

# Docker

---

INITIATION A DOCKER DESKTOP

Léo LAROU-CHALOT  
SAFE BEAR | PARIS

# Vous avez dis « conteneur » ?

A la différence d'une **machine virtuelle** (VM), qui fait ce que l'on appelle de la virtualisation lourde lorsque l'on recréer un système complet dans le système hôte, un **conteneur Linux** est un **processus** ou un ensemble de processus isolés du reste du système, tout en étant **légers**.

Un conteneur permet de faire de la **virtualisation légère**, c'est à dire qu'il ne virtualise pas les ressources, il ne crée qu'une **isolation des processus**. Le conteneur partage donc les ressources avec le système hôte.

Cette documentation abordera les sujets et problématiques rencontrés en entreprises.

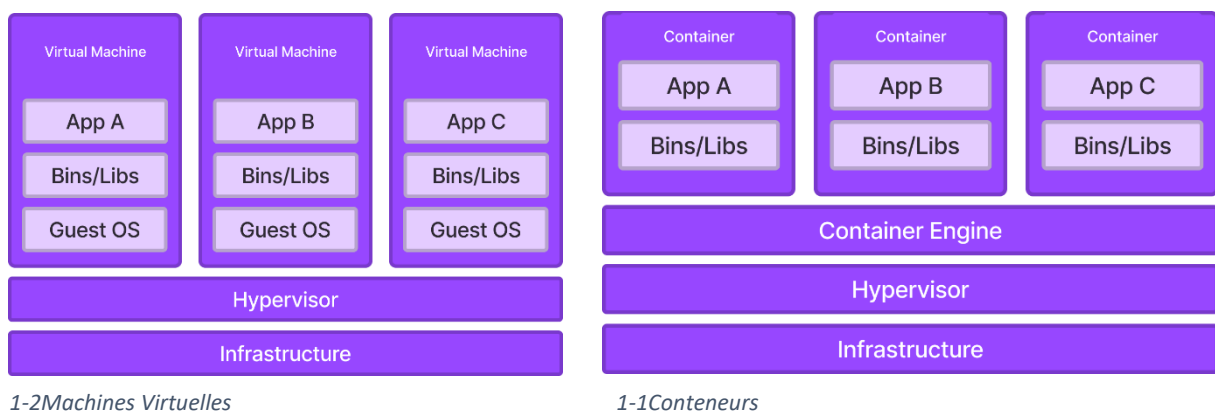
1	Machine Virtuelle VS Conteneur .....	3
1.1	La découverte .....	3
1.2	Pourquoi utiliser des conteneurs ? .....	3
2	Pourquoi utiliser Docker ? .....	4
2.1	Isolation des environnements .....	4
2.2	Portabilité .....	4
2.3	Rapidité et efficacité.....	4
2.4	Facilité de gestion.....	4
2.5	Scalabilité <sup>2</sup> .....	4
2.6	Résumé.....	4
3	Débuter avec Docker Desktop.....	5
3.1	Installation et références .....	5
3.2	L'interface en Lignes de Commandes.....	5
3.2.1	Quelques commandes.....	5
3.3	Création d'un dockerfile.....	7
4	Annexes .....	8
4.1	Définitions .....	8
4.1.1	Virtualisation .....	8
4.1.2	Scalabilité.....	8
4.1.3	Registry.....	8
4.1.4	Daemon .....	8

# 1 MACHINE VIRTUELLE VS CONTENEUR

## 1.1 LA DECOUVERTE

Les conteneurs et les machines virtuelles sont des technologies de virtualisation des ressources très similaires. La virtualisation<sup>1</sup> est le processus par lequel une ressource est singulière au système (RAM, CPU, disque ou réseau) peut être « virtualisé » et représenté sous forme de ressources multiples.

La principale différence entre les conteneurs et les machines virtuelles vient du fait que la machine virtuelle virtualisent toutes une machine, jusqu'aux couches matérielles, tandis que le conteneur ne virtualise que les couches logicielles au-dessus du système d'exploitation.



## 1.2 POURQUOI UTILISER DES CONTENEURS ?

Un conteneur ne réserve pas la quantité de CPU, RAM et disque attribué auprès du système hôte. Ainsi, nous pouvons attribuer autant d'espace que nous le souhaitons, mais si celui-ci n'utilise que 2Go, le reste ne sera pas bloqué.

Le conteneurs n'ayant pas besoin d'une virtualisation des ressources mais seulement d'une isolation de celles-ci, ils peuvent démarrer beaucoup plus rapidement et plus fréquemment qu'une VM sur les serveurs hôtes, et ainsi réduire un peu plus les frais de l'infrastructure.

Les conteneurs permettent de réduire les coût, d'augmenter densité de l'infrastructure, tout en améliorant le cycle de déploiement.

## 2 POURQUOI UTILISER DOCKER ?

---

### 2.1 ISOLATION DES ENVIRONNEMENTS

Les conteneurs, comme ceux fournis par Docker, encapsulent une application et toutes ses dépendances, garantissant ainsi une isolation totale par rapport au reste du système. Cela résout souvent des problèmes liés à la compatibilité entre différentes versions de bibliothèques ou de dépendances.

### 2.2 PORTABILITE

Un conteneur Docker est léger et peut être exécuté de manière cohérente sur n'importe quel système prenant en charge Docker. Cela simplifie le processus de déploiement en garantissant que l'application fonctionne de la même manière sur les environnements de développement, de test et de production.

### 2.3 RAPIDITE ET EFFICACITE

Les conteneurs partagent le même noyau du système d'exploitation hôte, ce qui les rend plus légers par rapport aux machines virtuelles. Ils peuvent démarrer rapidement et consommer moins de ressources système.

### 2.4 FACILITE DE GESTION

Docker fournit des outils pour créer, distribuer et gérer des conteneurs. L'automatisation intégrée facilite le déploiement d'applications complexes et la gestion des mises à jour.

### 2.5 SCALABILITE<sup>2</sup>

La conteneurisation facilite le déploiement et la mise à l'échelle d'applications, que ce soit horizontalement en ajoutant des instances ou verticalement en augmentant les ressources d'une instance.

### 2.6 RESUME

En résumé, l'utilisation de Docker et de la conteneurisation simplifie le cycle de vie du développement logiciel, améliore la portabilité des applications et offre une gestion plus efficace des ressources. Cela répond aux besoins de l'approche DevOps en favorisant la collaboration entre les équipes de développement et d'exploitation, tout en automatisant les processus pour des déploiements plus rapides et fiables.

## 3 DEBUTER AVEC DOCKER DESKTOP

### 3.1 INSTALLATION ET REFERENCES

Lien de téléchargement et registry<sup>3</sup> officielle Docker : <https://hub.docker.com>

### 3.2 L'INTERFACE EN LIGNES DE COMMANDES

L'interface en ligne de commande est l'outil qui va nous permettre de discuter avec le daemon<sup>4</sup> Docker installé sur notre ordinateur.

#### 3.2.1 Quelques commandes

**docker images** : Permet de lister les images<sup>5</sup> disponible localement ;

```
MINGW64/c/Users/larou
larou@Asus-Léo MINGW64 ~
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
node                latest             07308918cc5b       8 days ago         1.1GB
vsc-ubuntu-b54dcf6defd5cceca880779763a20412686fcc336d9703fcc9d5536b086ece2b-features  latest            829daf2baa3d       2 weeks ago        1.36GB
vsc-volume-bootstrap  latest            625b26892f91       3 weeks ago        786MB
mcr.microsoft.com/devcontainers/base               jammy              10720142a5ba       3 weeks ago        686MB
vsc-m2l_web-458f17b9fd539784d51dbd17fb021dfd96d0ed8182fabcc6aaad6e03dcfe11b-features  latest            24fec3fe888b       4 weeks ago        1.36GB
vsc-bwtch-bot-d794e3dd08151bebb8730a1304f35a3c8d6502aa0bf5e7f9c12b5f295c2aabc0-features  <none>            3ba066c8fd89       4 weeks ago        786MB
<none>              latest            5fe9266b140c       5 weeks ago        270MB
welcome-to-docker   node              d7eb1d080096       5 weeks ago        1.1GB
httpd                latest            75a48b16cd56       5 weeks ago        168MB
python               latest            17e65561fd2c       6 weeks ago        1.02GB
php                  latest            31a4051b14f9       7 weeks ago        529MB
ubuntu               latest            e4c58958131a       8 weeks ago        77.8MB
mysql                latest            2d9aad1b5856       4 months ago       574MB
docker/welcome-to-docker  latest            912b66cfd46e       5 months ago       13.4MB
hello-world          latest            9c7a54a9a43c       6 months ago       13.3kB
alpine                3.16.3            bfe296a52501       12 months ago      5.54MB
docker/desktop-git-helper  latest            c1e302e18fba86bb07f6b657155011bd6791dfc5  13 months ago      45.8MB
docker/dev-environments-default  stable-1         7eb85f44c229       14 months ago      610MB
crazymax/linguist      7.20.0            dc5f9ae3be6a       18 months ago      72.2MB
docker/desktop-git-helper  5a4fc126aadcd3f6cc3a011aa991de982ae7000  efe2d67c403b       2 years ago        44.2MB

larou@Asus-Léo MINGW64 ~
$ |
```

**docker ps** : Permet de lister les conteneurs<sup>6</sup> disponibles localement ;

```
MINGW64/c/Users/larou
larou@Asus-Léo MINGW64 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES

larou@Asus-Léo MINGW64 ~
$ |
```

**docker pull** : Télécharge depuis un dossier distant ou depuis le Docker Hub une image existante ;

```
MINGW64/c/Users/larou
larou@Asus-Léo MINGW64 ~
$ docker pull python
Using default tag: latest
latest: Pulling from library/python
90e5e7d8b97a: Already exists
27e1a8ca91d3: Already exists
d3a767d1d12e: Already exists
711be5dc5044: Already exists
7ad48fee4003: Pull complete
a319993f7bdd: Pull complete
c5bc2fe650db: Pull complete
0303a8131ddc: Pull complete
Digest: sha256:17715e8a627c47fa2a52829d470328426afd8de590aabb10a59aadabd7ff0099
Status: Downloaded newer image for python:latest
docker.io/library/python:latest

What's Next?
View a summary of image vulnerabilities and recommendations - docker scout quickview python

larou@Asus-Léo MINGW64 ~
$ |
```

**docker run -it [nom de l'image]** : Permet de faire tourner le conteneur de façon interactive ;  
Le drapeau -it dans le contexte de la commande docker run combine deux indicateurs : -i (interactive) et -t (tty). Ensemble, ils permettent d'ouvrir une session interactive dans le conteneur Docker, ce qui est utile pour les applications en mode terminal.

```
MINGW64:/Users/larou
larou@Asus-Léo MINGW64 ~
$ docker run -it python
Python 3.12.0 (main, Nov 21 2023, 17:38:35) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Je peux le vérifier avec **docker ps**

```
MINGW64:/
larou@Asus-Léo MINGW64 /
$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
e4eebdd6f0cd   python   "python3"   About a minute ago   Up About a minute   ports   gracious_jang

larou@Asus-Léo MINGW64 /
$
```

**docker run -it -d [nom de l'image]** : Permet de faire tourner le conteneur en détaché (arrière plan)

```
MINGW64:/
larou@Asus-Léo MINGW64 /
$ docker run -it -d python
6812c0ef62dca7248e5c0b05ad832ef459dc39c853b380cee1ec43bf3a339a81

larou@Asus-Léo MINGW64 /
$ |
```

**docker stop [id de l'image]** : Permet d'arrêter l'exécution du conteneur

```
MINGW64:/
$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
e4eebdd6f0cd   python   "python3"   About a minute ago   Up About a minute   ports   gracious_jang

larou@Asus-Léo MINGW64 /
$ docker stop e4eebdd6f0cd
e4eebdd6f0cd

larou@Asus-Léo MINGW64 /
$
```

**docker system prune** : Permet de nettoyer l'ensemble des conteneurs, réseaux, images et caches

```
MINGW64:/
larou@Asus-Léo MINGW64 /
$ docker system prune
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
6812c0ef62dca7248e5c0b05ad832ef459dc39c853b380cee1ec43bf3a339a81
e4eebdd6f0cd35d66fadef3712f4f31716dc74ede3d710f2baedfbaec736d145
7545cbaedd5eaf685732939f43fbb1b6ee1d6ef0ed9bc505250206dbdd10b513
5a97820839058b6c9b1ee3b611387aaf3e3767e2d4166fa1953e2392c9b5550
76ddf4c35add9eb4fee0222f514f941c60fb78b2216e7d1a1bcd1462bc01866c
d6ae5b0018e75701d745fb209ac72f7c052e71bdb752b9a9c55e64ad1867f12c

Deleted Images:
deleted: sha256:3ba066c8fd891cbe13e8abc1e273ea61a2e7027240236f16f5f5c455e7f46700
untagged: node@sha256:bff18fc580177cd927173c8617cf7f527a1b7f62c7de882ee17a42d065f4b70e
deleted: sha256:d7eb1d08009693a150db373a0c79e8eb98f7de258e607c7dd3757b7745d4fa7c
deleted: sha256:4b714c690dc4942af6179857844c0b895fed68312e355a0d711cb93e91f289f0
deleted: sha256:0ba9c0de22e1fe37cfff1366e90c7bb04ada203dd02e2a4245d8e0a347cefb5b08
deleted: sha256:09aba2a8bb52aef4dbf4acb04208982515685462f22523157ea8e926e20b11a
deleted: sha256:2fc10e23756a6ab59d01e7e285baedb8e335763843b92236c02e56962643272a
untagged: python@sha256:2586dd7abe015eeb6673bc66d18f0a628a997c293b41268bc981e826bc0b5a92
deleted: sha256:17e65561fd2c7ecae6297c2cc049960607ff94f92f6a1ca3857f94569fc9c199
deleted: sha256:781e8bd079dcc84a458a8f4f3a150194c7d4d97e99f46c969a5f021443245582
deleted: sha256:fe26881c5fffddca/e97ebdd52578444af7a9c2b490ba91b5d793de4d8c3c984
deleted: sha256:75a8404e2d8b812db31af6baf763ab5134484213933def00b09
deleted: sha256:a2624f76c8a0e684fba575d79ba0036d1e2c448d4e037d6e6eae596c3bd16bcd
deleted: sha256:935293de7e6812c3015bb510a60e06d0ff0528c47f74ddcc4e6528ab8f80686
deleted: sha256:3a78d30473881be3a54a8b6a1de5da7acd2d645c813dcb7331527a30e50a993
deleted: sha256:354f4b6949596277291588a6a2b924144fa72edbc21ff3c12e168187e233fd0d
deleted: sha256:2fa37f2ee66efbd308b9b91bce81c262f5e6ab6c3bf80536632af60cc602785c
```

### 3.3 CREATION D'UN DOCKERFILE

Pour pouvoir créer une image, nous aurons besoin de créer un **dockerfile**.

Un dockerfile est au conteneur ce que package.json est à Node ou ce que composer.json est à PHP.

Il faut voir cela comme un fichier de configuration.



## 4 ANNEXES

---

### 4.1 DEFINITIONS

#### 4.1.1 Virtualisation

Processus par lequel une ressource est singulière au système (RAM, CPU, disque ou réseau) peut être « virtualisé » et représenté sous forme de ressources multiples.

#### 4.1.2 Scalabilité

"Scalable" (ou "scalabilité" en français) se réfère à la capacité d'un système, d'une application ou d'une architecture à évoluer et à gérer une augmentation de la charge de travail de manière efficace, sans compromettre ses performances. En d'autres termes, un système scalable peut s'adapter à une croissance ou à une diminution de la demande sans sacrifier ses fonctionnalités ou sa qualité de service.

#### 4.1.3 Registry

Une registry est un logiciel qui permet de partager des images à d'autres personnes. C'est un composant majeur dans l'écosystème Docker, car il permet à :

- des développeurs de distribuer des images prêtes à l'emploi et de les versionner avec un système de tags ;
- des outils d'intégration en continu de jouer une suite de tests, sans avoir besoin d'autre chose que de Docker ;
- des systèmes automatisés de déployer ces applications sur vos environnements de développement et de production.

#### 4.1.4 Daemon

un "daemon" (ou démon en français) est un programme informatique qui s'exécute en arrière-plan de manière continue, sans l'interaction directe d'un utilisateur. Les démons sont souvent utilisés pour fournir des services ou des fonctionnalités système, et ils sont généralement initiés au démarrage du système d'exploitation.

#### 4.1.5 Image

Une image Docker est un ensemble immuable de fichiers, de configurations et de dépendances nécessaires pour exécuter une application. Ces images sont créées à partir d'un ensemble de directives appelées "Dockerfile", qui spécifient comment l'environnement de l'application doit être configuré.

#### 4.1.6 Conteneur

Un conteneur Docker est une instance en cours d'exécution d'une image. Il s'agit d'un processus léger qui encapsule l'application et ses dépendances, fonctionnant de manière isolée par rapport au système hôte et à d'autres conteneurs. Les conteneurs utilisent le noyau du système hôte et partagent certaines ressources tout en restant isolés les uns des autres.