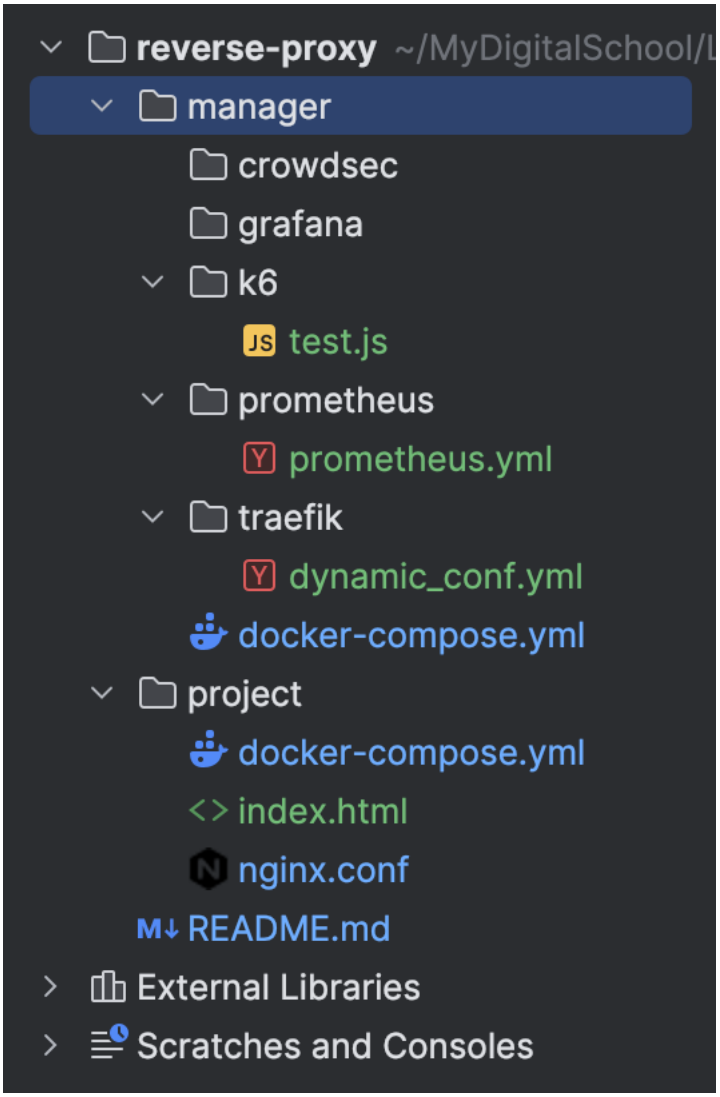
TP Linux individuel – Cahier technique

I. Architecture du TP



Le dossier /manager gère le conteneur pour le reverse proxy et le monitoring du serveur.
Le dossier /project contient un projet HTML simple et un conteneur Nginx avec son fichier de configuration.

J'ai choisi de séparer la gestion du serveur et les projets pour plus de visibilité et de maintenabilité.

II. Configuration du serveur

- Mettre le serveur à jour
  - apt update && apt upgrade

- Modifier le mot de passe root
  - passwd

- Créer un utilisateur, l'ajouter au groupe sudo
  - adduser <user>
  - usermod -aG sudo <user>

- Sécuriser SSH
  - nano /etc/ssh/sshd_config
  - Ajouter ou modifier les lignes :
    - Port <port>
    - AllowUsers <user>
    - PermitRootLogin no

- Installer et configurer Fail2ban
  - Activer le jail SSH

- Installer Docker :
  - Commandes sur https://docs.docker.com/engine/install/debian/

- Installer Ctop :
  - Commandes sur https://github.com/bcicen/ctop

- Ouverture des ports avec IPTables :
  - 8121 : SSH
  - 8890 : Métriques Prometheus
  - 9090 : Prometheus
  - 3000 : Grafana
  - 8086 : InfluxDB
  - 81 : Projet 1

  Ouverture d'un port au public :
  iptables -A INPUT -p tcp --dport <PORT> -j ACCEPT

  Ouverture d'un port à une adresse IP spécifique :
  iptables -A INPUT -p tcp --dport < PORT > -s <IP> -j ACCEPT

III. Création d'un réseau Docker

- docker network create app-network

- Définir le réseau nouvellement créé à tous les conteneurs :

```
networks:
  - "app-network"
```

IV. Création des conteneurs Docker du dossier /manager

À la racine du dossier /manager, dans un fichier docker-compose.yml, créer les conteneurs :

- **Traefik :**

```yaml
services:
  traefik:
    image: traefik:v2.10
    container_name: traefik
    restart: unless-stopped
    command:
      # Traefik configuration
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--providers.file.filename=/etc/traefik/dynamic_conf.yml"
      - "--entrypoints.websecure.address=:443"

      # Metrics configuration
      - "--metrics.prometheus=true"
      - "--metrics.prometheus.buckets=0.100000, 0.300000, 1.200000, 5.000000"
      - "--metrics.prometheus.addEntryPointsLabels=true"
      - "--metrics.prometheus.addServicesLabels=true"
      - "--entryPoints.metrics.address=:8890"
      - "--metrics.prometheus.entryPoint=metrics"
    ports:
      - "443:443" # Port SSL
      - "8890:8890" # Port metrics
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
      - "./traefik/dynamic_conf.yml:/etc/traefik/dynamic_conf.yml"
      - "./private/nikolasdev.cer:/etc/traefik/private/cert.cer" # Certificat SSL
      - "./private/nikolasdev.key:/etc/traefik/private/privkey.key" # Clé privée
    networks:
      - "app-network"
    labels:
      - "traefik.enable=true"
```

Activer les métriques pour Prometheus en renseignant le port 8890.

Le fichier dynamic_conf.yml permet de séparer la logique du reverse-proxy en dehors du docker-compose.yml.

```yaml
http:
  routers:
    traefik-dashboard:
      rule: "Host(`traefik.nikolasdev.com`)"
      entryPoints:
        - websecure
      service: api@internal
      tls: {}

  services:
    traefik-service:
       loadBalancer:
         servers:
           - url: "http://traefik:443"

tls:
  certificates:
    - certFile: "/etc/traefik/private/cert.cer"
      keyFile: "/etc/traefik/private/privkey.key"
```

Le dashboard est accessible sur un nom de domaine renseigné dans rule.
« websecure » indique à Traefik que seul l'accès sécurisé à Traefik est disponible.

Ce fichier permet à Traefik de configurer son accès en SSL, en configurant le certificat et la clé privée.

Fichier dynamic_conf.yml

- **Prometheus :**

```yaml
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    restart: unless-stopped
    volumes:
      - "./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml"
    ports:
      - "9090:9090"
    networks:
      - "app-network"
```

Le fichier prometheus.yml permet de configurer les accès aux métriques :

```yaml
scrape_configs:
  - job_name: 'traefik'
    static_configs:
      - targets: ['traefik:8890']
    scrape_timeout: 60s
  - job_name: 'crowdsec'
    static_configs:
      - targets: ['crowdsec:6060']
    scrape_timeout: 60s
```

Ce fichier permet à Prometheus de récupérer les métriques de Traefik au port 8890, ainsi que de Crowdsec au port 6060.

Avec un timeout de 60 secondes pour récupérer les métriques.

- **Grafana :**

```yaml
grafana:
  image: grafana/grafana
  depends_on:
    - prometheus
  container_name: grafana
  restart: unless-stopped
  volumes:
    - "grafana-storage:/var/lib/grafana"
    - "./grafana/provisioning/:/etc/grafana/provisioning/"
  environment:
    - "GF_SERVER_ROOT_URL=http://www.nikolasdev.com/"
    - "GF_SECURITY_ADMIN_USER=admin"
    - "GF_SECURITY_ADMIN_PASSWORD=admin"
  ports:
    - "3000:3000"
  networks:
    - "app-network"
```

Le dashboard Grafana est disponible sur le port 3000 sur le nom de domaine indiqué dans la variable d'environnement : GF_SERVER_ROOT_URL.

Un volume[1] « grafana-storage » est nécessaire pour stocker ses données persistantes et sa configuration

[1] Voir « Configurer le réseau Docker »

- **InfluxDB :**

```yaml
influxdb:
  image: influxdb:1.8
  container_name: influxdb
  ports:
    - "8086:8086"
  volumes:
    - influxdb-data:/var/lib/influxdb
  environment:
    - INFLUXDB_DB=k6
    - INFLUXDB_ADMIN_USER=admin
    - INFLUXDB_ADMIN_PASSWORD=admin
    - INFLUXDB_HTTP_AUTH_ENABLED=true
  networks:
    - "app-network"
```

J'utilise InfluxDB sur le port 8086 pour collecter, stocker et analyser des résultats de tests de performance réalisés avec l'outil K6.

Création du volume « influxdb-data » pour stocker ces données.

L'accès à la base de données est sécurisée grâce aux identifiants de connexion stockés dans des variables d'environnement.

- **K6 :**

```yaml
k6:
  image: grafana/k6:latest
  container_name: k6
  restart: on-failure
  command: run /scripts/test.js --out influxdb=http://admin:admin@influxdb:8086/k
  depends_on:
    - influxdb
  entrypoint: ["k6"]
  volumes:
    - "./k6:/scripts"
  networks:
    - "app-network"
```

K6 permet de tester à l'aide d'un script, une montée en charge du serveur.

La commande permet d'exécuter le script test.js et d'envoyer les résultats à la base de données disponible à l'adresse : http://admin:admin@influxdb:8086/k6.
« Depends_on » va spécifier que le conteneur InfluxDB est opérationnel avant l'exécution des tests.

- **Crowdsec :**

```yaml
crowdsec:
  image: crowdsecurity/crowdsec
  container_name: crowdsec
  restart: always
  command:
    - "cscli collections install crowdsecurity/nginx"
  environment:
    COLLECTIONS: "crowdsecurity/nginx"
    GID: "${GID-1000}"
  volumes:
    - "logs:/var/log/nginx:rw"
    - "./crowdsec/crowdsec-db:/var/lib/crowdsec/data:rw"
    - "./crowdsec/crowdsec-config:/etc/crowdsec:rw"
    - "./crowdsec/acquis.yaml:/etc/crowdsec/acquis.yaml"
    - "/var/run/docker.sock:/var/run/docker.sock"
  networks:
    - "app-network"
```

Surveillance des logs d'Nginx[2] grâce à la commande :

« cscli collections install crowdsecurity/nginx »

NB : Le conteneur Nginx est situé dans un autre dossier et un autre docker-compose.yml que celui du manager.
Les logs d'Nginx sont stockés par Docker, et non dans le conteneur Nginx situé dans /var/log

Pour corriger ce problème, mettre le conteneur Nginx dans le même docker-compose.yml que celui de Crowdsec.

[2] Voir « Création du conteneur du projet »

V. Création du conteneur Nginx pour le projet

```yaml
services:
  nginx:
    image: nginx:alpine
    container_name: nginx
    restart: unless-stopped
    ports:
      - "81:443"
    volumes:
      - "./index.html:/usr/share/nginx/html/index.html"
      - "./nginx.conf:/etc/nginx/conf.d/default.conf"
      - "manager_logs:/var/log/nginx:rw"
    networks:
      - "app-network"
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.nginx-router.rule=Host(`nikolasdev.com`)"
      - "traefik.http.routers.nginx-router.entrypoints=websecure"
      - "traefik.http.routers.nginx-router.tls=true"
      - "traefik.http.services.nginx-service.loadbalancer.server.port=443"

volumes:
  manager_logs:
    external: true

networks:
  app-network:
    external: true
```

Le projet HTML est disponible en HTTPS à l'adresse spécifié par nginx-router.rule sur le port 81:443.
Activation du reverse-proxy Traefik avec traefik.enable.
Configuration d'Nginx avec le fichier nginx.conf.

Dans cette configuration, je n'ai pas besoin de renseigner le port :81 à l'adresse du projet.

VI. Volumes

```
volumes:
  influxdb-data:
  grafana-storage: {}
  logs:
  crowdsec-db:
  crowdsec-config:
```

Les volumes utilisés sont à définir dans les docker-compose.yml

```
volumes:
  logs:
    external: true
```

VII. Configurer le réseau Docker

```
networks:
  app-network:
    external: true
```

Le réseau « app-network » doit être indiquer dans chaque docker-compose.yml.
External à true permet aux conteneurs de communiquer sur un réseau externe, partagé.

VIII. Lancement/Arrêt des conteneurs

Pour lancer les conteneurs Docker, se placer dans un répertoire avec le docker-compose.yml.

- « docker compose up -d » : Va télécharger les images, créer les volumes et exécuter les conteneurs indiqués dans le docker-compose.yml.

- « docker compose down » : Stoppe l'exécution des conteneurs