



Installation de Laravel & Docker

ROTHEUDT Thomas et MOUCHAMPS Antoine - 22 août 2023



N-HiTec

Allée de la Découverte 10, 4000 Liège, Belgium



nhitec.com | info@nhitec.com




Table des matières



I. Introduction


1 Qu'est-ce que Docker ?

Docker  est un outil open-source créé en 2012 par des français. Cet outil permet de créer, de déployer et de lancer des applications tournant dans un conteneur. Ces conteneurs sont en réalité des environnements dans lesquels les applications tournent. Ils sont créés grâce à des images qui sont des fichiers Docker , ainsi n'importe quel conteneur est assuré de tourner sur n'importe quel machine sans se soucier des configurations locales.

L'outil est tellement populaire que les mainteneurs de bibliothèques ou de logiciels maintiennent des images Docker  (dans notre cas on verra que Laravel Sail  possède une image Docker  qui est maintenue à jour régulièrement) Il permet donc de "centraliser" l'environnement sur lequel une application est développée.



2 Pourquoi Docker ?

Un conteneur Docker  possède de nombreux avantages comparé aux machines virtuelles. Premièrement, il utilise les ressources d'un système plus efficacement qu'une VM tout en gardant ses avantages (isolation et reproductibilité).



Il garantit d'être identique quel que soit le système, ainsi chaque membre d'un projet est certain de travailler sur le même environnement sans se soucier de la portabilité de l'application sur l'environnement final.

Les conteneurs sont isolés de la machine hôte, ainsi vous pouvez avoir plusieurs versions différentes de dépendances pour plusieurs projets.

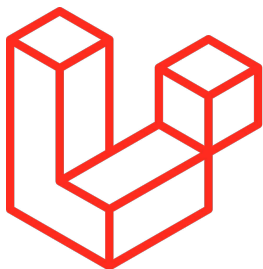
Enfin, la modification d'un conteneur est extrêmement simplifiée comparé à une VM où les mises à jour sont souvent complexes et laborieuses.

Pour résumer, un conteneur Docker  est flexible, léger, portable.



3 Qu'est-ce que Laravel Sail 🐳 ?

Laravel Sail 🐳 est une interface de ligne de commande légère pour interagir avec l'environnement de développement Docker 🐳 par défaut de Laravel 🦋.



Sail est un excellent point de départ pour construire une application Laravel 🦋 en utilisant PHP ^{php}, MySQL et Redis sans avoir besoin d'une expérience préalable de Docker 🐳.

Au cœur de Sail se trouve le fichier docker-compose.yml et le script sail qui est stocké à la racine de votre projet. Le script sail fournit une CLI (command line interface) avec des méthodes pratiques pour interagir avec les conteneurs Docker 🐳 définis par le fichier docker-compose.yml.

Laravel Sail 🐳 est pris en charge sur MacOS 🍏, Linux 🐧 et Windows 🪟 (via WSL2).

Pour résumer, le scrip sail permet de gérer le projet Laravel 🦋 à l'intérieur du conteneur du projet. De plus Laravel Sail 🐳 est complètement compatible avec Docker 🐳, ce qui permet des interactions plus simples avec les conteneurs.

II. Installation de Docker 🐳

Choisissez la section correspondant à votre système d'exploitation :



Linux : SECTION ??



Windows : SECTION ??



MacOS : SECTION ??

1 Installation Linux 🐧

L'Installation Linux 🐧 nécessite de suivre quelques étapes de plus que pour l'Installation MacOS 🍏 ou Windows 🪟.

1. Mettez à jour l'index des paquets apt et des paquets d'installation pour permettre à apt d'utiliser un dépôt sur HTTPS :

```
sudo apt-get update
```

2. Installez les dépendances de Docker 🐳 :

```
sudo apt-get install \
ca-certificates \
```


```
curl \
gnupg \
lsb-release
```

3. Ajoutez la clé GPG officielle de Docker :

```
sudo mkdir -m 0755 -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

4. Configurez le répertoire :

`$(lsb_release -cs)` doit dans quelques cas être remplacé par votre version de distribution Linux .
Dans l'installation de référence on a remplacé par *jammy*.

```
echo \
"deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable"
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Vérifiez que le répertoire est bien configuré :

```
sudo apt-get update
```

6. Installez Docker Engine, conteneur, et Docker Compose :

```
sudo apt-get install docker-ce docker-ce-cli
conteneurd.io docker-buildx-plugin docker-compose-plugin
```

Vous pouvez tester votre installation avec la commande : `sudo docker run hello-world`

7. Installez Docker Desktop :

Téléchargez le **Docker Destop** et l'installez avec la commande :





```
sudo apt-get install ./docker-desktop-<version>-<arch>.deb
```




L'Installation de référence pour ce guide a été faite sur une distribution Linux  Mint (distribution basé sur Ubuntu )

Pour passer à la suite

2 Installation Windows

2.1 Prérequis : WSL2

Pour utiliser Docker Desktop , il est fortement conseillé d'utiliser **WSL2**. C'est une application Windows  permettant d'installer un environnement Linux  avec la distribution de votre choix nativement sous Windows .


Pour l'installer, il suffit de lancer `wsl --install` dans le terminal de Windows  de base ou Powershell¹. Par défaut, cette commande installera Ubuntu  comme système Linux . Pour d'avantage d'informations concernant l'installation de WSL2, consultez le lien vers la page officielle de Microsoft. Si tout se passe bien vous devriez recevoir le message suivant :

```
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
Ubuntu has been installed.
The requested operation is successful. Changes will not be effective until the system is rebooted.
```

Après avoir redémarrer votre PC, la fenêtre suivante devrait apparaître. Si ce n'est pas le cas, chercher `wsl.exe` dans la barre de recherche windows et lancez le terminal.

```
Ubuntu is already installed.
Launching Ubuntu...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```


Ici, vous devez créer un utilisateur ainsi qu'ajouter un mot de passe². Attention, vous devrez utiliser votre mot de passe de temps en temps donc ne l'oubliez pas !

Une fois ces étapes effectuées sans accros, vous pouvez passer à l'installation de Docker Desktop  à proprement parlé.

2.2 Installation de Docker Desktop

1. Téléchargez **Docker Desktop** 
2. Exécutez le fichier d'installation `.exe`
3. Vérifiez l'installation en lançant votre environnement `wsl` et en tapant la commande suivante :

```
docker
```

Docker  et `wsl2` sont totalement compatibles. C'est pourquoi vous devez cocher l'option "use WSL2" pendant l'installation.

Pour passer à la suite

-
1. Attention, le terminal choisi doit être exécuté en tant qu'administrateur !
 2. Lorsque vous tapez votre mot de passe, il est normal que les caractères ne s'affichent pas.

3 Installation MacOS 🍏

1. Téléchargez Docker Desktop 🐳 **Apple Chip** ou **Intel Chip**
2. Installez Docker Desktop 🐳 dans le dossier application



Pour passer à la suite

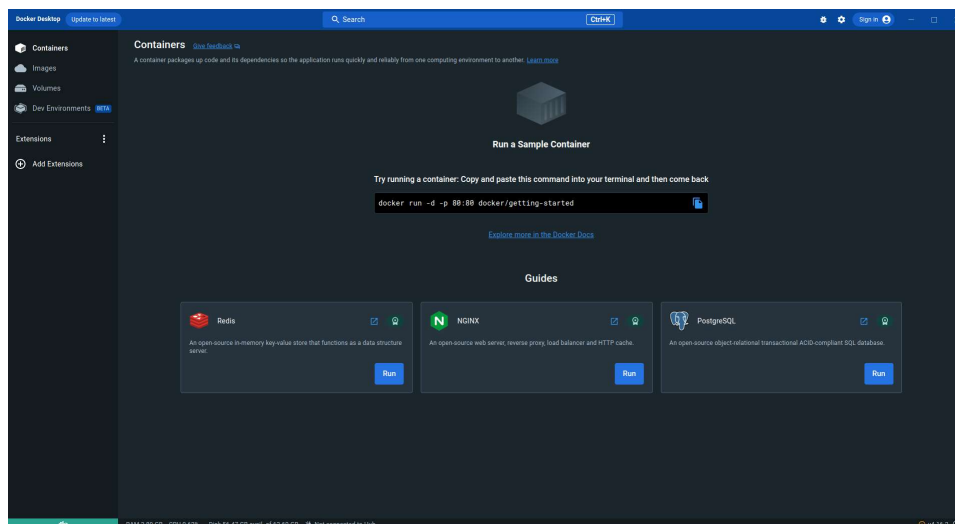


N-HiTec


Allée de la Découverte 10, 4000 Liège, Belgium
nhitec.com | info@nhitec.com

4 Aperçu



Si tout s'est bien passé vous êtes en mesure de lancer Docker Desktop . Vous avez donc le panneau de contrôle de Docker Desktop  qui devrait ressembler à ça :




Vous pouvez voir différents onglets sur la gauche :

- Conteneurs : Cet onglet va contenir tous les conteneurs que vous avez créés. Les conteneurs sont en quelque sorte des images déployées et exécutées lorsque Docker  tourne.
- Images : Cet onglet va contenir toutes les images d'installation des services installés dans nos conteneurs. Les images décrivent l'environnement du serveur, les packages installés et toutes les configurations de celui-ci.
- Volumes : Cet onglet contient les volumes liés aux services qui contiennent les données de nos applications. Les volumes servent à conserver les données des bases de données lorsque nous éteignons un container et qu'on le relance.
- Dev Environnements : une feature très intéressante permettant de créer des environnements de développement ([plus d'informations](#)).

5 Utilisez VS Code

Enfin, pour utiliser tout ce que nous venons d'installer, il faut un éditeur de code efficace. VS Code  permet, via ses extensions d'interagir avec Docker  et WSL2 en parfaite harmonie, c'est pourquoi nous le recommandons CHAQUEMENT.

Une fois VS Code  installé, ajouter les extensions WSL (de Microsoft) et Docker (de Microsoft également).

III. Création Projet Laravel

1 Création du dossier

Pour créer un nouveau projet Laravel 📁 dans un dossier “exemple-app” il suffit juste d’entrer la ligne de commande :

```
curl -s https://laravel.build/example-app | bash
```

Avec cette commande votre projet sera installé avec plusieurs services de base (mysql, redis, et d'autres). PS : la création peut prendre du temps (5-10 minutes).

il est possible de choisir les services à installer avec le mot clé *with*. Dans le cadre de ce tutoriel, nous aurons besoin de 3 services : MySQL, mailpit et PhpMyAdmin 🚢. Les 2 premiers peuvent être téléchargés avec laravel comme suit :

```
curl -s "https://laravel.build/example-app?with=mysql,mailpit" | bash
```

une fois la commande lancée le projet va se créer (cela peut prendre un certain temps)

[illegible]

Une fois le dossier du projet créé vous allez rentrer dans le Dossier et ouvrir le fichier *docker-compose.yml*.

Ce fichier représente la configuration générale de votre conteneur, c'est ici que l'on va définir les services à installer, les versions, les images, ect. Ce fichier est très important, sans lui, pas de conteneur.

Vous pouvez voir qu'il est composé de plusieurs **section**, *Version*, *services*, *network*, *volumes*. Chaque section à son utilité. Nous allons nous concentrer sur la section service du fichier. Plus précisément, nous allons ajouter un service.

2 Ajout PhpMyAdmin 🚢

On va ajouter un service à notre projet qui est nécessaire à la gestion de notre base de donnée, PhpMyAdmin 🚢.

Pour ce faire, rendez-vous à la fin de la section services, et ajoutez y PhpMyAdmin 🚢 comme suit :

```
phpmyadmin:
  image: 'phpmyadmin:latest'
  ports:
    - '8080:80'
  networks:
    - sail
  environment:
    - PMA_ARBITRARY=1
```

Une fois cela fait, PhpMyAdmin 🚢 est intégré au projet.

IV. Utilisation

Pour utiliser le script, il vous faudra utiliser la commande suivante dans le dossier du projet (sail est installé par default lors de la création du projet) :

```
./vendor/bin/sail [command] [options] [arguments]
```

Vous pouvez créer un alias pour sail en ajoutant la ligne suivante à votre .bashrc, zshrc, ect :

```
alias sail='[ -f sail ] && sh sail || sh vendor/bin/sail'
```

1 Démarrage du container

Pour lancer le container et ainsi pouvoir travailler sur le projet, entrez dans le dossier de votre projet via un terminal et lancer la commande :

```
sail up -d
```

Cette commande lancera le container en mode détacher pour garder l'accès au terminal (pratique si l'on travaille sur VSCode)

On peut aussi lancer dans un terminal avec `sail up`

Vous pouvez maintenant avoir accès à la page de votre projet en allant sur `localhost` via un moteur de recherche comme Mozilla, Google, ect.

2 Arrêt du container

Pour éteindre le container lorsque vous avez finis de travailler, tapez la commande

```
sail down
```

