Exécuter un conteneur Alpine Linux :

* Docker pull alpine 🡪 récupère l’image alpine du registre docker
* Docker run alpine ls -l

Pour voir tous les conteneurs en cours d’exécution : docker ps

run -it : est une commande très utile pour déboguer au niveau le plus bas d'un conteneur.

Pour en savoir plus sur une image Docker : docker inspect alpine

A white text box with black text

Description automatically generated

Exécuter un site web statique dans un conteneur

-d : processus détaché

-P : ports

-e : variables d’environnement

Commande pour télécharger et exécuter l’image directement en une seule fois :

* docker run -d dockersamples/static-site
* docker run --name static-site -e AUTHOR="Your Name" -d -P dockersamples/static-site 🡪 version complète

Pour arrêter, il faut récupérer l’ID avec la commande docker ps puis docker stop [ID] et docker rm [ID] ou remplacer ID par static-site

Pour voir les ports : docker port static-site

Les images Docker

Liste des images dispo : docker images

Question 1.1 :

docker pull postgres

docker pull adminer

docker run -p "8090:8080" --network app-network --name=adminer -d adminer // run l'adminer

docker build . -t mydb // build l'image en la nommant mydb

docker run -d --network app-network --name db mydb // run la database

ajouter ces 2 lignes dans le Dockerfile :

COPY CreateScheme.sql /docker-entrypoint-initdb.d

COPY InsertData.sql /docker-entrypoint-initdb.d

supprimer le conteneur db, re build l'image et re run le conteneur

Question 1.2 :

Nous avons besoin d’une construction en plusieurs étapes pour avoir une image Docker légère en ayant tous les composants nécessaires pour exécuter l’application.

Le Dockerfile est composé de 2 grandes étapes :

* La construction de l’application : en compilant avec Maven et Amazon Correto et en copiant pom.xml et le répertoire src dans le conteneur
* L’exécution de l’application : en copiant le fichier .jar créé à l’étape d’avant et en l’exécutant

Question 1.3 : Document docker-compose les commandes les plus importantes.

Créer un nouveau dossier http avec un Dockerfile composé des 2 lignes suivantes :

FROM httpd:2.4

COPY ./index.html /usr/local/apache2/htdocs/

Build l’image correspondante et run le conteneur :

docker build -t my-apache2 .

docker run -d --name my-front-app --network app-network -p 8080:80 my-apache2

puis utiliser la commande suivante pour le docker-compose :

docker cp my-front-app:/usr/local/apache2/conf/httpd.conf httpd.conf

et ajouter la ligne suivante dans le Dockerfile :

COPY ./httpd.conf /usr/local/apache2/conf/httpd.conf

Re build l’image et run le container

Question 1.4 :

1-4 Documentez votre fichier docker-compose.

version: '3.7'

services:

    backend:

      container\_name: my-running-app

      build: ./simple-api-student-main

      networks:

        - app-network

      depends\_on:

        - database

    database:

      container\_name: db

      build: ./Database

      networks:

        - app-network

    httpd:

      container\_name: my-front-app

      build: ./http

      ports:

        - "8080:80"

      networks:

        - app-network

      depends\_on:

        - backend

networks:

  app-network:

//demander à chat GPT

A screenshot of a computer

Description automatically generated

1-5 Documentez vos commandes de publication et vos images publiées dans Dockerhub.

docker compose up //pour tout run d’un coup

docker tag tp-database clemenced69/tp-database:1.0

puis

docker push clemenced69/tp-database:1.0

à faire pour chaque image et pour publier les images dans le Dockerhub

2-1 Que sont les conteneurs de test ?

2-2 Documentez vos configurations d'actions Github.

name: CI devops 2023  
on:  
 *#to begin you want to launch this job in main and develop* push:  
 branches:  
 - main  
 - develop  
 pull\_request:  
  
jobs:  
 test-backend:  
 runs-on: ubuntu-22.04  
 steps:  
 *#checkout your github code using actions/checkout@v2.5.0* - uses: actions/checkout@v2.5.0  
  
 *#do the same with another action (actions/setup-java@v3) that enable to setup jdk 17* - name: Set up JDK 17  
  
 uses: actions/setup-java@v3  
 with:  
 java-version: 17  
 distribution: 'adopt'  
  
 *#finally build your app with the latest command* - name: Build and test with Maven  
 working-directory: simple-api-student-main  
 run: mvn clean verify