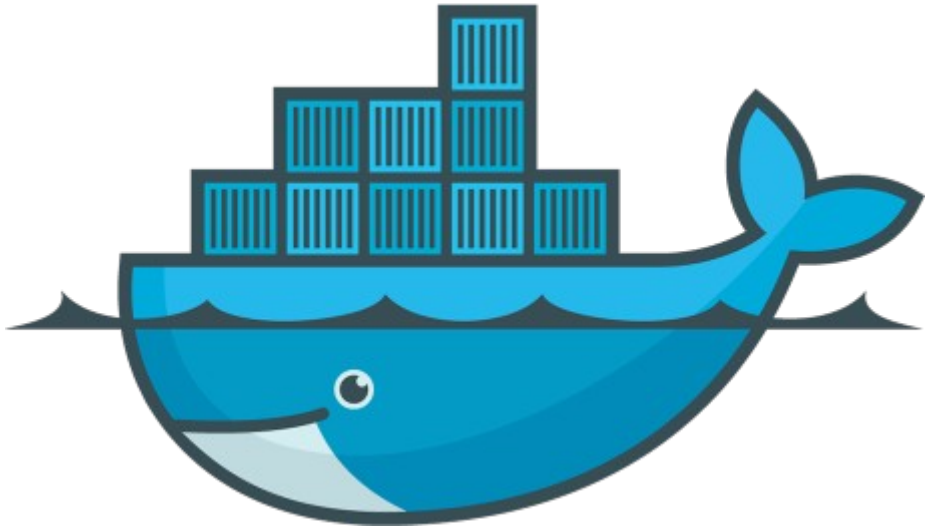
DAWAN Paris 11,rue Antoine Bourdelle, 75015 PARIS
DAWAN Nantes 32, Bd Vincent Gâche, 5e étage - 44200 NANTES
DAWAN Lyon Bt Banque Rhône Alpes, 2ème étage – 235 cours Lafayette 69006 LYON

Dawan
Training is our way



Formation Docker:

Virtualisation, Haute Disponibilité, Sécurité, Conteneur

Plus d'info sur http://www.dawan.fr ou 0810.001.917

Formateur: Pierre Sablé

Présentation de la formation

Qui suis-je? « mon parcours » « mes compétences »

Exprimer votre besoin:

- Le contexte **IT**
- Votre environnement de travail
- Vos **objectifs** pour cette formation.

Objectifs

Comprendre la virtualisation et ses concepts

La philosophie **Docker**

Installation et configuration de Docker

Configurations avancées / docker file / docker compose

Avec ces acquis vous serez en mesure de :

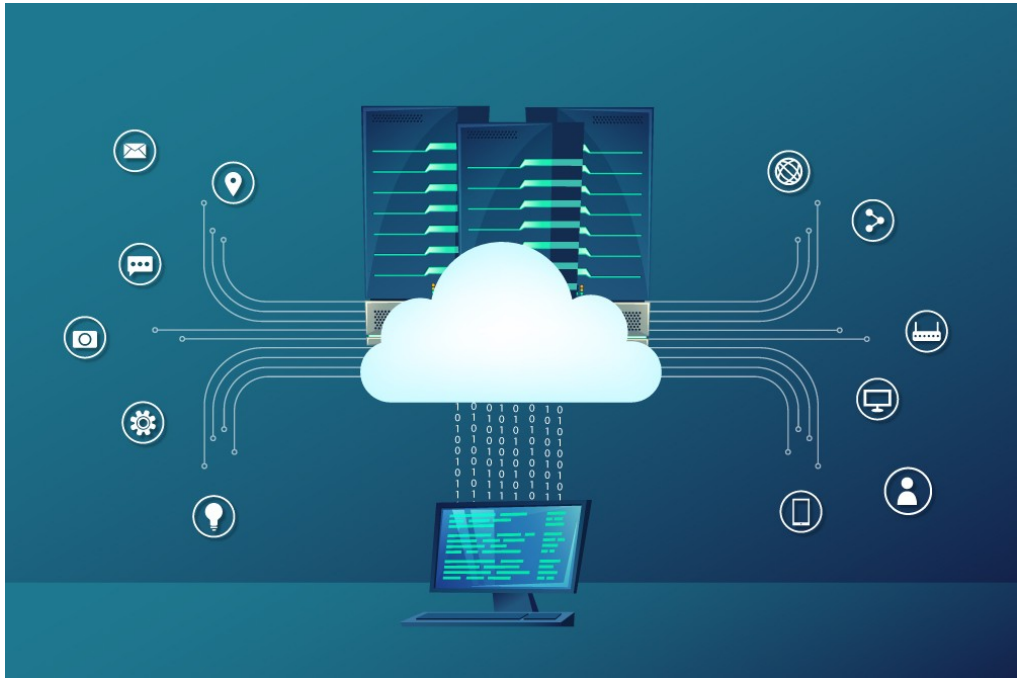
- Installer et configurer une plateforme Docker
- S’adapter facilement à des contextes différents

Que vous évoque la virtualisation?

Qu’est ce qu’un conteneur?

Quel est le rôle de **Dockerfile**?

Que désigne le terme **DevOps** ?



Introduction

Enjeux de la Virtualisation

Gain énergétique

Gain QOS

Une flexibilité accrue

Cibler, au mieux, les besoins de votre parc informatique

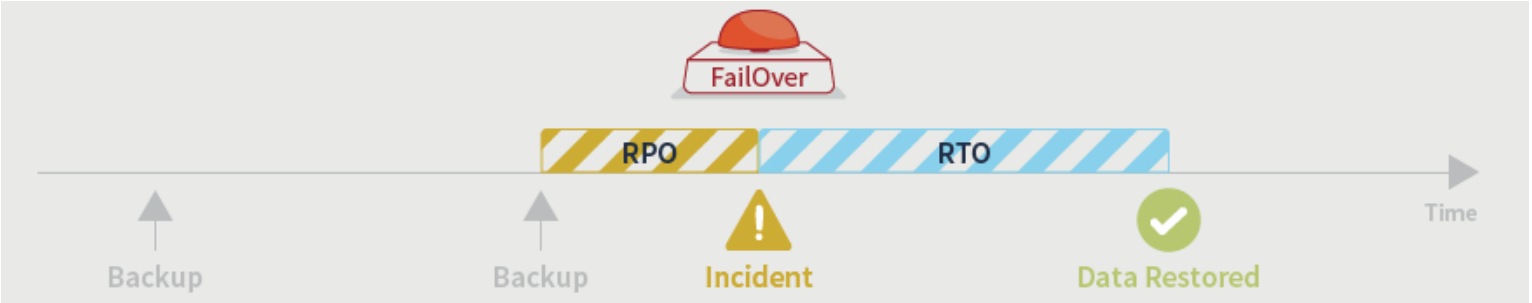
PRA - RTO/RPO

PRA: Plan de relance d'activité

RTO: Durée maximale d'interruption admissible

RPO: Durée maximum d'enregistrement des données qu'il est acceptable de perdre lors d'une panne.

Ex: sauvegarde toutes les 24H

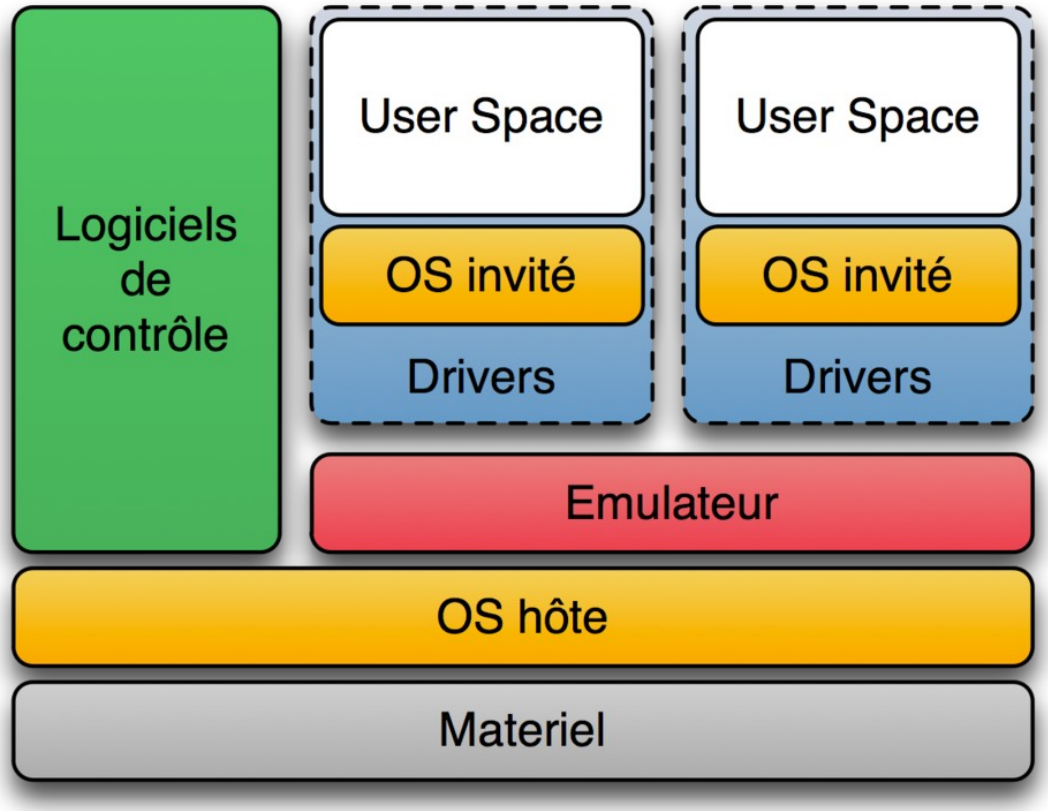


Ces facteurs permettrons de cibler les technologies en adéquation avec les contraintes internes au projet.

Les différents types de virtualisation

Hyperviseur niveau 2

Fonctionnement:

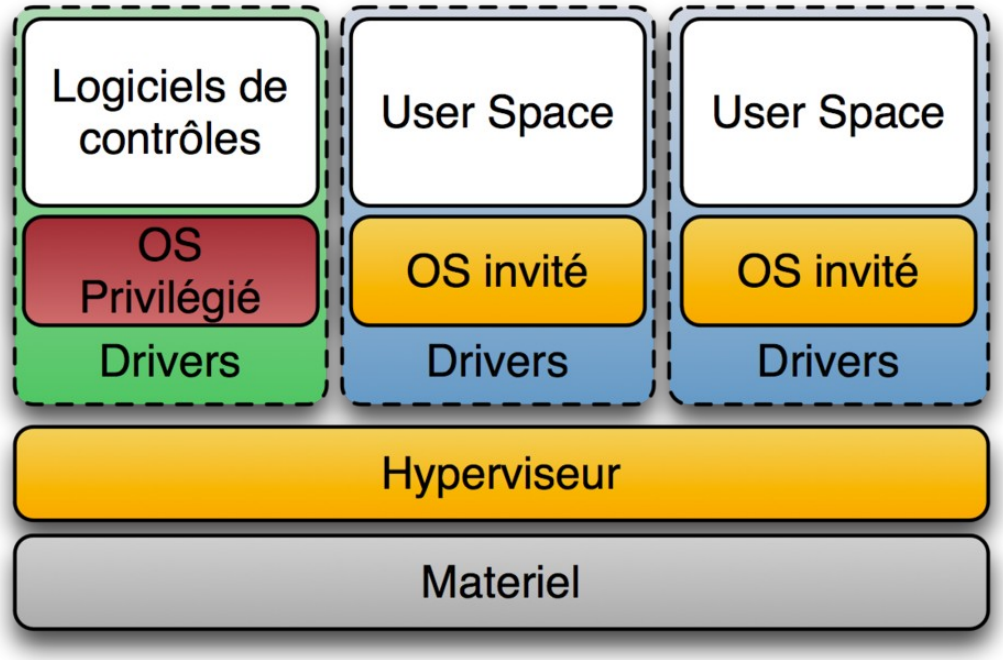


Avantage/Défaut:

- Très peu performant
- Limité en terme d'infrastructure
+ Facile d'installation

Hyperviseur niveau 1

Fonctionnement:

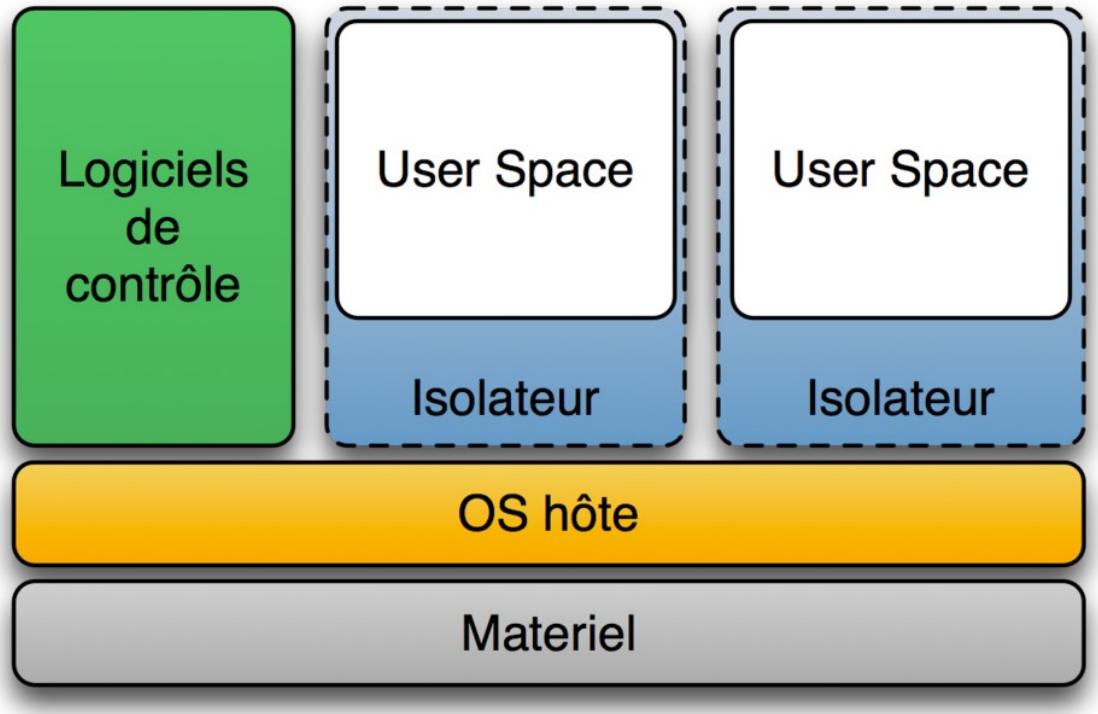


Avantage/Défaut:

- Plus complexe à mettre en œuvre
+ Performant
+ Modulable

Isolateur

Fonctionnement:



Avantage/Défaut:

- Lié au système hôte

\+ Performant (1~3% de Pénalité)

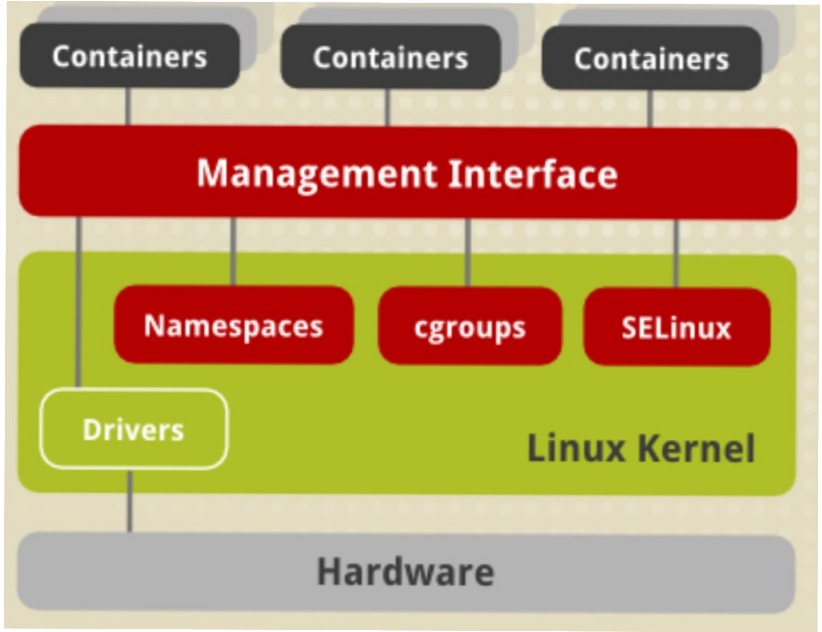
Exemple de solutions:

- OpenVZ (Isolateur d'OS)
- Chroot (Isolateur de racine)
- Linux VSERVER & BSD Jail (Isolateur d'espace utilisateur)

Les containers Linux :

Concept :

- **Un processus !**
- **Isolé des autres processus**
- **Avec sa propre vision du système sur lequel il tourne (Namespace)**
- **Limité dans les ressources qu'il peut utiliser (Control Groups)**
- **Partage le Kernel de la machine hôte avec les autres containers**



Les containers Linux : Namespaces

- Technologie Linux pour isoler un processus

- Les namespaces limitent ce qu'un container peut voir

- Différents namespaces :

- **pid : isolation de l'espace de processus**
- **net : donne une stack réseau privée**
- **mount : système de fichiers privés**
- **uts : nom du hosts**
- **ipc : isole les communications inter processus**
- **user : pamming des UID/GID entre l'hôte et les containers**

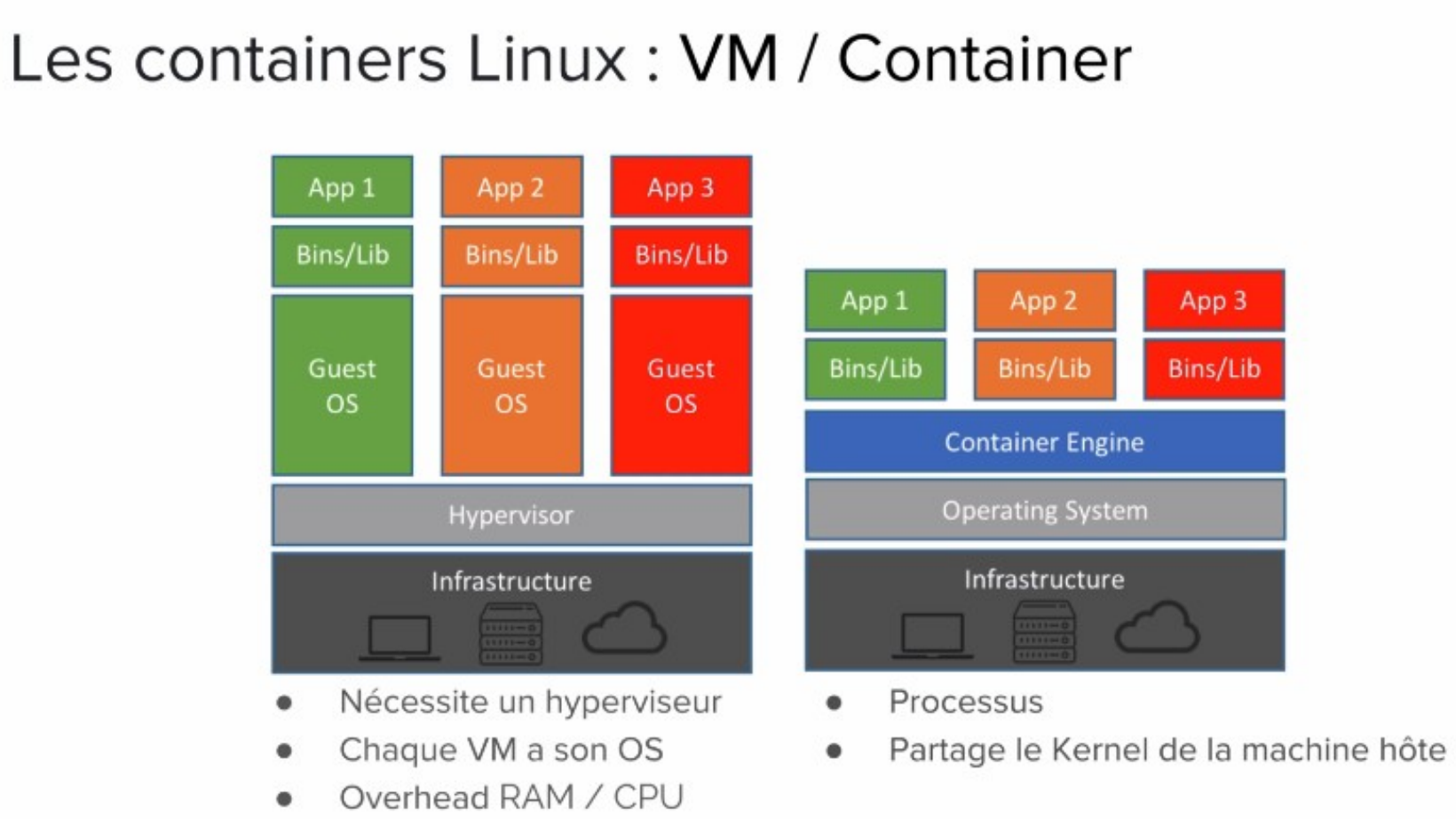
Les containers Linux : Control Groups

- Technologie Linux (cgroups)

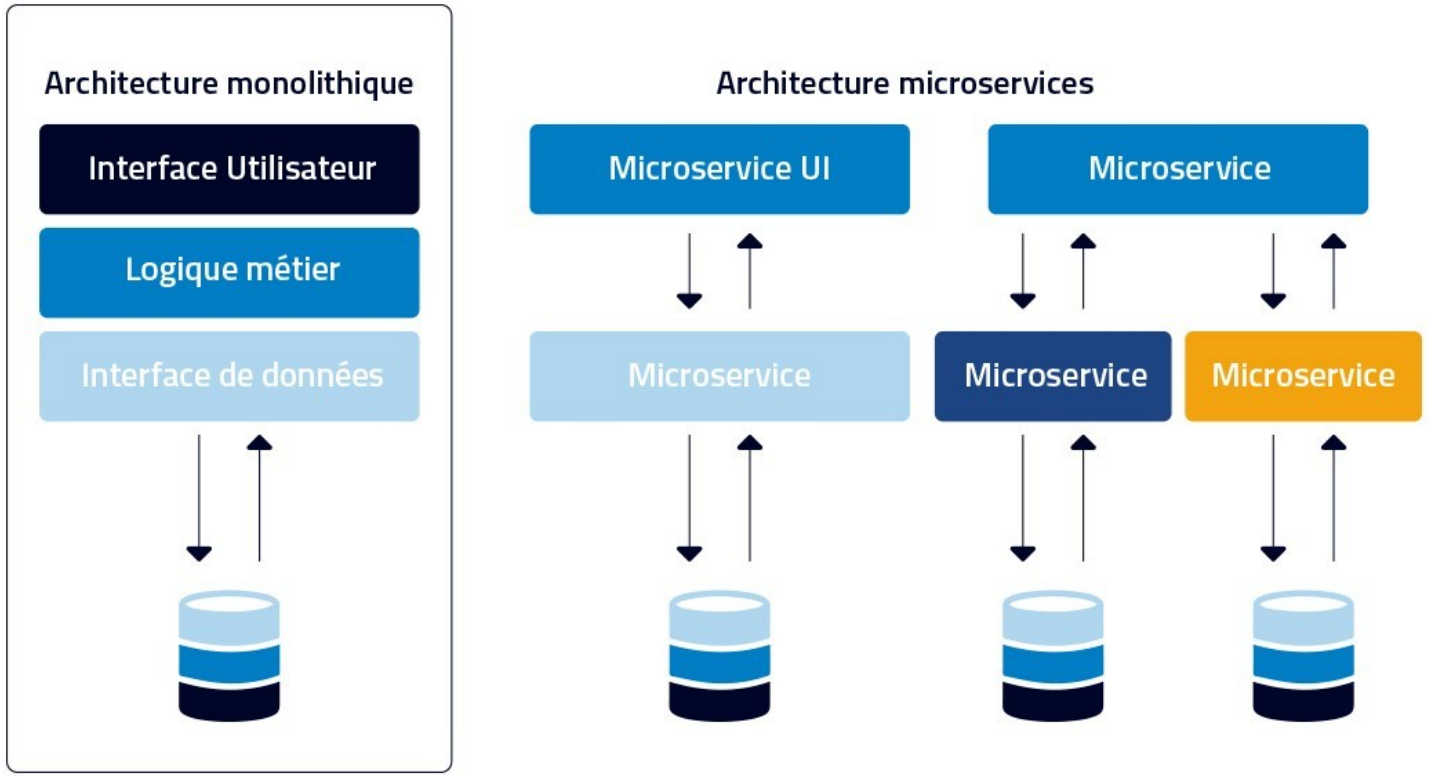
- Limite les ressources qu'un processus peut utiliser :

- **RAM**
- **CPU**
- **I/O**
- **Network**

Les containers Linux : VM / Container



Architecture micro-services



Architecture micro-services

Pros :

- **Découpage de l'application en processus (services) indépendants**
- **Chacun a sa propre responsabilité métier**
- **Equipe dédiée pour chaque service**
- **Plus de liberté de choix dans le langage**
- **Màj et scaling horizontal**
  - **Network**

Cons :

- **Nécessite des interfaces bien définies**
- **Focus sur les tests d'intégrations**
- **Déplace la complexité dans l'orchestration de l'appication globale**

Application Cloud Native

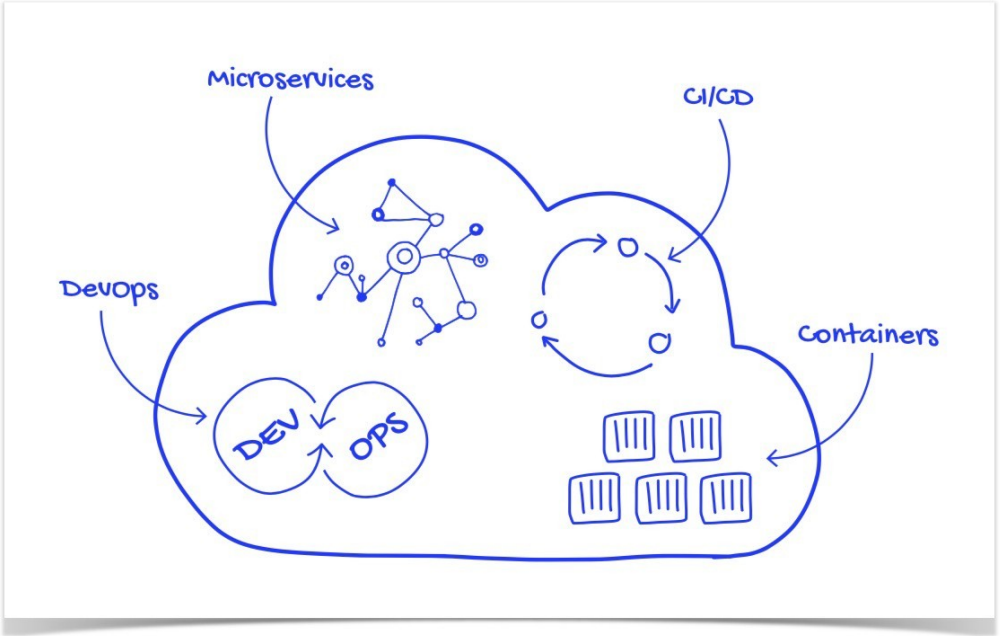
- **Application orientée microservice**

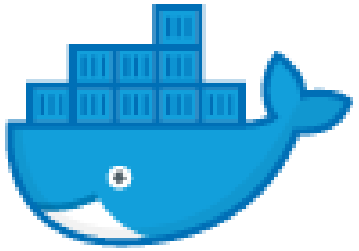
- **Packagée dans des container**

- **Orchestration dynamique**

- **cncf.io**

- **Nombreux projets portés par la CNCF**
  (Cloud Native Computing Foundation)
  Kubernetes
  Prometheus





docker : c’est quoi ?

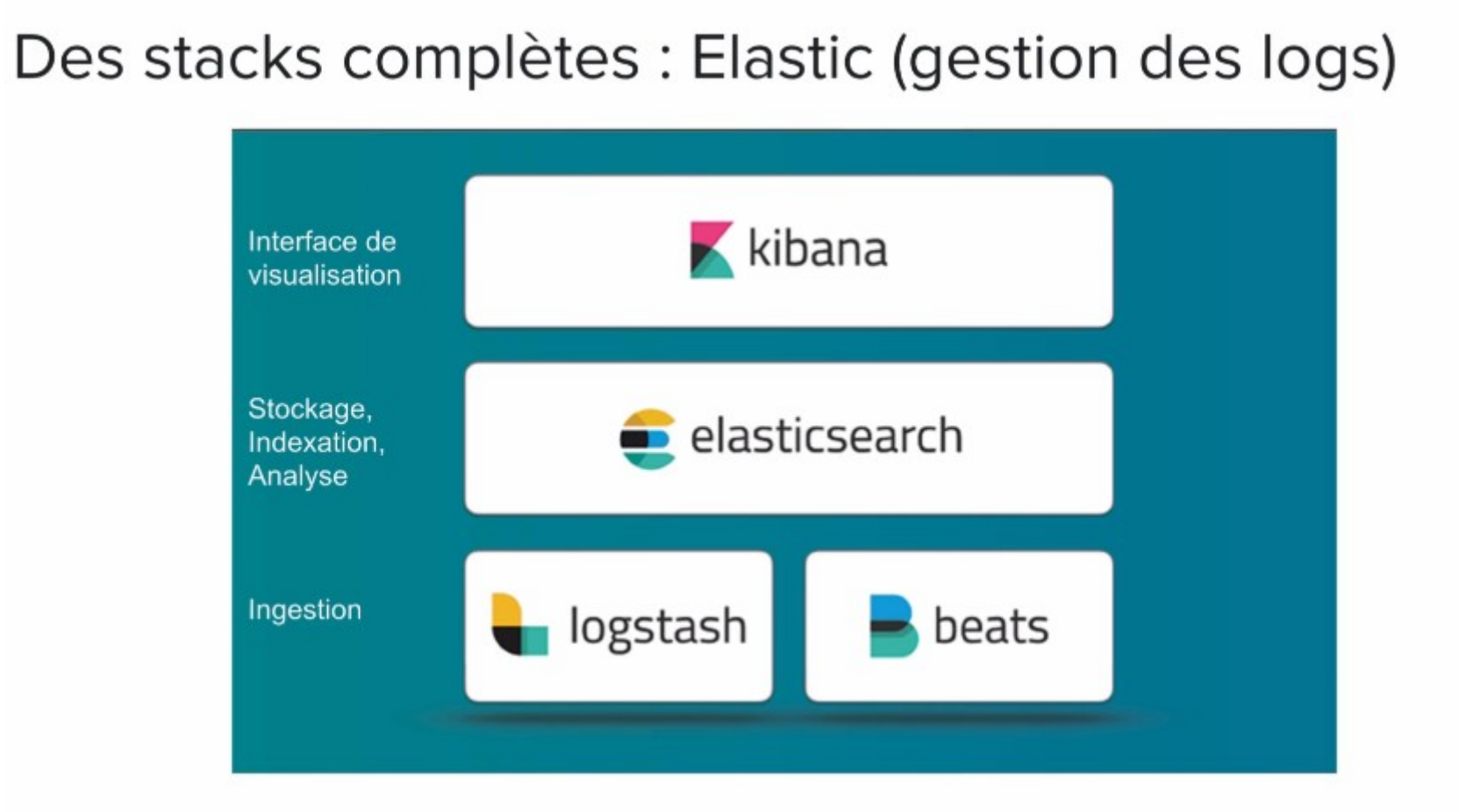
Ce que Docker permet de faire !

De nombreux logiciels

- Packagés dans des **images**
- Disponibles dans le Docker Hub
- Utilisables immédiatement

https://hub.docker.com

Ce que Docker permet de faire !



Introduction

Distribué en tant que projet open source à partir de **mars 2013**

Langage de programmation : Go

Docker CE : 19.03.3

https://docs.docker.com/engine/release-notes/

https://github.com/docker/docker/blob/master/CHANGELOG.md

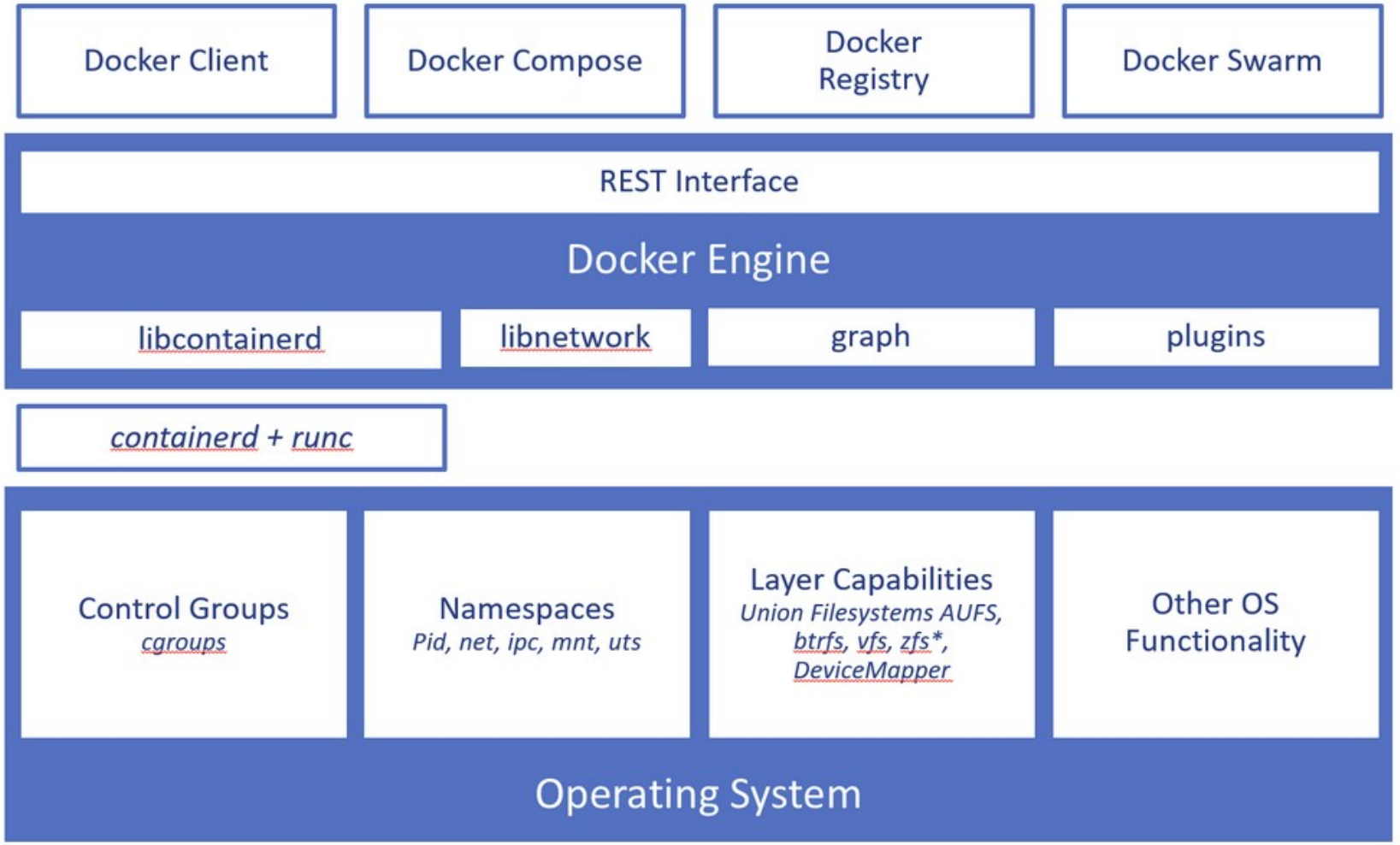
Concurrence:

- LXC
- OpenVZ
- Mesos
- Podman
- CRI-O

Solution payante:

- Docker Cloud & Docker entreprise

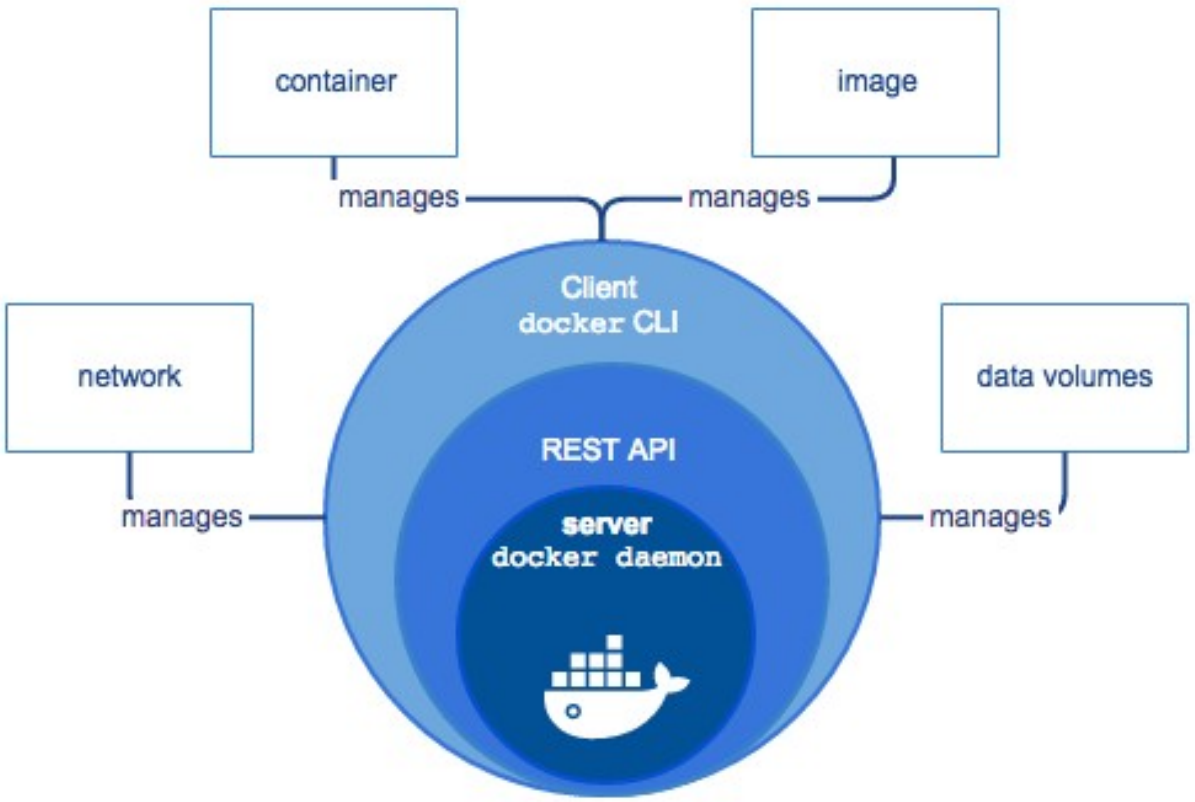
Docker : l'architecture



Docker : l'environnement

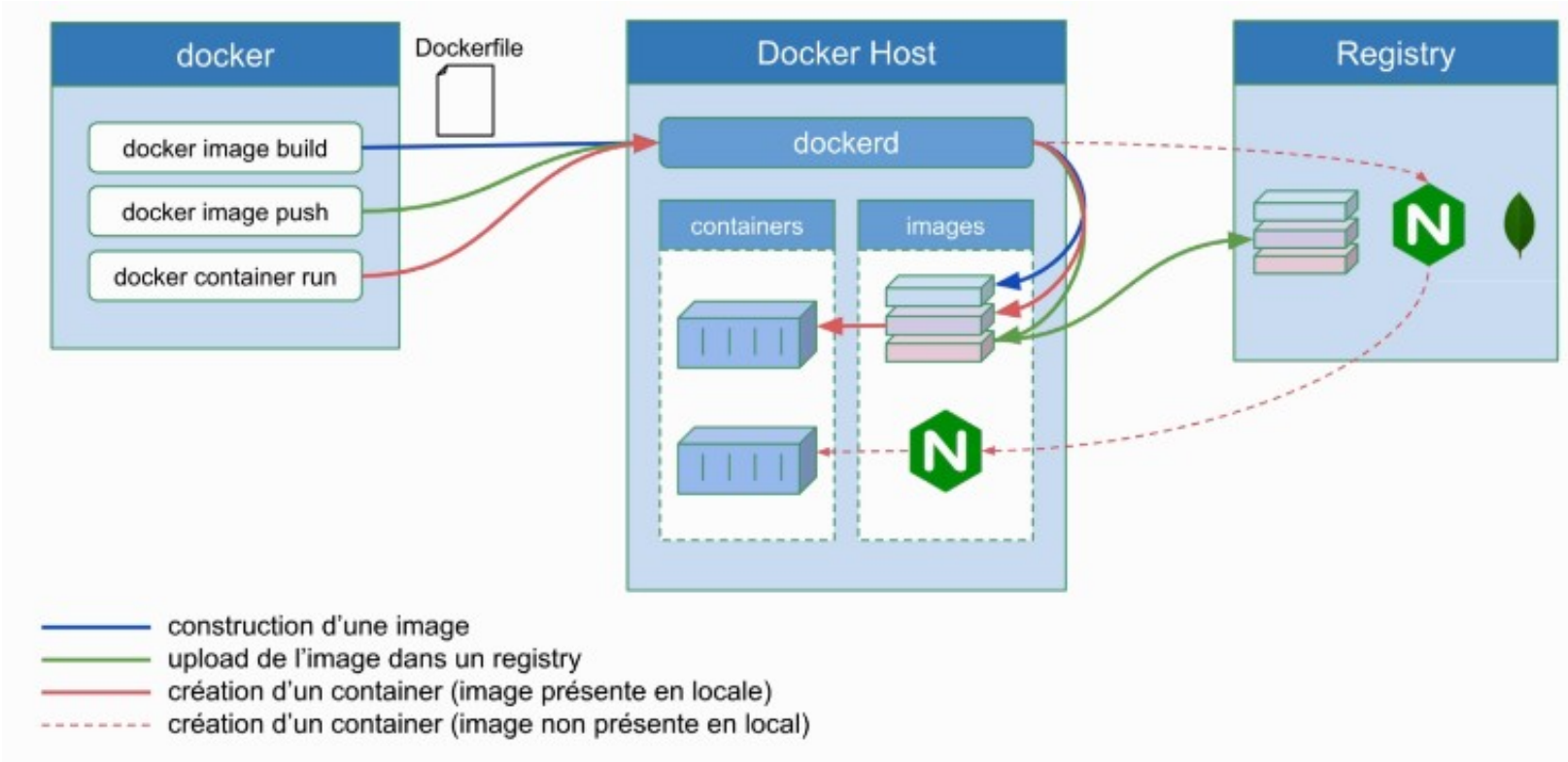
Les briques :

- **conteneurs**
- **images**
- **réseaux**
- **volumes**

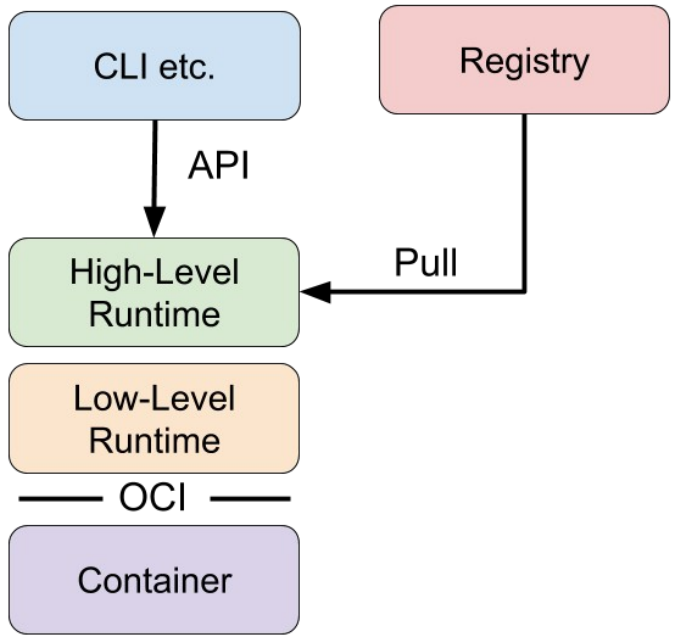


Les concepts essentiels

Docker facilite l'utilisation des **images** et des **containers** Linux



Docker : détails



Daemon: Dockerd (API REST)

- haut niveau
- image build

Containerd (projet CNCF) :

- Image push and pull
- Managing of storage
- Executing of Containers by calling runc with the right parameters
- Managing of network primitives for interfaces
- Management of network namespaces containers to join existing namespaces

containerd-shim : daemonless containers

OCI Runtime: Runc (standard) → créer un conteneur
https://github.com/opencontainers/runtime-spec

dockerd --> **containerd** --> **containerd-shim --> docker-runc**

Docker : Les évolutions

- **OCI runtime** : runc

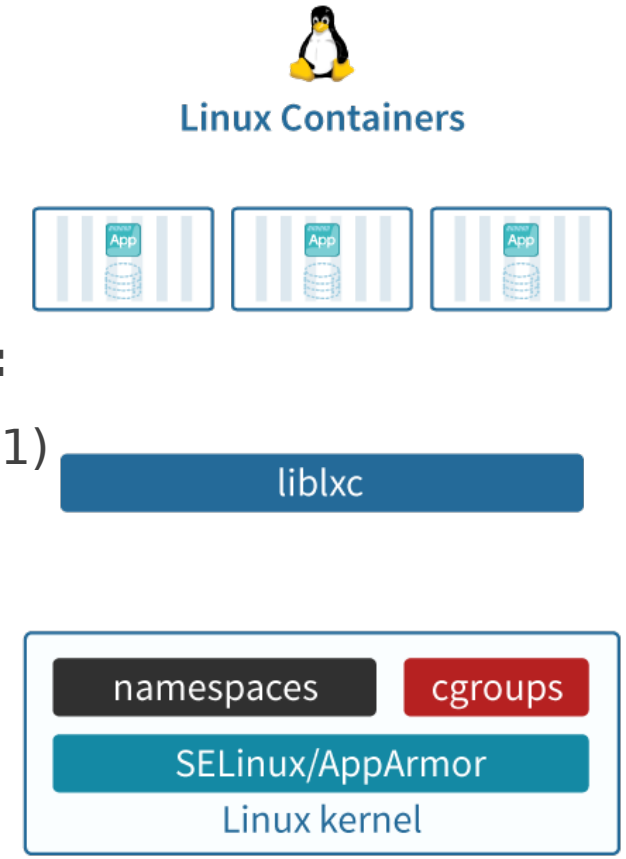
- **Open Container Initiative(OCI)**

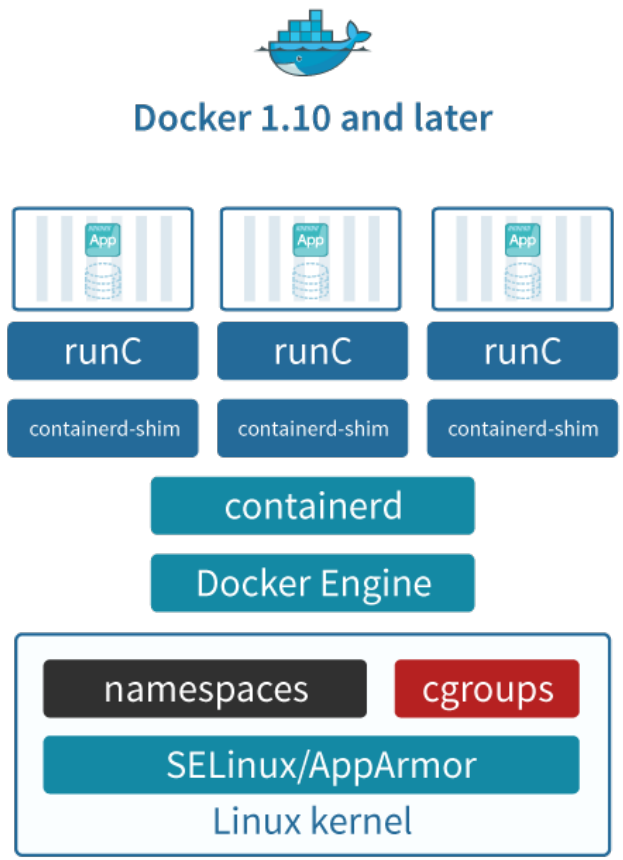
- **Container runtime :**

  Containerd(Docker 1.11)

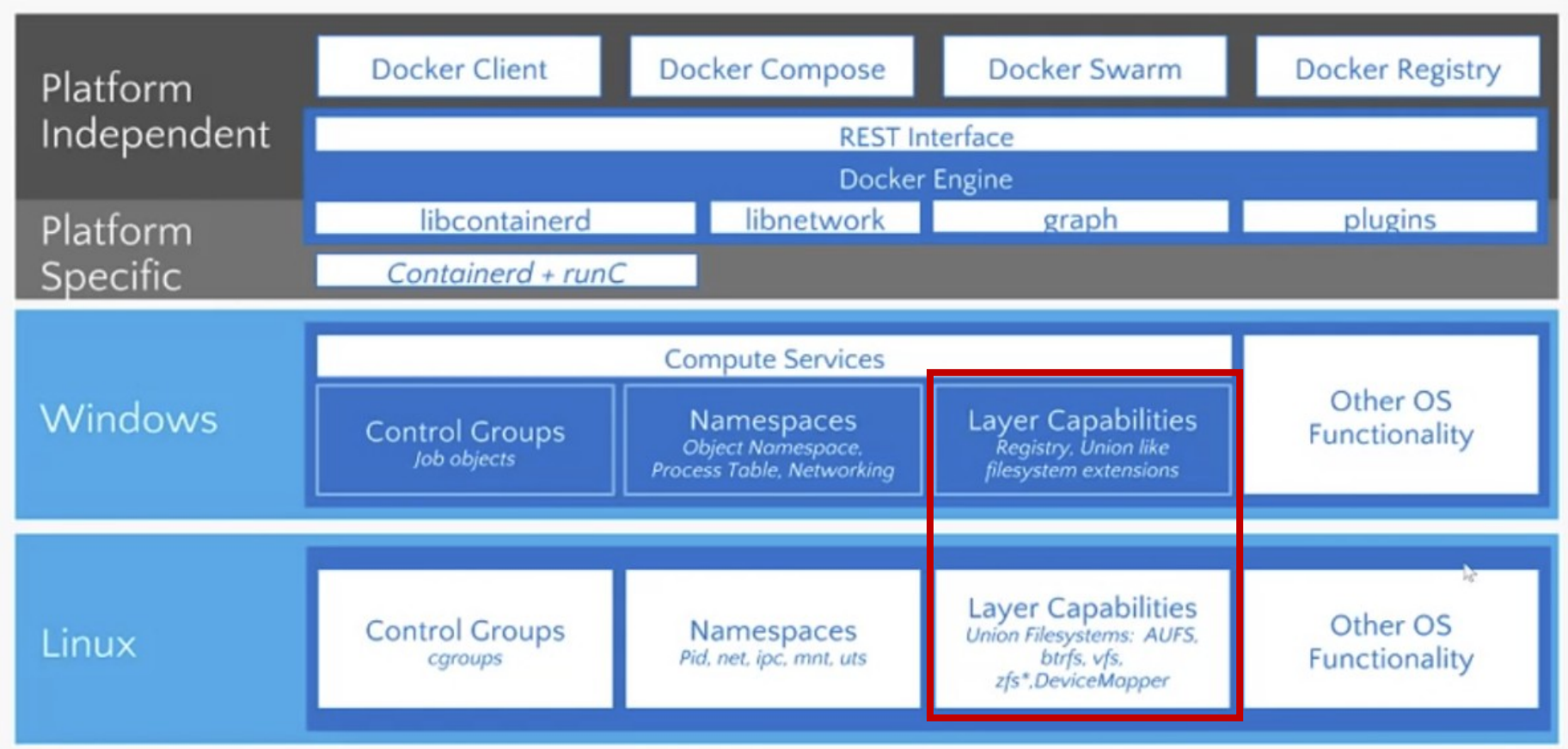
- **File System :**

  aufs => overlayfs





Plateformes



Windows

Hyper-V images:

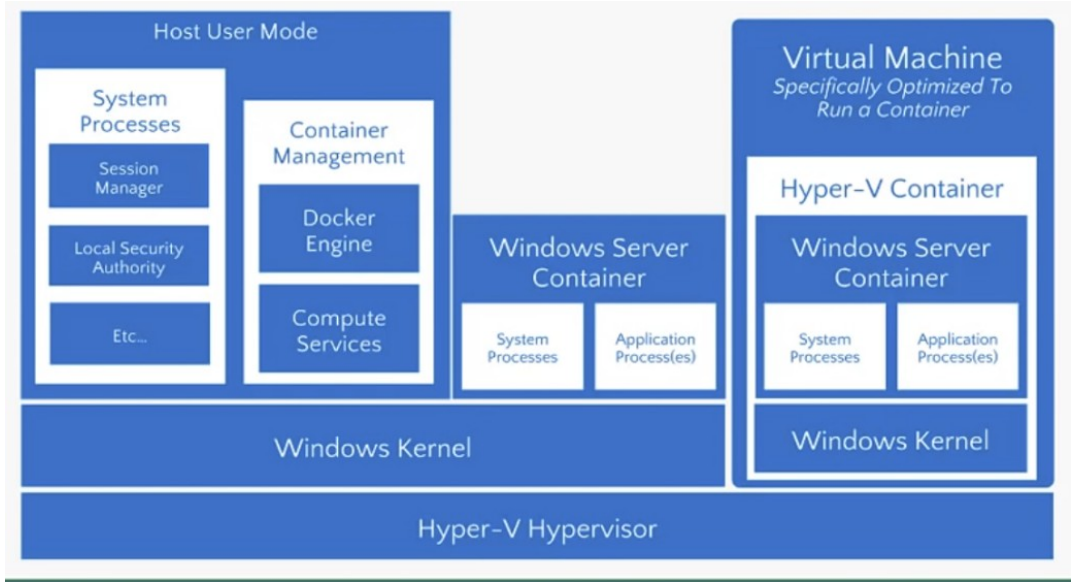
- Nanoserver

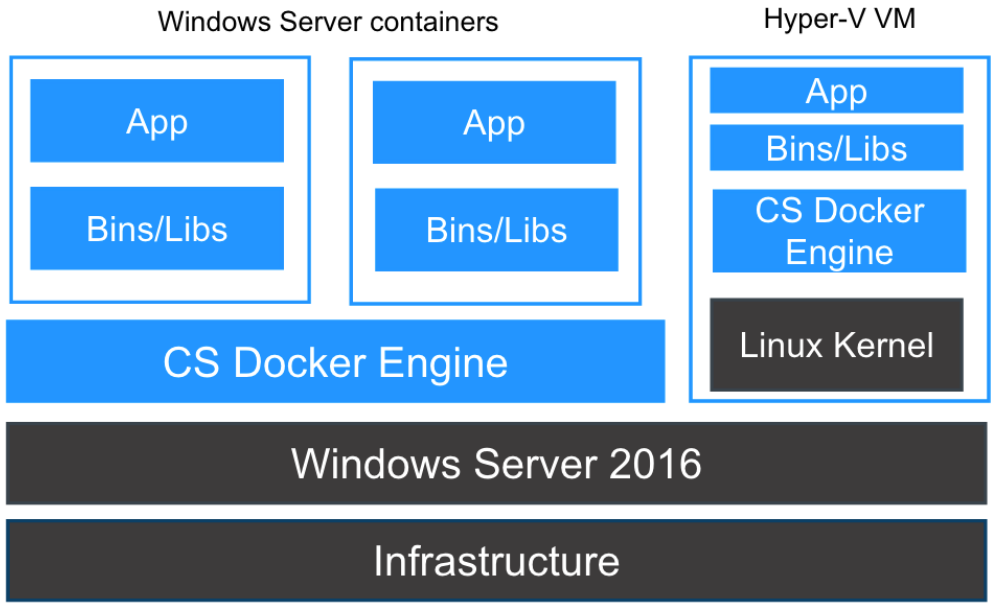
Minimal Windows Server

- Windows ServerCore

VM Linux :

- Linux containers





TP - Déploiement

Linux

Docker CLI

Connecté à un terminal, utilisation de l'invite de commande:

User@NomDeLaMachine~$

~ Dossier courant

$ normal

root

Exécution de la commande docker:

User@NomDeLaMachine~$ docker paramètres

Exemples:

User@NomDeLaMachine~$ docker -t

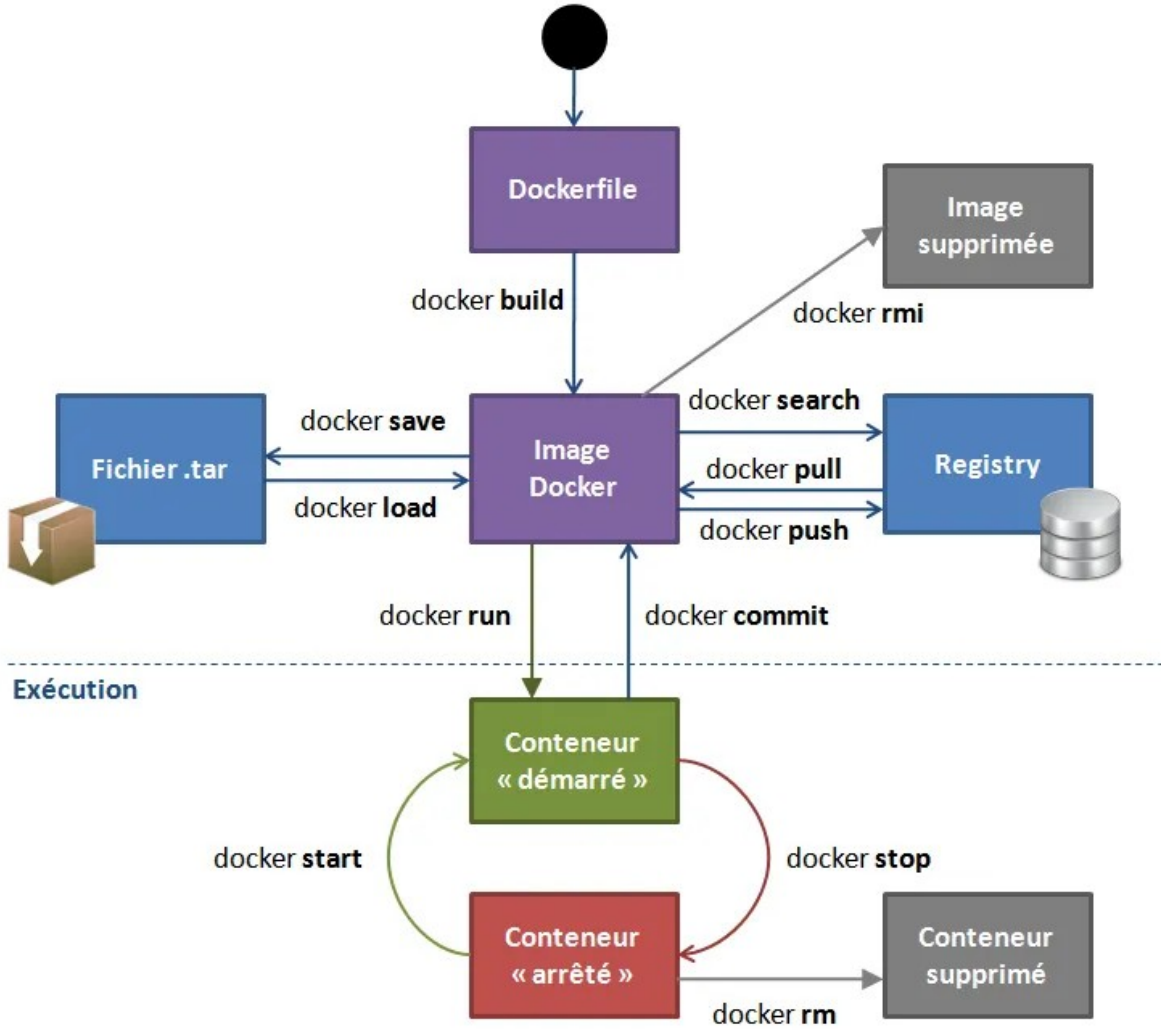
User@NomDeLaMachine~$ docker -t -i

User@NomDeLaMachine~$ docker -ti

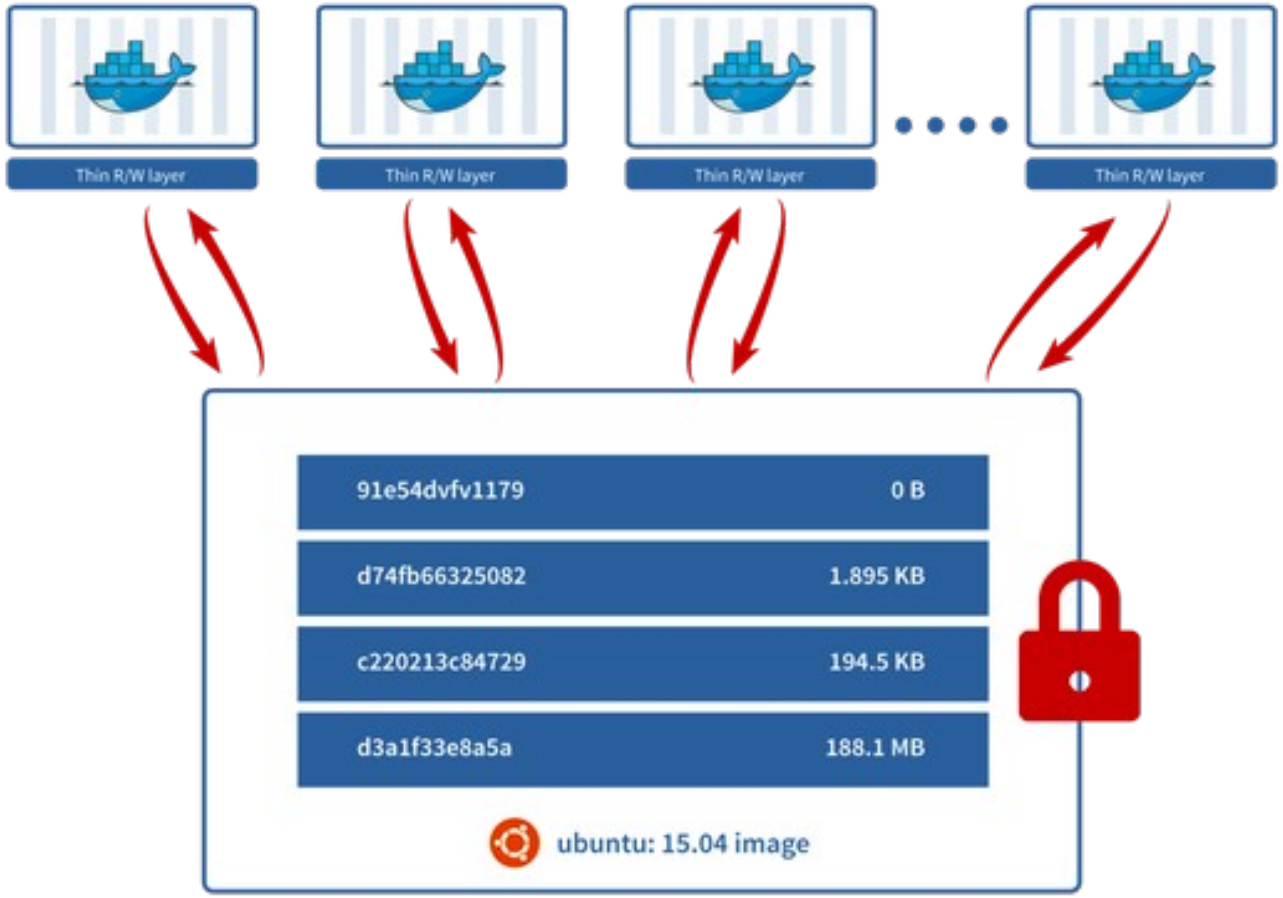
TP – Premier pas

Docker Hub – docker search – PULL & RUN

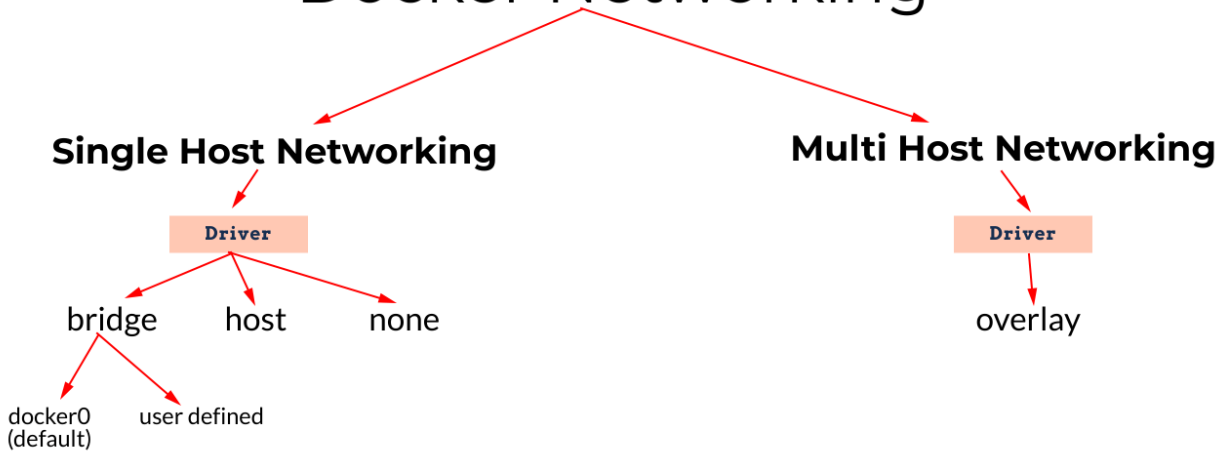
Les images


Dockerfile
docker build
Image supprimée
docker rmi
Fichier .tar
docker save
docker load
Image Docker
docker search
docker pull
docker push
Registry
docker run
docker commit
Exécution
Conteneur « démarré »
docker start
docker stop
Conteneur « arrêté »
Conteneur supprimé
docker rm

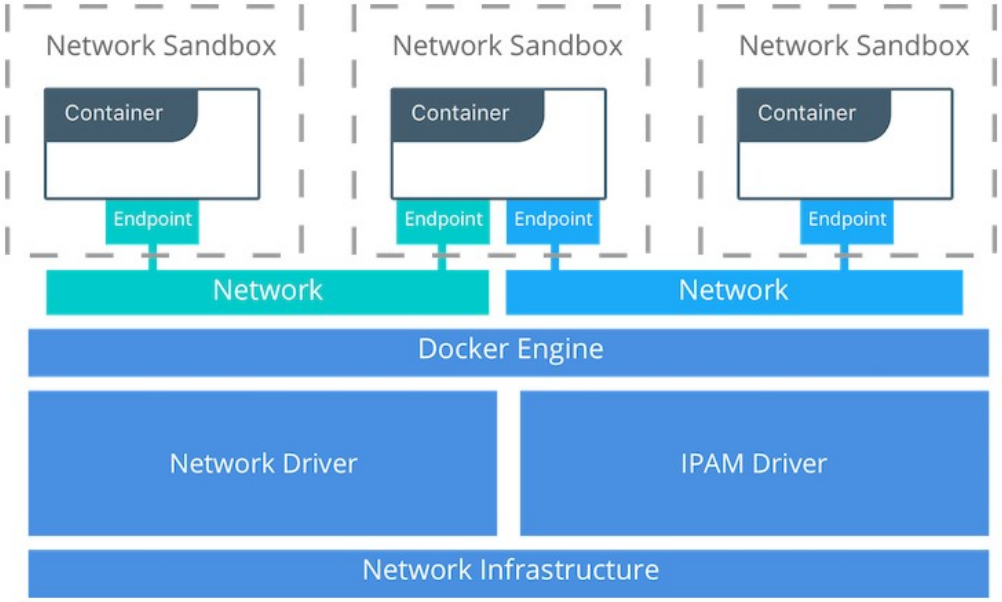
Les images

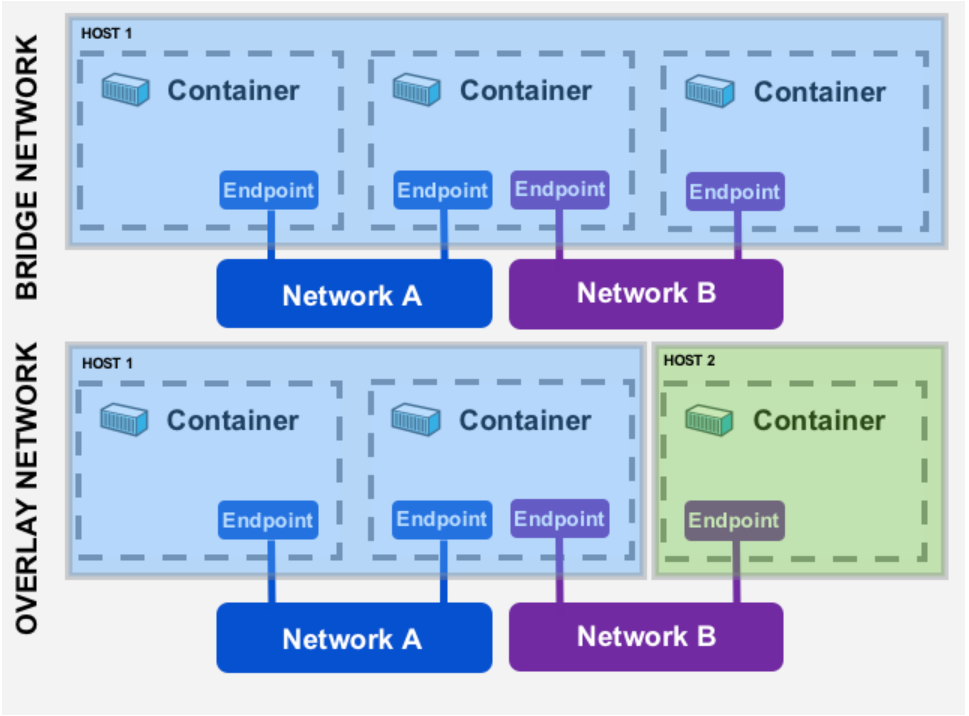

Thin R/W layer
Thin R/W layer
Thin R/W layer
Thin R/W layer
91e54dvfv1179 0 B
d74fb66325082 1.895 KB
c220213c84729 194.5 KB
d3a1f33e8a5a 188.1 MB
ubuntu: 15.04 image

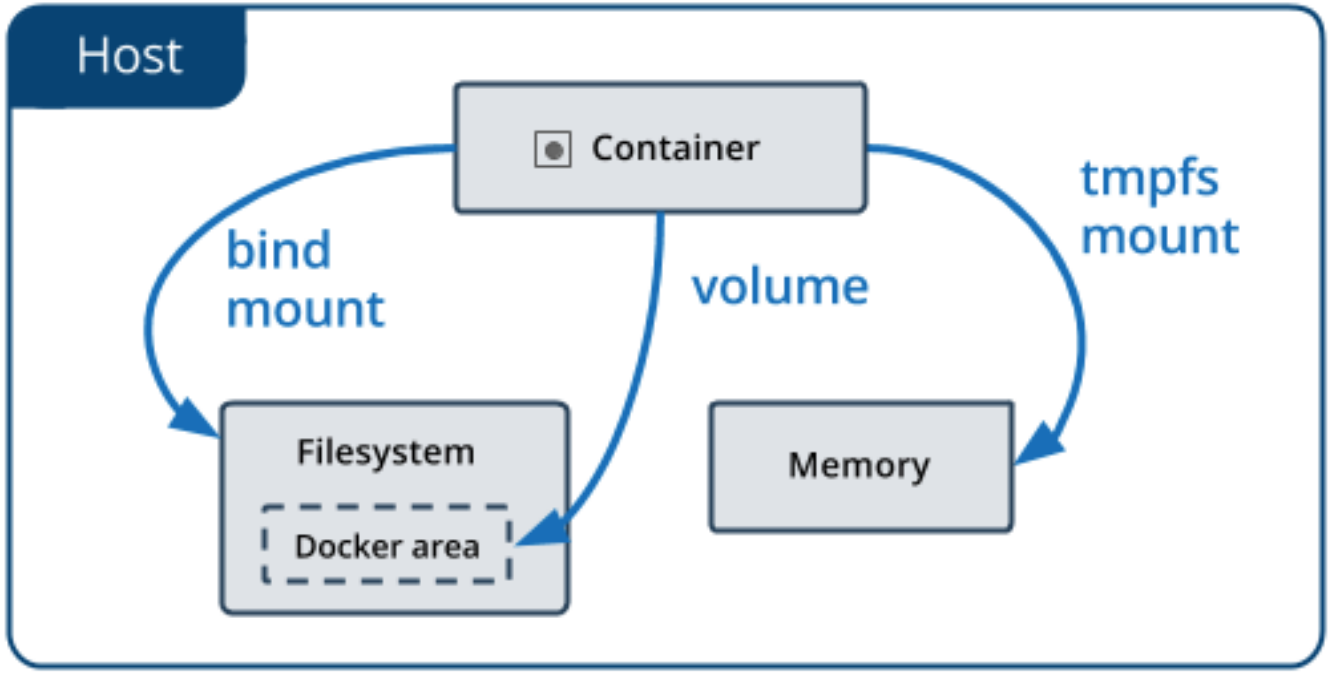
Docker Networking



$ docker network --help

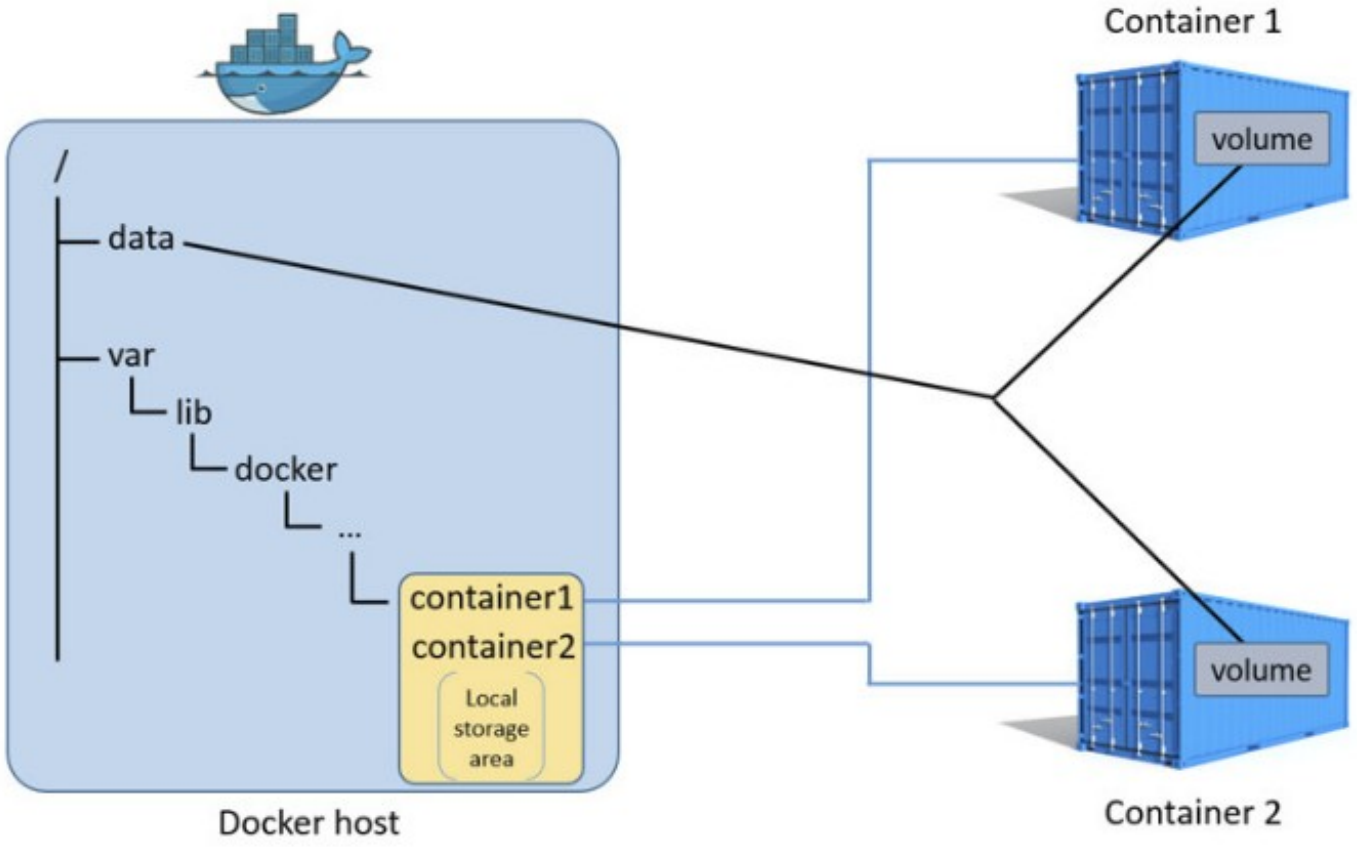






Docker – Le stockage

Structure


Container 1
volume
/
data
var
lib
docker
...
container1
container2
Local storage area
Docker host
volume
Container 2

Files Systems - Définition

Le stockage peut-être :

- Locale
- SAN(iSCSI) https://github.com/gluster/gluster-containers
- Distribué
- Dans la RAM
- SSHFS

```
{
  "storage-driver": "overlay2",
  "insecure-registries" : [ "172.30.0.0/16" ]
}
```

Technology	Storage driver name
OverlayFS	overlay or overlay2
AUFS	aufs
Btrfs	btrfs
Device Mapper	devicemapper
VFS	vfs
ZFS	zfs

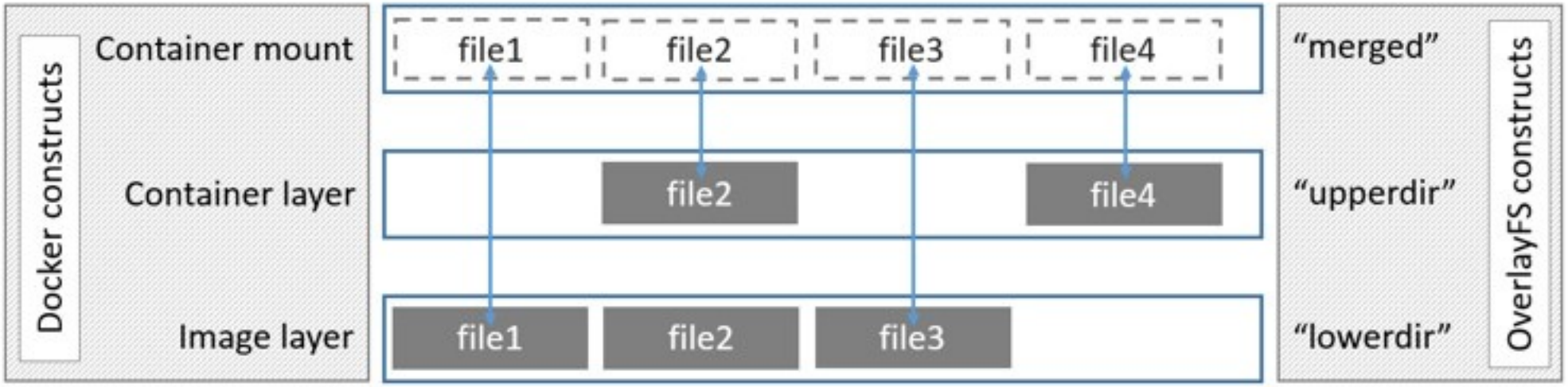
OverlayFS

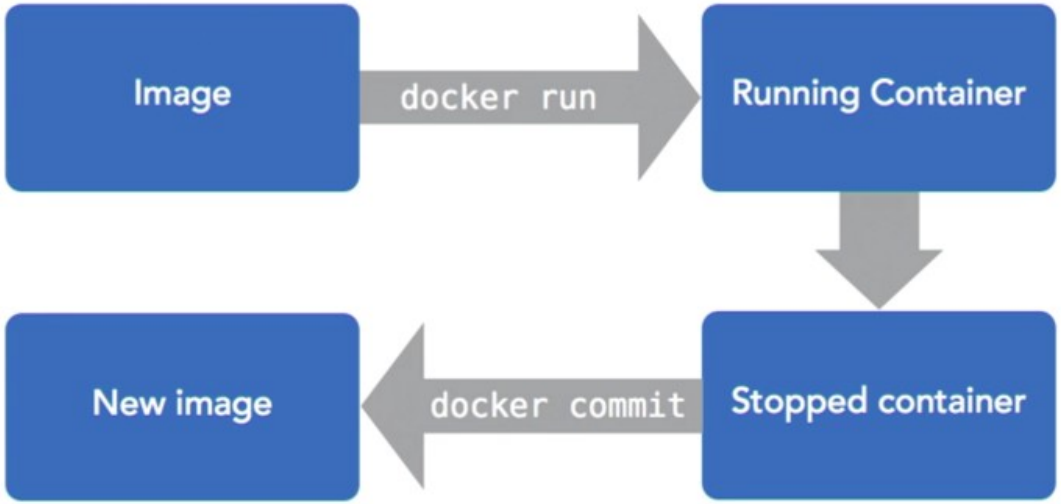
[size | virtual size]

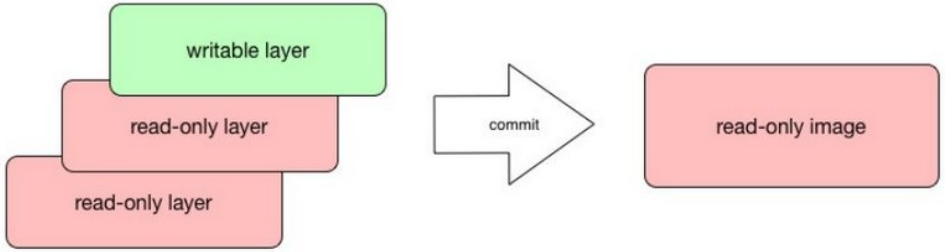
$ docker ps -s

$ docker ps --format "table {{.Size}}"

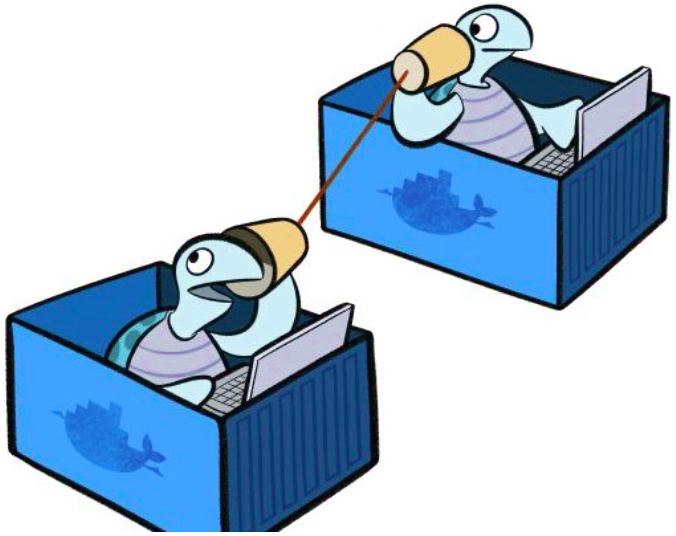
Copy on Write(CoW)





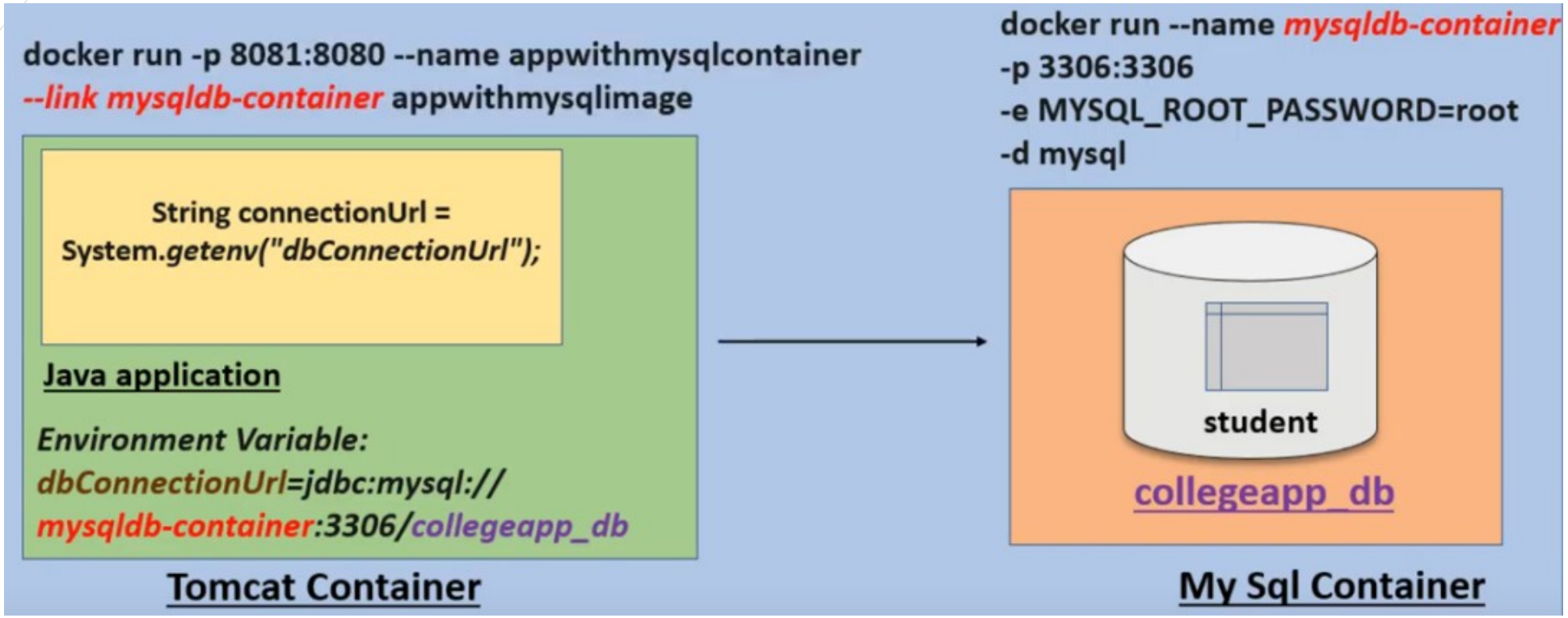


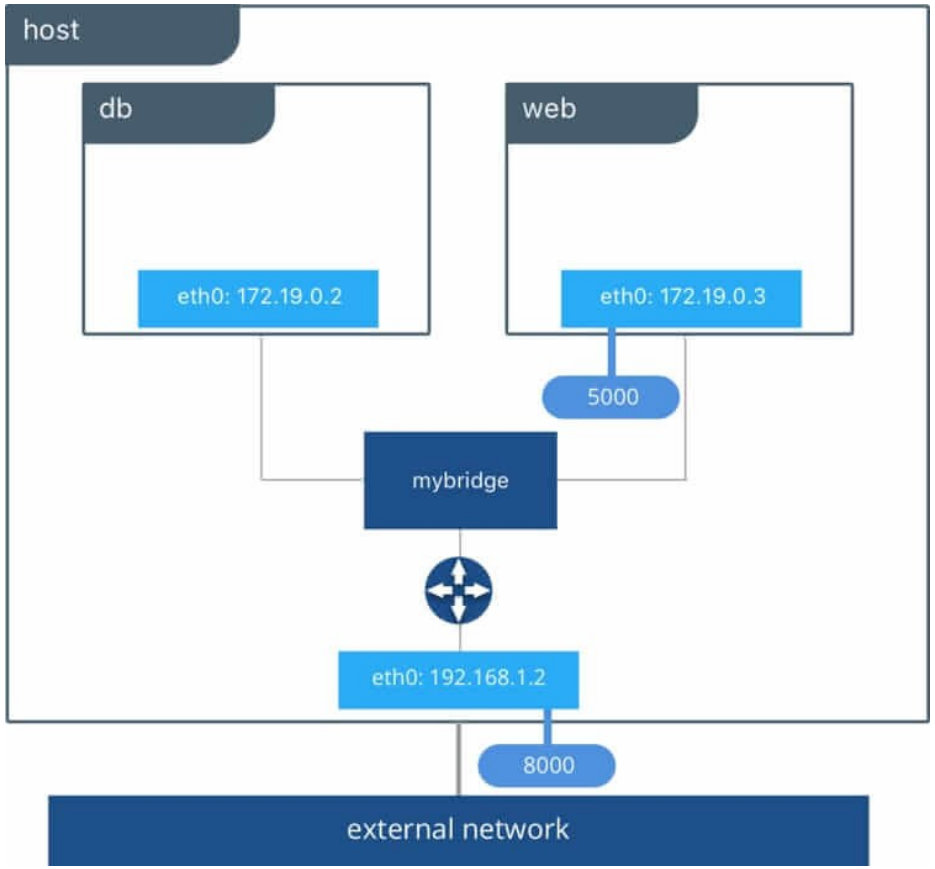
TP – Les commits



TP – Les liaisons

Créer plusieurs conteneurs et les lier (link ou nouveau network)

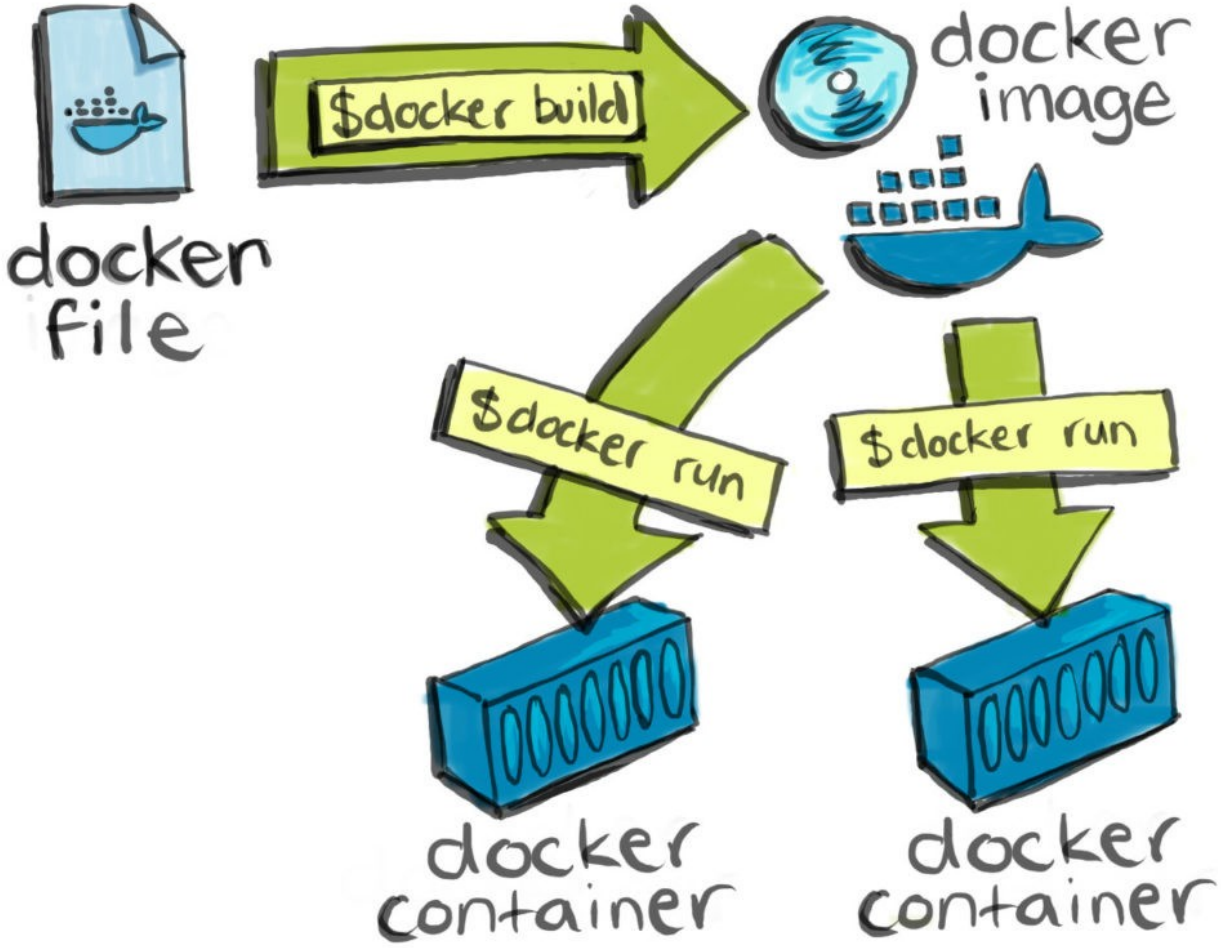

docker run -p 8081:8080 --name appwithmysqlcontainer
--link mysqldb-container appwithmysqlimage
String connectionUrl =
System.getenv("dbConnectionUrl");
Java application
Environment Variable:
dbConnectionUrl=jdbc:mysql://
mysqldb-container:3306/collegeapp_db
Tomcat Container
docker run --name mysqldb-container
-p 3306:3306
-e MYSQL_ROOT_PASSWORD=root
-d mysql
student
collegeapp_db
My Sql Container



TP – Les ports publiés

Accèder au conteneur depuis l’extérieur

Dockerfile


$docker build
docker image
docker file
$docker run
$docker run
docker container
docker container

Dockerfile

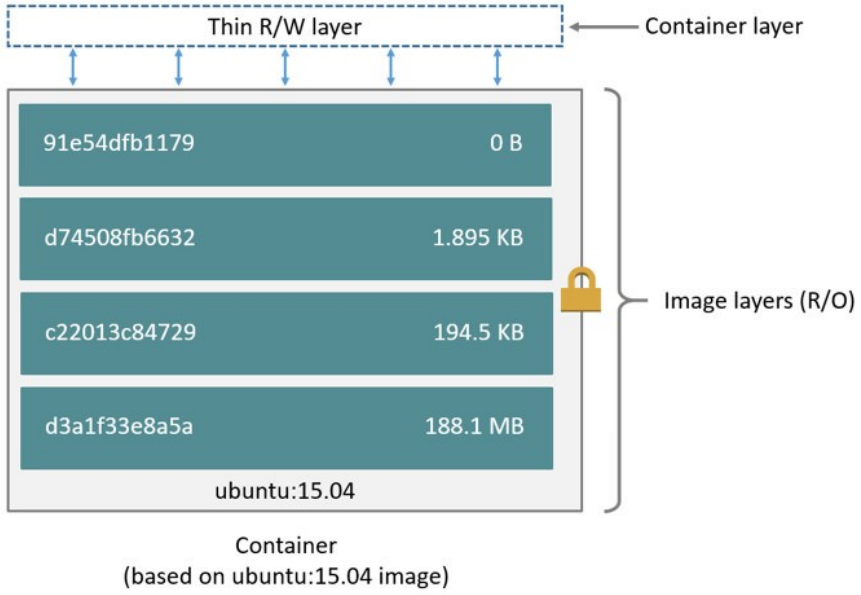
```
# Spécifie l'image parente à partir de laquelle vous créez
FROM ubuntu:bionic

MAINTAINER Pierre

# Ajout de métadonnées
LABEL example-release-date="2019-04-10"

# Ajout d'une variable d'environnement
ENV APACHE_RUN_USER=www-data APACHE_RUN_GROUP=www-data \
    APACHE_LOG_DIR=/var/log/apache2 APACHE_LOCK_DIR=/run/lock/apache2 \
    APACHE_PID_FILE=/run/apache2/apache2.pid APACHE_RUN_DIR=/run/apache2

# Commande utilisée pour construire l'image
RUN apt-get update && apt-get install -yq apache2 \
    && apt-get clean && rm -rf /var/lib/apt/lists/* \
    && mkdir -p $APACHE_RUN_DIR && mkdir -p /var/log/apache2

# Déclaration d'un point de montage
VOLUME /var/log/apache2

# Ajout de fichier dans l'image
COPY ./index.html /var/www/html/index.html

# Changement de répertoire courant
WORKDIR /data

# Port(s) écouté(s) par le conteneur
EXPOSE 80

# Exécuter une commande au démarrage du conteneur
# OU spécifier les paramètres de la commande déclarée dans ENTRYPOINT
CMD [ "-D", "FOREGROUND" ]

# Exécuter une commande au démarrage du conteneur
ENTRYPOINT [ "/usr/sbin/apache2" ]
```

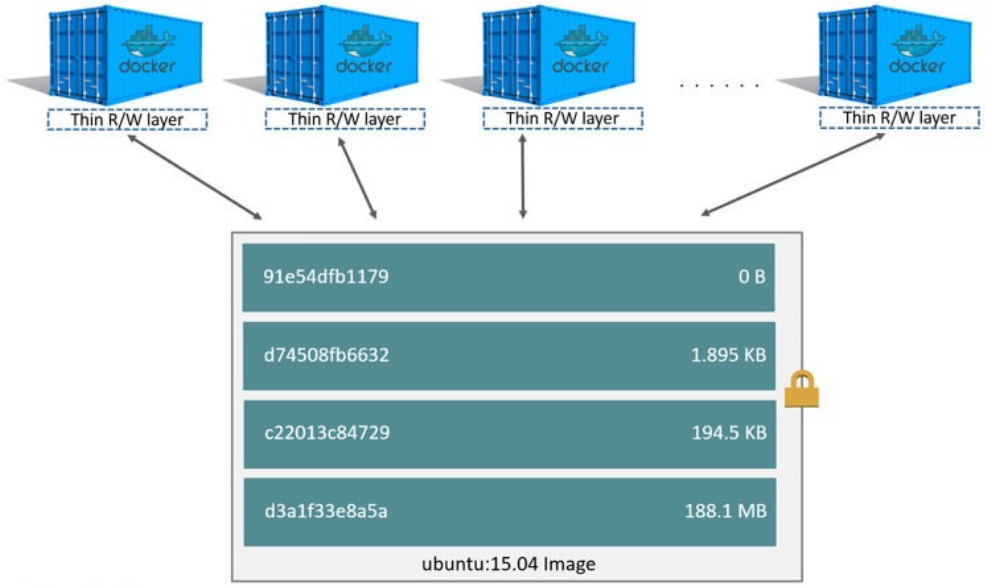

https://www.docker.com/blog/intro-guide-to-dockerfile-best-practices/

Les layers : fonctionnement

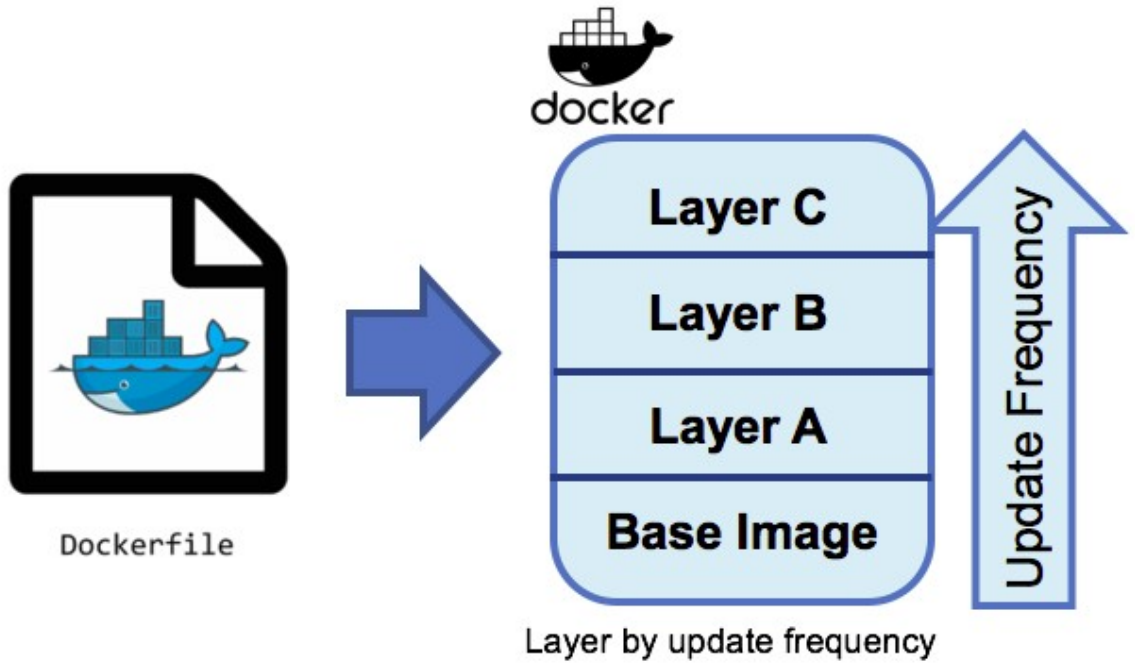
Image : Read only

Conteneur : Read & Write

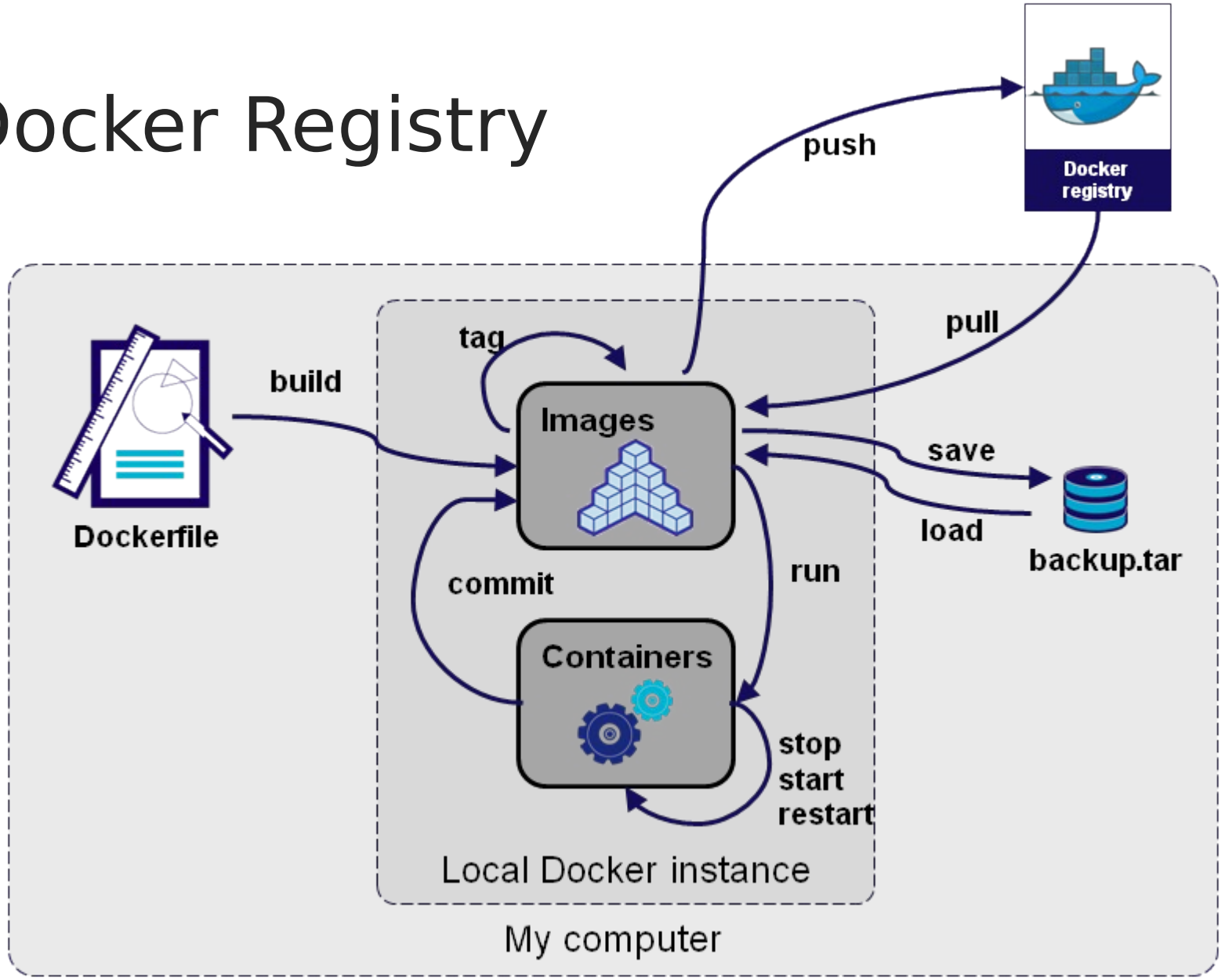




Les layers : fonctionnement



Docker Registry

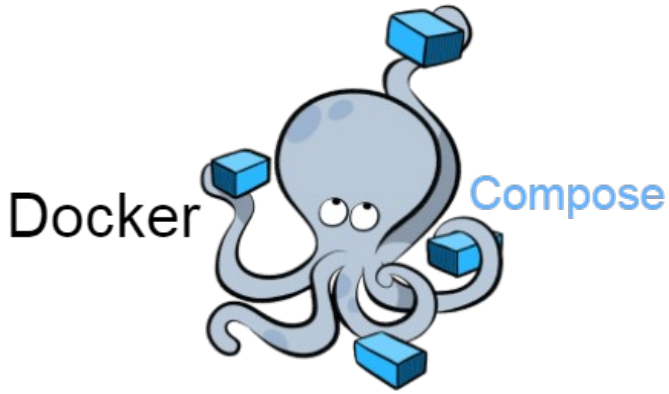


TP – Dockerfile - Registry

Créer ses propres images Docker, Installer un registry privée

Docker – Network

Docker-compose



IaC

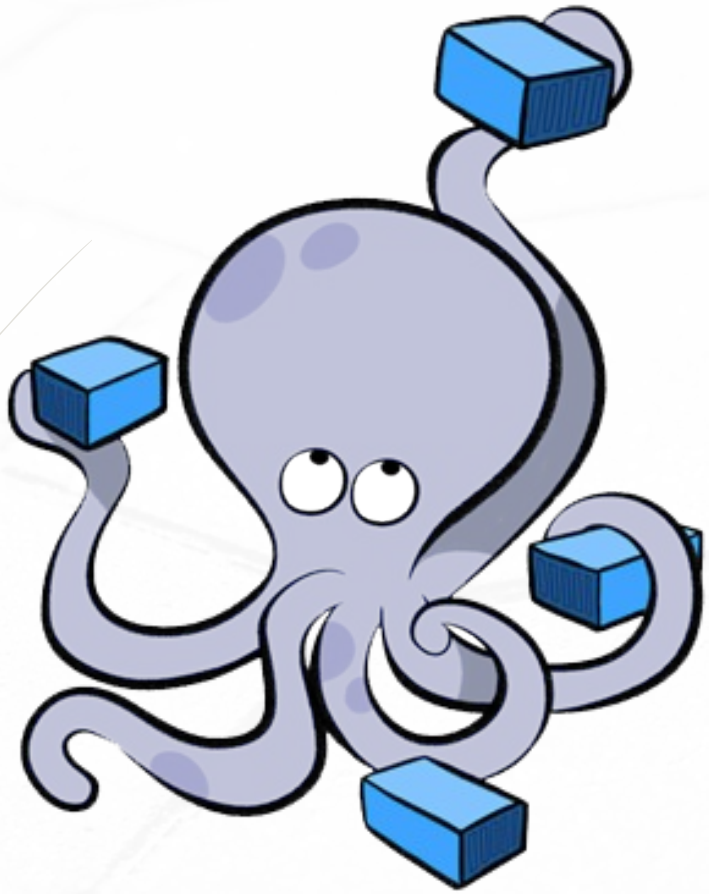
Dernière version : 3.7

Lien :

versions : https://docs.docker.com/compose/compose-file/

sources : https://github.com/docker/compose

releases : https://github.com/docker/compose/releases



```
php:
  build: php
  ports:
    - "80:80"
    - "443:443"
  volumes:
    - ./php/www:/var/www/html
  links:
    - db
```

```
$ docker-compose up
```

TP – docker-compose

Docker – Pour aller plus loin

Docker-machine & Remote

Outil de « Provisioning »

Multiples plugins :

- Les principaux « **cloud service providers** » CSP
- bare-metal
- « generic »

$ docker-machine create –driver=virtualbox $MACHINE_NAME

$ docker-machine create \
--driver generic \
--generic-ip-address=203.0.113.81 \
--generic-ssh-user=dawan \
--generic-ssh-key ~/.ssh/id_rsa \
${bastion}

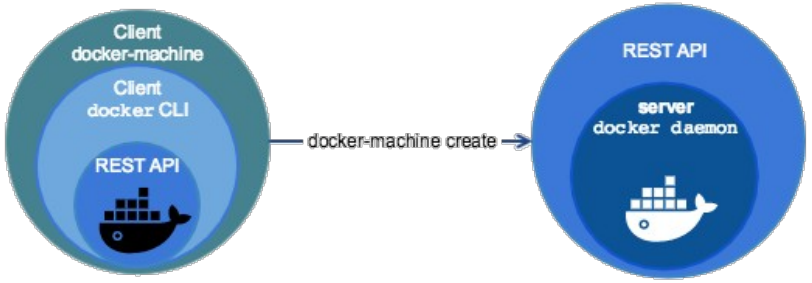
URL : https://docs.docker.com/machine/drivers/

Remote :

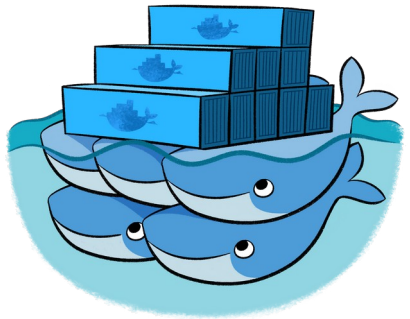
$ docker -H=your-remote-server.org:2375 %*

$ docker --tlsverify --tlscacert=ca.pem –tlscert=cert.pem \
--tlskey=key.pem -H=$HOST:2376 version

**$** export DOCKER_HOST=tcp://$HOST:2376 \
DOCKER_TLS_VERIFY=1



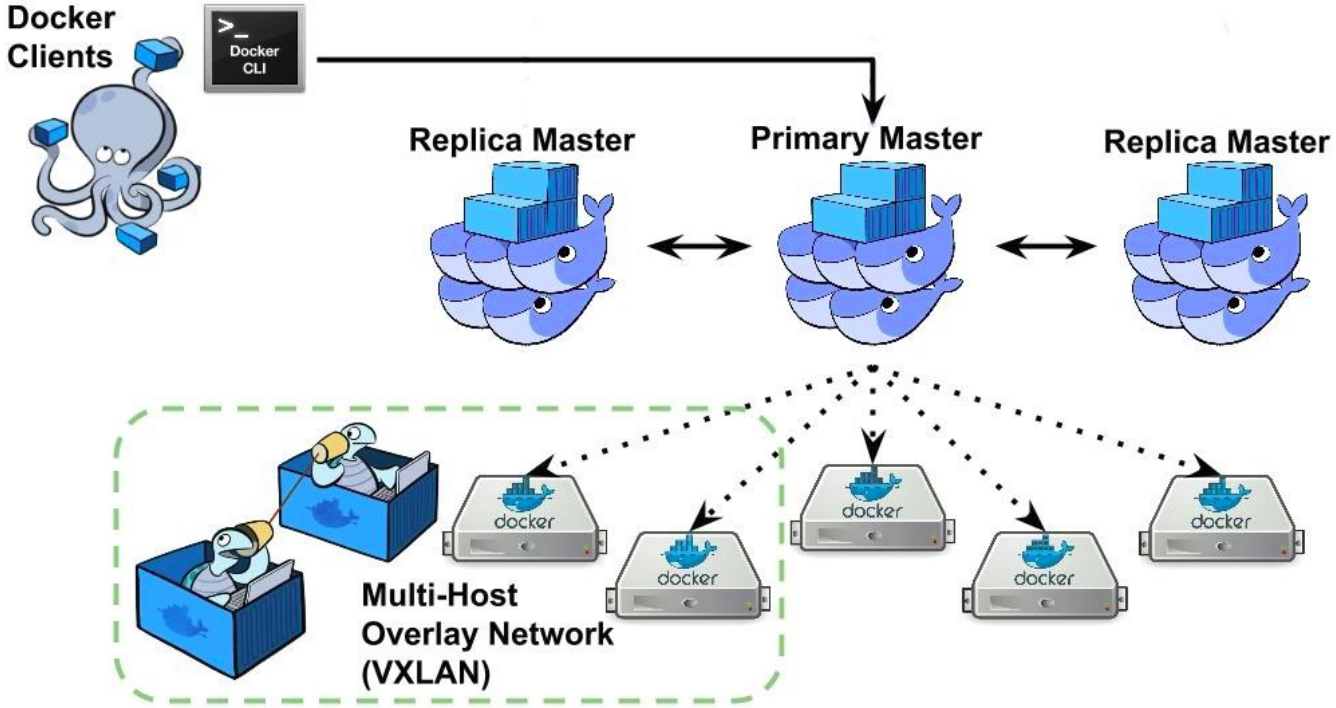
Docker swarm



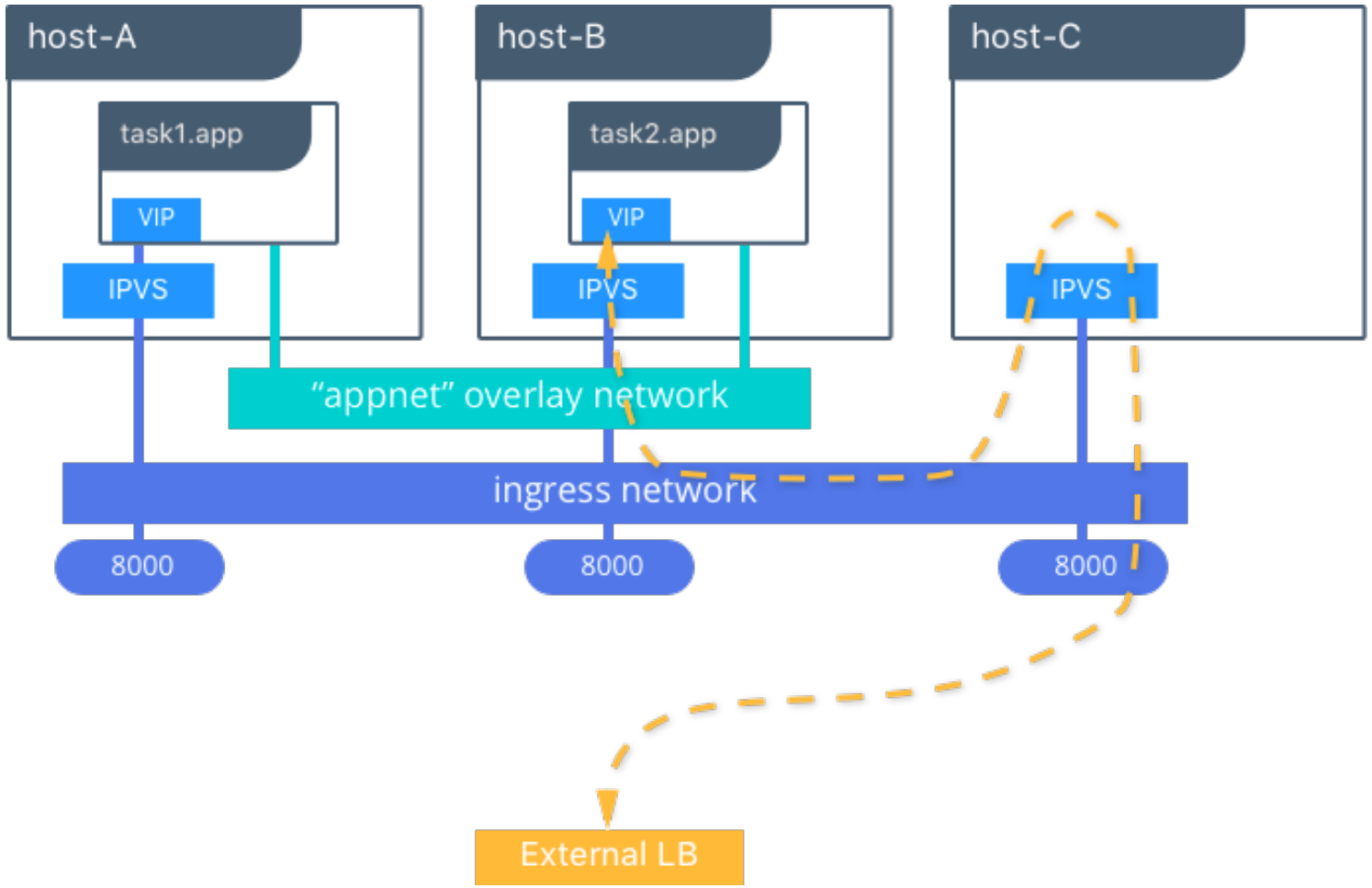
$ docker swarm init

$ docker swarm join

$ docker node ls



Routing mesh



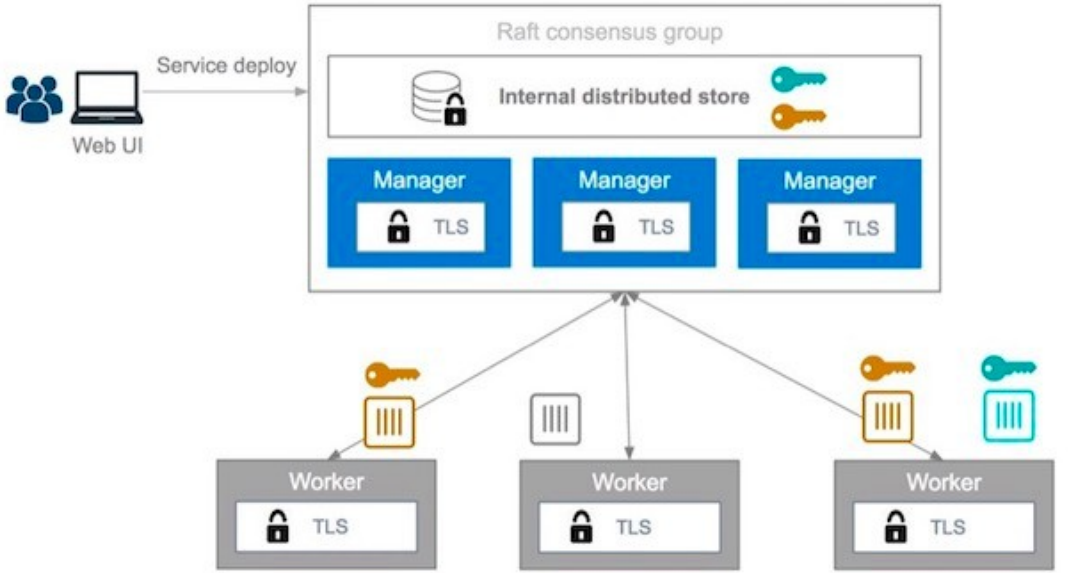
Docker secrets

Docker swarm : stack

Gérés de manière centralisée

Autorisation au conteneur qui ont besoin d’accès aux secrets

$ docker secret [opt]



Solutions implémentant Dockers

VMware VSphere 6.5 : VIC

Hyper-V 2016 & Cloud Azure

OpenStack

Kuburnetes

Rancher OS

Autres technologies

Buildpack-deps | Heroku

Guso |

sources: https://github.com/tianon/gosu

portainer : monitoring

sources : https://github.com/portainer/portainer

Cadvisor : metrics

sources : https://github.com/google/cadvisor

Notary

sources : https://github.com/theupdateframework/notary

Multi-stage builds to remove build deps

```
FROM maven:3.6-jdk-8-alpine AS builder
WORKDIR /app
COPY pom.xml .
RUN mvn -e -B dependency:resolve
COPY src ./src
RUN mvn -e -B package

FROM openjdk:8-jre-alpine
COPY --from=builder /app/target/app.jar /
CMD ["java", "-jar", "/app.jar"]
```