Utilisation de Docker sur CentOS Stream 8

Docker est un outil de création de conteneurs qui permet facilement de créer des environnements d'exécution qui soient plus simples et moins coûteux que des machines virtuelles.

1 Installation et premiers essais

Placez-vous sur la VM serveur et assurez-vous d'avoir traité le TP sur l'agrandissement [1].

Commençons par installer **Docker** :

```
# ajout du dépôt docker
dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

# installation de docker
dnf -y install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# démarrer Docker et l'activer
systemctl start docker
systemctl enable docker
```

Continuons avec quelques exemples (exécutez les commandes une à une) :

1. tp-agrandir.html

```
## charger l'image de centos
docker pull tgagor/centos:stream8

## lister les images
docker images

## Afficher Hello
docker run tgagor/centos:stream8 /bin/echo Hello

## Lancer un conteneur en mode interactif
docker run -it tgagor/centos:stream8 /bin/bash
[root@92afaafe4463 /]# uname -a
Linux 92afaafe4463 4.18.0-448.el8.x86_64 ...
[root@92afaafe4463 /]# cat /etc/redhat-release
CentOS Stream release 8
[root@92afaafe4463 /]# exit

## Visualiser les conteneurs (-a pour voir les stoppés)
docker ps -a

## Supprimer les conteneurs créés
docker rm $(docker ps -a -q)

## Lancer un conteneur en arrière plan
docker run -itd tgagor/centos:stream8 /bin/bash

## Repérer son ID
docker ps

## Se connecter à ce conteneur
docker exec -it 864ddc9c3821 /bin/bash
[root@864ddc9c3821 /]# echo hello > /tmp/hello.txt
[root@864ddc9c3821 /]# exit

## Vérifier que c'est bien le même
docker exec -it 864ddc9c3821 /bin/bash -c "cat␣/tmp/hello.txt"

## Le stopper et le supprimer
docker stop 864ddc9c3821
docker rm 864ddc9c3821
```

2 Ajout de conteneurs et accès aux services

Continuons en créant et sauvegardant des images :

```
## Créer un conteneur avec httpd
docker run tgagor/centos:stream8 dnf -y install httpd
id=$(docker ps -a -q | head -1)

## Créer une image
docker commit $id my-httpd
docker images

## Créer un conteneur avec httpd activé (et redirection des ports)
docker run -dt -p 8800:80 my-httpd /usr/sbin/apachectl -D FOREGROUND
id=$(docker ps -a -q | head -1)

## Changer la page par défaut
docker exec $id /bin/bash -c 'echo␣httpd␣on␣docker␣>␣/var/www/html/index.html'

## Vérifier
curl localhost:8800

# Récupérer des informations
docker inspect $id
```

3 Utiliser des Dockerfile

Nous allons maintenant créer des images de manière automatisée en utilisant des `Dockerfile`.

3.1 Une image pour java

```
## Créer un répertoire
mkdir java-docker

## Créer le dockerfile
cat > java-docker/Dockerfile <<FIN

FROM tgagor/centos:stream8
MAINTAINER JeanLucMassat <jean-luc.massat@univ-amu.fr>
RUN dnf -y install java-17-openjdk
RUN dnf -y clean all

FIN

## Construire l'image
docker build -t my-java:latest java-docker
docker images

## Tester l'image
docker run my-java:latest java -version
```

3.2 Une image pour une appli Spring-Boot

Nous pouvons maintenant créer une image pour déployer notre application Spring-Boot :

```
## Créer un répertoire
mkdir myapp-docker

## Préparer l'application dans le répertoire
wget https://jean-luc-massat.pedaweb.univ-amu.fr/ens/asr/spring-app.zip
unzip spring-app.zip
mv -v spring-app.war myapp-docker

## Créer le dockerfile
cat > myapp-docker/Dockerfile <<FIN

FROM my-java:latest
MAINTAINER JeanLucMassat <jean-luc.massat@univ-amu.fr>
WORKDIR /app
COPY spring-app.war /app/spring-app.war
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "/app/spring-app.war"]

FIN

## Construire l'image
docker build -t my-app:latest myapp-docker
docker images

## Tester l'image en mode interactif (la stopper avec Control-C)
docker run -it -p 9000:8081 my-app

## Tester l'image en mode démon
docker run -dt -p 9000:8081 my-app
# pour attendre le demarrage de l'application
sleep 25s
curl http://localhost:9000/movies
```

4 Déployer avec une BD MySQL

Commençons par supprimer tous nos conteneurs :

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
```

Continuons en chargeant une image MySQL :

```
docker pull docker.io/library/mysql
docker images
```

Créons un réseau qui va permettre de relier les conteneurs MySql et Spring-Boot :

```
docker network create my-database
```

Créons et configurons le conteneur qui héberge notre BD :

```
docker run --name mysqldb --network my-database \
    -e MYSQL_ROOT_PASSWORD=root \
    -e MYSQL_USER=moviesuser \
    -e MYSQL_PASSWORD=moviespass \
    -e MYSQL_DATABASE=moviesdb -d mysql
mysqlId=$(docker ps -a -q | head -1)
```

Créons maintenant l'image de notre application :

```
## Créer un répertoire
mkdir myappmysql-docker

## Préparer l'application dans le répertoire
wget https://jean-luc-massat.pedaweb.univ-amu.fr/ens/asr/spring-app.zip
unzip spring-app.zip
mv -v spring-app.war myappmysql-docker

## Créer le dockerfile
cat > myappmysql-docker/Dockerfile <<FIN

FROM my-java:latest
MAINTAINER JeanLucMassat <jean-luc.massat@univ-amu.fr>
WORKDIR /app
COPY spring-app.war /app/spring-app.war
COPY start.sh /app/start.sh
EXPOSE 8081
ENTRYPOINT ["/bin/bash", "/app/start.sh"]

FIN

## Créer le script de démarrage
cat > myappmysql-docker/start.sh <<FIN
java \
    -Dspring.datasource.driver-class-name=com.mysql.jdbc.Driver \
    -Dspring.datasource.url=jdbc:mysql://mysqldb:3306/moviesdb \
    -Dspring.jpa.generate-ddl=true \
    -Dspring.jpa.hibernate.ddl-auto=update \
    -Dspring.datasource.username=moviesuser \
    -Dspring.datasource.password=moviespass \
    -jar spring-app.war
FIN

## Construire l'image
docker build -t my-app-mysql:latest myappmysql-docker
docker images

## Tester l'image en mode interactif (la stopper avec Control-C)
docker run -it -p 9000:8081 --network my-database my-app-mysql:latest

## Tester l'image en mode démon
docker run -dt -p 9000:8081 --network my-database my-app-mysql:latest
# pour attendre le demarrage de l'application
sleep 25s
curl http://localhost:9000/movies
```

Vous pouvez maintenant vérifier que les tables du conteneur MySql sont bien à jour :

```
docker exec -it $mysqlId /bin/bash
bash-4.4# mysql -u moviesuser -p
USE moviesdb;
SELECT * FROM movie;
exit
bash-4.4# exit
```

Dernière vérification :

- Modifiez les données dans votre application (en passant par un tunnel ssh).
- Stoppez le conteneur Spring-Boot.
- Relancez-le et vérifiez que les données sont bien les mêmes.