# HAProxy als Reverse-Proxy mit Letâ??s-Encrypt-Zertifikaten einrichten

## Description

In diesem Artikel erklÃ¤re ich kurz, wie wir den HAProxy installieren und diesen als Reverse-Proxy konfigurieren. Dieser Reverse-Proxy lÃ¤sst uns unsere Anwendungen Ã¼ber SSL zugreifen. Dazu erstellt dieser automatisch entsprechende Letâ??s-Encrypt-Zertifikate. Die Zertifikate in diesem Beispiel werden Ã¼ber einen Hetzner-API-Key validiert und erstellt.

> **Info:** Jeder Dienst, der im Internet erreichbar ist, stellt ein potenzielles Sicherheitsrisiko dar.

## DurchfÃ¼hrung

Um den HAProxy zu installieren, brauchen wir einen Linux-Server, auf dem wir Root-Rechte haben. In meinem Beispiel verwende ich einen Debian-13-Server mit einer DNS-Verwaltung bei Hetzner.

Um einen HAProxy einzurichten, brauchen wir SSH-Zugriff auf einen Linux-Server. In meinem Beispiel verwende ich einen Debian-13-Server dafÃ¼r.

Im ersten Schritt mÃ¼ssen wir uns das Paket fÃ¼r HAProxy auf unserem Server installieren. Dazu fÃ¼hren wir den folgenden Befehl aus:

```
sudo apt update && sudo apt upgrade -y && sudo apt install haproxy -y
```

Sobald die Installation abgeschlossen ist, kÃ¶nnen wir mit dem folgenden Befehl prÃ¼fen, ob HAProxy installiert ist.

```
haproxy -v
```

## Grund-Konfiguration & Ordnerstruktur anlegen

HAProxy legt seine Konfigurationen unter **/etc/haproxy** ab. Diesen Ordner bauen wir etwas aus, damit die Konfiguration etwas Ã¼bersichtlicher ist.

Im ersten Schritt modifizieren wir einmal die â??Grund Konfigurationâ??. Die **haproxy.cfg** in meinem System sieht wie folgt aus:

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
```

Diese Konfiguration mÃ¼sste um folgenden Inhalt ergÃ¤nzt werden, damit die Anfragen auf Port 80 (HTTP) und Port 443 (HTTPS) angenommen und bearbeitet werden.

```
frontend fe_http
        bind *:80
        http-request redirect scheme https code 301

frontend fe_https
        bind *:443 ssl crt /etc/haproxy/certs/cert.pem

        http-request set-header X-Forwarded-Proto https
        http-request set-header X-Forwarded-Port 443

        use_backend %[req.hdr(host),lower,map_dom(
/etc/haproxy/maps/vhosts.map,be_default)]
```

Um das Debugging und Monitoring zu erleichtern, kÃ¶nnen wir uns die Stats Page von HaProxy und den Prometheus Exporter noch hinzufÃ¼gen. DafÃ¼r wÃ¤re der folgende Inhalt zustÃ¤ndig:

```
frontend fe_prometheus
        bind *:8405
        mode http
        http-request use-service prometheus-exporter
        no log

listen stats
        bind *:8080
        mode http
        stats enable
        stats uri /haproxy?stats
```

> **Info:** Wenn wir diese Konfiguration Ã¼bernehmen, kÃ¶nnen wir die Prometheus-Metriken Ã¼ber den Port 8405 abrufen. Falls wir aber uns die Stats-Page anschauen wollen, mÃ¼ssen wir den folgenden Link verwenden: ***<IP-Addresse>:8080/haproxy?stats***

Im nÃ¤chsten Schritt legen wir die Ordnerstruktur an. DafÃ¼r kÃ¶nnen wir die folgenden Befehle verwenden:

```
mkdir -p certs
mkdir -p maps
mkdir -p vhosts
```

Jetzt legen wir eine Default VHost-Datei an, die als Fallback fÃ¼r unseren Reverseproxy dient.

```
cat > /etc/haproxy/vhosts/default.cfg <<EOF
backend be_default
        mode http
        http-request return status 404 content-type text/plain string "Host nicht gefunde

EOF
```

## Zertifikate anlegen

Um die Zertifikate anzulegen, verwenden wir das Programm **Lego**. Dieses erstellt fÃ¼r uns die Zertifikate und erneuert diese sobald diese ablaufen. Um dieses Programm zu installieren, suchen wir einmal im GitHub Repo (https://github.com/go-acme/lego/releases) nach der neuesten aktuellen Version. In meinem Fall ist das die 5.0.2. Dieses laden wir einmal fÃ¼r unser System herunter und entpacken dieses.

```
cd /tmp
wget
https://github.com/go-acme/lego/releases/download/v5.0.2/lego_v5.0.2_linux_amd64.tar.gz

tar -xzf lego_v5.0.2_linux_amd64.tar.gz

mv lego /usr/local/bin
chmod +x /usr/local/bin/lego
```

Sobald wir diese Befehle ausgefÃ¼hrt haben, mÃ¼ssen wir einmal testen, ob wir **Lego** erfolgreich installiert haben. Dazu einmal den folgenden Befehl ausfÃ¼hren:

```
lego --version
```

Im nÃ¤chsten Schritt legen wir eine Datei an, indem wir unseren API-Token abspeichern. Dazu fÃ¼hren wir einmal wieder die folgenden Befehle aus:

```
mkdir -p /etc/lego
nano /etc/lego/dns_api.env
```

In meinem Fall kommt hier der folgende Inhalt rein, da ich hier die Hetzner DNS-Verwaltung verwende:

```
export HETZNER_API_TOKEN="DEIN_API_KEY"
```

Danach setzen wir einmal die Permissions richtig.

```
chmod 600 /etc/lego/dns_api.env
```

Jetzt verwenden wir den folgenden Befehl, um das Zertifikat zu erstellen. Beachte jedoch hier deine Domains, die ein Zertifikat erhalten sollen:

```
lego run \
  --env-file /etc/lego/dns_api.env \
  --accept-tos \
  --email administrator@domain.de \
  --dns hetzner \
  --dns.resolvers 1.1.1.1:53 \
  --dns.resolvers 8.8.8.8:53 \
  --domains "*.domain.de" \
  --domains "domain.de" \
  --server letsencrypt \
  --pem \
  --path /etc/lego \
  --dns.propagation.wait 60s
```

Sobald der Vorgang abgeschlossen ist, sollten wir den Pfad angezeigt bekommen, wo unser Zertifikat abgelegt wurde. Dieses kopieren wir uns einmal in unser HaProxy Verzeichnis. Und prÃ¼fen die Config und starten die Dienste neu.

```
mkdir -p /etc/haproxy/certs

cp /etc/lego/certificates/_.domain.de.pem /etc/haproxy/certs/domain.de.pem

chmod 600 /etc/haproxy/certs/domain.de.pem

haproxy -c -f /etc/haproxy/haproxy.cfg

systemctl reload haproxy
```

Um die automatische Erneuerung jetzt einzurichten, kÃ¶nnen wir den folgenden Cronjob verwenden. Hier mÃ¼ssen natÃ¼rlich die Pfade angepasst werden.

```
15 3 * * * lego run --env-file /etc/lego/dns_api.env --accept-tos --email
administrator@domain.de --dns hetzner --dns.propagation.wait 60s --domains
"*.domain.de" --domains "domain.de" --server letsencrypt --pem --path /etc/lego && cp
/etc/lego/certificates/_.domain.de.pem /etc/haproxy/certs/domain.de.pem && chmod 600
/etc/haproxy/certs/domain.de.pem && haproxy -c -f /etc/haproxy/haproxy.cfg && systemctl
 reload haproxy
```

## Virtual-Hosts anlegen

Im ersten Schritt legen wir die â??Mapping-Dateiâ?? an, die die Domainnamen mit dem entsprechenden Backend verknÃ¼pft:

Dazu erstellen wir die folgende Datei: **/etc/haproxy/maps/vhosts.map**

```
domain.de be_domainde
```

> **Info:** In dieser Datei legst du die Zuordnung fÃ¼r alle Virtualhosts an.

Im Anschluss erstellen wir eine Backend-Datei fÃ¼r unseren Virtual Host. Die Datei muss unter folgendem Pfad angelegt werden: ***/etc/haproxy/vhosts/BACKEND_NAME.cfg***

```
backend be_domainde
        mode http
        server domainde01 <IP>:<Port> check
```

Sobald wir die Datei angelegt haben, kÃ¶nnen wir einmal prÃ¼fen ob unsere Konfiguration sauber ist. DafÃ¼r verwenden wir den folgenden Befehl:

```
haproxy -c -f /etc/haproxy/haproxy.cfg
```

Wenn alles passt, kÃ¶nnen wir den Dienst einmal neuladen.

```
systemctl reload haproxy
```

**Category**

1. HaProxy
2. Linux
3. Software

**Date Created**
13.05.2026
**Author**
administrator