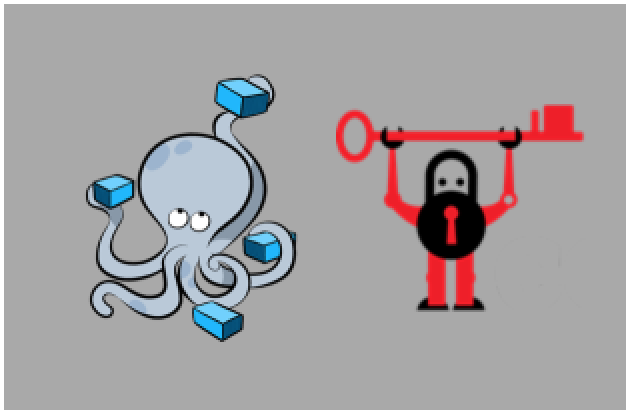

An Elegant way to use docker-compose to obtain and renew a Let's Encrypt SSL certificate with Certbot and configure the NGINX service to use it

9 April 2022 oussama Comments are off for this post.

Introduction

Docker and docker-compose provides an amazing way to quickly setup complicated applications that depends on several separate components running as services on a network.

This is evident in the amount of time and effort docker-compose spare when deploying a certain web-app like Rocket.Chat or Zammad on a new host.

Docker-compose allows for creating a single document to describe all standardised services needed for a web-app to run, to configure them, and to define how those services behave, and more. All that with minimal effort and with a predictable outcome every time.

(If you are reading this articles, you might also want to read the more recent post about the same topic…)

Context

Last night, I was working on a docker-compose.yaml file for the deploying and operating an instance of matomo, a great free and open-source web-app for web analytics. The work was straight forward at the beginning.

Matomo requires an SQL database such as MariaDB, an HTTP server such as NGINX, and the source code for the web-app. Because the team behind Matomo is great, and because they want to make life easier for open-source supporters like me, they have created an official docker image for their web-app and push it to https://hub.docker.com/_/matomo. Of course official images for MarianDB and NGINX also exists on the official docker repository, but you already know that.

The first version docker-compose.yaml file I drafted looked like this:

docker-compose.yaml:

```
version: "3"
services:
  db:
    image: mariadb
    container_name: db
    command: --max-allowed-packet=64MB
    restart: always
    environment:
      - MARIADB_DATABASE=matomo
      - MARIADB_USER
      - MARIADB_PASSWORD
      - MARIADB_ROOT_PASSWORD
    volumes:
      - /var/lib/mysql:/var/lib/mysql
  matomo:
    image: matomo
    container_name: matomo
    restart: always
    depends_on:
      - db
  nginx:
    container_name: nginx
    image: nginx:latest
    restart: unless-stopped
    environment:
      - DOMAIN
    depends_on:
      - matomo
    ports:
      - 80:80
    volumes:
```

And here are the additional auxiliary files needed for docker-compose to run all these services correctly:

.env:

```
MARIADB_USER=matomo
MARIADB_PASSWORD={{ANOTHER_STRONG_PASSWORD_HERE}}
MARIADB_ROOT_PASSWORD={{ANOTHE_RSTRONG_PASSWORD_HERE}
DOMAIN={{DOMAIN_NAME_HERE}}
```

./etc/nginx/templates/default.conf.template:

```
server {
    listen [::]:80;
    listen 80;

    server_name $DOMAIN;
    access_log  /var/log/nginx/access.log;
    error_log   /var/log/nginx/error.log;

    location / {
      proxy_set_header X-Forwarded-For $proxy_add_x_
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Proto http;
```

```
        proxy_pass http://matomo:80;
    }
```

Obviously this is a setup that does not support https as the configuraiton for the nginx service in the docker-compose.yaml does not expose port 443 not the nginx default configuraiton template default.conf.template defines a server listening to 443 and points the locations of the SSL certificate and private key.

To add support for https, we need a SSL certificate, and we need to configure nginx to use it, and we need to expose the port 443.

For this project I am using a free of charge SSL certificate from Let's Encrypt. I am using the certbot command line tool maintained by EFF to manage Let's Encrypt certificates (request, obtain, install, renew, revoke etc.)

The Certbot SSL certificate problem and a step-by-step solution:

Let us assume that I already had the SSL certificate. The docker-compose.yaml file will look like this:

./docker-compose.yaml

```
version: "3"
services:
  db:
    image: mariadb
    container_name: db
    command: --max-allowed-packet=64MB
    restart: always
    environment:
      - MARIADB_DATABASE=matomo
      - MARIADB_USER
      - MARIADB_PASSWORD
      - MARIADB_ROOT_PASSWORD
    volumes:
      - /var/lib/mysql:/var/lib/mysql
  matomo:
    image: matomo
    container_name: matomo
    restart: always
    depends_on:
      - db
  nginx:
    container_name: nginx
    image: nginx:latest
    restart: unless-stopped
    environment:
      - DOMAIN
    depends_on:
      - matomo
    ports:
      - 80:80
      - 443:443 # mapping port 443 to the container
    volumes:
      - ./etc/nginx/templates:/etc/nginx/templates:r
      - ./etc/letsencrypt:/etc/letsencrypt:ro # moun
```

and the nginx configuration file will look like this:

./etc/nginx/templates/default.conf.template:

```
server {
    listen [::]:80;
    listen 80;
    server_name $DOMAIN;
    return 301 https://$host$request_uri;
}

server {
    listen [::]:443 ssl http2;
    listen 443 ssl http2;
    server_name $DOMAIN;
    ssl_certificate /etc/letsencrypt/live/$DOMAIN/fu
    ssl_certificate_key /etc/letsencrypt/live/$DOMAI

    location / {
      proxy_set_header X-Forwarded-For $proxy_add_x_
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Proto https;
```

```
        proxy_pass http://matomo:80;
    }
```

This setup would work if the SSL certificate and key are already available to nginx in the /etc/letsencrypt folder. However this setup does not allow for the auto-renewal of the certificate and it does not address the problem of obtaining the certificate at the first place.

If we add a certbot service to the docker-compose.yaml file in order to renew the certificate the yaml file would look like this where the changes has been highlighted:

```
version: "3"
services:
  db:
    image: mariadb
    container_name: db
    command: --max-allowed-packet=64MB
    restart: always
    environment:
      - MARIADB_DATABASE=matomo
      - MARIADB_USER
      - MARIADB_PASSWORD
      - MARIADB_ROOT_PASSWORD
    volumes:
      - /var/lib/mysql:/var/lib/mysql
  matomo:
    image: matomo
    container_name: matomo
    restart: always
    depends_on:
      - db
  nginx:
    container_name: nginx
    image: nginx:latest
    restart: unless-stopped
    environment:
      - DOMAIN
    depends_on:
      - matomo
    ports:
      - 80:80
      - 443:443 # mapping port 443 to the container
    volumes:
      - ./etc/nginx/templates:/etc/nginx/templates:r
      - ./etc/letsencrypt:/etc/letsencrypt:ro # moun
      - ./certbot/data:/var/www/certbot
  certbot:
    container_name: certbot
    image: certbot/certbot:latest
    depends_on:
      - nginx
    command: >-
             certonly --reinstall --webroot --webroo
             --email ${EMAIL} --agree-tos --no-eff-e
             -d ${DOMAIN}
    volumes:
      - ./etc/letsencrypt:/etc/letsencrypt
      - ./certbot/data:/var/www/certbot
```

and accordingly, the ./etc/nginx/templates/default.conf.template will change to include the location of the certbot challenge folder as highlighted in:

```
server {
    listen [::]:80;
    listen 80;
    server_name $DOMAIN;
    return 301 https://$host$request_uri;
}

server {
    listen [::]:443 ssl http2;
    listen 443 ssl http2;
    server_name $DOMAIN;

    ssl_certificate /etc/letsencrypt/live/$DOMAIN/fu
    ssl_certificate_key /etc/letsencrypt/live/$DOMAI

    location ~ /.well-known/acme-challenge {
        allow all;
        root /var/www/certbot;
    }

    location / {
      proxy_set_header X-Forwarded-For $proxy_add_x_
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Proto https;
      proxy_pass http://matomo:80;
  }
```

With this setup, certbot will be called on docker-compose up, it will then attempt to renew the certificate. If it succeeds the certificate will be stored in the /etc/letsencrypt/live folder, then the certbot service container will exist and won’t start again until a specific command is trigger to start the renewal process again. The command is:

```
$ docker-compose -f /root/matomo/docker-compose.yaml
```

If certbot suceeded in obtaining a new cert and key, nginx needs to reload the configurations to make those changes effective. So it is necessary to force nginx to reload the configurations:

```
$ docker-compose -f /root/matomo/docker-compose.yaml
```

Now let’s get back to the remaining issue: Obtaining the SSL certificate with the container of certbot. This is often referred to as the “Egg and Chicken”

[illegible]

will fail because of the ssl_certificate, and ssl_certificate_key pointing to a non-existing location.

So the main question becomes, how to get the SSL certificate in an automated way that does not requires long complicated scripts.

The solution I opted for is to have a separate docker-compose setup just to obtain a valid certificate the first time, solving the chicken and egg problem. This separate initiation setup will obtain the certificate and place it in a folder accessible to the ordinary docker-compose setup discussed above.

The solution is described in two phases:

Phase 1:

- run docker-compose up with the initiation configuration file
- obtain a certificate using Certbot and store it in a folder on the host system
- run docker-compose down to finish the initiation phase

Phase 2:

- create a cron job for renewing the certificate with Certbot and reloading NGINX
- run docker-compose up -d with the web-app configuration file

In the following the files structure used for this solution and a listing of all configuration files required for phase 1, and phase 2.

The complete solution files and content:

```
~/matomo# tree .
  .
├── etc
  |   ├── crontab
  |   └── nginx
  |       ├── templates
  |        |   └── default.conf.template
  |       └── templates-initiate
  |           └── default.conf.template
├── docker-compose-initiate.yaml
├── docker-compose.yaml
├── cron.sh
└── install.sh
```

./install.sh:

```
#!/bin/bash
# takes two paramters, the domain name and the emai
```

```
DOMAIN=$1
EMAIL=$2

echo MARIADB_USER=matomo > .env
echo MARIADB_PASSWORD=`openssl rand 30 | base64 -w (
echo MARIADB_ROOT_PASSWORD=`openssl rand 30 | base64
echo DOMAIN=${DOMAIN} >> .env
echo EMAIL=${EMAIL} >> .env

# Phase 1
docker-compose -f ./docker-compose-initiate.yaml up
docker-compose -f ./docker-compose-initiate.yaml up
docker-compose -f ./docker-compose-initiate.yaml dow

# some configurations for let's encrypt
curl -L --create-dirs -o etc/letsencrypt/options-ssl
openssl dhparam -out etc/letsencrypt/ssl-dhparams.pe

# Phase 2
crontab ./etc/crontab
docker-compose -f ./docker-compose.yaml -d up
```

```
version: "3"
services:
  nginx:
    container_name: nginx
    image: nginx:latest
    environment:
      - DOMAIN
    ports:
      - 80:80
    volumes:
      - ./etc/nginx/templates-initiate:/etc/nginx/te
      - ./etc/letsencrypt:/etc/letsencrypt
      - ./certbot/data:/var/www/certbot
  certbot:
    container_name: certbot
    image: certbot/certbot:latest
    depends_on:
      - nginx
    command: >-
             certonly --reinstall --webroot --webroo
             --email ${EMAIL} --agree-tos --no-eff-e
             -d ${DOMAIN}
    volumes:
      - ./etc/letsencrypt:/etc/letsencrypt
```

./etc/nginx/templates-initiate/default.conf.template:

```
server {
    listen [::]:80;
    listen 80;
    server_name $DOMAIN;
    location ~/.well-known/acme-challenge {
        allow all;
        root /var/www/certbot;
    }
}
```

./docker-compose.yaml

```
version: "3"
services:
  db:
    image: mariadb
    container_name: db
    command: --max-allowed-packet=64MB
    restart: always
    environment:
      - MARIADB_DATABASE=matomo
      - MARIADB_USER
      - MARIADB_PASSWORD
      - MARIADB_ROOT_PASSWORD
    volumes:
      - ./db:/var/lib/mysql
  matomo:
    image: matomo
    container_name: matomo
    restart: always
    depends_on:
      - db
    volumes:
      - ./matomo:/var/www/html
  nginx:
    container_name: nginx
    image: nginx:latest
    restart: unless-stopped
    environment:
      - DOMAIN
    depends_on:
      - matomo
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./etc/nginx/templates:/etc/nginx/templates:r
      - ./etc/letsencrypt:/etc/letsencrypt:ro
      - ./certbot/data:/var/www/certbot
  certbot:
    container_name: certbot
    image: certbot/certbot:latest
    depends_on:
      - nginx
    command: >-
             certonly --reinstall --webroot --webroc
             --email ${EMAIL} --agree-tos --no-eff-e
             -d ${DOMAIN}
    volumes:
      - ./etc/letsencrypt:/etc/letsencrypt
```

./etc/nginx/templates/default.conf.template:

```
server {
    listen [::]:80;
    listen 80;
    server_name $DOMAIN;
    return 301 https://$host$request_uri;
}

server {
    listen [::]:443 ssl http2;
    listen 443 ssl http2;
    server_name $DOMAIN;

    ssl_certificate /etc/letsencrypt/live/$DOMAIN/fu
    ssl_certificate_key /etc/letsencrypt/live/$DOMAI
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location ~ /.well-known/acme-challenge {
        allow all;
        root /var/www/certbot;
    }

    location / {
      proxy_set_header X-Forwarded-For $proxy_add_x_
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Proto https;
```

```
        proxy_pass http://matomo:80;
    }
```

cron_job.sh:

```
#!/bin/bash
# cleanup exited docker containers
EXITED_CONTAINERS=$(docker ps -a | grep Exited | awk
if [ -z "$EXITED_CONTAINERS" ]
then
        echo "No exited containers to clean"
else
        docker rm $EXITED_CONTAINERS
fi

# renew certbot certificate
docker-compose -f /root/matomo/docker-compose.yaml r
docker-compose -f /root/matomo/docker-compose.yaml e
```

./etc/crontab:

```
# m h  dom mon dow   command
0 5  * * *  /root/matomo/cron_job.sh
```

That's it.

Tagged as: certbot, DevOps, Docker, docker-compose, How To, Let's Encrypt, matomo, Nginx

Categorized in: DevSecOps, HowTo, Open Source

Decode Theme by Macho Themes