Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Documentation of Docker based HTTPS Reverse Proxy for Securing Web Services Project

Badr Ibrahim, Hossam Alzamly

30.01.18

Instructor ..... Prof. Dr. Martin Leischner

The following list contains all major section of the project.

Contents

1 Purpose

Throughout this documentation, it is shown how this project is used and what are the requirements and how to prepare your machine to start deploying this project. First off, the documentation starts explaining different project parts, then it proceeds with pre-deployment practices. Finally, it explains how to deploy the project throughly.

2 Project Parts

Since the project consists of different parts that are working together, it is important to recognize each part separately first before joining them together. In the following subsections, this is explained.

2.1 Nginx Reverse Proxy

Nginx is infamous web server and reverse proxy that is widely deployed and has huge support that made it fit in this project. The version used in the project is the latest, which can differ according to the time the project is deployed in. Below the main features and specifications of the used Nginx is listed.

- Nginx-full is used, which provides maximum features and modules pre-included in the installation as it is not known whether the environment, in which the project is deployed, is going to need which modules or features, so it is safer and more reliable this way.
- The configuration file lies at

  ```
  /etc/nginx/nginx.conf
  ```

  which can be loaded while running the project to ensure flexibility and customizability.

2.2 Docker

Docker is a popular containerization solution for services deployment. Docker provides extensible images which can be improved and customized. Below is the specifications used for Docker used in this project.

- Version used is 1.13.1, build 092cba3. However, it always is advised to install the latest version.
- Docker file is placed in the root working directory, such that all mount and copy points work properly. The root working directory contains the Nginx configuration file and one folder which is used for certificates storage and their configuration files.
- The OS used in the image is ubnutu 16.04 which can be referred to as ubuntu:latest. This points to the latest LTS version of ubuntu. More info are below.

  ```
  DISTRIB_ID=Ubuntu
  DISTRIB_RELEASE=16.04
  DISTRIB_CODENAME=xenial
  DISTRIB_DESCRIPTION="Ubuntu 16.04.3 LTS"
  ```

2.3 Certbot Client

Certbot is an automatic client that can be used to get and install certificates for servers from Let's Encrypt. One of its big features is the integration with Nginx, such that the process of installing and updating certificates is easy and fast. Specifications and the used features are shown below.

- Version used is 0.19.0 which is the latest version of certbot. Automatically, the latest version is installed every time for each build of the docker image.
- Docker file is placed in the root working directory, such that all mount and copy points work properly. The root working directory contains the Nginx configuration file and one folder which is used for certificates storage and their configuration files.
- Requesting certificates is done using Certbot's standalone plugin, as it is most suited for this scenario. Installing the certificate is done using the Certbot's nginx plugin which eases and automatically configure nginx.conf to use the certificate immediately.

```
$ ls
Dockerfile letsencrypt/ nginx.conf
```

3 Pre-Deployment

It takes simple command to deploy the containerized reverse proxy, but it may produce different issues if the host machine or the network is not properly configured. Next, the configurations and practices that need to be taken into consideration are discussed.

3.1 Host Machine

Host machine is where the container run above. Since Let's Encrypt, which where the certificate is going to be requested from, is going to be used, the host machine requires to have valid domain name. For the testing purposes Digital Ocean droplet was with the following OS specifications;

```
# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.3 LTS"
```

For giving a domain name there are free service that was used for also testing purposes which is Freedns. Another thing that has to be taken care on the host machine of is the firewall rules. For testing purposes the firewall was left disabled just as the default case.

```
# ufw status
Status: inactive
```

3.2 Network's Firewall

Sometimes the host machine lie behind a firewall, which can be very problematic if certain ports like 80, 443 or 53 are blocked. The reason for this is that the challenges performed by Let's Encrypt use these ports. Since ACME protocol is used to perform these challenges, these three ports must be available to be used in the communication between Let's Encrypt and the Certbot. This can be viewed in ACME specifications [1]. One thing to be mentioned is that tls-sni-01 challenge is currently disabled due to a security problem related to proving the control over domains [2].

3.3 Adding Nginx Configurations

Here, it is explained how to add the Nginx configuration file with clear example to follow. First, it should be noted that this step is totally left for the user to do and it is pretty much the only thing that should be provided at first time the container runs. The Nginx configuration file contains what is called Virtual Host. The Virtual Host represents certain domain names and the corresponding actions to calling them. Below the steps to add new Virtual Host is added.

- Inside a 'server' directive, add the server name to reference this block when it is called using HTTP or HTTPS, and do not add HTTP or HTTP at the beginning
- Add the port to listen to.
- Add the target IP or domain name, so the request is forwarded to it.

This makes almost everything ready. Note that other configurations regarding HTTPS is automatically added by Certbot while installing certificates, thanks to the Nginx installer plugin. Below, an abstract example is provided.

```
server {
     listen <port>;
     server_name <You Domain Name>;
     location / {
      proxy_pass <IP or Domain Name of the target>;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
     }
}
```

3.4 Certificate Renewal

Let's Encrypt can issue certificates that are valid for 90 days. Of course it is not wise not to include auto renewal especially that Certbot makes it easy to do so. Assuming that the folder to be mounted is named letsencrypt, then a file that holds general configurations for the certificate renewal and issuance can be created inside that folder. Below, that configuration file is shown:

```
$ cat cli.ini
#domains = domain1, domain2, ....etc.
```

```
renew-by-default = true
```

In this configuration file domain names can be provided so Certbot automatically configure their corresponding VirtualHosts in the Nginx configuration file and also for adding their configuration files about the renewal. Below, renewal prompt is shown when running a container that has already existing certificates:

```
Renewing an existing certificate
Performing the following challenges:
http-01 challenge for <domain-name>
Waiting for verification...
Cleaning up challenges
Running post-hook command: service nginx start
Output from service:
 * Starting nginx nginx
   ...done.

Deployed Certificate to VirtualHost /etc/nginx/nginx.conf for set(['<domain-name>'])
```

After making sure that everything is ready, the build phase is conducted.

```
$ docker build -t image:latest <path-to-Dockerfile>
```

4 Deployment

Here, everything is set and ready to be deployed using the built docker image. During the run phase, ACME challenges are performed and if successful, it asks about domains that need certificates to be issued. These domains are automatically detected from the Nginx configuration file. Running docker container is trivial but certain considerations must be taken care of to ensure the container running properly. First is to mount the folder that is used to store and provide certificates and configuration of the domain. Also, port mapping has to be done properly so challenges can be transverse to the container without any problems. Next command all these considerations are shown:

```
$ docker run -it -p 80:80 -p 443:443 $(pwd)/letsencrypt:/etc/letsencrypt
  -v $(pwd)/nginx.conf:/etc/nginx/nginx.conf image:latest
```

If your current directory is not like what shown previously in section 2.3, then you can do the following:

```
$ docker run -it -p 80:80 -p 443:443
-v <path>/letsencrypt:/etc/letsencrypt
<path>/nginx.conf:/etc/nginx/nginx.conf image:latest
```

After running the previous command, it will prompt the user asking about domains to select and also about whether to redirect HTTP request to HTTPS or not. If the user wants to use existing certificates, he can use the same command above, and cancels the prompt of the Certbot. Automatically, Nginx will start using the old certificates. Note that this needs certificates to exist and the Nginx is configured to use them at their path.

At this point, container is running normally and HTTPS is deployed if certificates are available and Nginx is properly configured.

A Appendix

Next, the Dockerfile can be found.

```
FROM ubuntu
RUN apt-get update && \
    apt-get -y install nginx-full && \
    apt-get -y install software-properties-common && \
    apt-get -y upgrade && \
    add-apt-repository ppa:certbot/certbot-build && \
    apt-get update && \
    apt-get -y install python-certbot-nginx && \
    apt-get autoremove -y && \
    apt-get clean -y
EXPOSE 80 443
CMD certbot --authenticator standalone --installer nginx \
    --pre-hook "service nginx stop" --post-hook "service nginx start" --keep \
    --config /etc/letsencrypt/cli.ini || \
    service nginx start && \
    bash
```

Next, example of the nginx ssl configuration is shown.

```
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/badronline.crabdance.com/fullchain.pem;
# managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/badronline.crabdance.com/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


if ($scheme != "https") {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```