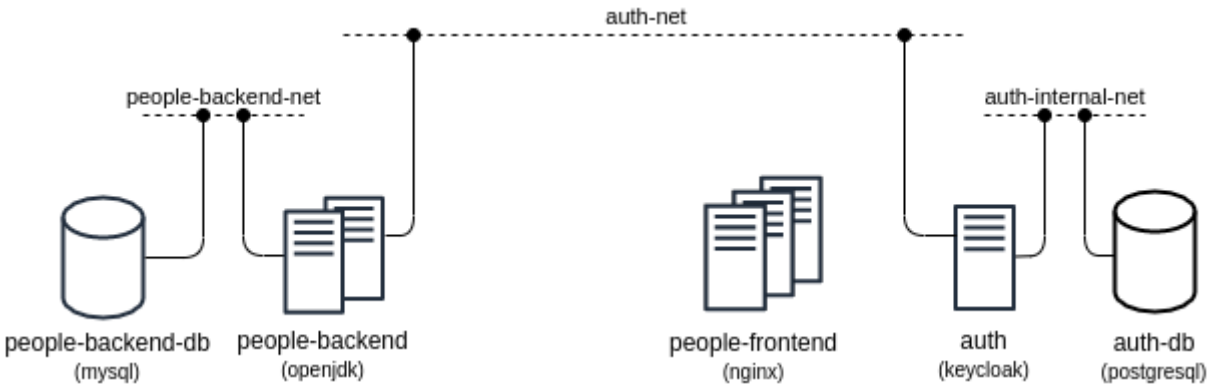
Application `People`

Nous allons déployer une application composée :

- d'un **backend** :
  - `people-backend` : une application java avec le framework Spring boot
  - `people-backend-db` : une base de données mysql
- d'un **frontend** :
  - `people-frontend` : une application angular 11, déployée avec nginx
- d'une **authentification** externe :
  - `auth` : un serveur keycloak
  - `auth-db` : une base de données postgresql



Le code sources de l'application est disponible sur le dépôt git .

Les étapes que nous suivrons sont les suivantes :

1. Construire les images pour chacun des services (quand c'est nécessaire)
2. Écrire le `docker-compsoe.yml` décrivant le péploiement de ces services
3. Déployer notre infrastructure sur play with docker

On supposera, dans le reste de ce tuto, que votre terminal est situé dans le répertoire principal du projet.

Image `people-backend`

1. Créez, dans le répertoire `people-backend` du projet, un fichier `Dockerfile` avec le contenu suivant :

```
FROM openjdk:11-jdk
RUN addgroup --system spring && adduser --system --group spring
USER spring:spring
COPY target/*.jar app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

2. Construisez l'image :

```
sudo docker build -t <your-dockerid>/people-backend people-backend
```

3. Publiez la sur le docker hub :

```
sudo docker login
sudo docker push <your-dockerid>/people-backend
sudo docker logout
```

Image `people-frontend`

1. Créez, dans le répertoire `people-frontend` du projet, un fichier `Dockerfile` avec le contenu suivant :

```
FROM nginx
COPY dist/people-frontend/* /usr/share/nginx/html/
```

2. Construisez l'image :

```
sudo docker build -t <your-dockerid>/people-frontend people-frontend
```

3. Publiez la sur le docker hub :

```
sudo docker login
sudo docker push <your-dockerid>/people-frontend
sudo docker logout
```

Image 'auth'

1. Créez, dans le répertoire `auth` du projet, un fichier `Dockerfile` avec le contenu suivant :

```
FROM jboss/keycloak
COPY realm-export.json /tmp/
ENV KEYCLOAK_IMPORT /tmp/realm-export.json
```

2. Construisez l'image :

```
sudo docker build -t <your-dockerid>/auth auth
```

3. Publiez la sur le docker hub :

```
sudo docker login
sudo docker push <your-dockerid>:auth
sudo docker logout
```

Création des services

1. Créez, à la racine de votre répertoire `people`, un fichier `docker-compose.yml`
2. Définissez les éléments suivants :
    - un service `people-backend` :
        - basé sur l'image `<your-dockerid>/people-backend`
        - avec les variables d'environnement suivantes :
            - `spring.datasource.username: root`
            - `spring.datasource.password: root`
            - `spring.datasource.url: jdbc:mysql://people-backend-db:3306/people?serverTimezone=Europe/Paris`
            - `keycloak.auth-server-url: http://localhost:8080/auth`
        - relié aux réseaux `people-backend-net` et `auth-net`
        - dépendant des services `auth` et `people-backend-db`
        - exposant le port interne 8080 sur le 8081 de l'hôte
    - un service `people-backend-db`
        - basé sur l'image `mysql`
        - avec les variables d'environnement suivantes :
            - `MYSQL_ROOT_PASSWORD: root`
            - `MYSQL_DATABASE: people`
        - relié au réseau `people-backend-net`
        - avec un volume `people-backend-db-vol` monté sur le chemin `/var/lib/mysql` du container
    - un service `people-frontend`
        - basé sur l'image `<your-dockerid>/people-frontend`
        - exposant le port interne 8080 sur le 8080 de l'hôte
        - dépendant du service `people-backend`
    - un service `auth` :
        - basé sur l'image `<your-dockerid>/auth`
        - avec les variables d'environnement suivantes :
            - `KEYCLOAK_USER: admin`
            - `KEYCLOAK_PASSWORD: admin`
            - `DB_VENDOR: postgres`
            - `DB_ADDR: auth-db`

        - `DB_PORT: 5432`
        - `DB_USER: auth`
        - `DB_PASSWORD: auth`
      - relié aux réseaux `auth-internal-net` et `auth-net`
      - exposant le port interne 8080 sur le 8180 de l'hôte
      - dépendant du service `auth-db`

  - un service `auth-db` :

      - basé sur l'image `postgres`
      - avec les variables d'environnement suivantes :
        - `POSTGRES_USER: auth`
        - `POSTGRES_PASSWORD: auth`
        - `POSTGRES_DB: keycloak`
      - relié au réseau `auth-internal-net`
      - avec un volume `auth-db-vol` monté sur le chemin `/var/lib/postgresql/data` du container

  - les réseaux suivants : `people-backend-net`, `auth-net` et `auth-internal-net`

  - les volumes suivants : `auth-db-vol` et `people-backend-db-vol`