TP 3 : Mise en application des volumes, variables d'environnement et réseaux

1. Construction des images Docker

Au cours de cette formation, nous allons travailler avec le code source d'une application web d'entraînement qui a déjà été développé. Cette application est composée d'un backend et d'un frontend : gitlab.com/groupomania/express-ts et gitlab.com/groupomania/vue3-ts.

1.1. Récupération du code

Nous avons déjà utilisé le backend au TP précédent. Pour mettre à jour le code et obtenir la dernière version du Dockerfile, exécutez les commandes suivantes :

```
git stash
git pull
```

Pour le frontend, il faut le récupérer via les commandes suivantes :

```
git clone https://gitlab.com/groupomania/vue3-ts
cd vue3-ts
```

1.2. Construction des images

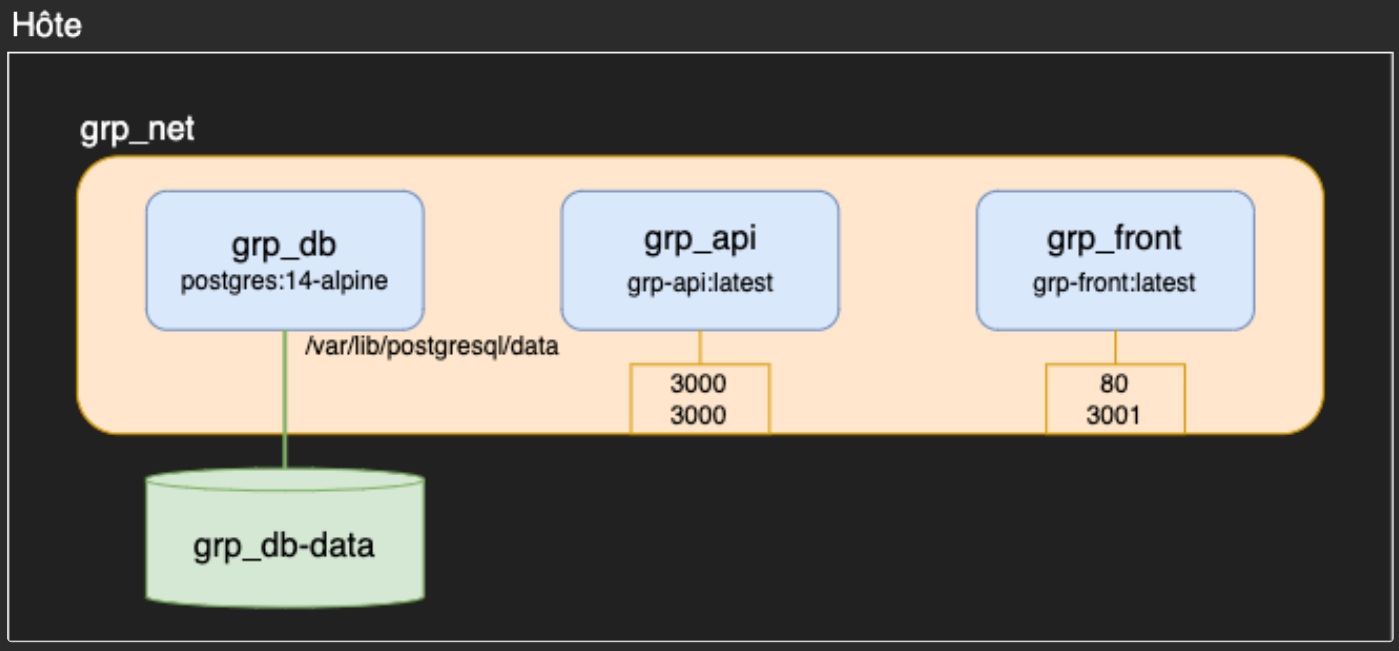
Construire les images ainsi :

```
# Backend (express-ts)
docker build -t grp-api .

# Frontend (vue3-ts)
docker build -t grp-front .
```

2. Conception du déploiement

Nous allons construire pas à pas l'architecture suivante :



Cette architecture représente l'ensemble du projet `groupomania`. Elle est composée d'un réseau, d'un volume et de trois conteneurs :

- Le réseau `grp_net` de type `bridge` qui permettra à tous les éléments du projet d'interagir ensemble
- Le volume nommé `grp_db-data`, qui sera monté à l'emplacement `/var/lib/postgresql/data` du système de fichiers du conteneur de la base de données
- Le conteneur de base de données `grp_db`, construit à partir de l'image `postgres:14-alpine`, avec les variables d'environnement suivantes :
  - `NODE_ENV` = `development` ⟹ Environnement
  - `POSTGRES_USER` = `admin` ⟹ Identifiant de la base
  - `POSTGRES_PASSWORD` = `password` ⟹ Mot de passe de la base
  - `POSTGRES_DB` = `grp` ⟹ Nom de la base
- Le conteneur du backend `grp_api` construit avec l'image `grp-api:latest` construite précédemment, dont le port `3000` est publié sur l'hôte, avec les variables d'environnement suivantes :
  - `DATABASE_URL` = `postgresql://admin:password@grp_db:5432/grp?schema=public` ⟹ Informations de connexion à la base de données
  - `SECRET_KEY` = `s3cr3t` ⟹ Clé secrète pour la sécurisation de l'API
- Le conteneur du frontend `grp_front` construit avec l'image `grp-front:latest` construite précédemment, dont le port `3001` est publié sur l'hôte.

> Prenez un instant pour réfléchir à la mise en œuvre par vous-même avant de passer à la partie suivante. Si vous vous sentez à l'aise, vous pouvez essayer de faire sans la partie 3.

3. Mise en œuvre

On commence par créer le volume et le réseau :

```
docker volume create grp_db-data
docker network create grp_net
```

On passe ensuite à la création de la base de données :

```
docker run -d -v grp_db-data:/var/lib/postgresql/data \
 --network grp_net \
 --name grp_db \
 -e POSTGRES_USER="admin" \
 -e POSTGRES_PASSWORD="password" \
 -e POSTGRES_DB="grp" \
 postgres:14-alpine
```

Maintenant, on crée le conteneur du backend :

```
docker run -d --network grp_net \
  --name grp_api \
  -p 3000:3000 \
  -e NODE_ENV="development" \
  -e DATABASE_URL="postgresql://admin:password@grp_db:5432/grp?schema=public" \
  -e SECRET_KEY="s3cr3t" \
  grp-api
```

Puis le conteneur du frontend :

```
docker run -d --network grp_net \
 --name grp_front \
 -p 3001:80 \
 grp-front
```

L'interface utilisateur est maintenant disponible à l'URL localhost:3001.

Créer un compte et se connecter pour valider le bon fonctionnement de l'infrastructure.

BONUS

Visualisation des données dans la base

On peut ajouter un conteneur de visualisation pour explorer la base de données :

```
docker run -d --network grp_net \
 --name grp_db-visu \
 -p 5555:5555 \
 -e NODE_ENV="development" \
 -e DATABASE_URL="postgresql://admin:password@grp_db:5432/grp?schema=public" \
 -e SECRET_KEY="s3cr3t" \
 grp-api \
 npm run db:studio
```

Maintenant, en accédant à `localhost:5555`, on pourra suivre l'évolution des données dans la base.

Réflexion

> Cette question nécessite une bonne compréhension du fonctionnement d'une application web frontend/backend en JavaScript

Que se passerait-t-il si le conteneur `grp_front` n'était pas dans le réseau `grp_net` ? Est-ce souhaitable ?