

DOCUMENTATION TECHNIQUE : INFRASTRUCTURE DE L'APPLICATION

Table des matières

PREREQUIS :	3
STACK TECHNIQUE :	3
SERVEUR WEB CLIENT :	4
SERVEUR BASE DE DONNÉES :	4
Création du container :	4
Création de la base de données :	5
SERVER API USER :	7
Création du container :	7
Connexion du container de l'api à celui de la base de données :	10
Création du réseau :	10
Ajout de la base de données au réseau :	10
Ajout de l'API au réseau :	10
SERVER API POSTS :	10
Création du container :	10
Connexion du container de l'api à celui de la base de données :	13
Ajout de l'API au réseau :	13

PREREQUIS :

- Docker
- Logiciel de gestion / administration de base de données (MySQL Workbench par exemple)
- Postman (pas nécessaire au fonctionnement mais utile pour test les APIs)

STACK TECHNIQUE :

- Node
- ExpressJS
- Ngnix
- MySQL

SERVEUR WEB CLIENT :

SERVEUR BASE DE DONNÉES :

Création du container :

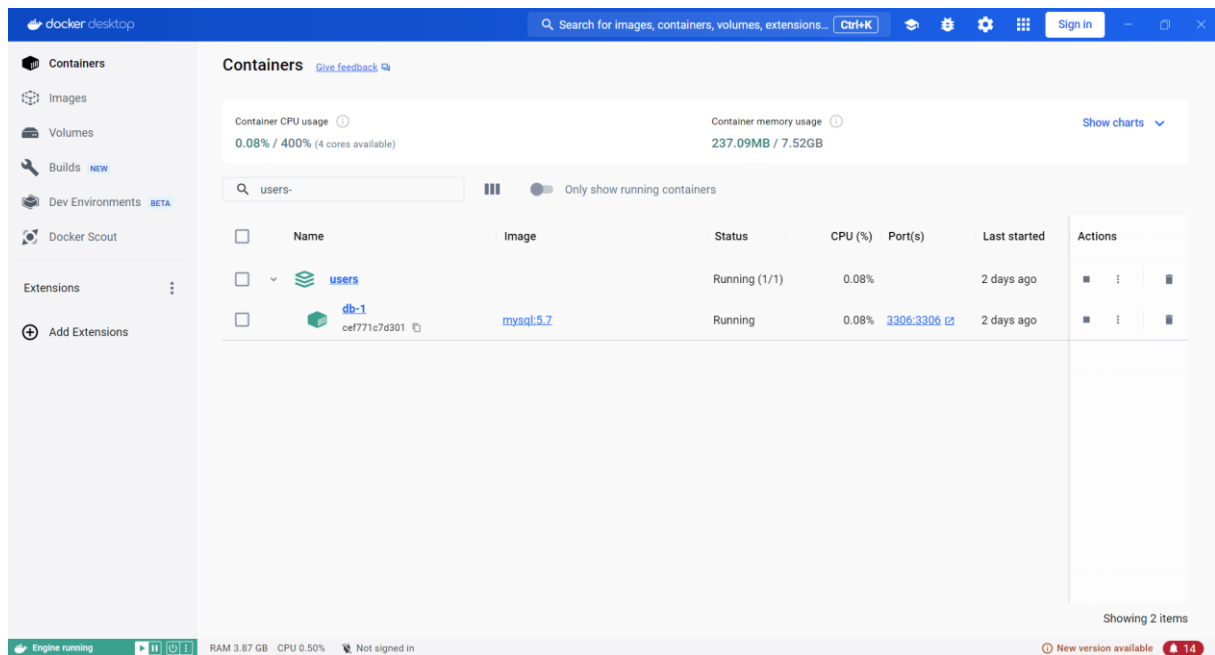
Pour créer le container docker sur lequel sera hébergé la base de données, il faut se rendre dans le dossier dédié :

```
cd server/databases/users
```

Ensuite il nous faudra exécuter la commande docker compose up :

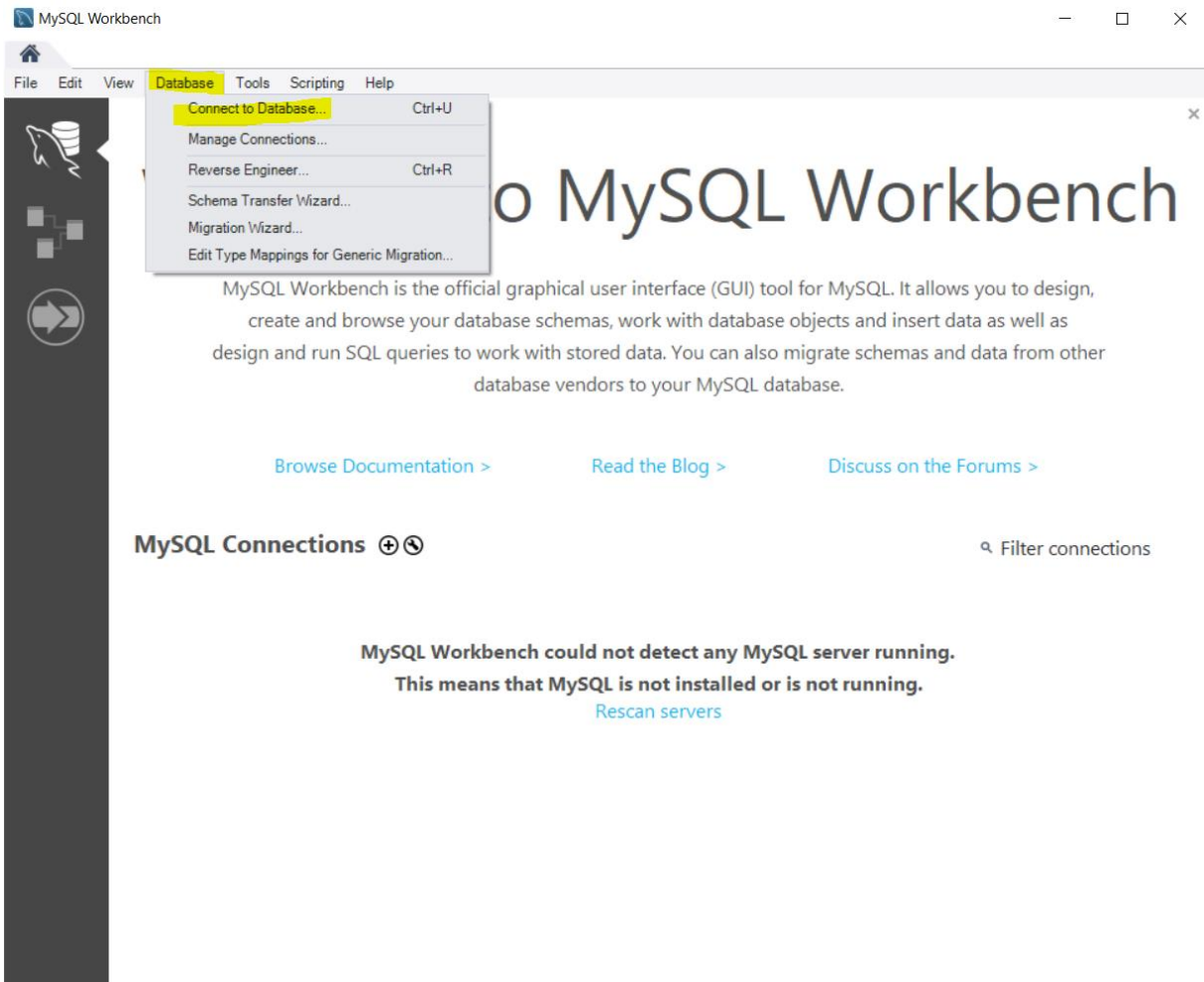
```
docker compose up -d
```

Une fois la commande exécuter, vous pouvez voir que le container à bien été créer sur docker desktop :



Création de la base de données :

A l'aide de MySQL Workbench (ou n'importe quel outil du genre), connectez-vous à la base de données :



Connect to Database

Stored Connection: Select from saved connection settings

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: 127.0.0.1 Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: root Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

Pensez bien à remplir le formulaire avec les informations correctes, dans notre cas il s'agit d'une installation locale alors nous mettrons l'adresse local 127.0.0.1.

Créer base de données « users » avec la requête suivante :

```
CREATE DATABASE users ;
```

```
USE DATABASE users ;
```

Ensuite, il ne vous restera plus qu'à exécuter la requête de création de table situé dans le même dossier que le docker-compose.

Vous avez installé le serveur de base de données avec succès.

SERVER API USER:

Afin d'établir une communication entre notre front-end et notre back-end, nous avons développé une API. Voici la procédure pour la mettre en marche.

Création du container :

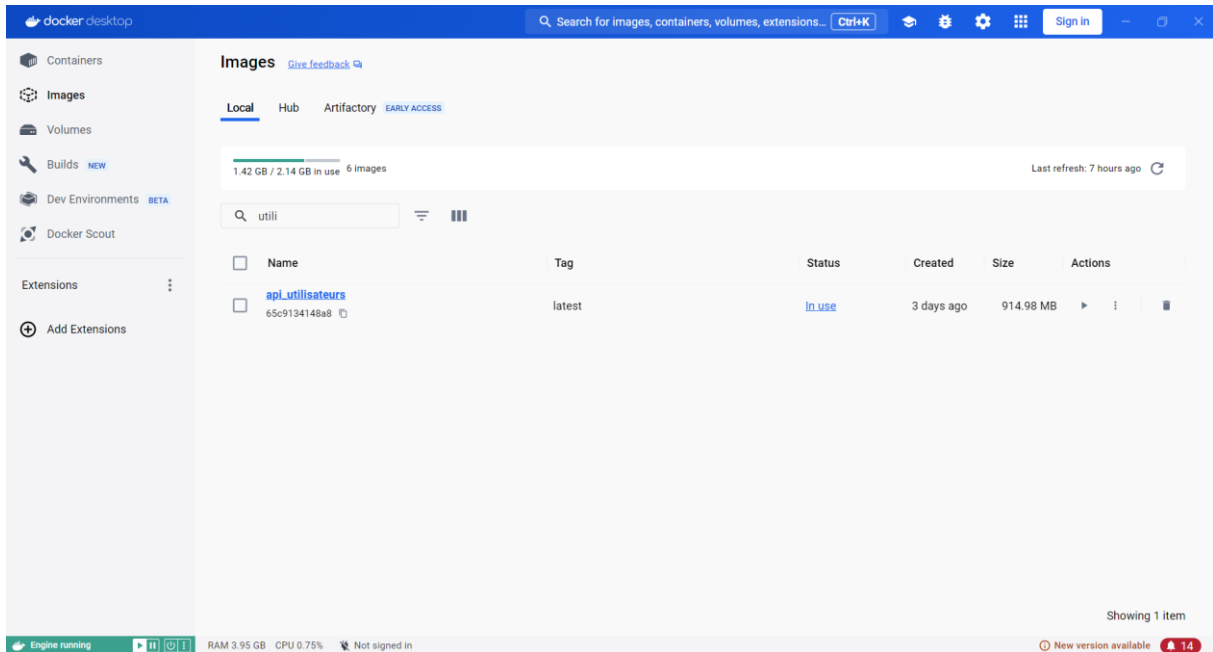
Pour créer le container docker sur lequel sera hébergé notre API, il faut se rendre dans le dossier dédié :

```
cd server/APIs/users
```

Ensuite il nous faudra exécuter la commande docker build :

```
docker build -t api_utilisateurs .
```

Une fois la commande exécuter, vous pouvez voir que l'image à bien été créer sur docker desktop :



Il nous faut maintenant créer un container à partir de cette image. Nous allons le nommer « api_utilisateurs » .



Run a new container

api_utilisateurs:latest

Optional settings



Container name

api_utilisateurs

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port

8080

:8080/tcp

Volumes

Host path



Container path



Environment variables

Variable

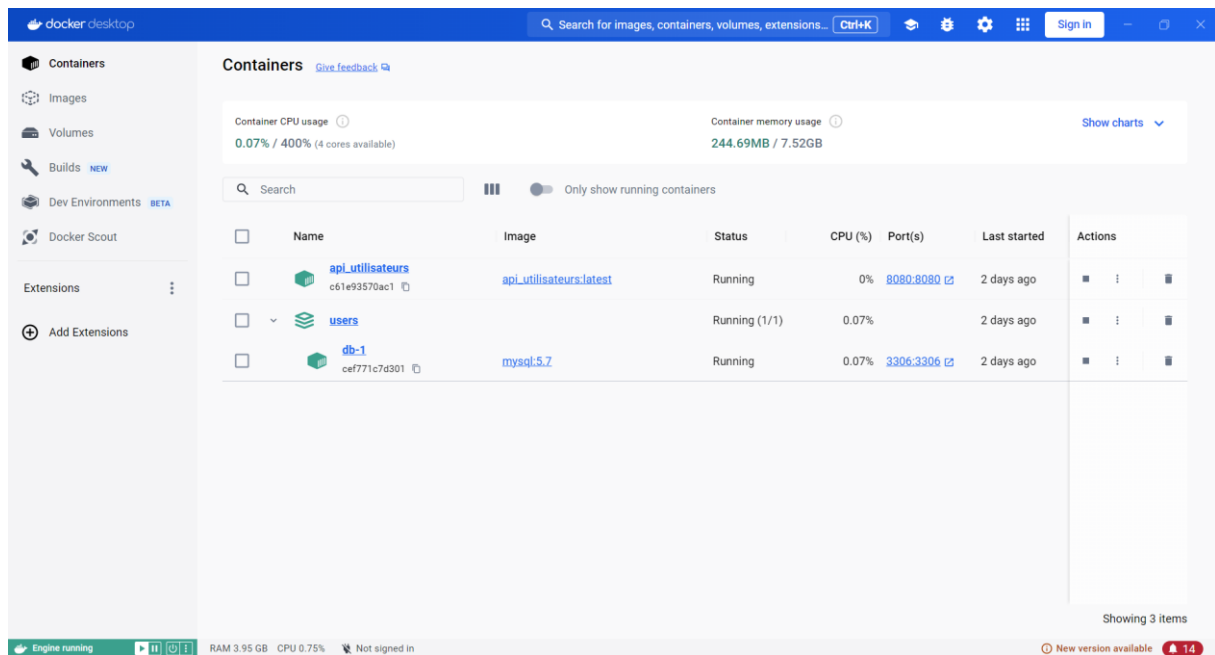
Value



Cancel

Run

Nous pouvons maintenant constater que le container a été créé avec succès.



Connexion du container de l'api à celui de la base de données :

Afin de faire communiquer 2 container docker, il faut que ces derniers soit sur le même réseau.

Création du réseau :

```
docker network create utilisateurs
```

Ajout de la base de données au réseau :

```
Docker network connect utilisateurs users-db-1
```

Ajout de l'API au réseau :

```
Docker network connect utilisateurs api_utilisateurs
```

Nos 2 containers sont à présents connectés et le client peut donc communiquer avec le serveur.

SERVER API POSTS:

Création du container :

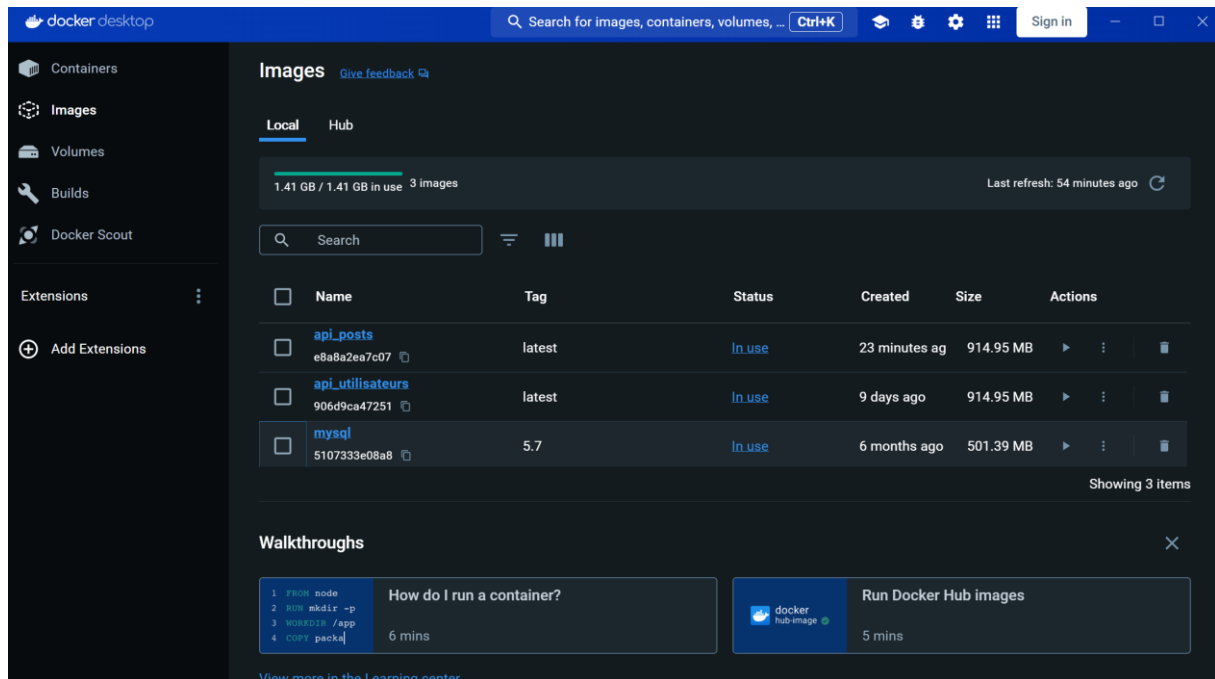
Pour créer le container docker sur lequel sera hébergé notre API, il faut se rendre dans le dossier dédié :

```
cd server/APIs/posts
```


Ensuite il nous faudra exécuter la commande docker build :

```
docker build -t api_posts .
```

Une fois la commande exécuter, vous pouvez voir que l'image à bien été créer sur docker desktop :



Il nous faut maintenant créer un container à partir de cette image. Nous allons le nommer « api_posts » .



Run a new container

api_posts:latest

Optional settings

Container name

api_posts

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port

:8081/tcp

Volumes

Host path

...

Container path

+

Environment variables

Variable

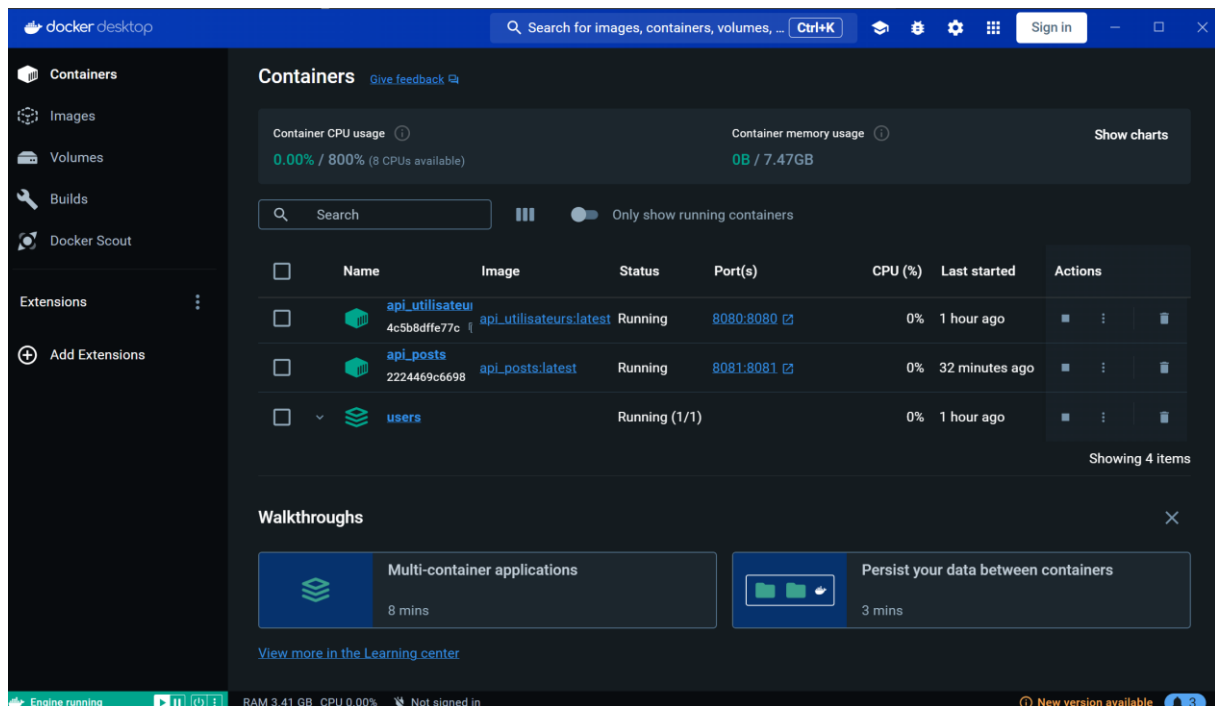
Value

+

Cancel

Run

Nous pouvons maintenant constater que le container a été créé avec succès.



Connexion du container de l'api à celui de la base de données :

Afin de faire communiquer 3 container docker, il faut que ces derniers soit sur le même réseau.

Ajout de l'API au réseau :

Docker network connect utilisateurs api_posts

Nos 3 containers sont à présents connectés et le client peut donc communiquer avec le serveur.