# TP 1 : Discover Docker

1.1 Database

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

**Manque les commandes**

1.2

Avec les multi-stage builds, on utilise plusieurs instructions FROM dans le Dockerfile. Chaque instruction FROM peut utiliser une image de base différente et commence une nouvelle étape de la construction. On peut copier des artefacts d'une étape à une autre, en laissant derrière tout ce dont on n’a pas besoin dans l'image finale.

Ça a plusieurs intérêts :

* Réduire l’image de la taille finale qui ne contiendra que les composants nécessaires
* Avoir une sécurité accrue du fait qu’on peut ne pas inclure les dépendances sensibles dans l’image finale de production.
* Chaque stage a un environnement clean et isolé ce qui peut éviter certains conflits ou problèmes pendant certains processus

Dans la première partie, on définit une image de base maven:3.8.6-amazoncorretto-17 pour la construction de l’application. Les étapes de cette partie sont les suivantes :

**FROM maven:3.8.6-amazoncorretto-17 AS myapp-build** : Cela configure la première étape et lui donne le nom "myapp-build".

**ENV MYAPP\_HOME /opt/myapp** : On définit une variable d'environnement pour le répertoire de base de l’application.

**WORKDIR $MYAPP\_HOME** : On définit le répertoire de travail au répertoire de base de l'application.

**COPY pom.xml . et COPY src ./src** : On copie le fichier pom.xml et le code source de l’application depuis le répertoire local vers le répertoire de travail dans le conteneur.

**RUN mvn package -DskipTests** : On exécute la commande mvn package pour construire l’application. L'option -DskipTests indique de ne pas exécuter les tests pendant le processus de construction.

Dans la deuxième partie, on utilise une image de base plus légère, amazoncorretto:17, pour exécuter l’application. Les étapes de cette partie sont les suivantes :

**FROM amazoncorretto:17** : On configure la deuxième étape en utilisant l'image de base Amazon Corretto 17.

**ENV MYAPP\_HOME /opt/myapp** : On définit à nouveau la variable d'environnement pour le répertoire de base de l’application, pour garantir la cohérence avec la première étape.

**WORKDIR $MYAPP\_HOME** : On définit le répertoire de travail dans cette étape également.

**COPY --from=myapp-build $MYAPP\_HOME/target/\*.jar $MYAPP\_HOME/myapp.jar** : On copie le fichier JAR de l'application construite dans la première étape vers cette étape, afin de l'inclure dans l'image finale.

**ENTRYPOINT java -jar myapp.jar** : On spécifie la commande à exécuter lorsque le conteneur démarre. Dans ce cas, il s'agit de l'exécution de l'application Java à l'aide du fichier JAR copié

1.3

**docker-compose up :** permet de démarrer l’application docker compose, elle lit le fichier docker-compose.yml, crée et démarre tous les services définis en tant que conteneurs

**docker-compose build :** reconstruit les images de tous les services définis dans le fichier docker-compose.yml

**docker-compose ps :** affiche l’état des conteneurs gérés par Docker Compose, indiquant s’ils sont en cours d’exécution ou arrêtés

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1.5

docker tag database thibaultclt/database:1.0

docker push thibaultclt/database:1.0

docker tag http thibaultclt/http:1.0

docker push thibaultclt/http:1.0

docker tag simpleapistudent thibaultclt/simpleapistudent:1.0

docker push thibaultclt/simpleapistudent:1.0

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

# TP 2: Discover Github Action

2.1

Les testcontainers sont des librairies java qui permettent de run des container docker tout en faisant les tests d’intégration et les tests unitaires.

2.2

Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement

2.3

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

# TP 3: Ansible

3.1

**Manque les base commands**

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

3.2

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

3.3

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement