



Samuel ANTUNES
Consultant Ingénieur DevSecOps
OCTO Technology
Email : contact@samuelantunes.fr

ICEBREAKER

shorturl.at/imMW9

1. Les bases de Docker

- a. Introduction: l'avant Docker
- b. Qu'est-ce que Docker
- c. Architecture et Concepts

2. Docker en Pratique

- a. Les images
- b. Les conteneurs
- c. Les volumes
- d. Les networks
- e. Création d'images (Dockerfile) et les registres
- f. Docker-compose

3. Docker en Prod

- a. Doable or not

“ Les bases de Docker ”

- De nombreuses problématiques liées aux applications
 - La portabilité des applications
 - La distribution des applications
 - Le besoin de décorréliser applications et infrastructure
 - La rationalisation des infrastructures
- La montée en puissance
 - Des solutions de PaaS
 - De la philosophie DevOps

Comment assurer le déploiement homogène
d'une application sur tous ses environnements ?



	Env de développement	Assurance Qualité (QA)	Serveur de Production	Cluster de machines	Cloud Public	Ordinateur Personnel	Serveur du Client
Site web statique	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Jobs en Arrière Plan	?	?	?	?	?	?	?
Base de données	?	?	?	?	?	?	?
Analytics	?	?	?	?	?	?	?
Files de messages	?	?	?	?	?	?	?

Comment distribuer un logiciel de façon simple et efficace ?

Les différentes méthodes de distribution logicielle



Binaire



Paquet



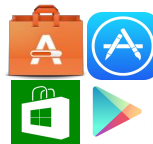
Installeur



**Dépôts
de paquets**



**Virtual
Appliance**



**Application
Store**

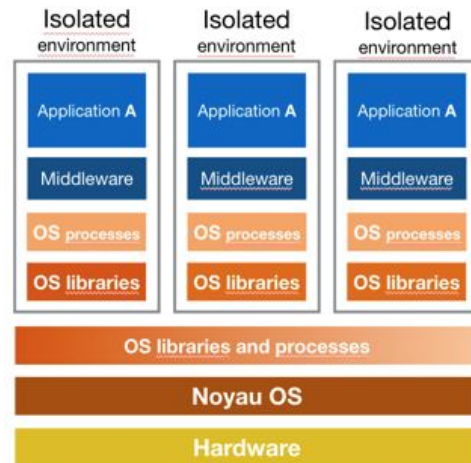
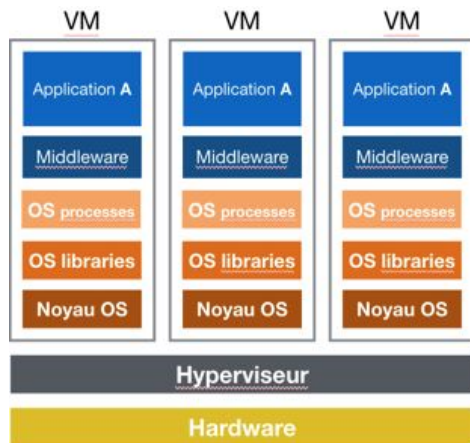
Comment optimiser l'utilisation des ressources ?
Comment décorréler application et infrastructure ?

Les 2 technologies de virtualisation des systèmes

Virtualisation



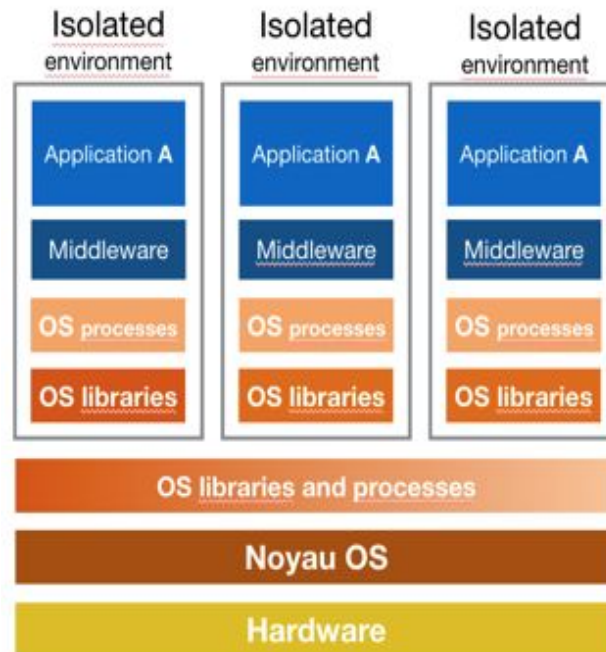
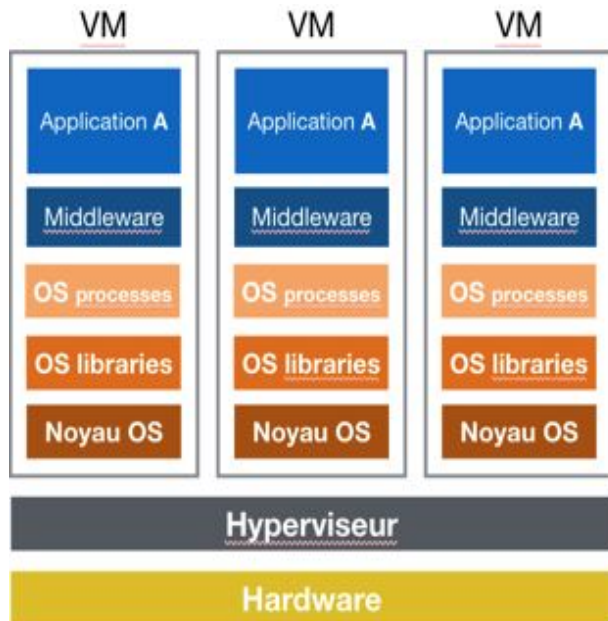
Isolation



VMWare

VS

LXC



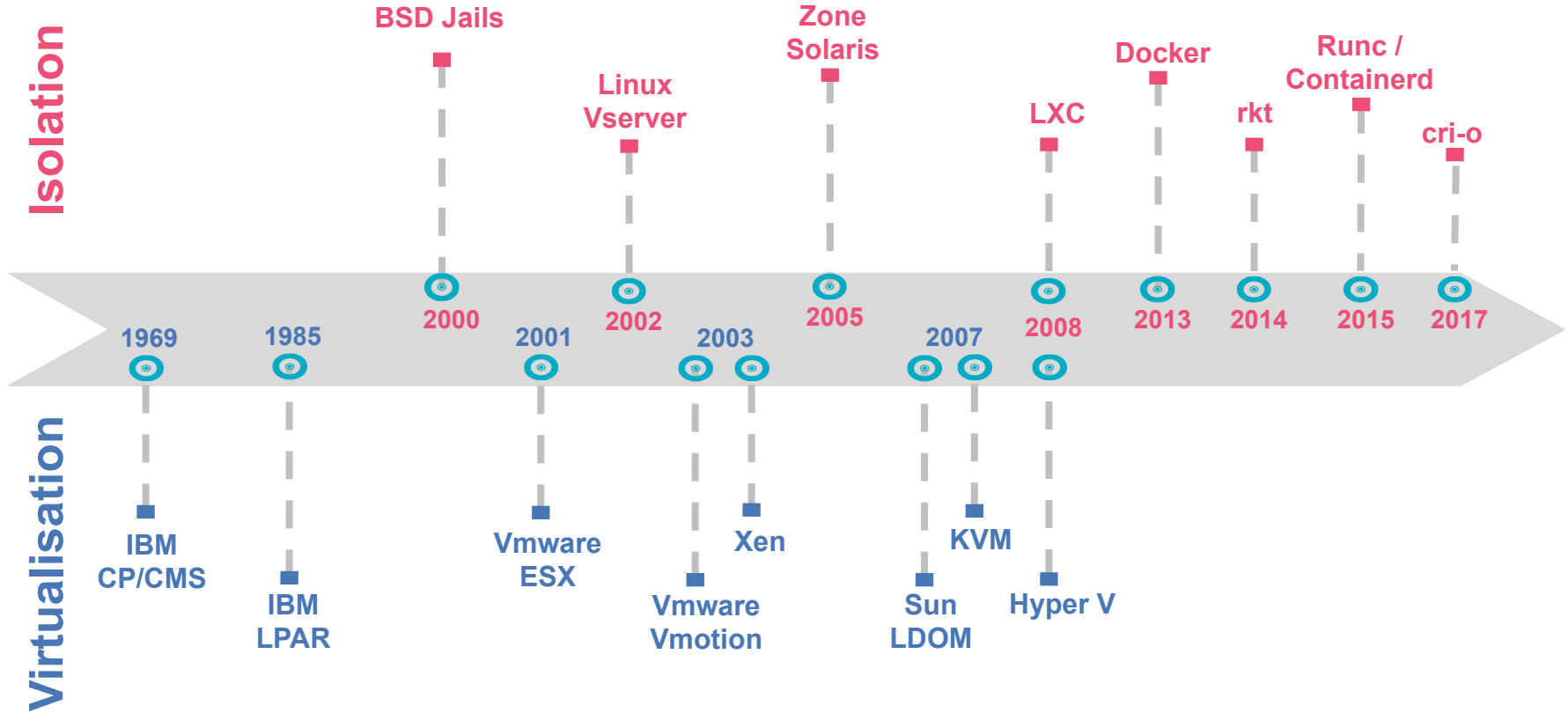
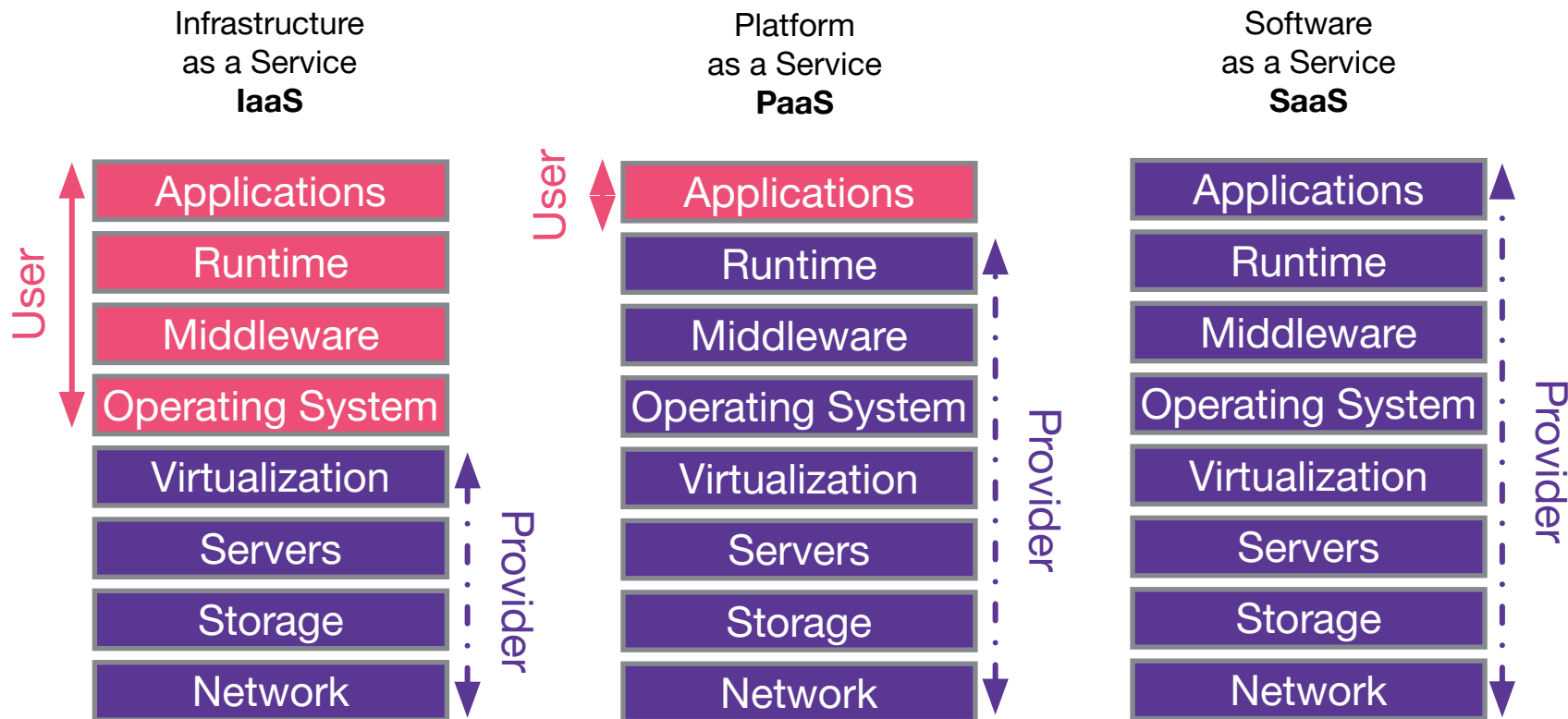


Schéma des différents niveaux de services Cloud



- Les PaaS concentrent toutes les problématiques indiquées et doivent en plus :
 - Déployer rapidement des nouvelles applications
 - Assurer une élasticité rapide
 - Isoler les applications entre elles

**→ Besoin d'une sur-couche légère d'isolation et d'abstraction
... des conteneurs !!**

Les principaux PaaS et leur technologie de conteneurs



OPENSIFT



LXC containers



« DevOps est un ensemble de pratiques qui visent à réduire le Time to Market et améliorer la Qualité en optimisant la coopération entre les **Développeurs** et la **Production** »

Docker (homonymie)

🔗 Cette page d'*homonymie* répertorie les différents sujets et articles partageant un même nom.

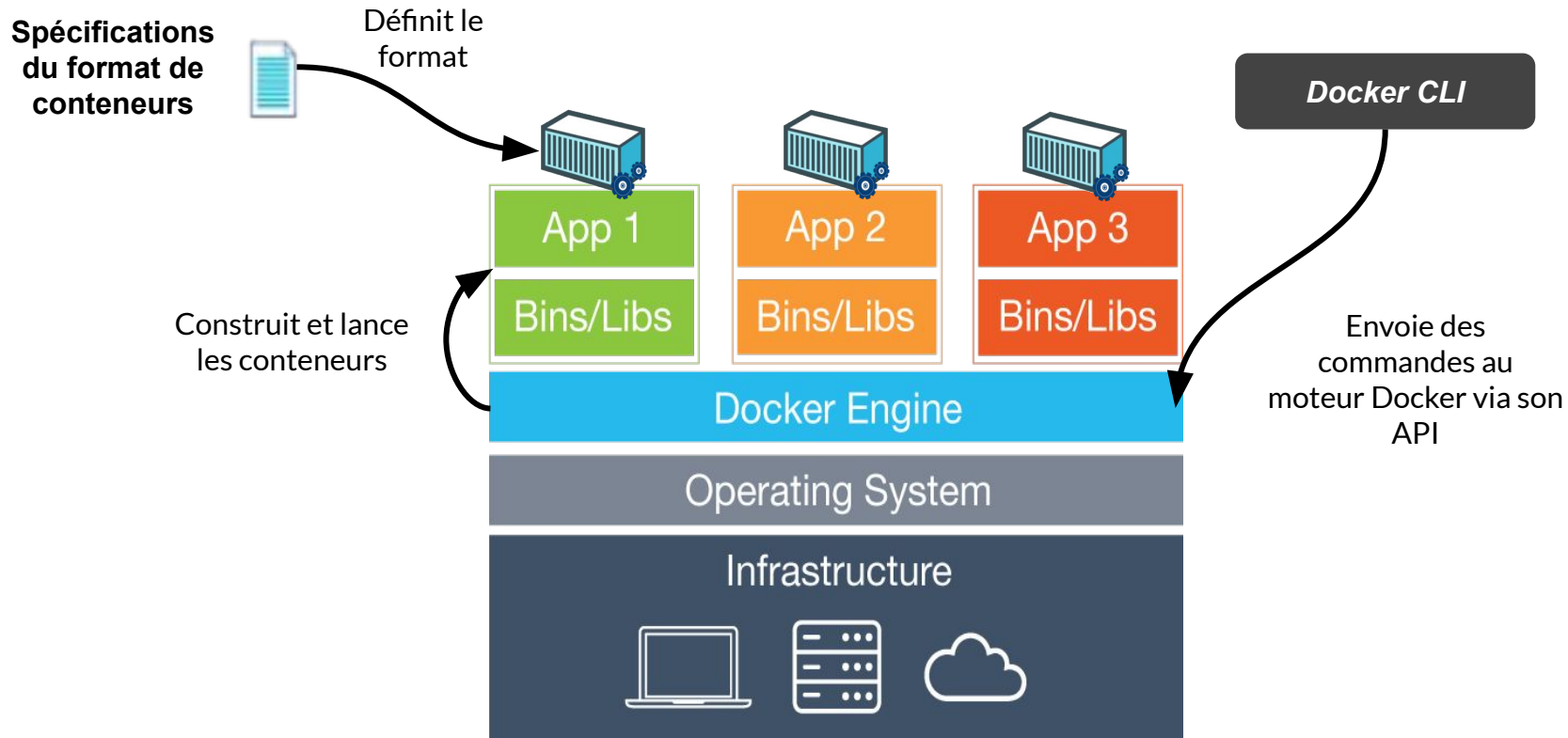
Informatique

- **Docker Inc**, la compagnie qui développe la plateforme Docker.
- **Docker Engine**, le logiciel qui construit et fait tourner les conteneurs.
- **Docker**, le format de conteneur.
- **Docker CLI**, l'outil en ligne de commande pour piloter les conteneurs.
- **Docker Platform**, l'ensemble des logiciels de Docker Inc permettant de gérer les conteneurs.

« Une **technologie** permettant de **standardiser** le **packaging** et l'**opération** des **applications** »



QU'EST-CE DOCKER ? ENGINE, CLI, CONTENEURS



QU'EST-CE DOCKER ? DOCKER INC

- Société créée initialement en 2008 à San Francisco sous le nom de DotCloud, pour offrir un service de PaaS
- **Renommée en Docker Inc fin 2013** pour se concentrer autour du projet Docker, puis revend la partie PaaS mi-2014
- **Mène depuis 2014 une politique d'acquisition** des solutions qui émergent de la communauté (orchard, kitematic, socketplane, tutum...)
- **Se positionne comme le leader du projet communautaire Docker** de développement d'une plateforme ouverte pour les applications distribuées
- Docker Inc modifie son système de packaging du **Docker Engine** avec l'introduction de **Moby** en 2017
- En 2017, Docker Inc lance le **Modernize Traditional Applications** (MTA) en s'associant avec des **éditeurs traditionnels**
- En 2018, Docker EE 2.0 intègre désormais **Kubernetes** et rends possible le déploiement des **stacks** et des fichiers **compose** au travers de Swarm ou Kubernetes

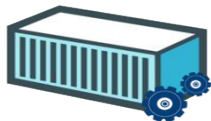
Des caractéristiques uniques

PO**PORTABLE****DI****DISPOSABLE****LI****LIVE****SO****SOCIAL**

```
git clone https://3AIW:rBrPi4tgoyEHXjLivp9y@gitlab.com/santunes-formations/docker.git
```

**L'image**

Une arborescence de fichiers contenant tous les éléments requis pour faire tourner une application

**Le conteneur**

Une instantiation d'une image en cours d'exécution sur un système hôte

**Le registre**

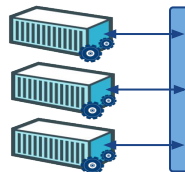
Un service centralisé de stockage et distribution d'images

**Le montage de répertoires**

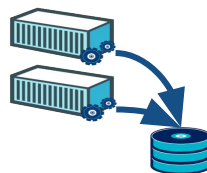
Un espace de stockage indépendant de l'image utilisable pour les données persistantes

**Le *Dockerfile***

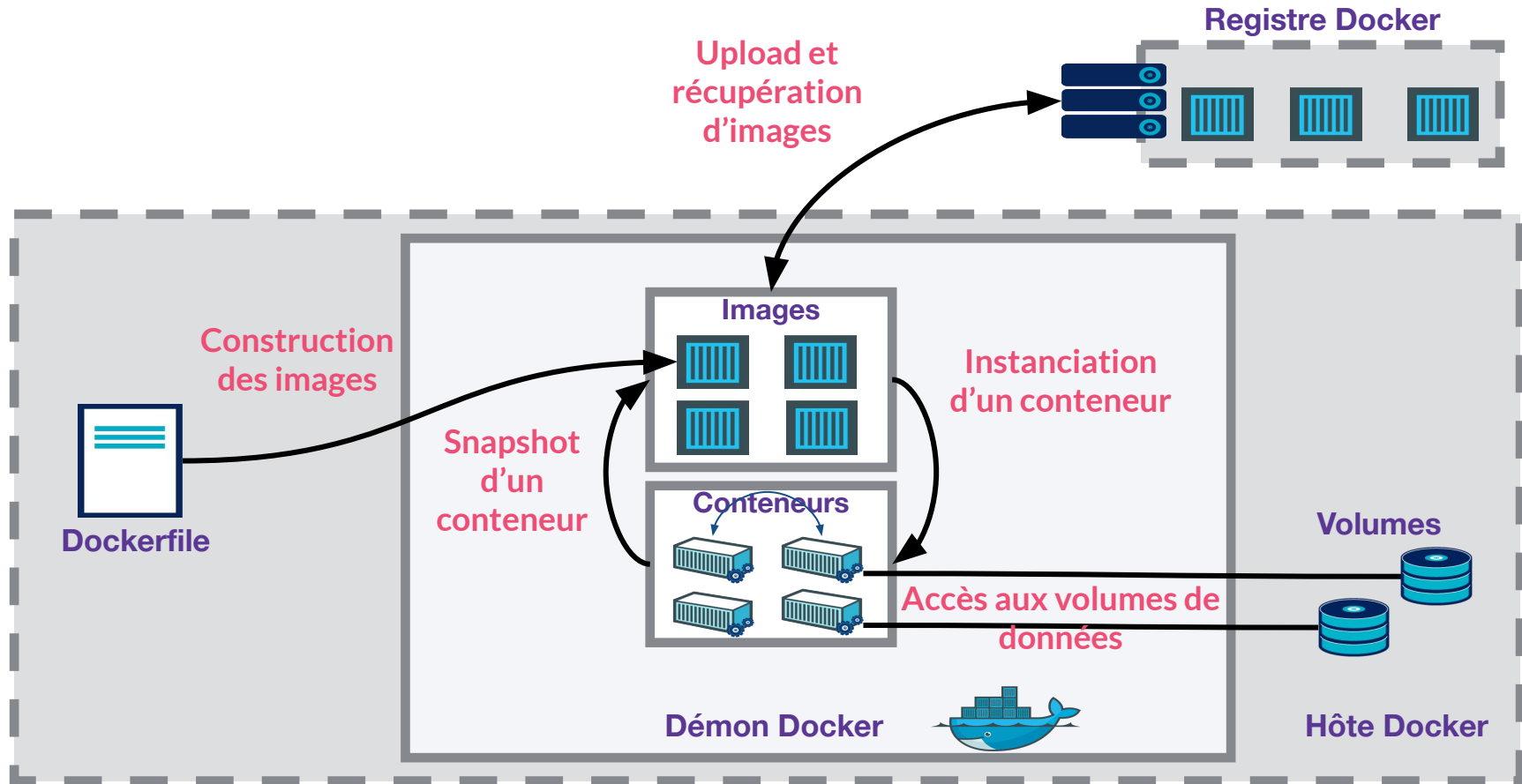
Un fichier contenant les instructions permettant de construire une image

**Les networks (docker network)**

Permet la communication de conteneurs dans un ou plusieurs réseaux sur une ou plusieurs machines hôtes

**Les volumes (docker volume)**

Un espace de stockage indépendant de l'image utilisable pour les données persistantes.

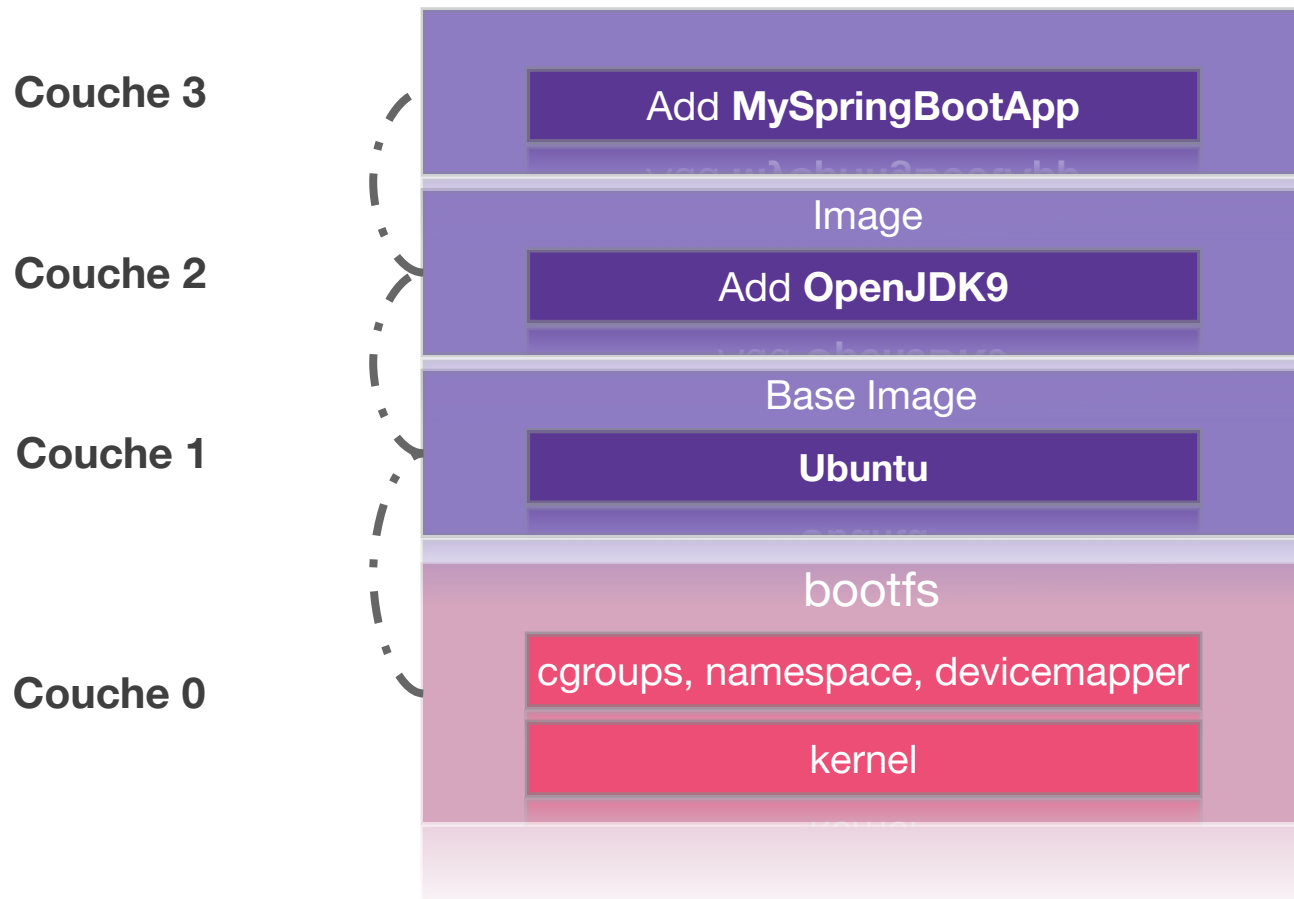


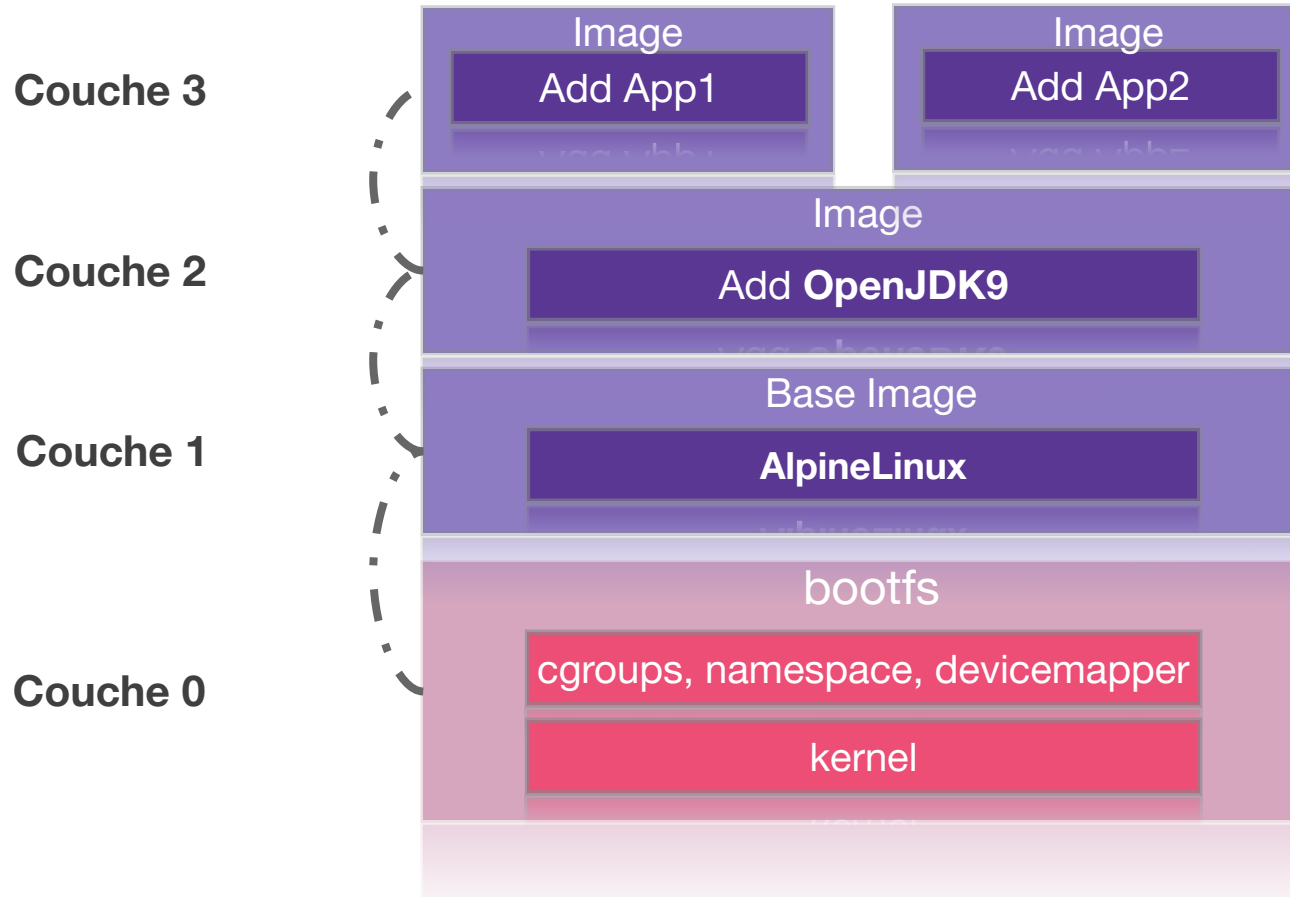
“ Docker en pratique ”

Une image Docker c'est

- **Un système de fichiers auto-suffisant** contenant *a minima* librairies et binaires de base (libc, libresolv, bash...)
- **Un identifiant unique** assigné à l'image à sa création
- **Des métadonnées** pour préciser la façon d'instancier l'image
 - le processus à exécuter à l'instanciation de l'image,
 - les variables d'environnement à positionner
 - l'utilisateur qui va lancer l'application
 - la configuration réseau (ports exposés, réseau...),
 - les volumes de données à connecter,
 - ...

- Docker utilise un système de fichiers avec un **système de couches** pour les images de conteneurs
- **Principe**
 - Un ensemble de couches partagées en lecture seule
 - Unifiées par le système pour simuler un unique système de fichiers à plat pour le conteneur
- **Avantages**
 - **Évite la perte d'espace** avec $n \times 500\text{Mo}$ par OS Ubuntu dans des VMs
 - Permet la récupération et le démarrage rapide des conteneurs





Lister les images
locales

docker image ls

Télécharger une
image à partir d'un
registre

docker image pull



docker image rm

Supprimer une image

docker image tag

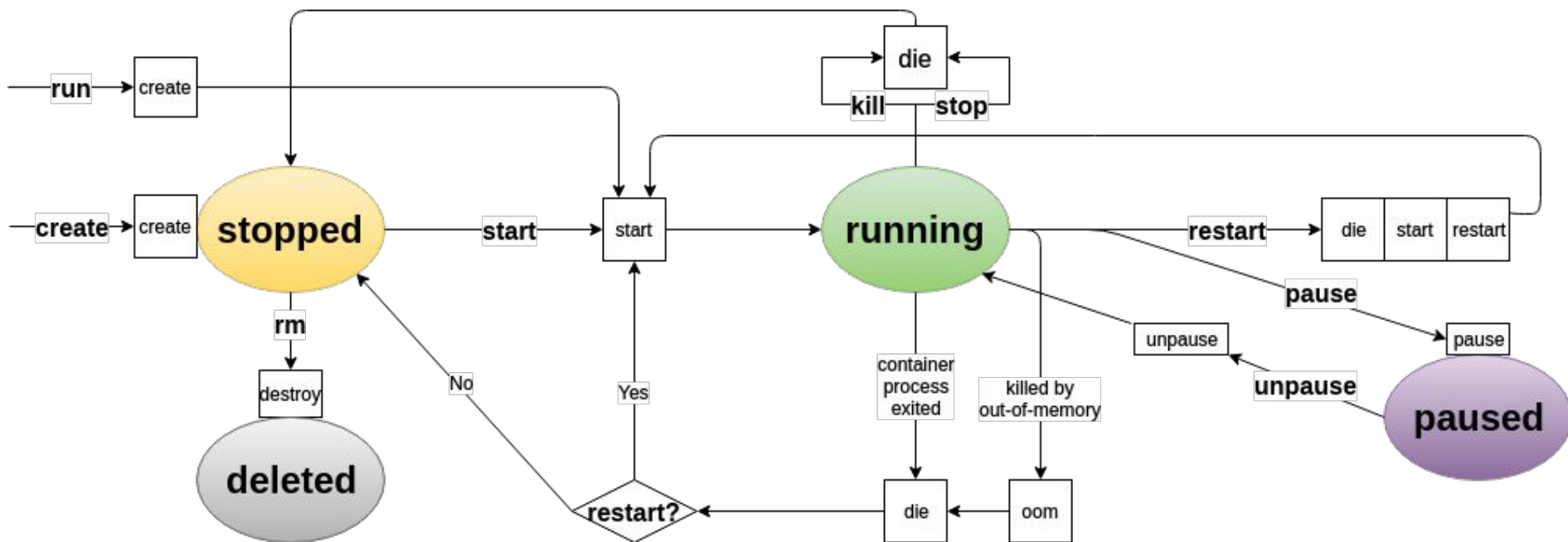
Labelliser une image

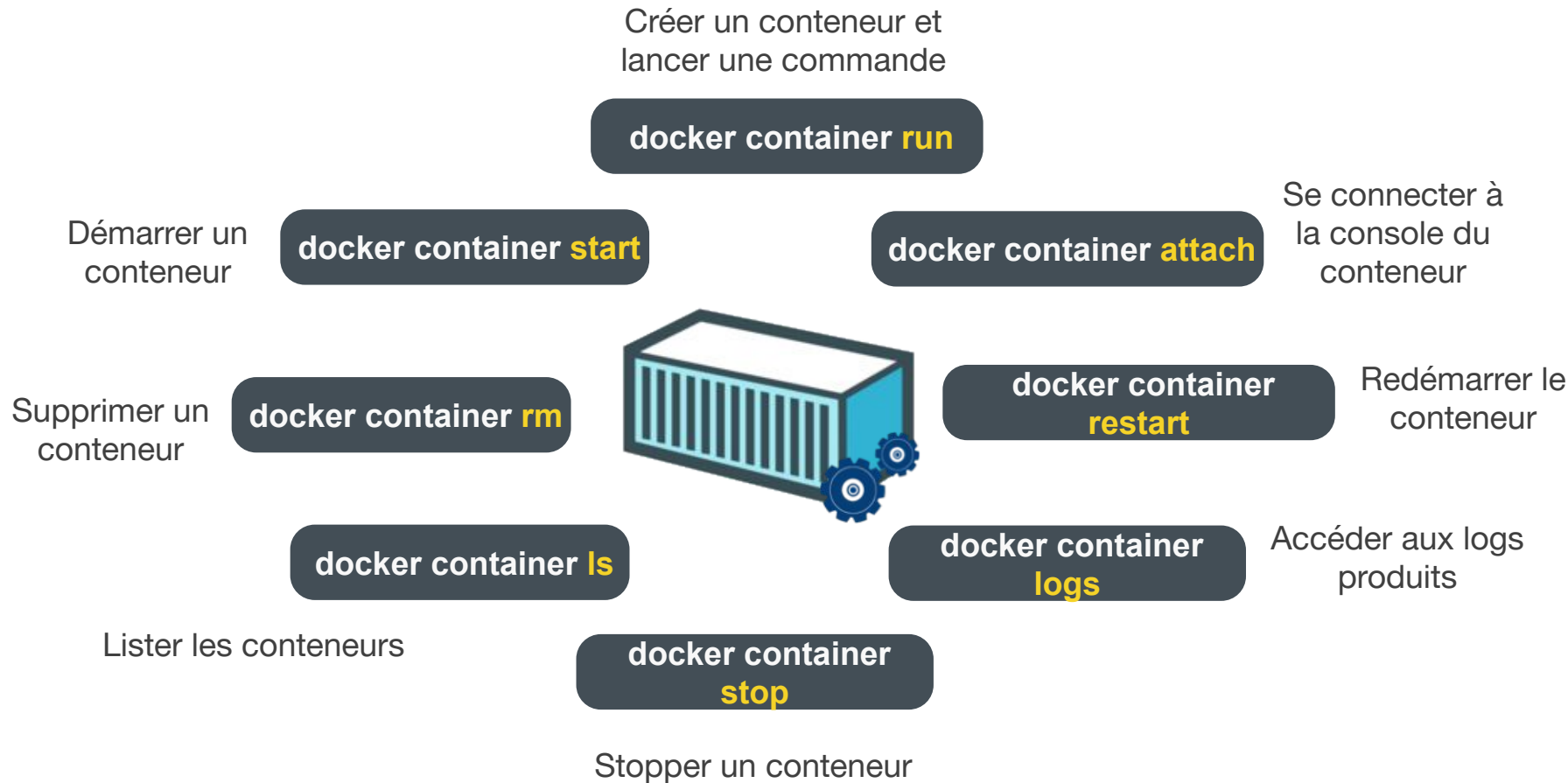
docker image history

Lister les couches d'une
image

```
git clone https://3AIW:rBrPi4tgoyEHXjLivp9y@gitlab.com/santunes-formations/docker.git
```

- Le conteneur est **la brique de base de Docker**
- Il est toujours **instancié à partir d'une image**
- Un conteneur **ne vit que pour les processus** qu'il contient
- **Si ces processus s'arrêtent**
 - Le conteneur est stoppé
 - Le contenu modifié subsiste tant que le conteneur n'est pas détruit





Créer un conteneur

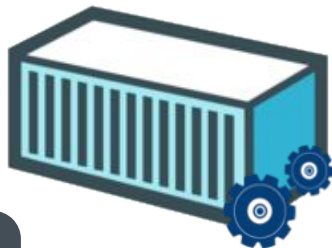
docker container
create

Lister les
processus

docker container
top

Exécuter une commande
dans le conteneur

docker container
exec



docker container
diff

Affiche les différences dans le
système de fichiers du
conteneur

docker container
inspect

Affiche des informations sur
un conteneur

docker container
kill

Tue un conteneur

```
git clone https://3AIW:rBrPi4tgoyEHXjLivp9y@gitlab.com/santunes-formations/docker.git
```

- Les conteneurs sont **légers et éphémères** et ne sont pas faits pour enregistrer des données de matières persistantes
- Les données doivent être stockées en dehors de l'image du conteneur dans des **volumes de données dédiés** à cet usage
- **2 techniques historiques pour accéder à des volumes** de données
 - l'accès au système de fichiers de l'hôte,
 - le volume lié au conteneur

Objectifs :

- **Gérer des applications stateful** : pour les bases de données et les applications à architecture traditionnelle
- **Conserver l'indépendance avec les hôtes** : les données ne doivent pas être liées à un hôte et doivent suivre le déplacement des conteneurs qui y sont associés
- **Pouvoir conserver des données indépendamment de l'application** : pour gérer le cycle de vie de la donnée
- **Faciliter les tâches opérationnelles** : snapshot, sauvegarde, restauration, copie...

Créer un volume de données

`docker volume create`

Supprimer un volume

`docker volume rm`



Obtenir les informations
sur un volume

`docker volume inspect`

`docker volume ls`

Lister les volumes existants

Exemples sur 2 cas d'utilisation:

- Création d'un volume de données pour MySQL

```
$ docker volume create --name mysql_data  
$ docker container run -d -v mysql_data:/var/lib/mysql mysql
```

- Sauvegarde puis restauration des données de MySQL

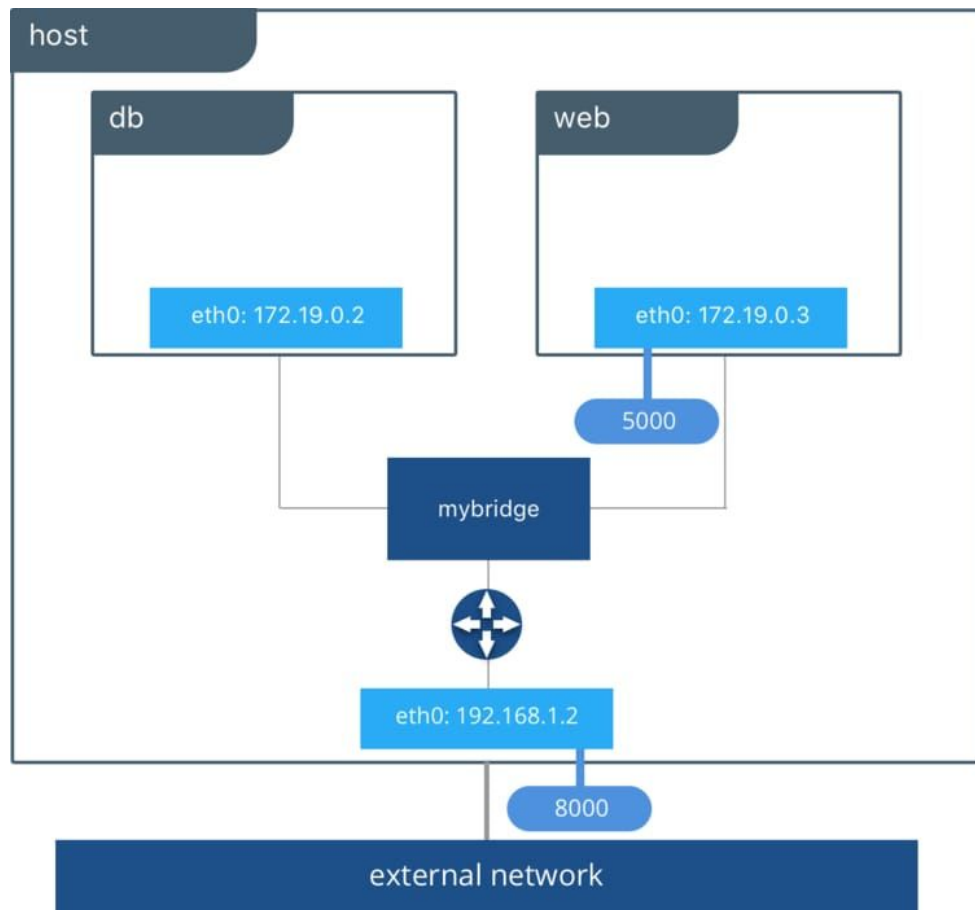
```
$ docker volume create --name mysql_backup  
$ docker container run -v mysql_data:/var/lib/mysql -v mysql_backup:/backups mysql  
tar cvjf /backups/backup.tar.bz2 /var/lib/mysql
```

```
$ docker container run -v mysql_data:/var/lib/mysql -v mysql_backup:/backups mysql bash  
-c 'cd /var/lib/mysql && tar xvjf /backups/backup.tar.bz2'
```

```
git clone https://3AIW:rBrPi4tgoyEHXjLivp9y@gitlab.com/santunes-formations/docker.git
```

- Les networks peuvent être des réseaux privés permettant d'isoler des conteneurs entre eux.
- Il existe plusieurs drivers de réseaux permettant de réaliser des actions particulières :
 - Le driver Bridge
 - Le driver none
 - Le driver Host
 - Le driver overlay
 - Le driver macvlan
- On peut ouvrir des ports entre notre machine hôte et notre conteneur grâce à l'argument `"-p <PORT_HOTE>:<PORT_CONTENEUR>"`

- A l'installation de Docker, un réseau nommé **bridge** connecté à l'interface réseau docker0 est créé et lié à chaque conteneur **par défaut**.
- Le réseau bridge est le type de réseau le plus **couramment** utilisé.
- Les conteneurs qui utilisent ce driver, **ne peuvent communiquer qu'entre eux**, cependant ils ne sont **pas accessibles** depuis l'extérieur.



- **None** : Réseau qui isole complètement un conteneur (ni entrées, ni sorties)
- **Host** : Le conteneur utilise l'interface réseau de son hôte (prendra son IP le rendant disponible à l'extérieur)
- **Overlay** : Permet de créer un lien réseau partagé entre plusieurs hôtes. Docker gère de manière transparente le routage.
- **Macvlan** : Ce type de réseau permet d'attribuer une adresse mac à un conteneur le faisant ainsi apparaître comme un périphérique physique

Créer un nouveau network

`docker network create`

Supprimer un network

`docker network rm`



Connecter un conteneur à un network existant

`docker network connect`

`docker network ls`

Lister les networks existants

`docker network disconnect`

Déconnecter un conteneur d'un network

```
git clone https://3AIW:rBrPi4tgoyEHXjLivp9y@gitlab.com/santunes-formations/docker.git
```