


# DOCKER PHP intro



**ELAN**

14 rue du Rhône - 67100 STRASBOURG

☎ 03 88 30 78 30 📧 [elan@elan-formation.fr](mailto:elan@elan-formation.fr)

[www.elan-formation.fr](http://www.elan-formation.fr)

SAS ELAN au capital de 37 000 € -

RCS Strasbourg B 390758241 – SIRET 39075824100041 – Code APE : 8559A

N° déclaration DRTEFP 42670182967 - Cet enregistrement ne vaut pas agrément de l'État

# SOMMAIRE

<b>I.</b>	<b>Introduction et besoin .....</b>	<b>3</b>
1.	Laragon (ou XAMP / MAMP) .....	3
<b>II.</b>	<b>Docker .....</b>	<b>5</b>
1.	Un peu de vocabulaire .....	5
a.	Les Images .....	5
b.	Les conteneurs (containers) .....	5
c.	Les registres.....	5
2.	Architecture de Docker .....	6
<b>III.</b>	<b>Installation .....</b>	<b>7</b>
<b>IV.</b>	<b>Docker Desktop – Son interface .....</b>	<b>8</b>
<b>V.</b>	<b>Premiers pas .....</b>	<b>8</b>
1.	What is a container ? .....	8
a.	Docker Desktop nous facilite la vie ! .....	9
b.	Stopper le container .....	10
c.	Supprimer le container.....	10
d.	Supprimer l'image .....	11
e.	Recommencer consciencieusement .....	11
2.	How do i run a container ? .....	13
a.	Cloner le dépôt .....	13
b.	Dockerfile .....	13
c.	Build.....	13
d.	Run.....	14
3.	Créer une simple image d'une appli php .....	15
a.	Dockerfile .....	15
b.	Index.php.....	15
c.	Build.....	16
d.	Run.....	16
e.	Volume .....	16

## I. Introduction et besoin

### 1. Laragon (ou XAMP / MAMP)

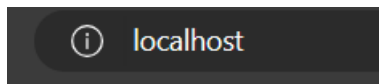
Tout d'abord, nous avons utilisé pour le moment Laragon, XAMP ou MAMP afin d'interpréter du code PHP et de gérer une BDD MySQL.



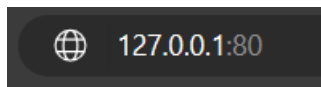
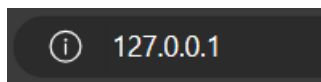
En effet, on distingue bien ci-dessus que 2 services sont lancés :

- 1 pour Apache sur le port 80
- 1 pour MySQL sur le port 3306

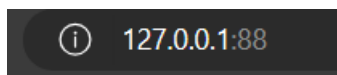
Nous avons eu l'habitude jusqu'à présent de taper « localhost » (littéralement l'hôte local, c'est-à-dire le serveur local) dans la barre de recherche dans le navigateur



L'adresse correspondant à localhost est en fait à défaut 127.0.0.1 (peu importe le système d'exploitation). Donc si on remplace localhost par 127.0.0.1 ça ne change rien ; On peut aussi ajouter le port 80.



En revanche, si on met un autre port ou une autre adresse ip, fatalement on ne tombe pas sur le service lancé par laragon





## Désolé, impossible d'accéder à cette page.

127.0.0.1 a refusé la connexion.

### Essayez :

- Effectuez une recherche Bing pour [127 0 0 1](#)
- Vérification de la connexion
- [Vérification du proxy et du pare-feu](#)

ERR\_CONNECTION\_REFUSED

Et l'un des problèmes de ces logiciels cités, c'est que sachant que nous utilisons actuellement des BDD en local, **les données ne persistent pas au sein du projet** (d'où la nécessité d'exporter la BDD à chaque fois).

Et encore, nous avons fait jusqu'à présent des applications qui utilisent peu de services (pas de mailer, pas de compilateur, pas de système de cache etc...)

Lorsque vous arrivez sur une nouvelle machine, plusieurs choses peuvent changer. Par exemple les paramètres dans le PHP.ini ne seront pas forcément les mêmes, ou même les versions de PHP et MySQL ne seront pas les bonnes. Et afin d'éviter ça...

## II. Docker

Docker va nous permettre de paramétrer un environnement et de pouvoir l'utiliser sur différentes machines (et sur différents systèmes d'exploitation) ! Et en plus, nous allons voir dans ce support comme faire directement persister les données de la BDD d'un projet directement dans celui-ci.

### 1. Un peu de vocabulaire

Avant de plonger dans le monde de Docker, familiarisons-nous avec quelques termes clés.

#### a. Les Images

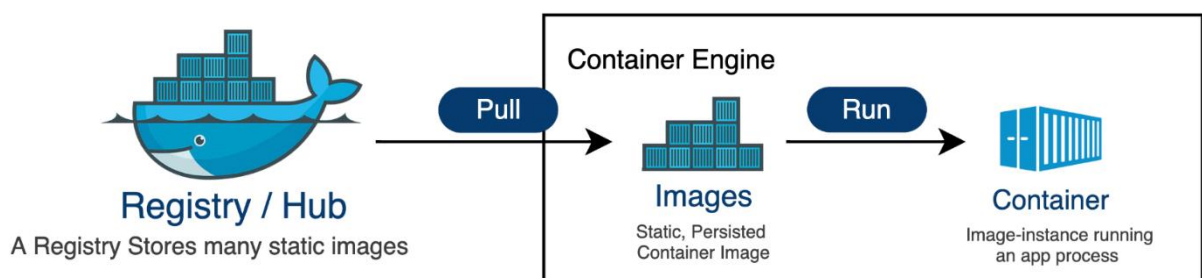
Une **image Docker** est un **ensemble immuable de fichiers**, de configurations et de dépendances nécessaires pour exécuter une application (exemple : apache).

#### b. Les conteneurs (containers)

Un **conteneur** est l'**instance d'une image**. Contrairement à une machine virtuelle, un conteneur partage le noyau du système d'exploitation avec le système hôte, ce qui le rend plus efficace et portable (exemple : je vais créer un conteneur grâce à l'image apache, ce qui va créer une nouvelle instance de cette image. C'est ce que fait **grossièrement** Laragon lorsque l'on lance le serveur).

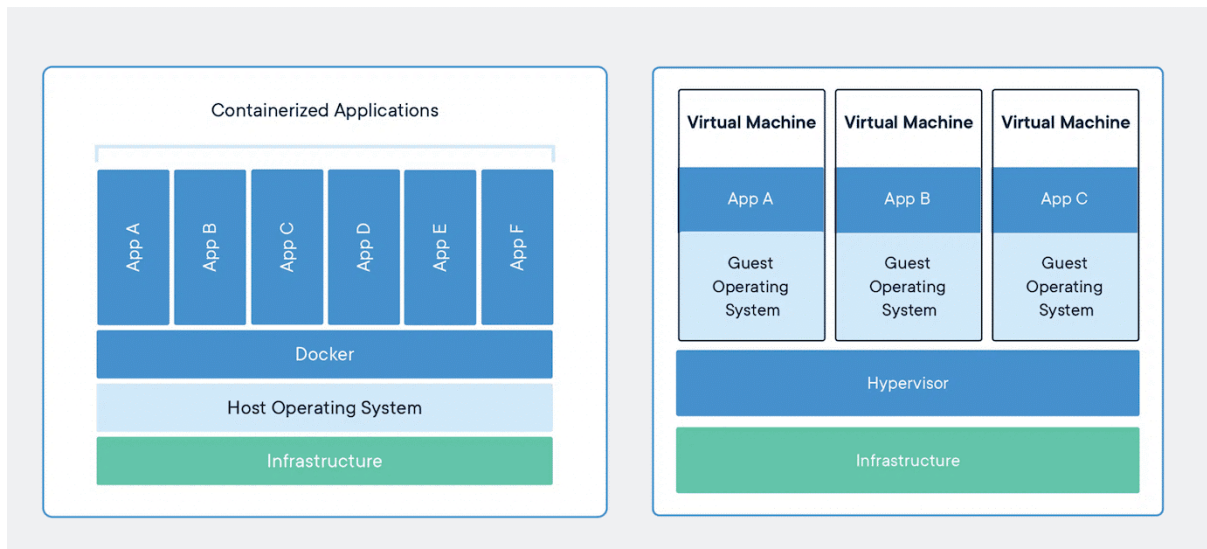
#### c. Les registres

Les registres sont des **dépôts centralisés où les images peuvent être téléchargées** et partagées. Le registre officiel (et par défaut) de Docker est le Docker Hub



DU REGISTRE AU CONTENEUR

## 2. Architecture de Docker



ARCHITECTURE DE DOCKER

ARCHITECTURE D'UNE MACHINE VIRTUELLE

Avant Docker, on recourait fréquemment aux **machines virtuelles (VM)** pour exécuter des applications. Ces VM pouvaient émuler des ordinateurs physiques, permettant aux développeurs de subdiviser un serveur en plusieurs (voir la figure de droite ci-dessus).

Mais cette méthode comportait des inconvénients. **Chaque VM renfermait une reproduction complète du système d'exploitation**, de l'application, et des éléments nécessaires, occupant parfois plusieurs dizaines de gigaoctets. De plus, la virtualisation matérielle pour un système d'exploitation invité entraînait des coûts significatifs.

Dans Docker (voir la figure de gauche ci-dessus), les conteneurs sont des abstractions au niveau de la couche applicative qui peuvent contenir à la fois du code et des dépendances. **Sur une même machine, plusieurs conteneurs peuvent fonctionner comme des processus isolés.**

### III. Installation

Pour lancer un container Docker, on le fait soit avec l'application Docker Desktop, soit en ligne de commande avec le Docker-cli. Les 2 façons de faire nécessitent le **Docker engine**. Et ça tombe bien, il est contenu dans l'installation du Docker Desktop ! (Comme ça on pourra à la fois visualiser les conteneurs avec une interface user-friendly, ou bien dans une console).

Choisissez en fonction de votre OS :

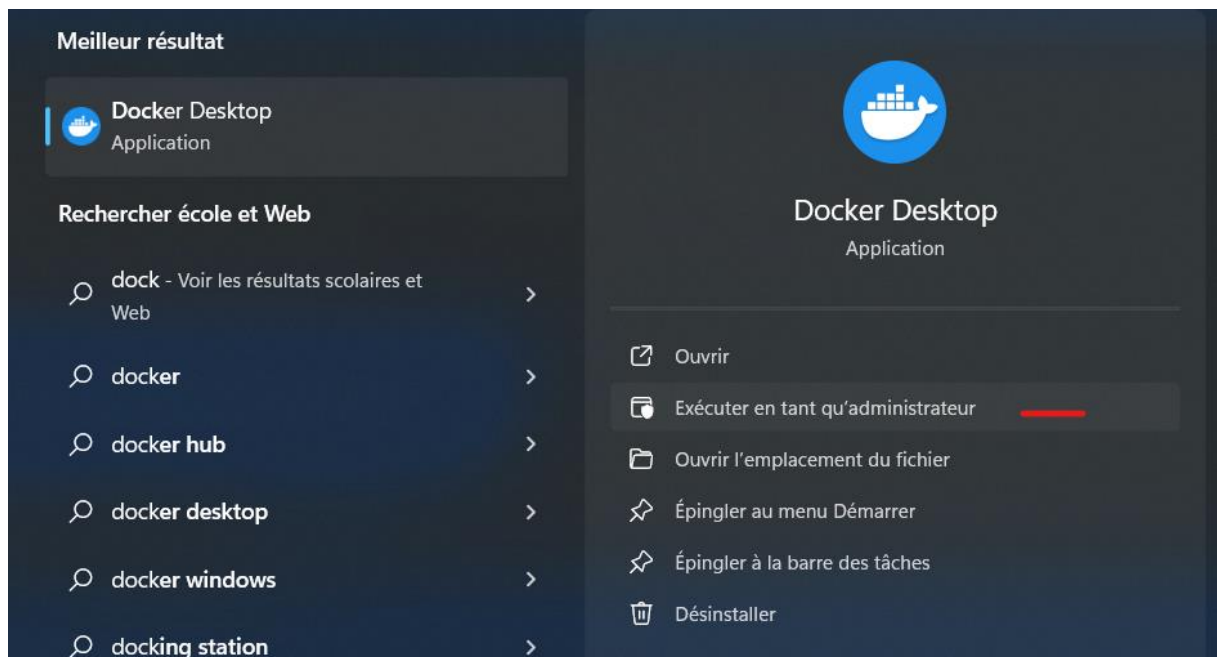
[Install Docker Desktop on Windows | Docker Docs](#)

[Install Docker Desktop on Linux | Docker Docs](#)

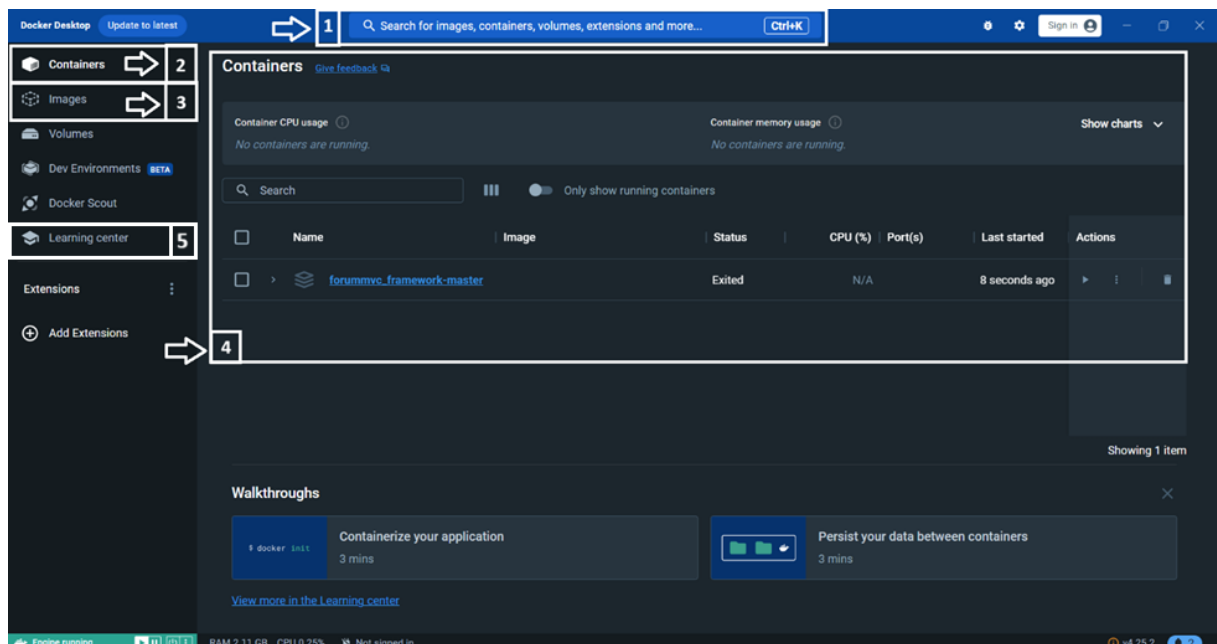
[Install Docker Desktop on Mac | Docker Docs](#)

Installez simplement l'application, vous pouvez laisser les paramètres à défaut.

Pensez ensuite à ouvrir Docker Desktop en tant qu'administrateur :



## IV. Docker Desktop – Son interface

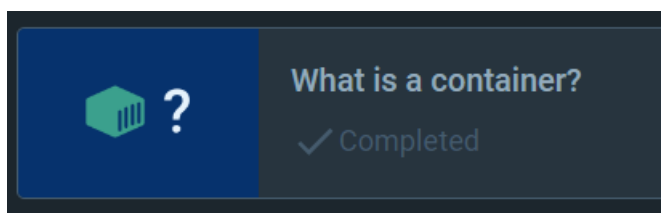


- 1 – la barre de recherche. Si vous recherchez une image l'application regardera dans le Docker Hub (rappelez vous du glossaire, le Docker Hub est un registre !)
- 2 – les containers. C'est la page qui répertorie les containers ayant déjà été utilisés sur votre machine
- 3 – les images. C'est la page qui répertorie les images ayant déjà été utilisées sur votre machine
- 4 – le corps : nous sommes actuellement sur la page des containers. Ils sont listés au sein d'un tableau
- 5 – learning center : nous voyons ça tout de suite

## V. Premiers pas

Docker Desktop offre plusieurs tutos afin de prendre en main l'application (dans le learning center)

### 1. What is a container ?





Nous allons commencer par celui-ci




## a. Docker Desktop nous facilite la vie !

Lorsque que l'on débute ce mini tuto, Docker Desktop nous crée un container :

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started
<input type="checkbox"/>	 <a href="#">welcome-to-docker</a> 8ac4d1f3cf0f	<a href="#">docker/welcome-to-docker:latest</a>	Running	0%	<a href="#">8088:80</a> 	7 minutes ago

Celui-ci est composé de plusieurs éléments :

- Son nom : welcome-to-docker
- Son id : 8ac4d1f3cf0f dans mon cas
- L'image par laquelle le container a été créé : docker/welcome-to-docker:latest
- Son statut : « Running » (en cours)
- L'utilisation (pour le container) en pourcentage du processeur
- Le port sur lequel il est redirigé :
  - 🔗 Nous voyons ici la notation « 8088 :80 » ce qui signifie qu'au lieu de le lancer sur le port 80 (comme nous avons l'habitude de faire avec Laragon par exemple), nous allons le lancer sur le port 8088
  - 🔗 Le but d'utiliser un port différent de 80 est de pouvoir lancer plusieurs containers en même temps : il faut donc utiliser un port différent à chaque fois
- « Last started » nous indique la dernière fois que le container a été lancé (run)

Lorsque l'on clique sur le port [8088:80](#) , nous sommes redirigé sur le front-end de l'application (on peut aussi y accéder avec l'adresse ip : 127.0.0.1:8088)



Nous allons retrouver ces informations en ligne de commande (**pensez à ouvrir l'invité de commande de votre choix en tant qu'admin !**) :

C:\Windows\System32>docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8ac4d1f3cf0f	docker/welcome-to-docker:latest	"/docker-entrypoint..."	2 minutes ago	Up 2 minutes	0.0.0.0:8088->80/tcp	welcome-to-docker

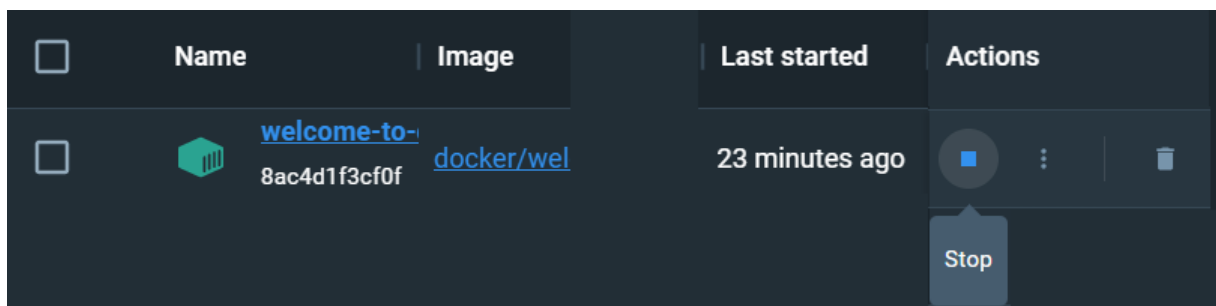
- **docker ps** : liste l'ensemble des containers

```
C:\Windows\System32>docker image ls
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
docker/welcome-to-docker  latest         c1f619b6477e   2 months ago   18.6MB
```

■ **docker image ls** : répertorie les images

Nous allons à présent **décortiquer** en ligne de commande ce que Docker Desktop vient de nous prémâcher. D'abord, repartons sur de bonnes bases :

## b. Stopper le container

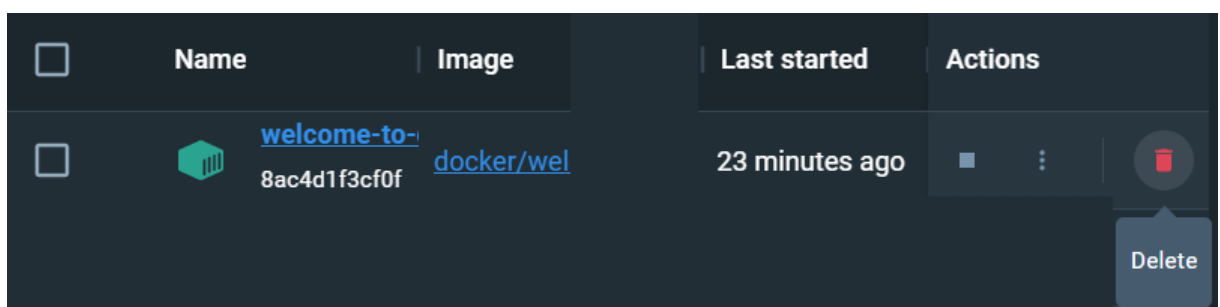


■ On commence par stopper le container

```
C:\Windows\System32>docker stop 8ac4d1f3cf0f
8ac4d1f3cf0f
```

👉 Ou dans un terminal, on utilise « `docker stop <id>` »

## c. Supprimer le container



■ Puis on supprime le container

```
C:\Windows\System32>docker rm 8ac4d1f3cf0f
8ac4d1f3cf0f
```

👉 Ou dans un terminal, on utilise « `docker rm <id>` »

#### d. Supprimer l'image

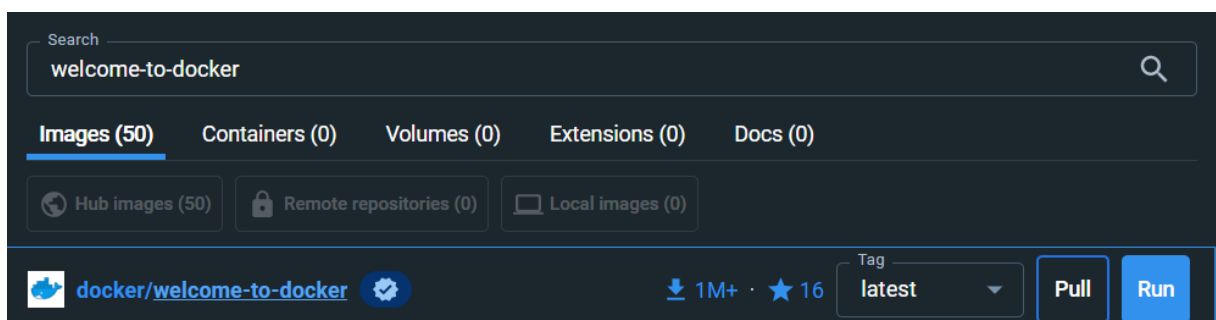
<input type="checkbox"/>	Name	Tag ↓	Status	Created	Size	Actions
<input type="checkbox"/>	<a href="#">docker/welcome-to-docker</a> c1f619b6477e	latest	Unused	2 months ago	18.55 MB	 

- On supprime l'image welcome-to-docker (depuis l'onglet images de Docker Desktop !)

```
C:\Windows\System32>docker image rm c1f619b6477e
Untagged: docker/welcome-to-docker:latest
```

↳ Ou dans un terminal, on utilise « `docker image rm <id>` »

#### e. Recommencer consciencieusement



- Dans la barre de recherche on peut retrouver l'image, et de là on aurait pu simplement faire « run ». Mais nous allons le faire en ligne de commande pour bien comprendre ce qu'il s'y passe :

```
C:\Windows\System32>docker run --name welcome-to-docker -d -p 8088:80 docker/welcome-to-docker:latest
Unable to find image 'docker/welcome-to-docker:latest' locally
latest: Pulling from docker/welcome-to-docker
96526aa774ef: Already exists
740091335c74: Pull complete
da9c2e764c5b: Pull complete
ade17ad21ef4: Pull complete
4e6f462c8a69: Pull complete
1324d9977cd2: Pull complete
1b9b96da2c74: Pull complete
5d329b1e101a: Pull complete
Digest: sha256:eedaff45e3c78538087bdd9dc7afafac7e110061bbdd836af4104b10f10ab693
Status: Downloaded newer image for docker/welcome-to-docker:latest
```


↳ Dans un terminal, on utilise « `docker run --name welcome-to-docker -d -p 8088:80 docker/welcome-to-docker:latest` »

- **docker run** lance le container, et s'il n'existe pas le créer d'abord
- **-- name <name>** est optionnel, il sert à donner un nom au container
- **-d** est optionnel, il sert à lancer le container en mode « détaché » (en fond)
- **-p <nouveauPort :défautPort>** sert à rediriger le port (de base 80, dans cet exemple on le redirige vers le port 8088)
- **<image>** enfin on précise de quelle image on va créer une instance de celle-ci (dans cet exemple docker/welcome-to-docker:latest)

```
Unable to find image 'docker/welcome-to-docker:latest' locally
latest: Pulling from docker/welcome-to-docker
96526aa774ef: Already exists
740091335c74: Pull complete
da9c2e764c5b: Pull complete
ade17ad21ef4: Pull complete
4e6f462c8a69: Pull complete
1324d9977cd2: Pull complete
1b9b96da2c74: Pull complete
5d329b1e101a: Pull complete
```

On remarque d'ailleurs que le Docker Engine a pull (récupéré) automatiquement l'image, quand nous avons « run » une image n'étant pas déjà présente sur la machine (on l'a supprimée juste avant).

On peut à présent admirer le résultat en tapant : 127.0.0.1:8088 sur la barre URL de votre navigateur préféré.



**Congratulations!!!**  
You ran your first container.

Par la suite on pourra **stopper / supprimer le container, puis supprimer l'image**.

## 2. How do i run a container ?

### How do I run a container?



#### Images are used to run containers

In this guide, you create an image using a Dockerfile and a sample application.

Comme précisé en première étape, ce tuto va nous apprendre comment créer une image avec un Dockerfile à travers un exemple d'application.

### a. Cloner le dépôt

```
C:\>git clone https://github.com/docker/welcome-to-docker
Cloning into 'welcome-to-docker'...
remote: Enumerating objects: 125, done.
remote: Counting objects: 100% (58/58), done.
```

- Comme demandé à la 2<sup>ème</sup> étape, on clone le dépôt (dans le dossier de votre choix, n'oubliez pas de lancer le terminal en tant qu'admin), puis on y accède avec « cd ».

### b. Dockerfile

Dans ce dossier nous voyons **un fichier Dockerfile**, qui comporte les instructions permettant de créer l'image. **Jetez-y un œil**. Nous n'allons pas rentrer dans le détail pour le moment.

### c. Build

A présent nous allons créer une image en se basant sur le Dockerfile avec « **docker build -t welcome-to-docker .** »

- **docker build** nous sert à créer l'image
- L'option **-t <nom>** nous permet de la nommer
- L'option **.** précise qu'on l'a build dans le dossier courant

```
C:\>cd welcome-to-docker>docker build -t welcome-to-docker .
[+] Building 50.1s (11/11) FINISHED
```

```
C:\Windows\System32>docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
welcome-to-docker   latest       260ff0b5c6d8  47 seconds ago 226MB
```

En faisant un « **docker image ls** », on découvre alors la nouvelle image créée.

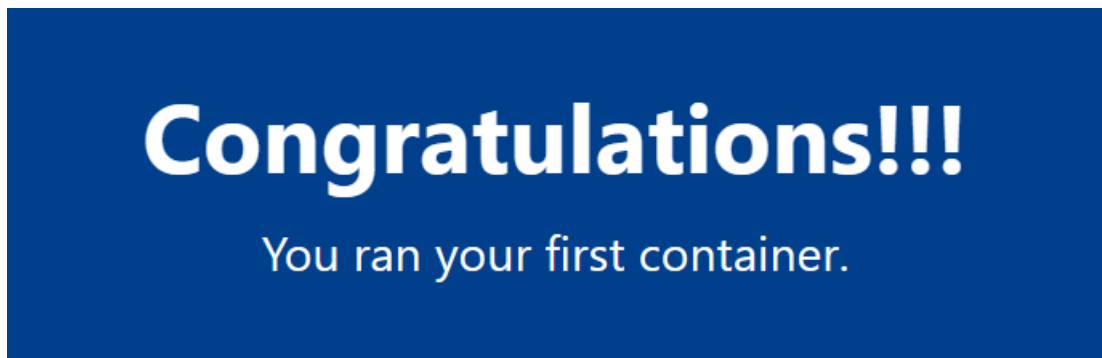
#### d. Run

Nous pouvons à présent lancer de nouveau un conteneur avec l'image que nous venons de créer

```
C:\Windows\System32>docker run --name welcome-to-docker -d -p 8088:80 welcome-to-docker  
c319768310d95f4a0841a049447910907d5e143dccd241f78abcaff45f266fa1
```

« **docker run --name welcome-to-docker -d -p 8088:80 welcome-to-docker** »

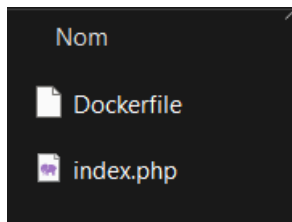
On peut une fois de plus admirer le résultat en tapant : 127.0.0.1:8088 sur la barre URL de votre navigateur préféré.



Par la suite on n'oubliera pas de **stopper / supprimer le container, puis supprimer l'image.**

### 3. Créer une simple image d'une appli php

Nous allons maintenant voir comment créer l'image d'une appli php grâce à apache. Tout d'abord dans un nouveau dossier on va ajouter 2 fichiers :



- Un Dockerfile (sans extension)
- Un fichier index.php (pour l'exemple)

#### a. Dockerfile

Après avoir ouvert le dossier sur VSCode, ajoutons quelques lignes au Dockerfile :

```
Dockerfile > ...
1  # use a version of php and apache
2  FROM php:8.2-apache
3
4  # copy the local directory to the current local directory of our docker image
5  COPY . /var/www/html/
6
7  # port 80 is the default port for HTTP
8  EXPOSE 80
```

- La ligne 2 nous permet de récupérer PHP et apache depuis registre Docker Hub (dans ce cas précis on utilise la version 8.2, on aurait pu en choisir une autre depuis le Docker Hub)
- La ligne 4 va nous permettre de copier le contenu de dossier courant (c'est-à-dire le Dockerfile et surtout notre fichier index.php) vers le dossier html (équivalent au dossier www de laragon ou htdoc de MAMP) dans notre image
- Enfin en ligne 8 on indique que le port de ce service Apache sera le port 80 (nous le redirigerons au moment de créer le conteneur)

#### b. Index.php

On ajoute du contenu dans notre fichier index.php :

```
index.php
1  <?php
2
3  var_dump("Test");
```

### c. Build

On build (créé) l'image (comme on l'a déjà fait précédemment) :

```
C:\... \docker-php>docker build -t simple-php .  
[+] Building 0.2s (8/8) FINISHED
```

### d. Run

Puis on run (lance) un nouveau container avec cette nouvelle image :

```
C:\... \docker-php>docker run --name test -d -p 8066:80 simple-php
```

Par la suite nous pouvons admirer le résultat sur l'ip 127.0.0.1:8066 :

```
string(4) "Test"
```

A présent si on change notre index.php :

```
index.php  
1 <?php  
2  
3 var_dump("Woohooo!");
```

Eh bien... rien ne se passe :

```
string(4) "Test"
```

En effet, on a fait une copie de notre dossier au moment de la création d'image, de ce fait les changements ne sont pas pris en compte. Pour pallier ce problème, nous allons utiliser un volume.

### e. Volume

Après avoir **stopé et supprimé le container**, on va en créer un nouveau mais cette fois ci avec un volume :

```
C:\... \docker-php>docker run --name test -d -p 8066:80 -v ./var/www/html/ simple-php
```

- **-v <chemin du volume sur la machine> :<chemin du volume sur le futur container>**  
cette option nous permet de synchroniser le dossier que l'on souhaite (en l'occurrence ici « . » vu que notre terminal est ouvert dans le dossier de notre projet) avec le dossier présent dans notre container Docker.

A présent les modifications sont prises en compte !

```
string(8) "Woohooo!"
```