# Tp Docker

- **Définition**
  Docker est une plateforme qui va vous permettre d'exécuter votre code à l'intérieur d'un conteneur indépendamment de la machine sur laquelle vous êtes ! Un conteneur ressemble à une machine virtuelle sauf qu'il n'embarque pas tout un système d'exploitation avec lui ce qui lui permet de s'exécuter en quelque secondes et d'être beaucoup plus léger.
- **Différence entre images et containers**

## Qu’est-ce qu’une image Docker?

Une image Docker est un fichier immuable, qui constitue une capture instantanée d’un conteneur. Généralement, les images sont créées avec la commande « docker build ». Et puis, ils vont produire un conteneur quand ils sont lancés avec la commande « run ». En revanche, dans un registre Docker, les images sont stockées comme « registry.hub.docker.com ». Comme elles peuvent devenir assez volumineuses, les images sont conçues pour composer des couches de d’autres images, ce qui permet d’envoyer une quantité minimale de données lors du transfert des images sur le réseau.

Les images sont stockées dans un registre Docker, tel que Docker Hub, et peuvent être téléchargées à l’aide de la commande docker pull:

```
$ docker pull ubuntu

Using default tag: latest

latest: Pulling from library/ubuntu

fdd5d7827f33: Pull complete

a3ed95caeb02: Pull complete

fd2731e4c50c: Pull complete

Digest:
sha256:45b23dee08af5e43a7fea6c4cf9c25ccf269ee113168c19722f87876677c5cb2

Status: Downloaded newer image for ubuntu:14.04
```

Pour afficher les images Docker téléchargées, exécutez « docker images »:

```
$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu 14.04 25fd21cfc6c0 3 weeks ago 125MB
```

## Qu’est-ce qu’un conteneur Docker?

Un conteneur Docker est une instance exécutable d’une image. En utilisant l’API ou la CLI de Docker, nous pouvons créer, démarrer, arrêter, déplacer ou supprimer un conteneur. De manière avantageuse, nous pouvons connecter un conteneur à un ou plusieurs réseaux, y attacher de la mémoire ou créer une nouvelle image sur la base de son état actuel. De plus, il consiste en une image Docker, un environnement d’exécution et un ensemble d’instructions standard.

Si on applique le concept de l’orienté objet. Si une image est une classe, un conteneur est une instance d’une classe, c’est-à-dire un objet d’exécution. Nous pouvons également dire que les conteneurs sont en quelque sorte la raison pour laquelle vous utilisez Docker car ils constituent des encapsulations légères et portables d’un environnement permettant d’exécuter des applications.

## Différenrence :

Ce que nous pouvons retenir, c’est que les images sont des instantanés figés de conteneurs vivants. Alors que les conteneurs exécutent les instances d’une image.

- **A quoi sert un Dockerfile**

  Un Dockerfile est un fichier texte décrivant les différentes étapes permettant de partir d'une base pour aller vers une application fonctionnelle. Docker lit les instructions que vous mettez dans le Dockerfile pour créer automatiquement l’image requise.

- **lancer votre premier container ubuntu, l'équivalent du *hello-world* de docker**

  ```
  $docker run hello-world
  ```

- **regarder si votre container est bien lancer**

```
$docker ps -a  # Visualiser les conteneurs actifs
```

- **faire un résumé type `cheat sheet` des principales commandes dockers relatives aux images et containers**

```
docker ps # Visualiser les conteneurs actifs
$ docker ps -a # Visualiser tous les conteneurs
$ docker rm [container] # Supprimer un conteneur inactif
$ docker rm -f [container] # Forcer la suppression d'un conteneur actif
$ docker images # Lister les images existantes
$ docker rmi [image] # Supprimer une image docker
$ docker exec -t -i [container] /bin/bash # Exécuter des commandes dans un conteneur actif
$ docker inspect [container] # Inspecter la configuration d'un conteneur
$ docker build -t [image] . # Construire une image à partir d'un Dockerfile
$ docker history [image] # Visualiser l'ensemble des couches d'une image
$ docker logs --tail 5 [container] # Visualiser les logs d'un conteneur (les 5 dernières lignes)

# Intéractions avec le registry
$ docker login # Se connecter au registry
$ docker search [name] # Rechercher une image
$ docker pull [image] # Récupérer une image
$ docker push [image] # Pouser une image du cache local au registry
$ docker tag [UUID] [image]:[tag] # Tagger une image
```

### Commandes Docker Compose

```
$ docker-compose up -d # Démarre un ensemble de conteneurs en arrière-plan
$ docker-compose down # Stoppe un ensemble de conteneurs
$ docker-compose exec [service] [command] # Exécute une commande au sein d'un service
```

- **Explication des commandes docker `build`, `run` et `exec`**

`$docker run` commande pour définir les ressources du conteneur lors de l'exécution.
`$docker build` commande pour crée des images Docker à partir d'un Dockerfile et d'un «contexte».
`$docker exec` commande pour exécuter une nouvelle commande dans un conteneur en cours d'exécution.

- **expliquer ce qu'est un port dans un container**

`$-p int:int` : cette option permet de partager le port de la machine avec le port du conteneur. Le premier nombre est le port de votre machine et le deuxième le port dans le conteneur.

- **vérifier que le container est activer**

  - ✓ Pour lancer les conteneurs qui constituent cette application, il suffit d'utiliser la commande : `$docker-compose up`, l'option `-d` peut être ajoutée si l'on souhaite que cette application fonctionne en mode détaché.

  - ✓ La commande `$Docker-compose ps` permet de vérifier les conteneurs en cours.
  - ✓ Pour stopper l'application et le cluster de conteneurs on utilise la commande `$docker-compose stop`
  - ✓ Enfin, pour supprimer les conteneurs et vérifier s'ils existent encore avec les commandes `$docker-compose down` et `$docker-compose ps`