**Rapport TP Docker**

* Why should we run the container with a flag -e to give the environment variables?

L’indicateur -e dans Docker permet de définir des variables d’environnement. Cela permet de modifier le comportement de l’application sans toucher au code source. Dans notre cas, cela permet de protéger des informations sensibles ( Mot de passe de la base de donnée ) en évitant leur inclusion dans l’image, et d’adapter facilement les paramètres selon différents environnements .

* Why do we need a volume to be attached to our postgres container?

Attacher un volume à un conteneur PostgreSQL permet de garantir la persistance des données. En effet, les volumes conservent les données même si le conteneur est arrêté ou supprimé, évitant ainsi toute perte de données. Les volumes facilitent la sauvegarde et la restauration des données. De plus les volumes rendent le processus de migration des données d'un conteneur à un autre beaucoup plus facile, ce qui facilite les mises à jour ou la migration vers un nouvel environnement.

* 1. Document your database container essentials: commands and Dockerfile :

1. Ecriture du Dockerfile
2. “*Docker build -t username/nom\_image .*” permet de construire une image avec comme tag ‘username/nom\_image‘
3. ‘*Docker network create app-network’* permet de créer un réseau personnalisé appelé ‘network-app’. Permet de mieux gérer les communications entre les conteneurs
4. ‘Docker volume my-own-datadir’ créer son volume.
5. ‘*Docker run -p 5432:5432 --name dbtp --network network-app -e POSTGRES\_PASSWORD=pwd -d -v /my/own/datadir:/var/lib/postgresql/data username/nom\_image*’ permet de créer le conteneur à partir de l’image ‘username/nom\_image’. ‘-p’ permet de définir le port du conteneur ( port externe / port interne ). ‘--name’ permet de nommer son conteneur. ‘--network ‘ permet de préciser le réseau auquel se connecter. ‘-e’ permet de préciser une variable sans quelle soit coder en dur dans le Dockerfile. ‘-d’ permet d’executer le coneneur ‘on detach’ donc le conteneur d’ouvre en arrière-plan. ‘-v’ permet d’attacher un volume a notre conteneur pour enregistré les données.
   1. Why do we need a multistage build? And explain each step of this dockerfile.

Le multistage build dans ce Dockerfile permet de séparer le processus de construction de l'application Java de son exécution, en utilisant une image contenant Maven pour compiler le code et une image plus légère pour l'exécution. Cela réduit la taille de l'image finale et améliore la sécurité en n'incluant que les fichiers nécessaires à l'exécution de l'application, sans les outils de développement.

FROM maven:3.9.9-amazoncorretto-21 AS myapp-build : utilise l’image Maven comme base pour la première étape, nommée myapp-build. Cela signifie que nous allons utiliser Maven pour construire l'application.

ENV MYAPP\_HOME /opt/myapp : définit une variable d'environnement MYAPP\_HOME, qui spécifie le répertoire où l'application sera installée.

WORKDIR $MYAPP\_HOME : définit le répertoire de travail pour toutes les instructions suivantes dans le Dockerfile

COPY pom.xml . : copie le fichier pom.xml (fichier de configuration de Maven) de l'hôte vers le répertoire de travail dans le conteneur

COPY src ./src : copie le répertoire src de l'hôte vers le répertoire de travail dans le conteneur

RUN mvn package -DskipTests : exécute Maven pour construire l'application. L'option -DskipTests indique à Maven de ne pas exécuter les tests lors de la construction

FROM amazoncorretto:21 : utilise une image Docker basée sur Amazon Corretto 21

ENV MYAPP\_HOME /opt/myapp : définit à nouveau la variable d'environnement MYAPP\_HOME

WORKDIR $MYAPP\_HOME : définit le répertoire de travail pour la deuxième étape, comme précédemment.

COPY --from=myapp-build $MYAPP\_HOME/target/\*.jar $MYAPP\_HOME/myapp.jar : copie le fichier .jar généré par Maven depuis la première étape (myapp-build) vers le répertoire de travail de la deuxième étape. --from=myapp-build indique que nous récupérons le fichier de l'étape précédente.

ENTRYPOINT java -jar myapp.jar : définit la commande qui sera exécutée lorsque le conteneur sera démarré.

* Why do we need a reverse proxy?

Le reverse proxy sert à gérer la communication entre le front, le back et la base de données. Il permet de répartir la charge entre plusieurs serveurs d'application. Cela augmente la sécurité et la scalabilité. De plus, il facilite la gestion des certificats SSL et des règles de routage, simplifiant ainsi l'architecture du système.

* Why is docker-compose so important?

Docker Compose est essentiel car il permet de gérer facilement des applications multi-conteneurs à l'aide d'un simple fichier de configuration yaml, cela simplifie déploiement et la communication entre les services. Cela assure aussi un environnement de développement cohérents et facilite l'orchestration des conteneurs avec des commandes simples, améliorant ainsi l'efficacité du flux de travail.

* 1. Document docker-compose most important commands.
* *docker-compose up* : ça démarre tous les services définis dans le fichier docker-compose.yml. On peut rajouter *–build* pour forcer la reconstruction des images.
* *docker-compose down* : ça arrête et supprime tous les conteneurs, réseaux et volumes créés par docker-compose up.
* *docker-compose logs* : Cette commande permet de visualiser les journaux des services en cours d'exécution.
  1. Document your docker-compose file.

Une image contenant texte, capture d’écran

Description générée automatiquement

build : Spécifie le chemin vers le Dockerfile pour construire l'image.

container\_name : Définit le nom du conteneur pour le service backend.

networks : Connecte le service au réseau network-app.

ports : Redirige le port 8080 de l'hôte vers le conteneur.

depends\_on : Indique que le backend dépend du service database.

environment : Définit les variables d'environnement pour la connexion à la base de données.

volumes : Montre un volume persistant pour stocker les données de la base de données.

network-app : Définition d'un réseau pour permettre la communication entre les services.

volume-tp : Définition d'un volume pour la persistance des données de la base de données.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1-5 Document your publication commands and published images in dockerhub.

*docker login* : commande pour vous connecter à votre compte Docker Hub.

*docker tag my-database USERNAME/my-database:1.0* : Cette commande tague l'image locale my-database avec le nom USERNAME/my-database et la version 1.0.

*docker push USERNAME/my-database:1.0* : télécharge l'image taguée vers votre dépôt Docker Hub.

* Why do we put our images into an online repo?

Nous mettons nos images dans un dépôt en ligne pour faciliter le partage et la distribution de nos applications, permettant à d'autres utilisateurs de les télécharger et de les exécuter facilement. De plus, cela assure une sauvegarde de nos images et permet de gérer différentes versions de manière centralisée.