Documentation DOCKER

1 – Installation :

git clone https://github.com/Yann-Fournier/Projet-C-sharpe-B2.git

ouvrir un terminal dans la racine du projet et lancer cette commande :

docker compose up -d –build

pour éteindre l'image tapez cette commande :

docker compose down

PS : pensé à installer docker desktop au préalable et à vous y connecter.

2 – Explication :

Le docker fonctionne avec 2 type de fichier, Le Docker compose qui est le fichier « principal » et le dockerfile qui se charge de générer l'image. Dans notre cas il y aura deux dockerfile, un pour le serveur BDD et l'autre pour le serveur WEB.

Le docker compose (docker-compose.yml) :

Version de Docker Compose :

- La version utilisée dans ce fichier est '3'.

Services :

web :

- Image : albericwalsh/ecommerce:WEB-ecommerce
- Construit à partir du répertoire local ./WEB/.
- Nom du conteneur : server_ecomerce_web
- Expose le port 8080 localement, qui est mappé au port 80 dans le conteneur.
- Utilise un réseau défini (network) avec une adresse IP spécifique (10.92.100.10).
- Dépend du service db.

db :

- Image : albericwalsh/ecommerce:BDD-ecomerce
- Construit à partir du répertoire local ./BDD/.

- Nom du conteneur : BDD
- Expose le port 3306 localement, qui est mappé au port 3306 dans le conteneur.
- Utilise un réseau défini (network) avec une adresse IP spécifique (10.92.100.11).
- Utilise le plugin d'authentification caching_sha2_password.
- Redémarre toujours le conteneur.
- Définit les variables d'environnement pour la base de données (nom de la base de données, mot de passe root).

Réseaux :

- Un réseau défini nommé network avec le pilote de réseau bridge.
- Le réseau a une plage d'adresses IP définie comme 10.92.100.0/24.

Le dockerfile (WEB) :

Instruction FROM :

- FROM albericwalsh/ecommerce:WEB-ecomerce : Utilise une image de base provenant du référentiel albericwalsh/ecommerce avec l'étiquette WEB-ecomerce.
- FROM ubuntu : Utilise ensuite l'image officielle Ubuntu comme nouvelle image de base. Notez que seul le dernier FROM est pris en compte dans le Dockerfile.

Installation des locales :

- RUN apt update && apt -y install locales && locale-gen en_US.UTF-8 : Met à jour le système et installe les paquets nécessaires pour les locales, puis génère la locale en_US.UTF-8.

Configuration des variables d'environnement pour les locales :

- ENV LANG en_US.UTF-8 : Définit la variable d'environnement LANG avec la valeur en_US.UTF-8.
- ENV LANGUAGE en_US:en : Définit la variable d'environnement LANGUAGE avec la valeur en_US:en.
- ENV LC_ALL en_US.UTF-8 : Définit la variable d'environnement LC_ALL avec la valeur en_US.UTF-8.

Copie du fichier de configuration Nginx :

- COPY ./default.conf /etc/nginx/conf.d/default.conf : Copie le fichier de configuration Nginx depuis le répertoire local vers le chemin spécifié dans l'image.

Répertoire de travail pour l'entrée du conteneur :

- WORKDIR /docker-entrypoint.d : Définit le répertoire de travail à /docker-entrypoint.d.

Ajout de scripts et exécution :

- ADD dotnet_install.sh . : Ajoute le script dotnet_install.sh au répertoire de travail du conteneur.
- RUN chmod +x dotnet_install.sh : Donne les permissions d'exécution au script.
- RUN ./dotnet_install.sh : Exécute le script d'installation de Dotnet.

Ajout de scripts supplémentaires et configuration :

- ADD ./start.sh . : Ajoute le script start.sh au répertoire de travail du conteneur.
- EXPOSE 80 : Expose le port 80.

Configuration du répertoire de travail final :

- WORKDIR /app : Définit le répertoire de travail à /app.

Copie des fichiers du projet .NET :

- COPY ./Test_Projet.csproj . : Copie le fichier de projet du projet .NET.
- COPY ./Program.cs . : Copie le fichier de programme du projet .NET.
- COPY ./SQLRequest.cs . : Copie le fichier de requêtes SQL du projet.

Commandes CMD :

- CMD ["nginx", "-g", "daemon off;"] : Définit la commande par défaut pour le conteneur, lançant Nginx en mode démon.
- CMD ["dotnet", "run"] : Définit une autre commande par défaut pour le conteneur, exécutant la commande dotnet run.

Dockerfile (BDD) :

Instruction FROM :

- FROM albericwalsh/ecommerce:BDD-ecomerce : Utilise une image de base provenant du référentiel albericwalsh/ecommerce avec l'étiquette BDD-ecomerce.

Répertoire de travail :

- WORKDIR /app : Définit le répertoire de travail à /app.

Commande RUN :

- RUN echo "BDD Dockerfile processing..." : Exécute une commande pour afficher un message indiquant que le traitement du Dockerfile de la base de données est en cours.

Copie du contenu local dans le conteneur :

- COPY . /app : Copie l'ensemble du contenu du répertoire local (où se trouve le Dockerfile) dans le répertoire de travail du conteneur (/app).

Ajout d'un script SQL dans le répertoire d'initialisation de la base de données :

- ADD script.sql /docker-entrypoint-initdb.d : Ajoute le fichier script.sql au répertoire d'initialisation de la base de données (/docker-entrypoint-initdb.d), ce qui suggère qu'il sera exécuté lors de l'initialisation de la base de données.

Exposition du port 3306 :

- EXPOSE 3306 : Expose le port 3306, le port par défaut utilisé par MySQL/MariaDB, pour permettre la connexion à la base de données depuis l'extérieur du conteneur.