

Université de Yaoundé I
Faculté des Sciences
Département d'Informatique
Master I Pro
Matière : INF462



University of Yaounde I
Faculty of Sciences
Department Of Computer Science
Level : Master I

Travaux Pratiques N°1

Prise en main avec Docker

Groupe N°5

Nom et Prénoms des Etudiants :	Matricule
❖ MOUSSA ABAKAR ABBAZENE	23V2834
❖ ABDELAZIZ MAHAMAT LOUKY	18T2916
❖ SEMDI HONORE	14L1992
❖ MESSI DZOU SYLVESTRE CEDRIC	23V2853

Enseignant : Dr. KIMBI XAVERA
Supervisé par : M. ATEMENGUE REGIS

Année Académique : 2023 - 2024

Table des matières

Liste des figures	Erreur ! Signet non défini.
Introduction	3
I. Téléchargement et installation des outils	3
II. Description de TP	3
Conclusion	7

Introduction

Les travaux pratiques en Master sont une composante essentielle de la formation des étudiants, leur offrant une expérience pratique, des compétences techniques et professionnelles, et les préparant à leur future carrière. C'est ainsi que dans l'Unité INF462, un premier TP consiste à la prise en main avec la technologie docker et l'utilisation des conteneurs a été donné pour permettre un contact avec les conteneurs, les images, l'outils git et aussi l'aussi Github pour pusher le code dans une branche.

I. Téléchargement et installation des outils

Pour réaliser le premier TP, il est nécessaire de télécharger des outils comme le logiciel docker desktop et de l'installer. Ce logiciel est téléchargeable sur l'URL suivant : <https://www.docker.com/products/docker-desktop/> dont on choisira l'environnement (c'est-à-dire windows, Linux, MacOS,...). Une fois téléchargé, on créer un compte docker-hub sur le site suivant : <https://hub.docker.com> où on a l'ensemble des images disponibles pour le téléchargement. Dans notre cas, l'image la plus importante est Node puisque les microservices sont conçues à l'aide de nodejs.

Dans le logiciel Docker-desktop, on se connecte à l'aide du compte créé sur docker-hub synchronisant ainsi les deux applications.

L'outil git est aussi téléchargé et installé sur la machine permettant ainsi d'entrer les commandes pour le push de code.

Ajoutons à cela un éditeur de texte comme VS-code est aussi recommandé pour les codages.

II. Description de TP

Il s'agit de cinq micro-services conçus dont on veut les mettre dans les conteneurs et les exécuter automatiquement.

C'est ainsi que nous avons fait les étapes suivantes :

- **Créer un Dockerfile à la racine de chaque application ou microservice**
: Un Dockerfile est un fichier texte qui contient les instructions pour construire une image Docker.

```
# Utiliser une image Node.js officielle en tant que base
FROM node:latest
# Définir le répertoire de travail dans le conteneur
```

```
WORKDIR /app

# Copier les fichiers nécessaires dans le conteneur
COPY package*.json ./
COPY src ./src
COPY public ./public
# Installer les dépendances
RUN npm install
# Exposer le port nécessaire par l'application
EXPOSE 3000
# Commande pour démarrer l'application
CMD ["npm", "start"]
```

NB : L'expose des ports sera changé pour chaque micro-service comme indiqué sur le sujet du TP.

- **Créer un fichier .dockerignore** : Ce fichier spécifie les fichiers et répertoires à exclure lors de la construction de l'image Docker. Vous pouvez y inclure les fichiers qui ne sont pas nécessaires pour l'exécution de votre application, comme les fichiers de configuration locaux ou les fichiers de build.
- **Construire l'image Docker pour chaque application** : Pour chaque application, placez le Dockerfile et le fichier .dockerignore dans le répertoire racine de l'application. Ensuite, utilisez la commande docker build pour construire l'image Docker. Par exemple :

```
>docker build -t nom_image:version .
```

Cette commande permet de créer les images pour chaque micro-service dont ici, on change le nom de l'image par le micro-service et le point (.) indique le chemin où l'on trouve le fichier de configuration (le fameux Dockerfile).

- **Exécuter et tester localement les conteneurs Docker** : Vous pouvez exécuter les conteneurs Docker localement pour tester vos applications. Utilisez la commande docker run.

```
docker run -p 3000:3000 nom_image:version
```

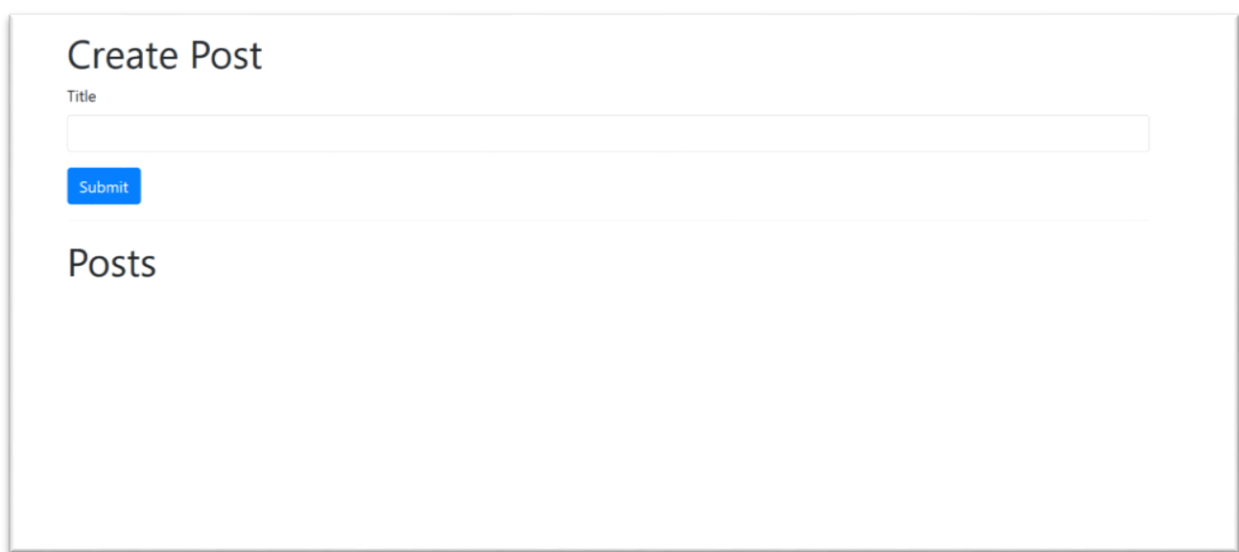
Une fois exécutée, l'image sera déjà installée dans le docker-desktop.

Déployer les conteneurs Docker : Une fois que vous avez testé localement et que tout fonctionne comme prévu, vous pouvez déployer les conteneurs Docker dans un environnement de production.

Une fois les images créées sur le docker desktop, on peut lancer les micro-services pour les voir sur le navigateur. Par exemple, le service client on l'avait lancé avec le port 3000 à l'aide de la commande suivante :

```
docker run -p 3000:3000 client
```

Ainsi ce formulaire apparait

The screenshot shows a web interface with a 'Create Post' section at the top. It contains a text input field labeled 'Title' and a blue 'Submit' button. Below this is a section titled 'Posts' which is currently empty. The interface is simple and clean, with a white background and light gray borders.

Les autres services n'ont pas des front-end alors sur le navigateur, il apparait vide sans erreur 404.

Pour automatiser le déploiement de plusieurs conteneurs Docker avec Docker Compose, nous avons fait les étapes :

Créer un fichier docker-compose.yml : Une fois docker-compose installer sur l'éditeur VScode, on crée un fichier docker-compose.yml à la racine de votre projet. Ce fichier contiendra la configuration pour chaque service que vous souhaitez déployer. Voici un exemple de fichier docker-compose.yml pour vos six applications :

```
version: '3'

services:
  client:
    build: ./client
```

```
ports:
  - "3000:3000"

comments:
  build: ./comments
  # Spécifiez les autres configurations nécessaires pour ce
service

event-bus:
  build: ./event-bus
  # Spécifiez les autres configurations nécessaires pour ce
service

moderation:
  build: ./moderation
  # Spécifiez les autres configurations nécessaires pour ce
service

posts:
  build: ./posts
  # Spécifiez les autres configurations nécessaires pour ce
service

query:
  build: ./query
  # Spécifiez les autres configurations nécessaires pour ce
service
```

Exécuter Docker Compose : Une fois que votre fichier docker-compose.yml est configuré, vous pouvez exécuter Docker Compose pour créer et démarrer tous les conteneurs en une seule commande.

```
docker-compose up -d
```

Vérifier l'état des conteneurs : Vous pouvez vérifier l'état des conteneurs en exécutant la commande suivante :

```
docker-compose ps
```

Arrêter et supprimer les conteneurs : Pour arrêter et supprimer les conteneurs, utilisez la commande suivante

```
docker-compose down
```

III. Push du code avec l'outil git

Une fois fini, on peut utiliser la commande suivante :

```
git init
```

Cette commande permet d'initialiser un dépôt git.

Ajouter les fichiers au suivi de Git : Utilisez la commande git add pour ajouter les fichiers que vous souhaitez inclure dans votre commit.

Associer votre dépôt local à un dépôt distant GitHub : Si vous n'avez pas encore associé votre dépôt local à un dépôt GitHub, vous devez le faire en ajoutant l'URL du dépôt distant GitHub. Remplacez URL_du_dépôt par l'URL de votre dépôt GitHub.

```
git remote add origin URL_du_dépôt
```

Conclusion

En fin, malgré les difficultés rencontrées, on a pu mettre la main sur docker et les outils qui vont avec. Alors, cette technologie est bien en terme de contenerisation des images surtout en utilisant les micro-services. Il faut signaler que ce TP nous a permis de familiariser avec les technologies citées ci-haut.