

# RAPPORT du TP docker

Thomas PLOIX

## Database

*Why should we run the container with a flag -e to give the environment variables?*

Car ce sont des données sensibles ou des données d'environnement propre à l'utilisateur, elles sont donc privées et ne doivent pas être partager directement dans le Dockerfiles ou autre fichier.

*Why do we need a volume to be attached to our postgres container?*

Cela nous permet de garder nos données de la base de données si jamais le container plante, crash. On pourra relancer le container et réavoir les données sauvegarder, c'est la persistance des données.

Commande utilisée :

```
docker build -t Tploix/database .
```

➔ Création de l'image Tploix/database depuis le dockerfile du répertoire

```
docker run -e POSTGRES_DB=db -e POSTGRES_USER=usr -e POSTGRES_PASSWORD=pwd --  
name=database --net=app-network -v /my/own/datadir:/var/lib/postgresql/data  
Tploix/database
```

➔ Lancement du conteneur database depuis l'image Tploix/database, avec des variables d'env, un network et un volume attaché.

```
docker rm -f database
```

➔ Suppression du conteneur database

Dockerfile :

```
FROM postgres:14.1-alpine  
COPY /docker-entrypoint-initdb.d/* /docker-entrypoint-initdb.d/
```

## Backend API

*1-2 Why do we need a multistage build? And explain each step of this dockerfile.*

On utilise un multistage build car on doit le faire en deux fois. Une première partie pour Build le projet avec le téléchargement de toutes les dépendances Maven utilisées par notre projet. Puis on utilise Java JDK, pour compiler le code en Bytecode.

Ensuite la deuxième partie on va utiliser Java JRE qui est le Run Time Environment. Il va récupérer le bytecode qui est le résultat du stage précédent pour le compiler et le lancer.

```
# Build
FROM maven:3.8.6-amazoncorretto-17 AS myapp-build
On utilise l'image de base de Maven qui contient un JDK qu'on appelle myapp-build

ENV MYAPP_HOME /opt/myapp
On pose une variable d'environnement pour l'utiliser plus tard

WORKDIR $MYAPP_HOME
On se place dans le nouvel emplacement de travail

COPY pom.xml .
On copie le pom.xml qui contient toutes les configurations et dépendances de notre projet

Run mvn dependency:go-offline
On run la commande qui permet de checker les dépendances du pom.xml, il va télécharger les
dépendances manquantes si jamais

COPY src ./src
On copie tous nos fichiers utilisés par notre projet

RUN mvn package -DskipTests
On run la commande pour compiler notre projet et créer l'artefact

# Run
FROM amazoncorretto:17
On utilise l'image contenant un Java JRE

ENV MYAPP_HOME /opt/myapp
WORKDIR $MYAPP_HOME
On se place dans l'emplacement de travail

COPY --from=myapp-build $MYAPP_HOME/target/*.jar $MYAPP_HOME/myapp.jar
On récupère l'artefact créé par le stage précédent qui contient toute notre application en bytecode.

ENTRYPOINT java -jar myapp.jar
On lance la commande Entrypoint pour Run notre application
```

Commande utilisée :

`docker build -t Tplox/backend .`

```
docker run --env-file .env --name=backend --net=app-network -p 8080:8080 Tploix/backend
```

On utilise un fichier d'environnement qui contient nos variables d'environnement ; On ouvre les ports 8080 :8080 pour communiquer entre le conteneur et nous. On utilise le network pour « établir une ligne de communication » entre le backend et la base de données.

## Frontend

DockerFile :

```
FROM httpd:2.4

COPY ./public-html/ /usr/local/apache2/htdocs/

COPY proxy.conf /tmp/proxy.conf

RUN cat /tmp/proxy.conf >> /usr/local/apache2/conf/httpd.conf
```

Proxy.conf :

```
<VirtualHost *:80>
ServerName localhost

ProxyPreserveHost On
ProxyPass /api/ http://backend:8080/
ProxyPassReverse /api/ http://backend:8080/
</VirtualHost>
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

*Why do we need a reverse proxy?*

Le Reverse proxy va nous permettre de faire un portail entre le client qui veut accéder à notre application et le front end /backend. Il peut nous permettre d'ajouter des règles de connexion, de mapper des url Spécifiques

## Docker-compose

*Why is **docker-compose** so important?*

Le docker compose permet de tout paramétrer dans un seul fichier, de gérer le build des images, les variables d'environnements, les networks et les volumes

*1-3 Document docker-compose most important commands.*

docker compose up

permet de lancer tous les services qui sont décrit dans le docker-compose.yml

on peut rajouter --build qui va forcé le build des images

docker compose down

permet de kill les instances des conteneurs

On peut rajouter -V pour supprimer les volumes en même temps

#### 1-4 Document your docker-compose file.

```
version: '3.7'

services:
  backend:
    build: "/backend/."
    env_file:
      - .env
    networks:
      - my-network
    depends_on:
      - database
  database:
    build: "/database/."
    env_file:
      - .env
    networks:
      - my-network
    volumes:
      - db-data:/var/lib/postgresql/data
  httpd:
    build: "/Frontend/."
    ports:
      - "80:80"
    networks:
      - my-network
    depends_on:
      - backend

networks:
  my-network:
  volumes:
    db-data:
```

On utilise les variables d'environnement contenu dans le .env

On connecte avec le network de l'application

On a besoin que la database soit lancée avant de lancer le backend


On créer notre volume qu'on a défini plus bas, c'est un named volume



On créer le network qui va relier tous nos services entre eux

On définit le volume qu'on va utiliser pour notre base de donnée

## Publish


#### 1-5 Document your publication commands and published images in dockerhub.




**Tploix**  
 Community User  Joined June 4, 2024

Repositories


Starred

 Search by repository name


Displaying 1 to 3 of 3 repositories



**tploix/backend**  
By [tploix](#) · Updated an hour ago



**tploix/frontend**  
By [tploix](#) · Updated an hour ago



**tploix/database**  
By [tploix](#) · Updated an hour ago

Commandes utilisées :

Database :

```
docker tag Tploix/database tploix/database:1.0
```

```
docker push tploix/database:1.0
```

Backend :

```
docker tag Tploix/backend tploix/ backend:1.0
```

```
docker push tploix/ backend:1.0
```

Frontend :

```
docker tag Tploix/frontend tploix/ frontend:1.0
```

```
docker push tploix/ frontend:1.0
```

*Why do we put our images into an online repo ?*

En mettant nos images dans un répo en ligne, nous les rendons disponibles et on permet à n'importe qui de pull nos images et les utiliser. On les rends aussi disponibles pour les déployer sur nos serveurs si on en a.

## Bonus :

Utilisation de fichier d'environnement dans le .env OK

Utilisation de Volume pour la persistance des données. OK