HAProxy as Reverse Proxy + SSL Termination

Here i will show you how to install HAProxy on Ubuntu Server 18.04 LTS and also how to configure it as a reverse proxy. A reverse proxy means that you can access multiple web servers through one port, usually 80 for http or 443 for https.

I will also show you how to implement SSL encryption using Lets Encrypt and set up automatic renewal of the certificate. And also how to re-direct all HTTP requests to HTTPS.

HTTP Config | HTTPS Config | SSL Certificate

Installing HAProxy

Step 1

First of all, we are going to update our system packages:

```
sudo apt-get update
sudo apt-get upgrade
```

Depending on your network setup, you may also want to set Ubuntu to use a Static IP address rather than using DHCP.

Step 2

Now we need to enable the PPA Repository for HAProxy and install it, at the time of writing the latest stable version is 2.1, but you can check haproxy.org for the latest available version.

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:vbernat/haproxy-2.1
sudo apt-get update
sudo apt-get install haproxy=2.1.\*
```

Now HAProxy has successfuly installed on your server and is ready for configuration.

Configuring as a Reverse Proxy (HTTP)

So, you have multiple servers that you would like to access externally but you only have one public IP address available? A reverse proxy is for you, it can look at the URL that is being requested and return the correct site from a backend server. This also adds a layer of security since there is no direct access to your backend servers. This is typically known as a DMZ if you configure your network right too with firewall rules. So, lets configure this:

```
sudo nano /etc/haproxy/haproxy.cfg
```

This will take you to your standard configuration file, there are multiple parts to your config. We will only be focusing on frontends and backends today. A frontend is basically a port that the server is listening on, and a backend is one of your servers with the site/files on it. **Example** In this example i will have my server listening on port 80. If the URL requested is example1.deanlongstaff.com, then the server 192.168.20.10 will be served, if it is example2.deanlongstaff.com then the server 192.168.20.30 will be served.

```
    mode http
    use_backend server1 if { hdr(host) -i example1.deanlongstaff.com }
    use_backend server2 if { hdr(host) -i example2.deanlongstaff.com }

backend server1
    server wordpress 192.168.20.10:80

backend server2
    server wiki 192.168.20.30:80
```

Now you've configured it to work with HTTP. Well done! You can also change your backend to retreive different ports, so for example you have a backend that uses port 8200 for its web UI. You're backend would look like this:

```
backend example
    server example 192.168.20.40:8200
```

You could now access this backend by using a URL rule like above on port 80. But what if one of your backends uses https? Well, in that case we need to add a few more conditions:

```
backend example
    server example 192.168.20.40:443 check ssl verify none
```

```
sudo service haproxy reload
```

This is then accessible via port 80 aslong as you have a URL rule set for it.

Configuration for HTTPS

HTTP is great and all, but who uses that for public access these days? Well unless you are a loonatic… no one! A HTTPS configuration is pretty similar in HAProxy, we just need to add a bits and bobs. However you also need an SSL Certificate, you can get one for free from Lets Encrypt, here is how. First we need to add port 443 to our frontend since this is the port HTTPS runs on. We also need to specify our certificate location, but you choose where that is.

```
    bind *:443 ssl crt /etc/ssl/example.com/example.pem
```

Now, you want to redirect all HTTP traffic to HTTPS? This means if anyone tries to access your sites on HTTP, they will be enforced to use HTTPS. Add this to your frontend configuration:

```
    # Redirect HTTP to  HTTPS
    redirect scheme https code 301 if !{ ssl_fc }
```

And there you go, HTTPS is configured, but wont work if you dont have an SSL certificate, scroll down to get one for free.

Lets Encrypt SSL Certificate

Installing Lets Encrypt

First we need to install certbot to our server, this is the program that can retrieve us an SSL Certificate from Lets Encrypt.

```
sudo add-apt-repository -y ppa:certbot/certbot
sudo apt-get update
sudo apt-get install -y certbot
```

Configure Lets Encrypt

If you have followed this blog from top to bottom, then you may already have your config ready. Now, you need to copy your config, paste it into a notepad and save it. As you also need to delete it from your haproxy.cfg. We need to add a Frontend and Backend for Lets Encrypt:

```
sudo nano /etc/haproxy/haproxy.cfg

# The frontend only listens on port 80
# If it detects a LetsEncrypt request, is uses the LE backend
```

```
    # Test URI to see if its a letsencrypt request
    acl letsencrypt-acl path_beg /.well-known/acme-challenge/
    use_backend letsencrypt-backend if letsencrypt-acl

    default_backend be-scalinglaravel

# LE Backend
backend letsencrypt-backend
    server letsencrypt 127.0.0.1:8888

# Normal (default) Backend
# for web app servers
backend be-scalinglaravel
    # Config omitted here
```

Now that you have the Lets Encrypt config ready, we need to reload HAProxy:

```
sudo service haproxy reload
```

New Certificate
Okay, so now you want to get a certificate from lets encrypt..... make sure these are in place:

- Public DNS to point your domains to your Public IP Address
- Port Forwarding to send port 80 to your HAProxy instance (Best to leave port 443 disabled for this)

Now, this is the certbot command we will use:

```
sudo certbot certonly --standalone -d example.com \
    --non-interactive --agree-tos --johnsmith@example.com \
    --http-01-port=8888
```

You need to fill your information into this command:

1. "-d example.com" This is the domain we are creating the certificate for. You can use multiple -d flags for multiple domains in a single certificate, but all of the domains must have DNS entries. For example -d example.com -d test.example.com -d yougetit.com
2. "–agree-tos –johnsmith@example.com" This is your email address that will be used to send expiration alerts, so replace it with your email.

Make SSL useable by HAProxy

Lets Encrypt gives you an SSL Certificate in 2 files, but HAProxy requires 1 ".pem" file. No problem, we can merge them!
First we will make a folder for the file we want to save and then we will run a command to take both those files and save them into one.

```
sudo mkdir -p /etc/ssl/example.com

sudo cat /etc/letsencrypt/live/example.com/fullchain.pem \
    /etc/letsencrypt/live/example.com/privkey.pem \
    | sudo tee /etc/ssl/example.com/example.pem
```

So, now you have your certificate saved in /etc/ssl/example.com/example.pem
Remember, you need to specify your SSL Certificate location in your config when you bind port 443:

```
    bind *:443 ssl crt /etc/ssl/example.com/example.com.pem
```

Obviously you want to replace "example.com" to your own domain when doing this.

Automatic Certificate Renewal

Yep that's right, that long tedious task to get an SSL certificate can all be automated so you never need to do it again! As long as your DNS entries and port forwarding stays intact you should have no issues with this! **Step 1** We need to edit the CRON job that lets encrypt automatically creates. Our is going to run a script on the first day of every month. Replace whatever is in it with this line:

```
sudo nano /etc/cron.d/certbot

0 0 1 * * root bash /opt/update-certs.sh
```

Now we need to create the script that we just told the system to run every month. This script is basically going to run all the commands that we've just ran to get our

```
sudo nano /opt/update-certs.sh

#!/usr/bin/env bash

# Renew the certificate
certbot renew --force-renewal --tls-sni-01-port=8888

# Concatenate new cert files, with less output (avoiding the use tee and its output to stdout)
bash -c "cat /etc/letsencrypt/live/example.com/fullchain.pem /etc/letsencrypt/live/example.com/privkey.pem > /etc/ssl/example.com/examp

# Reload  HAProxy
service haproxy reload
```

You also need to add a bit to your frontend config and add a backend to your **haproxy.cfg** file. This is basically so that it requests a new certificate over HTTPS since port 443 will be available when renewing.

```
frontend www-https

    # New line to test URI to see if its a letsencrypt request
    acl letsencrypt-acl path_beg /.well-known/acme-challenge/
    use_backend letsencrypt-backend if letsencrypt-acl

# Lets Encypt Backend
backend letsencrypt-backend
    server letsencrypt 127.0.0.1:8888
```

Full Example Configuration

This is a full example of **haproxy.cfg** that is listening on both http and https, has https re-direction enabled, a backend that uses https, lets encrypt automatic renewal configurations and 3 separate URL rules and backends:

```
frontend www-https
    bind *:80
    bind *:443 ssl crt /etc/ssl/example.com/example.com.pem

    # Redirect HTTP to  HTTPS
    redirect scheme https code 301 if !{ ssl_fc }

    #Lets Encrypt Renewal URI Test
    acl letsencrypt-acl path_beg /.well-known/acme-challenge/
    use_backend letsencrypt-backend if letsencrypt-acl

    mode http
    use_backend example1 if { hdr(host) -i example1.deanlongstaff.com }
    use_backend example2 if { hdr(host) -i example2.deanlongstaff.com }
    use_backend example3 if { hdr(host) -i example3.deanlongstaff.com }

    #Backend to use if no URL specified
    default_backend example1


backend example1
    server server1 192.168.40.2:80

backend example2
    server example2 192.168.40.10:80

backend example3
    server example3 192.168.40.30:443 check ssl verify none

# Lets Encrypt Backend
backend letsencrypt-backend
    server letsencrypt 127.0.0.1:8888
```

Dean Longstaff © 2021 All Rights Reserved