Heziode

Zeste de savoir

Traefik v2 HTTPS (SSL) en localhost

17 septembre 2020

Table des matières

Souhaitant pouvoir bénéficier d'un reverse-proxy avec support du SSL lors de mes développements (donc sur une machine en local), je me suis heurté à quelques petites difficultés sur la configuration de Traefik. On trouve en effet, facilement des ressources pour la première version de Traefik, mais depuis les 26 août, Traefik est passé en 2.X.X. Ainsi, les configurations de la 1.X.X ne sont plus entièrement applicables pour cette nouvelle version.

Ce billet a pour but de te fournir une configuration minimale te permettant de pouvoir bénéficier du SSL en local avec Traefik, en utilisant Docker.

TL ;DR:

Un petit projet minimal est disponnible sur mon GitHub .

1. Un certificat auto-signé

Pour avoir dur SSL en localhost, il n'est pas possible d'utiliser let's encrypt. Il faudra passer par un certificat auto-signé. Pour faciliter tout cela, tu peux utiliser mkcert , un utilitaire permettant de créer des certificats auto-signés pour ses domaines et de les approuver pour ne pas avoir le vilain message des navigateurs.

Une fois installé, il faut exécuter la commande suivante pour créer une autorité de confiance local, qui nous servira pour signer nos certificats:

```
mkcert -install
```

Maintenant, imaginons que nous souhaitons avoir des certificats pour `docker.localhost` et `domain.local` (et leurs sous-domaines), nous pouvons généré un des certificats signé avec notre autorité de confiance local ainsi:

```
# On créer un répertoire qui va contenir nos certificats
mkdir certs

# Puis on créer les certificats
```

```
5 mkcert -cert-file certs/local-cert.pem -key-file
      certs/local-key.pem "docker.localhost" "*.docker.localhost"
      "domain.local" "*.domain.local"
```

Note: pour les certificats wildcards (des sous-domaines), il ne support que le niveau de wildcard que l'on met. Autrement dit, si je prends par exemple `*.docker.localhost`, le certificat fonctionnera pour `foo.docker.localhost`, `bar.docker.localhost`, etc. mais pas pour `foo.bar.docker.localhost`.

Nous avons maintenant des certificats prêts à l'emploi pour notre dev.

2. Configuration de Traefik

À présent, passons à la configuration de Traefik. Ici rien de bien compliquer, il suffit de bien lire la doc . Pour t'éviter trop de lecture, voici un résumé.

Traefik 2 fonctionne avec un système de configuration statique et dynamique. La configuration statique est chargée au démarrage, et tout changement dans cette configuration nécessitera un redémarrage de Traefik. La configuration dynamique quant à elle permet d'être changé à chaud.

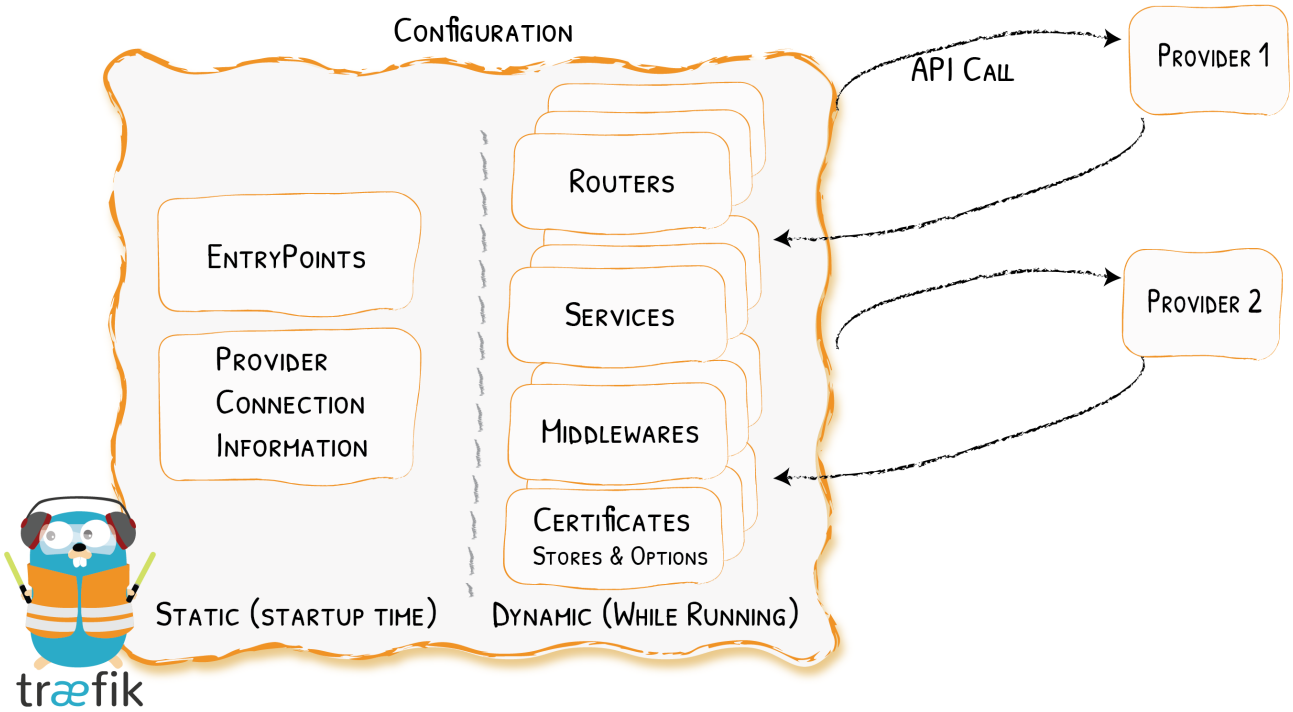

Figure 2.1. – Configuration statiquie et dynamique

Source de l'image.

? Qu'est-ce qui change par rapport à la V1?

Dans le guide de migration il est dit que la configuration du SSL est déplacée de l'*entrypoint* au *router*.

Voici la configuration statique:

```
# On créer un répertoire qui va contenir nos certificats
mkdir certs

# Puis on créer les certificats
mkcert -cert-file certs/local-cert.pem -key-file
    certs/local-key.pem "docker.localhost" "*.docker.localhost"
    "domain.local" "*.domain.local"
```

Comme on peut le constater, on force une redirection de HTTP sur HTTPS.

Au niveau de la configuration dynamique nous avons ceci:

```
# On créer un répertoire qui va contenir nos certificats
mkdir certs

# Puis on créer les certificats
mkcert -cert-file certs/local-cert.pem -key-file
    certs/local-key.pem "docker.localhost" "*.docker.localhost"
    "domain.local" "*.domain.local"
```

Ici, on spécifie le certificat à utiliser pour le SSL, et on rajoute la configuration d'un router nommé "traefik". Cela évitera d'avoir à le faire dans le `docker-compose.yml`

3. Configuration des conteneurs avec docker-compose

Maintenant qu'on a des certificats auto-signés et qu'on a configuré Traefik, il est temps de s'attaquer à la configuration de nos conteneurs pour qu'ils puissent être accessible depuis Traefik.

Dans un premier temps, nous allons créer un réseau docker pour notre proxy :

```
docker network create proxy
```

Je me suis basé sur l'exemple de base de la doc de Traefik pour l'exemple:

```yaml
# docker-compose.yml
version: '3'

services:
  reverse-proxy:
    image: traefik:v2.3
    container_name: traefik
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    ports:
      # Web
      - 80:80
      - 443:443
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      # On map la conf statique dans le conteneur
      - ./traefik/traefik.yml:/etc/traefik/traefik.yml:ro
      # On map la conf dynamique statique dans le conteneur
      - ./traefik/config.yml:/etc/traefik/config.yml:ro
      # On map les certificats dans le conteneur
      - ./certs:/etc/certs:ro
    networks:
      - proxy
    labels:
      # Permettre à ce conteneur d'être accessible par traefik
      # Pour plus d'information, voir :
          https://docs.traefik.io/providers/docker/#exposedbydefault
      - "traefik.enable=true"
      # Utilise la configuration du routeur "traefik" définie dans
          le fichier de configuration dynamique :
          ./traefik/config.yml
      - "traefik.http.routers.traefik=true"

  whoami:
    image: containous/whoami
    container_name: whoami
    security_opt:
      - no-new-privileges:true
    labels:
      - "traefik.enable=true"
      # URL pour accéder à ce conteneur
      -
          "traefik.http.routers.whoami.rule=Host(`whoami.docker.localhost`)"
      # Activation de TLS
      - "traefik.http.routers.whoami.tls=true"
      # Si le port est différent de 80, utilisez le service
          suivant:
```

```
      # -
          "traefik.http.services.<service_name>.loadbalancer.server.port=<port>
    networks:
      - proxy

networks:
  proxy:
    external: true
```

Il suffit ensuite de démarrer nos conteneurs avec `docker-compose -f docker-compose.yml up` et le tour est joué!

Vous pouvez accéder au dashboard de Træfik via l'URL suivante: traefik.docker.localhost .

Et vous pouvez accéder à l'example (whoami) via l'URL suivante: whoami.docker.localhost

---

C'est tout pour ce billet. De ce simple exemple tu pourras ensuite y intégrer toute ta constellation de conteneurs sans trop de difficulté.

Et garder à l'esprit que vous pouvez également mapper du TCP et de l'UDP avec Træfik.