

TP Multi-services Docker

Toufik Frederick

MongoDB ° Express ° React ° NodeJs
Redis ° Elasticsearch ° Nginx

Tables des matiere

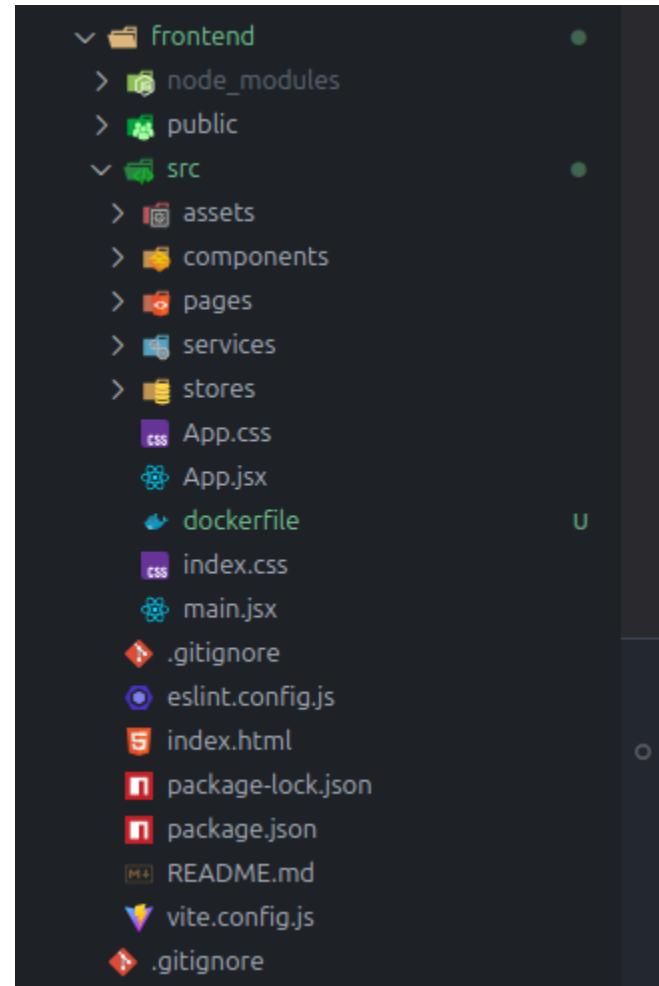
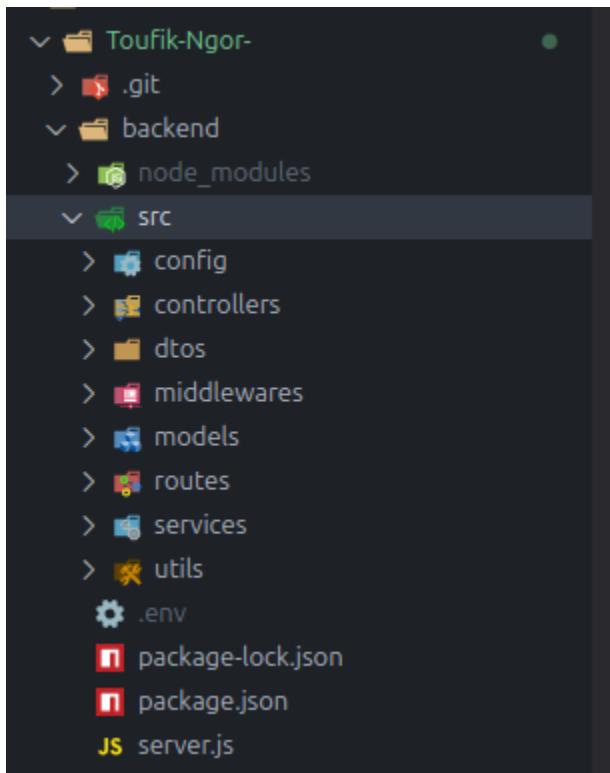
Etape 1 – Récupération du projet.....	2
Etape 2 – Configuration de Ngnix	2
Etape 3 – Configuration de MongoDB.....	4
Etape 4 – Configuration du dockerfile dans le backend.....	7
Etape 5 – Test du backend	7
Etape 6 – Frontend	9
Etape 7 - Docker-compose.yaml	11

Etape 1 – Récupération du projet

Avec un git clone comme le projet était déjà existant et sur github

```
o (base) ➤ ~/Documents/Docker/tp-docker ➤ git clone https://github.com/Neruaka/Toufik-Ngor-█
```

-- Voilà l'architecture du dossier :



Etape 2 – Configuration de Ngnix

- Il a ensuite fallu créer dossier Ngnix :

```
• (base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ➤ mkdir -p nginx
```

2. Ensuite il faut créer le fichier dockerFile pour nginx



The screenshot shows a code editor with a dark theme. In the top bar, there are tabs for "TP (1).pdf", ".../Dockerfile", ".../nginx", and ".../Dockerfile". Below the tabs, the file path is shown as "new > Toufik-Ngor- > nginx > Dockerfile > ...". The main content area contains a Dockerfile with the following code:

```
1 FROM nginx:alpine
2
3 RUN rm /etc/nginx/conf.d/default.conf
4
5 COPY nginx.conf /etc/nginx/conf.d/
6
7 EXPOSE 80
8
9 CMD ["nginx", "-g", "daemon off;"]
10
```

- a. Le run est pour supprimer la conf par defaut
- b. Puis on copie la conf que l'on va vers pour la mettre a la place

3. Puis du coup on fait le fichier conf personnalisé : (fichier conf fait par IA car je n'ai jamais utilisé nginx)

```
new > Toufik-Ngor-> nginx > N nginx.conf
  1 # Upstream pour le backend Node.js
  2 upstream backend_api {
  3   server backend:5000;
  4 }
  5
  6 # Upstream pour le frontend (en dev avec Vite)
  7 upstream frontend_dev {
  8   server frontend:3000;
  9 }
 10
 11 server {
 12   listen 80;
 13   server_name localhost;
 14
 15   # Logs
 16   access_log /var/log/nginx/access.log;
 17   error_log /var/log/nginx/error.log;
 18
 19   # Route API -> Backend Node.js
 20   location /api/ {
 21     # Retire le préfixe /api avant de transmettre
 22     rewrite ^/api/(.*) /$1 break;
 23
 24     proxy_pass http://backend_api;
 25     proxy_http_version 1.1;
 26     proxy_set_header Upgrade $http_upgrade;
 27     proxy_set_header Connection 'upgrade';
 28     proxy_set_header Host $host;
 29     proxy_set_header X-Real-IP $remote_addr;
 30     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 31     proxy_cache_bypass $http_upgrade;
 32   }
 33
 34   # Health check direct du backend
 35   location /health {
 36     proxy_pass http://backend_api/health;
 37     proxy_http_version 1.1;
 38     proxy_set_header Host $host;
 39   }
 40
 41   # Tout le reste -> Frontend React
 42   location / {
 43     proxy_pass http://frontend_dev;
 44     proxy_http_version 1.1;
 45     proxy_set_header Upgrade $http_upgrade;
 46     proxy_set_header Connection 'upgrade';
 47     proxy_set_header Host $host;
 48     proxy_cache_bypass $http_upgrade;
 49   }
 50 }
```

Etape 3 – Configuration de MongoDB

La il faut faire la même chose que pour nginx mais avec mongo

1. Creation du dossier mongodb

```
(base) > ~/Documents/Doc
mkdir -p mongodb
(base) > ~/Documents/Doc
```

2. Ensuite il faut faire le dockerfile avec es éléments correspondant pour se connecter a mongo :

```
FROM mongo:7.0
ENV MONGO_INITDB_ROOT_USERNAME=admin
ENV MONGO_INITDB_ROOT_PASSWORD=admin_password
ENV MONGO_INITDB_DATABASE=bibliotheque
EXPOSE 27017
```

- a. Pas de run ou de cmd car il démarre automatiquement avec mongod
3. Voilà ce que j'ai trouvé correspondant les variables de mongo

l'image officielle mongo détecte ces variables au premier démarrage et :

1. crée un utilisateur admin avec le username/password donné
2. crée la database spécifiée

4. Ensuite on test le build de mongo

a. Construire l'image

```
docker build -t tp-mongodb ./mongodb
=> sha256:211ac2b07d3c0a6629e7fe328b9467a585250b0d1a7bcf521d4e811ad0c3e44c 264B /
=> sha256:a901f9c6e1828f2577f03d2109e419f0b9ac35280ca0045a291a20a2b0a06dac 1.51MB
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:58c6c9c9143b7a57680294c4b7059d627ed53932a556c8c29f17
=> => exporting config sha256:8c41d817cc7a65bf4ff9e1c6604782a18d721a698c75906fde978d
=> => exporting attestation manifest sha256:7b0c6c5487eb1265aa1be3b26df305b200240c9e
=> => exporting manifest list sha256:df335b649a8cc35473c613785e17167b15754a8cd1ffa62
=> => naming to docker.io/library/tp-mongodb:latest
=> => unpacking to docker.io/library/tp-mongodb:latest

1 warning found (use docker --debug to expand):
- SecretsUsedInArgOrEnv: Do not use ARG or ENV instructions for sensitive data (ENV)
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

b. Ensuite avec un docker image on vérifie qu'elle a bien été créer

```
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker images | grep mongodb
WARNING: This output is designed for human readability. For machine-readable output, please use --format.
tp-mongodb:latest                                         df335b649a8c      1.13GB      280MB
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

c. Ensuite on test le conteneur

```
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker run -d --name test-mongo -p 27017:27017 tp-mongodb
62803file21e0cce46bd1844948da9bd5ddd3a3473078387d71c50c2bb2a15
docker: Error response from daemon: ports are not available: exposing port TCP 0.0.0.0:27017 -> 127.0.0.1:0: listen tcp 0.0.0.0:27017: bind: address already in use
Run 'docker run --help' for more information
(base) ✘ ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps --format
(base) ✘ ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps --format
flag needs an argument: --format
Usage: docker ps [OPTIONS]
Run 'docker ps -h' for more information
(base) ✘ ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps --format 'table {{.Names}}\t{{.Image}}\t{{.Ports}}' | grep 27017
(base) ✘ ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps | grep 27017
(base) ✘ ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± sudo ss -ltnp | grep ':27017'
[sudo] password for fred:
LISTEN 0      4096  127.0.0.1:27017  0.0.0.0:*   users:(("mongod",pid=2527,fd=9))
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± sudo systemctl stop mongod
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

- Ici on voit que le port 27017 est déjà pris donc j'ai vérifié s'il était pris pas un conteneur docker, et non, donc j'ai ensuite regardé si c'était sur linux directement.

- Et c'était bien le mongo de la machine qui prenait le port, je l'ai donc arrêté. Et j'ai ensuite démarré le conteneur via docker desktop

d. Faire un log pour vérifier qu'il est bien démarré:

```
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker logs test-mongo

MongoDB init process complete; ready for start up.

{"t": {"$date": "2025-12-16T19:28:46.474+00:00"}, "s": "T", "c": "NETWORK", "id": 4915701, "ctx": "main"
i. Et si l'on veut une réponse moins verbale on fait juste un ping
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker exec -it test-mongo mongosh --eval 'db.runCommand({ ping: 1 })'
{ ok: 1 }
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

e. Et pour finir on nettoie les tests

```
docker stop test-mongo && docker rm test-mongo
test-mongo
(base) [ ~ /Documents/Docker/new/Toufik-Ngor- ] main+ docker container ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
c1c502c7469d openjdk:27-ea-trixie "jshell" 5 hours ago Exited (0) 5 hours ago
d7b9a04bd25d neruaka/geting-started:latest "docker-entrypoint.s..." 9 hours ago Created
0cb18ba088bf node:lts-alpine "docker-entrypoint.s..." 9 hours ago Exited (1) 4 hours ago
664ffcbc505 ubuntu "bash" 9 hours ago Exited (0) 9 hours ago
cleaa736cc47 neruaka/geting-started:latest "docker-entrypoint.s..." 9 hours ago Exited (143) 9 hours ago
f216b18f1f64 ubuntu "/bin/bash" 27 hours ago Exited (130) 27 hours ago
6ed42d4fd4ec ubuntu "/bin/bash" 27 hours ago Exited (127) 27 hours ago
d2990aef83f4 ubuntu "/bin/bash" 27 hours ago Created
ce872659d06c postgres:18 "docker-entrypoint.s..." 28 hours ago Exited (255) 11 hours ago
10400b8fa9f7 eaabe2347d10 "python site_a_scrap..." 29 hours ago Exited (0) 29 hours ago
9a2e7636dd72 ubuntu "/bin/bash" 30 hours ago Exited (127) 30 hours ago
c07286fefc2b ubuntu "/bin/bash" 32 hours ago Exited (137) 30 hours ago
5593d9b8bbcd bash "docker-entrypoint.s..." 32 hours ago Exited (127) 32 hours ago
192c409ae84d bash "docker-entrypoint.s..." 32 hours ago Exited (127) 32 hours ago
e1927f127fe9 bash "docker-entrypoint.s..." 32 hours ago Exited (127) 32 hours ago
67a091b03e64 bash "docker-entrypoint.s..." 32 hours ago Exited (127) 32 hours ago
e14429740f4f hello-world "-d" 34 hours ago Created
8ca186bfb011 docker/welcome-to-docker "/docker-entrypoint..." 35 hours ago Exited (0) 30 hours ago
(base) [ ~ /Documents/Docker/new/Toufik-Ngor- ] main+ docker container ps
```

Il n'est plus dans la liste des conteneurs (Il n'y a pas en dessous de 5h pour le created donc on est large)

Etape 4 – Configuration du dockerfile dans le backend

1. Créer le fichier dockerfile :

```
new > Toufik-Ngor- > backend > Dockerfile > ...
1  FROM node:20-alpine
2
3  WORKDIR /appServer
4  COPY package*.json .
5  RUN npm install
6  COPY . .
7  EXPOSE 5000
8
9  CMD [ "node", "server.js" ]
10
```

2. Ensuite comme il n'y avait pas redis et elasticSearch dans le projet je l'ai ajouté au package.json
3. Ensuite il fallait ajouter les éléments suivants à mon server.js :
 - a. connexion à Elasticsearch
 - b. endpoint /health qui vérifie que MongoDB est accessible
 - c. endpoint /cache-test pour tester Redis
 - d. logging vers Elasticsearch
4. Une fois cela fait on build et on test

a. Build :

```
docker build -t tp-backend ./backend
[+] Building 22.4s (11/11) FINISHED
--> [internal] load build definition from Dockerfile
--> <-- transferring dockerfile: 158B
--> [internal] load metadata for docker.io/library/node:20-alpine
--> [auth] library/node:pull token for registry-1.docker.io
--> [internal] load .dockerignore
--> <-- transferring context: 2B
--> CACHED [1/5] FROM docker.io/library/node:20-alpine@sha256:643e7036aa985317ebfee460005e322aa550c6b6883000d01daefb58689a58e2
--> <-- resolve docker.io/library/node:20-alpine@sha256:643e7036aa985317ebfee460005e322aa550c6b6883000d01daefb58689a58e2
--> [internal] load build context
--> <-- transferring context: 15.56MB
--> [2/5] WORKDIR /appS
--> [3/5] COPY package*.json .
--> [4/5] RUN npm install
--> [5/5] COPY .
--> <-- exporting to image
--> <-- exporting layers
--> <-- exporting manifest sha256:ef6361185e8051dcab3fb1f68143dee9e9e1ef845725cd25f0d68c00e0fe65
--> <-- exporting config sha256:2c82c96833e86074d78be2025f01f14db19954f6756ccf580687cdc5955b0cb2
--> <-- exporting attestation manifest sha256:743cec647c5760d50ce37c0fa53f6b65a2abe5ea855f8bb7d9d2acealb5bc6
--> <-- exporting manifest list sha256:c6b263e1deb9e7d08a9b0b8ab969697cacfd650f6e6bbba1a753697152bb04
--> <-- naming to docker.io/library/tp-backend:latest
--> <-- unpacking to docker.io/library/tp-backend:latest
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

Etape 5 – Test du backend

Maintenant que l'on sait que tout nos services backend sont fait il faut les tester, pour ce faire il faut mongodb soit up sinon ca crash, on va donc faire un network pour tout tester .

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker network create tp-network
4b3d8030498b2dea1d2fde4ff5c0f28f8b0f63b7537a6890aa4de910ela44bcc
○ (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

Une fois que le serveur est créé on lance mongo dessus et on vérifie qu'il est bien up :

```
(base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker run -d \
--name mongodb \
--network tp-network \
-p 27017:27017 \
--mongod
b8a6c3b6e51b4e43f135247cd8c40252db0808c5d92bce149072bef7f2175edc
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps | grep mongo
b8a6c3b6e51b_tp-mongodb "docker-entrypoint.s..." 28 seconds ago Up 28 seconds 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp mongodb
○ (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

Ensuite on lance le backend sur le même réseau et on vérifie aussi :

```
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker run -d \
--name backend \
--network tp-network \
-p 5000:5000 \
-e MONGODB_URI="mongodb://admin:admin_password@mongodb:27017/bibliotheque?authSource=admin" \
--backend
bad8d0b6a83f1fc7fea2424a0d18dba1286aa7e551f1fae319e6964f0afab7e6
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker logs backend
[dotenv@17.2.3] injecting env (2) from .env -- tip: ✨ add observability to secrets: https://dotenvx.com/ops
✓ MongoDB connecté
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps | grep backend
bad8d0b6a83f_tp-backend "docker-entrypoint.s..." 18 seconds ago Up 17 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp backend
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main± docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bad8d0b6a83f_tp-backend "docker-entrypoint.s..." 23 seconds ago Up 22 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp backend
b8a6c3b6e51b_tp-mongodb "docker-entrypoint.s..." 6 minutes ago Up 6 minutes 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp mongodb
○ (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main±
```

Et ensuite je test le endpoints avec curl

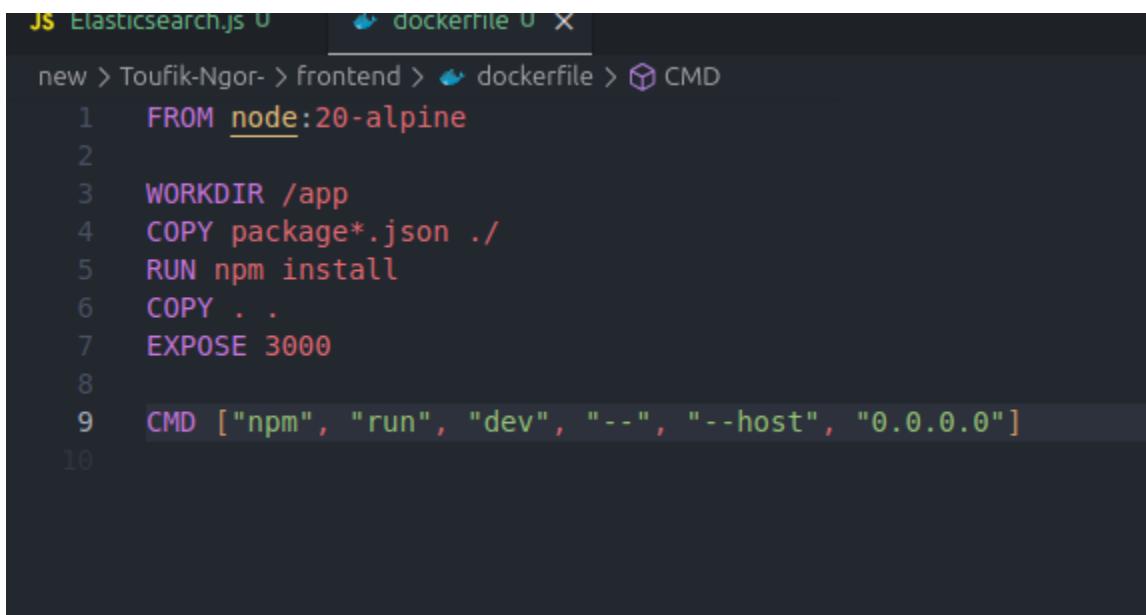
```
(base) ✘ ~/Documents/Docker/new/Toufik-Ngor- main± curl http://localhost:5000/health
{"status": "OK", "message": "Tous les services critiques sont opérationnels", "details": {"mongodb": true, "redis": false, "elasticsearch": false}}
```

Redis et elasticsearch sont en false et passeront à true avec le docker compose.

Donc maintenant que l'on sait que le back est OKAY on passe au front.

Etape 6 – Frontend

1. Tout d'abord il faut faire le dockerfile :



```
JS Elasticsearch.js ✘ dockerfile ✘ CMD
new > Toufik-Ngor- > frontend > dockerfile > CMD
1  FROM node:20-alpine
2
3  WORKDIR /app
4  COPY package*.json .
5  RUN npm install
6  COPY . .
7  EXPOSE 3000
8
9  CMD ["npm", "run", "dev", "--", "--host", "0.0.0.0"]
10
```

- pourquoi --host 0.0.0.0 ?
- par défaut, Vite écoute sur localhost (127.0.0.1), ce qui veut dire qu'il n'accepte que les connexions depuis le même conteneur.
- avec --host 0.0.0.0, Vite écoute sur toutes les interfaces réseau, donc il accepte les connexions depuis l'extérieur du conteneur (nginx, navigateur, etc.)

2. Ensuite on build et on test:

```

• (base) ➜ ~/Documents/Docker/new/Toufik-Ngor ➜ main± docker build -t tp-frontend ./frontend
[+] Building 27.5s (11/11) FINISHED
  => [internal] load build definition from dockerfile
  => => transferring dockerfile: 186B
  => [internal] load metadata for docker.io/library/node:20-alpine
  => [auth] library/node:pull token for registry-1.docker.io
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/5] FROM docker.io/library/node:20-alpine@sha256:643e7036aa985317ebfee460005e322aa550c6b6883000d01daefb58689a58e2
  => => resolve docker.io/library/node:20-alpine@sha256:643e7036aa985317ebfee460005e322aa550c6b6883000d01daefb58689a58e2
  => [internal] load build context
  => => transferring context: 74.80MB
  => CACHED [2/5] WORKDIR /app
  => [3/5] COPY package*.json .
  => [4/5] RUN npm install
  => [5/5] COPY .
  => exporting to image
  => => exporting layers
  => => exporting manifest sha256:76bd68f717b65f27a4f9fdd1d2338d63210de467316e6e88f01cacc6a0e12176
  => => exporting config sha256:034ed1c632d03b496f8a5d224210952a81cad348f9f14d441dada2df70a1195
  => => exporting attestation manifest sha256:0e80af8eb690b2f509759e8a0b731308c6b79a8ad3d469f7da3bf32c219063cb
  => => exporting manifest list sha256:17bc94cbe5aeece942fe81d7c4e0104d7e5ba2192709b889baf6a1922c544d006
  => => naming to docker.io/library/tp-frontend:latest
  => => unpacking to docker.io/library/tp-frontend:latest
○ (base) ➜ ~/Documents/Docker/new/Toufik-Ngor ➜ main±

• (base) ➜ ~/Documents/Docker/new/Toufik-Ngor ➜ main± docker run -d \
  --name frontend \
  -p 3000:3000 \
  tp-frontend
b4dff6b6620c02675016d8778d3ce1c037f533d652c09f017889adabdd0234
• (base) ➜ ~/Documents/Docker/new/Toufik-Ngor ➜ main± docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
b4dff6b6620 tp-frontend "docker-entrypoint.s..." 8 seconds ago Up 7 seconds 0.0.0.0:3000->3000/tcp, [:]:3000->3000/tcp frontend
658e3650ec2a tp-backend "docker-entrypoint.s..." 11 minutes ago Up 11 minutes 0.0.0.0:5000->5000/tcp, [:]:5000->5000/tcp backend
3c5f7bc58633 tp-mongodb "docker-entrypoint.s..." 11 minutes ago Up 11 minutes 0.0.0.0:27017->27017/tcp, [:]:27017->27017/tcp mongodb
○ (base) ➜ ~/Documents/Docker/new/Toufik-Ngor ➜ main±

```

3. Maintenant il faut adapter le vite.config de docker :

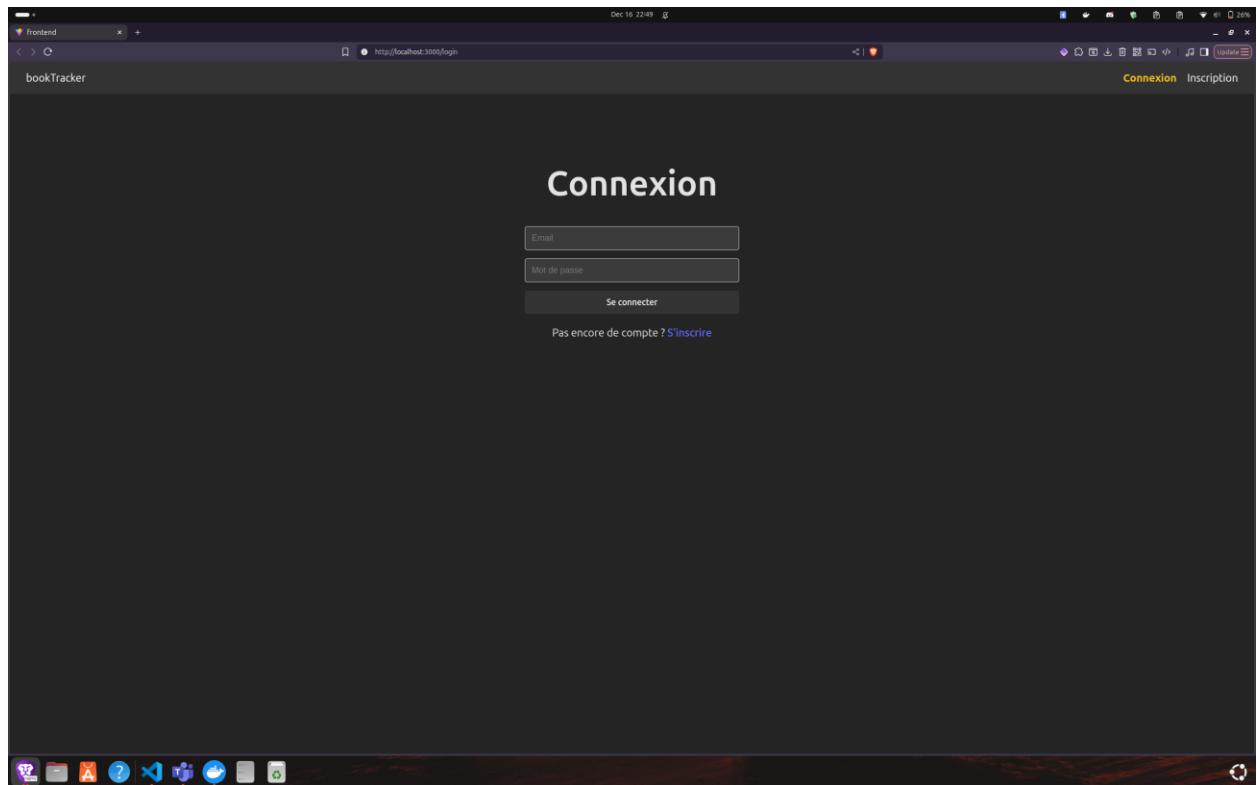
```

You, 3 minutes ago | 2 authors (You and one other)
1 import { defineConfig } from 'vite'
2 import react from '@vitejs/plugin-react'
3
4 export default defineConfig({
5   plugins: [react()],
6   server: {
7     port: 3000,
8     host: '0.0.0.0', // Important pour Docker
9     watch: {
10       usePolling: true // Nécessaire pour le hot reload dans Docker
11     }
12   }
13 }) You, 3 minutes ago • Uncommitted changes

```

- Il a fallu ajouter server: avec le port/host et le watch

4. Et la quand j'ouvre localhost:3000 j'ai bien mon front



5. Ensuite on nettoie tout ca :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
● (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- docker stop frontend && docker rm frontend
frontend
frontend
○ (base) ➜ ~/Documents/Docker/new/Toufik-Ngor- docker rm frontend
```

Etape 7 - Docker-compose.yaml

1. Tout d'abord il faut centraliser les variables d'environnement pour redis, elasticsearch et mongodb dans un fichier .env a la racine
2. Puis on fait le docker compose
 - a. Depends_on : ca dit a docker compose de lancer mongo et reddit avant le backend et garantie l'ordre de demarrage
 - b. Networks : on met tout sur le meme network pour que les services puissent parler entre eux via leurs noms
 - c. Volumes: c'est pour la persistance des données
 - d. Deploy.resources.limits.memory: limite la mémoire ram du conteneur
 - e. Et le restart: unless-stopped c'est si le conteneur crash, docker le redem automatiquement

3. Ensuite on lance l'archi entière :

- a. Ps: ne pas oublier de fermer les anciens conteneurs

```
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+ docker compose up -d --build
=> CACHED [frontend 4/5] RUN npm install
=> CACHED [frontend 5/5] COPY .
=> [frontend] exporting to image
=> => exporting layers
=> => exporting manifest sha256:731172ad7a1c488ebaf37629b752bede6c0467fc3b2f8211638fe883406649fc
=> => exporting config sha256:19d55eb87e7795f384ae29123a10a4e6ae0a96396d9e9ec1eeaea6041dfb083e
=> => exporting attestation manifest sha256:5d4a9569a004668840a0d21239ef4fa7e59e0396d82bd0ced5e45a24b0f6f6
=> => exporting manifest list sha256:33539772e6e355ae09e82e5ae63fd6058d5b89734b6ad4aad784dd0e92e1d10c
=> => naming to docker.io/library/toufik-ngor--frontend:latest
=> => unpacking to docker.io/library/toufik-ngor--frontend:latest
=> [mongodb] resolving provenance for metadata file
=> [backend] resolving provenance for metadata file
=> [nginx] resolving provenance for metadata file
=> [frontend] resolving provenance for metadata file
[+] Running 10/10
✓ toufik-ngor--mongodb    Built
✓ toufik-ngor--backend   Built
✓ toufik-ngor--frontend  Built
✓ toufik-ngor--nginx     Built
✓ Container mongodb      Started
✓ Container backend      Started
✓ Container frontend     Started
✓ Container nginx        Started
✓ Container elasticsearch Started
✓ Container redis         Started
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+ ⌘
△ 0 docker-desktop default
```

4. Puis on vérifie que tout tourne :

NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS	PORTS
backend	toufik-ngor--backend	"docker-entrypoint.s_"	backend	52 seconds ago	Up 49 seconds	0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
elasticsearch	docker.elastic.co/elasticsearch/elasticsearch:8.11.0	"bin/elasticsearc...	elasticsearch	2 minutes ago	Up 50 seconds	0.0.0.0:3200->9200/tcp, [::]:9200->9200/tcp
frontend	toufik-ngor--frontend	"docker-entrypoint.s_"	frontend	52 seconds ago	Up 48 seconds	0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp
mongodb	toufik-ngor--mongodb	"docker-entrypoint.s_"	mongodb	53 seconds ago	Up 50 seconds	0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp
nginx	toufik-ngor--nginx	"docker-entrypoint.s_"	nginx	51 seconds ago	Up 47 seconds	0.0.0.0:80->80/tcp, [::]:80->80/tcp
redis	redis:7-alpine	"docker-entrypoint.s_"	redis	2 minutes ago	Up 50 seconds	0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp

5. Et enfin on finit par tester tout les endpoints

```
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+ curl http://localhost:5000/health
{"status":"OK","message":"Tous les services critiques sont opérationnels","details":{"mongodb":true,"redis":true,"elasticsearch":true}}%
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+ curl http://localhost:5000/cache-test
{"message":"Cache Redis fonctionne !","visits":2}%
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+ curl http://localhost/health
{"status":"OK","message":"Tous les services critiques sont opérationnels","details":{"mongodb":true,"redis":true,"elasticsearch":true}}%
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+ curl http://localhost/api/health
{"status":"OK","message":"Tous les services critiques sont opérationnels","details":{"mongodb":true,"redis":true,"elasticsearch":true}}%
(base) ➤ ~/Documents/Docker/new/Toufik-Ngor- ⌘ main+
```

Maintenant que tout fonctionne on va tester le site et si l'inscription / connexion fonctionne c'est que tout est okay 😊

