Cours 06 – Exercice

# Plan de leçon

## Retour sur le dernier cours

* Dockerfile
  + #, FROM, ENV, WORKDIR, COPY, RUN, EXPOSE, CMD
  + Docker Compose (build:, context: dockerfile:)
  + Commandes (docker compose build, docker compose build --no-cache)

## Alpine

* Installation d’applications
  + apk add --no-cache <application>
* Connexion au conteneur
  + Parfois bash n’est pas installé
    - docker exec -it <container-id> sh
  + Parfois l’utilisateur n’est pas root par défaut et il n’est pas possible de faire des modifications ou accèder aux fichiers nécessaires
    - docker exec -it <container-id> -u root sh

## Dockerfile

* Empêcher la copie de fichiers non nécessaires lors de COPY dans les Dockerfile
  + Fichiers Docker
    - compose.yaml
    - Dockerfile
  + Etc.
* Créer un fichier .dockerfile à la racine au même endroit que le compose.yaml
* Syntaxe similaire à .gitignore

# Exercice

À partir du code fourni, déployer une application multi-conteneur à l’aide de Docker Compose

* Copier le dossier à partir de Teams
* Créer le fichier compose.yaml à la racine du code fourni

## frontend

Serveur web Apache qui fournis une page web simple avec du code Javascript côté client qui appelle les 2 services web côté serveur pour les users et les transactions.

1. Dans le dossier frontend, créer un Dockerfile.frontend
   1. Démarrer du tag spécifique le plus récent d’alpine, ne pas utiliser latest  
      <https://hub.docker.com/_/alpine/>
   2. Installer les applications : apache2, netcat
   3. Copier les fichiers du dossier frontend à l’emplacement /var/www/localhost/htdocs/
   4. Ouvrir le port 80
   5. Au lancement du contenant, démarrer Apache avec la commande httpd -D FOREGROUND
2. Dans le Docker Compose, ajouter le service pour le frontend
   1. Build : dans le dossier frontend, le Dockerfile.frontend
   2. Port : 8080 (localement) 80 (interne)
   3. Networks :
      1. Doit uniquement communiquer avec les services de transactions et users
      2. Ne doit pas communiquer directement avec les bases de données
3. Vérification
   1. Vérifier que la page web et les 3 images sont disponibles sur le port local à <http://127.0.0.1:8080>

## Users

Serveur Node qui appelle la base de données de users MySQL et retourne les réponses au frontend.

1. Dans le dossier users, créer un Dockerfile.users
   1. Démarrer du tag spécifique le plus récent de node, ne pas utiliser latest

<https://hub.docker.com/_/node/>

* 1. Copier tous les fichiers du dossier users dans /app
  2. Changer le dossier pour /app
  3. Exécuter la commande npm install pour installer les dépendances du package.json
  4. Pas besoin d’ouvrir le port 3000, car il est déjà ouvert dans le Dockerfile de base de node
  5. Au lancement du contenant, démarrer Node avec la commande npm start

1. Dans le Docker Compose, ajouter le service pour les transactions
   1. Build : dans le dossier users, le Dockerfile.users
   2. Port : 33000 (localement) 3000 (interne)
   3. Networks :
      1. Doit communiquer le frontend et la base de données de users
      2. Ne doit pas communiquer directement avec les autres éléments
2. Vérification
   1. Vérifier à partir du conteneur frontend que le port 3000 du service users est accessible  
      nc -zv users 3000

## Users-db

Base de données MySQL qui contient les users.

1. Dans le Docker Compose, ajouter le service pour les transactions-db
   1. Démarrer du tag spécifique le plus récent de mysql, ne pas utiliser latest
   2. Port : 33306 (localement) 3306 (interne)
   3. Volume bind mount : fournir le fichier init-users.sql à l’emplacement /docker-entrypoint-initdb.d/init.sql
   4. Environnement : Configurer un mot de passe pour root et trouver dans la page Docker Hub comment configurer un user et un password <https://hub.docker.com/_/mysql>
   5. Networks :
      1. Doit uniquement communiquer avec les services de transactions et users
      2. Ne doit pas communiquer directement avec les bases de données
2. Dans le app.js du dossier users, mettre à jour la connexion de la base de données
3. Vérification
   1. Vérifier dans WSL en se connectant avec mysql que la base de données contient bien les valeurs  
      mysql -h <host> -P <port> -u <user> -p  
      use user\_management;  
      select \* from users;
   2. Vérifier dans le conteneur users que le service users-db est accessible au port 3306
   3. Vérifier que la route de users retourne des valeurs <http://127.0.0.1:33000/users>
   4. Vérifier que la liste dans la page web du frontend affiche les valeurs des users
4. Volumes
   1. Une fois le tout fonctionnel, ajouter un volume persistent pour /var/lib/mysql
   2. Relancer les conteneurs et s’assurer que le volume est utilisé, sinon supprimer tous les conteneurs

## Transactions

Serveur Node qui appelle la base de données de transactions MongoDB et retourne les réponses au frontend.

1. Dans le dossier transactions, créer un Dockerfile.transactions
   1. Démarrer du tag spécifique le plus récent de node, ne pas utiliser latest
   2. Copier tous les fichiers du dossier transactions dans /app
   3. Changer le dossier pour /app
   4. Exécuter la commande npm install pour installer les dépendances du package.json
   5. Ouvrir le port 3301
   6. Au lancement du contenant, démarrer Node avec la commande npm start
2. Dans le Docker Compose, ajouter le service pour les transactions
   1. Build : dans le dossier transactions, le Dockerfile.transactions
   2. Port : 33001 (localement) 3001 (interne)
   3. Networks :
      1. Doit communiquer le frontend et la base de données de transactions
      2. Ne doit pas communiquer directement avec les autres éléments
3. Vérification
   1. Vérifier à partir du conteneur frontend que le service transactions est accessible au port 3001

## Transactions-db

Base de données MongoDB qui contient les transactions.

1. Dans le Docker Compose, ajouter le service pour les transactions-db
   1. Démarrer du tag spécifique le plus récent de mongo, ne pas utiliser latest  
      <https://hub.docker.com/_/mongo>
   2. Port : 37017 (localement) 27017 (interne)
   3. Volume bind mount : fournir le fichier init-transactions.js à l’emplacement /docker-entrypoint-initdb.d/init.js
   4. User : Trouver dans la page Docker Hub comment configurer un user et un password  
      <https://hub.docker.com/_/mongo>
   5. Networks :
      1. Doit uniquement communiquer avec les services de transactions et users
      2. Ne doit pas communiquer directement avec les bases de données
2. Dans le app.js du dossier transactions, mettre à jour la connexion de la base de données
3. Vérification
   1. Vérifier dans WSL en se connectant avec mongosh que la base de données contient bien les valeurs  
      mongosh --host <host> --port <port> -u <user> -p <password>  
      use transactions  
      db.transactions.find().pretty()
   2. Vérifier dans le conteneur transactions que le service transactions-db est accessible au port 27017
   3. Vérifier que la route de transactions retourne des valeurs <http://127.0.0.1:33001/transactions>
   4. Vérifier que la liste dans la page web du frontend affiche les valeurs des transactions
4. Volumes
   1. Une fois le tout fonctionnel, ajouter un volume persistent pour /data/db
   2. Relancer les conteneurs et s’assurer que le volume est utilisé, sinon supprimer tous les conteneurs

## GitHub

* Faire un commit et push de l’exercice dans GitHub

# Devoir

* Terminer les exercices