Rapport Evaluation finale

Nom : MOHELLEBI
Prénom : Koceila

Partie 1 : Gestion de Code avec GitHub

1/ Pour créer un compte, il suffit d'aller sur github.com
2/ Pour cloner le dépôt, il faut utiliser la commande suivante dans un terminal:

git clone git@github.com:mohellebikoceila/DEVOPS.git

3/ pour créer une branche on utilise la commande suivante:

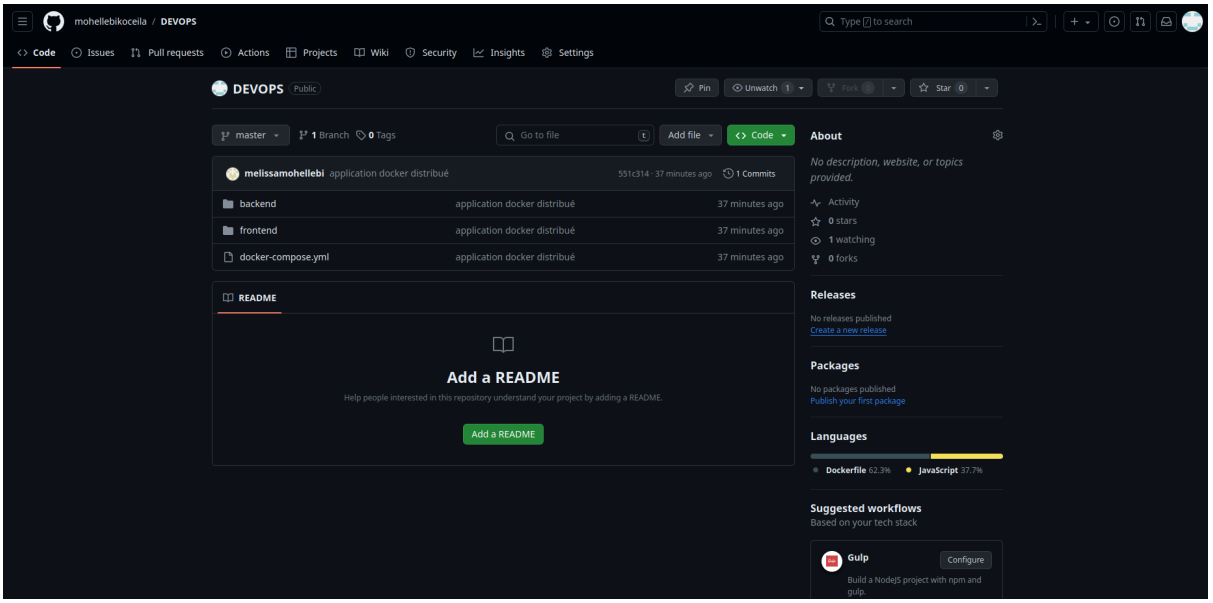
git switch -c appli_docker

4/ pour pousser le code dans la branche distante il faut utiliser les commandes suivantes dans le répertoir git racine:

git add *
git commit -m "application distribué avec docker"
git push

```
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$ git push
Username for 'https://github.com': mohellebikoceila
Password for 'https://mohellebikoceila@github.com':
Énumération des objets: 9, fait.
Décompte des objets: 100% (9/9), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (9/9), fait.
Écriture des objets: 100% (9/9), 1.19 Kio | 1.19 Mio/s, fait.
Total 9 (delta 0), réutilisés 0 (delta 0)
To https://github.com/mohellebikoceila/DEVOPS.git
 * [new branch]      master -> master
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$ git config --global credential.helper store
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$
(failed reverse-i-search)`': sudo apt install^Citk
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$
```

Voici un aperçue sur GitHub



voici le lien git du dépot : https://github.com/mohellebikoceila/DEVOPS

Partie 2 : Conteneurisation avec docker

4/ supposant une application web composée d'un frontend en angular et d'un backend node.js avec une base de donnée MongoDB
Le Dockerfile pour le frontend est le suivant :

```
FROM nginx:latest

COPY nginx.conf /etc/nginx/conf.d/default.conf
COPY dist /usr/share/nginx/html
```

le répertoir dist contiendra à la suite de l'exécution du conteneur, le code compilé du frontend
continue du fichier nginx.conf

```
server {
    listen 80;
    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

Le Dockerfile pour le backend est le suivant:

```
FROM node:latest

COPY package*.json ./
RUN npm install -g @angular/cli
COPY ./app.js ./app.js
RUN npm install express
CMD ["node", "app.js"]
```

L'application exécutée est mise dans un fichier app.js contenant le code suivant:

```
const express = require('express');
const app = express();
const PORT = 3000;

app.get('/api/data', (req, res) => {
 res.json({ message: 'Hello World' });
});

app.listen(PORT, () => {
 console.log(`Server listening on port ${PORT}`);
});
```

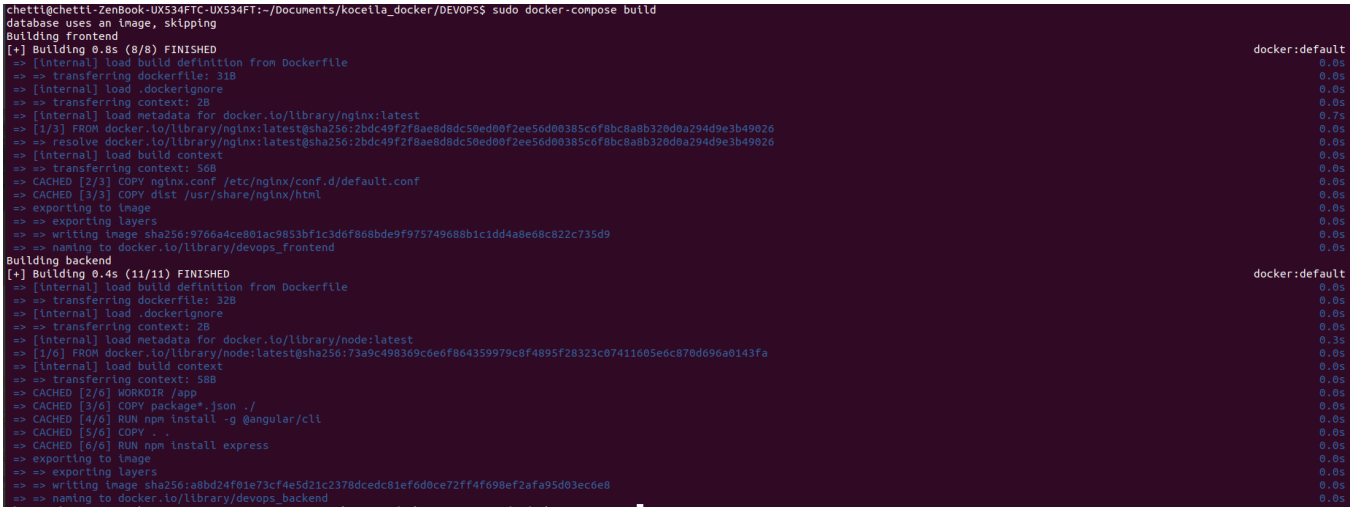
5/ Pour commencer voici l'arborescence du projet

Afin d'utiliser docker compose pour déployer les conteneurs, un fichier docker-compose.yml est requis, dans notre cas il contient la configuration suivante:

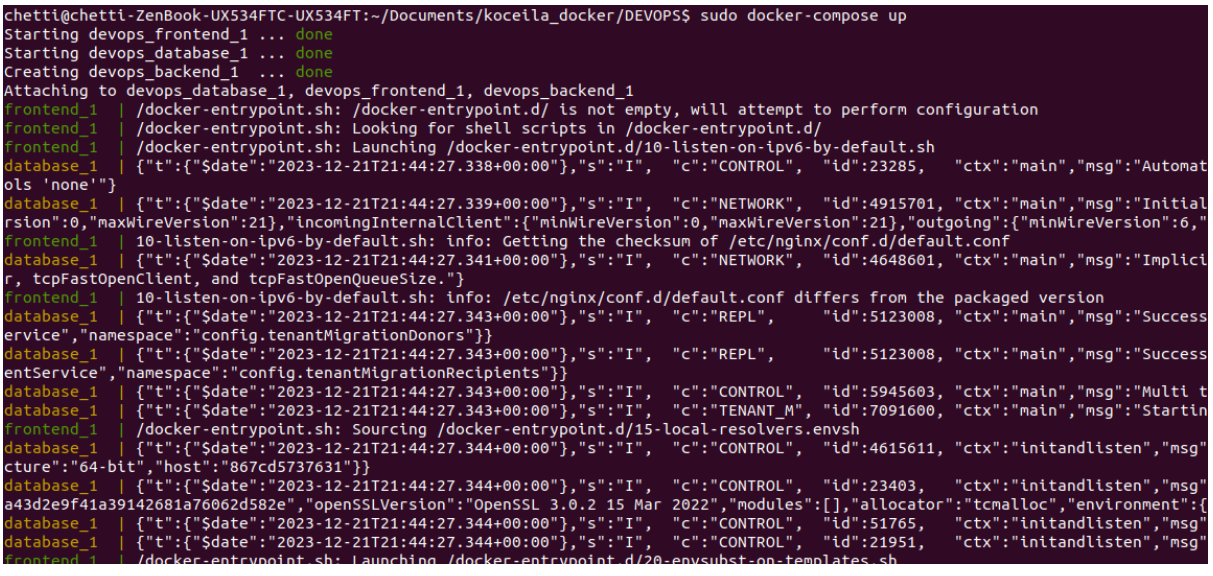
```
version: '3.7'
services:
 frontend:
   build: ./frontend
   ports:
```

```
    - 80:80

backend:
  build: ./backend
  ports:
    - 3000:3000
  depends_on:
    - database

database:
  image: mongo
  ports:
    - 27017:27017
```

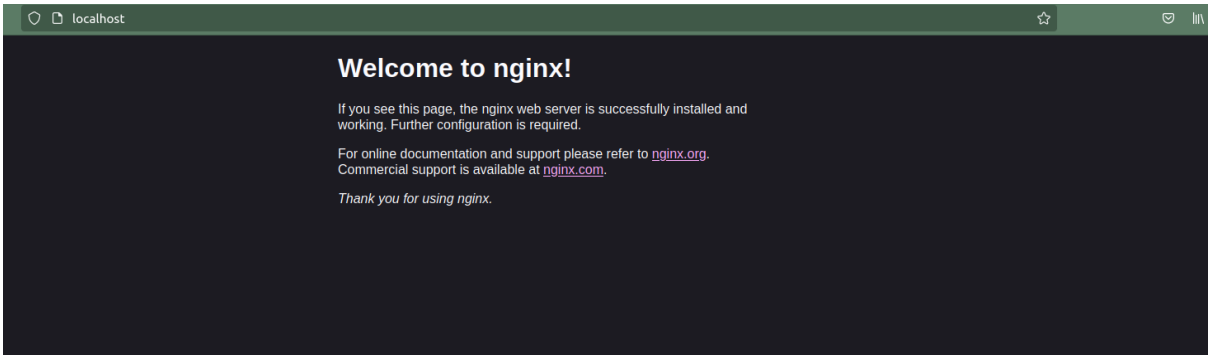
6/ pour tester l'application il faut :
faire un docker-compose build, pour build les dockerfiles



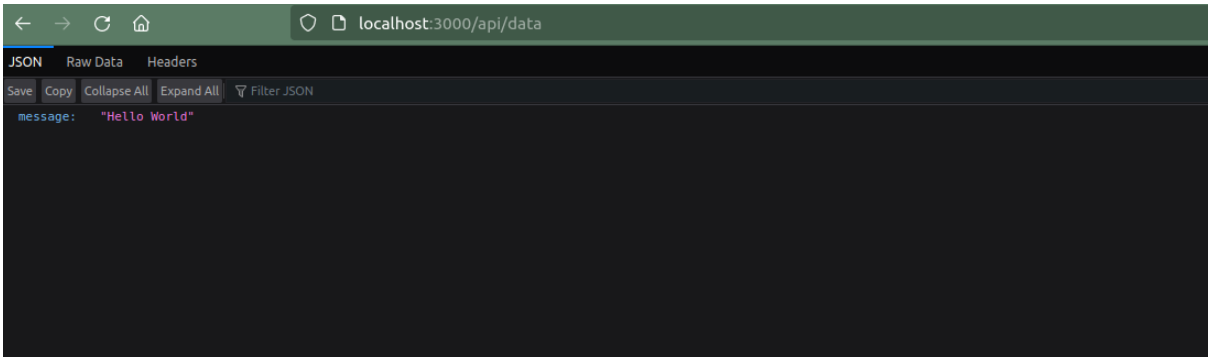
ensuite faire un docker-compose up pour démarrer les conteneur



si on se connecte à http://localhost on a cet affichage :



et si on se connecte à l'adresse suivante http://localhost:3000/api/data on a cette affichage



Partie 3 : Réseaux Docker

7/ pour configurer des réseaux docker personnalisé il faut:
créer un réseau en utilisant cette commande:
sudo docker network create –driver bridge my_bridge_network

```
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$ sudo docker network create --driver bridge my_bridge_network
ede3a24528a117335c62bd1c9fb74fca0c1ebf3c27eac0ed342383577ba4aee8
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$
```

il faut ensuite modifier le docker-compose.yml comme suite :

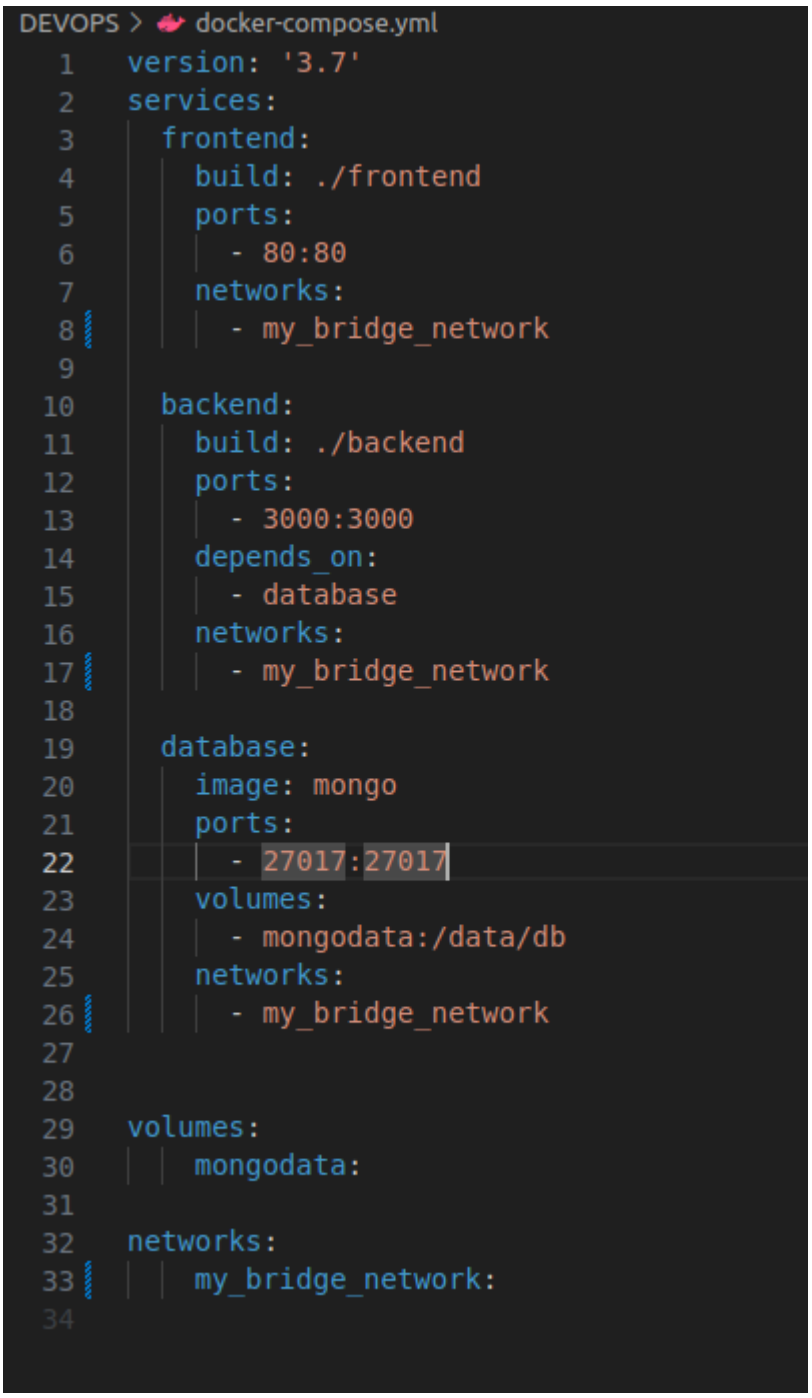
en ajoutant à la ligne 7, 16 et 24 le network créé et à la ligne 29 le networks à créer s'il n'est pas déjà fait

DEVOPS > docker-compose.yml

```yaml
version: '3.7'
services:
  frontend:
    build: ./frontend
    ports:
      - 80:80
    networks:
      - my_bridge_network

  backend:
    build: ./backend
    ports:
      - 3000:3000
    depends_on:
      - database
    networks:
      - my_bridge_network

  database:
    image: mongo
    ports:
      - 27017:27017

    networks:
      - my_bridge_network


networks:
    my_bridge_network:
```

Partie 4 : Volumes Docker

9/ pour ce faire on modifie la configuration du fichier docker-compose.yml comme suite:
en ajoutant à la ligne 23 le nom du volume et à la ligne 29 le volume à créer si ce n'est pas encore fait



10/ après avoir arrêter les conteneur en utilisant
sudo docker-compose down

on peut vérifier le contenue du volume en utilisant la commande
sudo docker volumes inspect mogodata

On remarque que les données enregistrées par la base sont toujours présentes

```
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$ sudo docker volume  inspect mongodata
[
    {
        "CreatedAt": "2023-12-21T20:17:17+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/snap/docker/common/var-lib-docker/volumes/mongodata/_data",
        "Name": "mongodata",
        "Options": null,
        "Scope": "local"
    }
]
```