





TRAEFIK

Le projet :

- 1- Créer une application Web en **Html/Css et JavaScript** et une api en ASP.NET Web API 6.
- 2- Installer et configurer le serveur web **Apache2** sur le serveur linux **Debian 192.168.153.131** afin de pouvoir héberger l'app web.
- 3- Installer et configurer le serveur web **Apache2** pour héberger l'app web sur le serveur **CentOS 192.168.153.133**.
- 4- Installer et configurer le serveur web **IIS** sur le contrôleur de domaine **Windows 192.168.153.130**.
- 5- Installer et configurer un serveur **Jenkins** sur **192.168.153.132** qui permettra de déployer et intégrer de façon continue les sources de l'application web sur les différents nœuds (Linux **Debian**, **CentOS** et **Windows**).
- 6- Faire la même chose que (5) mais déployer sur des workers d'un node master kubernetes.
- 7- Configurer **Jenkins** pour déployer automatiquement sur le serveur linux **Debian** et **CentOS** les sources de l'application au moment du merge sur la branche master de **GitHub**.
- 8- Ne pas installer docker sur le serveur linux cible **Debian** mais d'ajouter un plugins **Docker** dans **Jenkins** afin de Builder le **Dockerfile** ou même de lancer le conteneur.
- 9- Créer un serveur de base de données **MSSQL** définit dans le contrôleur de domaine **192.168.153.130**.
- 10- Créer une image pour une **API** web en NET6.0 définit dans un service **Docker**.
- 11- Installer et configurer l'utilitaire **Traefik** permettant de concilier dans le même réseau les deux services **Docker (api et App)**.

-  Caching
-  Répartition de charges

Rédigé par : [Artur Lambo DevOps engineer](#)



SSL/TLS

12- On va utiliser **Traefik** de deux façons :

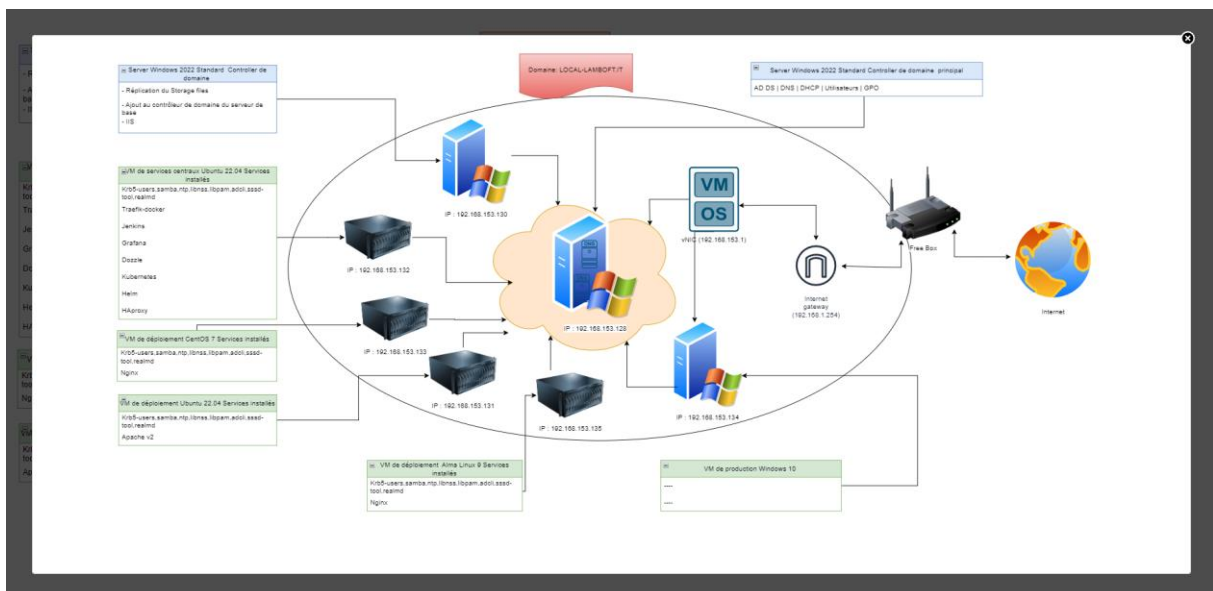
- Avec un fournisseur docker
- Avec un fournisseur kube

13- Installer et configurer un **DOZZLE** permettant de lire les logs des applications isolées dans **Docker**.

14- Utiliser **Ansible** notamment les rôles ansible pour déployer les configurations automatiquement sur le serveur Linux **Debian** et **CentOS**.

15- Installation de **Grafana** pour contrôler et monitorer les **VM**.

16- Plugger **prométhéums** , **Traefik** et **Grafana** ???



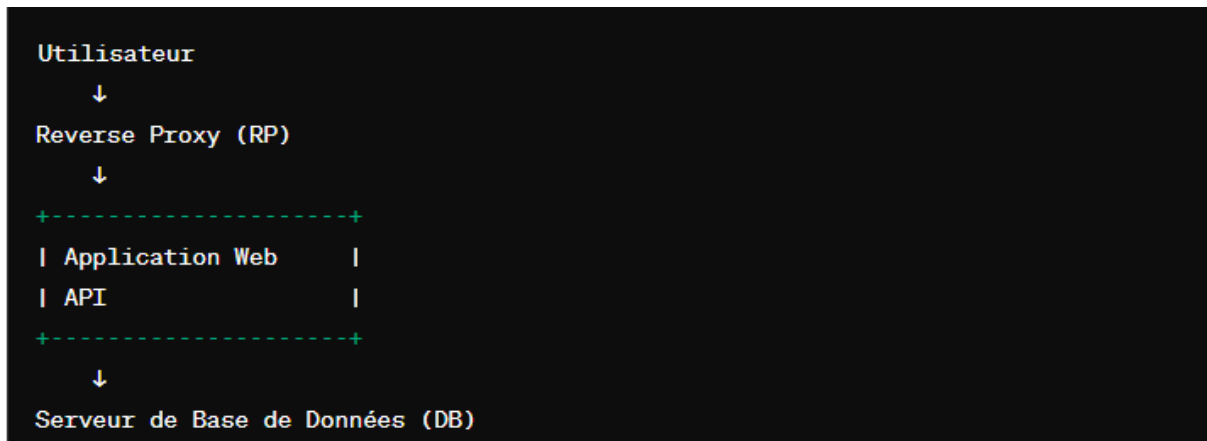
Objectif :

On souhaite mettre entre place une solution nous permettant de **loadbalancer**, de **chiffrer les communications TLS/SSL** et gérer le **caching** entre notre Api et notre application web.

A noter que l'application web et API .net6.0 sont tous les deux des services docker.

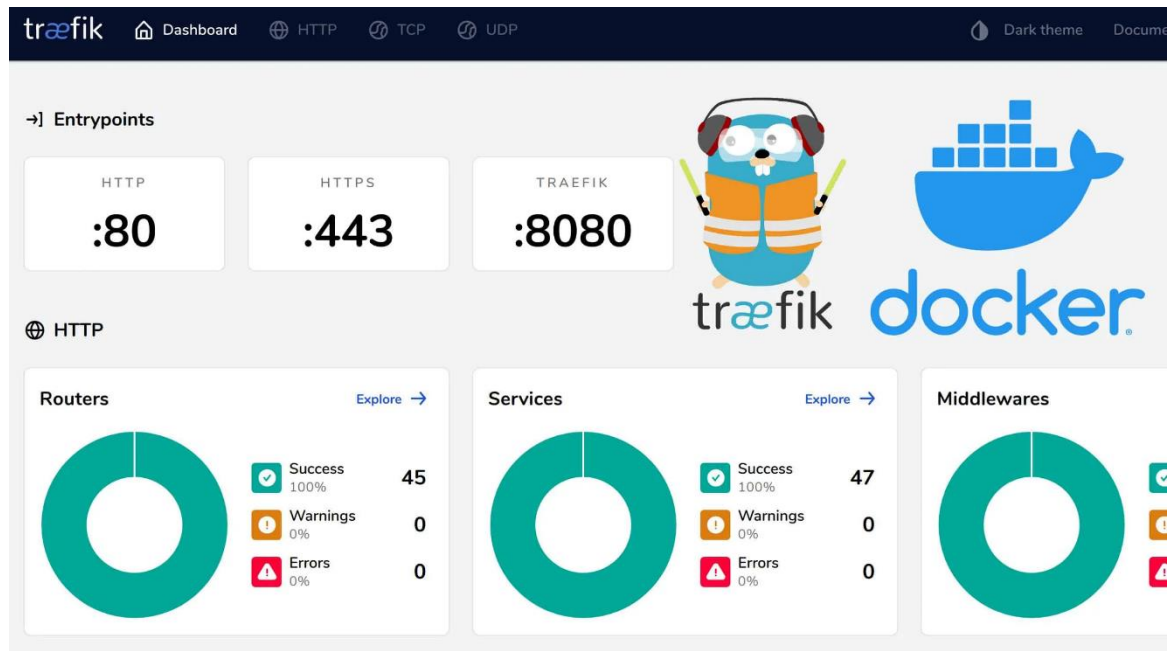
Sachant que seul l'API interagit avec la base de données.

On va utiliser **Traefik** comme RP pour faire cela.



Solution :

Dashboard de Traefik



Apprentissage de Traefik

Rédigé par : [Artur Lambo](#) DevOps engineer

Traefik : C'est un **RP** et un routeur de conteneur qui permet de gérer et d'orchestrer le service de trafic dans les environnements de conteneurs tels que : **Kubernetes** et **docker**

Sagesse : Traefik , par défaut ne contient pas de fichier de configurations on doit éventuellement les créer à la Mano.

On peut installer **Traefik** de plusieurs façons:

- Les binaires pour l'utiliser avec **systemd**
- Dans **Docker**
- Via **helm** pour l'utiliser dans un cluster **Kubernetes**

1 – Via les binaires

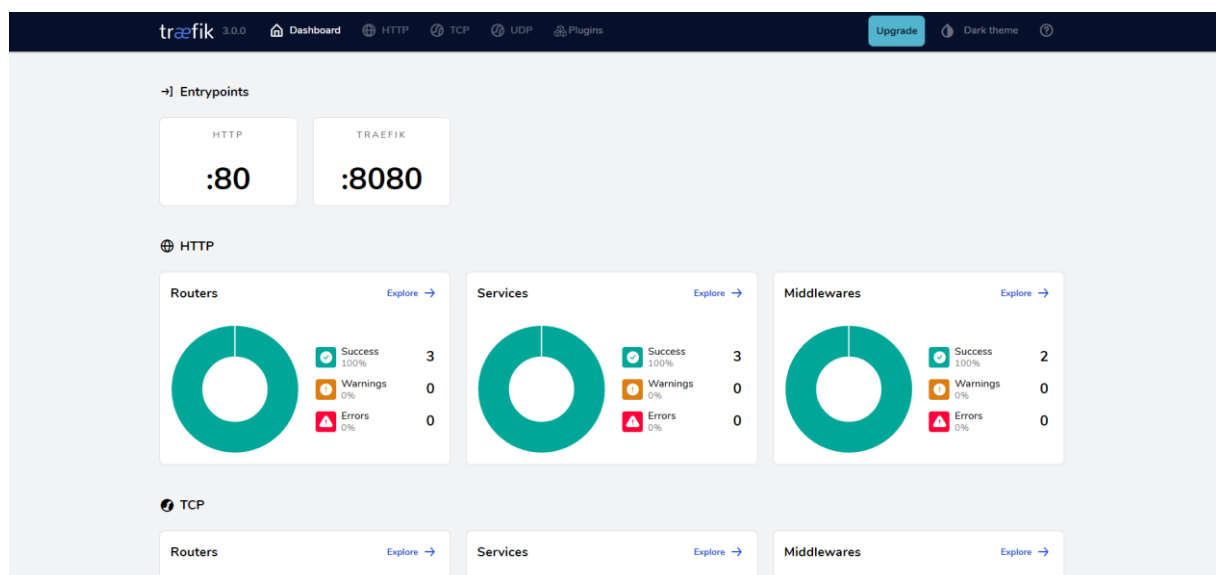
On doit télécharger le fichier **.tar.gz** de Traefik en fonction de l'architecture de son os linux.

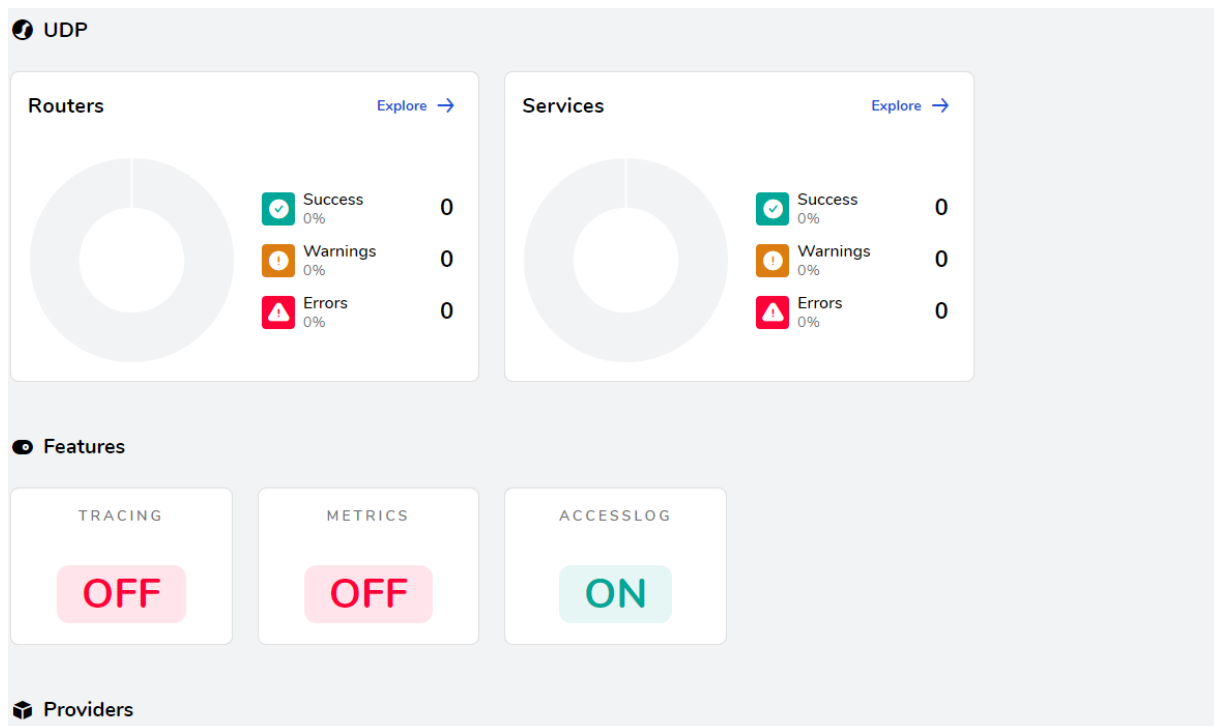
L'extraire du fichier compressé

Déplacer Traefik vers le répertoire abritant les binaires sur la **vm** linux

Le rendre exécutable

```
sn0w@TLA:~/TraefikFiles$ ls
CHANGELOG.md LICENSE.md traefik traefik_v3.0.0_linux_amd64.tar.gz
sn0w@TLA:~/TraefikFiles$ sudo mv traefik /usr/local/bin/traefik
sn0w@TLA:~/TraefikFiles$ sudo chmod +x /usr/local/bin/traefik
sn0w@TLA:~/TraefikFiles$ which traefik
/usr/local/bin/traefik
sn0w@TLA:~/TraefikFiles$ traefik --accesslog=true --api=true --api.dashboard=true --api.insecure=true --api.debug=true --log.level=INFO
```





On peut décider de les créer :

- Via des fichiers de configurations en **Yaml** ou en **Toml** que l'on spécifie lors du démarrage de **Traefik**.
- Utiliser **Traefik** avec **systemd** tout en créant un service pour ce dernier.

Via **systemd** mettre la configuration ci-dessous dans
/etc/systemd/system/traefik.service

```
[Unit]
Description=Traefik Service
After=network.target

[Service]
ExecStart=/usr/local/bin/traefik --configFile=/etc/traefik/traefik.toml
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

```
chown root:root /etc/traefik/traefik.toml
chmod 644 /etc/traefik/traefik.toml
```

Via un fichier de configuration :

Par défaut Traefik tourne sur le port **8080**

NB : Il faut le changer

Pour le lancer en http on a juste besoin de définir la directive API avec **insecure = true**

(à prendre avec les pincettes car il se lance en http et non https)

Insecure = false présuppose qu'on a un certificat TLS/SSL à lui passer pour avoir **Traefik** en **HTTPS**

```
snow@TLA:~/Traefik$ sudo traefik --configFile=.traefik.toml
```

On a créé un fichier de configuration **.traefik.toml** et on charge sa configuration lors du lancement de **Traefik**.

```
[api]
insecure = true
```

Ça c'est la configuration minimale de Traefik en mode non sécurisé (**à ne jamais reproduire en production**)

```
[entryPoints.traefik]
address = ":9091"
```

```
[api]
dashboard = true
insecure = true
```

Ces deux entrées ci-haut sont importantes dans la mesure où elles nous permettent de pouvoir définir le Dashboard de Traefik sur le port 9091 et non sur le port par défaut 8080.

Créer un utilisateur et un groupe dédié à Traefik afin de lui donner la permission pour accéder aux logs.

Cas d'étude :

- ❖ On va faire une redirection de ports 8089 d'une entypoint de **Traefik** vers **Apache2** port **8081** et **8085 de Python**.
- ❖ Mettre en place un middleware d'authentification basique pour le protocole http
- ❖ Mettre en place un middleware header pour la suppression des cookies
- ❖ Mettre en place un middleware de CORS
- ❖ Mettre en place un **Jaeger** pour suivre le trafic dans **Traefik**

NB : Pour un provider Traefik basé sur des fichiers de configuration, on peut utiliser plusieurs directives : directory , filename , templating et Watch (pour dire à Traefik de détecter automatiquement les mises à jour des configurations).

- Installation de **Apache2**
- Provider **Traefik** par un fichier de configuration
- Rajouter les middlewares 1- cookies 2-basicauth

A) Redirection de ports 8089 → 8085 et 8089 → 8081

Dans la configuration statique on a ceci

Rédigé par : [Artur Lambo](#) DevOps engineer

```

[entryPoints]

  [entryPoints.web]
    address = ":80"

  [entryPoints.websecure]
    address = ":443"

  [entryPoints.traefik]
    address = ":9091"

  [entryPoints.http_server]
    address = ":8089"

-

[api]
  dashboard = true
  insecure = true

[log]
  filePath = "/var/log/traefik/access.log"
  format = "json"

[providers]

  [providers.file]
    directory = "/root/.Traefik"
    watch = true

```

On va définir un endpoint pour le service **Python et Apache2** dans le fichier de configuration de **Traefik**

Sur le port 8089

```

[entryPoints.python_server]
  address = ":8089"

[entryPoints.nginx]
  address = ":8089"

```

On va dire à **Traefik** de prendre en considération le provider file dont le filename renvoie à la configuration permettant de rediriger les connexions entrantes sur le port 8089 vers le port 8081 de Apache2 et en même temps de permettre une répartition de charge sur le port 8085 du serveur Python.

```

providers]

[providers.file]
  directory = "/root/.Traefik"
  watch = true

```

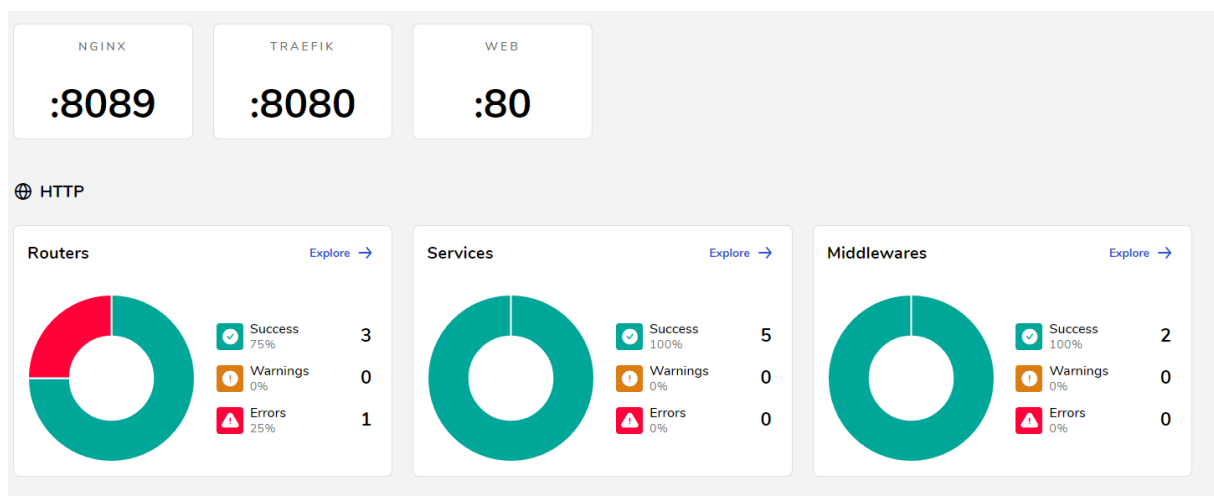
```
[http]
  [http.services]
    [http.services.server_http.loadBalancer]
      [[http.services.server_http.loadBalancer.servers]]
        url = "http://192.168.153.132:8085"
      [[http.services.server_http.loadBalancer.servers]]
        url = "http://192.168.153.132:8081"

  [http.routers]
    [http.routers.to-web_servers]
      entryPoints = ["use_servers"]
      service = "server_http"
      middlewares = ["remove-cookies", "users-authentification"]
      rule = "PathPrefix(`/`)"

  [http.middlewares]
    [http.middlewares.users-authentification.basicAuth]
      users = [
        "python:$2y$05$uj81PRHs6ZegTK7hHD/RPOW4Lt22zMNCkzU/6X6m93vE1Lu6SZJbu",
        "test:$2y$05$.vT0eSkn2JjCWlJYEQyF3e6j6R.xmts9ZBZLAic6T2TjNI79.kLam",
      ]
```

- **http.services** : Répertorie la liste des services écoutant les connexions en http.
- **http.services.server_http.loadBalancer** : Répertorie une instance du service **Apache2** pour laquelle on souhaite appliquer une redirection/équilibre de charge.
- **http.services.server_http.loadBalancer.servers** : Répertorie la liste des serveurs backend sur lesquelles on souhaite faire une redirection .
- **http.routers** : Liste des routes pour des connexions en http.
- **http.routers.to-web_servers** : redirige vers les services en backend
- **rule** : On définit le PATH

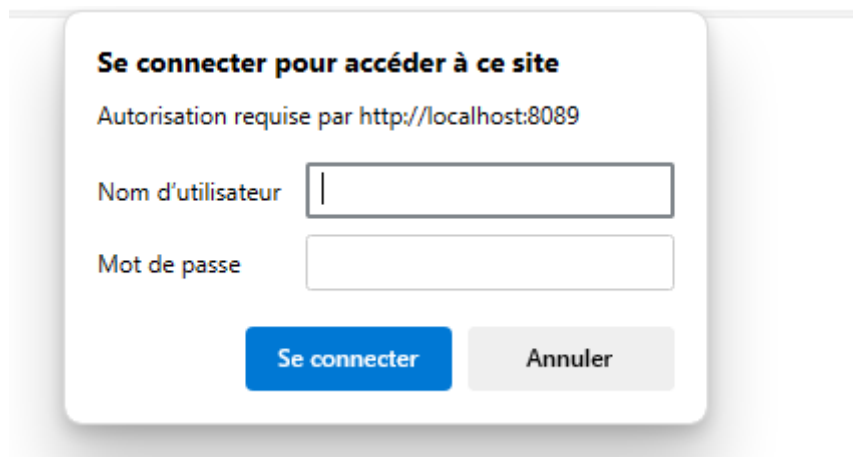
Avec cette configuration Apache2 sera accessible via le port **8089** depuis l'extérieur de notre écosystème.



B) Mise en place d'un middleware d'authentification

Pour une authentification basique, on va :

- Définir le nom du middleware
- Définir le type de hashage du mot de passe des utilisateurs
- Rattacher le middleware à la route pour accéder au service Apache2



```
[http.middlewares]
[http.middlewares.users-authentification.basicAuth]
  users = [
    "python:$2y$05$uj81PRHs6ZegTK7hHD/RPOW4Lt22zMNcKzU/6X6m93vE1Lu6SZJbu",
    "test:$2y$05$.vT0eSkn2JjCWlJYEQyF3e6j6R.xmts9ZBZLAic6T2TjNI79.kLam",
  ]

[http.middlewares.remove-cookies.headers]
[http.middlewares.remove-cookies.headers.customResponseHeaders]
  Set-Cookie = "cookie_name=deleted; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT"
```

En occurrence on a choisi un hashage **BCrypt** qui est le hashage le plus sécurisé.

- Installer apache2-utils : qui va nous permettre de générer un mot de passe encrypté
- `Htpasswd -nB <username>`

Traefik supporte : **MD5**, **BCrypt**, **PBKDF2** et **SHA1**

```
[http.middlewares]
[http.middlewares.users-authentification.basicAuth]
  users = [
    "python:$2y$05$uj81PRHs6ZegTK7hHD/RPOW4Lt22zMNcKzU/6X6m93vE1Lu6SZJbu",
    "test:$2y$05$.vT0eSkn2JjCWlJYEQyF3e6j6R.xmts9ZBZLAic6T2TjNI79.kLam",
  ]

[http.middlewares.remove-cookies.headers]
[http.middlewares.remove-cookies.headers.customResponseHeaders]
  Set-Cookie = "cookie_name=deleted; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT"
```

Pour le même point de terminaison **usecase :8089** on a défini deux serveurs backend

Python 8085 et Apache2 8081

Deux **Middlewares** :

- **remove-cookies** (de type headers) : dans l'entête de la requête http on aura le nom de la cookie, l'action que l'on souhaite faire dessus.
- **users-authentication** : Pour une authentification basique entre le endpoint et le service en backend.

Types de providers de **Traefik** :

- Docker
- Kubernetes
- Files system
- Consul, ETCD (Base de données dans un cluster Kubernetes)

Les **Middlewares** : qui sont des composants intermédiaires entre le Endpoint ou requête (en fonction du protocole http, udp, tcp) du client et les services qui ont pour rôle de modifier ou traiter le trafic avant qu'il atteigne le service voulu.

Les types de middlewares :

- **Authentication** : On peut demander au client de la requête de s'authentifier avant d'avoir un accès au service demandé (via un token jwt, via une authentification basic, via un certificat TLS/SSL)
- **Redirection** : Permettre des redirections des requêtes http vers https
- **Rate limiting** : Limite le nombre de connexions à un service
- **CORS** : Autorise les url d'origines diverses.

<http://localhost:8090/dashboard/> : C'est l'url d'accès au Dashboard **Traefik** sur le port **8080**.

On voit qu'il y'a plusieurs endpoints. Sachant que les endpoints définissent la façon avec laquelle **Traefik** doit écouter et accepter des connexions entrantes.

- **HTTP/S** : Pour la couche applicative : **Traefik** gère le routage des requêtes en http/s on peut voir les routes, les middlewares et les services liés à ces requêtes http/s.
- **TCP** :
- **UDP** :

2 – Dans docker pour permettre l'auto-Discovery

3 – Via Helm pour Kubernetes

Pour le projet on veut Traefik dans un cluster Kubernetes

1. Routage du trafic

Traefik peut diriger le trafic entrant vers les bons pods en fonction des règles que tu définis. Par exemple, il peut rediriger les requêtes HTTP/S vers le pod `app web` pour les pages web et vers le pod `api` pour les appels API.

2. Balanceur de charge

Si tu as plusieurs instances de tes pods `app web` ou `api`, Traefik peut répartir la charge de manière équilibrée entre eux, assurant ainsi une meilleure performance et disponibilité.

3. SSL/TLS Termination

Traefik peut gérer les certificats SSL pour sécuriser les communications avec ton cluster. Il peut automatiquement obtenir et renouveler des certificats SSL/TLS via Let's Encrypt.

4. Monitoring et observabilité

Traefik fournit des métriques et des dashboards intégrés (par exemple avec Prometheus) pour surveiller le trafic réseau, les erreurs, les temps de réponse, etc.

5. Authentification et autorisation

Traefik peut intégrer des middlewares pour gérer l'authentification et l'autorisation, ajoutant ainsi une couche de sécurité supplémentaire à tes services.

A) Installation des prérequis

Sur la VM **yms-002-ubuntu-centerServices** → 192.168.153.132

- Installer Docker : on veut faire de l'auto-Discovery avec docker : version 24.0.5
- Installer Kube version 1.22 : qui nécessite Docker
- Installer Helm version 3.9.4 : c'est le ansible de Kubernetes
- Installer Traefik via Helm

- 1- Installation de Kubernetes
- 2- Installation de docker
- 3- Installation de helm

```
# Télécharger l'archive
wget https://get.helm.sh/helm-v3.9.4-linux-amd64.tar.gz

# Extraire l'archive
tar -zxvf helm-v3.9.4-linux-amd64.tar.gz

# Déplacer l'exécutable
sudo mv linux-amd64/helm /usr/local/bin/helm

# Vérifier l'installation
helm version
```

Nettoyage

Après avoir installé Helm, vous pouvez nettoyer les fichiers téléchargés et extraits :

```
bash Copier le code

rm helm-v3.9.4-linux-amd64.tar.gz
rm -rf linux-amd64
```

Lors de l'installation de Helm un dossier caché est créé dans le répertoire courant nous permettant d'organiser tous les services qui seront installés par helm.

```
root@vms-002-ubuntu-centerServices:~# tree .config/helm/
.config/helm/
├── repositories.lock
└── repositories.yaml

0 directories, 2 files
root@vms-002-ubuntu-centerServices:~#
```

Là on peut voir qu'on a un fichier yaml qui contient l'ensemble des dépôts nécessaires aux téléchargement des services via helm.

On souhaite installer **Traefik** via **helm**

```
apiVersion: ""
generated: "0001-01-01T00:00:00Z"
repositories:
- caFile: ""
  certFile: ""
  insecure_skip_tls_verify: false
  keyFile: ""
  name: traefik
  pass_credentials_all: false
  password: ""
  url: https://helm.traefik.io/traefik
  username: ""
.config/helm/repositories.yaml (END)
```

On voit que là on a l'URL de téléchargement du paquet Traefik

4- Installation de Traefik

1. Exécutez la commande **install** :

```
sudo apt install kubeadm kubelet kubectl
```

```
marko@pnap:~$ sudo apt install kubeadm kubelet kubectl -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 67 not upgraded.
Need to get 81.6 MB of archives.
After this operation, 327 MB of additional disk space will be used.
```

2. Marquez les packages comme retenus pour empêcher l'installation, la mise à niveau ou la suppression automatique :

```
sudo apt-mark hold kubeadm kubelet kubectl
```

```
marko@pnap:~$ sudo apt-mark hold kubeadm kubelet kubectl
kubeadm set on hold.
kubelet set on hold.
kubectl set on hold.
marko@pnap:~$
```

Apt-mark permet de dire que les mises à jour de ces paquets ne sont pas envisageables sur ces services.

Traefik VS HAProxy

Nb : pour exécuter docker sans le sudo

- Add docker in sudo group (sudo addgroup docker)
- Sudo usermod -aG docker <username>
- Rewgrp docker

Après avoir configuré le middleware de certificats **let's encrypt**

Rédigé par : [Artur Lambo DevOps engineer](#)

NB : tjrs sauvegarder le fichier **acme.json** si on ne fait pas de docker

Dans le cas du docker stocker dans un volume

Il faut enregistrer le domaine lamboft.it auprès d'un registra (déjà fait)

III Grandes Finitions

IV Astuces

telnet 192.168.153.132 9091 : elle permet de voir si le port 9091 est ouvert sur la machine
192.168.153.132