Commandes utilisées :

Rf. Document dockerfile pour certaines partie

**docker pull docker/whalesay** : récupération d'image d'un conteneur "whalesay"  
**docker run docker/whalesay cowsay Hello Juliette !** : démarre le conteneur whalesay et appel le process "cowsay" avec comme paramètre "Hello Juliette !" **docker container ls** : liste tous les conteneur qui tournent  
**docker container ls -a** : liste tous les conteneurs  
**--rm** : suprime le conteneur dès qu'il s'eteint (bonne pratique)  
**docker run --rm -it docker/whalesay sh** : Ouvrir un terminal sur le conteneur (et le ferme à la fin)

**docker run busybox sh -c "while true; do $(echo date);sleep 1;done"** : affiche la date toutes les secondes **docker logs determined\_banach** : récupérer les logs du conteneur nommé determined\_banach  
**docker rm -f determined\_banach** : supprimer un conteneur (même si il tourne grâce au -f)  
**docker container prune** : supprimer tous les conteneurs éteints

NB : On utilise ici des conteneurs à partir d'images disponibles en ligne

Créer une image à partir d'un fichier Dockerfile : **docker build --tag first-test .** (dans le bon répertoire)

On peut ensuite run notre image : **docker run first-test** *En l'occurrence, le fichier Dockerfile contenait un echo*

*L'image second-test fait le petit train*

Un peu d'explication sur le contenu du fichier Dockerfile : FROM ... : démarre ton travail depuis ...  
ENV : crée une variable d'environnement  
RUN : création de l'image : c'est sa préparation

CDM : config pour dire à l'image quoi faire au démarrage

LE BUILD :

Build un docker : docker build -t back:2.0 . -t etant le nom du tag souhaité

LES LISTES :

Voir la liste d’image : docker image ls

Voir la liste de conteneurs : docker container ls

LES LOGS :

docker logs f70010214eda ou nom du container

LE RUN :

Run un docker : docker run --rm back:2.0

Si utilisation d’un même paramètre il faut le répéter

* –rm permettant de le supprimer immédiatement
* -e permettant de lire une variable d’environnement style postegres14
* -d lance l’application en arrière-plan et donc renvoie un numéro de processus
* -p pour mapper le port -p 5432:5432 partie de droite notre pc gauche ne change pas
* -P pour mapper un port random

ARRETER UN CONTAINER :

docker container rm -f zealous\_ride nom du container

SUPPRIMER TOUT LES CONTAINER :

docker container prune

TERMINAL DOCKER :

Ouvrir un terminal sur le docker :

docker exec -it zealous\_ride bash nom du container

root@13c26f100c9e:/#

Dans le terminal se connecter a une base de données :

psql -U databaseUser -W myDatabase

quitter la bdd \q

Quitter le terminal exit

INFO SUR LE CONTAINER :

Obtenir des infos sur le container : docker inspect zealous\_ride

nom du container

CONNEXION BDD :

COMMANDE : docker run -d -p 5432:5432 --env POSTGRES\_DB=myDatabase --env POSTGRES\_USER=databaseUser --env POSTGRES\_PASSWORD=databasePassword postgres:14

SPRING\_DATASOURCE\_URL = jdbc:postgresql://host.docker.internal5433/myDatabase

SPRING\_DATASOURCE\_USERNAME=databaseUser

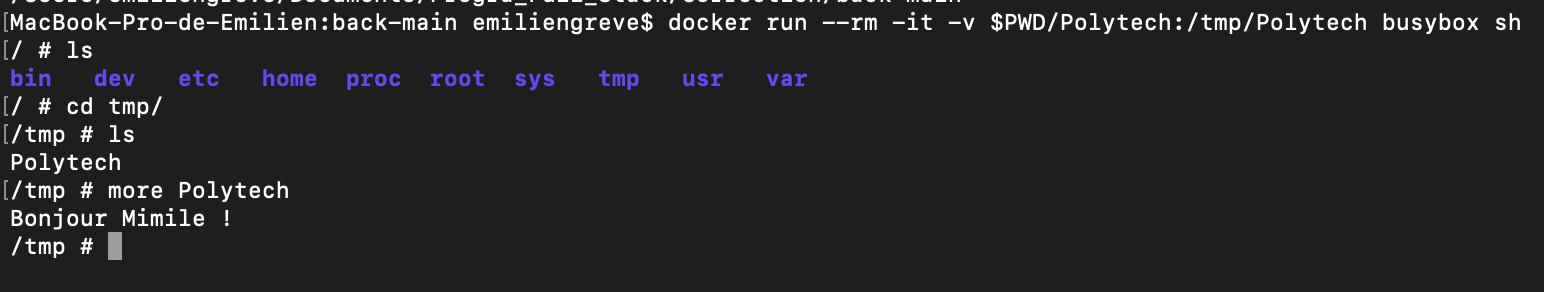
SPRING\_DATASOURCE\_PASSWORD=mypassword

Commande pour run un containers qui se connecte a la bdd précédemment crée :

docker run -d -p 8080:8080 –env SPRING\_DATASOURCE\_URL="jdbc:postgresql://host.docker.internal:5433/myDatabase" --env SPRING\_DATASOURCE\_USERNAME=databaseUser --env SPRING\_DATASOURCE\_PASSWORD=databasePassword back:2.0

PARTAGER UN FICHIER LOCAL DANS LE CONTAINER :

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

Permet d’indiquer le chemin actuel

CREER UN VOLUME DOCKER :

docker volume create myVolume

AFFICHER LES LOGS DU VOLUME :

docker inspect myVolume

AFFICHER LES DIFFERENTS VOLUMES :

docker volume ls

INTERAGIR AVEC LE VOLUME :

docker run --rm -it -v myVolume:/tmp busybox

LES NETWORK :

Créer un network : docker network create myNetwork

Lister les network : docker network ls

Connecter un container au network : docker run --name c1 -it --network myNetwork busybox sh

RUN UN DOCKER AVEC UNE VARIABLE D ENVIRONEMENT :

docker run --name frontend -d -p 4200:4200 --env BACKEND\_URL\_ENV\_VAR="localhost:8080" front:1.0

DOCKER COMPOSE :

Voir le fichier docker-compose.yml

Fonctionne sans tiret

Build : docker-compose build

Demarrer docker compose : docker-compose up -d

Eteindre docker compose : docker-compose down

Modifier dockerfile re build pas de rebuild si modifie juste le docker compose

Préparer une image pour push AWS : docker tag 6f5de4a9512a public.ecr.aws/u1k0q0r7/front:emilien

Push une. Image sur AWS : docker push public.ecr.aws/u1k0q0r7/front:emilien

Télécharger une image sur AWS : docker pull public.ecr.aws/u1k0q0r7/front:guillaumefront

Supprimer une image : docker image rm tag ou id