Conteneurisation des Applications

Filière : ASEDS Semestre : 3 Année universitaire : 2023/2024

Prof. Driss ALLAKI

Corrigé du TP Docker

Considérons une application web 3-tiers (composée de frontend, backend et base de données) développée avec le MERN stack (MongoDB, Express, React, Node.js).

Le code source de cette application est disponible sur ces repos : frontend et backend.

Travail à réaliser :

Dans ce TP, il est demandé de réaliser les manipulations suivantes en utilisant un outil de conteneurisation comme Docker.

1. Créer un conteneur (nommé *mongodb-service*) pour la base de données MongoDB en instanciant l'image officielle *mongo*.

   NB : Choisir l'option de stockage la plus adéquate, puis effectuer les actions nécessaires pour rendre le stockage de cette base de données persistant. Justifier votre choix.

```
> docker run -d --name mongodb-service -v mongovol:/data/db -v mongovolconfig:/data/configdb -p 27017:27017 mongo:5.0
```

2. Créer un réseau Docker (tout en lui précisant un driver convenable). Puis connecter le conteneur mongodb avec ce réseau.

```
> docker network create --driver bridge appnetwork

> docker network connect appnetwork mongodb-service
```

3. Créer un fichier Dockerfile pour le projet Backend.

```
FROM node:19.9.0-alpine
ENV NODE_ENV=production
WORKDIR /app
COPY ["package.json", "package-lock.json", "./"]
RUN npm install --${NODE_ENV}
COPY . .
EXPOSE 5000
CMD ["node", "server.js"]
```

4. Enumérer et expliquer "en détails" les bonnes pratiques utilisées pour l'écriture de ce fichier Dockerfile.

```
   1. Utilisation de l'image officielle de node
   2. Spécification d'une version précise de l'image node
   3. Utilisation d'une image minimaliste (alpine) de l'image node
   4. Utilisation du fichier .dockerignore (pour les node_modules)
   5. Installation que des dépendances de production
   6. etc.

Voir plus dans cet article : https://snyk.io/blog/10-best-practices-to-containerize-nodejs-web-applications-with-docker/
```

5. Taper les commandes docker nécessaires pour :
   a. Créer une image docker en local à partir de ce Dockerfile.

```
> docker build –t dall49/backend:1.0.0 .
```

   b. Publier cette image dans Docker Hub.

```
> docker login

> docker push dall49/backend:1.0.0
```

   c. Instancier cette image en créant un conteneur (nommé *backend*) s'exécutant en local et partageant le même réseau avec le conteneur de la base de données mongodb.

```
> docker run –d --network appnetwork --name backend –p 5000:5000 dall49/backend:1.0.0
```

   d. Inspecter ce conteneur backend.

```
> docker inspect backend
```

   e. Afficher les logs liés à ce conteneur backend.

```
> docker logs backend
```

NB : Commenter les attributs utilisés dans chacune de ces commandes.

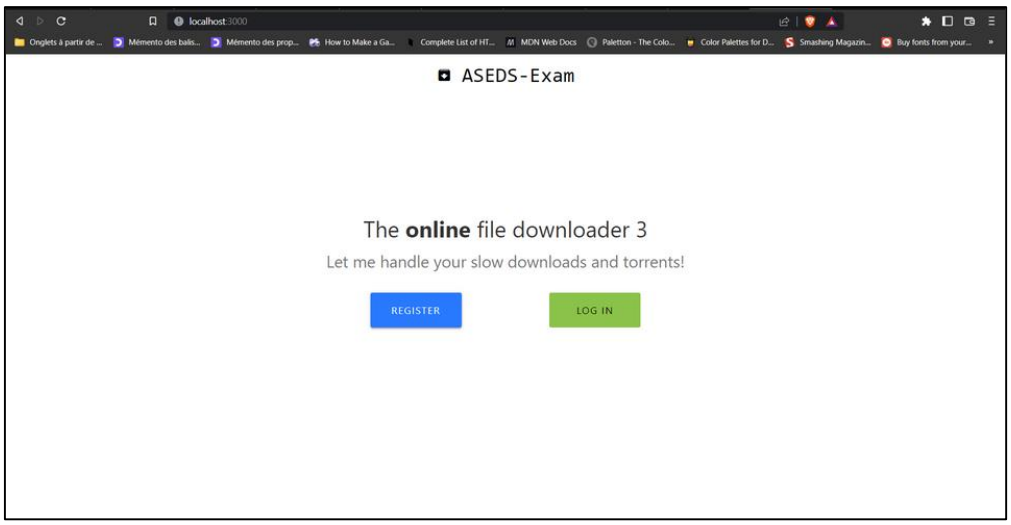
6. Refaire le travail demandé dans les questions 3), 4) et 5) pour le projet Frontend.

```
# phase 1
FROM node:16-alpine as builder
WORKDIR /app
COPY ["package.json", "package-lock.json", "./"]
RUN npm install
COPY . .
RUN npm run build

# phase 2
FROM nginx:1.23.2-alpine as production
ENV NODE_ENV production
COPY --from=builder /app/build /usr/share/nginx/html
COPY --from=builder /app/nginx/nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
```

NB. Au niveau du projet React, on doit remplacer la valeur de « proxy_pass » dans le fichier « nginx.conf » par la valeur « http://backend:5000 » pour rediriger les requêtes vers le serveur du backend.

```
> docker build -t dall49/frontend:1.0.0 .

> docker push dall49/frontend:1.0.0

> docker run -d --network appnetwork --name frontend -p 3000:80
dall49/frontend:1.0.0

> docker inspect frontend

> docker logs frontend
```

7. S'assurer que l'application a été bien conteneurisée et qu'elle fonctionne correctement. (Prendre des prises d'écran du navigateur)



8. Supprimer les 3 conteneurs en exécution.

```
> docker rm -f mongo-service backend frontend
```

9. Redéployer l'application (frontend, backend et base de données) en utilisant docker-compose.

    a. Créer le fichier docker-compose.

```
version: '3.8'

services:
  mongodb-service:
    image: mongo:5.0
    container_name: mongodb-service
    ports:
    - 27017:27017
    networks:
    - appnetwork
    volumes:
    - dbvol:/data/db
    - dbvolconfig:/data/configdb
  backend:
    image: dall49/backend:1.0.0
    container_name: backend
    ports:
    - 5000:5000
    networks:
    - appnetwork
    depends_on:
    - mongodb-service
  frontend:
    image: dall49/frontend:1.0.0
    container_name: frontend
    ports:
    - 3000:80
    networks:
    - appnetwork
    depends_on:
    - backend

volumes:
  dbvol:
    driver: local
  dbvolconfig:
    driver: local

networks:
  appnetwork:
    driver: bridge
```

b. Taper la commande docker permettant de l'exécuter.

```
> docker compose up -d
ou
> docker-compose up
```

c. S'assurer que l'application fonctionne correctement (prises d'écran du navigateur)