

NOUVEAU

TUTORIEL PRATIQUE



traefik
proxy **3.6**

INSTALLER TRAEFIK 3.6.5 EN BARE METAL SUR DEBIAN 13

De l'installation de base aux premières manipulations

POINTS CLES DU TUTORIEL



Installation pas à pas de **Traefik 3.6.5** (au 1^{er}/1/2026)



La sécurisation de l'accès au tableau de bord de Traefik 3.5



L'ajout d'un service « Cockpit » à l'aide de la configuration dynamique de Traefik



La gestion du protocole « QUIC » (http/3)

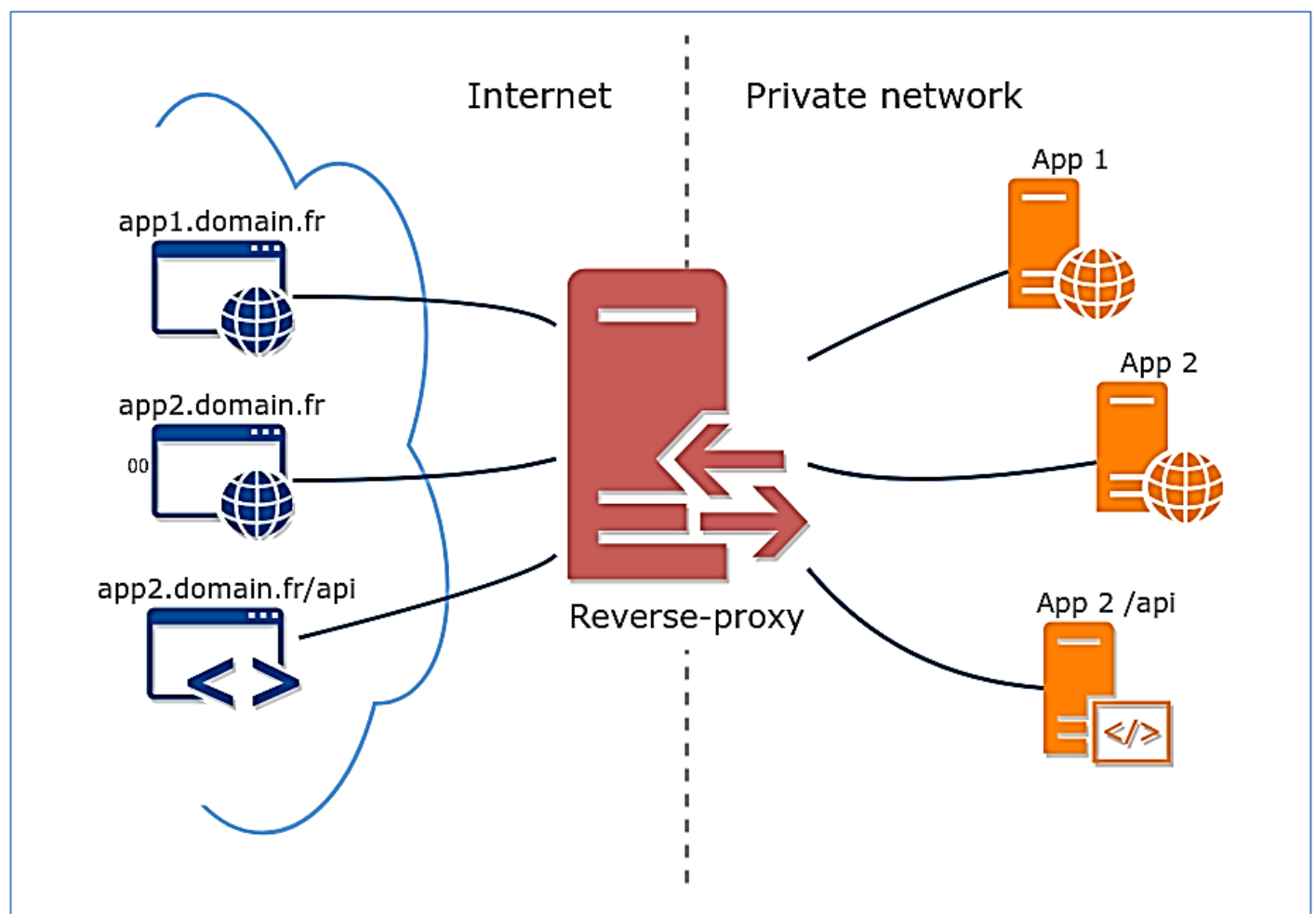
Notes :

Ce tutoriel nécessite une version Debian 13 (appelée « Trixie ») fonctionnelle (voir tutoriel « Installer et configurer Debian 13 » sur le site). Pour plus de facilité dans l'utilisation des commandes, nous recommandons un accès à la machine via SSH (voir tutoriel « Installer et configurer Debian 13 » sur le site). La machine Debian devra disposer d'un espace de stockage d'environ 20 Go et de 2 Go de RAM pour fonctionner correctement.

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 votre 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).

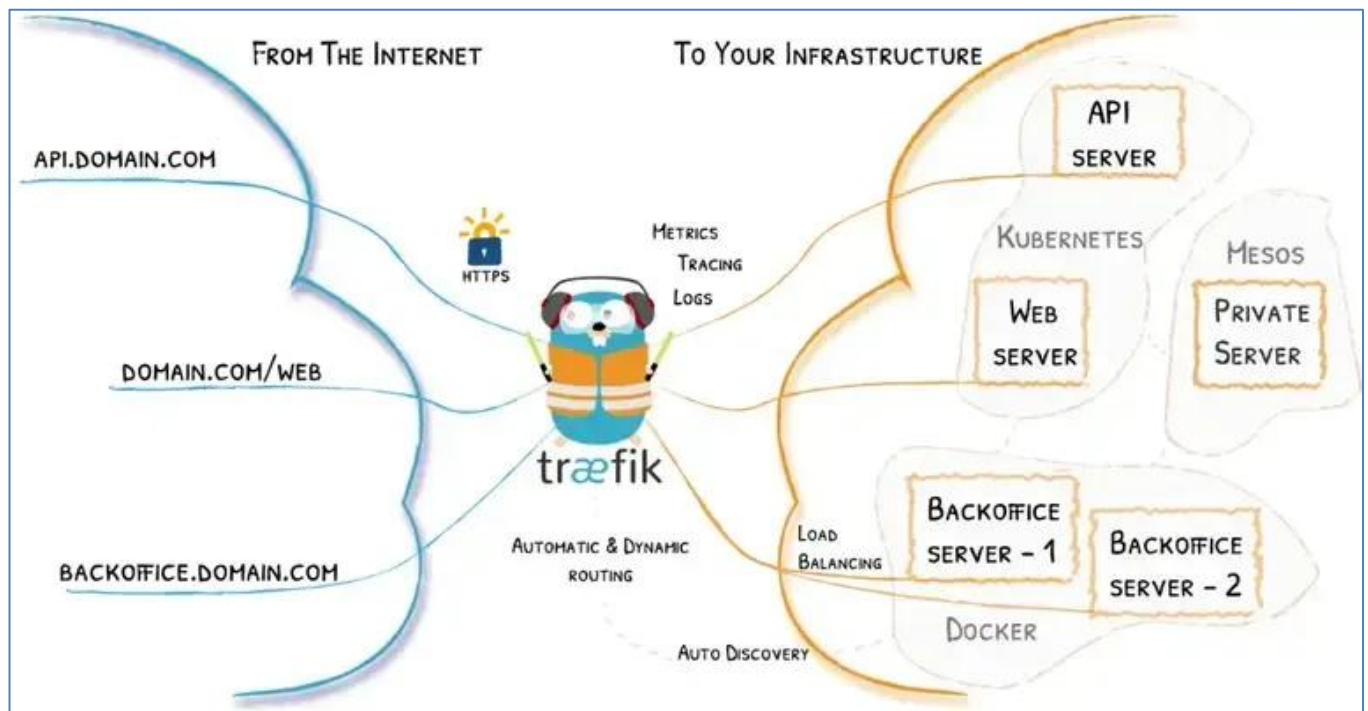


La véritable **force de Traefik** c'est que **sa configuration est dynamique** : il va se connecter au socket Docker et récupérer toutes les informations sur les containers qui tournent en temps réel de façon à pouvoir router les requêtes sur ces derniers de manière automatique ! Traefik supporte plusieurs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

À ce jour (janvier 2026), la **dernière version en date stable** de Traefik est la **3.6.5**

Traefik est conçu pour les applications conteneurisées et les architectures microservices. En tant que tel, il joue un rôle important dans le routage des requêtes des clients vers les services appropriés au sein d'une infrastructure.

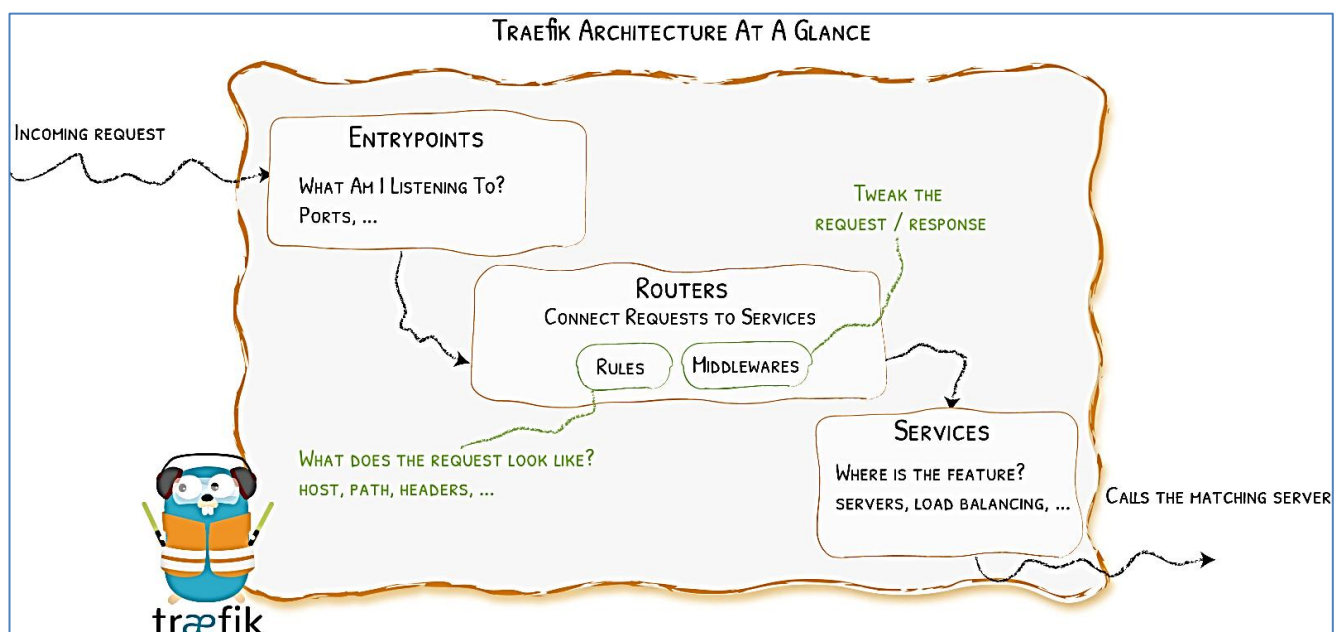


Lorsqu'un utilisateur envoie une requête HTTP à Traefik, celle-ci contient toutes les informations nécessaires pour indiquer à Traefik sa destination. En général, on utilise le nom de domaine et le chemin de la requête pour déterminer sa destination.

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 »).



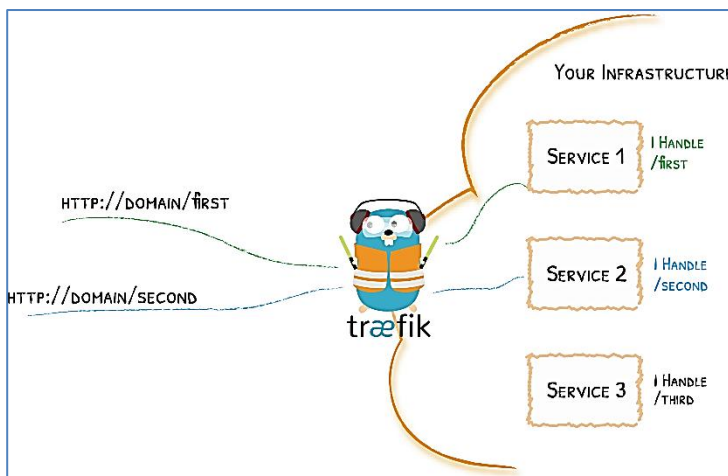
Les **ENTRYPOINTS** sont les **points d'entrées**. Il s'agit des adresses et ports exposés par Traefik qui permettront d'exposer nos applications. La plupart du temps, ce sont les **ports 443 et 80** qui sont exposés par défaut.

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

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.

1 – INSTALLATION DE TRAEFIK 3.6.5 EN BARE METAL SUR DEBIAN 13

Dans les précédents tutoriels, nous avons réalisé l'installation de Traefik avec Docker. Ici, nous présentons l'installation en « bare métal » c'est-à-dire à partir des binaires officiels.

- Connectez-vous à votre machine Debian 13 avec un utilisateur « sudo » (ou à défaut en « root »).
- Commencez par mettre à jour les paquets avec la commande suivante :

sudo apt update && sudo apt upgrade -y

- Installez les paquets « curl » et « unzip » avec la commande suivante :

sudo apt install curl -y

- Créez un utilisateur dédié qui administrera Traefik avec les commandes suivantes :

```
sudo useradd --system --home /etc/traefik --shell /bin/false traefik
sudo mkdir -p /etc/traefik /var/lib/traefik
sudo chown traefik:traefik /etc/traefik /var/lib/traefik
```

- Dans le répertoire « home » de l'utilisateur actuellement connecté, téléchargez **la dernière version de Traefik depuis les sources officielles** ([Releases · traefik/traefik](https://github.com/traefik/traefik/releases) – version 3.6.5 au 1^{er}/1/2026) avec les commandes suivantes :

```
wget https://github.com/traefik/traefik/releases/download/v3.6.5/traefik_v3.6.5_linux_amd64.tar.gz
```

- Décompressez l'archive téléchargée dans le dossier « /usr/local/bin » avec la commande suivante :

```
sudo tar -xvzf traefik_v3.6.5_linux_amd64.tar.gz -C /usr/local/bin traefik
```

- Modifiez les droits sur le fichier « traefik » avec la commande suivante :

```
sudo chmod +x /usr/local/bin/traefik
```

CREATION DU FICHIER DE CONFIGURATION STATIQUE (avec gestion du protocole « QUIC »)

Note : 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).

- Créez le fichier « **traefik.yml** » dans « **/etc/traefik** » avec la commande suivante :

```
nano /etc/traefik/traefik.yml
```

- Collez le contenu suivant (en conservant les indentations) **en indiquant votre mail** :

```
log:
  level: INFO

api:
  dashboard: true
  insecure: false

entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
          permanent: true

  websecure:
    address: ":443"
    http3:
      advertisedPort: 443 # ← Important : annonce le port QUIC aux clients
    http:
      tls:
        certResolver: le # ← Assure que TLS est bien configuré

# Charge la config dynamique
providers:
  file:
    directory: "/etc/traefik/dynamic"
    watch: true
```

certificatesResolvers:

le:

acme:

email: "xxx@xxx.xxx"

storage: "/var/lib/traefik/acme.json"

httpChallenge:

entryPoint: web

- Sauvegardez et quittez avec **CTRL + O** et pressez la touche **Entrée**.

PREPARATION DU FICHIER « acme.json »

Pour le stockage des certificats Let's Encrypt, nous avons besoin d'un fichier particulier nommé « **acme.json** » avec les droits « 600 ». Pour cela, saisissez les commandes suivantes :

```
sudo touch /var/lib/traefik/acme.json
```

```
sudo chown traefik:traefik /var/lib/traefik/acme.json
```

```
sudo chmod 600 /var/lib/traefik/acme.json
```

GENERATION D'UN MOT DE PASSE CHIFFRE POUR L'ACCES AU TABLEAU DE BORD DE TRAEFIK

Cette étape permet de **créer un mot de passe chiffré pour l'accès futur au tableau de bord de Traefik**. En effet, le tableau de bord sera exposé sur le web et il est important de sécuriser son accès. Pour cela, effectuez les manipulations suivantes :

- Ajoutez le paquet « **apache2-utils** » qui permet la génération des mots de passe chiffrés avec la commande suivante :

```
sudo apt install apache2-utils -y
```

- Générez un mot de passe chiffré, avec le hash Bcrypt, à l'aide de la commande suivante (ici, on génère un mot de passe pour l'utilisateur « admin » et le mot de passe « mon_mot_de_passe » qui est à adapter) :

```
htpasswd -nbB admin mon_mot_de_passe
```

- Conservez le résultat qui est sous la forme suivante :

(**admin:\$2y\$05\$.6w3wSTa50BLhHfZuaEQMOwPhkGHQ\$juYBO.KVtzE1BBvu**) en le copiant/collant dans un fichier « txt » par exemple

CREATION DU TABLEAU DE BORD AVEC UN FICHIER DYNAMIQUE

La création du « dashboard » s'effectue en créant un **fichier au format « yml » qui sera enregistré dans la configuration dynamique de Traefik**. Pour cela, saisissez les commandes suivantes :

```
mkdir /etc/traefik/dynamic
```

```
nano /etc/traefik/dynamic/dashboard.yml
```

- Saisissez ou copiez le contenu suivant :

http:

routers:

dashboard:

rule: "Host(`traefik.mon\_domaine.fr`) && (PathPrefix(`/api`) || PathPrefix(`/dashboard`))"

entryPoints: ["websecure"]

service: api@internal

tls:

```
certResolver: le
middlewares:
- auth
- SecureHeaders
- Compression
```

```
dashboard-root:
rule: "Host(`traefik.mon_domaine.fr`) && Path(`/`)"
entryPoints: ["websecure"]
service: api@internal
tls:
certResolver: le
middlewares:
- dash-redirect
- SecureHeaders
```

```
middlewares:
auth:
basicAuth:
realm: "Traefik"
users:
- "admin:coller_ici_le_mot_de_passe_chiffré"
```

```
dash-redirect:
redirectRegex:
regex: "^/$"
replacement: "/dashboard/"
permanent: true
```

- Quittez et sauvegardez le fichier « dashboard.yml » avec **CTRL + O** et pressez la touche **Entrée**.

CONFIGURATION DES MIDDLEWARES DE SECURITE

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.

1^{ère} étape : création du middleware « tls.yml »

- Créez le middleware en ajoutant un fichier « **tls.yml** » dans la configuration dynamique de Traefik à l'aide de la commande suivante :

nano /etc/traefik/dynamic/tls.yml

- Saisissez ou copiez le contenu suivant :

Configuration TLS

tls:

options:

default:

minVersion: "VersionTLS12"

sniStrict: true

preferServerCipherSuites: 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_GCM_SHA256"
- "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
- "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305"

curvePreferences:

- CurveP256
- CurveP384
- CurveP521

- Quittez et sauvegardez le fichier avec **CTRL + O** et pressez la touche **Entrée**.

EXPLICATION DU FICHIER « tls.yml »

Les règles de chiffrement TLS que Traefik applique par défaut permettent de dire à Traefik *quels protocoles/algorithmes* chiffrés il accepte côté client (navigateur). Cela influence directement la note SSL Labs et la sécurité.

Dans notre version durcie 2025 « **tls.yml** » proposée ici, on a (idées clés) :

minVersion: "VersionTLS12"

→ On **refuse TLS 1.0/1.1** (obsolètes). Les navigateurs modernes parleront **TLS 1.2 ou 1.3**.

sniStrict: true

→ Si le client demande un nom (SNI) pour lequel **Traefik n'a pas de certificat**, la connexion est **refusée** (plutôt que de servir un "mauvais" cert).

Gain : évite des certs "mismatch" et des confusions entre hôtes.

preferServerCipherSuites: true

→ Pour **TLS 1.2**, c'est le **serveur** (Traefik) qui choisit sa suite préférée parmi celles communes (au lieu du client).

Gain : on impose des suites "fortes".

cipherSuites: [...] (TLS 1.2)

→ Liste blanche de suites **modernes** uniquement (AES-GCM et **ChaCha20-Poly1305**), **sans CBC** ni SHA1.

Gain : on évite des algorithmes fragiles et on reste compatible avec le parc actuel.

curvePreferences: [CurveP256, CurveP384, CurveP521]

→ Ordre de préférence des courbes ECC pour l'échange de clés (ECDHE).

Gain : P256/P384 sont rapides et sûres, P521 en dernier.

Important : pour **TLS 1.3**, Go/Traefik gère tout automatiquement. On **ne configure pas** les suites TLS 1.3 dans « tls.yml » (c'est normal).

2^{ème} étape : création du middleware « 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 utilise certainement des plugins qui ne sont pas à jour et qui ont permis l'injection de code javascript malveillant sur les pages visitées.

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 [SSLabs](https://www.ssllabs.com/). Vous devriez avoir une évaluation positive. À 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.

- Créez le middleware « **headers.yml** », dans la configuration dynamique, avec la commande suivante :

nano /etc/traefik/dynamic/headers.yml

- Saisissez ou copiez le contenu suivant :

http:

middlewares:

Compression:

compress:

excludedContentTypes:

- text/event-stream

SecureHeaders:

headers:

HTTPS/HSTS

forceSTSHeader: true

stsSeconds: 63072000

stsIncludeSubdomains: true

stsPreload: true

X-Frame-Options (garde SAMEORIGIN)
customFrameOptionsValue: "SAMEORIGIN"

Divers sûreté
contentTypeNosniff: true
referrerPolicy: "same-origin"

Permissions-Policy
permissionsPolicy: >-
 accelerometer=(), camera=(), geolocation=(),
 gyroscope=(), magnetometer=(), microphone=(),
 payment=(), usb=()

CSP moyennement durcie avec compatibilité Dashboard Traefik et Cockpit

- autorise inline JS/CSS + eval (Cockpit en a besoin)
- autorise WebSockets (wss:) pour /api & Cockpit
- garde les garde-fous (frame-ancestors/base-uri/form-action)
contentSecurityPolicy: >-
 default-src 'self';
 script-src 'self' 'unsafe-inline' 'unsafe-eval';
 style-src 'self' 'unsafe-inline';
 img-src 'self' data: blob;;
 font-src 'self' data;;
 connect-src 'self' https: wss;;
 frame-ancestors 'self';
 base-uri 'self';
 form-action 'self';
 upgrade-insecure-requests

CORS (laisse désactivé si inutile)
accessControlAllowOriginList:
- "https://app.exemple.com"
accessControlAllowMethods:
- "GET"
- "POST"
- "OPTIONS"
accessControlAllowHeaders:
- "Authorization"
- "Content-Type"
accessControlMaxAge: 600
addVaryHeader: true

- Quittez et sauvegardez le contenu avec **CTRL + O** et pressez la touche **Entrée**.

EXPLICATION DU FICHER « headers.yml »

Le but du fichier « headers.yml » est d'envoyer des **en-têtes de sécurité** aux navigateurs (HSTS, CSP, etc.). Ça protège contre le "mixte", XSS, clickjacking, et contribue à l'obtention de la note « A+ » sur SSL Labs.

Dans notre configuration dynamique Traefik, nous avons **2 middlewares** : « **Compression** » et « **SecureHeaders** » :

- **Compression**
→ Active la compression **sauf** pour text/event-stream (SSE ne doit pas être compressé).
Gain : performanace sans casser SSE.

- **SecureHeaders**

→ Le paquet d'en-têtes de sécurité. Détails :

- **HSTS**

- forceSTSTHeader: true
 - stsSeconds: 63072000 (≈ 2 ans)
 - stsIncludeSubdomains: true
 - stsPreload: true

→ Le navigateur **force HTTPS** pour le domaine mentionné (et les sous-domaines).
Impact SSL Labs : indispensable pour obtention note **A+**.

- **X-Frame-Options**

- customFrameOptionsValue: "SAMEORIGIN"

→ Empêche l'intégration de tes pages en iframe sur d'autres domaines (clickjacking).
Si vous devez intégrer depuis un autre domaine, on passera plutôt par frame-ancestors dans la CSP (plus fin).

- **NoSniff & Référent**

- contentTypeNosniff: true → évite un "mauvais" type MIME interprété.
 - referrerPolicy: "same-origin" → ne fuite pas l'URL complète vers des sites tiers.

- **Permissions-Policy (remplace "Feature-Policy")**

- On **désactive** des capacités (caméra, micro, géoloc, etc.) par défaut :
accelerometer=(), camera=(), geolocation=(), etc.

→ Un site tiers ne peut pas activer ces capteurs "par surprise".

- **CSP (Content-Security-Policy) — le cœur anti-XSS / anti contenu tiers**

- **CORS (commenté)**

→ On laisse **désactivé** si on n'a pas d'API publique cross-domain.

Règle d'or : commencer par une politique **stricte**, puis **ouverture ciblée** par **domaine** et par **directive** selon les besoins de chaque application.

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

Qualys. SSL Labs

[Home](#) [Projects](#) [Qualys Free Trial](#) [Contact](#)

You are here: [Home](#) > [Projects](#) > SSL Server Test

SSL Server Test

This free online service performs a deep analysis of the configuration of any SSL web server on the public Internet. **Please note that the information you submit here is used only to provide you the service. We don't use the domain names or the test results, and we never will.**

Hostname:

Submit

☐ Do not show the results on the boards

Avec notre configuration, nous obtenons une note « A+ » pour l’accès à notre dashboard Traefik :

Qualys. SSL Labs

[Home](#) [Projects](#) [Qualys Free Trial](#) [Contact](#)

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > 1

SSL Report: traefik

Assessed on: Sat, 16 Aug 2025 09:31:59 UTC | **HIDDEN** | [Clear cache](#)

[Scan Another »](#)

Summary

Overall Rating



Certificate	
Protocol Support	
Key Exchange	
Cipher Strength	

Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This site works only in browsers with SNI support.

This server supports TLS 1.3. [MORE INFO »](#)

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. [MORE INFO »](#)

12

CREATION DU SERVICE « SYSTEMD »

- Créez le fichier « **traefik.service** » dans « **/etc/systemd/system** » avec la commande suivante :

nano /etc/systemd/system/traefik.service

- Copiez le contenu suivant :

[Unit]

Description=Traefik Reverse Proxy

After=network-online.target

Wants=network-online.target

[Service]

User=traefik

Group=traefik

ExecStart=/usr/local/bin/traefik --configfile=/etc/traefik/traefik.yml

WorkingDirectory=/etc/traefik

Restart=on-failure

AmbientCapabilities=CAP_NET_BIND_SERVICE

CapabilityBoundingSet=CAP_NET_BIND_SERVICE

NoNewPrivileges=true

LimitNOFILE=1048576

ProtectSystem=full

ProtectHome=true

PrivateTmp=true

ReadWritePaths=/var/lib/traefik /etc/traefik

[Install]

WantedBy=multi-user.target

- Quittez et sauvegardez le fichier avec **CTRL + O** et pressez la touche **Entrée**.

DEMARRAGE DE TRAEFIK

- Démarrez Traefik avec les commandes suivantes :

sudo systemctl daemon-reload

sudo systemctl enable --now traefik

sudo systemctl status traefik

Note : attention, à partir de ce moment, il faut que vous activiez des règles dans votre pare-feu si vous en avez un.

Par exemple, si vous disposez de pfSense, il faut avoir créé les règles suivantes (voir tutoriels pfSense sur le site) :

- Rediriger WAN:80 → DebianTraefik:80
- Rediriger WAN:443 → DebianTraefik:443
- Pas de filtrage ou bloc RFC1918 sur ces règles

CONNEXION AU TABLEAU DE BORD DE TRAEFIK

- Ouvrez votre navigateur et saisissez l'URL de votre domaine (pour cela il faut avoir un domaine hébergé et un enregistrement « A » qui pointe sur votre IP publique et avoir géré les règles dans votre pare-feu).

Logiquement un pop-up s'affichera et vous demander de vous authentifier avec l'utilisateur généré préalablement « admin » et mot de passe.

Se connecter pour accéder à ce site

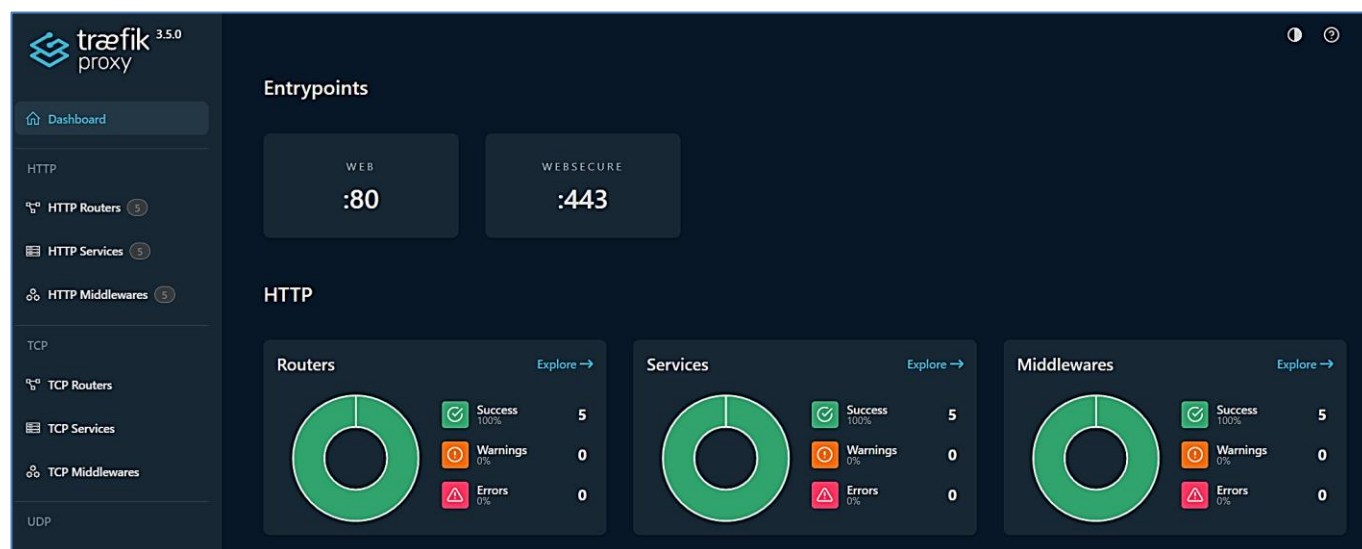
Autorisation requise par https://traefik.██████.fr

Nom d'utilisateur

Mot de passe

[Se connecter](#) [Annuler](#)

Tableau de bord de Traefik 3.5.0



En cliquant le lien « **Explore** » de la rubrique « **Middlewares** », on obtient l'ensemble des middlewares qui ont été définis lors de la préparation : le « **SecureHeaders@file** » avec la configuration des options TLS et des en-têtes, le « **Compression@file** » pour accélérer les flux, la redirection « **redirect-web-to-websecure@internal** » et l'authentification au tableau de bord « **auth@file** ». Le « **dash-redirect@file** » prend en charge le protocole QUIC (http/3).

On voit bien ici que l'authentification et les en-têtes ont bien été définis dans un fichier (indication « **@file** ») et que la redirection, définie dans le fichier interne de configuration statique, a été appliquée par Traefik (indication « **@internal** ») :

Status	Name	Type	Provider
✓	Compression@file	compress	📄
✓	SecureHeaders@file	headers	📄
✓	auth@file	basicauth	📄
✓	dash-redirect@file	redirectregex	📄
✓	redirect-web-to-websecure@internal	redirectscheme	⚙️

Note : la réalisation de cette partie suppose que vous disposez d'une autre machine Debian avec Cockpit installé.

Dans cette partie, nous allons expliquer comment ajouter, de façon dynamique, un **SERVICE** tel que « Cockpit » derrière notre reverse proxy Traefik. Nous possédons une autre machine Debian13 sur laquelle nous avons installé Cockpit (voir autre tutoriel sur le site). Nous avons un enregistrement DNS « A » chez notre hébergeur qui pointe vers notre IP publique et une règle DNAT dans notre pare-feu pfSense.

CREATION DU SERVICE « cockpit.yml »

Dans cette partie, nous allons ajouter notre nouveau service à Traefik en utilisant un fichier de configuration dynamique.

Quand on utilise Traefik comme reverse proxy et que l'on souhaite ajouter un service, il suffit d'utiliser un **fichier de configuration dynamique** pour le nouveau service au format « **yml** » **dans lequel nous allons déclarer le routeur et le service sans avoir à relancer Traefik**. Traefik va instantanément interpréter ce fichier dynamique et appliquer les règles de routage définies !

- Créez le fichier « **cockpit.yml** » avec la commande suivante :

nano /etc/traefik/dynamic/cockpit.yml

- Copiez le contenu suivant :

```
http:
  routers:
    cockpit:
      rule: "Host(`cockpit.mon_domaine.fr`)"
      entryPoints: ["websecure"]
      service: cockpit-svc
      middlewares:
        - SecureHeaders
        - Compression
      tls:
        certResolver: le # Let's Encrypt

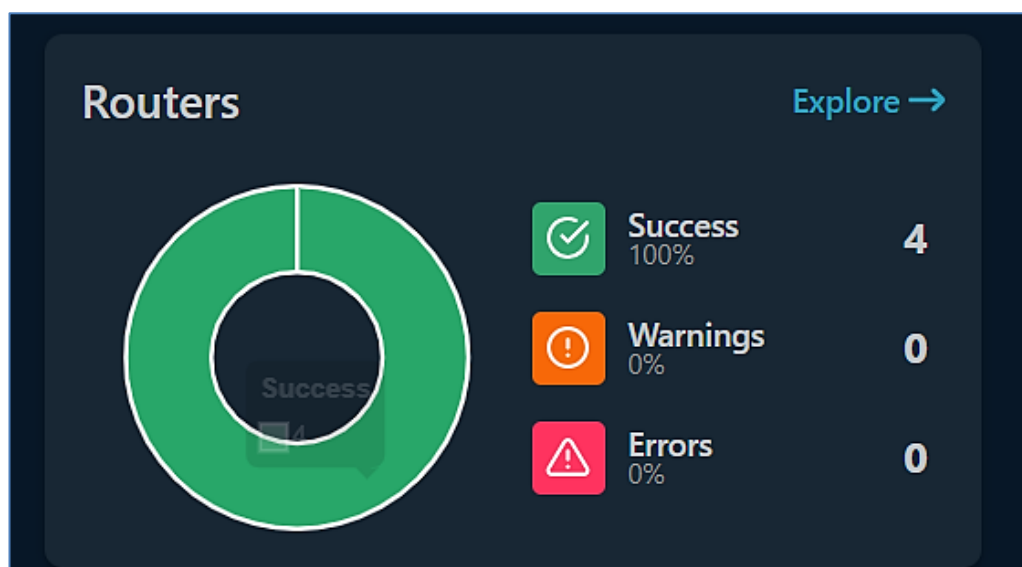
services:
  cockpit-svc:
    loadBalancer:
      servers:
        - url: "https://10.10.10.13:9090"
        passHostHeader: true
        serversTransport: cockpit-transport

serversTransports:
  cockpit-transport:
    insecureSkipVerify: true
```

- Quittez et sauvegardez le fichier avec **CTRL + O** et pressez la touche **Entrée**.

A ce stade, le « routeur » est prêt et comme la configuration est dynamique il est directement pris en charge par Traefik !

- Depuis le tableau de bord de Traefik, cliquez sur « **Explore** » dans la rubrique « **Routers** » :



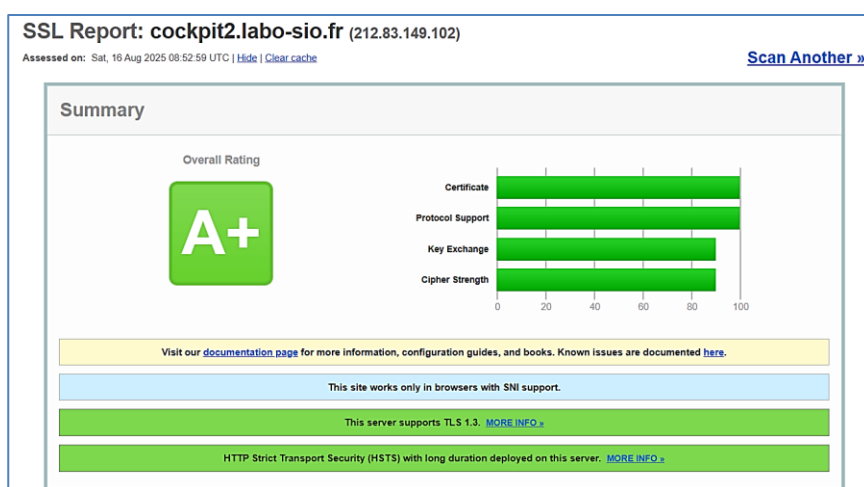
On voit ici que les routeurs « **cockpit** » et « **traefik** » sont présents avec un certificat Let's Encrypt automatiquement généré (colonne « **TLS** » avec **icône verte** :

Status	TLS	Rule	Entrypoints	Name	Service	Provider	Priority
		PathPrefix('/.well-known/acme-c...)	web	acme-http@internal	acme-http@internal		922337...
		Host('cockpit...fr')	websecure	cockpit2@file	cockpit2-svc		28
		Host('traefik...fr')	websecure	dashboard@file	api@internal		27
		HostRegexp('^.+\$.')	web	web-to-websecure@internal	noop@internal		922337...

L'avantage avec Traefik réside dans la gestion de la configuration dynamique.

L'ajout d'un simple fichier « yml » dans la configuration dynamique entraîne la mise en service immédiate du load balancer !

Pour notre service Cockpit, nous obtenons une note « **A+** » avec SSL Labs !



Note : lorsque vous ajoutez un service, il n'est pas utile de redémarrer Traefik car Traefik le gère dynamiquement. Pour information, il est possible de relancer Traefik avec la commande suivante : « **sudo systemctl restart traefik** ».

Qu'est-ce que le HTTP/3 ?

Le protocole de transfert hypertexte (HTTP) est un élément essentiel d'Internet ; il détermine la manière dont les plateformes et les appareils de communication échangent des informations et récupèrent les ressources. En résumé, il s'agit de ce qui permet aux utilisateurs de charger les sites web.

HTTP/3 est la dernière version majeure de [HTTP](#). Les navigateurs et les serveurs web peuvent l'utiliser pour **améliorer considérablement** l'expérience utilisateur, notamment en ce qui **concerne les performances, la fiabilité et la sécurité**. La négociation des versions HTTP s'effectue de manière fluide, sans qu'aucune modification de code du site web ne soit nécessaire.

Quelles sont les nouveautés de HTTP/3 ?

HTTP/3 est la première mise à niveau majeure de HTTP depuis l'approbation de [HTTP/2](#) en 2015.

HTTP/3 se distingue de la version précédente notamment par le fait qu'il **s'exécute sur QUIC**, un **nouveau protocole de transport**. **QUIC est conçu pour être rapide** et pour permettre le passage rapide d'un réseau à l'autre. **Il repose sur le protocole UDP (User Datagram Protocol)** au lieu du protocole **TCP** (Transmission Control Protocol), ce qui atténue un problème appelé blocage en tête de ligne dans TCP, qui se traduit par la perte de paquets réseau ou le ralentissement des connexions à hautes transactions en raison d'une réorganisation. Par ailleurs, QUIC sépare la connexion de transport de la couche 4 du flux d'adresse IP de la couche 3, permettant ainsi la migration entre différents réseaux sans perturbation.

QUIC est plus à même de prendre en charge une utilisation intensive Internet pour les appareils mobiles caractérisée par le fait que les propriétaires de smartphones passent constamment d'un réseau à l'autre au gré de leurs activités quotidiennes. Ce type d'utilisation d'Internet n'était pas courant lorsque les premières adresses IP ont été développées : les appareils étaient moins portables et ne changeaient pas très souvent de réseau.

Google a commencé à travailler sur une première version de QUIC en 2012. En 2016, il a été adopté par l'Internet Engineering Task Force (IETF), une organisation de normalisation indépendante des fournisseurs, alors qu'elle commençait à créer la nouvelle norme HTTP/3. Après avoir consulté des experts du monde entier, l'IETF a apporté une série de modifications permettant d'élaborer la version désormais standard de QUIC publiée dans le cadre de la [RFC 9000](#).

MISE EN PLACE DE http/3 DANS TRAEFIK

Pour mettre en place le protocole QUIC (http/3) dans Traefik, il faut modifier le fichier statique « **traefik.yml** » que nous avons créé initialement. Pour cela, effectuez les manipulations suivantes :

- Editez le fichier « traefik.yml » avec la commande suivante :

nano /etc/traefik/traefik.yml

- Saisissez ou copiez le contenu suivant :

```
# /etc/traefik/traefik.yml
```

```
log:
```

```
  level: INFO
```

```
api:
```

```
  dashboard: true
```

```
  insecure: false
```

```
entryPoints:
```

```
web:
  address: ":80"
http:
  redirections:
    entryPoint:
      to: websecure
    scheme: https
    permanent: true
```

```
websecure:
  address: ":443"
  http3: {}      # ← active HTTP/3 (QUIC) sur 443
```

Charge la config dynamique (dashboard.yml, headers.yml, tls.yml, cockpit.yml, etc.)

```
providers:
  file:
    directory: "/etc/traefik/dynamic"
    watch: true
```

```
certificatesResolvers:
  le:
    acme:
      email: "gilles.hommet@ik.me"
      storage: "/var/lib/traefik/acme.json"
      httpChallenge:
        entryPoint: web
```

- Quittez et sauvegardez le fichier avec **CTRL + O** et pressez la touche **Entrée**.
- Relancez Traefik avec la commande suivante :

systemctl restart traefik

Note : attention, pour que le protocole QUIC (http/3) soit pris en charge, vous devez également créer une règle DNAT dans votre pare-feu en ouvrant UDP vers Traefik.

Dans notre cas, nous utilisons pfSense et les règles NAT dans « **Port forward** » sont les suivantes pour permettre l'utilisation du protocole « QUIC » (port UDP) :

Port Forward 1:1 Outbound NPT										
Rules										
☐	Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
MACHINE DEBIAN 13 TRAEFIK										
☐	☒		WAN	TCP	*	*	WAN address	80 (HTTP)	10.10.10.14 80 (HTTP)	NAT HTTP VERS TRAEFIK
☐	☒		WAN	UDP	*	*	WAN address	80 (HTTP)	10.10.10.14 80 (HTTP)	NAT ACTIVATION UDP HTTP POUR HTTP3 QUIC TRAEFIK
☐	☒		WAN	TCP	*	*	WAN address	443 (HTTPS)	10.10.10.14 443 (HTTPS)	NAT HTTPS VERS TRAEFIK
☐	☒		WAN	UDP	*	*	WAN address	443 (HTTPS)	10.10.10.14 443 (HTTPS)	NAT ACTIVATION UDP HTTPS POUR HTTP3 QUIC TRAEFIK

L'interface WAN doit afficher les règles autorisant « QUIC » également (protocole UDP) :

Floating WAN LAN											
Rules (Drag to Change Order)											
☐	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
MACHINE DEBIAN 13 TRAEFIK											
☐	✓ 0/0 B	IPv4 TCP	*	*	10.10.10.14	80 (HTTP)	*	none		NAT HTTP VERS TRAEFIK	
☐	✓ 0/0 B	IPv4 TCP	*	*	10.10.10.14	443 (HTTPS)	*	none		NAT HTTPS VERS TRAEFIK	
☐	✓ 0/0 B	IPv4 UDP	*	*	10.10.10.14	80 (HTTP)	*	none		NAT NAT ACTIVATION UDP HTTP POUR HTTP3 QUIC TRAEFIK	
☐	✓ 0/0 B	IPv4 UDP	*	*	10.10.10.14	443 (HTTPS)	*	none		NAT NAT ACTIVATION UDP HTTPS POUR HTTP3 QUIC TRAEFIK	

FICHER « traefik.yml »

```
# /etc/traefik/traefik.yml
log:
  level: INFO

api:
  dashboard: true
  insecure: false

entryPoints:
  web:
    address: ":80"
  http:
    redirections:
      entryPoint:
        to: websecure
        scheme: https
        permanent: true

  websecure:
    address: ":443"
    http3: {}      # ← active HTTP/3 (QUIC) sur 443

# Charge la config dynamique (dashboard.yml, headers.yml, tls.yml, cockpit.yml, etc.)
providers:
  file:
    directory: "/etc/traefik/dynamic"
    watch: true

certificatesResolvers:
  le:
    acme:
      email: "ton_mail@mail.fr"
      storage: "/var/lib/traefik/acme.json"
      httpChallenge:
        entryPoint: web
```

Configuration TLS

tls:

options:

default:

minVersion: "VersionTLS12"

sniStrict: true

preferServerCipherSuites: 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_GCM_SHA256"
- "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
- "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305"

curvePreferences:

- CurveP256
- CurveP384
- CurveP521

```

http:
  routers:
    # existant
  dashboard:
    rule: "Host(`traefik.mon_domaine.fr`) && (PathPrefix(`/api`) || PathPrefix(`/dashboard`))"
    entryPoints: ["websecure"]
    service: api@internal
    tls:
      certResolver: le
    middlewares:
      - auth
      - SecureHeaders
      - Compression

# Router pour la racine /
dashboard-root:
  rule: "Host(`traefik.mon_domaine.fr`) && Path(`/`)"
  entryPoints: ["websecure"]
  service: noop@internal    # service neutre, on applique surtout les middlewares
  tls:
    certResolver: le
  middlewares:
    - SecureHeaders        # => HSTS présent aussi sur /
    - dash-redirect        # => redirige / vers /dashboard/

middlewares:
  auth:
    basicAuth:
      realm: "Traefik"
    users:
      - "admin:mon_mot_de_passe_chiffré_bcrypt"

# redirection
dash-redirect:
  redirectRegex:
    regex: "^/$"
    replacement: "/dashboard/"
  permanent: true

```

http:

middlewares:

Compression:

compress:

excludedContentTypes:
- text/event-stream

SecureHeaders:

headers:

HTTPS/HSTS

forceSTSHeader: true

stsSeconds: 63072000

stsIncludeSubdomains: true

stsPreload: true

X-Frame-Options (garde SAMEORIGIN)

customFrameOptionsValue: "SAMEORIGIN"

Divers sûreté

contentTypeNosniff: true

referrerPolicy: "same-origin"

Permissions-Policy (remplace Feature-Policy)

permissionsPolicy: >-

accelerometer=(), camera=(), geolocation=(),

gyroscope=(), magnetometer=(), microphone=(),

payment=(), usb=()

CSP "compat globale" (Dashboard + Cockpit)

- autorise inline JS/CSS + eval (Cockpit en a besoin)

contentSecurityPolicy: >-

default-src 'self';

script-src 'self' 'unsafe-inline' 'unsafe-eval';

style-src 'self' 'unsafe-inline';

img-src 'self' data: blob;;

font-src 'self' data;;

connect-src 'self' https: wss;;

frame-ancestors 'self';

base-uri 'self';

form-action 'self';

upgrade-insecure-requests

CORS (laisse désactivé si inutile)

accessControlAllowOriginList:

- "https://app.exemple.com"

accessControlAllowMethods:

- "GET"

- "POST"

- "OPTIONS"

accessControlAllowHeaders:

- "Authorization"

- "Content-Type"

accessControlMaxAge: 600

addVaryHeader: true

```
http:
  routers:
    cockpit:
      rule: "Host(`cockpit.mon_domaine.fr`)"
      entryPoints: ["websecure"]
      service: cockpit-svc
      middlewares:
        - SecureHeaders
        - Compression
      tls:
        certResolver: le # Let's Encrypt

services:
  cockpit-svc:
    loadBalancer:
      servers:
        - url: "https://ip_machine_debian_avec_cockpit:9090"
      passHostHeader: true
      serversTransport: cockpit-transport

serversTransports:
  cockpit-transport:
    insecureSkipVerify: true
```