TP_hebergement_ssh

Hébergement SSH avec Docker

Objectifs

- Comprendre le protocole SSH et son rôle dans l'administration d'un serveur distant.
- Mettre en place un serveur SSH dans un conteneur Docker et s'y connecter depuis un client dédié.
- Manipuler l'authentification par mot de passe puis par clé publique.
- Automatiser la configuration du serveur via un `Dockerfile`.

Prérequis

- Avoir réalisé le TP d'introduction à Docker (images, conteneurs, ports, volumes).
- Disposer de Docker fonctionnel sur votre machine.
- Savoir utiliser un terminal Bash, `zsh`, `fish` ou PowerShell.
- Aucune connaissance préalable de SSH n'est nécessaire.

Résultats attendus

- Deux images Docker (`ssh-server`, `ssh-client`) construites à partir de vos Dockerfiles.
- Une connexion SSH fonctionnelle entre deux conteneurs reliés par un réseau Docker interne.
- Un rapport court décrivant le protocole SSH, les modes d'authentification et les commandes utilisées.

Architecture du TP

Vous simulerez une petite infrastructure distante composée de :

- Un conteneur `ssh-server` basé sur Ubuntu et exposant `openssh-server` sur le port 22.
- Un conteneur `ssh-client` utilisé pour initier la connexion SSH.
- Un réseau Docker interne (`ssh-net`) pour relier les deux conteneurs.

Un dossier de travail unique est recommandé, par exemple : `~/workspace/tp-hebergement-ssh`.

Étape 0 — Préparer l'environnement Docker

1. Créez le dossier de travail et placez-vous dedans :

```
mkdir -p ~/workspace/tp-hebergement-ssh && cd ~/workspace/tp-hebergement-ssh
```

2. Créez le réseau Docker qui servira de lien entre les conteneurs :

```
docker network create ssh-net
```

3. Vérifiez la présence du réseau :

```
docker network ls
```

Étape 1 — Construire l'image du serveur SSH

1. Créez un fichier `Dockerfile.server` contenant :

```
FROM ubuntu:24.04

RUN apt update \
    && apt install -y openssh-server sudo \
    && mkdir /var/run/sshd

# Crée un utilisateur non-root "student"
RUN useradd -m student \
    && echo "student:password" | chpasswd \
    && adduser student sudo

# Active l'authentification par mot de passe (temporaire pour le TP)
RUN sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config \
    && sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin no/' /etc/ssh/sshd_config

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]
```

2. Construisez l'image Docker du serveur :

```
docker build -t ssh-server -f Dockerfile.server .
```

3. Lancez le conteneur serveur en arrière-plan :

```
docker run -d --name ssh-server --network ssh-net ssh-server
```

4. Vérifiez que le service SSH est bien démarré :

```
docker exec -it ssh-server ps aux | grep sshd
```

Étape 2 — Construire l'image du client SSH

1. Créez un fichier `Dockerfile.client` :

```
FROM ubuntu:24.04

RUN apt update \
    && apt install -y openssh-client vim iputils-ping

WORKDIR /root
CMD ["bash"]
```

2. Construisez puis lancez le conteneur client :

```
docker build -t ssh-client -f Dockerfile.client .
docker run -it --rm --network ssh-net --name ssh-client ssh-client
```

3. Depuis le terminal du client, vérifiez la connectivité réseau :

```
ping -c 2 ssh-server
```

Étape 3 — Connexion SSH par mot de passe

1. Depuis le terminal du client, initiez une connexion SSH :

```
ssh student@ssh-server
```

2. Acceptez l'empreinte de la clé du serveur puis saisissez le mot de passe `password`.
3. Vérifiez que vous êtes bien connecté sur le serveur :

```
whoami
hostname
```

Étape 4 — Connexion SSH par clé publique

1. Toujours depuis le conteneur client, générez une paire de clés RSA :

```
ssh-keygen -t rsa -b 4096 -C "student@docker"
```

   Appuyez sur Entrée pour accepter les options par défaut.
2. Copiez la clé publique vers le serveur :

```
ssh-copy-id student@ssh-server
```

3. Testez la connexion sans mot de passe :

```
ssh student@ssh-server
```

4. Vérifiez sur le serveur que la clé a été enregistrée :

```
cat ~/.ssh/authorized_keys
```

Étape 5 — Sécuriser le serveur et automatiser la configuration

1. Modifiez `Dockerfile.server` pour désactiver l'authentification par mot de passe :

```
RUN sed -i 's/PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config
```

2. Reconstruisez et relancez le conteneur serveur :

```
docker stop ssh-server && docker rm ssh-server
docker build -t ssh-server:secure -f Dockerfile.server .
docker run -d --name ssh-server --network ssh-net ssh-server:secure
```

3. Vérifiez que la connexion fonctionne uniquement via la clé :

```
ssh student@ssh-server
```

Étape 6 — Observation et nettoyage

1. Listez les conteneurs actifs :

```
docker ps
```

2. Consultez les derniers logs SSH du serveur :

```
docker logs ssh-server | tail -n 10
```

3. Nettoyez l'environnement :

```
docker stop ssh-server
docker rm ssh-server
docker network rm ssh-net
docker image prune
```

Livrables

- Les fichiers `Dockerfile.server` et `Dockerfile.client` utilisés durant le TP.
- Une capture d'écran prouvant la connexion SSH réussie.
- Un court rapport (Markdown) précisant :
  - le fonctionnement du protocole SSH,
  - la différence entre authentification par mot de passe et par clé,
  - les commandes Docker exécutées,
  - le rôle du port 22 et la notion de chiffrement.

Pour aller plus loin (optionnel)

- Configurer un port SSH non standard (ex. 2222) et adapter les commandes de connexion.
- Tester un transfert de fichiers sécurisé avec `scp` :

```
scp fichier.txt student@ssh-server:/home/student/
```

- Déployer un petit serveur web et l'administrer via SSH.
- Expérimenter avec un conteneur léger (ex. Alpine + dropbear).