DOCUMENTATION INFRASTRUCTURE

Projet Infra et Dev B2

Adriana Pullig

Yann Fournier

Table des matières

Qu’est-ce que Docker ?

Docker est une plateforme open source qui permet de créer, de déployer, d’exécuter, de mettre à jour et de gérer des conteneurs, des unités logicielles standard qui rassemblent le code source et toutes ses dépendances, afin que les applications puissent s’exécuter rapidement et de manière fiable d’un environnement à un autre. Les conteneurs ne sont pas persistants et sont lancés à partir d’images.

Comment fonctionne Docker ?

Docker utilise un système de conteneurisation qui sépare les processus pour qu’ils puissent s’exécuter de façon indépendante. Les conteneurs sont créés à partir d’images, qui sont des modèles en lecture seule qui définissent le code et les dépendances nécessaires pour exécuter une application. Les conteneurs peuvent être lancés, arrêtés, redémarrés et supprimés à l’aide de la commande Docker.

Avantages de Docker

- **Portabilité** : les applications Docker peuvent s’exécuter sur n’importe quel système d’exploitation qui supporte Docker, sans avoir à adapter le code ou les dépendances.
- **Sécurité** : les conteneurs isolent les applications et les dépendances, ce qui réduit les risques de vulnérabilités et d’attaques.
- **Gestion des dépendances** : Docker gère les dépendances pour les applications, ce qui facilite la mise à jour et la maintenance des applications.
- **Échelle** : Docker permet d’échelonner les applications facilement, en créant ou en supprimant des conteneurs en fonction des besoins.

Quels sont les différents éléments de Docker ?

(seul ceux que nous allons utiliser dans notre TP)

Docker Daemon

Le Docker Daemon traite les requêtes API afin de gérer les différents aspects de l'installation tels que les images, les conteneurs ou les volumes de stockage.

Docker Client

Le client Docker est la principale interface permettant de communiquer avec le système Docker. Il reçoit les commandes par le biais de l'interface de ligne de commande et les transmet au Docker Daemon.

Dockerfile

Chaque conteneur Docker débute avec un " Dockerfile ". Il s'agit d'un fichier texte rédigé dans une syntaxe compréhensible, comportant les instructions de création d'une image Docker.

Un Dockerfile précise le système d'exploitation sur lequel sera basé le conteneur, et les langages, variables environnementales, emplacements de fichiers, ports réseaux et autres composants requis.

Les conteneurs Docker

Un conteneur Docker ou Docker Container est une instance d'image Docker exécutée sur un micro-service individuel ou un stack d'application complet. En lançant un conteneur, on ajoute une couche inscriptible sur l'image. Ceci permet de stocker tous les changements apportés au conteneur durant le runtime.

Les images Docker

Une image Docker est un modèle en lecture seule, utiliser pour créer des conteneurs Docker. Elle est composée de plusieurs couches empaquetant toutes les installations, dépendances, bibliothèques, processus et codes d'application nécessaires pour un environnement de conteneur pleinement opérationnel.

Après avoir écrit le Dockerfile, on invoque l'utilitaire " build " pour créer une image basée sur ce fichier. Cette image se présente comme un fichier portable indiquant quels composants logiciels le conteneur exécutera et de quelle façon.

Qu'est-ce que Docker Compose ?

Il s'agit d'un outil de ligne de commande, similaire au client Docker, utilisant un fichier de description spécifiquement formaté pour assembler les applications à partir de conteneurs multiples et de les exécuter sur un hôte unique.

Lorsqu'une application est prête à être déployée sur Docker, il est nécessaire de pouvoir approvisionner, configurer, étendre et surveiller les conteneurs sur l'architecture de micro-service.

Pour y parvenir, on utilise des systèmes d'orchestration de conteneurs open source tels que Kubernetes, Mesos et Docker Swarm. Ces systèmes fournissent les outils nécessaires pour gérer les clusters de conteneurs.

Ces solutions permettent notamment de répartir les ressources entre les conteneurs, d'ajouter ou de supprimer des conteneurs, de gérer les interactions entre les conteneurs, de surveiller leur statut, ou d'équilibrer la charge entre les micro-services.

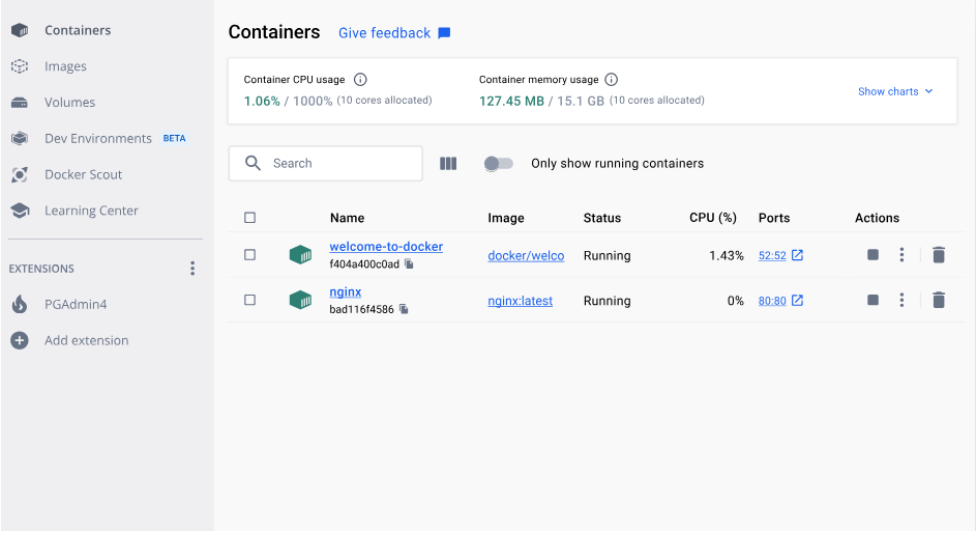
Qu’est-ce que Docker Desktop ?

Docker Desktop est l’application PC native conçue par Docker pour Windows et Mac. C’est la façon la plus simple d’exécuter, de construire, de déboguer et de tester des applications Dockerisées.

Ce logiciel rassemble les fonctionnalités principales comme les cycles de test rapides, les notifications de changement de fichier, une prise en charge du réseau d’entreprise, ou encore une flexibilité complète pour le choix de proxies et de VPN.

L’application Docker Desktop regroupe des outils développeurs, Docker App, Kubernetes et la synchronisation de version. Elle permet de créer des images et des templates en choisissant les langages et outils.

Les principaux avantages sont la vitesse, la sécurité et la flexibilité. On distingue l’édition Community gratuite, et l’édition Enterprise payante ajoutant des fonctionnalités supplémentaires pour la sécurité, la gestion, l’orchestration et la gestion.



Qu’est-ce que Nginx ?

Nginx, prononcé comme « engine-ex », est un serveur web open-source qui, depuis son succès initial en tant que serveur web, est maintenant aussi utilisé comme reverse proxy, cache HTTP, et load balancer.

Comment fonctionne Nginx ?

Nginx est conçu pour offrir une faible utilisation de la mémoire et une grande simultanéité. Plutôt que de créer de nouveaux processus pour chaque requête Web, Nginx utilise une approche asynchrone et événementielle où les requêtes sont traitées dans un seul thread.

Avec Nginx, un processus maître peut contrôler plusieurs processus de travailleurs. Le maître gère les processus du travailleur, tandis que les travailleurs effectuent le traitement proprement dit. Comme Nginx est asynchrone, chaque requête peut être exécutée simultanément par le travailleur sans bloquer les autres requêtes.

Installation de Docker

1. Installation interactive

1. Téléchargez le programme d'installation sur le site de docker.docs ou en cliquant sur ce lien : https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Installer.exe?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-win-amd64
2. Double-cliquez ***Docker Desktop Installer.exe*** pour exécuter le programme d'installation. Par défaut, Docker Desktop est installé sur ***C:\Program Files\Docker\Docker.***
3. Lorsque vous y êtes invité, assurez-vous que l'option ***Utiliser WSL 2 au lieu d'Hyper-V*** sur la page de configuration est sélectionnée ou non en fonction de votre choix de backend.
4. Si votre système ne prend en charge qu'une des deux options, vous ne pourrez pas sélectionner le backend à utiliser.
5. Suivez les instructions de l'assistant d'installation pour autoriser le programme d'installation et poursuivre l'installation.
6. Une fois l'installation réussie, sélectionnez Fermer pour terminer le processus d'installation.

Si votre compte administrateur est différent de votre compte utilisateur, vous devez ajouter l'utilisateur au groupe docker-users :

1. Exécutez la gestion de l'ordinateur en tant qu'administrateur.
2. Accédez à Utilisateurs et groupes locaux > Groupes > docker-users.
3. Cliquez avec le bouton droit pour ajouter l'utilisateur au groupe.
4. Déconnectez-vous et reconnectez-vous pour que les modifications prennent effet.

2. Installation par ligne de commande

Après le téléchargement Docker Desktop Installer.exe, exécutez la commande suivante dans un terminal pour installer Docker Desktop :

"Docker Desktop Installer.exe" install

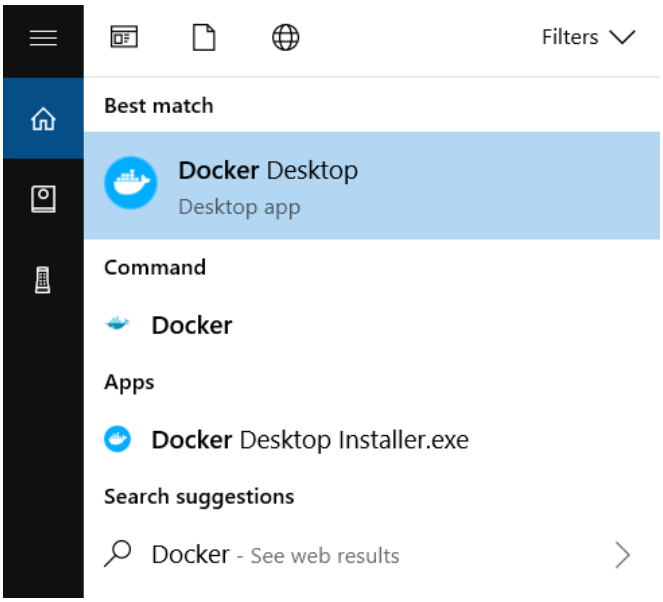
Si vous utilisez PowerShell, vous devez l'exécuter comme :

Start-Process 'Docker Desktop Installer.exe' -Wait install

Si vous utilisez l'invite de commande Windows :

start /w "" "Docker Desktop Installer.exe" install

Par défaut, Docker Desktop est installé sur ***C:\Program Files\Docker\Docker***.



3. Installation de Docker Compose

Suivez ces instructions si vous exécutez le daemon et le client Docker directement sur Microsoft Windows Server et que vous souhaitez installer Docker Compose.

1. Exécutez PowerShell en tant qu'administrateur. Lorsqu'on vous demande si vous souhaitez autoriser cette application à apporter des modifications à votre appareil, sélectionnez **Oui** afin de procéder à l'installation.
2. GitHub nécessite désormais TLS1.2. Dans PowerShell, exécutez ce qui suit :

[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

3. Exécutez la commande suivante pour télécharger la dernière version de Compose (v2.27.1) :

Start-BitsTransfer -Source "https://github.com/dock

4 .Testez l'installation.

$ docker-compose.exe version

Résultat : Docker Compose version v2.27.1

4. Démarrer Docker Desktop (Docker Bureau)

1. Recherchez Docker dans votre barre de recherche et sélectionnez Docker Desktop dans les résultats.
2. Le menu Docker affiche le contrat de service d'abonnement Docker.

3. Sélectionnez **Accepter** pour continuer. Docker Desktop démarre une fois que vous avez accepté les conditions.

Notez que Docker Desktop ne fonctionnera pas si vous n'acceptez pas les conditions. Vous pouvez choisir d'accepter les conditions ultérieurement en ouvrant Docker Desktop.

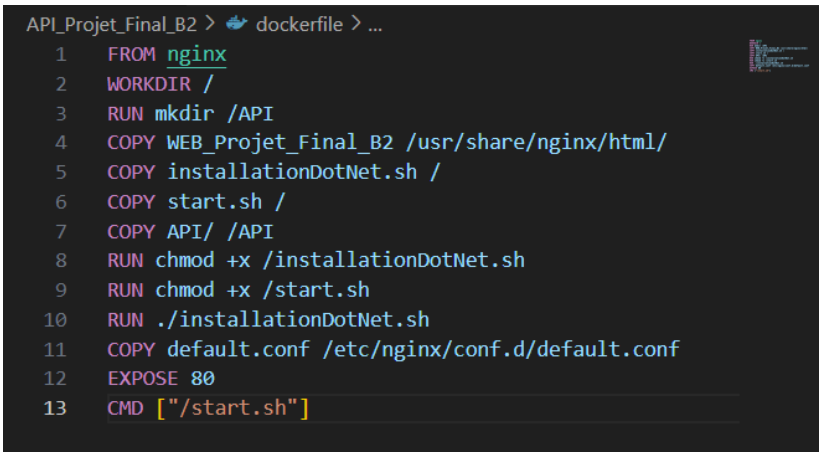
Configuration

1. Dockerfile

a. Dockerfile Nginx

Nous allons ici créer un Dockerfile pour Nginx

Créez un fichier nommé 'dockerfile' dans votre éditeur de code et à l'intérieur ajoutez-y le contenu suivant :



- **FROM** :
- **WORKDIR** : Définit le répertoire de travail pour les opérations suivantes, ici la racine.
- **RUN mkdir /API** : Crée un répertoire pour organiser les fichiers de l'application, ici l'API.
- **COPY** : Copie les fichiers nécessaires de l'hôte vers le conteneur, ici le front, dotnet, start et le dossier API dans le nouveau répertoire API.
- **RUN chmod +x** : Rend les scripts exécutables pour la plateforme.
- **RUN ./installationDotNet.sh** : Installation de .Net en utilisant le script que l'on a rempli.
- **COPY deafault**.conf : Configure nginx pour répondre aux besoins des fichiers.
- **EXPOSE 80** : Prépare le conteneur à écouter sur le port 80.
- **CMD [« /start.sh »]** : Lance le script lorsque le conteneur démarre

b. Dockerfile MYSQL

Nous allons ici créer un Dockerfile MYSQL.

Créez un fichier nommé ‘dockerfile’ dans votre éditeur de code et à l’intérieur ajoutez-y le contenu suivant :

BDD > BDD_Projet_Final_B2 > dockerfile > FROM

```
FROM mysql
WORKDIR /app
COPY . /app
ADD script.sql /docker-entrypoint-initdb.d
EXPOSE 3306
```

- **FROM** : Utilise une image de base qui est spécifique comme départ.
- **WORKDIR** : Définit le répertoire de travail pour les opérations suivantes, ici le répertoire app.
- **COPY** : Copie les fichiers nécessaires de l’hôte vers le conteneur.
- **ADD** : récupère le script.sql (notre db) et l’ajoute dans le dossier /docker-entrypoint-initdb.d.
- **EXPOSE 3306** : Prépare le conteneur à écouter sur le port 3306.

2. Installation DotNet

Le script Bash permet de mettre à jour les paquets, d’installer ***wget*** et de configurer grâce à ***sudo*** les dépôts Microsoft pour Debian, puis installe .NET SDK 8.0. Pour finir il installe les runtimes ASP.NET Core et .NET 8.0.

API_Projet_Final_B2 > $ installationDotNet.sh

```
#!/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
```

3. Script start.sh

Ce script démarre l’application .NET en arrière-plan et démarre Nginx au premier plan.

API_Projet_Final_B2 > $ start.sh

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

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

4. Fichier default.conf

Dans le server, appelé **localhost** ici, on autorise Nginx écouter les requêtes http sur le port 80.

Les requêtes à la racine permettent aux fichiers depuis le répertoire /sur/share/nginx/html de servir de liaison.

Ici le fichier index.html de nginx est remplacé par mon propre fichier index qui se trouve dans le dossier html/index.html.

Les requêtes à /api sont redirigées vers http://localhost:5000, où notre API est hébergée.

Les header http sont configurés pour transmettre des informations sur le client original :

- **Host** : Le nom d'hôte de la requête.
- **X-Real-IP** : L'addresse IP du client.
- **X-Forwarded-For** : Liste des adresses IP impliquées dans la requête HTTP
- **X-Forwarded-Proto** : Le schéma utilisé par le client.

API_Projet_Final_B2 > default.conf

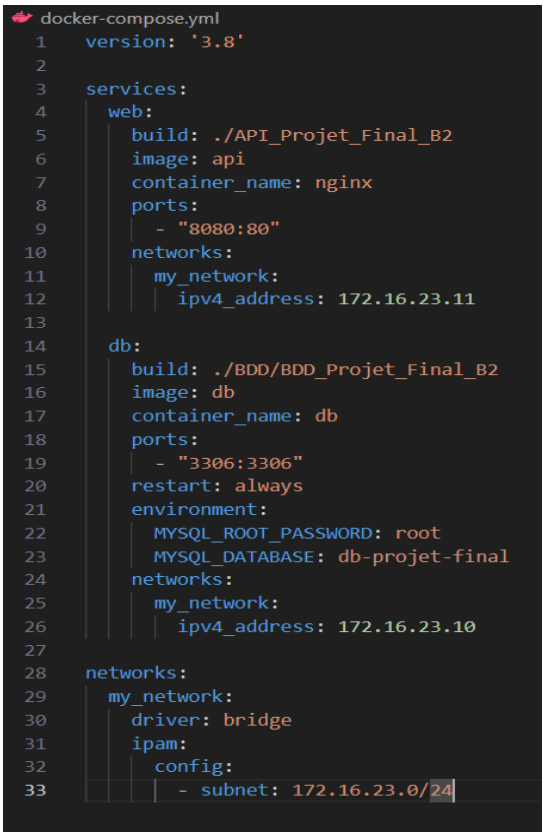
```
server {
    listen       80;
    server_name  localhost;

    location / {
        root   /usr/share/nginx/html;
        index  html/index.html index.html;
    }
    location /api {
        proxy_pass http://localhost: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;
    }
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

}
```

En cas d'erreurs de serveur, Nginx sert au back up du fichier /50x.html depuis le répertoire /usr/share/nginx/html.

5. Configuration du Docker Compose

Fichier docker-compose.yml

Créez un fichier nommé 'docker-compose.yml' :



Ce fichier définit deux services Docker :

1. Un service WEB qui utilise Nginx, qui est construit à partir du répertoire ./API_Projet_Final_B2 , accessible sur le port 80 et connecté au réseau my_network avec une IP spécifique, ici 172.16.23.11 .

2. Un service de base de données MySQL construit à partir du répertoire ./BDD/BDD_Projet_Final_B2 , accessible sur le port 3306 et connecté au réseau my_network avec un IP spécifique, ici 172.16.23.10 .

Ces services sont configurés pour être sur le même réseau my_network avec des adresses IP définies sur 172.16.23.0/24 .

Déployer une Application

Il y a deux manières de déployer une application avec docker-compose, soit en deux étapes soit en une.

Construction de l'image docker

```
C:\Users\patch\Desktop\B2\Projet_B2>docker-compose --build
```

Démarrage des services

```
C:\Users\patch\Desktop\B2\Projet_B2>docker-compose up
```

Construction et démarrage

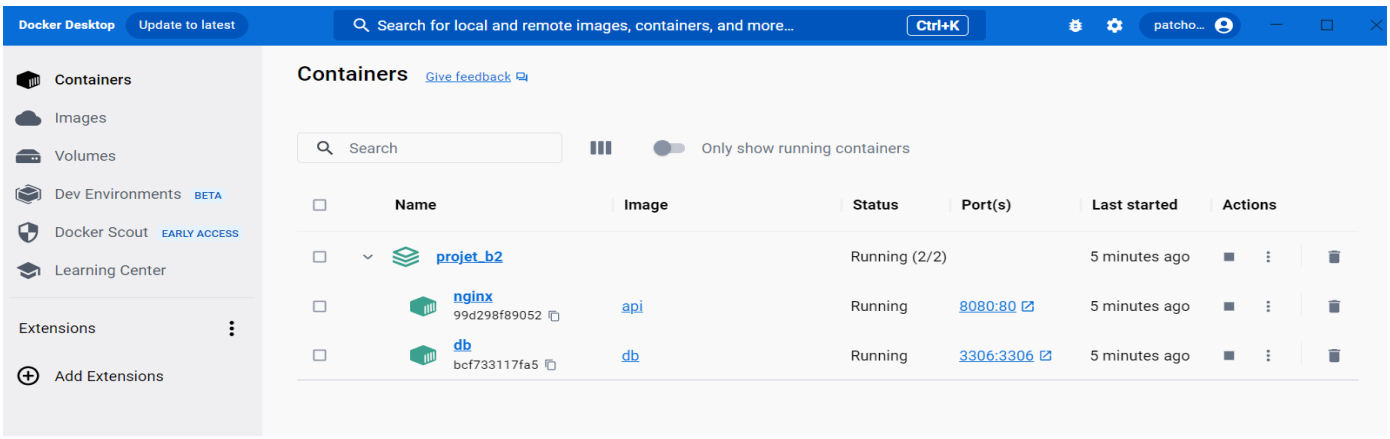
```
C:\Users\patch\Desktop\B2\Projet_B2>docker-compose up --build
```

Résultat

Si vous avez tout bien suivi, vous devriez avoir ce résultat :

```
C:\Users\patch\Desktop\B2\Projet_B2>docker-compose up --build
[+] Building 0.0s (0/0)
[+] Building 0.2s (0/2)
 => [internal] load .dockerignore                                                                0.2s
 => => transferring context: 2B                                                                  0.0s
 => [internal] load build definition from dockerfile                                             0.2s
[+] Building 0.3s (2/3)
[+] Building 1.2s (3/4)
[+] Building 2.1s (10/10) FINISHED
 => [internal] load .dockerignore                                                                0.2s
 => => transferring context: 2B                                                                  0.0s
 => [internal] load build definition from dockerfile                                             0.2s
 => => transferring dockerfile: 611B                                                             0.0s
 => [internal] load metadata for docker.io/library/mysql:latest                                  1.6s
 => [auth] library/mysql:pull token for registry-1.docker.io                                     0.0s
 => [1/4] FROM docker.io/library/mysql@sha256:aa021e164da6aacbefc59ed0b933427e4835636be380f3b6523f4a6c9  0.0s
 => [internal] load build context                                                                0.2s
 => => transferring context: 13.59kB                                                             0.1s
[+] Building 20.3s (13/15)
 => [internal] load .dockerignore                                                                0.2s
 => => transferring context: 2B                                                                  0.0s
 => [internal] load build definition from dockerfile                                             0.2s
 => => transferring dockerfile: 358B                                                             0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                  2.0s
 => [auth] library/nginx:pull token for registry-1.docker.io                                     0.0s
 => [internal] load build context                                                                0.2s
 => => transferring context: 11.54kB                                                             0.1s
 => [ 1/11] FROM docker.io/library/nginx@sha256:0f04e4f646a3f14bf31d8bc8d885b6c951fdcf42589d06845f64d18aec6a3c4  0.0s
 => CACHED [ 2/11] RUN mkdir /API                                                                0.0s
 => [ 3/11] COPY WEB_Projet_Final_B2 /usr/share/nginx/html/                                      0.3s
 => [ 4/11] COPY installationDotNet.sh /                                                         0.2s
 => [ 5/11] COPY start.sh /                                                                      0.2s
 => [ 6/11] COPY API/ /API                                                                       2.2s
 => [ 7/11] RUN chmod +x /installationDotNet.sh                                                  0.8s
 => [ 8/11] RUN chmod +x /start.sh                                                               0.8s
 => [ 9/11] RUN ./installationDotNet.sh                                                         13.3s
```

Accéder à l'application



Annexe

https://docs.docker.com/desktop/install/windows-install/

https://datascientest.com/docker-guide-complet

https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-nginx/