# Setup nginx for a glassfish app with ssl

Clément Levallois

2017-04-09

# Table of Contents

last modified: 2017-04-09

# System

- I use Debian, version 8.7 (why?)
- Vi is used as a text editor in the following

# Why nginx?

When I was introduced to nginx a long time ago, it was as a convenient replacement for Apache Web Server: lighter, faster.

Indeed, nginx can help you direct the traffic received by your domain www.example.com to the html / php / js files or Java apps sitting on your server.

But it can do more than being a web server. Nginx is a "reverse proxy", quite a barbaric term. I'd describe it as **a tool doing useful stuff between your firewall and your web app**. For security, it helps you with:

- dealing with SSL certificates
- help you mitigate DDOS attacks, and generally, controlling traffic in an easier way than using iptables
- hide the server from view

So I suppose you could do much of what nginx does via iptable and whatever app server you use, but concentring stuff in nginx makes it easier.

# why nginx + glassfish + ssl?

Why a tutorial specifically on nginx used for a glassfish app with ssl certificates?

- this is my personal use case
- there are no tutorials on it, though this file on github does 99% of the job (and we'll use it here).

So, this tutorial assumes you have created certificates with letsencrypt's certbot, as explained in a previous tutorial.

Let's start:

# installing nginx

```
apt-get install nginx
```

```
vi /etc/nginx/conf.d/yourdomain.com.conf
```

(note: the ".conf" extension is necessary for the file to be loaded in nginx)

Paste the content of this file in yourdomain.com.conf.

Explanations on the lines of this file:

```
upstream glassfish_server {
        server 127.0.0.1:8080 fail_timeout=0;
    }
```

This tells on which port the glassfish server can be reached. Remember that the default port 8080 can be changed to a more random port. See the glassfish installation guide in the same series of tutorials.

```
    server {
           listen         80;
           server_name    berat.com;
           return         301 https://$server_name$request_uri; ①
    }
```

① This means you can still be contacted through **http**, it will simply be redirected to **https**. Nice!

```
 ssl_certificate /var/certs/server.crt;
 ssl_certificate_key /var/certs/server.key;
```

For a ssl certificate created with letsencrypt, these lines should be changed for:

- ssl_certificate /etc/letsencrypt/live/yourdomain.com/cert.pem;
- ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;

That's it. Restart nginx to load your config:

/etc/init.d/nginx restart

Make sure the port 443 is open in your firewall.

Test your domain can be reached with SSL, with https and a nice green OK in the url bar: https://yourdomain.com

(don't write https://yourdomain.com:8080, it's silly but it blocked me a long time…)

# the end

Author of this tutorial: Clement Levallois

All resources on linux security: https://seinecle.github.io/linux-security-tutorials/