Linux Professional Institute

Linux Professional Institute
DEVOPS TOOLS ENGINEER

Présentation de Docker

Formation Certifiante DevOps Tools Engineer

DevOps Tools Ingénieur | Linux Professional Institute

docker

La conteneurisation est une technique de virtualisation qui permet à plusieurs processus, appelés conteneurs, de fonctionner sur une machine hôte sans interférences.

Les conteneurs partagent un système d'exploitation, ils sont plus légers et démarrent plus rapidement que des machines virtuelles.

Cette approche permet d'accroître

- la flexibilité
- la portabilité d'exécution d'une application
- permet d'assurer le déploiement rapide et stable des applications dans n'importe quel environnement informatique.

La méthode Google

De Gmail à YouTube en passant par le moteur de recherche, tout fonctionne sous forme de conteneurs chez Google. La mise en conteneur permet à nos équipes de développement d'agir rapidement, de déployer efficacement des logiciels et d'œuvrer à une échelle sans précédent. Nous recréons chaque semaine plus de deux milliards de conteneurs.

Docker est un logiciel libre permettant facilement de

- empaqueter une application et ses dépendances dans un **conteneur isolé**
- exécuté sur n'importe quel serveur

Images, Dockerfiles, et conteneurs

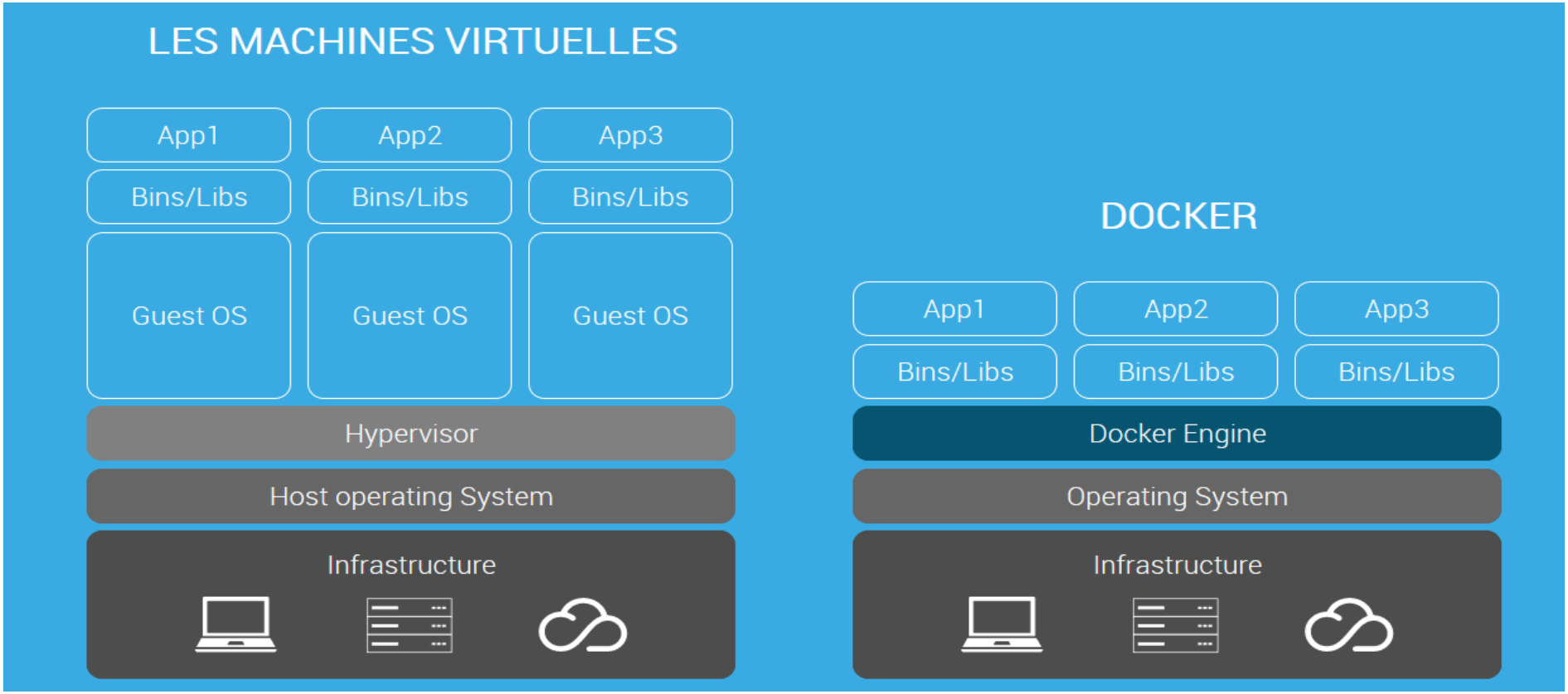
- Votre **système d'exploitation** est majoritairement composé de
  - un système de fichiers
  - des processus.
- **Une image Docker** représente le système de fichiers, sans les processus, Elle contient tout (Java, une base de donnée, un script que vous allez lancer, etc…)
- Les images sont créées à partir de fichiers de configuration, nommés "**Dockerfile**"
- **Un conteneur** est l'exécution d'une image : il possède la copie du système de fichiers de l'image, ainsi que la capacité de lancer des processus.
- **Dans ce conteneur**, vous allez donc pouvoir interagir avec les applications installées dans l'image, exécuter des scripts et faire tourner un serveur, etc

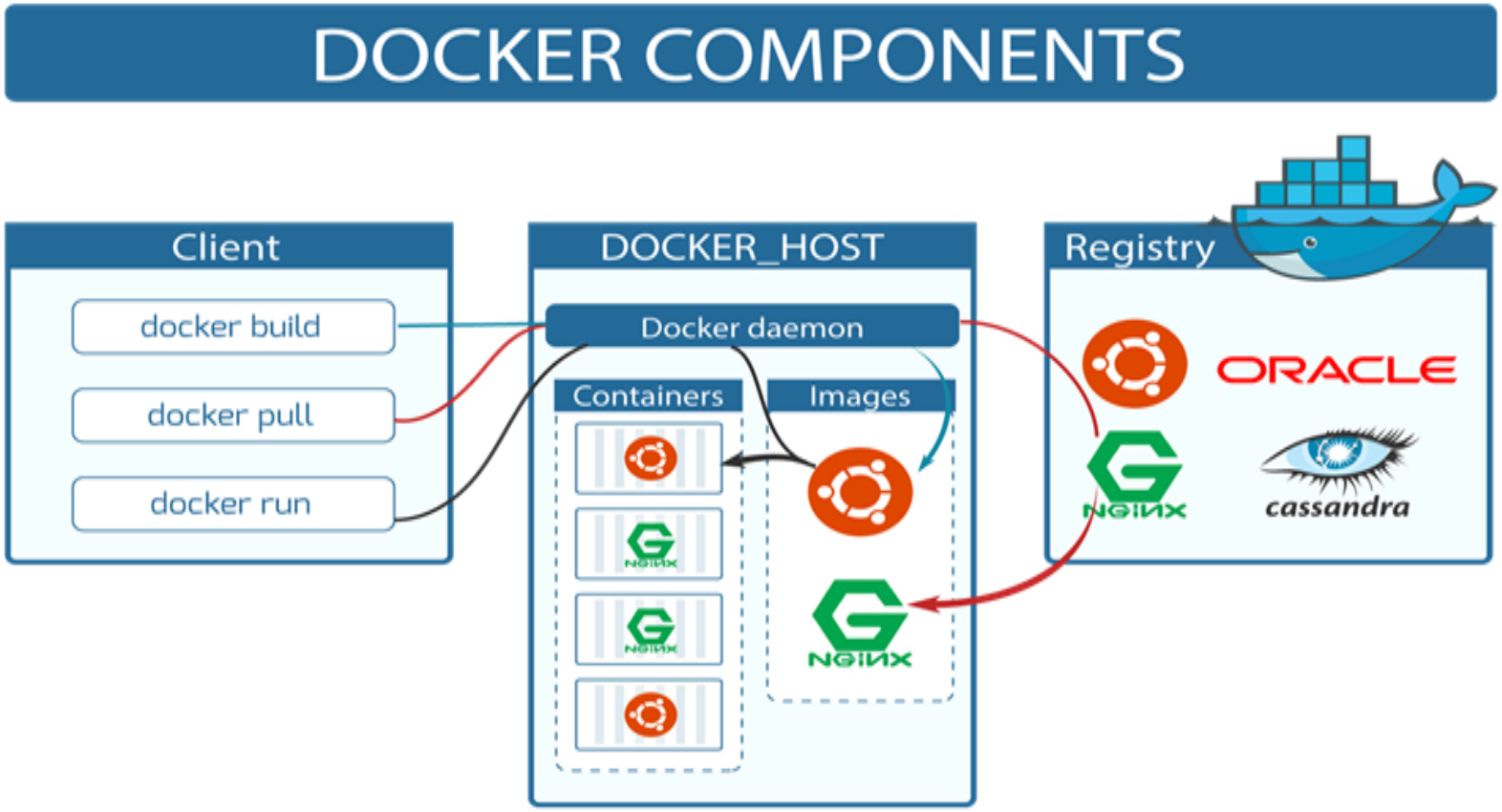
Contrairement aux machines virtuelles traditionnelles, **un conteneur Docker n'inclut pas de système d'exploitation** fournies par l'infrastructure sous-jacente il va simplement utiliser un **noyau Linux.**

Dans un serveur virtualisé type, chaque VM « invitée » contient **un système d'exploitation complet, avec ses pilotes, fichiers binaires ou bibliothèques**, ainsi que l'application elle-même. Chaque VM s'exécute alors sur un hyperviseur, qui s'exécute à son tour sur un système d'exploitation hôte, qui lui-même fait fonctionner le matériel du serveur physique.

En isolant les conteneurs les uns des autres, la conteneurisation assure la sécurité des applications et empêche la prolifération de logiciels malveillants entre les instances

Docker. La virtualisation



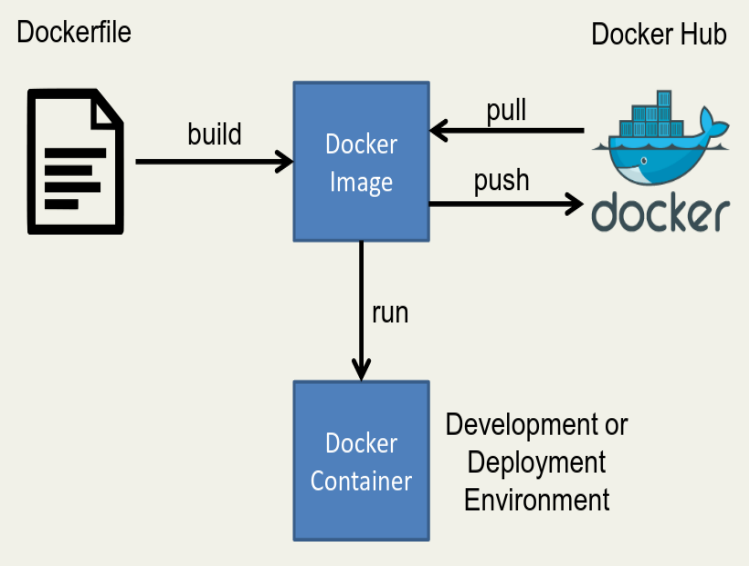

DOCKER COMPONENTS
Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
ORACLE
NGINX
cassandra

Installation de Docker sous Linux

https://docs.docker.com/install/linux/docker-ce/centos/

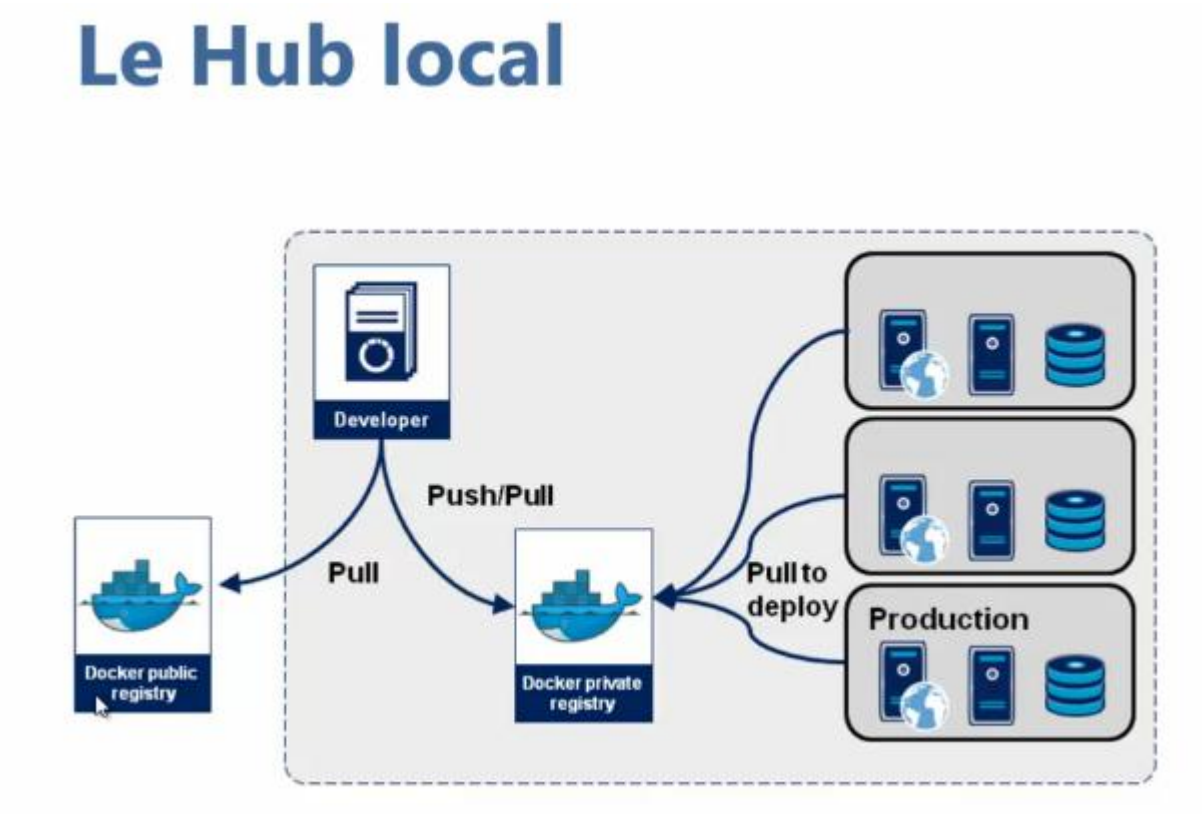
Docker Hub

```
docker login --username=fekiayman
Password:

docker tag hello-world:latest fekiayman/formation:v0.1
docker push fekiayman/formation:v0.1
```



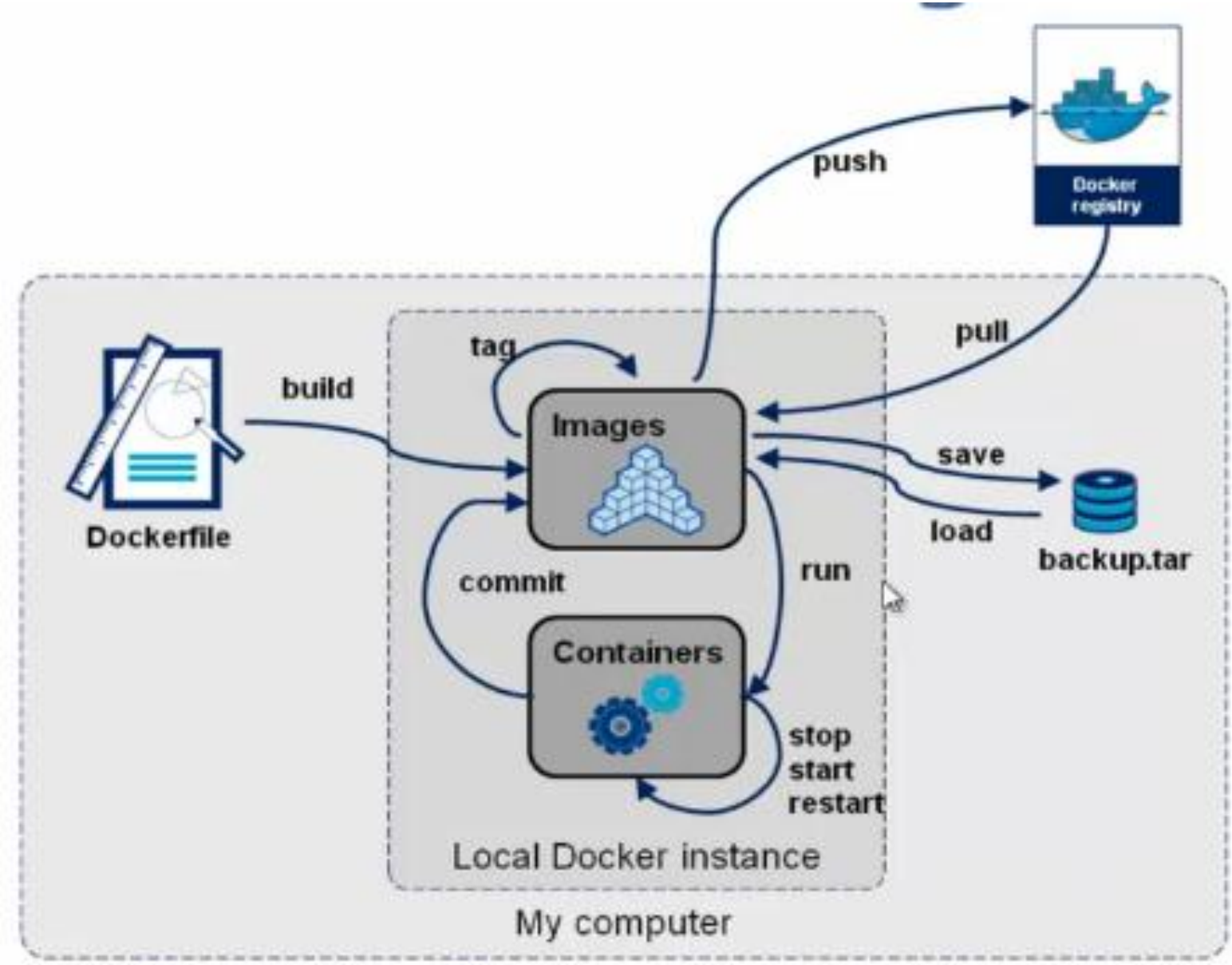
Administrer les images

```
docker search nginx |more
docker image
docker rmi
docker inspect nginx
docker history nginx

docker save fekiayman/formation:v0.1 -o /tmp/formation.0.1.tar
ll /tmp/formation.0.1.tar
docker rmi fce289e99eb9 –force
docker load -i /tmp/formation.0.1.tar
```



docker search registry |more

docker run -d -p 5000:5000 --restart=always --name=registry registry:2
docker tag nginx:latest localhost:5000/nginx_local
docker push localhost:5000/nginx_local


Docker registry
push
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer

docker run centos
docker run centos echo "Bienvenue à tous "

---

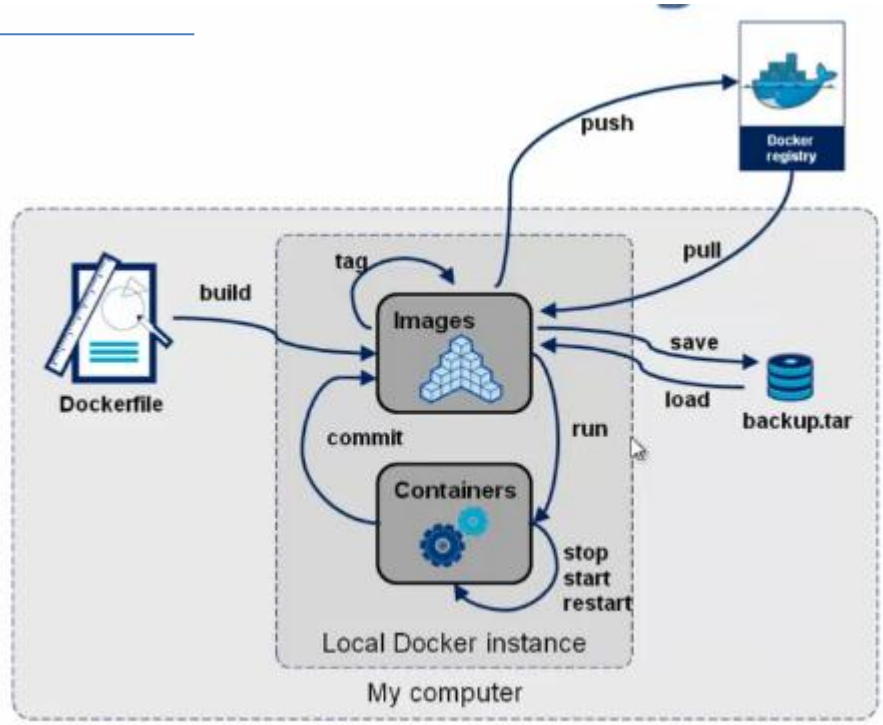
docker run -it centos /bin/bash
mkdir test
Exit

---

docker run -it centos /bin/bash
touch doc.txt
Ctrl pq

---

docker attach 793f591a1be0
ls
Exit



docker exec e4ba96bc4c5d touch /root/file_exec

---

docker start 793f591a1be0

---

docker run –help

---

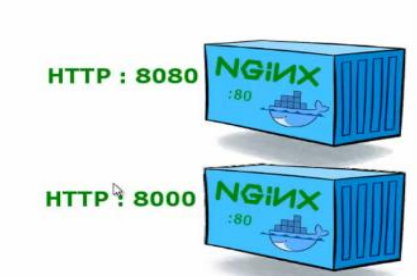
docker run -d --name web_server centos /bin/bash -c "yum install -y httpd"
docker attach

docker run -d -p 8080:80 --name webNginx nginx

ip add

http://192.168.0.102:8080/

Welcome to nginx!



docker exec webNginx /bin/bash -c 'echo "<H1> Mon Nginx <H1>" >/usr/share/nginx/html/index.html'

Start /stop

Container → Image

docker **commit** 5aca607b4bfe new_nginx

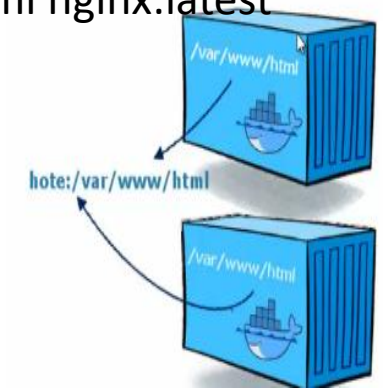
---

docker volume create my-vol

docker volume ls

docker volume inspect my-vol

docker run –d -p 8006:80 --name=nginxtest -v my-vol:/usr/share/nginx/html nginx:latest



mkdir data

cd data

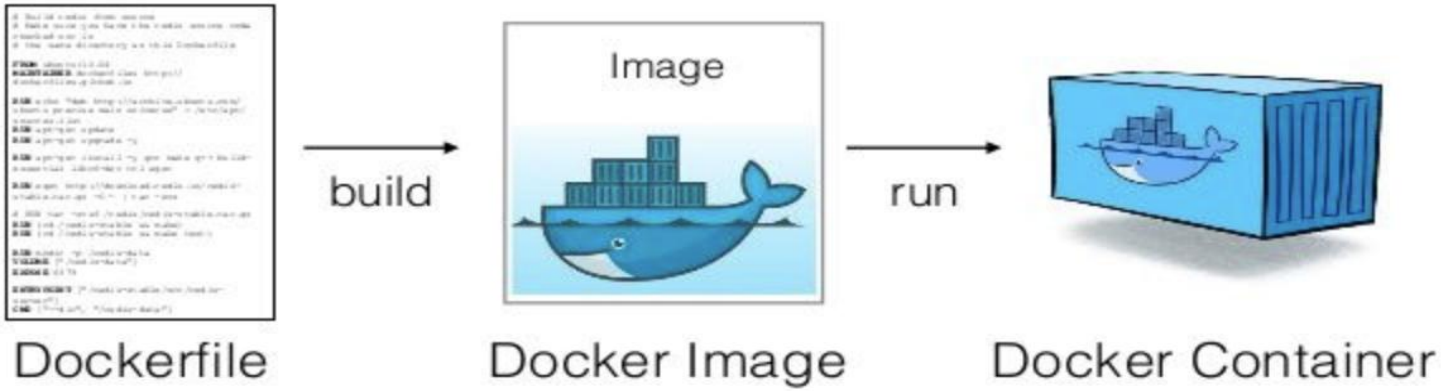
touch index.html

vi index.html

echo '<html><body>Hey! c bon</body></html>' > /usr/data/index.html

docker run -d -p 8006:80 --name webV3 **-v** /usr/data:/usr/share/nginx/html/ nginx

Les **Dockerfiles** sont des fichiers qui permettent de construire une image Docker adaptée à nos besoins, étape par étape.



Instructions Dockerfile

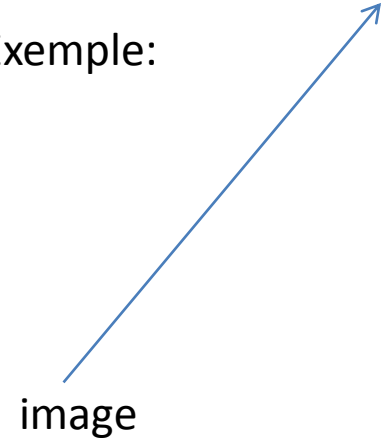
Ordre	Instruction	Description
1	FROM	Image parente
2	MAINTAINER	Auteur
3	ARG	Variables passées comme paramètres à la construction de l'image
4	ENV	Variable d'environnement
4	LABEL	Ajout de métadonnées
5	VOLUME	Crée un point de montage
6	RUN	Commande(s) utilisée(s) pour construire l'image
6	COPY	Ajoute un fichier dans l'image
6	WORKDIR	Permet de changer le chemin courant
7	EXPOSE	Port(s) écouté(s) par le conteneur
8	USER	Nom d'utilisateur ou UID à utiliser
9	ONBUILD	Instructions exécutées lors de la construction d'images enfants
10	CMD	Exécuter une commande au démarrage du conteneur
10	ENTRYPOINT	Exécuter une commande au démarrage du conteneur

Exemple Dockerfile

Fichier :Dockerfile

docker build -t fullstack-js .

Exemple:



image

```
# Image de base
FROM debian:jessie

# Installation de curl avec apt-get
RUN apt-get update \
&& apt-get install -y curl \
&& rm -rf /var/lib/apt/lists/*

# Installation de Node.js à partir du site officiel
RUN curl -LO "https://nodejs.org/dist/v0.12.5/node-v0.12.5-linux-x64.tar.gz"
\
&& tar -xzf node-v0.12.5-linux-x64.tar.gz -C /usr/local --strip-components=1 \
&& rm node-v0.12.5-linux-x64.tar.gz

# Ajout du fichier de dépendances package.json
ADD package.json  /app/

# Changement du repertoire courant
WORKDIR /app

# Installation des dépendances
RUN npm install

# Ajout des sources
ADD . /app/

# On expose le port 3000
EXPOSE 3000

# On partage un dossier de log
VOLUME . /app/log

# On lance le serveur quand on démarre le conteneur
CMD node server.js
```