

---

# Docker

Conteneurs et orchestrations



# Plan

## Début de la formation

Présentation et théorie

## Pause déjeuner

On mange où ?

## Clôture

Questions et évaluations

9H00

10h45

12h

13h30

17h

### Pause

15 minutes! Et après on réattaque du pratique

### On re-attaque

Pratique, pratique pratique !!!



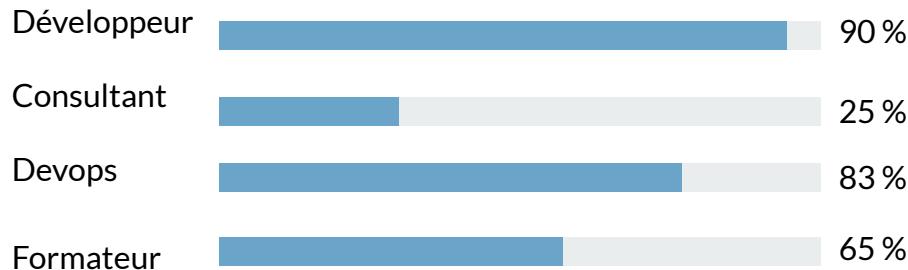
Où c'est qu'on mange ?  
Qui appelle pour  
réserver ?



Président Velvet Lemons Consulting

# Fabien Lamarque

Développeur, TechLead



 @Fabinout

 <https://github.com/Fabinout>



---

# Tour de table

- 01 | Qui suis-je ?
- 02 | Quel est mon métier ?
- 03 | Ce que j'attends de la formation ?
- 04 | Mes connaissances en Docker ?
- 05 | Mes connaissances en Devops





# Premier TP

Installation Docker & création du compte Docker

**<https://docs.docker.com/install/>**

```
$ docker --version  
$ docker search hello-world  
$ docker run hello-world  
$ docker run -it ubuntu bash
```

# Concepts de base

01

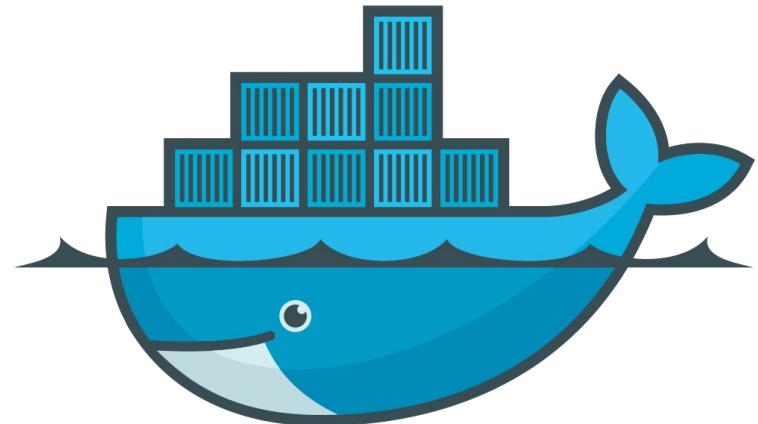
Qu'est-ce que Docker ?



---

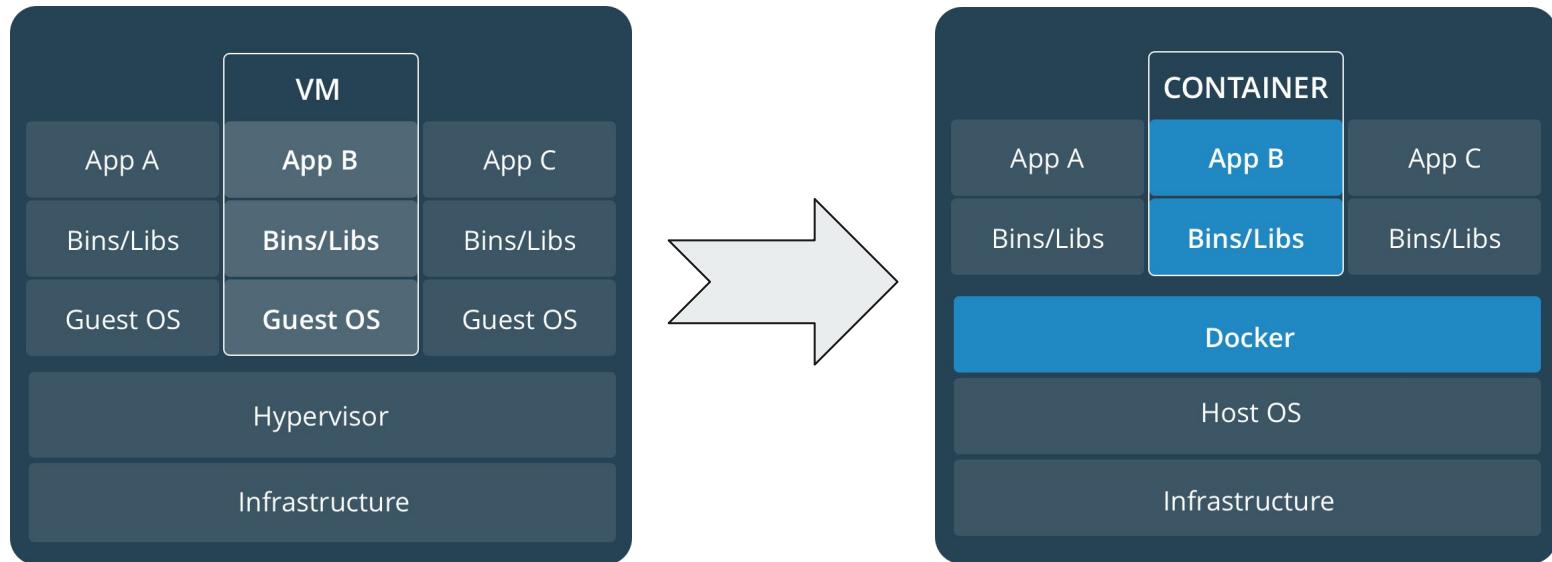
# Docker

It is a platform for developers and sysadmins to **develop, deploy, and run** applications with containers. The use of Linux containers to deploy applications is called *containerization*. Containers are not new, but their use for easily deploying applications is.

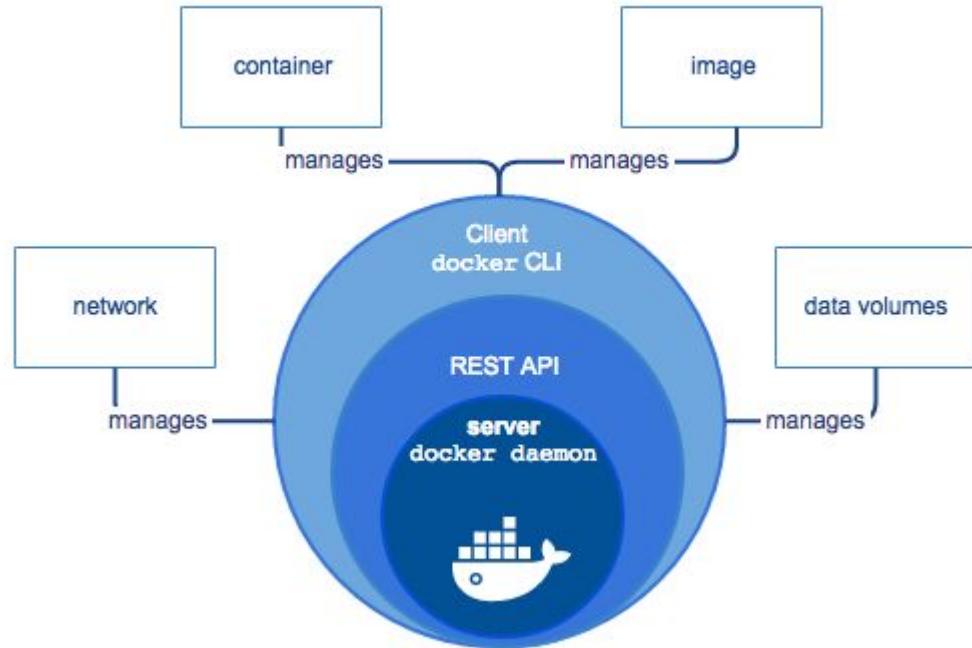


# docker

# Archi et concepts

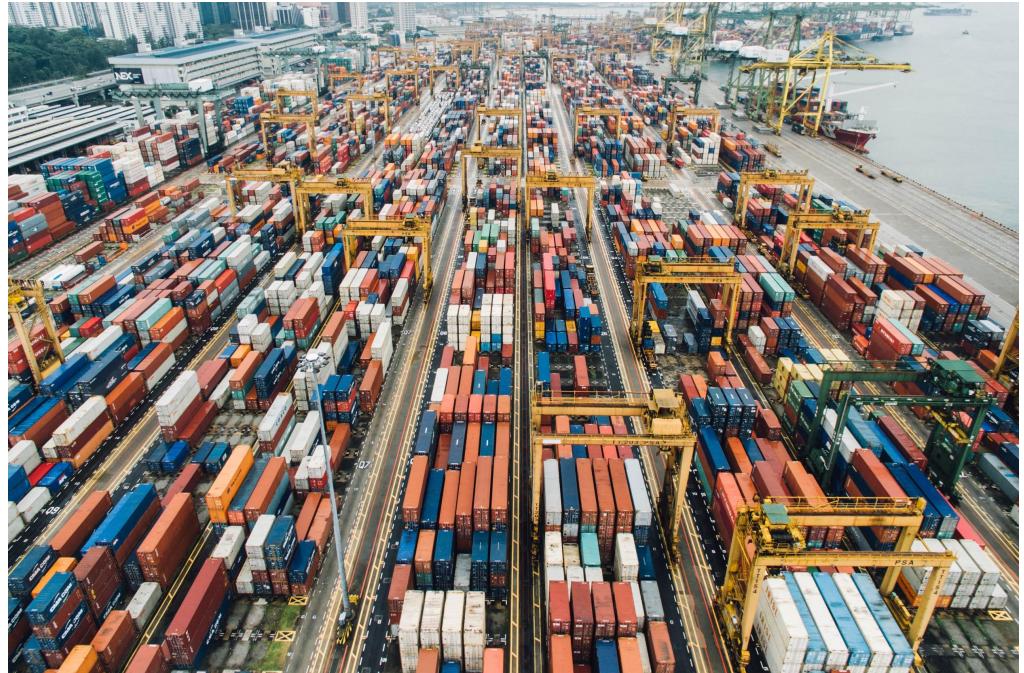


# Archi et concepts



---

# Quelle est la différence entre une image et un conteneur Docker ?



# Une image Docker

C'est un ensemble ordonnés de changements de filesystems organisé en couches. Une image est statique, elle ne change pas et n'a pas d'état.



# Un conteneur Docker

Un conteneur est l'instanciation (active ou inactive) d'une image Docker.



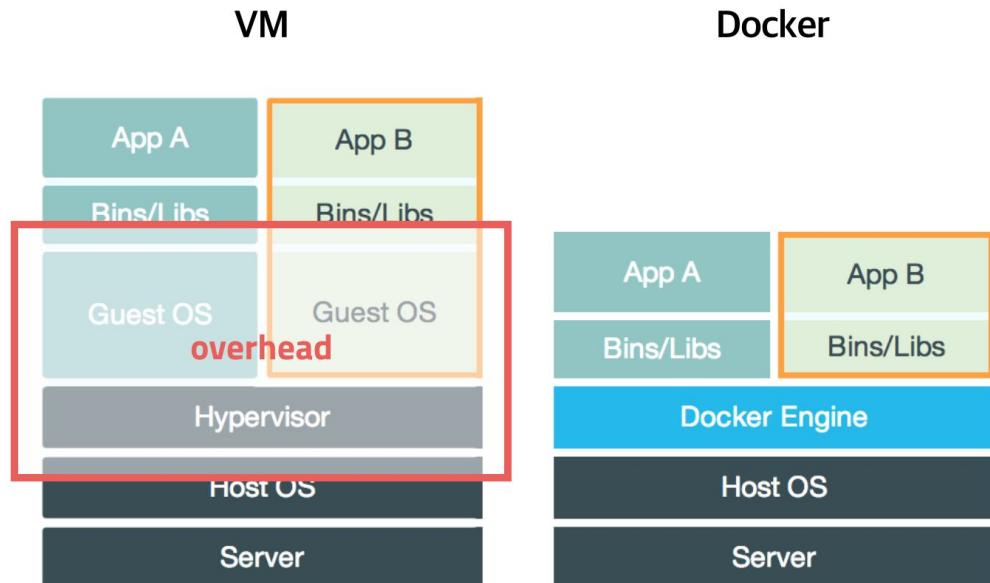
---

# Les avantages à utiliser Docker



# Docker est léger

Les conteneurs partagent les ressources de la machine.





---

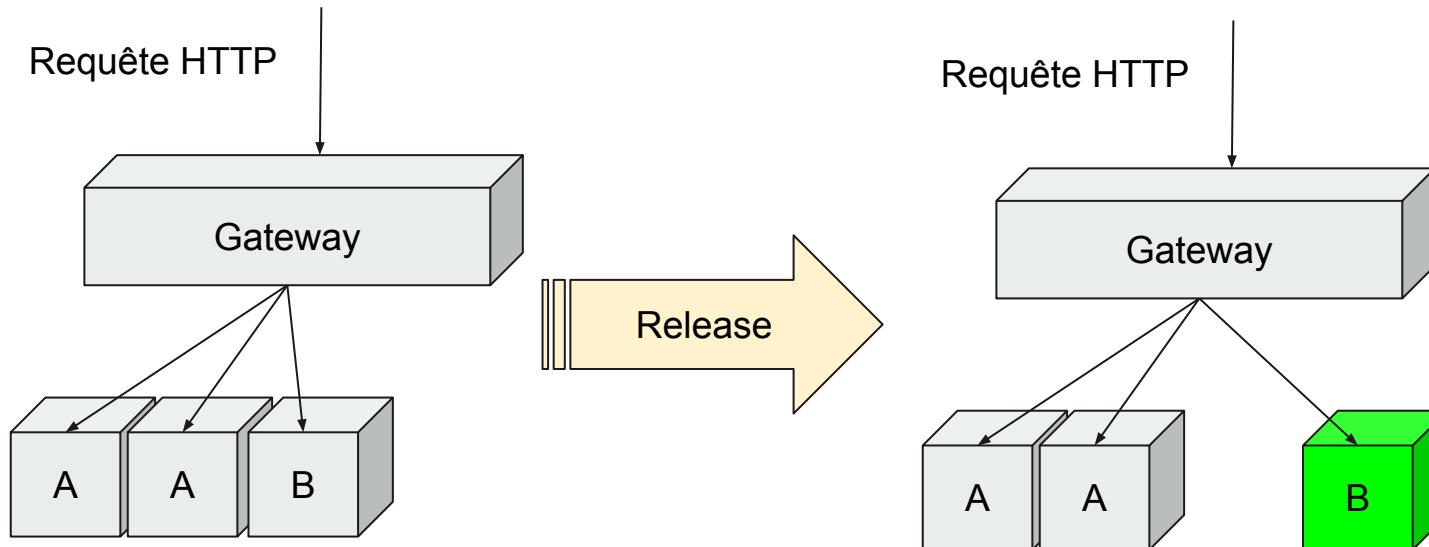
## Docker est flexible

Toutes les applications peuvent être mises dans des conteneurs Docker

Pour que ce soit facile :

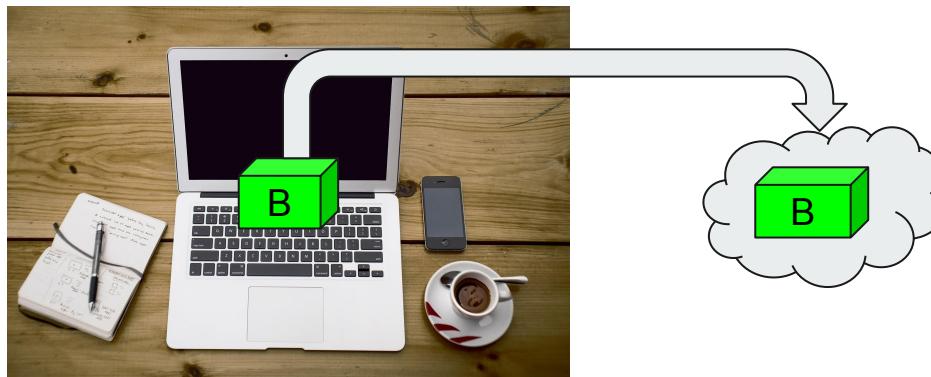
- Utilisez-vous un langage populaire ? Un framework populaire ?
- L'application est-elle majoritairement stateless ? Sinon utilisation de storage tools

# Les conteneurs sont interchangeables



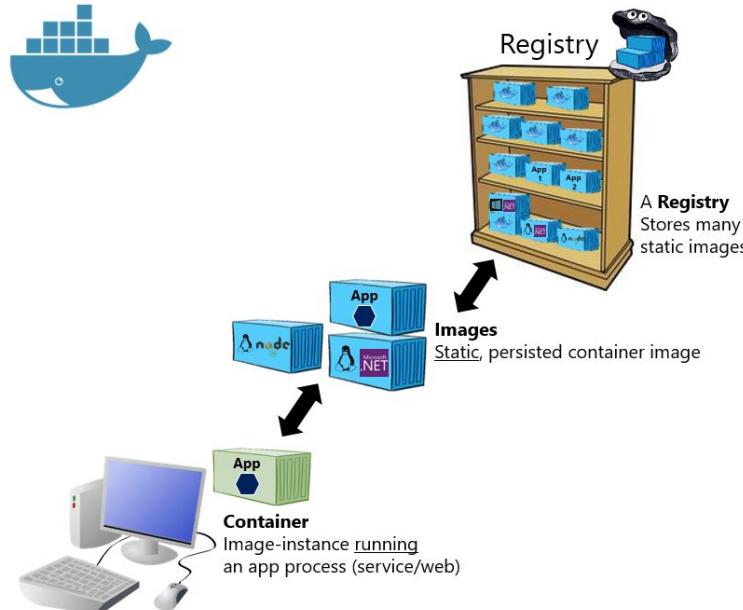
# Les conteneurs sont portables

Portability means the ability to move an application—in other words, *port it*—from one host environment to another.



Build once,  
run  
everywhere

## Basic taxonomy in Docker



Hosted Docker Registry

Docker Trusted Registry on-prem.

Docker Hub Registry

Docker Trusted Registry on-cloud

Azure Container Registry

AWS Container Registry

Google Container Registry

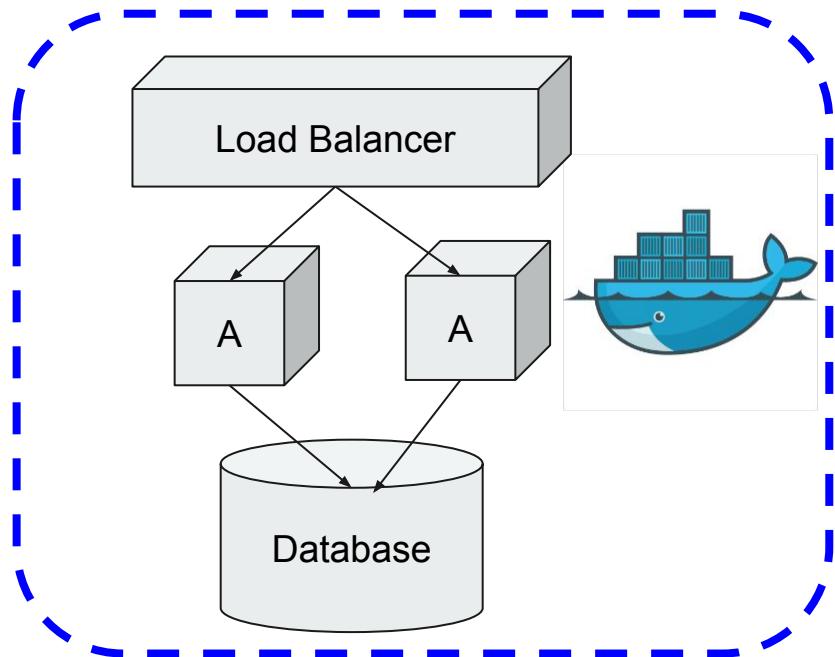
Quay Registry

Other Cloud

On-premises  
(‘n’ private organizations)

Public Cloud  
(specific vendors)

# Docker compose !





---

# Les applications containerisées sont cools

Les conteneurs sont populaires car ils sont :

- Flexibles: Toutes les applications peuvent être mises dans des conteneurs Docker.
- Légers: Les conteneurs partagent les ressources de la machine.
- Interchangeables: Les mises à jour sont déployables unitairement à la volée.
- Portables: L'image qui est exécutée sur le poste de dev est la même que celle qui est déployée
- Scalables: On peut automatiquement scaler horizontalement et verticalement nos conteneurs
- Stackables: On peut créer facilement recréer toute une stack technique.



# Faire une bonne image Docker est compliqué

Ubuntu et Debian ne sont pas conçus pour être runné à l'intérieur d'une image Docker, beaucoup de leurs fonctionnalités sont alors inutiles (init system, upstart, et tout ce qui encourage à runner Ubuntu sur une VM).

Dans Docker on ne veut pas un OS complet, on veut un OS minimaliste, c'est souvent dur.

Choix :

1. customiser une image vous même qui vous correspond
2. prendre une image toute faite, bien préparée (Phusion Passenger)
3. prendre une image hyper légère

---

# Des problèmes de sécurité

Les images Docker d'un même hôte partagent le même kernel.

- Fork bomb
- CPU
- RAM
- Pids
- UIDs
- etc.



---

## Des données essentielles

- L'image que vous utilisez peut avoir été modifié pour porter une faille.
- Vous pouvez connaître l/etc/random de la machine.





TP



---

# Installez docker

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

Hello from Docker!  
This message shows that  
your installation appears  
to be working correctly.

# Commande de base / run

```
docker run hello-world
```

```
docker ps
```

```
docker run busybox
```

```
docker ps
```

```
docker ps -a
```

```
docker run -it busybox sh
```

<nouveau terminal>

```
docker ps
```

```
docker ps -a
```

Clean-up

```
docker rm 305297d7a235 ff0a5c3750b9
```

```
docker container prune
```



# Commandes de manipulation d'images : 1

docker pull jenkins

docker run jenkins

<nouveau terminal>

docker ps

docker kill jenkins

Trouver un logiciel que l'on connaît bien sur Docker hub :

<https://hub.docker.com/search?q=&type=image>

par exemple : mysql, jenkins, Oracle JDK, wordpress, docker



---

# Commandes de manipulation d'images /2

```
docker kill <container_id>
```

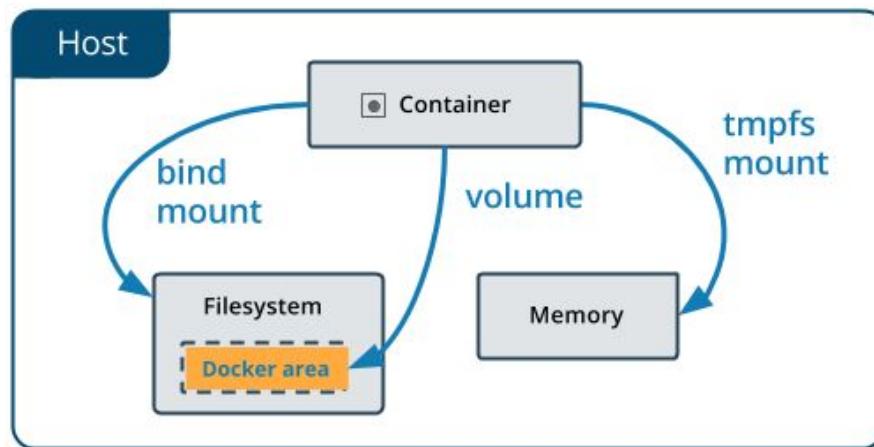
```
docker run -p 8080:8080 -p 50000:50000 -d jenkins
```

```
curl localhost:8080
```

```
docker ps
```

```
docker logs <id du conteneur>
```

# Trois solutions pour garder un état :





# Création et utilisation d'un volume

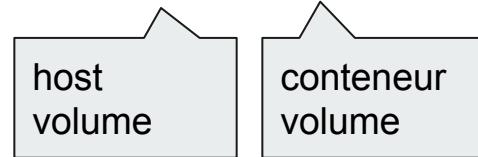
```
$ docker volume create my-vol
$ docker volume ls
$ docker volume inspect my-vol
$ docker run -d --name devtest --mount source=myvol2,target=/app nginx:latest
$ docker inspect devtest

$ #clean up
$ docker container stop devtest
$ docker container rm devtest
$ docker volume rm myvol2
```

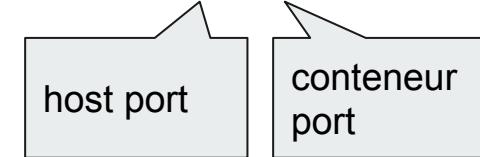
# Comment monter un volume :

Les conteneurs docker sont éphémères, les fichiers systèmes sont détruits sur l'host quand le conteneur est détruit.

```
docker run --name=nginx -d -v ~/nginxlogs:/var/log/nginx
```



```
-p 5000:80 nginx
```



*Tips : Pour monter un volume, utiliser --mount plutôt que -v*

# Gestion des ressources au runtime

Par défaut : les conteneurs prennent toutes les ressources de la machine hôte.

```
$ docker run -it  
--cpus=".5"  
ubuntu /bin/bash
```

```
$ docker run -it  
--cpu-period=1000  
00  
--cpu-quota=50000  
ubuntu /bin/bash  
$ #en micro  
secondes
```

```
$ docker run -it  
--rm --gpus all  
ubuntu  
nvidia-smi
```

```
$ docker run -it  
--pids-limit 100  
$ # fork bombs
```



---

# La sécurité Docker

4 points d'attention à garder en tête concernant la sécurité dans docker

- La sécurité intrinsèque du Kernel et du support pour les namespaces et cgroups;
- la surface d'attaque du démon docker en lui-même
- les trous de sécurité dans la manière de run un conteneur
- y a pleins d'outils supplémentaires pour la sécurité



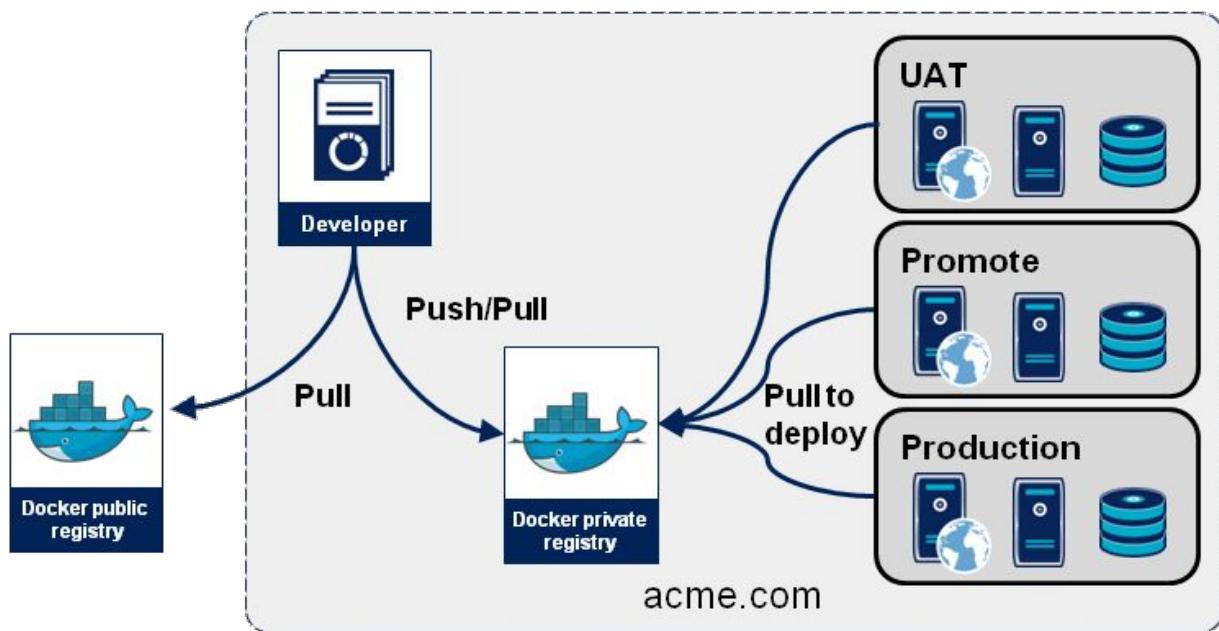
---

## Et tout le reste...

Logging, monitoring, networking, plusieurs services dans le même conteneurs,  
ci/cd, scaling horizontal, zéro downtime update...

# Docker pull, docker push

Image registry





---

TP 1 -> 21

<https://www.katacoda.com/courses/docker>



katacoda



# Cheat Sheet

---

## OBLIGATOIRE

- FROM <image>
- FROM <image>:<tag>
- FROM <image>@<digest>

ADD ["<src>", ... "<dest>"]

CMD ["<executable>","<param1>","<param2>"]  
ou  
ENTRYPOINT["<executable>","<param1>"]

## Optionnel

- MAINTAINER <name>
- EXPOSE <port>
- ENV <key> <value>
- VOLUME [<path>, ...]
- USER <user | UID>
- WORKDIR </path/to/directory>

- RUN <command>
- RUN [<executable>, "<param1>","<param2>"] (exec form)



---

Merci !

