Principes de base de Docker et interface de ligne de commande

1. Télécharger l'image officielle **nginx** depuis Docker Hub

Je vérifie si docker est installé :

`docker —version`

```
Docker version 28.5.1, build e180ab8
```

Je télécharge l'image officielle de nginx :

`docker pull nginx:latest`

Je vérifie les images installées :

`docker images`

```
nginx        latest    c881927c4077   2 weeks ago
255MB
```

2. Lancer un conteneur à partir de cette image en mode détaché, en mappant le port 8080 de l'host au port 80 du conteneur

`docker run -d --name nginx-gomycode -p 8080:80 nginx:latest`

Je liste les conteneurs actifs:

`docker ps`

```
816d344b1d1b   nginx:latest   "/docker-
entrypoint.…"   53 seconds ago   Up 53 seconds
0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   nginx-
gomycode
```

Dans le navigateur je lance : `http://localhost:8080` pour verifier si nains tourne correctement

3. Utiliser **docker exec** pour entrer dans le conteneur et afficher le contenu de : **/usr/share/nginx/html**

`docker exec -it nginx-gomycode sh`

A l'invite de commande, je lance la commande : `cd /usr/share/nginx/html` pour me positionner dans le repertoire html

Ensuite je lance la commande : `ls` pour voir le contenu du repertoire

```
50x.html  index.html
```

Je peux aussi lancer la commande : `cat index.html` pour lire le contenu du fichier index.html

4. Redémarrer le conteneur et vérifier qu'il conserve sa configuration

Pour redémarrer le conteneur :

`docker restart nginx-gomycode`

Pour verifier si le conteneur tourne après le redémarrage

docker ps —filter ‘’name=nginx-gomycode”

```
816d344b1d1b   nginx:latest   "/docker-entrypoint.…"   27 minutes ago   Up 2 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   nginx-gomycode
```

Je peux aussi vérifier que la configuration est sauvegardée en modifiant le fichier index.html, après le redémarrage, le fichier est toujours conservé:

docker exec -it nginx-gomycode sh

installe nano : apt update && apt install nano -y

nano **/usr/share/nginx/html/index.html**

Après modification et sauvegarde, je redémarre le conteneur : docker restart nginx-gomycode

Je relance le navigateur pour verifier si la configuration n’est pas perdue.

5. Supprimer le conteneur et l’image

J’arrete le conteneur : docker stop nginx-gomycode

Je supprime le conteneur : docker rm nginx-gomycode

Je supprime l’image : docker rmi nginx:latest

Images Docker et Dockerfiles

1. Creation de mon projet python (gomycode-python-app)

Les fichiers de mon projet :

- app.py
- requirements.txt

2. Création du fichier Dockerfile

- Dockerfile

NB: je le crée à la racine du projet

3. Utilise python:3.12-slim comme image de base

```
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .
```

```
EXPOSE 5000

CMD ["python", "app.py"]
```

4. Construit l'image personnalisé

```
docker build -t gomycode-python-app:v1 .
```

5. Lance le conteneur à partir de l'image:

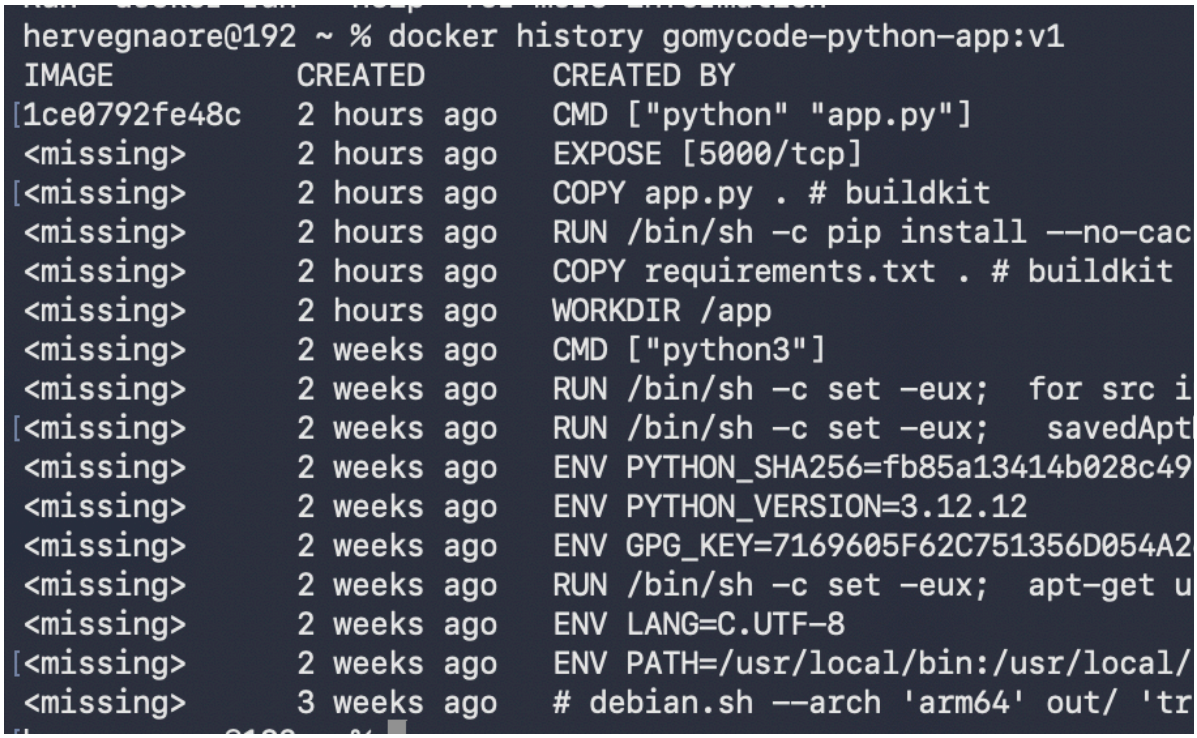
```
docker run -d —name python-gomycode -p 5001:5000 gomycode-python-app:v1
```

NB: j'ai utiliser le port 5001 de ma machine car le port 5000 est déjà occupé.

Pour vérifier que l'application fonctionne je lance dans mon navigateur : http://localhost:5001

6. Inspecter les couches de l'image :

```
docker history gomycode-python-app:v1
```



Volumes et stockage Docker

1. Creation d'un volume nommé mysql_data

```
docker volume create mysql_data
```

2 . Vérifie que le volume est créé :

```
docker volume ps | grep mysql_data
local     mysql_data
```

3. Lance le conteneur MySQL

```
docker run -d \
 --name mysql-gomycode \
 -e MYSQL_ROOT_PASSWORD=gomycodepass \
 -e MYSQL_DATABASE=gomycodedb \
 -v mysql_data:/var/lib/mysql \
 mysql:8
```

- Vérifie si mon conteneur est bien créé : `docker ps | grep mysql-gomycode`
- Se connecter à MySQL : `docker exec -it mysql-gomycode mysql -u root -p`
- Après avoir entré mon mot de passe root, je me connecte à la base de données : `USE gomycodedb`
- Création de la table users :

```
CREATE TABLE users (
 id INT AUTO_INCREMENT PRIMARY KEY,
 name VARCHAR(100)
);
```

- Insertion de données :

```
INSERT INTO users (name) VALUES ("Hervé"), ("Allassane");
```

- Arrêter et supprimer le conteneur:

```
docker stop mysql-gomycode
```

```
docker rm mysql-gomycode
```

- Relancer un conteneur avec le même volume

```
docker run -d \
 --name mysql-gomycode \
 -e MYSQL_ROOT_PASSWORD=gomycodepass \
 -e MYSQL_DATABASE=gomycodedb \
 -v mysql_data:/var/lib/mysql \
 mysql:8
```

- Je vérifie que mes données persistent :

```
docker exec -it mysql-gomycode mysql -u root -p
USE gomycodedb
SELECT * FROM users;
```

Réseau Docker

1. Création du réseau internal-net

```
docker network create internal-net
```

Pour verifier la création : `docker network ls | grep internal-net`

```
5e0b32c93b6c   internal-net   bridge    local
```

2. Lance deux conteneurs

```
docker run -d --name web --network internal-net nginx
docker run -it --name client --network internal-net alpine sh
```

NB: le conteneur client est lancer en mode interactif, ce qui me permet d'être à l'intérieur pour executer des commandes:

Lance un ping depuis client : `ping web`

```
/ # ping web
PING web (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.252 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.236 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.254 ms
64 bytes from 172.18.0.2: seq=3 ttl=64 time=0.247 ms
64 bytes from 172.18.0.2: seq=4 ttl=64 time=0.256 ms
```

3. Expose une petite application sur un conteneur:

```
docker run -d \
 --name app \
 --network internal-net \
 python:3.12-slim \
 python -m http.server 8000
```

4. Depuis l'autre conteneur, faire un curl vers cette app via le nom du conteneur

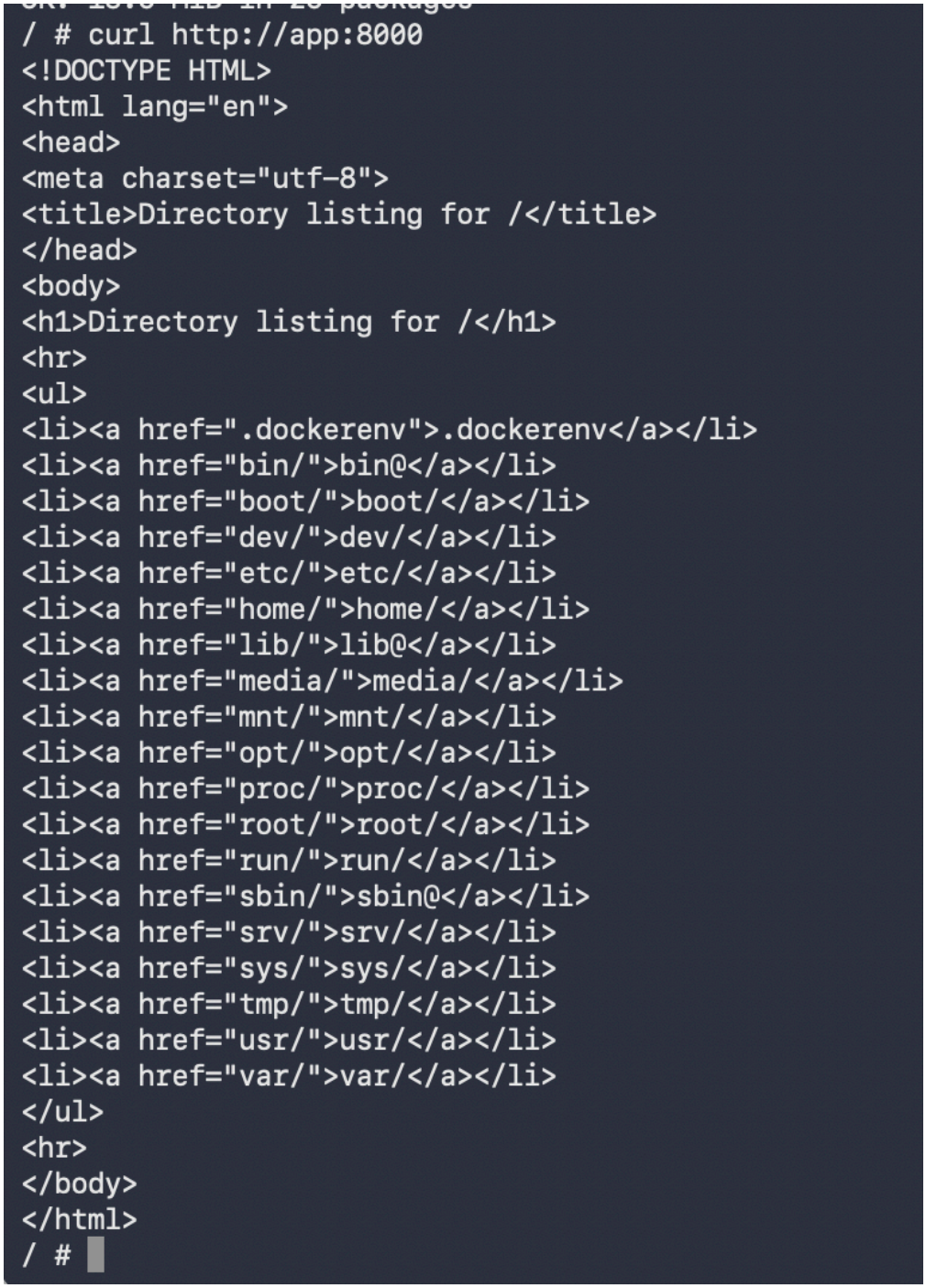
Je me connecte au conteneur client en mode interactif:

```
docker start client
docker exec -it client sh
```

Pour lancer curl sur mon conteneur client, je dois l'installer :

```
apk update
apk add curl
```

Execute la commande : `curl http://app:8000`



Docker Compose

1. Creation d'un dossier dockerCompose qui va contenir les deux applications backend et nginx

```
mkdir dockerCompose
cd dockerCompose
mkdir backend nginx
```

2. Creation de l'application backend:

Ajout des fichiers (app.py et requirements.txt) avec leur contenu

nano backend/app.py

nano backend/requirements.txt

Creation du Dockerfile à partir de python:3.12-slim

nano backend/Dockerfile

3. Creation de l'application nginx:

Ajout du fichier (default.conf) avec son contenu

Creation du Dockerfile à partir de nginx:alpine

4. Creation du fichier docker-compose.yml

Je me positionne à la racine du dossier « dockerCompose » qui contient « backend et nginx »

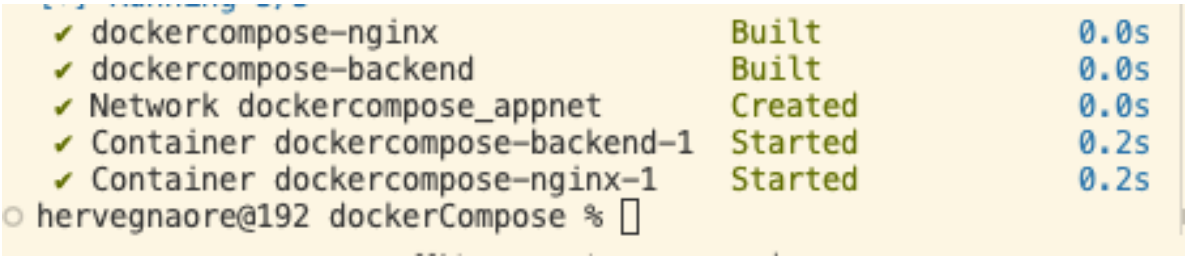
Le contenu de mon docker-compose.yml :

```
version: "3.9"

services:
  backend:
    build: ./backend
    networks:
      - appnet

  nginx:
    build: ./nginx
    ports:
      - "8080:80"
    depends_on:
      - backend
    networks:
      - appnet
```

```
networks:
  appnet:
```

NB:
La version du docker-compose est 3.9, les services exécutés sont : backend et nginx, tous deux à la racine du projet dans les dossiers respectif `backend et nginx.`
Tous deux utilise le réseau : appnet, on demande explicitement de lancer le service backend avant nginx. Le port 80 du conteneur nginx est mappé sur le port 8080 de l'host.

5. Je lance la stack : `docker-compose up -d`



6. Test dans le navigateur :
   `http://localhost:8080`
7. Vérifications des logs :
   `docker-compose logs`

```
hervegnaore@192 dockerCompose % docker-compose logs
WARN[0000] /Users/hervegnaore/Documents/FORMATION DEVOPS/dockerC
emove it to avoid potential confusion
backend-1  |  * Serving Flask app 'app'
backend-1  |  * Debug mode: off
backend-1  | WARNING: This is a development server. Do not use i
backend-1  |  * Running on all addresses (0.0.0.0)
backend-1  |  * Running on http://127.0.0.1:5000
backend-1  |  * Running on http://172.19.0.2:5000
backend-1  | Press CTRL+C to quit
backend-1  | 172.19.0.3 - - [02/Feb/2026 21:00:08] "GET / HTTP/1
nginx-1    | /docker-entrypoint.sh: /docker-entrypoint.d/ is not
nginx-1    | /docker-entrypoint.sh: Looking for shell scripts in
nginx-1    | /docker-entrypoint.sh: Launching /docker-entrypoint
nginx-1    | 10-listen-on-ipv6-by-default.sh: info: Getting the
nginx-1    | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/c
nginx-1    | /docker-entrypoint.sh: Sourcing /docker-entrypoint.
nginx-1    | /docker-entrypoint.sh: Launching /docker-entrypoint
nginx-1    | /docker-entrypoint.sh: Launching /docker-entrypoint
nginx-1    | /docker-entrypoint.sh: Configuration complete; read
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: using the "epoll"
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: nginx/1.29.4
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: built by gcc 15.2
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: OS: Linux 6.11.11
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: getrlimit(RLIMIT_
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 2026/02/02 20:48:45 [notice] 1#1: start worker proc
nginx-1    | 151.101.0.223 - - [02/Feb/2026:21:00:08 +0000] "GET
(KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36" "-"
hervegnaore@192 dockerCompose %
```

Les logs montre que l'attribut « version » est obsolète et aussi que mon projet n'est pas optimisé pour la production

NB: J’ai envoyé certaines images sur mon hub.

Processus:

1 - Creer le repository sur le hub (https://hub.docker.com/)

2 - Mettre un tag sur l’image : docker tag dockercompose-backend hgnaore/dockercompose-backend:V1

3- Push sur le hub : docker push hgnaore/dockercompose-backend:V1

Lien des images sur le hub:

https://hub.docker.com/r/hgnaore/gomycode-python-app

https://hub.docker.com/r/hgnaore/dockercompose-backend

https://hub.docker.com/r/hgnaore/dockercompose-nginx