TUTOS-INFO
L'informatique par l'exemple

TUTORIEL

# TRAEFIK

Le « modern reverse-proxy » français

Version 3.2.2

# SOMMAIRE


1. QU'EST-CE QUE TRAEFIK ?
2. FONCTIONNEMENT DE TRAEFIK 3.2 (VOCABULAIRE)
3. INSTALLATION DE TRAEFIK 3.2 AVEC DOCKER COMPOSE
    a. Création du fichier de configuration statique
    b. Création des fichiers de configuration dynamiques
4. LANCEMENT DE TRAEFIK 3.2 AVEC DOCKER COMPOSE
    a. Création de la stack avec Docker Compose (avec ajout de labels)
    b. Test d'accès sécurisé au tableau de bord de Traefik




EXPERT

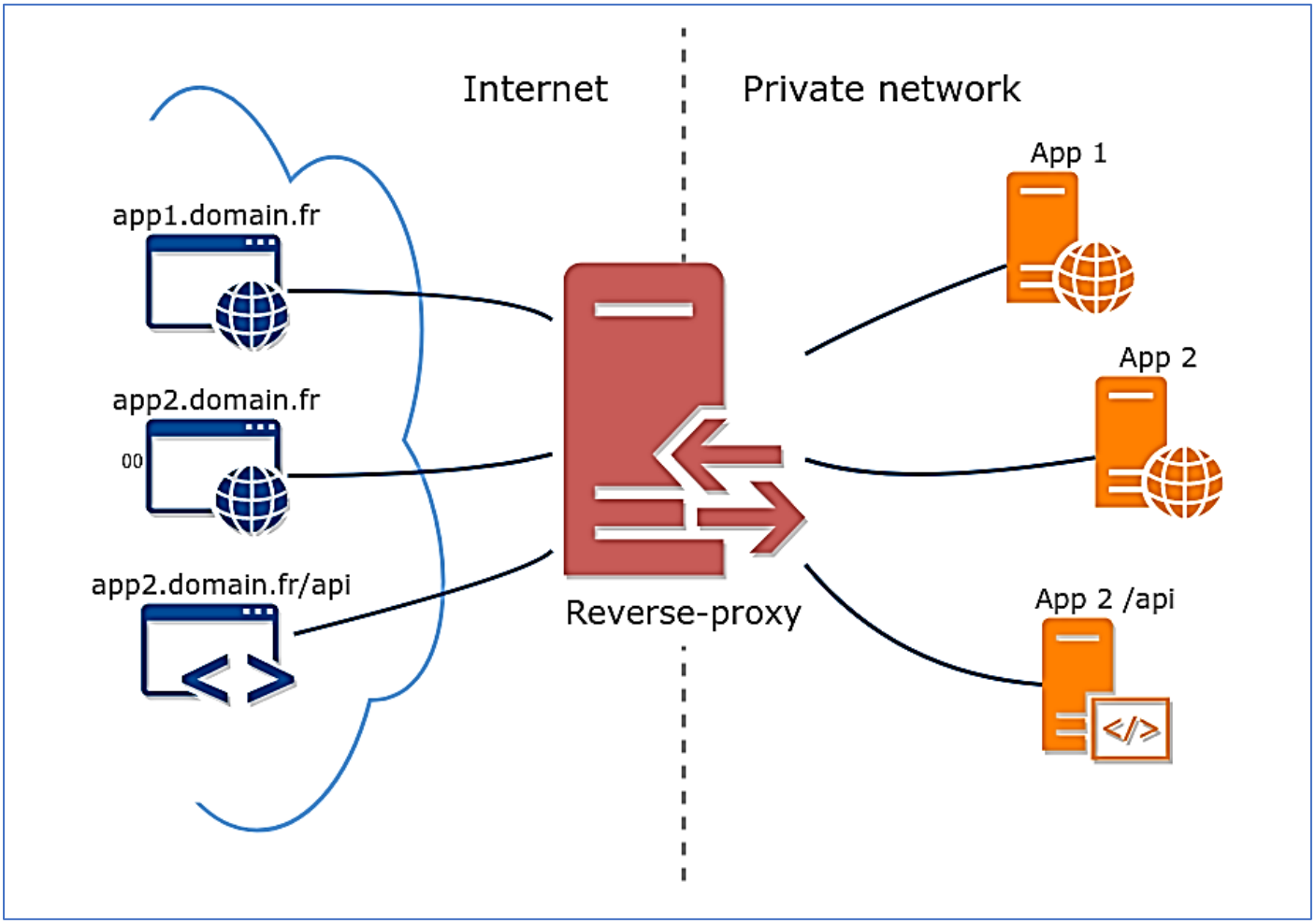
DIFFICULTE

# 1 – QU'EST-CE QUE TRAEFIK ?

**Traefik** est un outil écrit en Golang par Emile Vauge et c'est français ! Traefik est un **reverse-proxy**.

Un reverse-proxy est une brique de notre infrastructure, qui permet d'être un intermédiaire de communication, entre un réseau public et un réseau privé, le nôtre par exemple. C'est sur ce réseau privé que l'on trouvera toutes les applications de notre système informatique qui ne sont pas accessibles depuis l'extérieur pour des raisons de sécurité principalement.

Le reverse-proxy doit donc connaître toutes les « routes » disponibles pour **transférer chaque requête vers le bon service**. Comme il est le point central du trafic, il peut également faire du « load balancing », ou encore appliquer des règles de sécurité comme le HTTPS (via Let's Encrypt par exemple).




La véritable **force de traefik** c'est que **sa configuration est dynamique** : il va se connecter au socket Docker et récupérer toutes les informations sur les containers qui tournent en temps réel de façon à pouvoir router les requêtes sur ces derniers de manière automatique ! Traefik supporte plusieurs algorithmes de répartition des charges et gère :

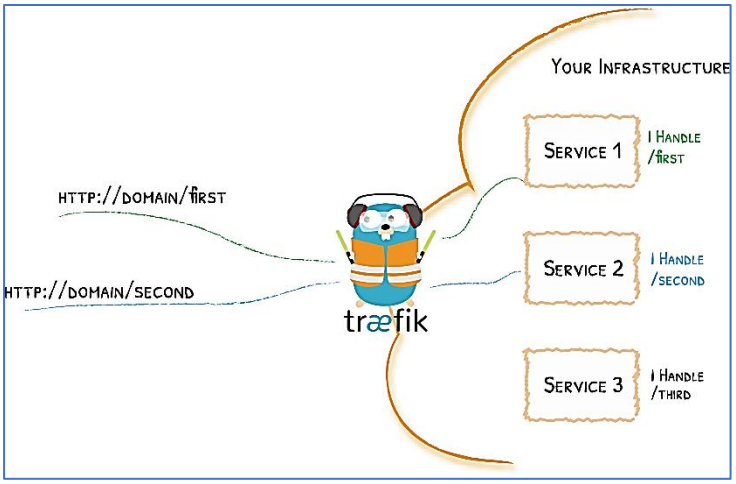
- le HTTPS/TLS et la génération des certificats via Let's encrypt (plus besoin de Certbot)
- le HTTP et le HTTP/2 mais aussi Websocket, GRPC, TCP et UDP
- une interface web pour surveiller l'état des services
- une exposition de métriques vers Prometheus et la rétention des logs d'accès

A ce jour (décembre 2024), la **dernière version en date stable** de Traefik est la **3.2.2**

# 2 – PRINCIPE DE FONCTIONNEMENT DE TRAEFIK

Lorsqu'un utilisateur envoie une requête HTTP à Traefik, celle-ci contient toutes les informations nécessaires pour indiquer à Traefik sa destination. En général, on utilise le nom de domaine et le chemin de la requête pour déterminer sa destination.

**Traefik analyse la requête** et cherche dans sa liste de « routes » si l'une d'elles peut correspondre. Si c'est le cas **il va transférer la requête vers son « service » associé**. Dans le cas contraire il va retourner à l'utilisateur une « erreur 404 ».

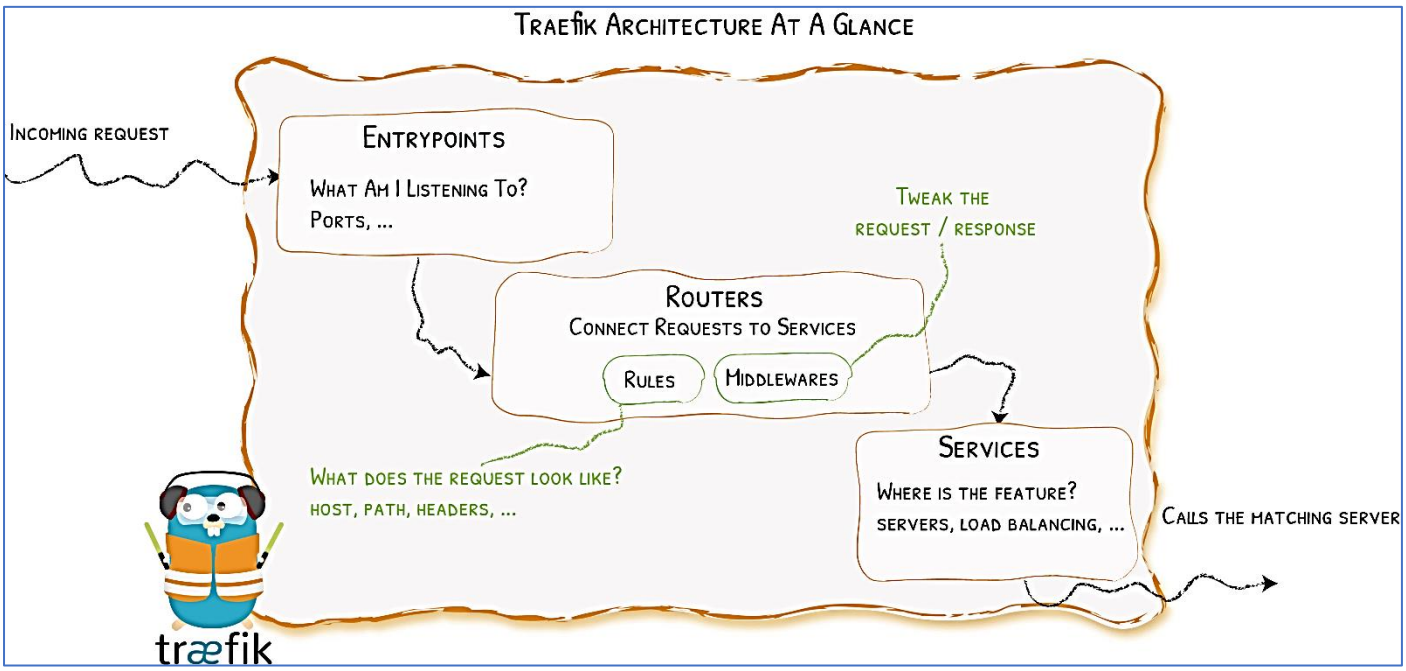



Pour fonctionner, Traefik utilise 2 types de configurations qui sont importantes à connaître :

- la configuration dite **STATIQUE** (qui nécessitera un redémarrage de l'outil) ;
- la configuration dite **DYNAMIQUE** (qui permettra d'appliquer des modifications « à la volée ».

*Les procédures qui vont suivre nécessitent que Docker et Docker Compose soient préalablement installés sur votre machine Debian (voir tutoriels précédents sur https://cloud/tutos-info.fr).*

### VOCABULAIRE TRAEFIK

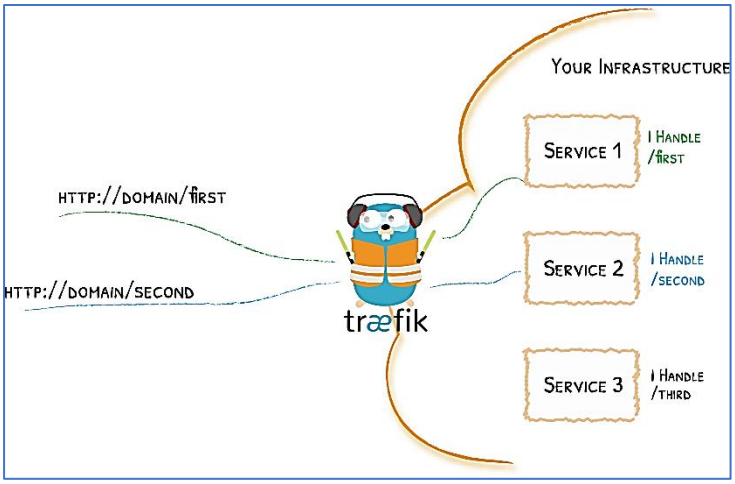



Les **ENTRYPOINTS** **sont les points d'entrées**. Il s'agit des adresses et ports exposés par Traefik qui permettront d'exposer nos applications. La plupart du temps, ce sont les **ports 443 et 80** qui sont exposés par défaut.

Les **ROUTERS** **gèrent les règles de redirection** du reverse. Un routeur pointe un service. On affecte au routeur des URL, des entrypoints.

Les **SERVICES** **sont les applications**, par exemple Nextcloud. Nous pouvons recenser 3 types de services : HTTP, TCP et UDP.

Les **PROVIDERS** **sont les sources de génération de la configuration**. C'est ce qui fait la force de Traefik. Il existe plusieurs types de « **providers** » parmi lesquels :




- Moteur de conteneur :
  - **Docker** : Traefik obtient l'accès au socket de docker et pourra gérer les services (conteneurs).
  - **Kubernetes**
  - **Rancher**
  - **Marathon**
- Base de clé/valeur :
  - **consul**
  - **consulcatalog**
  - **etcd**
  - **redis**
  - **zookeeper**
- **file** (ce provider permet d'indiquer un chemin vers un fichier de configuration par exemple)

Dans ce guide, nous nous contenterons d'étudier les providers « Docker » et « file ».

Les **MIDDLEWARES** **sont des étapes intermédiaires entre le router et le service**. Le « **middleware** » peut être utilisé pour faire de la **compression**, de la **sécurisation**, de l'**authentification** à un tableau de bord par exemple.

## 3 – INSTALLATION DE TRAEFIK 3.2 AVEC DOCKER COMPOSE

On commence par créer l'arborescence de notre environnement de travail. Afin de mieux comprendre le fonctionnement général, nous allons travailler dans un dossier « **srv** » situé à la racine de la machine Debian.

Organisation des dossiers sur la machine Debian :

| | |
|---|---|
| ***/srv/traefik*** | Contiendra le fichier « **docker-compose.yml** » qui lancera la stack Traefik |
| ***/srv/traefik/static*** | Contiendra les fichiers « **traefik.yml** » |
| ***/srv/traefik/dynamic*** | Contiendra les fichiers relatifs aux « **middlewares** » (TLS, Headers et Auth) |
| ***/srv/traefik/logs*** | Contiendra les fichiers de logs de Traefik (traefik_error et traefik_access) |
| ***/srv/traefik/certs*** | Contiendra les fichiers liés à Let's Encrypt (certificats OpenSSL) |

```
.
├── certs
│   └── acme.json
├── docker-compose.yml
├── dynamic
│   ├── auth.yml
│   ├── headers.yml
│   └── tls.yml
├── logs
│   ├── traefik_access.log
│   └── traefik_error.log
└── static
    └── traefik.yml
```

ARBORESCENCE COMPLETE DE NOTRE ENVIRONNEMENT TRAEFIK 2.9.10

On commence par la création de l'ensemble des dossiers nécessaires dans le dossier « home » de l'utilisateur et à l'aide des commandes suivantes :

***cd /***
***mkdir -p /srv/traefik /srv/traefik/static /srv/traefik/dynamic /srv/traefik/logs /srv/traefik/certs***
***touch /srv/traefik/logs/traefik_error.log***
***touch /srv/traefik/logs/traefik_access.log***

On peut installer, ici, le paquet « **apache2-utils** » sur notre machine Debian. Cet outil nous permettra de générer des mots de passe cryptés (nous en aurons besoin ultérieurement). Pour cela, exécutez la commande suivante :

***apt install apache2-utils -y***

On créé également un réseau dédié à Traefik, que l'on nommera « **traefik-proxy** », avec la commande suivante :

***docker network create traefik-proxy***

Un fichier vierge « **acme.json** » permettant de stocker les futurs certificats Let's Encrypt doit être créé avec les droits « rw », c'est-à-dire « 600 » en octal. **Il est impératif que ce fichier existe avant le lancement de la stack**. Dans le cas contraire, un dossier serait créé à sa place et génèrerait une erreur.

On créé ce fichier dans le dossier « ***/srv/traefik/certs*** » avec la commande suivante :

***touch /srv/traefik/certs/acme.json***
***chmod 600 /srv/traefik/certs/acme.json***

a) Création du **FICHIER DE CONFIGURATION STATIQUE** de Traefik

On commence par créer un fichier de **configuration statique** nommé « **traefik.yml** » dans notre environnement de travail **« *srv/traefik/static* »** :

***cd /srv/traefik/static***
***nano traefik.yml***

L'éditeur nano ouvre une fenêtre vide dans laquelle nous allons saisir un ensemble de d'arguments dits « statiques ». **Le fichier de configuration statique est le premier fichier exécuté par Traefik** et **il contient des éléments qui ne sont pas amenés à être modifiés par la suite** (sinon il faudra redémarrer Traefik en cas de modification).

Contenu du fichier de configuration statique « **traefik.yml** » (attention, indiquez votre mail pour Let's Encrypt) :

```
api:
  dashboard: true
  debug: true

entryPoints:
  http:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: https
          scheme: https
  https:
    address: ":443"
```

Ici on active l'accès au tableau de bord de Traefik.

Ici on active les **ENTRYPOINTS** (points d'entrée), en l'occurrence le port http « 80 » et on active la redirection automatique, à l'aide d'un **MIDDLEWARE** qui redirigera les flux entrants vers le port d'écoute https « 443 ».

```
providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    exposedByDefault: false
    watch: true

  file:
    directory: "/dynamic"
    watch: true

log:
  level: "ERROR"
  filePath: "/etc/traefik/traefik_error.log"
  format: "json"

accessLog:
  filePath: "/etc/traefik/traefik_access.log"
  format: "json"

certificatesResolvers:
  letsencrypt:
    acme:
      caServer: https://acme-v02.api.letsencrypt.org/directory
      email: xxx@xxx.com
      storage: acme.json
      keyType: EC256
      httpChallenge:
        entryPoint: http
```

Ici on active le **PROVIDER** « Docker » pour spécifier à Traefik que nous utilisons le moteur Docker.

Ici on active le **PROVIDER** « file » et on indique à Traefik que les fichiers de la configuration dynamique sont situés dans le sous-dossier « /dynamic » du dossier parent.

Ici on indique que Traefik enregistrera les fichiers d'erreurs dans un sous-dossier spécifique du conteneur et au format « json ».

Ici on indique que Traefik enregistrera les fichiers recensant les accès dans un sous-dossier spécifique du conteneur et au format « json ».

Ici on utilise le protocole « acme » qui permet à Traefik d'obtenir des certificats auprès de Let's Encrypt pour chacun des services qui seront exposés.

Une fois le fichier de configuration statique défini, il est important de créer des « middlewares » qui permettront de sécuriser l'accès à nos services exposés derrière Traefik. Pour cela, nous allons gérer la sécurité au niveau des « **headers** », ajouter des **authentifications** sécurisées par mot de passe et paramétrer les options « **tls** ».

b) Création des **FICHIERS DE CONFIGURATION DYNAMIQUES** de Traefik (les middlewares de sécurité ici)

**Création du fichier relatif aux options TLS (fichier « tls.yml »)**

Dans cette étape nous allons configurer les **options TLS**. Le **TLS** est un protocole permettant l'échange d'informations sécurisées entre un client et un serveur. Il est l'évolution logique du SSL. Il faut ensuite comprendre qu'il y a plusieurs versions de TLS allant de TLS 1.0 à TLS 1.3 et prendre en compte le fait que tous les navigateurs et OS ne prennent pas forcément en charge toutes les versions TLS.

De manière générale, on autorisera, au minimum la version **TLS 1.2**. **Les versions TLS 1.0 et 1.1 sont obsolètes à ce jour (ne pas les utiliser !).** Le TLS est composé de sous parties : les ***ciphers*** et les ***curves***.

Un ***cipher*** (suite cryptographique en français) est l'algorithme de chiffrement utilisé par le serveur pour négocier avec son client. En effet, lors d'une connexion, le client indique au serveur les algorithmes qu'il supporte et lui répondra donc avec l'algorithme le plus fort disponible si cela est possible.

Tout comme la version TLS, les *ciphers* ne sont pas compatibles avec tous les navigateurs et/ou OS, il s'agit donc de choisir avec soin les algorithmes que l'on autorise en étant conscient que certains clients ne pourront pas se connecter sur notre site, ou en tout cas, sans profiter du chiffrement en transit.

Les ***elliptic curves*** (courbes elliptiques en français) qui sont associées avec certains *ciphers* (les ciphers dit elliptiques) vont permettre de définir la façon dont le *cipher* va faire ses calculs.
On se place dans le dossier adéquat :

***cd /srv/traefik/dynamic***
**nano tls.yml**

L'éditeur s'ouvre ; on copie le contenu du fichier « yml » (voir page suivante) et on l'enregistre :

```
# Configuration TLS
tls:
  options:
    default:
      minVersion: "VersionTLS12"
      sniStrict: true
      cipherSuites:
        - "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256"
        - "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384"
        - "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305"
        - "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256"
        - "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
        - "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305"
        - "TLS_AES_128_GCM_SHA256"
        - "TLS_AES_256_GCM_SHA384"
        - "TLS_CHACHA20_POLY1305_SHA256"
      curvePreferences:
        - CurveP521
        - CurveP384
        - CurveP256
```

Dans la configuration des options TLS, on indique à Traefik que nous souhaitons, au minimum, utiliser la version 1.2 de TLS (les versions précédentes sont désormais considérées obsolètes).

On définit, ici, les « ciphers » qui sont des algorithmes de chiffrement utilisés par le serveur pour négocier avec son client. Attention, les *ciphers* ne sont pas compatibles avec tous les navigateurs et/ou OS. Il s'agit donc de choisir avec soin les algorithmes que l'on autorise en étant conscient que certains clients ne pourront pas se connecter sur notre site

Les « curves » qui sont associées avec certains *ciphers* (les ciphers dit elliptiques) vont permettre de définir la façon dont le *cipher* va faire ses calculs. Attention au choix de ces dernières qui peuvent occasionner des incompatibilités.

**Création fichier relatif au middleware HEADERS (fichier « headers.yml »)**

Une autre couche de sécurité (en plus de TLS) est disponible et passe par les **headers** que l'on renvoie depuis le serveur vers le navigateur. **Ces headers vont permettre d'indiquer au navigateur le comportement à suivre et vont permettre d'indiquer notre politique de sécurité**. Attention toutefois, il faut garder en tête que les headers sont indicatifs, et peuvent tout à fait être ignorés par le navigateur.

Les « **headers** » ont donc pour but de **vous protéger d'une éventuelle infection d'un site web ciblé par un malware**. Qui n'a pas déjà cliqué sur un site qui semblait légitime (puisqu'en HTTPS) et finalement on arrive sur tout autre chose (jeu truqué, pop-up, publicités invasives, etc…). Même si ce site est en HTTPS, il utile certainement des plugins qui ne sont pas à jour et qui ont permis l'injection de code javascript malveillant sur les pages visitées.

On peut lister différents **headers "utilisateur"** :

- **frameDeny** : indique au navigateur que la page ne peut pas être chargée dans une « iframe ».
- **sslRedirect** : permet d'indiquer de rediriger une connexion http vers du https lorsque c'est possible.
- **browserXssFilter** : permet de limiter les attaques de type "Cross Site Scripting" depuis le navigateur (plus d'infos sur le XSS ici).
- **contentTypeNosniff** : indique au navigateur de ne pas faire de détection de *mime type,* le but ici est de limiter les attaques de type "Drive by download" (plus d'infos sur le drive by download ici).

On peut les retrouver, également, sous cette forme :

- **X-Frame-Options** : interdire ou autoriser le chargement du site depuis un autre site (technique de l'iframe).
- **X-Xss-Protection** : permet de bloquer l'injection de code HTML ou Javascript sur votre site.
- **X-Content-Type-Options** : Force le navigateur à utiliser le type de données envoyées par le serveur et uniquement celui-ci.
- **Content-Security-Policy** : ce header permet de restreindre l'origine des scripts exécutés par vos pages aux sites que vous autorisez.

Parmi les « headers » récents on recense :

- **Strict-Transport-Security** : permet de forcer votre navigateur à utiliser le HTTPS pour accéder à votre site.
- **Referrer-Policy** : permet de contrôler les informations de pages "référentes" envoyées aux autres urls.
- **Feature-Policy** : votre navigateur peut accéder à votre micro, votre webcam, à la géolocalisation, etc... Cet en-tête permet de déclarer à quel type de ressource vous allez accéder. On peut, par exemple, informer notre navigateur que notre site ne demandera jamais l'accès à la webcam de l'internaute.
- **Expect-Ct** : vise à renforcer la sécurité des certificats. Les autorités de certification vont maintenant enregistrer chaque certificat délivré et rendre cette information accessible via une API. Cela permettra de s'assurer que le certificat reçu par le navigateur est bien valide et certifié.

Les **headers HSTS** (HTTP Strict Transport Security) **vont indiquer au navigateur de ne communiquer que de manière sécurisée avec le serveur**. La différence majeure par rapport à un simple header 301/Redirect vers du https est que le navigateur n'enverra aucune information vers le serveur comme les cookies utilisateur par exemple. En cas d'attaque de type « man in the middle », un site enregistré en HSTS sera protégé.
Nous avons ici 3 headers HSTS principaux :

- **stsSeconds** : indique au navigateur pendant combien de temps ce domaine doit être accéder exclusivement en https. **Pour pouvoir utiliser le HSTS Preloading, il est nécessaire que ce paramètre soit d'au moins 31536000 secondes (un an)**.
- **stsPreload** : permet d'indiquer que ce site supporte le preloading HSTS, **cela permet de sécuriser la connexion dès la première tentative au lieu de la seconde**. Le preloading nécessite d'enregistrer son site au préalable sur https://hstspreload.org/ qui permettra qu'il soit directement présent dans le navigateur.
- **stsIncludeSubdomains** : permet d'indiquer que nous souhaitons que tous les sous-domaines rattachés au domaine demandé soient aussi en HSTS.

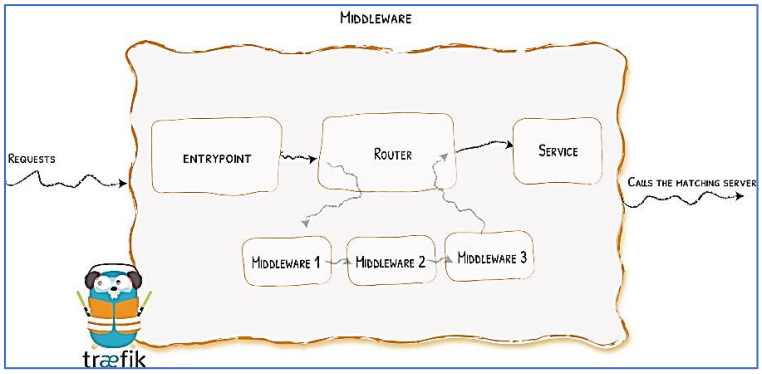
**Note importante sur le HSTS :** afin de pouvoir utiliser sereinement le HSTS avec les fonctions de preloading, il est nécessaire de s'assurer que le https est prévu pour être utilisé à long terme sur le site. Il est simple de s'enregistrer en HSTS, il est difficile de demander la suppression de son site. De plus une fois un site enregistré, le non support du https peut empêcher les utilisateurs de se connecter.

Le « **full https** » apporte de nombreux avantages :

- C'est sécurisant pour l'utilisateur : personne ne veut voir, en consultant un site, la petite indication près de la barre d'adresse "Ce site n'est pas sécurisé". Ce n'est pas rassurant pour un utilisateur. De plus de nombreux sites utilisent des interfaces de back-office avec authentification, cela permet donc d'échanger de manière chiffrée les informations et de se prémunir d'attaques de type "man in the middle".

- C'est sécurisant pour vous : le https, et notamment le « preloading HSTS », permettent de prévenir que l'on n'usurpe pas aussi facilement votre domaine pour publier des malwares par exemple.

- C'est important pour votre SEO : de plus en plus de moteurs de recherche prennent en compte la configuration du https dans le référencement. Avoir un site sécurisé permet donc d'être mieux référencé sur le web.




- Il est simple d'obtenir un certificat de nos jours : avoir un certificat HTTPS est maintenant une question de secondes, il est aussi possible d'en obtenir gratuitement (via Let's Encrypt par exemple).

Pour gérer toutes ses sécurités, nous allons « placer » des « middlewares » en interception entre le frontend et le backend. **Ces « middlewares » permettront de modifier le comportement des requêtes avant d'atteindre le backend**.

Cette phase de sécurisation est importante puisqu'elle vous permettra de tester votre site sur SSLLabs. Vous devriez avoir une évaluation positive. A noter que ce test n'est ni intrusif, ni risqué. Le test consiste à émuler des navigateurs/OS pour observer le comportement du site, et tester d'éventuelles attaques.

Le format d'un « middleware » est le suivant

```
http:
  middlewares:
    <nomdumiddleware>:
      <typedumiddleware>:
        <options>

    <nomdu2ememiddleware>:
      <typedumiddleware>:
        <options>
```

On se place dans le dossier adéquat :

***cd /srv/traefik/dynamic***
**nano headers.yml**

L'éditeur s'ouvre ; ajoutez le contenu suivant à votre fichier « yml » et enregistrez les modifications :

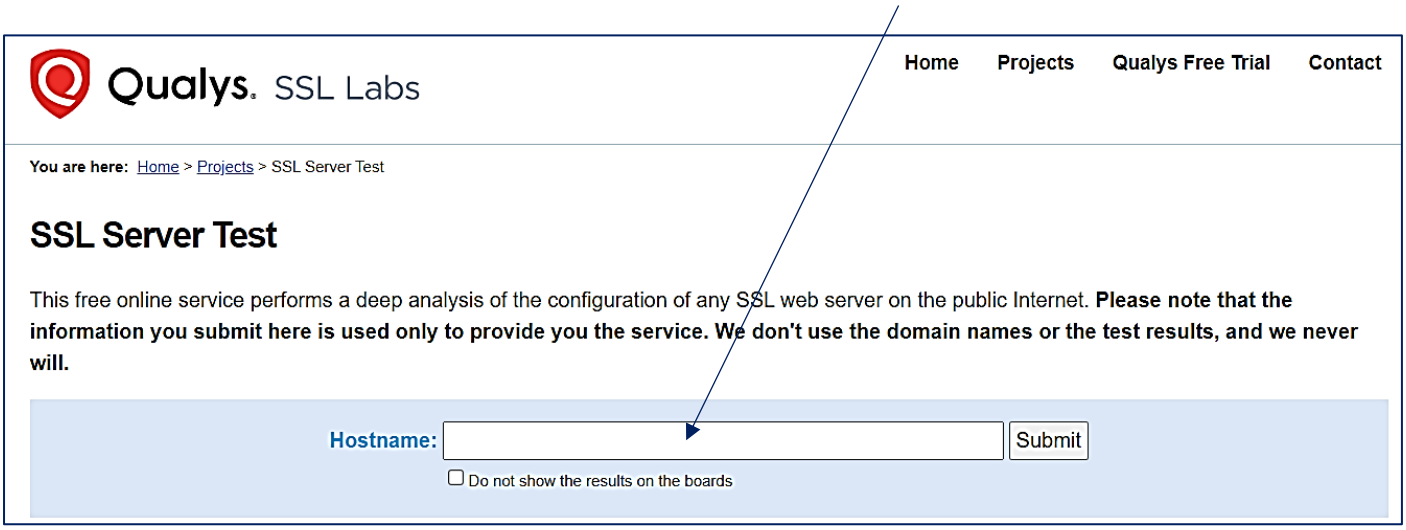
```
# Configuration des Headers
http:
  middlewares:
    Compression:
      compress:
        excludedContentTypes:
          - text/event-stream
    SecureHeaders:
      headers:
        accessControlAllowMethods:
          - GET
          - OPTIONS
          - PUT
        frameDeny: true
        sslRedirect: true
        browserXssFilter: true
        contentTypeNosniff: true
        forceSTSHeader: true
        stsIncludeSubdomains: true
        stsPreload: true
```

MIDDLEWARE « **compress** » pour compresser les réponses avant de les envoyer au client (permet d'accélérer les flux).

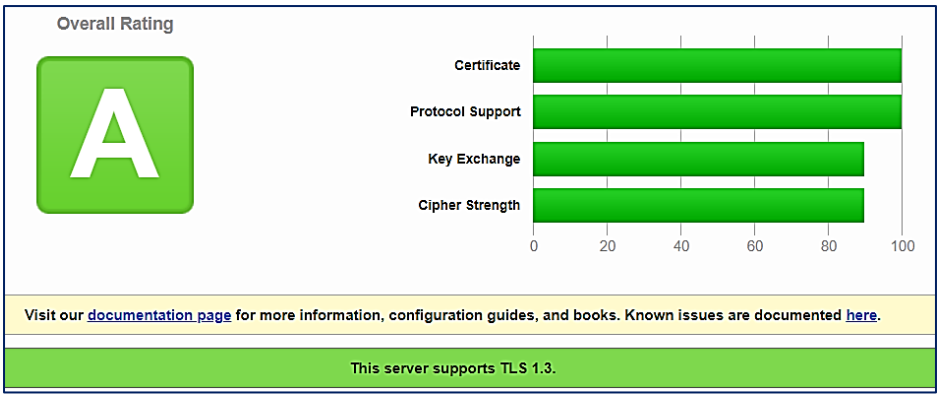
MIDDLEWARE « **headers** ». Ces headers vont permettre d'indiquer au navigateur le comportement à suivre et vont permettre d'indiquer notre politique de sécurité. Ces « headers » permettront d'obtenir une note de sécurité élevée sur des sites comme **SSL Labls**.

```
accessControlMaxAge: 100
addVaryHeader: true
customFrameOptionsValue: "SAMEORIGIN"
referrerPolicy: "same-origin"
featurePolicy: "vibrate 'self'"
stsSeconds: 315360000
permissionsPolicy: "camera 'none'; microphone 'none'; geolocation 'none'; payment 'none';"
hostsProxyHeaders:
  - "X-Forwarded-Host"
```

Le site SSL Labs permet d'obtenir un diagnostic de sécurité à partir d'une URL que l'on souhaite tester :




Exemple de note obtenue à partir de l'URL https://tutos-info.fr




## Création d'une authentification (avec un mot de passe sécurisé) pour l'accès au tableau de bord de Traefik

**1ère méthode** : utilisation du paquet « apache2-utils » :

Pour accéder au tableau de bord de Traefik, il est préférable de sécuriser l'accès et de créer un utilisateur qui s'authentifiera avec un mot de passe sécurisé. Pour cela, on peut utiliser l'outil « **htpasswd** » qui est disponible avec le paquet « apache2-utils » précédemment installé.

Par exemple, si nous souhaitons créer un utilisateur nommé « **admin** » avec le mot de passe « **TraefikAdmin** », on saisira la commande suivante :

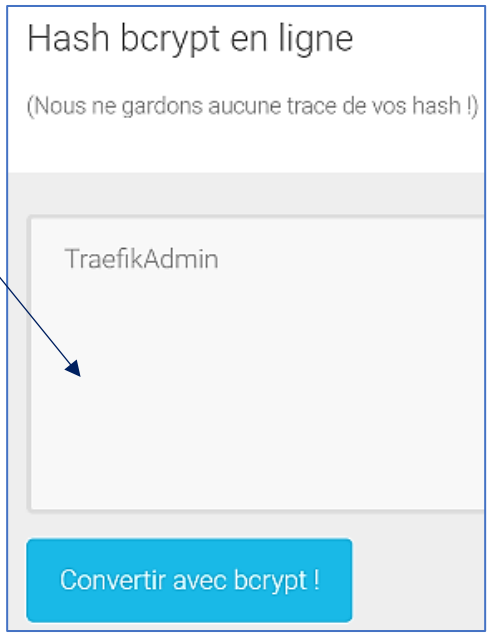
***htpasswd -nb admin TraefikAdmin***

On obtient ceci : **admin:$apr1$3j9TsNtO$nI5HMMFkIiYK7qR3Nm/.L/**

C'est ce mot de passe qu'il faudra utiliser lors de l'authentification au tableau de bord de Traefik (voir plus bas).

**2ème méthode** : utilisation du site Bcrypt

Dans ce tutoriel, nous allons utiliser une autre solution de cryptage avec le **hash bcrypt en ligne**. Pour cela, connectez-vous au site https://www.bcrypt.fr/




Une fois sur le site, saisissez le mot de passe que vous désirez (par exemple « TraefikAdmin ») et cliquez sur le bouton « **Convertir avec bcrypt !** ».

Vous obtenez le mot de passe haché suivant :
**2y$10$pUE7h9.qWGKs446No0ZvTuVFlsonCxn1OTI9gkArnt1BQyl1ZDh2i**

On va maintenant créer un middleware qui permet de gérer l'authentification d'un utilisateur avec un mot de passe sécurisé. Pour cela, on va créer ici un fichier « **auth.yml** ». On se place dans le dossier adéquat :

***cd /srv/traefik/dynamic***
**nano auth.yml**

L'éditeur s'ouvre ; ajoutez le contenu suivant à votre fichier « yml » et enregistrez les modifications :

**Contenu du fichier « auth.yml »**

```
#Authentification dashboard Traefik
http:
  middlewares:
    user-auth:
      basicAuth:
        users:
          - "admin:$2y$10$pUE7h9.qWGKs446No0ZvTuVFlsonCxn1OTI9gkArnt1BQyl1ZDh2i"
```

Ici on définit un **MIDDLEWARE** nommé « **basic-Auth** » qui reprend le mot de passe obtenu sur le site bcrypt pour un utilisateur que nous avons nommé « admin ».

## 4 – LANCEMENT DE TRAEFIK 3.2 AVEC DOCKER COMPOSE

C'est la création du fichier « **docker-compose.yml** » qui lance la création de la stack complète

Le fichier « **docker-compose.yml** » doit être créé dans le dossier parent « **/srv/traefik** » :

***cd /srv/traefik***
***nano docker-compose.yml***

L'éditeur s'ouvre ; on copie le contenu du fichier « yml » listé ci-dessous et **on l'enregistre avec le nom suivant :** « **docker-compose.yml** » (**attention, saisissez bien ce nom de fichier** !).

***Note importante :*** *pour que vos redirections fonctionnent (voir rubrique « labels »), vous devez obligatoirement posséder un nom de domaine hébergé et avoir créé des enregistrements « A » ou « CNAME » pointant vers votre serveur (nous ne pouvons pas expliquer cette notion ici car elle varie selon votre hébergeur).*

**Contenu du fichier « docker-compose.yml »**

```
---
services:
  traefik:
    image: traefik:latest
    container_name: traefik
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    networks:
      - traefik-proxy
    ports:
      - 80:80
      - 443:443
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./static/traefik.yml:/traefik.yml:ro
      - ./certs/acme.json:/acme.json
      - ./dynamic:/dynamic
      - ./logs/traefik_error.log:/etc/traefik/logs/traefik_error.log
      - ./logs/traefik_access.log:/etc/traefik/logs/traefik_access.log
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.traefik.rule=Host(`traefikprof.sio-ndlp.fr`)"
      - "traefik.http.routers.traefik-secure.middlewares=user-auth@file,SecureHeaders@file"
      - "traefik.http.routers.traefik-secure.rule=Host(`traefikprof.sio-ndlp.fr`)"
      - "traefik.http.routers.traefik-secure.tls=true"
      - "traefik.http.routers.traefik-secure.tls.certresolver=letsencrypt"
      - "traefik.http.routers.traefik-secure.service=api@internal"

networks:
  traefik-proxy:
    external: true
```

Création du conteneur « TRAEFIK » avec une politique de redémarrage automatique en cas de défaillance du serveur. Traefik utilisera le réseau « traefik-proxy » et écoutera les ports « 80 » et « 443 ».

Ici, on « monte » l'ensemble des volumes nécessaires au fonctionnement de Traefik.

Ici, on indique à Traefik l'emplacement des fichiers de la configuration statique., dynamique, des certificats Let's Encrypt et des logs.

Ici, on indique à Traefik que l'accès au tableau de bord s'effectuera via le nom de domaine hébergé par l'utilisateur.

Ici, on indique à Traefik d'utiliser des « **LABELS** » renvoyant vers les différents services et les « middlewares » de sécurité à utiliser. Tous ces labels sont à saisir pour chaque service derrière Traefik. Pensez à relancer un docker-compose up -d à chaque ajout d'un label (nous verrons une autre méthode dans un autre tutoriel).

Vos fichiers de configuration sont maintenant prêts ! Il faut donc lancer la création de notre stack Traefik. Pour cela, **on se place dans le dossier où le fichier « docker-compose.yml » a été enregistré** et on saisit la commande suivante : « **docker compose up -d** » qui va lancer la création complète de l'environnement Traefik 3.2.2 :

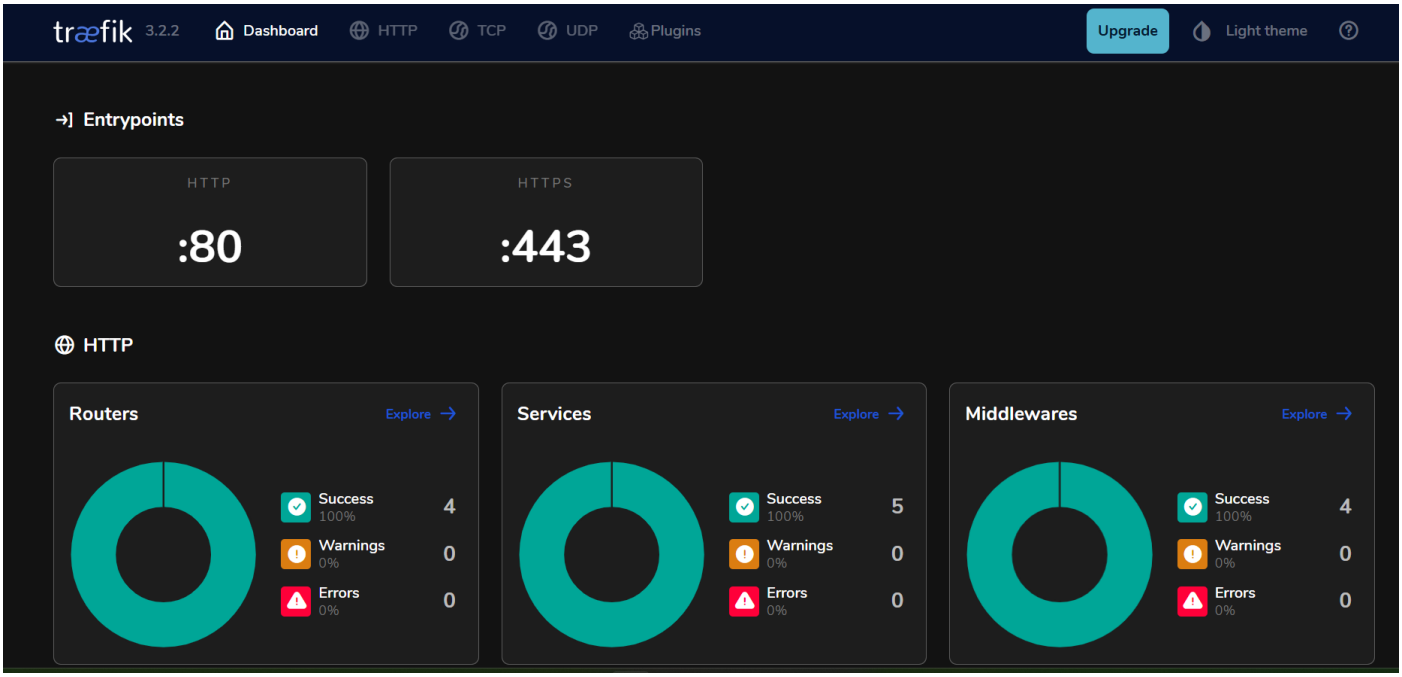
***cd /srv/traefik***
***docker compose up -d***

Patientez quelques instants le temps que les images et les conteneurs se créent. Si vous n'avez pas de message d'erreur, votre environnement complet est opérationnel !

Lancez votre navigateur et tester la connexion à votre tableau de bord Traefik via l'adresse que vous avez définie précédemment : **https://traefik.votredomaine**

Si tout est opérationnel, vous devriez voir s'afficher la page d'accueil du tableau de bord de Traefik avec l'authentification.

Une fois authentifié(e), la page d'accueil de Traefik 3.2.2 s'affiche :




En cliquant le lien « Explore » des Middlewares, on obtient l'ensemble des middlewares qui ont été définis lors de la préparation (le « SecureHeaders » avec la configuration des options TLS et des en-têtes, la redirection « http-to-https » et l'authentification au tableau de bord « user-auth ».

On voit bien ici que l'authentification et les en-têtes ont bien été définis dans un fichier (voir logo Provider) et que la redirection a été appliquée par Traefik.