



Installation de Laravel & Docker

ROTUEUDT Thomas et MOUCHAMPS Antoine - 13 décembre 2023



N-HiTec

Allée de la Découverte 10, 4000 Liège, Belgium

nhitec.com | info@nhitec.com






Table des matières

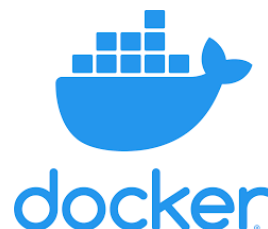
Table des matières	2
I Introduction	3
1 Qu'est-ce que Docker ?	3
2 Pourquoi Docker ?	3
3 Qu'est-ce que Laravel Sail ?	3
II Installation de Docker	5
1 Installation Linux	5
2 Installation Windows	7
2.1 Prérequis	7
2.2 Installation de Docker Desktop	7
3 Installation MacOS	8
4 Aperçu	9
5 VS Code	9
III Utilisation	10
1 Commandes	10
2 Démarrage du container	10
3 Arrêt du container	10
IV Création Projet Laravel	12
1 Création d'un nouveau projet	12
1.1 Création du dossier	12
1.2 Ajout PhpMyAdmin	13
1.3 Setup	13
1.4 Localhosts	14
2 Ajout d'un projet depuis GitHub	15
2.1 Setup	15
2.2 Dernières étapes	16
V Ensuite	17




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 quelle 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.

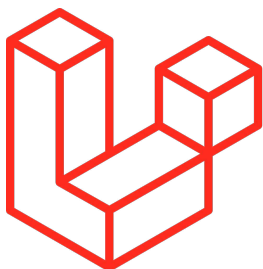
Les conteneurs sont isolés de la machine hôte, ce qui vous permet d'avoir plusieurs versions différentes de dépendances pour plusieurs projets.


Enfin, la modification d'un conteneur est extrêmement simplifiée comparée 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 , 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 1






Windows : SECTION 2



MacOS : SECTION 3

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
```

3. Ajoutez la clé GPG officielle de Docker  :

```
sudo install -m 0755 -d /etc/apt/keyrings
```

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

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

4. Configurez le répertoire :

```
echo \  
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] \  
https://download.docker.com/linux/ubuntu \  
"$(. /etc/os-release && echo "$VERSION_CODENAME)" 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, containerd, et Docker Compose :

```
sudo apt-get install docker-ce docker-ce-cli containerd.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 **Docker Destop** et installez le avec la commande :

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

Attention à remplacer <version> <arch> par ceux de l'archive téléchargée.





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 Powershell ¹. 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 redémarré votre PC, la fenêtre suivante devrait apparaître. Si ce n'est pas le cas, cherchez `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 passe ². Attention, vous devrez utiliser votre mot de passe de temps en temps, donc ne l'oubliez pas !

Enfin, exécutez `sudo apt update` et puis `sudo apt upgrade` afin d'être sur que votre environnement est à jour.

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

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 🍏

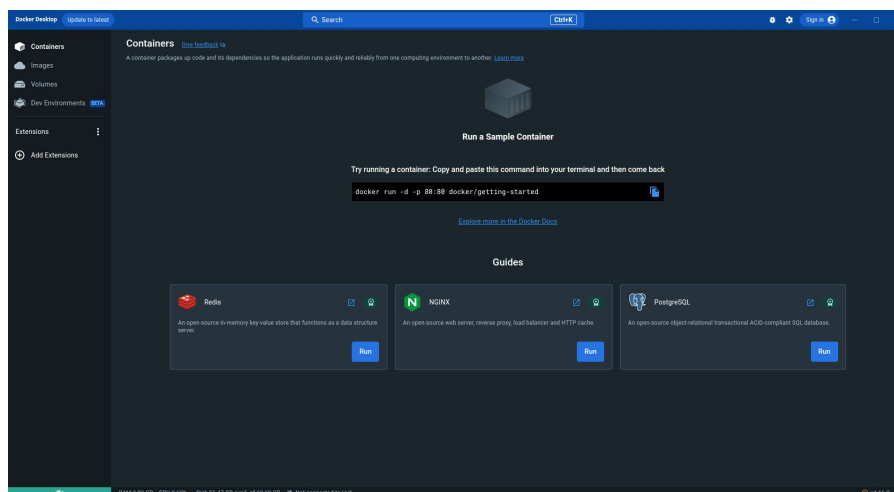
1. Téléchargez Docker Desktop 🐳 **Apple Chip** ou **Intel Chip**.
2. Installez Docker Desktop 🐳 dans le dossier application.

Pour passer à la suite




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 :

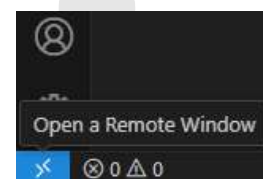
- Conteneurs : Cet onglet va contenir tous les conteneurs que vous avez créés. Les conteneurs sont en quelque sorte des images déployées et exécutées lorsque Docker  tourne.
- Images : Cet onglet va contenir toutes les images d'installation des services installés dans nos conteneurs. Les images décrivent 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 !! (surtout Windows)

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, ajoutez les extensions WSL (de Microsoft) et Docker (de Microsoft également). Une fois cela fait, cliquez sur le petit carré montré ci-joint.

Ensuite, sélectionnez Connect to WSL. Enfin, File → Open Folder... et appuyez sur ok. Normalement, vous devriez vous trouver dans le dossier source.



III. Utilisation

1 commandes

Avant de créer votre premier projet, nous allons voir les commandes de base pour utiliser Laravel Sail 🐳.

Pour utiliser le script, il vous faudra utiliser la commande suivante dans le dossier du projet (que vous créez à la section suivante) :

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

Comme taper `./vendor/bin/sail` devant chaque commande est fort fastidieux, vous pouvez créer un alias pour sail en ajoutant la ligne suivante tout en bas de votre fichier `.bashrc` (pour Windows 🪟 (dans WSL2)) ou `.zshrc` pour MacOS 🍏. Ces fichiers devraient se trouver dans les dossiers sources (`~`)

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

Ce qui vous permettra de ne plus devoir taper `./vendor/bin/sail` à chaque commande mais simplement `sail`.

2 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 lancez la commande³ :

```
sail up -d
```

Cette commande lancera le container en mode détaché pour garder l'accès au terminal (pratique puisque vous devrez exécuter de nombreuses commandes, même une fois le container démarré)⁴

C'est une fois cette commande faite (et les autres configurations effectuées) que vous pourrez accéder à votre site en allant sur <http://localhost/> ou votre URL personnalisée.

3 Arrêt du container

Pour éteindre le container lorsque vous avez finis de travailler (mais le laisser déployé dans Docker Desktop 🐳), tapez la commande

3. Si un container est déjà lancé, vous aurez une erreur ! Avec seulement Laravel Sail 🐳, on ne peut lancer qu'un container à la fois.

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


```
sail stop
```

Enfin, pour éteindre le container et le supprimer, tapez

```
sail down
```




IV. Création Projet Laravel

1. Si vous souhaitez créer un nouveau projet vierge, allez à la SECTION 1.
2. Si vous souhaitez importer un projet depuis GitHub , allez à la SECTION 2.

1 Création d'un nouveau projet


1.1 Création du dossier

Avant toute chose, créez un dossier `www/` dans votre dossier source. Ensuite, rentrer dans ce dossier car c'est ici que vous ajouterez vos projets.

Pour créer un nouveau projet Laravel  dans un dossier "exemple-app" il suffit juste d'entrer la ligne de commande (mais ne le faites pas) :

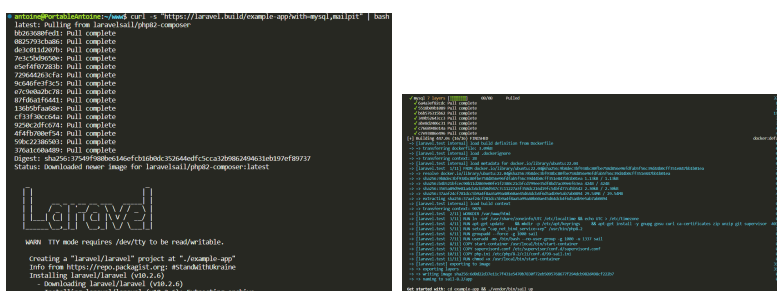
```
curl -s https://laravel.build/example-app | bash
```

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

```
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)



Une fois le dossier du projet créé, entrez dans le dossier du site (*File → Open Folder...*) et ouvrez le fichier `docker-compose.yml`.

⁵. Dans cette commande, vous pouvez changer `example-app` pour modifier le nom de votre projet. Celui-ci sera le nom du dossier créé.

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 **sections**, *Version*, *services*, *network*, *volumes*. Chaque section à son utilité. Nous allons nous concentrer sur la section **service** du fichier. Plus précisément, nous allons en ajouter un.

1.2 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/phpmyadmin'
  ports:
    - '8080:80'
  networks:
    - sail
  environment:
    - PMA_ARBITRARY=1
    - PMA_HOST=mysql
    - PMA_PORT=3306
  depends_on:
    - mysql
```

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

1.3 Setup

Il y'a plusieurs choses à faire avant de pouvoir accéder à votre site. Premièrement, configurer le fichier `.env` :

- `APP_NAME` : changez le nom de votre application.
- `APP_URL` : changez l'URL pour accéder à votre site, attention, cette url doit être conforme à celle écrite dans le fichier `hosts` de la section suivante. Par exemple, `http://example`.
- `DB_DATABASE` : changez le nom de votre base de donnée comme bon vous semble.

Les autres champs peuvent à priori rester inchangés.

Une fois cela fait, vous pouvez démarrer le container avec `sail up -d`.


Ensuite, installez les librairies Javascript 🧩 telle que Vite 🚀 avec `sail npm install`.

La dernière chose à faire est une modification **propre à WSL2**. Ajoutez donc les lignes de code renseignées au fichier `vite.config.js`.



Enfin, afin de pouvoir accéder à votre site avec l'URL fournie dans le `.env`, il est nécessaire d'ajouter un *localhost*.


1.4 Localhosts


La méthode diffère selon le système d'exploitation :

- Si vous êtes sous Windows , allez dans `C:\Windows\System32\drivers\etc\` et ouvrez le fichier `hosts`⁶. Dedans, ajoutez deux lignes comme ceci :

```
127.0.0.1 example  
::1 example
```

- Si vous êtes sous Linux , pareil mais le fichier se trouve dans `/etc/`.
- Si vous êtes sous MacOS , pareil mais le fichier se trouve dans `/private/etc/`.

Une fois cela fait, redémarrez votre navigateur et allez sur l'URL que vous avez choisie, vous devriez voir la page d'accueil de Laravel .


Vous pouvez également vous connecter à l'interface de PhpMyAdmin  en vous rendant sur **http://example:8080**. Les identifiants demandés sont ceux correspondants aux champs `DB_USERNAME` et `DB_PASSWORD` renseignés dans le `.env`⁷.

Pour passer à la suite


6. ATTENTION, vous devez ouvrir ce fichier avec les droits d'administrateur, sinon vous ne pourrez pas sauvegarder vos modifications.

7. le champs `DB_PASSWORD` peut rester vide.

2 Ajout d'un projet depuis GitHub


Lorsque vous clonez un repo sur votre machine, le site manque de composants essentiels à son fonctionnement : ses packages. Pour installer ces packages, il est nécessaire d'effectuer la commande `composer install`, sauf que vous n'avez pas `composer` d'installé. En revanche, `composer` est utilisable grâce à Laravel Sail  par `sail composer install` !


Le petit hic, c'est que Laravel Sail  est un package... à installer avec `composer install` ⁸.

Heureusement, Laravel  a la solution pour nous. Il suffit de se rendre dans le dossier de votre projet fraîchement installé et de taper :

```
docker run --rm \
  -u "$(id -u):$(id -g)" \
  -v "$(pwd):/var/www/html" \
  -w /var/www/html \
  laravelsail/php82-composer:latest \
  composer install --ignore-platform-reqs
```

Cette commande horrible permet *en gros* de lancer un container temporaire dans lequel `composer` est installé, et de l'utiliser afin de lancer la commande que l'on souhaite ⁹.

de cette manière, les packages de `composer` seront installés. Plus de renseignements sur la [Doc Laravel](#) .

Avec les packages (et donc Laravel Sail ) d'installés, vous pouvez démarer le container avec `sail up -d`.

En ce qui concerne les librairies Javascript , vous pouvez executer

```
sail npm install
```


pour installer, entres autres, Vite .

2.1 Setup

Enfin, il faut setup les paramètres du site en créant un fichier `.env` ¹⁰. Vous pouvez modifier :

- `APP_NAME` : changez le nom de votre application.
- `APP_URL` : changez l'url pour accéder à votre site. Attention, cette url doit être conforme à celle écrite dans le fichier `hosts` de la [SECTION 1.4](#)
- `DB_DATABASE` : changez le nom de votre base de donnée comme bon vous semble.

8. Le serpent qui se mord la queue, l'oeuf ou la poule, appelez ce paradoxe comme vous le voulez.

9. Remarquez que par conséquent, n'importe quelle commande PHP  peut également être exécutée en remplaçant `composer install --ignore-platform-reqs` par la commande de votre choix

10. ps : vous pouvez copier et renommer le fichier `.env` `example`, mais ne supprimez pas l'original !

— DB_HOST : à changer en mysql.

A priori, les autres champs doivent rester inchangés.



Enfin, il ne vous reste plus qu'à exécuter


```
sail artisan key: generate
```

et qu'à ajouter l'URL de votre site dans la liste de vos localhosts comme décrit dans la SECTION 1.4.

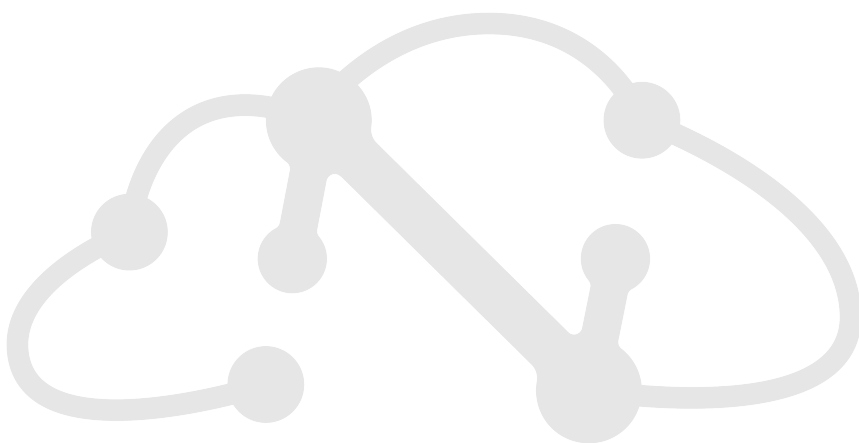
2.2 Dernières étapes

Enfin, les dernière étapes consistent à taper ces deux commandes, que nous expliquerons dans la formation suivante :

1. `sail artisan migrate:fresh --seed` pour lancer la base de donnée.
2. `sail npm run dev` pour compiler CSS  et Javascript .

Une fois cela fait, redémarrez votre navigateur et allez sur l'URL que vous avez choisie, vous devriez voir la page d'accueil de Laravel .

Pour passer à la suite



V. Ensuite

Bien, avec votre site déployé et opérationnel, vous êtes prêts pour passer à la formation suivante dans laquelle vous créerez véritablement votre site.

