Documentation Docker pour Mountain Journey

Table des matières

Introduction à Docker

Docker est une plateforme open source qui permet de développer, expédier et exécuter des applications dans des conteneurs. Les conteneurs permettent de packager une application avec toutes ses dépendances afin qu'elle puisse s'exécuter de manière cohérente dans n'importe quel environnement. Docker simplifie le déploiement d'applications, l'isolation des processus et la gestion des environnements.

Cette documentation vous guidera à travers les étapes nécessaires pour configurer et déployer l'application Mountain Journey en utilisant Docker. L'application comprend une page d'accueil avec une carte interactive, des filtres de recherche et une liste d'itinéraires. Vous apprendrez à installer Docker, créer des images Docker, les pousser sur Docker Hub, configurer Docker Compose et déployer votre application.

Prérequis

Avant de commencer, assurez-vous d'avoir les éléments suivants installés sur votre machine :

- Docker : Installation de Docker
- Docker Compose Compose
- Un compte sur Docker Hub : Inscription à Docker Hub

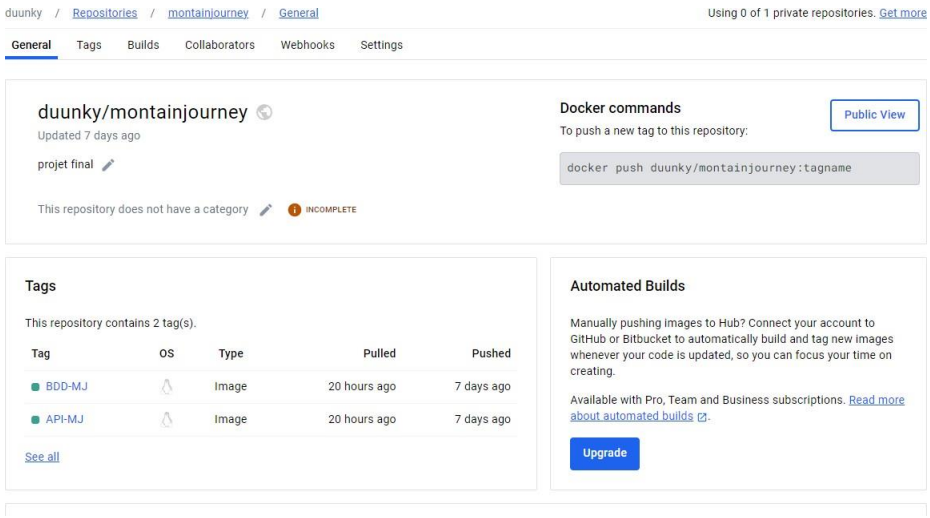
Utilisation de Docker Hub

Docker Hub est un registre public pour stocker et partager des images Docker. Vous pouvez y pousser vos images pour les utiliser sur différents environnements.

Créer un Compte Docker Hub (Facultatif)

1. Allez sur Docker Hub et cliquez sur "Sign Up".
2. Remplissez le formulaire d'inscription pour créer un compte.

Une fois le compte créé vous devrez créer un repo.



Connecter Docker à Docker Hub

1. Ouvrez une invite de commande ou un terminal.
2. Exécutez `docker login` et saisissez vos identifiants Docker Hub.

Créer et Pousser une Image Docker

1. Construisez l'image Docker avec `docker build -t yourusername/yourimagename` .
2. Poussez l'image sur Docker Hub avec `docker push yourusername/yourimagename`.

   Sur votre cmd :

```
495  docker tag nginx duunky/montainjourney:API-MJ
496  docker tag mysql duunky/montainjourney:BDD-MJ
497  docker push duunky/montainjourney:API-MJ
498  docker push duunky/montainjourney:BDD-MJ
```

   Ces commandes permettent de taguer vos images Docker locales et de les pousser vers Docker Hub sous des noms spécifiques.

Configuration des Dockerfiles

Un `Dockerfile` est un fichier texte qui contient toutes les instructions nécessaires pour construire une image Docker.

Dockerfile pour NGINX

Créez un fichier nommé `Dockerfile` avec le contenu suivant :

```
FROM duunky/montainjourney:API_MJ
WORKDIR /
RUN mkdir /API
COPY Front /usr/share/nginx/html/
COPY installationDotNet.sh /
COPY start.sh /
COPY API_MJ/ /API
RUN chmod +x /installationDotNet.sh
RUN chmod +x /start.sh
RUN ./installationDotNet.sh
COPY default.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["/start.sh"]
```

- **FROM** : Utilise une image de base spécifique comme point de départ.
- **WORKDIR** : Définit le répertoire de travail pour les opérations suivantes.
- **RUN mkdir /API** : Crée un répertoire pour organiser les fichiers de l'application.
- **COPY** : Copie les fichiers nécessaires de l'hôte vers le conteneur.
- **RUN chmod +x** : Rend les scripts exécutables.
- **RUN ./installationDotNet.sh** : Installe .NET en utilisant le script fourni.
- **COPY default.conf** : Configure nginx pour servir les fichiers.
- **EXPOSE 80** : Prépare le conteneur à écouter sur le port 80.
- **CMD ["/start.sh"]** : Lance le script « start.sh » lorsque le conteneur démarre.

Installation de DotNet :

```
#!/bin/bash

apt-get update && apt upgrade -y && apt-get install -y wget && apt-get install -y sudo
wget https://packages.microsoft.com/config/debian/12/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt-get update && sudo apt-get install -y dotnet-sdk-8.0
sudo apt-get update && sudo apt-get install -y aspnetcore-runtime-8.0
sudo apt-get install -y dotnet-runtime-8.0      You, il y a 3 jours • Mise en place de l'api dans le docker, changeme…
```

Le script bash met à jour les paquets, installe « wget » et « sudo » configure les dépôts Microsoft pour Debian, puis installe .NET SDK 8.0 et les runtimes ASP.NET Core et .NET 8.0.

Voici le script start.sh :

```
#!/bin/bash
# Démarre l'application .NET en arrière-plan
dotnet run --project /API/ &

# Démarre NGINX en avant-plan
nginx -g 'daemon off;'
```

Voici le default.conf pour la configuration NGINX :

```
server {
    listen       80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.html;
    }
    location /api {
        proxy_pass http://localhost:5000;  # Redirection de l'API sur le port 5000
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
      You, il y a 3 jours • Mise en place de l'api dans le docker, changeme…
    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
```

Nginx écoute les requêtes http sur le port 80.

Le serveur a le nom d'hôte : « localhost »

Les requêtes à la racine (/) serviront les fichiers depuis le répertoire /usr/share/nginx/html.

Les fichiers index.html et index.htm sont les fichiers index par défaut.

Les requêtes à /api sont redirigées vers http://localhost:5000, où votre API est hébergée.
Les en-têtes HTTP sont configurés pour transmettre des informations sur le client original :

Host: Le nom d'hôte de la requête originale.

X-Real-IP: L'adresse IP du client.

X-Forwarded-For: Une liste des adresses IP impliquées dans la requête HTTP.

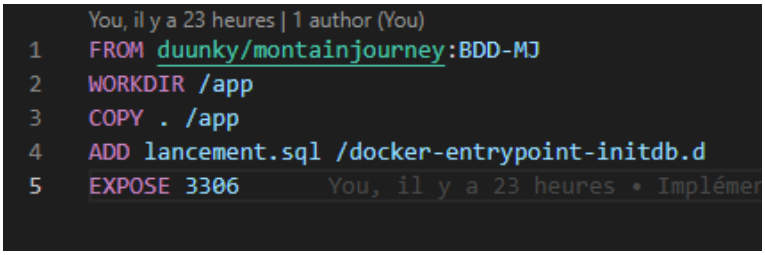
X-Forwarded-Proto: Le schéma (HTTP ou HTTPS) utilisé par le client original.

En cas d'erreur 404 (page non trouvée), Nginx servira le fichier /404.html.

En cas d'erreurs de serveur (500, 502, 503, 504), Nginx servira le fichier /50x.html depuis le répertoire /usr/share/nginx/html.

Dockerfile pour MYSQL

Créez un fichier nommé `Dockerfile` avec le contenu suivant :

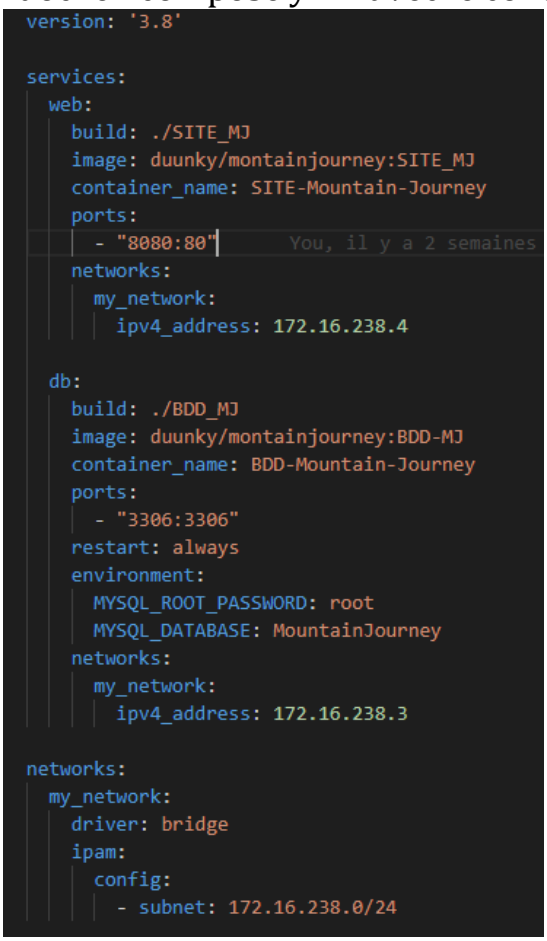


Configuration de Docker Compose

Docker Compose est un outil permettant de définir et de gérer des applications multi-conteneurs.

docker-compose.yml

Créez un fichier nommé `docker-compose.yml` avec le contenu suivant :



```yaml
version: '3.8'

services:
  web:
    build: ./SITE_MJ
    image: duunky/montainjourney:SITE_MJ
    container_name: SITE-Mountain-Journey
    ports:
      - "8080:80"      You, il y a 2 semaines
    networks:
      my_network:
        ipv4_address: 172.16.238.4

  db:
    build: ./BDD_MJ
    image: duunky/montainjourney:BDD-MJ
    container_name: BDD-Mountain-Journey
    ports:
      - "3306:3306"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: MountainJourney
    networks:
      my_network:
        ipv4_address: 172.16.238.3

networks:
  my_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.16.238.0/24
```

Ce fichier docker-compose.yml définit deux services Docker :

- **Un service web** qui utilise Nginx, construit à partir du répertoire ./SITE_MJ, accessible sur le port 8080 de l'hôte, et connecté au réseau my_network avec une IP spécifique.
- **Un service de base de données MySQL** construit à partir du répertoire ./BDD_MJ, accessible sur le port 3306 de l'hôte, avec des variables d'environnement pour configurer MySQL, et également connecté au réseau my_network avec une IP spécifique.

Ces services sont configurés pour être sur le même réseau my_network avec des adresses IP définies, ce qui facilite leur communication.

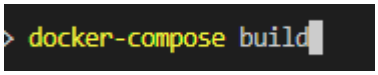
Déploiement de l'Application

Pour déployer l'application, suivez les étapes ci-dessous :

Construire l'image Docker

Exécutez la commande suivante pour construire l'image Docker :

docker-compose build



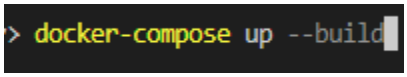
Démarrer les services Docker

Exécutez la commande suivante pour démarrer les services Docker :

docker-compose up

Ou faire les deux en même temps :

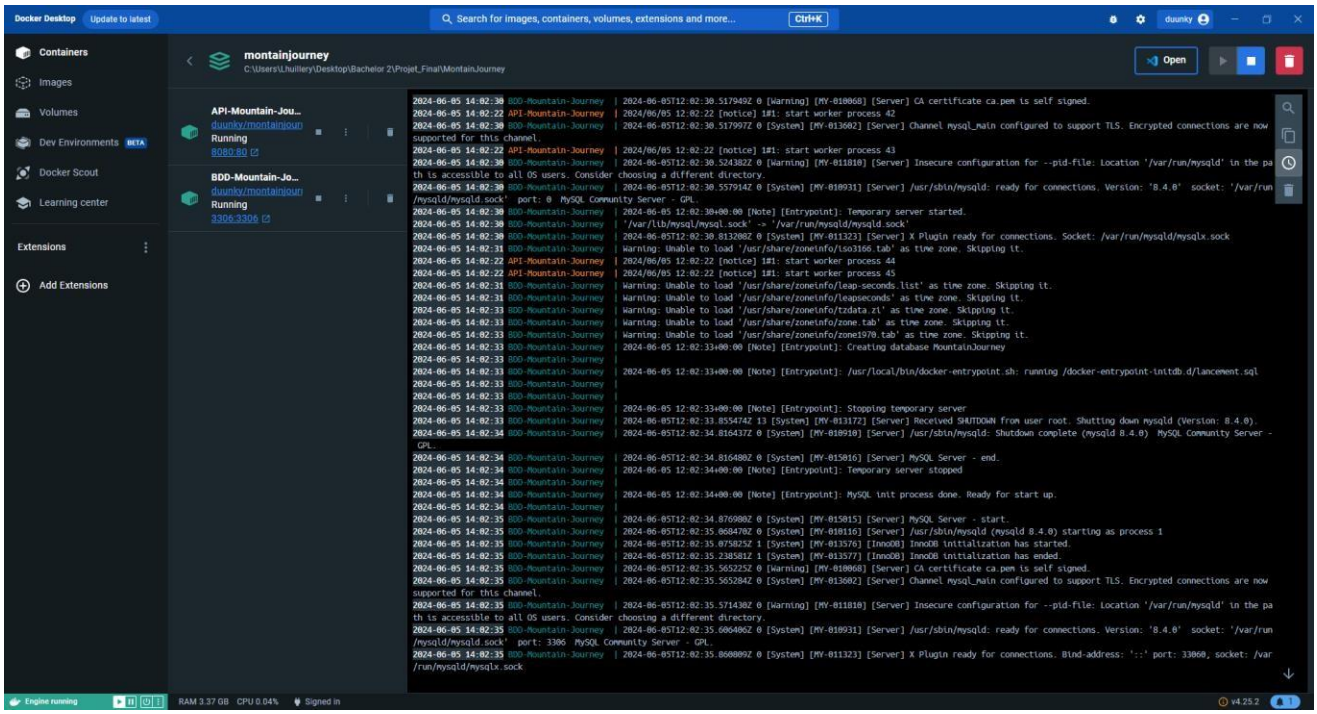
Docker-compose up --build

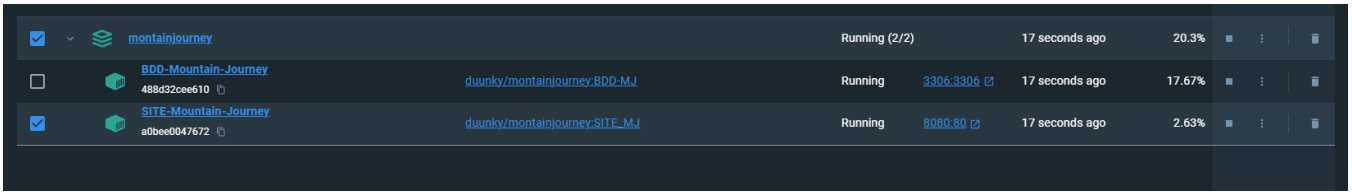


Accès à l'Application

Une fois les services Docker démarrés, accédez à l'application en ouvrant votre navigateur et en allant à l'adresse suivante : http://localhost:8080

Ou bien vous pouvez aller dans le logiciel et cliquer sur le port du conteneur en question .





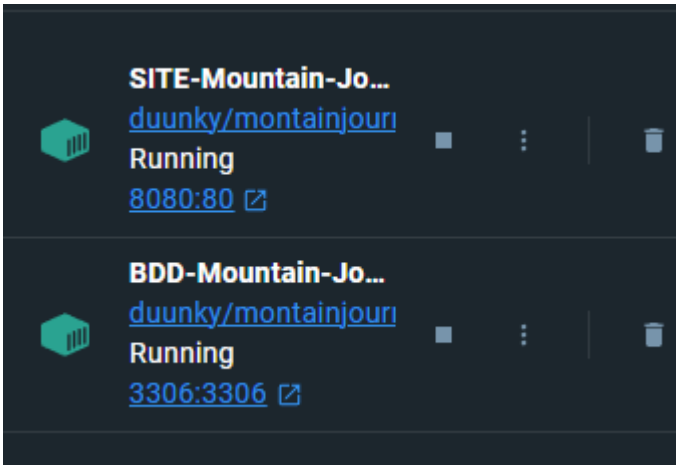
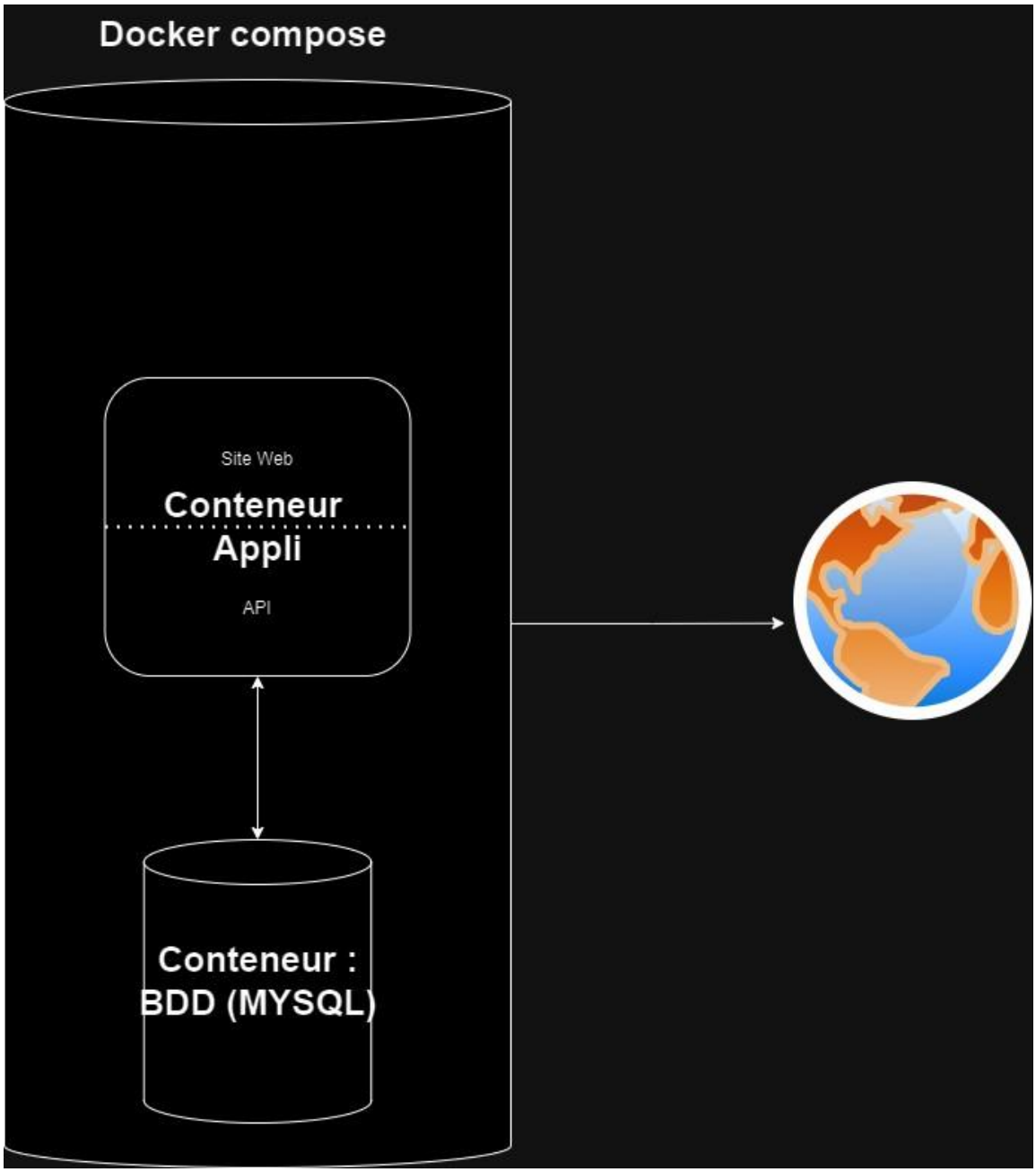


Schéma de l'Infrastructure



L'infrastructure Docker pour l'application Mountain Journey se compose d'un conteneur NGINX qui sert les fichiers de l'application et d'un conteneur Mysql servant de BDD.