Séance 2

vendredi 18 novembre 2022 08:21

Commandes intéressantes :

docker **exec** : tapper des commandes dans le docker directement
docker exec -it hopeful_goodall **bash** : ouvrir un terminal sur le conteneur
psql -U databaseUser -W myDatabase : on se connecte sur la base de donnée
\q : quitter la bdd
exit : quitter le terminal

docker inspect hopeful_goodall : avoir toutes les infos sur le conteneur (json)

docker run -p 5432:5432 : Mapping de ports : on peut lier un conteneur au localhost de notre machine

EXPOSE : ajoute des infos (visibles avec inspect)

Projet bibliothèque :

```
FROMmaven:3.8-openjdk-18

WORKDIRusr/src/app

COPY..

CMD["mvn","spring-boot:run"]
```

Prob : outils orienté dev -> surcouche outillage non nécessaire pour démarrer l'appli + le mvn run compile, dl les dépendances et run le projet
On peut remplacer les action par un fat-jar (créable avec docker aussi) et le run avec java

Run le fatjar grâce à java (fait à chaque run)

```
FROM maven:3.8-openjdk-18asbuilder
WORKDIR usr/src/app
COPY ..
RUN mvn cleanpackage-DskipTests=true
```

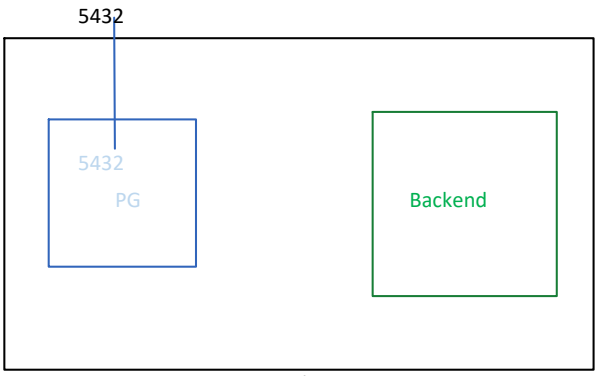
Création fat jar grâce à mvn (fait que au build)

```
FROM openjdk:18-jdk
WORKDIR usr/src/app
COPY-- from=builderusr/src/app/target/fs-back-*.jarproject.jar
CMD ["java","-jar","project.jar"]
```

Build : `docker build --tag projet`

Run : `docker run --rm projet`

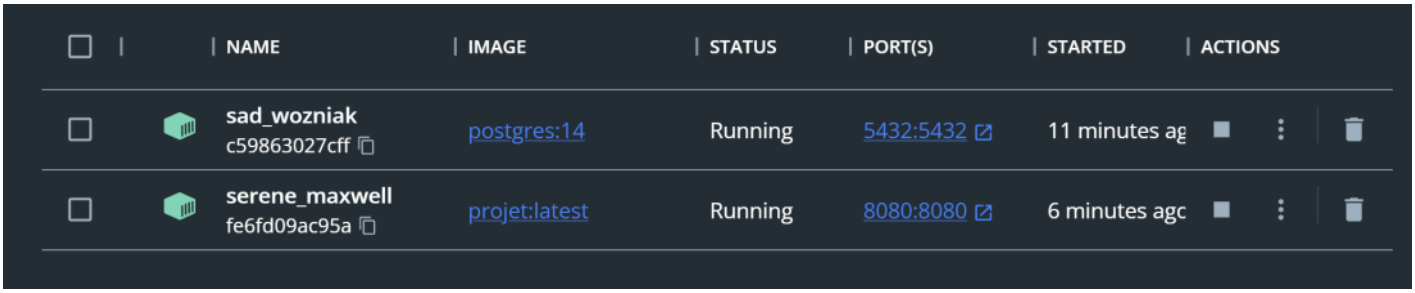
On veut maintenant lier notre base de donnée :
On va donc run un conteneur postgres avec toutes les bonnes variable d'environnement (comme le password par exemple) : docker run -d --env POSTGRES_DB=myDatabase --env POSTGRES_USER=databaseUser --env POSTGRES_PASSWORD=databasePassword postgres:14
(retourne l'id du conteneur grace au **-d**)



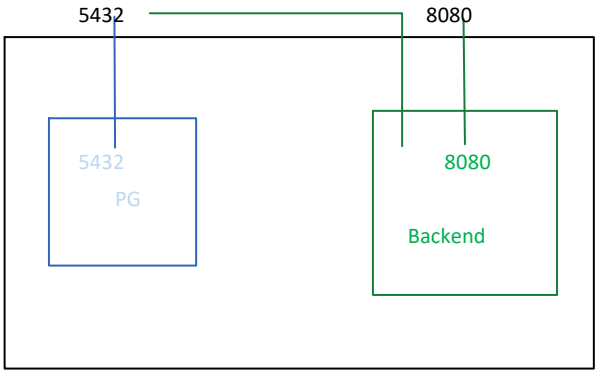
Machine perso

On veut faire le lien entre le backend et le reste :
SPRING_DATASOURCE_URL=jdbc:postgresql://host.docker.internal:5432/myDatabase
SPRING_DATASOURCE_USERNAME=databaseUser
SPRING_DATASOURCE_PASSWORD=databasePassword
c'est exactement les infos dans ressources/application.properties mais point de vue boite backend

Attention : j'ai eu un problème avec le host.docker.internal -> 172.17.0.1 (Gateway)

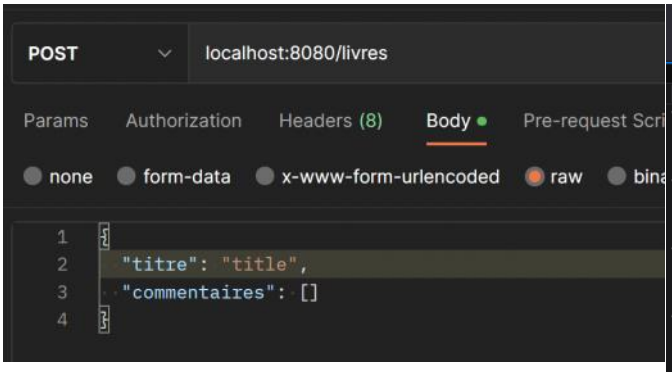


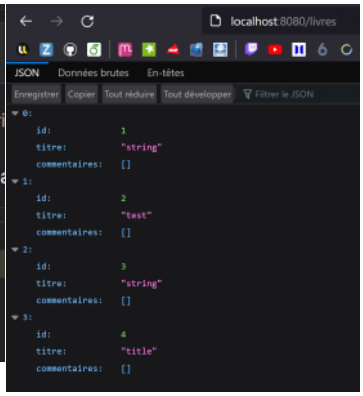
localhost:8080/livres -> retourne tableau de livres (potentiellement vide)



Machine perso

On peut maintenant faire des requêtes :

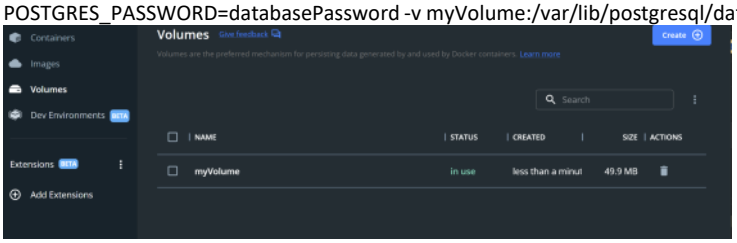




Si le docker de bdd se stop, il faut pouvoir récupérer nos données !

On va donc faire un docker de volume qui stockera les données de notre base :

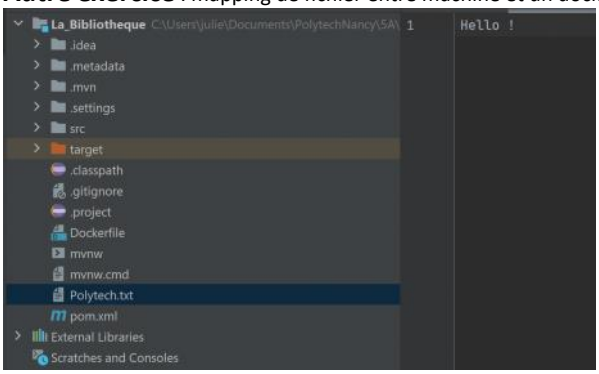
On le crée directement à la créa du docker postgres
docker run -d -p 5432:5432 --env POSTGRES_DB=myDatabase --env POSTGRES_USER=databaseUser --env POSTGRES_PASSWORD=databasePassword -v myVolume:/var/lib/postgresql/data:rw postgres:14



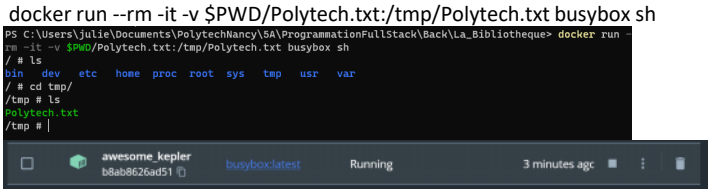
On démarre ensuite notre docker back
docker run -d -p 8080:8080 --env SPRING_DATASOURCE_URL="jdbc:postgresql://172.17.0.1:5432/myDatabase" --env SPRING_DATASOURCE_USERNAME=databaseUser --env SPRING_DATASOURCE_PASSWORD=databasePassword projet

C'est tout bon ! Même en stoppant et supprimant le conteneur postgres, si on le redémarre, les données sont toujours là !

Autre exercice : mapping de fichier entre machine et un docker :



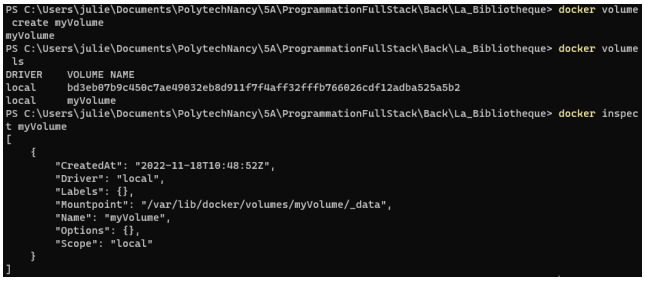
docker run --rm -it -v $PWD/Polytech.txt:/tmp/Polytech.txt busybox sh



En gros, on a un fichier partagé entre notre machine et un docker ! (et ce de façon dynamique !!)

PS : cela fonctionne aussi avec des répertoires

Autre exercice : Docker de volume



(on peut en faire deux et voir qu'ils "communiquent" entre eux !)