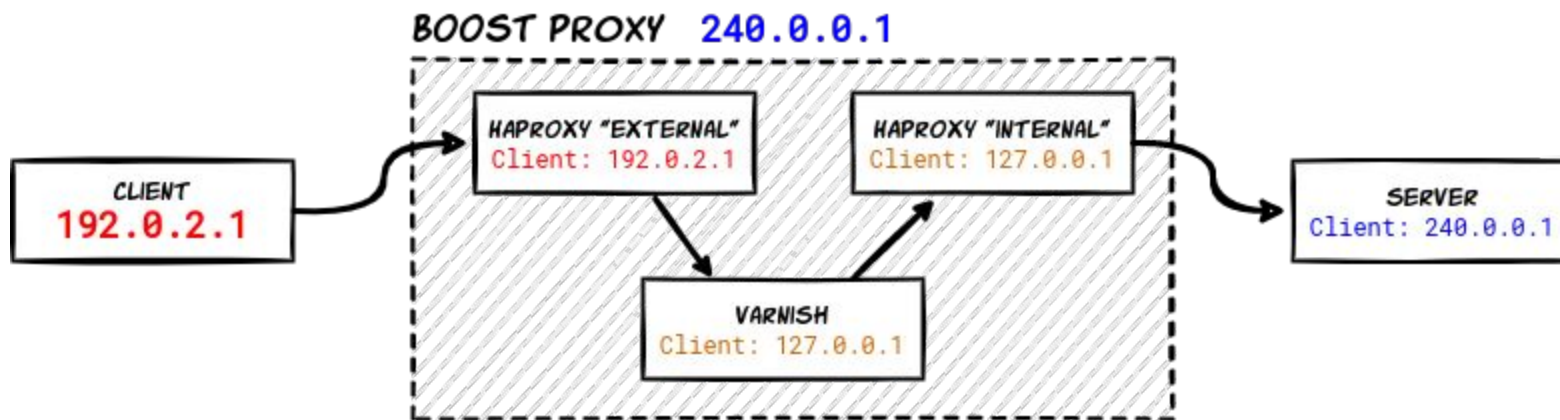
```
frontend external
  acl use_cache if hdr(host) -f /etc/haproxy/cached_domains
  [...]
  use_backend varnish if use_cache
```

haproxy.cfg

```
frontend external
  acl varnish_http_verb method GET HEAD PURGE
  [...]
  use_backend varnish if varnish_http_verb
```

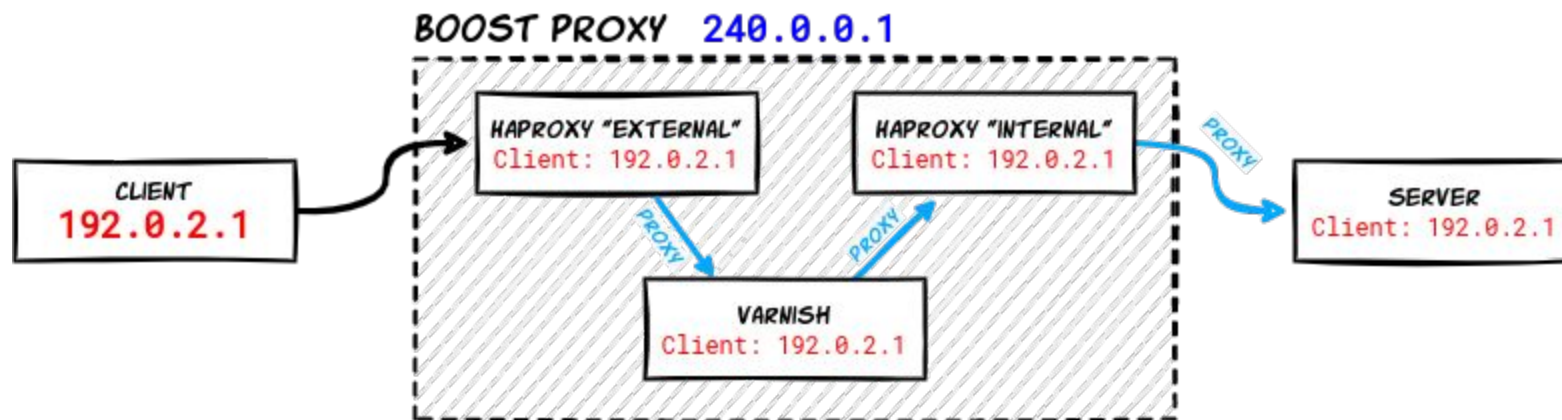
haproxy.cfg

# PROXY protocol – who is talking to me?



X-Forwarded-For: 192.0.2.1, 127.0.0.1, 127.0.0.1, 240.0.0.1

# PROXY protocol – who is talking to me?



# PROXY protocol – how are you doing?

```
frontend internal
    bind /run/haproxy-frontend-default.sock user root mode 666 accept-proxy

backend varnish
    server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2

backend example_com
    server example-hostname 1.2.3.4:443 check observe layer4 ssl verify none send-proxy-v2
```

haproxy.cfg

```
$ /usr/sbin/varnishd [...] -a /run/varnish.sock,PROXY
```

init

```
backend default {
    .path = "/run/haproxy-frontend-default.sock";
    .proxy_header = 1;
}
```

varnish.vcl

# PROXY protocol – how are you doing?

```
frontend internal                                     haproxy.cfg
  bind /run/haproxy-frontend-default.sock user root mode 666 accept-proxy

backend varnish
  server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2

backend example_com
  server example-hostname 1.2.3.4:443 check observe layer4 ssl verify none send-proxy-v2
```

```
$ /usr/sbin/varnishd [...] -a /run/varnish.sock,PROXY                                     init
```

```
backend default {                                     varnish.vcl
  .path = "/run/haproxy-frontend-default.sock";
  .proxy_header = 1;
}
```

# PROXY protocol – how are you doing?

```
frontend internal
  bind /run/haproxy-frontend-default.sock user root mode 666 accept-proxy

backend varnish
  server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2

backend example_com
  server example-hostname 1.2.3.4:443 check observe layer4 ssl verify none send-proxy-v2
```

haproxy.cfg

```
$ /usr/sbin/varnishd [...] -a /run/varnish.sock,PROXY
```

init

```
backend default {
  .path = "/run/haproxy-frontend-default.sock";
  .proxy_header = 1;
}
```

varnish.vcl

# PROXY protocol – how are you doing?

```
frontend internal                                     haproxy.cfg
  bind /run/haproxy-frontend-default.sock user root mode 666 accept-proxy

backend varnish
  server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2

backend example_com
  server example-hostname 1.2.3.4:443 check observe layer4 ssl verify none send-proxy-v2
```

```
$ /usr/sbin/varnishd [...] -a /run/varnish.sock,PROXY                                     init
```

```
backend default {                                     varnish.vcl
  .path = "/run/haproxy-frontend-default.sock";
  .proxy_header = 1;
}
```



# PROXY protocol – how are you doing?

```
frontend internal                                     haproxy.cfg
  bind /run/haproxy-frontend-default.sock user root mode 666 accept-proxy

backend varnish
  server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2

backend example_com
  server example-hostname 1.2.3.4:443 check observe layer4 ssl verify none send-proxy-v2
```

```
$ /usr/sbin/varnishd [...] -a /run/varnish.sock,PROXY                                     init
```

```
backend default {                                     varnish.vcl
  .path = "/run/haproxy-frontend-default.sock";
  .proxy_header = 1;
}
```

# PROXY protocol – how are you doing?

```
frontend internal                                     haproxy.cfg
  bind /run/haproxy-frontend-default.sock user root mode 666 accept-proxy

backend varnish
  server varnish_sock /run/varnish.sock check observe layer7 maxconn 3000 inter 1s send-proxy-v2

backend example_com
  server example-hostname 1.2.3.4:443 check observe layer4 ssl verify none send-proxy-v2
```

```
$ /usr/sbin/varnishd [...] -a /run/varnish.sock,PROXY                                     init
```

```
backend default {                                     varnish.vcl
  .path = "/run/haproxy-frontend-default.sock";
  .proxy_header = 1;
}
```

# PROXY protocol – debug all the things

```
$ /usr/sbin/varnishd [...] -a 127.0.0.1:82 -a /run/varnish.sock,PROXY
```

init

```
$ curl --resolve www.example.com:82:127.0.0.1  
  ↳ --header "X-Forwarded-Proto: https"  
  ↳ http://www.example.com:82/foo/bar
```

command line

# PROXY protocol – all the way down

**If possible** : use PROXY protocol to the final servers  
add a secondary non-PROXY interface

**Pro-tip** : Apache + ForensicLog

```
+X@IkelsspdiNAko5YHK9HAAAAC4 | forensic.log
  ↳ GET /blog/ HTTP/1.1|
  ↳ host:jeremy.lecour.fr|user-agent:curl/7.64.0|accept:*/*|x-varnish:65545|
  ↳ x-forwarded-for:192.0.2.1, 240.0.0.1|x-forwarded-port:443|x-forwarded-proto:https
-X@IkelsspdiNAko5YHK9HAAAAC4
```

# HTTP tagging – standard and custom headers

```
$ curl -sv https://jeremy.lecour.fr/blog/
GET /blog/ HTTP/1.1
Host: jeremy.lecour.fr
User-Agent: curl/7.74.0
Accept: */*

HTTP/1.1 200 OK
Date: Thu, 8 Nov 2022 09:32:30 GMT
Server: Apache
Last-Modified: Tue, 19 May 2020 16:59:15 GMT
ETag: "23c8-5a603330a9ec0"
Accept-Ranges: bytes
Content-Type: text/html
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; img-src 'self'; script-src 'self'; style-src 'self'
Strict-Transport-Security: max-age=63072000
```

command line

# HTTP tagging – X-Forwarded-\*

```
frontend external
  bind 0.0.0.0:80,:::80
  bind 0.0.0.0:443,:::443 ssl

  option forwardfor

  http-request set-header X-Forwarded-Port %[dst_port]

  http-request set-header X-Forwarded-Proto http if !{ ssl_fc }
  http-request set-header X-Forwarded-Proto https if { ssl_fc }
```

haproxy.cfg

# HTTP tagging – X-Unique-ID

```
frontend external
  # Add unique ID to each request
  http-request set-header X-Unique-ID %[uuid()] unless { hdr(X-Unique-ID) -m found }
```

haproxy.cfg

# HTTP tagging – X-Boost-Step1

```
frontend external
    # Tag the request for Step 1
    http-request add-header X-Boost-Step1 "haproxy-external"

    # Tag the response for Step 1
    http-response add-header X-Boost-Step1 "haproxy-external; ssl-frontend"    if { ssl_fc }
    http-response add-header X-Boost-Step1 "haproxy-external; no-ssl-frontend" if !{ ssl_fc }
    http-response set-header X-Boost-Server my-hostname
```

haproxy.cfg



# HTTP tagging – X-Boost-Step2

```
sub vcl_recv {  
    # Tag the request for Step 2  
    set req.http.X-Boost-Step2 = "varnish";  
}
```

varnish.vcl

```
sub vcl_deliver {  
    # Tag the response for Step 2  
    if (resp.http.Set-Cookie && resp.http.Cache-Control) {  
        set resp.http.X-Boost-Step2 = "varnish; set-cookie; cache-control";  
    } elseif (resp.http.Set-Cookie) {  
        set resp.http.X-Boost-Step2 = "varnish; set-cookie; no-cache-control";  
    } elseif (resp.http.Cache-Control) {  
        set resp.http.X-Boost-Step2 = "varnish; no-set-cookie; cache-control";  
    } else {  
        set resp.http.X-Boost-Step2 = "varnish; no-set-cookie; no-cache-control";  
    }  
}
```

varnish.vcl

# HTTP tagging – X-Boost-Step3

```
frontend internal
  # Tag the request for Step 3
  http-request add-header X-Boost-Step3 "haproxy-internal"
  # Tag the response for Step 3
  http-response add-header X-Boost-Step3 "haproxy-internal; ssl-backend"    if { ssl_bc }
  http-response add-header X-Boost-Step3 "haproxy-internal; no-ssl-backend" if !{ ssl_bc }
```

haproxy.cfg

# HTTP tagging – Full HAProxy log

```
frontend external haproxy.cfg
  [...]
  http-response add-header X-Haproxy-Log-External "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta
  ↳ %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r"

frontend internal
  [...]
  http-response add-header X-Haproxy-Log-Internal "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta
  ↳ %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r"
```



**Don't do this in production**

# Filtering at HAProxy level

```
frontend external
  [...]
  # Reject the request at the TCP level if source is in the denylist
  tcp-request connection reject if { src -f /etc/haproxy/deny_ips }
```

haproxy.cfg

```
userlist vip_users
  user johndoe password $6$k6y3o.eP$JlKBx9za9667qe4 (...) xHSwRv6J.C0/D7cV91
```

haproxy.cfg

```
frontend external
  [...]
  redirect scheme https code 301 if !{ ssl_fc }
  redirect prefix https://example-to.org code 301 if { hdr(host) -i example-from.org }
  http-request auth realm "VIP Section" if !{ http_auth(vip_users) }
```

# Maintenance mode – global

```
frontend external
  [...]
  # List of IP that will not go the maintenance backend
  acl maintenance_ips src -f /etc/haproxy/maintenance_ips
  # Go to maintenance backend, unless your IP is whitelisted
  use_backend maintenance if !maintenance_ips

backend maintenance
  http-request set-log-level silent
  # Custom 503 error page
  errorfile 503 /etc/haproxy/errors/maintenance.http
  # With no server defined, a 503 is returned for every request
```

haproxy.cfg

# Maintenance mode – per site

```
frontend external
  [...]
  acl example_com_domains hdr(host) -i example.com

  acl maintenance_ips src -f /etc/haproxy/maintenance_ips
  acl example_com_maintenance_ips src -f /etc/haproxy/example_com/maintenance_ips

  use_backend example_com_maintenance if example_com_domains !example_com_maintenance_ips
  ↪ !maintenance_ips
```

haproxy.cfg

# Local services

```
frontend external
# Is the request coming for the server itself (stats...)
acl self      hdr(host) -i my-hostname my-hostname.domain.tld
acl munin     hdr(host) -i munin
# Detect Let's Encrypt challenge requests
acl letsencrypt path_dir -i /.well-known/acme-challenge

use_backend local      if self
use_backend local      if munin
use_backend letsencrypt if letsencrypt

backend local
option httpchk HEAD /haproxy-check
server localhost 127.0.0.1:81 send-proxy-v2 maxconn 10

backend letsencrypt
# Use this if the challenge is managed locally
server localhost 127.0.0.1:81 send-proxy-v2 maxconn 10
# Use this if the challenge is managed remotely
### server my-certbot-challenge-manager 192.168.2.1:80 maxconn 10
```

haproxy.cfg

# High availability – for the application





# High availability – for HAProxy and Varnish

BOOST PROXY 1

BOOST PROXY 2

# High availability – for HAProxy and Varnish – round-robin DNS



# High availability – for HAProxy and Varnish – active-active

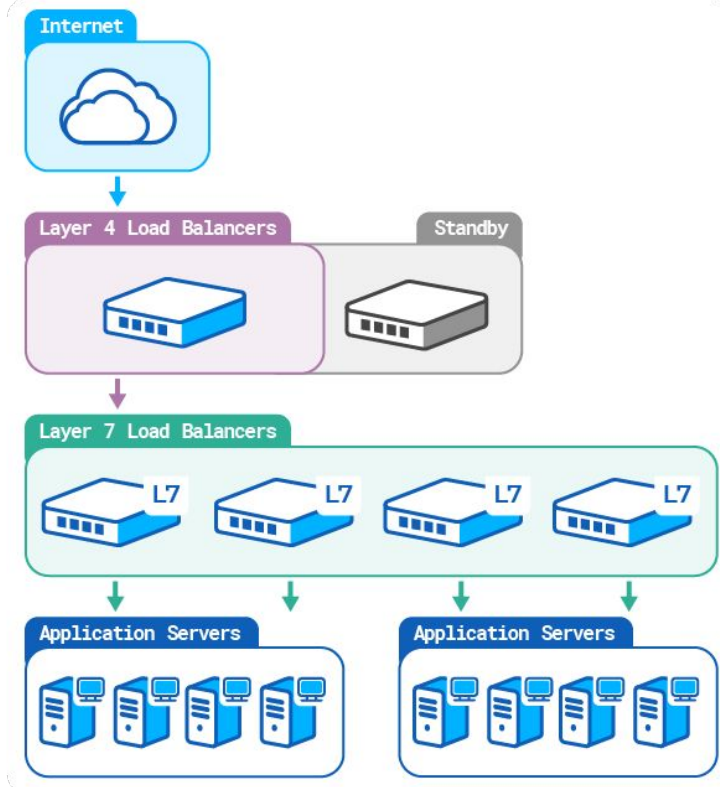
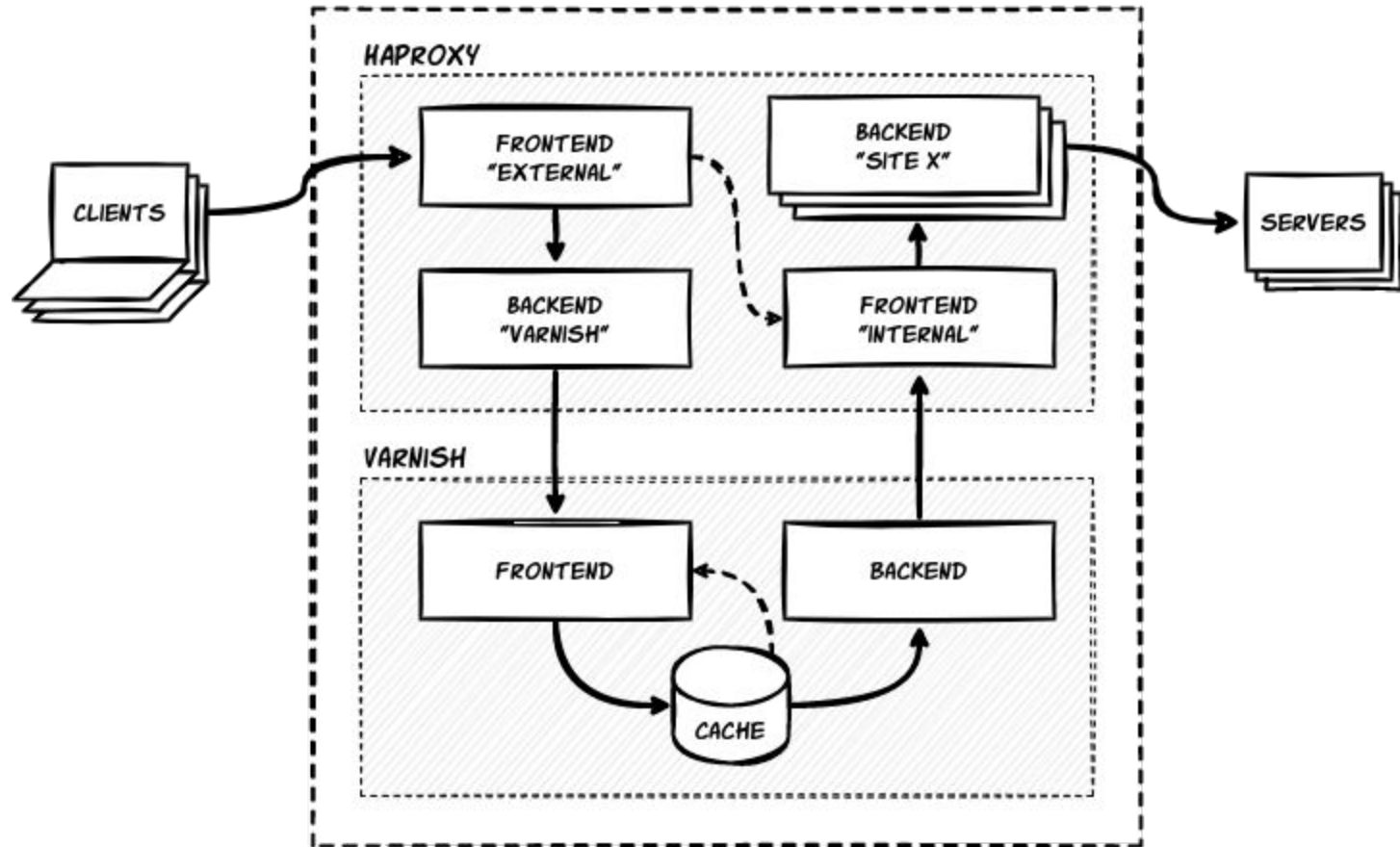


Image shamelessly stolen from HAProxy docs

# Questions & Answers

# BOOST PROXY



# Thank you



**HAPROXY**  
**Conf 2022**

8-9 Nov, Paris

**Jérémy Lecour**  
@jlecour – CTO

**evolix**



<https://gitea.evolix.org/evolix/haproxyconf-2022>

