

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 répertoire git racine:

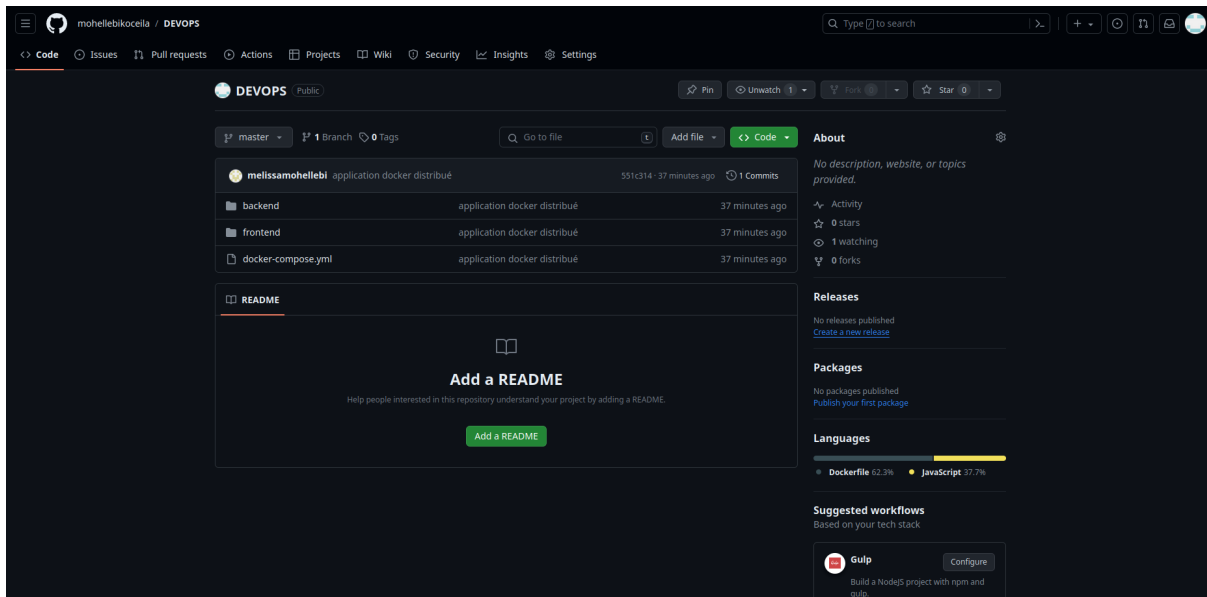
```
git add *
```

```
git commit -m "application distribué avec docker"
```

```
git push
```

```
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPSS$ 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/DEVOPSS$ git config --global credential.helper store
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPSS$
(chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPSS$ sudo apt install^Citk
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPSS$
```

Voici un aperçue sur GitHub



voici le lien git du dépôt : <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 répertoire 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

```
chetti@chetti-ZenBook-UX534FTC-UX534FT:~/Documents/koceila_docker/DEVOPS$ tree .
.
├── backend
│   ├── app.js
│   └── Dockerfile
├── docker-compose.yml
└── frontend
    ├── dist
    ├── Dockerfile
    └── nginx.conf

3 directories, 5 files
```

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

```

chett@chett-ZenBook-UX534FTC-UX534FT:~/Documents/kocella_docker/DEVOPS$ sudo docker-compose build
database uses an image, skipping
Building frontend
[+] Building 0.8s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [1/3] FROM docker.io/library/nginx:latest@sha256:2bdc49f2f8ae8d8dc50ed00f2ee56d00385c0f8bc8a8b320d8a294d9e3b49026
=> => resolve docker.io/library/nginx:latest@sha256:2bdc49f2f8ae8d8dc50ed00f2ee56d00385c0f8bc8a8b320d8a294d9e3b49026
=> [internal] load build context
=> => transferring context: 56B
=> CACHED [2/3] COPY nginx.conf /etc/nginx/conf.d/default.conf
=> CACHED [3/3] COPY dist /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:9766a4ce801ac9853bf1c3d6f86bde9f975749688b1c1dd4a8e68c822c735d9
=> => naming to docker.io/library/devops_frontend
Building backend
[+] Building 0.4s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:latest
=> [1/6] FROM docker.io/library/node:latest@sha256:73a9c498369c6e6f864359979c8f4895f728323c07411605e6c870d696a0143fa
=> [internal] load build context
=> => transferring context: 58B
=> CACHED [2/6] WORKDIR /app
=> CACHED [3/6] COPY package*.json ./
=> CACHED [4/6] RUN npm install -g @angular/cli
=> CACHED [5/6] COPY . .
=> CACHED [6/6] RUN npm install express
=> exporting to image
=> => exporting layers
=> => writing image sha256:a8bd24f01e73cf4e5d2c2378dcedc81ef6d0ce72ff4f698ef2afa95d03ec6e8
=> => naming to docker.io/library/devops_backend

```

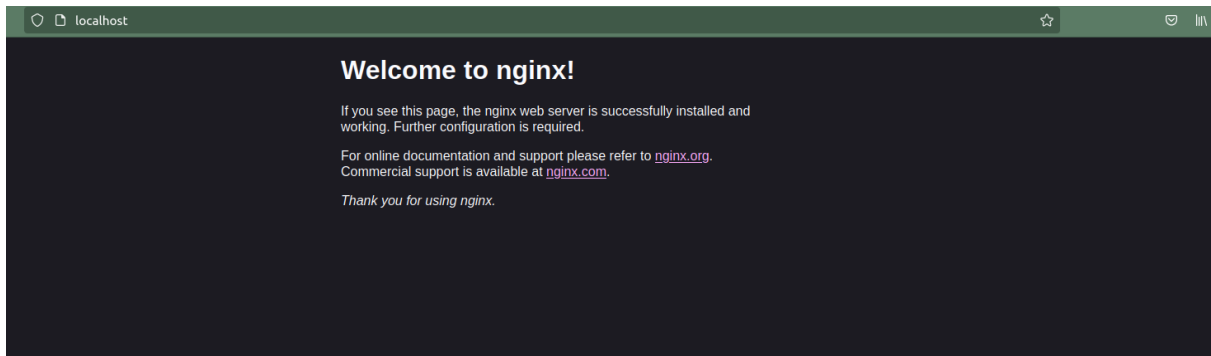
ensuite faire un docker-compose up pour démarrer les conteneur

```

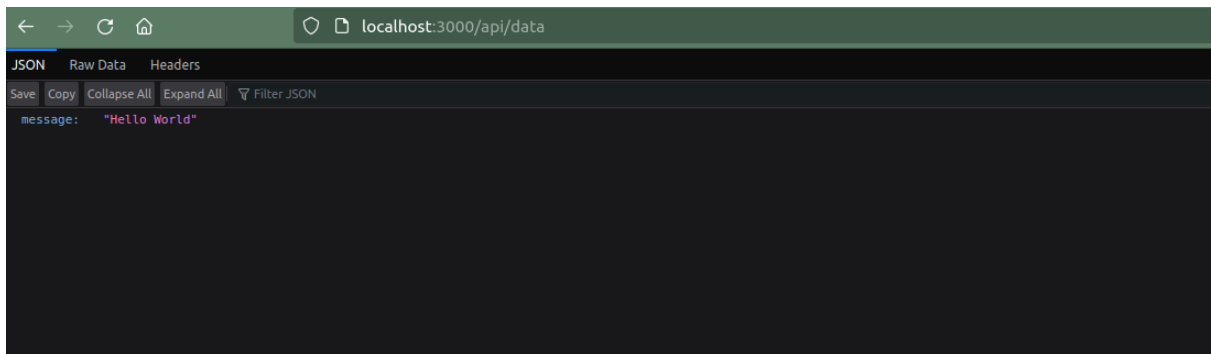
chett@chett-ZenBook-UX534FTC-UX534FT:~/Documents/kocella_docker/DEVOPS$ sudo docker-compose up
Starting devops_frontend_1 ... done
Starting devops_database_1 ... done
Creating devops_backend_1 ... done
Attaching to devops_database_1, devops_frontend_1, devops_backend_1
frontend_1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
frontend_1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
frontend_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
database_1 | {"t":{"$date":"2023-12-21T21:44:27.338+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automat
ols 'none'"}
database_1 | {"t":{"$date":"2023-12-21T21:44:27.339+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"main","msg":"Initial
rsion":0,"maxWireVersion":21},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"
frontend_1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
database_1 | {"t":{"$date":"2023-12-21T21:44:27.341+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main","msg":"Implicit
r, tcpFastOpenClient, and tcpFastOpenQueueSize."}
frontend_1 | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
database_1 | {"t":{"$date":"2023-12-21T21:44:27.343+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main","msg":"Success
ervice","namespace":"config.tenantMigrationDonors"}}
database_1 | {"t":{"$date":"2023-12-21T21:44:27.343+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main","msg":"Success
entService","namespace":"config.tenantMigrationRecipients"}}
database_1 | {"t":{"$date":"2023-12-21T21:44:27.343+00:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"main","msg":"Multi t
database_1 | {"t":{"$date":"2023-12-21T21:44:27.343+00:00"},"s":"I", "c":"TENANT_M", "id":7091600, "ctx":"main","msg":"Startin
frontend_1 | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
database_1 | {"t":{"$date":"2023-12-21T21:44:27.344+00:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten","msg"
cture":"64-bit","host":"867cd5737631"}
database_1 | {"t":{"$date":"2023-12-21T21:44:27.344+00:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg"
a43d2e9f41a39142681a76062d582e","opensslVersion":"OpenSSL 3.0.2 15 Mar 2022","modules":[],"allocator":"tcmalloc","environment":{"
database_1 | {"t":{"$date":"2023-12-21T21:44:27.344+00:00"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg"
database_1 | {"t":{"$date":"2023-12-21T21:44:27.344+00:00"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg"
frontend_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh

```

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
1  version: '3.7'
2  services:
3    frontend:
4      build: ./frontend
5      ports:
6        - 80:80
7      networks:
8        - my_bridge_network
9
10   backend:
11     build: ./backend
12     ports:
13       - 3000:3000
14     depends_on:
15       - database
16     networks:
17       - my_bridge_network
18
19   database:
20     image: mongo
21     ports:
22       - 27017:27017
23
24     networks:
25       - my_bridge_network
26
27
28
29   networks:
30     my_bridge_network:
31

```

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

```

DEVOPS > 🚀 docker-compose.yml
1  version: '3.7'
2  services:
3    frontend:
4      build: ./frontend
5      ports:
6        - 80:80
7      networks:
8        - my_bridge_network
9
10   backend:
11     build: ./backend
12     ports:
13       - 3000:3000
14     depends_on:
15       - database
16     networks:
17       - my_bridge_network
18
19   database:
20     image: mongo
21     ports:
22       - 27017:27017
23     volumes:
24       - mongodata:/data/db
25     networks:
26       - my_bridge_network
27
28
29   volumes:
30     mongodata:
31
32   networks:
33     my_bridge_network:
34

```

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

on peut vérifier le contenu du volume en utilisant la commande
 sudo docker volumes inspect mongodata

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"
  }
]
```