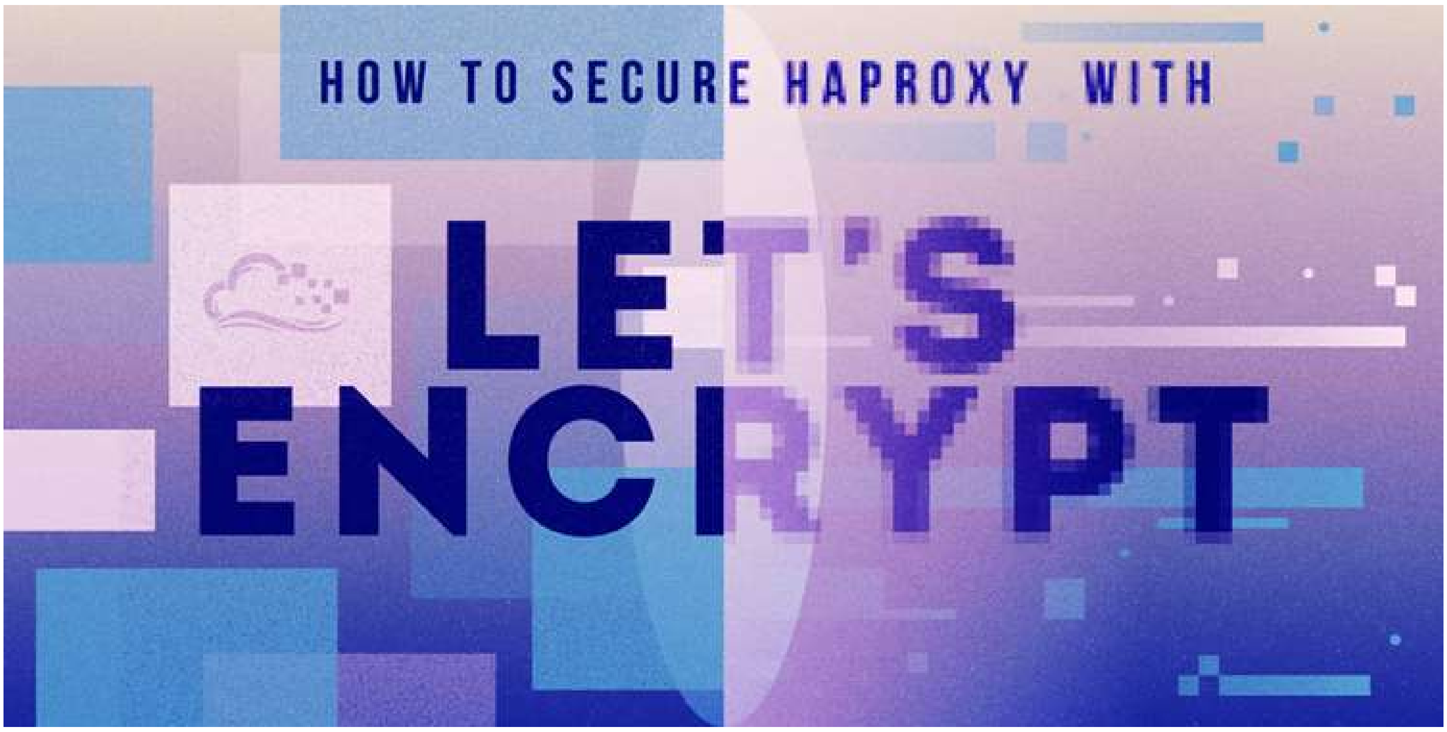
DigitalOcean | Community Menu

By: Mitchell Anicas Subscribe



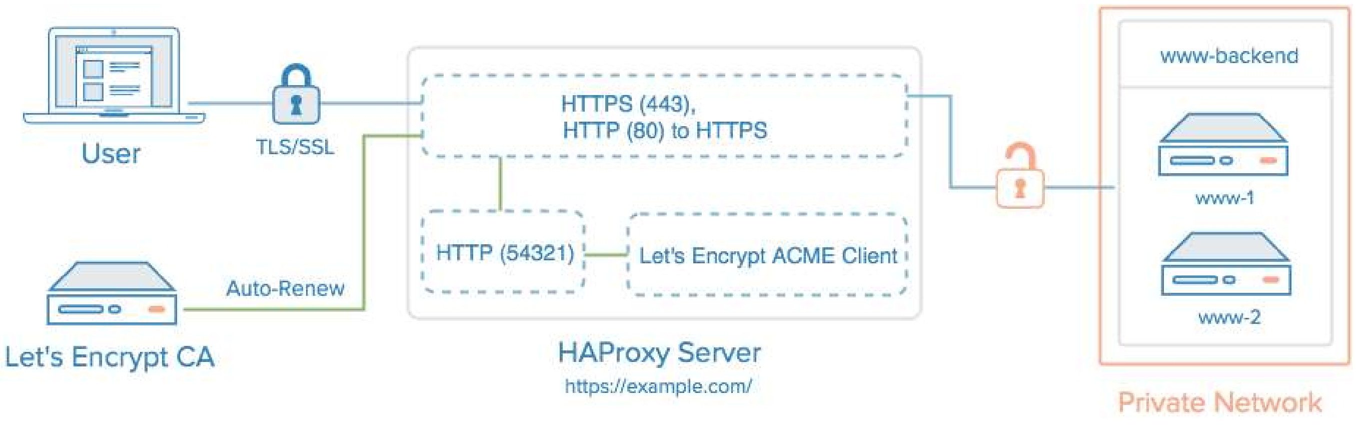
How To Secure HAProxy with Let's Encrypt on Ubuntu 14.04

Posted Jan 22, 2016 10 14.7k Let's Encrypt Security Load Balancing Ubuntu

Introduction

Let's Encrypt is a new Certificate Authority (CA) that provides an easy way to obtain and install free TLS/SSL certificates, thereby enabling encrypted HTTPS on web servers. It simplifies the process by providing a software client, `letsencrypt`, that attempts to automate most (if not all) of the required steps. Currently, as Let's Encrypt is still in open beta, the entire process of obtaining and installing a certificate is fully automated only on Apache web servers. However, Let's Encrypt can be used to easily obtain a free SSL certificate, which can be installed manually, regardless of your choice of web server software.

In this tutorial, we will show you how to use Let's Encrypt to obtain a free SSL certificate and use it with HAProxy on Ubuntu 14.04. We will also show you how to automatically renew your SSL certificate.



Prerequisites

Before following this tutorial, you'll need a few things.

You should have an Ubuntu 14.04 server with a non-root user who has `sudo` privileges. You can learn how to set up such a user account by following steps 1-3 in our initial server setup for Ubuntu 14.04.

You must own or control the registered domain name that you wish to use the certificate with. If you do not already have a registered domain name, you may register one with one of the many domain name registrars out there (e.g. Namecheap, GoDaddy, etc.).

If you haven't already, be sure to create an **A Record** that points your domain to the public IP address of your server. This is required because of how Let's Encrypt validates that you own the domain it is issuing a certificate for. For example, if you want to obtain a certificate for `example.com`, that domain must resolve to your server for the validation process to work. Our setup will use `example.com` and `www.example.com` as the domain names, so **both DNS records are required**.

Once you have all of the prerequisites out of the way, let's move on to installing the Let's Encrypt client software.

Step 1 — Install Let's Encrypt Client

The first step to using Let's Encrypt to obtain an SSL certificate is to install the `letsencrypt` software on your server. Currently, the best way to install Let's Encrypt is to simply clone it from the official GitHub repository. In the future, it will likely be available via a package manager.

Install Git and bc

Let's install Git and bc now, so we can clone the Let's Encrypt repository.

Update your server's package manager with this command:

```
$ sudo apt-get update
```

Then install the `git` and `bc` packages with apt-get:

```
$ sudo apt-get -y install git bc
```

With `git` and `bc` installed, we can easily download `letsencrypt` by cloning the repository from GitHub.

Clone Let's Encrypt

We can now clone the Let’s Encrypt repository in `/opt` with this command:

```
$ sudo git clone https://github.com/letsencrypt/letsencrypt /opt/letsencrypt
```

You should now have a copy of the `letsencrypt` repository in the `/opt/letsencrypt` directory.

Step 2 — Obtain a Certificate

Let's Encrypt provides a variety of ways to obtain SSL certificates, through various plugins. Unlike the Apache plugin, which is covered in a different tutorial, most of the plugins will only help you with obtaining a certificate which you must manually configure your web server to use. Plugins that only obtain certificates, and don't install them, are referred to as "authenticators" because they are used to authenticate whether a server should be issued a certificate.

We'll show you how to use the **Standalone** plugin to obtain an SSL certificate.

Verify Port 80 is Open

The Standalone plugin provides a very simple way to obtain SSL certificates. It works by temporarily running a small web server, on port `80`, on your server, to which the Let's Encrypt CA can connect and validate your server's identity before issuing a certificate. As such, this method requires that port `80` is not in use. That is, be sure to stop your normal web server, if it's using port `80` (i.e. `http`), before attempting to use this plugin.

For example, if you're using HAProxy, you can stop it by running this command:

```
$ sudo service haproxy stop
```

If you're not sure if port `80` is in use, you can run this command:

```
netstat -na | grep ':80.*LISTEN'
```

If there is no output when you run this command, you can use the Standalone plugin.

Run Let's Encrypt

Before using Let's Encrypt, change to the `letsencrypt` directory:

```
$ cd /opt/letsencrypt
```

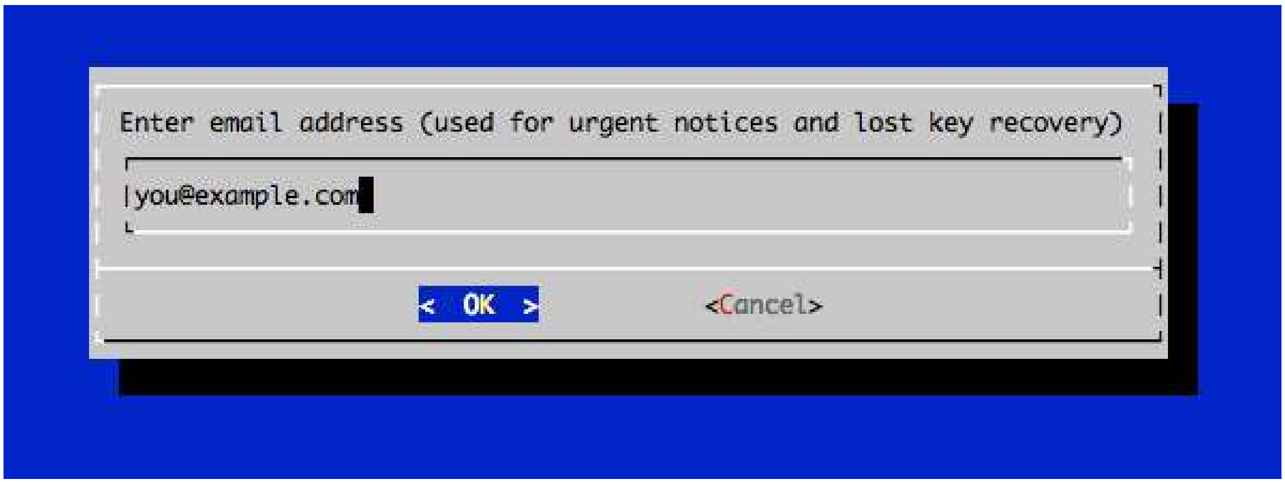
Now use the Standalone plugin by running this command:

```
$ ./letsencrypt-auto certonly --standalone
```

> **Note:** The Let's Encrypt software requires superuser privileges, so you will be required to enter your password if you haven't used `sudo` recently.

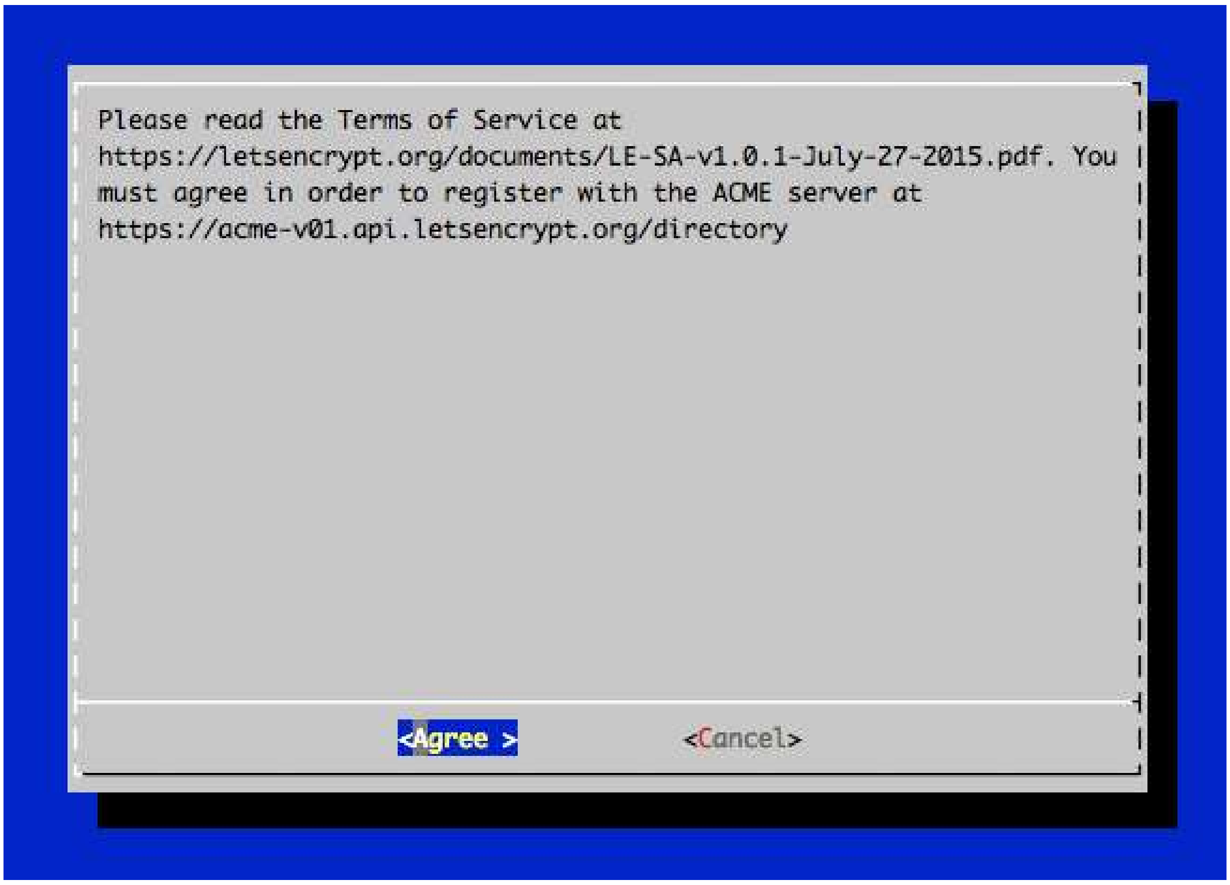
After `letsencrypt` initializes, you will be prompted for some information. This exact prompts may vary depending on if you've used Let's Encrypt before, but we'll step you through the first time.

At the prompt, enter an email address that will be used for notices and lost key recovery:



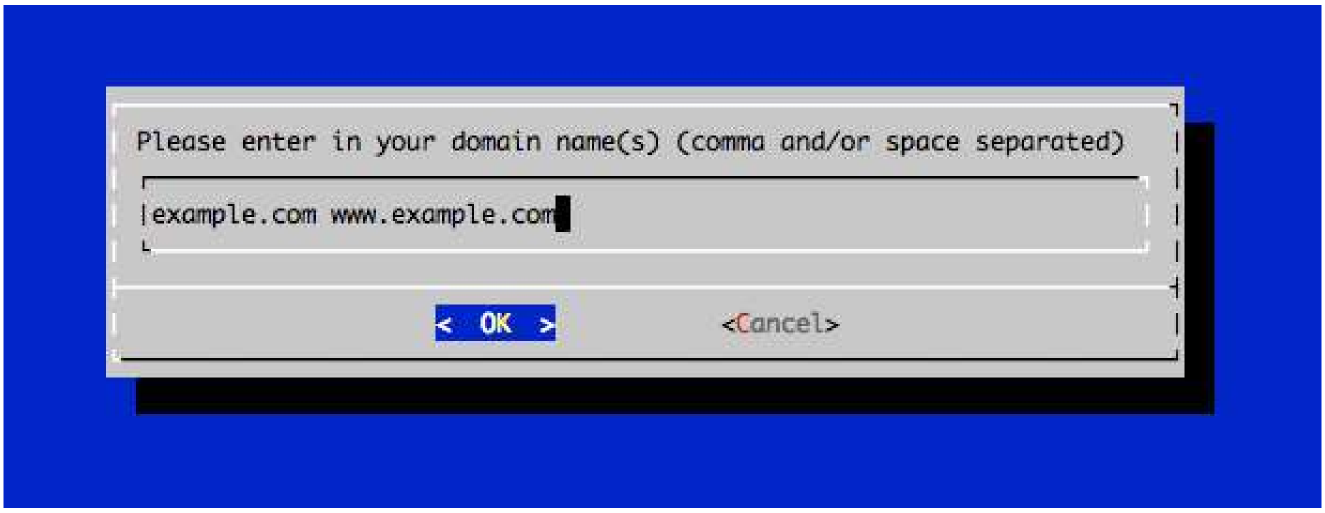
```
Enter email address (used for urgent notices and lost key recovery)

you@example.com
```

Then you must agree to the Let's Encrypt Subscribe Agreement. Select Agree:



```
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.0.1-July-27-2015.pdf. You
must agree in order to register with the ACME server at
https://acme-v01.api.letsencrypt.org/directory
```

Then enter your domain name(s). Note that if you want a single cert to work with multiple domain names (e.g. `example.com` and `www.example.com`), be sure to include all of them:



```
Please enter in your domain name(s) (comma and/or space separated)
example.com www.example.com
```

If everything was successful, you should see an output message that looks something like this:

Output:

```
IMPORTANT NOTES:
 - If you lose your account credentials, you can recover through
   e-mails sent to sammy@digitalocean.com
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/example.com/fullchain.pem. Your
   cert will expire on 2016-03-15. To obtain a new version of the
   certificate in the future, simply run Let's Encrypt again.
 - Your account credentials have been saved in your Let's Encrypt
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Let's
   Encrypt so making regular backups of this folder is ideal.
 - If like Let's Encrypt, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

You will want to note the path and expiration date of your certificate, which was highlighted in the example output.

> **Note:** If your domain is routing through a DNS service like CloudFlare, you will need to temporarily disable it until you have obtained the certificate.

Certificate Files

After obtaining the cert, you will have the following PEM-encoded files:

- **cert.pem:** Your domain's certificate
- **chain.pem:** The Let's Encrypt chain certificate
- **fullchain.pem:** `cert.pem` and `chain.pem` combined
- **privkey.pem:** Your certificate's private key

It's important that you are aware of the location of the certificate files that were just created, so you can use them in your web server configuration. The files themselves are placed in a subdirectory in `/etc/letsencrypt/archive`. However, Let's Encrypt creates symbolic links to the most recent certificate files in the `/etc/letsencrypt/live/your_domain_name` directory.

You can check that the files exist by running this command (substituting in your domain name):

```
$ sudo ls /etc/letsencrypt/live/your_domain_name
```

The output should be the four previously mentioned certificate files.

Combine Fullchain.pem and Privkey.pem

When configuring HAProxy to perform SSL termination, so it will encrypt traffic between itself and the end user, you must combine `fullchain.pem` and `privkey.pem` into a single file.

First, create the directory where the combined file will be placed, `/etc/haproxy/certs`:

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

Next, create the combined file with this `cat` command (substitute the highlighted `example.com` with your domain name):

```
'$DOMAIN/fullchain.pem /etc/letsencrypt/live/$DOMAIN/privkey.pem > /etc/haproxy/certs/$D
```

Secure access to the combined file, which contains the private key, with this command:

```
$ sudo chmod -R go-rwx /etc/haproxy/certs
```

Now we're ready to use the SSL cert and private key with HAProxy.

Step 3 — Install HAProxy

This step covers the installation of HAProxy. If it's already installed on your server, skip this step.

We will install HAProxy 1.6, which is not in the default Ubuntu repositories. However, we can still use a package manager to install HAProxy 1.6, if we use a PPA, with this command:

```
$ sudo add-apt-repository ppa:vbernat/haproxy-1.6
```

Update the local package index on your load balancers and install HAProxy by typing:

```
$ sudo apt-get update
$ sudo apt-get install haproxy
```

HAProxy is now installed but needs to be configured.

Step 4 — Configure HAProxy

This section will show you how to configure basic HAProxy with SSL setup. It also covers how to configure HAProxy to allow us to auto-renew our Let's Encrypt certificate.

Open `haproxy.cfg` in a text editor:

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

Keep this file open as we edit it in the next several sections.

Global Section

Let's add some basic settings under the `global` section.

The first thing you will want to do is set *maxconn* to a reasonable number. This affects how many concurrent connections HAProxy will allow, which can affect QoS and prevent your web servers from crashing from trying to serve too many requests. You will need to play around with it to find what works for your environment. Add the following line (with a value you think is reasonable) to the **global** section:

haproxy.cfg — 1 of 7

```
maxconn 2048
```

Next, add this line, to configure the maximum size of temporary DHE keys that are generated:

haproxy.cfg — 2 of 7

```
tune.ssl.default-dh-param 2048
```

Defaults Section

Add the following lines under the **defaults** section:

haproxy.cfg — 3 of 7

```
option forwardfor
option http-server-close
```

The forwardfor option sets HAProxy to add `X-Forwarded-For` headers to each request, and the `http-server-close` option reduces latency between HAProxy and your users by closing connections but maintaining keep-alives.

Frontend Sections

Now we're ready to define our `frontend` sections.

The first thing we want to add is a frontend to handle incoming HTTP connections, and send them to a default backend (which we'll define later). At the end of the file, let's add a frontend called **www-http**. Be sure to replace `haproxy_public_IP` with the public IP address of your HAProxy server:

haproxy.cfg — 4 of 7

```
frontend www-http
   bind haproxy_www_public_IP:80
   reqadd X-Forwarded-Proto:\ http
   default_backend www-backend
```

Next, we will add a frontend to handle incoming HTTPS connections. At the end of the file, add a frontend called **www-https**. Be sure to replace `haproxy_www_public_IP` with the public IP of your HAProxy server. Also, you will need to replace `example.com` with your domain name (which should correspond to the certificate file you created earlier):

haproxy.cfg — 5 of 7

```
frontend www-https
   bind haproxy_www_public_IP:443 ssl crt /etc/haproxy/certs/example.com.pem
   reqadd X-Forwarded-Proto:\ https
   acl letsencrypt-acl path_beg /.well-known/acme-challenge/
   use_backend letsencrypt-backend if letsencrypt-acl
   default_backend www-backend
```

This frontend uses an ACL (`letsencrypt-acl`) to send Let's Encrypt validation requests (for `/.well-known/acme-challenge`) to the `letsencrypt-backend` backend, which will enable us to renew the certificate without stopping the HAProxy service. All other requests will be forwarded to the `www-backend`, which is the backend that will serve our web application or site.

Backend Sections

After you are finished configuring the frontends, add the `www-backend` backend by adding the following lines. Be sure to replace the highlighted words with the respective private IP addresses of your web servers (adjust the number of `server` lines to match how many backend servers you have):

haproxy.cfg — 6 of 7

```
backend www-backend
   redirect scheme https if !{ ssl_fc }
   server www-1 www_1_private_IP:80 check
```

```
    server www-2 www_2_private_IP:80 check
```

Any traffic that this backend receives will be balanced across its `server` entries, over HTTP (port 80).

Lastly, add the `letsencrypt-backend` backend, by adding these lines

haproxy.cfg — 7 of 7

```
backend letsencrypt-backend
    server letsencrypt 127.0.0.1:54321
```

This backend, which only handles Let's Encrypt ACME challenges that are used for certificate requests and renewals, sends traffic to the localhost on port `54321`. We'll use this port instead of `80` and `443` when we renew our Let's Encrypt SSL certificate.

Now we're ready to start HAProxy:

```
$ sudo service haproxy restart
```

Note: If you're having trouble with the `haproxy.cfg` configuration file, check out this GitHub Gist for an example.

The Let's Encrypt TLS/SSL certificate is now in place, and we're ready to set up the auto-renewal script. At this point, you should test that the TLS/SSL certificate works by visiting your domain in a web browser.

Step 5 — Set Up Auto Renewal

Let's Encrypt certificates are valid for 90 days, but it's recommended that you renew the certificates every 60 days to allow a margin of error. At the time of this writing, automatic renewal is still not available as a feature of the client itself, but you can manually renew your certificates by running the Let's Encrypt client again.

A practical way to ensure your certificates won't get outdated is to create a cron job that will automatically handle the renewal process for you. In order to avoid the interactive, menu-driven process that we used earlier, we will use different parameters when calling the Let's Encrypt client in the cron job.

We will use the Standalone plugin used earlier, but configure it to use port `54321` so it doesn't conflict with HAProxy (which is listening on port `80` and `443`). To do so, we'll use this command (substituting in your domain name for both highlighted `example.com` domains):

```
$ cd /opt/letsencrypt
$ ./letsencrypt-auto certonly --agree-tos --renew-by-default --standalone-supported-c
```

Once that succeeds, you will need to create a new combined certificate file (replace `example.com` with your domain name):

```
$ DOMAIN='example.com' sudo -E bash -c 'cat /etc/letsencrypt/live/$DOMAIN/fullchain.p
```

Then reload HAProxy to start using the new certificate:

```
$ sudo service haproxy reload
```

Now that we know the commands that we need to renew our certificate, we can automate this process using scripts and a cron job.

Create a Let's Encrypt Configuration File

Before moving on, let's simplify our renewal process by creating a Let's Encrypt configuration file at `/usr/local/etc/le-renew-haproxy.ini`.

```
$ sudo cp /opt/letsencrypt/examples/cli.ini /usr/local/etc/le-renew-haproxy.ini
```

Now open the file for editing;

```
$ sudo nano /usr/local/etc/le-renew-haproxy.ini
```

Next, uncomment the `email` and `domains` lines, and update them with your own information. The file (with comments removed) should look something like this:

le-cli-example.com.ini — 1 of 2

```
rsa-key-size = 4096

email = you@example.com

domains = example.com, www.example.com
```

Next, uncomment the `standalone-supported-challenges` line, and replace its value with `http-01`. This tells Let's Encrypt to use It should look like this (with the changed value highlighted red):

le-cli-example.com.ini — 2 of 2

```
standalone-supported-challenges = http-01
```

Now, instead of specifying the domain names in the command, we can use the Let's Encrypt configuration file to fill in the blanks. Assuming your configuration file is correct, this command can be used to renew your certificate:

```
cd /opt/letsencrypt
./letsencrypt-auto certonly --renew-by-default --config /usr/local/etc/le-renew-hapro
```

Note: If you're having trouble with the `le-renew-haproxy.ini` configuration file, check out this GitHub Gist for an example.

Now let's create a script that we can use to renew our certificate.

Create a Renewal Script

To automate the renewal process, we will use a shell script that will verify the certificate expiration date for the provided domain and request a renewal when the expiration is less than 30 days away. This script will be scheduled to run once a week. This way, even if a cron job fails, there's a 30-day window to try again every week.

First, download the script and make it executable. Feel free to review the contents of the script before downloading it.

```
$ sudo curl -L -o /usr/local/sbin/le-renew-haproxy https://gist.githubusercontent.com
$ sudo chmod +x /usr/local/sbin/le-renew-haproxy
```

The `le-renew-haproxy` script takes as argument the domain name whose certificate you want to check for renewal. When the renewal is not yet necessary, it will simply output how many days are left until the given certificate expiration.

> **Note:** The script will not run if the `/usr/local/etc/le-renew-haproxy.ini` file does not exist. Also, be sure that the first domain that is specified in the configuration file is the same as the first domain you specified when you originally created the certificate.

If you run the script now, you will be able to see how many days are left for this certificate to expire:

```
$ sudo le-renew-haproxy
```

```
output
Checking expiration date for example.com...
The certificate is up to date, no need for renewal (89 days left).
```

Next, we will edit the crontab to create a new job that will run this command every week. To edit the crontab for the root user, run:

```
$ sudo crontab -e
```

Include the following content, all in one line:

crontab entry

```
30
```

Sa

co d

to

Conclusion

SCROLL TO TOP

That's it! HAProxy is now using a free Let's Encrypt TLS/SSL certificate to securely serve HTTPS traffic.

Author:
Mitchell Anicas

Related Tutorials

How To Set Up a Node.js Application for Production on Ubuntu 16.04

How To Secure Nginx with Let's Encrypt on Ubuntu 16.04

How To Migrate a Parse App to Parse Server on Ubuntu 14.04

How To Secure HAProxy with Let's Encrypt on CentOS 7

How To Secure Nginx with Let's Encrypt on CentOS 7

6 Comments

Leave a comment...

Logged in as: ✓ Notify me of replies to my comment Comment

mikey99 *March 9, 2016*

[https://support.cloudflare.com/hc/en-us/articles/214820528-How-to-Validate-a-Let-s-Encrypt-Certificate-on-a-Site-Already-Active-on-CloudFlare]

♡

samcoenen *March 26, 2016*

I've followed this DO tutorial:

how-to-create-a-high-availability-haproxy-setup-with-corosync-pacemaker-and-floating-ips

And now I have 4 droplets: 2 load balancers and 2 webservers. Now that I'm trying to follow this tutorial to add https to my website, it's failing at Step 2, where I need to run the :

`./letsencrypt-auto certonly --standalone` command.

The error ouput is **Failed authorization procedure. mydomain.com (tls-sni-01): urn:acme:error:connection :: The server could not connect to the client to verify the domain :: Failed to connect to host for DVSNI challenge** for every subdomain I'm trying to get a certificate for.

Are these tutorials incompatible with each other? I've set up my domain like it should be:

```
A NAME
@       MY_FLOATING_IP



CNAME
*         MY_DOMAIN.
```

And I can reach both my webservers via the domain. Of course, when I need to shut down haproxy as explained in this tutorial, I can't connect to them anymore.

Did I miss something? How should I make this work?

♡ 1

soufianedevc *April 18, 2016*

I have the same problem !
Did you found any solution for that ?

♡

samcoenen *April 18, 2016*

I'm afraid not.

I managed to get around the problem doing this:

1. Remove the HAProxy services
2. Install Nginx
3. Create certificates using Let's Encrypt
4. Remove Nginx again (the certificates will stay on the server)
5. Follow the tutorial like you would have starting from section 2. Part "Combine Fullchain.pem and Privkey.pem"

I know this probably wasn't the best way to get this to work, but I needed it fixed ASAP and this was the only thing I could think off. It worked like a charm this way, hopefully it can help you as well.

I used this to get the certs: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-14-04

♡

jpillora *April 29, 2016*

It'd be awesome if someone were to package this up into a Docker image!

♡

svh1985 *May 20, 2016*

Is there a way to request the new cert via port 443? http://domain.com/.well-known/acme-challenge/
Changing the --standalone-supported-challenges parameter from http-01 to tls-sni-01 doesen't work.

And I also recommend to add authenticator = standalone to the /usr/local/etc/le-renew-haproxy.ini

♡

Community Tutorials Questions Projects Tags RSS

Distros & One-Click Apps Terms, Privacy, & Copyright Security Report a Bug Get Paid to Write