



Home



Training



Playground



Quick Training



Hands-On Labs



Learning Paths



Community

[← NGINX Web Server Deep Dive](#)

Lecture: Use Case: Generating SSL Certificates using Let's Encrypt

[Flash Cards](#) [Notes](#) [Transcript](#) [Downloads](#)

Looks like you haven't created any flash cards for this deck yet.

[Create My First Flash Card](#)[Popular Decks](#) [Popular Cards](#)

Nginx Web Server Deep Dive

NGINX Web Server Deep Dive

NGINX Web Server Deep Dive

Until now, we've used a self-signed certificate and `example.com` based domains because there's no requirement to own a domain to learn NGINX. In this lesson, we'll go through setting up a real domain with an SSL certificate from Let's Encrypt. If you own a domain then you can follow along.

Note: the commands in this video are run as the `root` user.

Documentation For This Video

- [Let's Encrypt](#)
- [Certbot](#)

Setting Up The DNS Entry

This lesson I'll be using a domain that I own called `chord.tools`. Here are the DNS records that I'll be using. You can do something similar with your own domain:

NAME	TYPE	DATA
@	A	MY_PUBLIC_IP_ADDRESS
www	CNAME	chord.tools

Installing Certbot

The generation of SSL certificates using Let's Encrypt can be automated, and one of the best tools for doing so is `certbot`. With the `epel-release` already activated, we can install `certbot-nginx` which will give us `certbot` and a plugin for interacting with NGINX:

```
[root] $ yum install -y certbot-nginx
```

The `certbot` NGINX plugin will allow it to make modifications to our NGINX configuration for us and restart the server, but to make sure that it modifies the proper file, we'll create a simple configuration to start:

/etc/nginx/conf.d/chord.tools.conf

```
server {
    listen 80;
    server_name chord.tools www.chord.tools;
```

```
    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

Generating Certificates

There are a number of ways that we could use `certbot`, but for right now, we're going to keep it simple. Here's the command to have certbot generate certificates for our 2 domains:

```
[root] $ certbot --nginx -d chord.tools -d www.chord.tools
```

This will send us through a few different prompts, but after answering them all, we'll have a new configuration that will even force SSL automatically.

Here's what the final configuration looks like:

/etc/nginx/conf.d/chord.tools.conf

```
server {
    server_name chord.tools www.chord.tools;
```

```
    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

```
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/chord.tools/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/chord.tools/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
```

```
}
```

```
server {
    if ($host = www.chord.tools) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
if ($host = chord.tools) {
    return 301 https://$host$request_uri;
} # managed by Certbot
```

```
listen 80;
server_name chord.tools www.chord.tools;
return 404; # managed by Certbot
```

```
}
```

Renewing Certificates Automatically

Certificates generated by Let's Encrypt are valid for 90 days, so we'll need to renew them more often than if we paid a different certificate authority for a certificate. Thankfully, `certbot` once again has this covered. The `certbot renew` command will check which certificates on the server need to be renewed soon and fetch new ones. Behind the scenes, `certbot` has already created a `certbot-renew` systemd service and timer (found at `/lib/systemd/system/certbot-renew.{service,timer}`). We're going to make a small modification to the configuration file that the service uses (located at `/etc/sysconfig/certbot`) to make sure that NGINX is reloaded after a certificate is renewed.

Find the uncommented line containing `POST_HOOK` and set it to match this:

/etc/sysconfig/certbot

```
POST_HOOK="--post-hook 'systemctl reload nginx'"
```

Finally, let's enable the service and timer:

```
[root] $ systemctl start certbot-renew
[root] $ systemctl enable certbot-renew
[root] $ systemctl start certbot-renew.timer
[root] $ systemctl enable certbot-renew.timer
```

With this timer and service enabled, there will be a daily check to see if our certificate needs to be renewed. If the certificate does need renewed, `certbot` will fetch the new certificate(s) and reload the NGINX configuration automatically after the new certificates are downloaded.



Exceeded my Expectations



Room for Improvement

[← View Previous Lesson](#)[✓ Complete Section & Return to Course Module](#)