HAProxy Config Files

HAProxy

```
# /etc/haproxy/haproxy.cfg
# (c) 2026, Bernard Condrau (this config file)
#
# 2026-09-14: initial
#
# Features:
#   HTTP/1, HTTP/2, HTTP/3, acme integration, static blacklist, static and
dynamic whitelist
#   handles transition from one backend server without reverse proxy to
HAProxy with several backend servers
# Notes:
#   # = comment, ## = acme integration, remove ## once you have set up acme
(https://www.haproxy.com/blog/haproxy-and-let-s-encrypt)
#   Remove the 'default_backend' once transition is done from Certbot /
Apache to acme / HAProxy
#   Remove 'public_inbound_https' once all hosts are SSL terminated here
#
# 0. Settings
global
    log /dev/log local0
    log /dev/log local1 notice
    chroot /var/lib/haproxy
    user haproxy
    group haproxy
    daemon

    # Modern SSL/TLS security settings for Debian 13
    # Minimum TLS version requirement
    ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

    # TLS 1.3 Cipher Suites (handled by ssl-default-bind-ciphersuites)
    ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256

    # TLS 1.2 Ciphers (handled by ssl-default-bind-ciphers)
    ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-
GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-
SHA256:DHE-RSA-AES256-GCM-SHA384

    # Administrative socket for stats/management
    stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd
listeners
```

```
    stats timeout 30s
    ## setenv ACCOUNT_THUMBPRINT
'lCufto4sDRTHdmWL0EugFywGV54hBCuTTXvwifi65R4'

defaults
    log     global
    mode    tcp
    log-format "%ci:%cp [%t] %ft %b/%s %Tw/%Tc/%Tt %B %ts
%ac/%fc/%bc/%sc/%rc %sq/%bq"
    timeout connect 5s
    timeout client  50s
    timeout server  50s

# 1. Public Interface (SNI Router)
# Receives traffic through TCP (HTTP)
frontend public_inbound_http
    bind *:80
    mode http

    # Handle Letsencrypt requests
    ##http-request return status 200 content-type text/plain lf-string
"%[path,field(-1,/)].${ACCOUNT_THUMBPRINT}\n" if { path_beg '/.well-
known/acme-challenge/' }

    # Deny traffic based on origin
    http-request deny if { src -f /etc/haproxy/blacklist.lst }

    # Redirect all incoming traffic to HTTPS permanently
    http-request redirect scheme https code 301 unless { ssl_fc }

# Receives traffic through TCP (HTTP/1, HTTP/2)
frontend public_inbound_https
    bind :443
    mode tcp
    # Deny traffic based on origin
    tcp-request connection reject if { src -f /etc/haproxy/blacklist.lst }

    # Wait for the TLS client hello to parse the SNI extension
    tcp-request inspect-delay 5s
    tcp-request content accept if { req.ssl_hello_type 1 }

    # Define the domains (one line for each) you want to terminate SSL for
    # Remove once all hosts are terminated here
    acl terminate_ssl req.ssl_sni -i bak.condrau.com

    # Route conditionally based on SNI
    use_backend http_dispatcher if terminate_ssl
    default_backend passthrough_server

# Receives traffic through UDP (HTTP/3), decrypts SSL, and processes HTTP
rules
```

```
# HTTP/3 cannot be routed to apache web server without decrypting SSL first
frontend public_inbound_https3

    # pass certs directory path so all pem files will be loaded
    # uncomment the following line once all hosts are SSL terminated here
    # bind :443
    bind quic4@:443 ssl crt /etc/haproxy/certs/ accept-proxy strict-sni
    mode http
    http-reuse aggressive

    # Handle Letsencrypt requests
    ##http-request return status 200 content-type text/plain lf-string
"%[path,field(-1,/)].${ACCOUNT_THUMBPRINT}\n" if { path_beg '/.well-
known/acme-challenge/' }

    # Deny traffic based on origin (host aware)
    acl trusted_ip src -f /etc/haproxy/whitelist.lst # static whitelist
    acl whitelist_ip src,in_table(stick_table_whitelist) # dynamic whitelist
    acl backuppc hdr(host) -i bak.condrau.com
    http-request deny if backuppc !trusted_ip !whitelist_ip

    # Let clients know we support HTTP/3
    http-response set-header Alt-Svc 'h3=":443"; ma=86400'

    # Inject headers so backend servers know the original request was HTTPS
    http-request set-header X-Forwarded-Proto https
    http-request set-header X-Forwarded-Port 443
    http-request set-header X-Forwarded-For %[src]

    # Route traffic based on HTTP host headers
    use_backend backuppc_server if backuppc
    default_backend http_server

# 2. Transparent passthrough for encrypted TLS domains
backend passthrough_server
    mode tcp

    # downstream server handling its own TLS certificates (Calypso)
    server calypso 192.168.3.18:443 check

# 3. Intermediary loopback backend
# This forwards traffic to the Layer 7 (HTTP) processing frontend
backend http_dispatcher
    mode tcp
    server loopback 127.0.0.1:8443 send-proxy-v2

# 4. Layer 7 http frontend (HTTP/1, HTTP/2)
# Receives traffic from the loopback, decrypts SSL, and processes HTTP rules
frontend internal_inbound_http

    # pass certs directory path so all pem files will be loaded
```

```
    bind 127.0.0.1:8443 ssl crt /etc/haproxy/certs/ accept-proxy strict-sni
    mode http
    http-reuse aggressive

    # Handle Letsencrypt requests
    ##http-request return status 200 content-type text/plain lf-string
"%[path,field(-1,/)].${ACCOUNT_THUMBPRINT}\n" if { path_beg '/.well-
known/acme-challenge/' }

    # Deny traffic based on origin (host aware)
    acl trusted_ip src -f /etc/haproxy/whitelist.lst # static whitelist
    acl whitelist_ip src,in_table(stick_table_whitelist) # dynamic whitelist
    acl backuppc hdr(host) -i bak.condrau.com
    http-request deny if backuppc !trusted_ip !whitelist_ip

    # Let clients know we support HTTP/3
    http-response set-header Alt-Svc 'h3=":443"; ma=86400'

    # Inject headers so backend servers know the original request was HTTPS
    http-request set-header X-Forwarded-Proto https
    http-request set-header X-Forwarded-Port 443
    http-request set-header X-Forwarded-For %[src]

    # Route traffic based on HTTP host headers
    use_backend backuppc_server if backuppc
    default_backend http_server

# 5. HTTP backends
backend backuppc_server
    mode http
    cookie SERVERID insert indirect nocache
    server epione 192.168.1.50:8080 check

backend http_server
    mode http
    cookie SERVERID insert indirect nocache
    server serenity 192.168.3.100:8080 check

# 6. stick table that stores IPv4 addresses
backend stick_table_whitelist
    mode http
    stick-table type ip size 10 expire 1h
```

Whitelist

- Add a static whitelist file for IPs which always have access:

```
# /etc/haproxy/whitelist.lst
# Allow static addresses
```

```
183.88.218.149
109.205.169.18
```

Blacklist

- Add a blacklist file for IPs not allowed to access:

```
# /etc/haproxy/blacklist.lst
# Deny malicious IP addresses
148.66.10.94
5.188.62.0/24
54.39.29.64
185.185.134.115
185.46.46.0/24
```

Dynamic Whitelist

- Add IP address to whitelist stick table:

```
echo "set table stick_table_whitelist key 192.168.1.99" | socat
/run/haproxy/admin.sock stdio
```

- Remove IP address from whitelist stick table:

```
echo "clear table stick_table_whitelist key 192.168.1.99" | socat
/run/haproxy/admin.sock stdio
```

- Show content of whitelist stick table:

```
echo "show table stick_table_whitelist" | socat /run/haproxy/admin.sock
stdio
```

- To add an IP address from a DDNS service you should set the expire duration short in `HAProxy.cfg` because you cannot rely on the IP address being the same when you want to remove the IP address. Add this short bash script to update the stick table, it's an example for duckdns.org:

```
#!/bin/bash
IP=$(dig +short host.duckdns.org)
ERROR=$?
#
# do not update if DIG timed out
if [ $ERROR -eq 0 ]; then
    # update HAProxy stick table
    echo "set table stick_table_whitelist key $IP" | socat
/run/haproxy/admin.sock stdio
```

```
fi
```

- Call the script from another linux computer or WSL (on Windows with a shortcut to):

```
wsl ssh -p <port> <domain> "sudo /home/<user>/batch/update.sh"
```

- To make this work you need to add <user> to file `update-ip` in `/etc/sudoers.d` so he can execute the command as root without password:

```
vim /etc/sudoers.d/update-ip
<user> ALL=(ALL:ALL) NOPASSWD: /home/<user>/batch/update.sh
```

- To remove an IP address from a DDNS service you should remove the entire table because you cannot rely on the IP address being the same when you want to clear the table:

```
#!/bin/bash
#
# update HAProxy stick table
echo "clear table stick_table_whitelist" | socat
/run/haproxy/admin.sock stdio
```

- This way you can add your computer to the trusted IP list (whitelist) no matter what is the current IP address, provided that you set up a dynamic domain name which you can update with your Powerhell script below.

Powershell

- Create the following Powershell script and use it with a shortcut to `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -File “D:\update-dns.ps1”`:

```
# D:\update-dns.ps1
# This is an example Powershell file for DuckDNS updater, see
https://www.duckdns.org
param($domain)
$duckdomain = if (!$domain) { "my-default-domain-name" } else { $domain
}
$addr = curl.exe -s "https://api.ipify.org"
curl.exe -k
"https://www.duckdns.org/update?domains=$duckdomain&token=xxxxxxxx-yyyy
-zzzz-*my*-*duckdnstoken*&ip=$addr"
```

From:
https://wiki.condrau.com/ - **Bernard's Wiki**

Permanent link:
https://wiki.condrau.com/deb13:haconfig?rev=1789450763

Last update: **2026/09/15 12:39**