

GUIDE DE MISE EN ŒUVRE APACHE – DOCKER - TRAEFIK

Exposer 2 services web derrière Traefik 3.6.5

Environnement de travail : Debian 13.2



BTS SIO - Option SISR
Administration des systèmes et réseaux

1. Création du docker compose

Dans ce guide nous allons créer 2 conteneurs « Apache » en exposant les ports 8081 et 8082 pour chacun de ces conteneurs. Ces conteneurs seront accessibles depuis le web via Traefik.

- Connectez-vous, en SSH, à votre machine Debian « docker » et exécutez les commandes suivantes :

1^{ère} étape : création de l'arborescence sur la machine « Debian docker »

```
sudo mkdir -p /opt/apache-lab/web1 /opt/apache-lab/web2
cd /opt/apache-lab
```

2^{ème} étape : création du docker-compose (fichier « docker-compose.yml »)

```
sudo nano /opt/apache-lab/docker-compose.yml
```

Contenu à copier (respecter l'indentation) :

```
services:
  web1:
    image: httpd:2.4-alpine
    container_name: web1
    restart: unless-stopped
    ports:
      - "8081:80"
    volumes:
      - ./web1/index.html:/usr/local/apache2/htdocs/index.html:ro

  web2:
    image: httpd:2.4-alpine
    container_name: web2
    restart: unless-stopped
    ports:
      - "8082:80"
    volumes:
      - ./web2/index.html:/usr/local/apache2/htdocs/index.html:ro
```

3^{ème} étape : création d'une page d'accueil simple pour chaque site

```
sudo nano /opt/apache-lab/web1/index.html
```

Contenu à copier pour le site 1 (respecter l'indentation et modifier le contenu en rouge) :

```
<!doctype html>
<html lang="fr">
<head><meta charset="utf-8"><title>WEB1</title></head>
<body>
  <h1>WEB1 - Apache</h1>
  <p>Servi par 192.168.100.110:8081</p>
</body>
</html>
```

sudo nano /opt/apache-lab/web2/index.html

Contenu à copier pour le site 2 (respecter l'indentation et modifier le contenu en **rouge**) :

```
<!doctype html>
<html lang="fr">
<head><meta charset="utf-8"><title>WEB2</title></head>
<body>
  <h1>WEB2 - Apache</h1>
  <p>Servi par 192.168.100.110:8082</p>
</body>
</html>
```

4^{ème} étape : lancement de la « stack »

sudo docker compose up -d

Patientez pendant la création des conteneurs :

```
docker@DEBIAN-DOCKER-PROF:/opt/apache-lab$ sudo docker compose up -d
[+] up 13/13
✓Image httpd:2.4-alpine      Pulled
✓Network apache-lab_default Created
✓Container web1              Created
✓Container web2              Created
```

- Vérifiez, depuis l'interface PortainerCE, que les conteneurs ont bien été créés :

☐	web1	running	   	apache-lab	httpd:2.4-alpine	2025-12-23 18:21:40	172.18.0.3	 8081:80	 administrators
☐	web2	running	   	apache-lab	httpd:2.4-alpine	2025-12-23 18:21:40	172.18.0.2	 8082:80	 administrators

2. Création du fichier de configuration dynamique Traefik

- Connectez-vous, en SSH, à votre machine Traefik pour créer le fichier de configuration dynamique avec la commande suivante :

sudo nano /etc/traefik/dynamic/apache-web1-web2.yml

Copier le contenu suivant (en respectant l'indentation et en modifiant les valeurs en **rouge**) :

```
http:
  routers:
    web1-https:
      entryPoints: ["websecure"]
      rule: "Host(`web1.teamX.sio-ndlp.fr`)"
      service: web1-svc
  tls:
    certResolver: le
```

```
web2-https:
  entryPoints: ["websecure"]
  rule: "Host(`web2.teamX.sio-ndlp.fr`)"
  service: web2-svc
  tls:
    certResolver: le
```

```
services:
  web1-svc:
    loadBalancer:
      servers:
        - url: "http://192.168.100.110:8081"
```

```
web2-svc:
  loadBalancer:
    servers:
      - url: http://192.168.100.110:8082
```

Lancez un navigateur et testez chacune des URL pour vérifier que Traefik redirige bien chaque requête vers le bon site web et qu'un certificat Let's Encrypt a bien été généré :

WEB1 - Apache

Servi par 192.168.100.110:8081

WEB2 - Apache

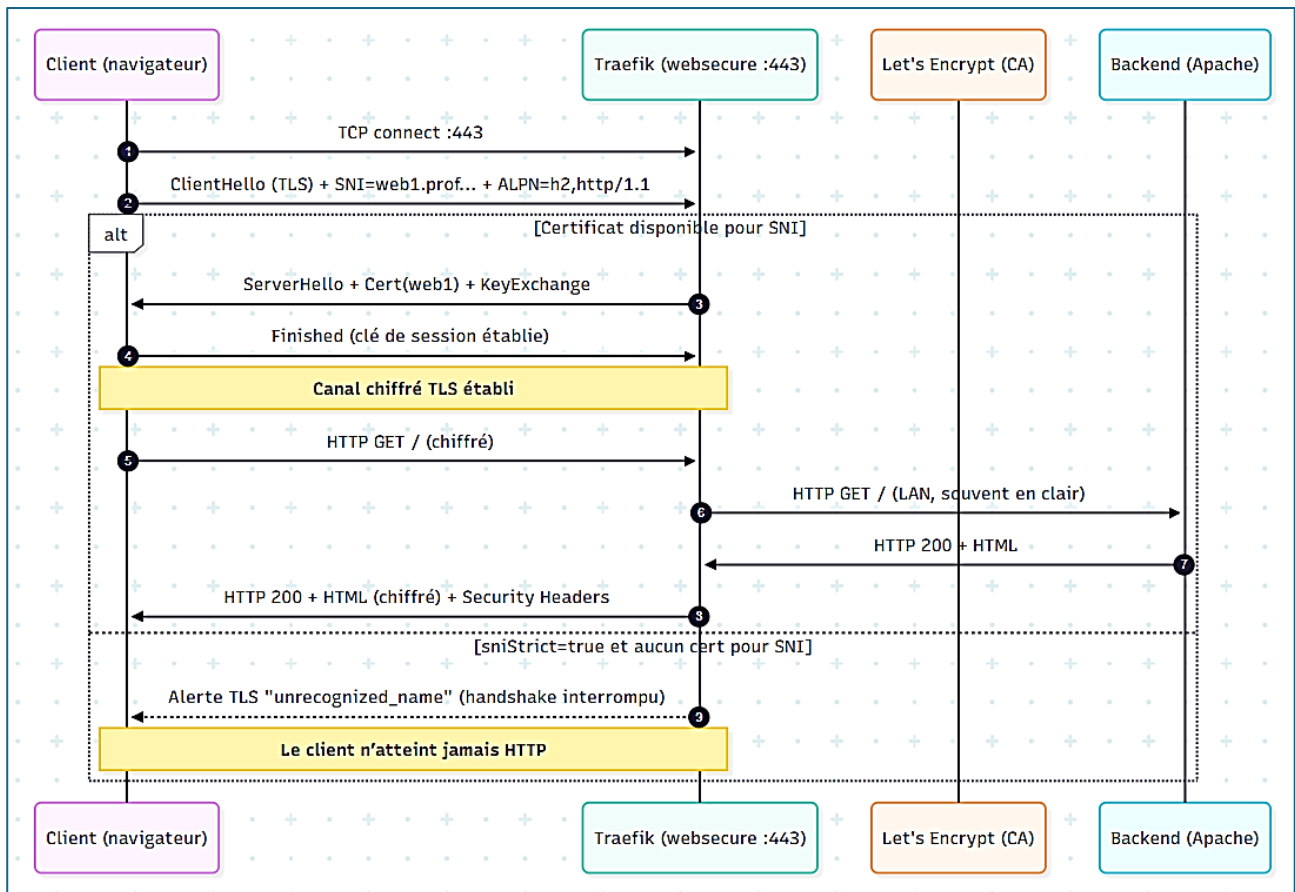
Servi par 192.168.100.110:8082

3. TLS et en-têtes http (headers) avec TRAEFIK

Le reverse proxy Traefik est le **frontal** (« **frontend** ») exposé sur Internet, qui :

- reçoit les connexions HTTP/HTTPS,
- choisit le bon service (on parle de « **routing** »),
- applique des « **middlewares** » (headers, auth, compress),
- et relaie vers des « **backends** » (Apache Docker, Portainer, etc.).

Schéma de fonctionnement général :



On peut résumer ainsi par rapport à notre schéma actuel :

```

Client (navigateur)
| HTTPS (TLS handshake + HTTP chiffré)
v
[ Traefik 192.168.100.109 ]
| HTTP (souvent en LAN, non chiffré) ou HTTPS interne (optionnel)
v
Backend (Apache, Portainer...) 192.168.100.110:8081/8082/...
  
```

TLS : notions essentielles**Les 3 garanties de TLS :**

1. **Confidentialité** : données illisibles pour un écouteur réseau
2. **Intégrité** : modification détectable
3. **Authentification du serveur** : via **certificat** (X.509) valide pour le nom demandé

Certificat, CA et chaîne de confiance :

Un navigateur fait confiance à un serveur si :

- le certificat est **valide** (dates),
- signé par une **CA** de confiance (Let's Encrypt via ACME),
- le nom demandé (**Server Name Indication** ou **SNI**) correspond à un certificat.

```
Certificat serveur (web1.prof.sio-ndlp.fr)
  signé par
Intermédiaire Let's Encrypt
  signé par
Racine (déjà dans le magasin de confiance du navigateur)
```

Sur une même IP:443, Traefik sert plusieurs sous-domaines. Le client indique le nom voulu dès le handshake : c'est le **SNI**.

```
Client -> "Je veux web1.prof.sio-ndlp.fr" (SNI)
Traefik -> choisit le cert correspondant à web1
```

Dans notre fichier « `tls.yml` » nous avons indiqué ceci : **sniStrict: true**

- si Traefik **n'a pas** de certificat pour le SNI demandé → handshake refusé
- effet immédiat : erreur type *unrecognized_name* côté client

La variable « `sniStrict: true` » montre l'exigence "zéro défaut". Cela signifie qu'un sous-domaine sans certificat est inaccessible immédiatement.

ALPN (Application-Layer Protocol Negotiation) :

Permet de négocier le protocole applicatif sur TLS :

- h2 (HTTP/2), http/1.1, et parfois HTTP/3 (QUIC)

Versions TLS, suites cryptographiques, courbes :

- **minVersion** : refuse TLS 1.0/1.1 (obsolètes).
- **cipherSuites** (TLS 1.2) : choix des algorithmes (ECDHE, AES-GCM, CHACHA20...).
- **curvePreferences** : choix des courbes ECDHE (P-256, P-384...).

À retenir :

- TLS 1.2 minimum est un standard raisonnable,
- TLS 1.3 simplifie et sécurise, mais on garde souvent 1.2 pour la compatibilité,
- ECDHE = échange de clés éphémères.

HEADERS : notions essentielles

Un header est une “métadonnée” HTTP. On parle de :

- **Request headers** : envoyés par le client (Host, User-Agent, Accept...)
- **Response headers** : envoyés par le serveur (Content-Type, Cache-Control, etc.)

Avec Traefik, on agit surtout sur les **response headers** pour la sécurité navigateur.

Headers de sécurité majeurs à connaître :

HSTS (Strict-Transport-Security) : le navigateur force HTTPS pour ce domaine (et parfois sous-domaines).

X-Content-Type-Options: nosniff : Empêche certaines interprétations dangereuses de fichiers.

X-Frame-Options : Réduit le clickjacking (iframe).

SAMEORIGIN : autorise l'iframe sur le même site.

Permissions-Policy : Bloque l'accès à caméra, micro, géoloc, etc.

CSP (Content-Security-Policy) : Très puissant : impose d'où viennent scripts/styles/images. Mais il peut casser des applications (Cockpit, Portainer, dashboards...) si trop strict.

Schéma explicatif sur l'obtention d'un certificat Let's Encrypt par Traefik :

