



**#Présentation**

# DevOps

**ZZ3 – F2**

2024 - 2025



# PLAN

## 1. Docker

- Présentation
- Installer Docker
- Les containers
- Les images
- Le réseau
- Persistance des données

## 2. DevOps

- Présentation d'une chaîne d'intégration continue

# 1. Docker

## >\_ Présentation

- Installer Docker
- Les containers
- Les images
- Le réseau
- Persistance des données

# Au sujet de Docker



Docker est le résultat de nombreuses années d'évolutions dans le but de séparer en microservices les applications.

Docker est un outil qui facilite la création, le déploiement, et l'exécution d'applications dans des containers. Solomon Hykes est à l'initiative du projet. La première version a été mise à disposition en mars 2013.

Cet outil vise à être utilisé aussi bien par les développeurs que par les responsables réseaux.

Docker est à la fois le nom de l'outil mais aussi de l'entreprise qui l'exploite.

Très vite, cette solution s'est imposée dans de nombreuses entreprises en changeant définitivement les modèles d'infrastructure jusqu'à présent utilisés.

Précédemment DotCloud

Plusieurs levées de fond

Novembre 2019 :

➤ Docker Enterprise acheté par Mirantis



## Partenaires certifiés



...

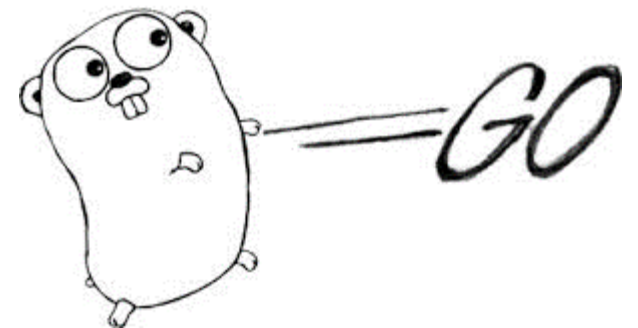
## Clients



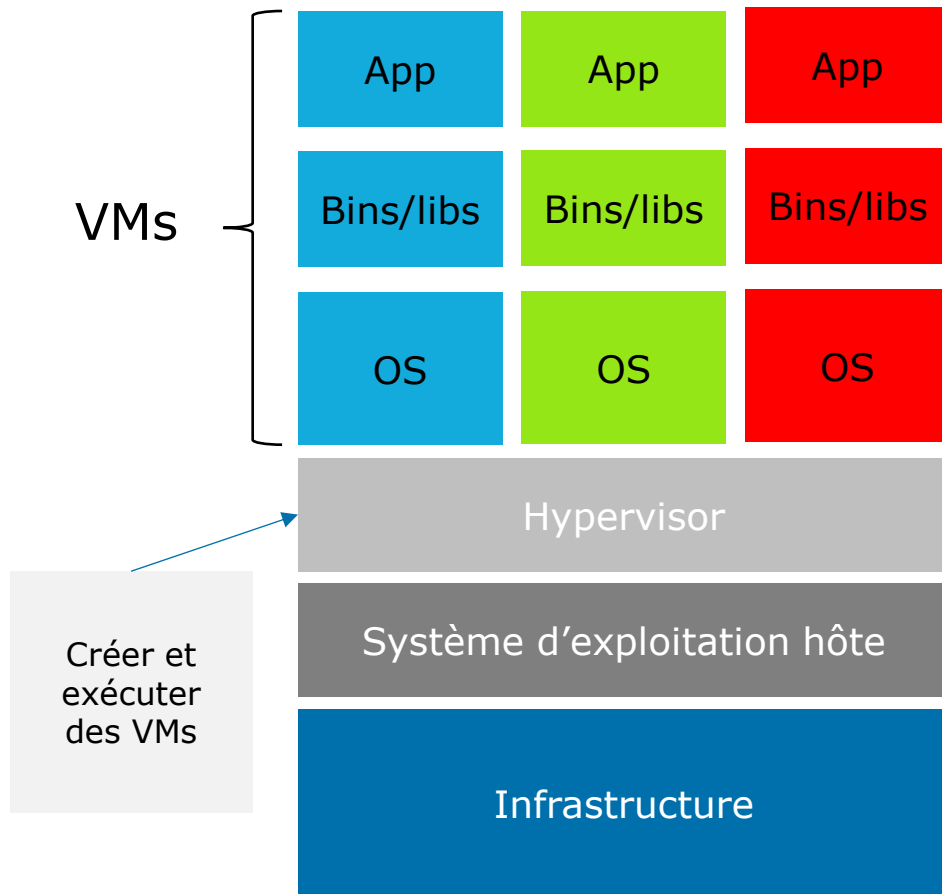
...

Docker repose sur :

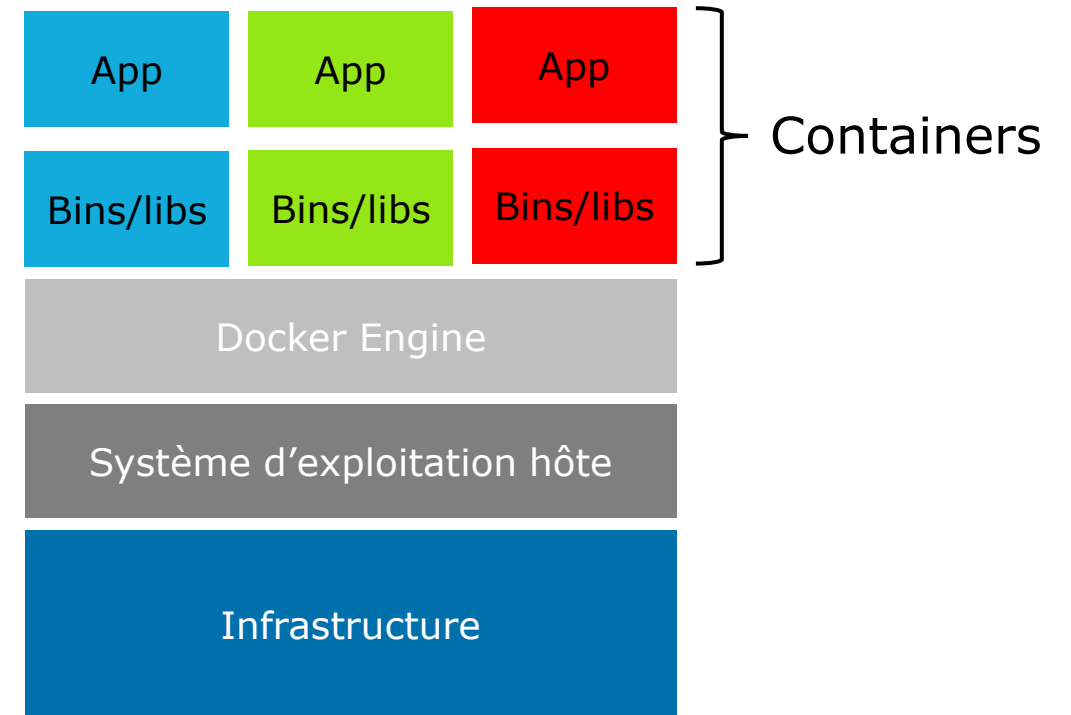
- Cgroups (LXC) et namespaces
- Écrit en Go (open source)
- LibContainer (namespace, filesystem, resources, security)



# Architectures VM vs Docker



Machine virtuelle (VM)



Docker



**32.000**

Github stars

**3000+**

Contributeurs

**8B+**

Containers Docker téléchargés

**100,000+**

Third party projects using Docker

**270+**

groupes meetup dans plus de 70 pays

**500,000+**

applications "dockerisées" dans dockerhub



# Les offres proposées



## Personal

Ideal for individual developers, education, open source communities, and small businesses.

# \$0

- Docker Desktop ⓘ
- Unlimited public repositories
- Docker Engine + Kubernetes ⓘ
- 200 image pulls per 6 hours
- Unlimited scoped tokens ⓘ

Start Now

## Pro

Includes pro tools for individual developers who want to accelerate their productivity.

# \$5

 /month

### ← Everything in Personal plus:

- Docker Desktop ⓘ
- Unlimited private repositories
- 5,000 image pulls per day
- 5 concurrent builds ⓘ
- 300 Hub vulnerability scans

Buy Now

Billed annually for \$60.

## Team

Ideal for teams and includes capabilities for collaboration, productivity and security.

# \$7

 /user/month  
Start with minimum 5 users for \$25.

### ← Everything in Pro, plus:

- Docker Desktop ⓘ
- Unlimited teams
- 15 concurrent builds ⓘ
- Unlimited Hub vulnerability scans
- Add users in bulk
- Audit logs ⓘ

Buy Now

Billed annually starting at \$300.

## Business

Ideal for medium and large businesses who need centralized management and advanced security capabilities.

# \$21

 /user/month

### ← Everything in Team, plus:

- Hardened Docker Desktop
- Enhanced Container Isolation
- Settings management
- Centralized management
- Registry Access Management
- Image Access Management
- Single Sign-On (SSO)
- SCIM user provisioning
- VDI support
- Purchase via invoice
- Volume Pricing Available

Contact Sales

Buy Now

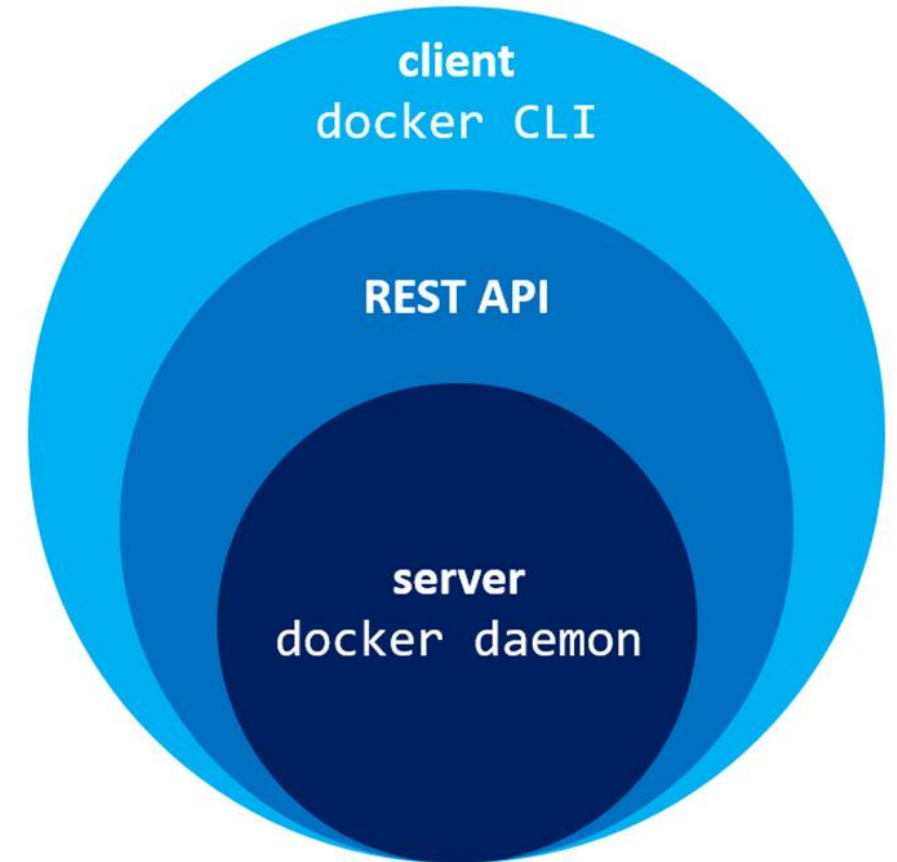
Our [Docker Subscription Service Agreement](#) includes a change to the terms for Docker Desktop

- It **remains free** for small businesses (fewer than 250 employees AND less than \$10 million in annual revenue), personal use, education, and non-commercial open source projects.
- It requires a paid subscription ([Pro](#), [Team](#) or [Business](#)), for as little as \$5 per user per month, for professional use in larger businesses.
- The effective date of these terms is August 31, 2021. **There is a grace period until January 31, 2022** for those that will require a paid subscription to use Docker Desktop.
- The Docker Pro, Docker Team, and Docker Business subscriptions **now include** commercial use of Docker Desktop.
- Check out our [FAQ](#) for more information. Or read our [latest blog](#).

Consultation des prix  
<https://www.docker.com/pricing>



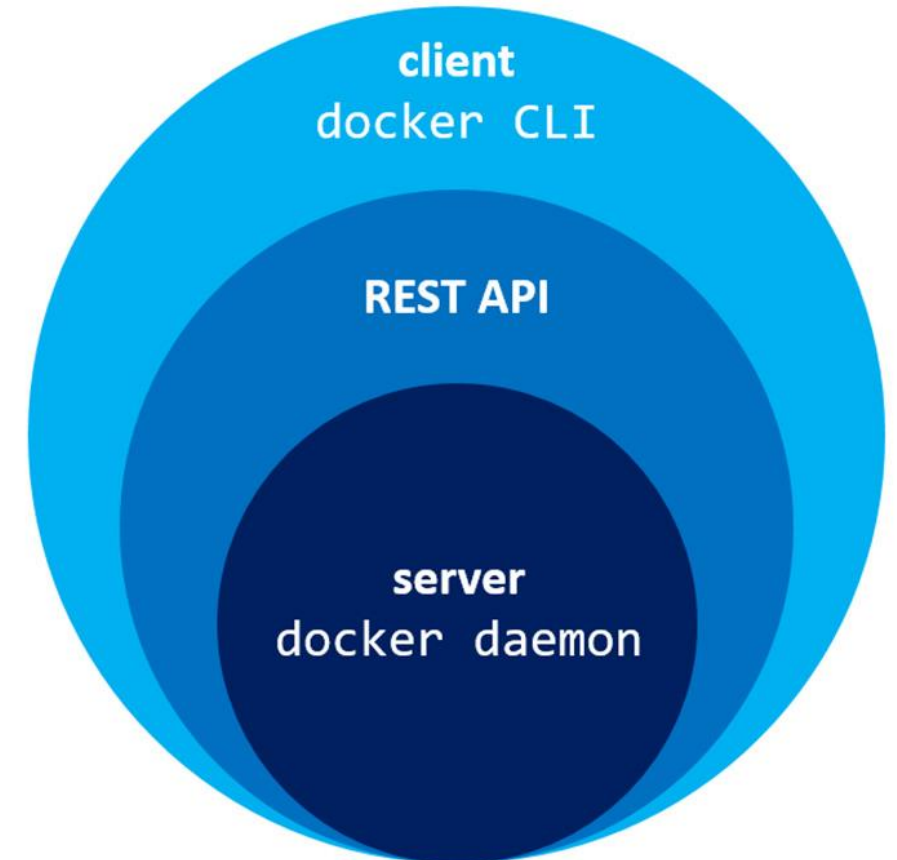
- Docker CLI : interface en ligne de commandes
- REST API
- Processus démon (tâche de fond)



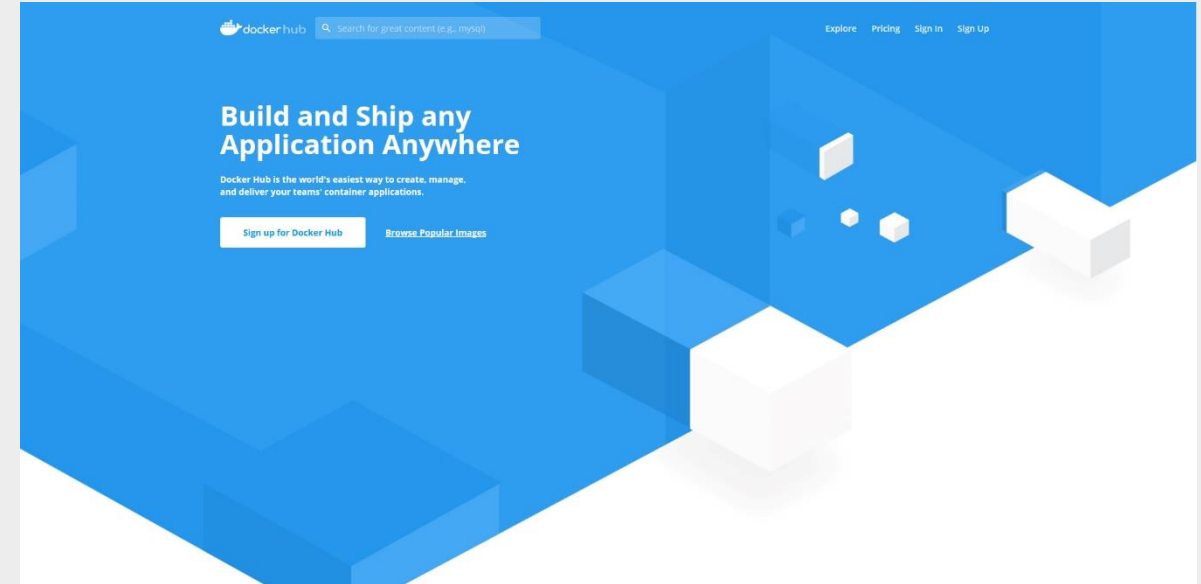


### Toutes les fonctionnalités du Docker Engine sont exposées et peuvent-être pilotées via l'API

- API en REST
- 2 SDK officiels proposés : Python et en Go
- Librairies non officielles : C, C#, C++, Java, PHP...
- Dernière version disponible : 1.28
- Requêtes en POST et en GET
- Réponse en JSON
- Docker Remote API -> Docker Engine API



- Portail d'images Docker, extensions et plugins
- + 100 000 images
- Dépôts officiels et d'utilisateurs
- Dépôts publics et privés
- Communauté active
- Offre d'un docker Hub d'entreprise
- Esprit "open source"
- Mise à disposition par les entreprises de leurs applications dans des containers
- Commercialisation des containers
- Vérifications et certifications de l'origine des images



URL : <https://hub.docker.com/>

Démonstration du site

# 1. Docker

- Présentation

**>\_ Installer Docker**

- Les containers
- Les images
- Le réseau
- Persistance des données



## Prérequis (Repository)

Préparation de la distribution Linux Ubuntu

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME)" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

URL : <https://docs.docker.com/engine/install/ubuntu/>



## Installation

### Docker Engine sous Ubuntu

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Si vous rencontrez le message suivant : « docker: cannot connect to the docker daemon at unix:///var/run/docker.sock. is the docker daemon running? »

```
sudo service docker start
```

## Compatibilité (64 bits)

Ubuntu Lunar 23.04

Ubuntu Kinetic 22.10

Ubuntu Jammy 22.04 (LTS) <- lsb\_release -a

Ubuntu Focal 20.04 (LTS)



URL : <https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/>

### Auparavant

- Docker Toolbox (anciens systèmes)
- Docker For Windows

### Prérequis

- Windows 10 64-bit: Pro, Enterprise, or Education (Build 16299 or later)
- Activation de la virtualisation : Hyper-V virtualization + Containers Windows

### Installation

Elle comprend :

- Docker Engine
- Docker CLI Client
- Docker Compose
- Docker Machine
- Kitematic

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
• Reboot, then enable the containers feature.
Enable-WindowsOptionalFeature -Online -FeatureName Containers -All
```



URL : <https://docs.docker.com/docker-for-windows/install/>



### Problème lors du lancement de Docker

1. Open "Window Security"
  2. Open "App & Browser control"
  3. Click "Exploit protection settings" at the bottom
  4. Switch to "Program settings" tab
  5. Locate "C:\WINDOWS\System32\vmcompute.exe" in the list and expand it
  6. Click "Edit"
  7. Scroll down to "Code flow guard (CFG)" and uncheck "Override system settings"
  8. Start vmcompute from powershell "net start vmcompute"
- B. Run powershell command MOFCOMP %SYSTEMROOT%\System32\WindowsVirtualization.V2.mof



URL : <https://docs.docker.com/docker-for-windows/install/>

# Docker

## Hyper-V (ou Windows Server Virtualisation)



Disponible sur Windows : 7,8, 10 et 11

Système de virtualisation, Hyperviseur 64 bits : gérer et héberger des machines virtuelles

Virtualisation matériel

Supporté :

- Windows
- Linux
- BSD

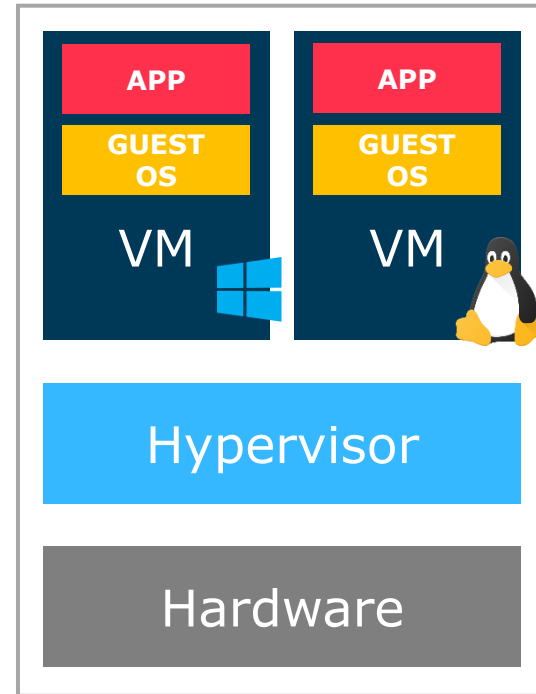




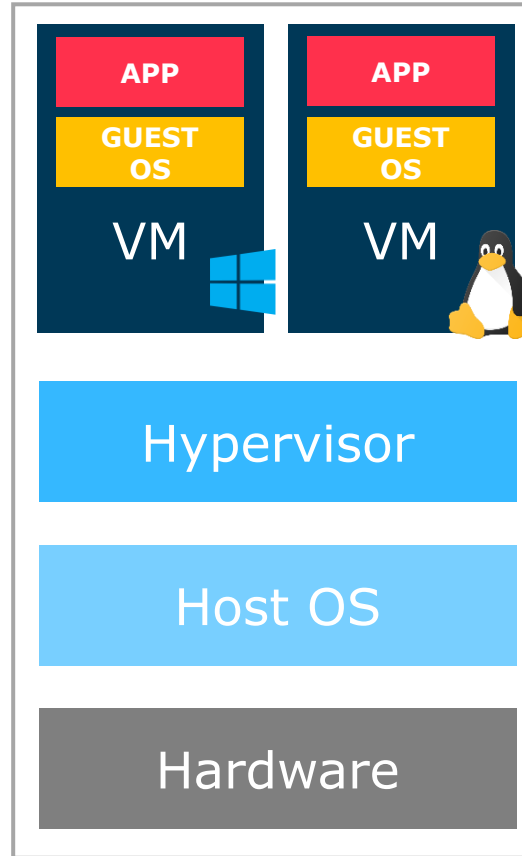
Microsoft  
Hyper-V

- L'Hypervisor est exécuté directement sur le matériel de la machine
- Sécurité renforcée
- L'accélération matérielle doit être activée via le BIOS

Hypervisors : Mware ESXi, Microsoft Hyper-V server, open source KVM



Type 1 Hypervisor  
(Bare-Metal  
Architecture)



Type 2 Hypervisor  
(Hosted  
Architecture)



VirtualBox

- L'Hypervisor est installé sur le système d'exploitation de la machine
- Risque potentiel de sécurité via le système d'exploitation hôte
- Utilisation de l'Accélération matérielle si disponible

Hypervisors : VMware Fusion, Oracle VM VirtualBox, Oracle VM Server for x86

# Docker

## Kitematic (docker Toolbox)



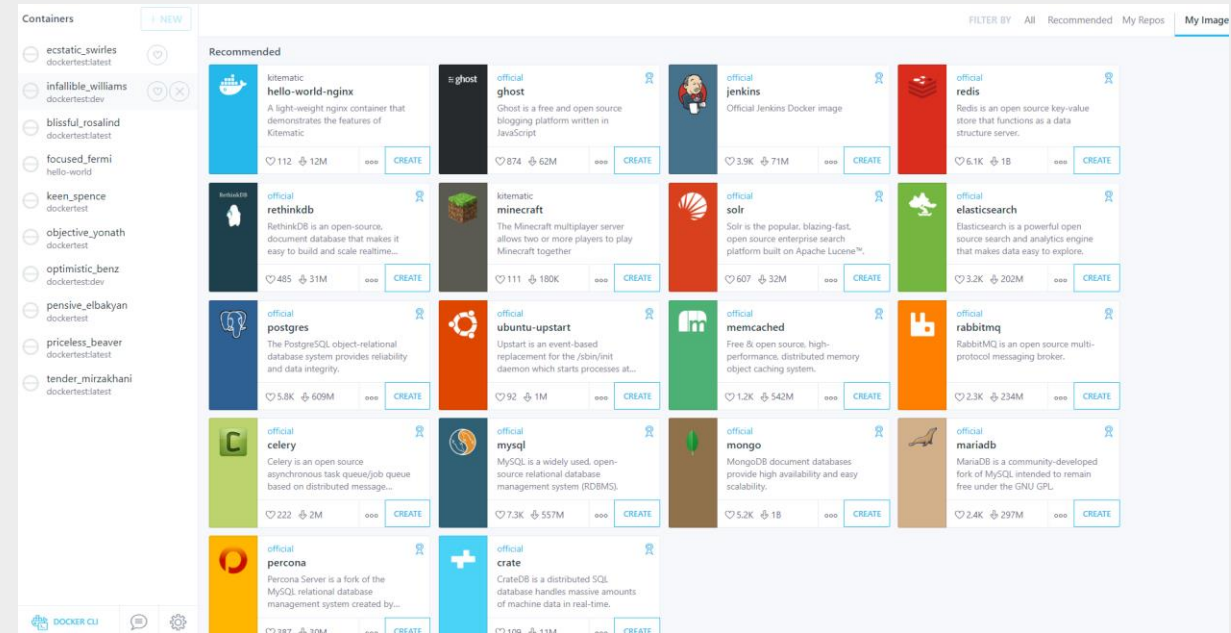
Développé en 2013

Open Source

Compatible Windows et Linux

Interface graphique qui permet de déployer des containers Docker

Intégration de Docker Hub



URL : <https://kitematic.com/>

Fonctionnalité obsolète

# 1. Docker

- Présentation
- Installer Docker

## >\_ Les containers

- Les images
- Le réseau
- Persistance des données



### Qu'est-ce qu'un container ?

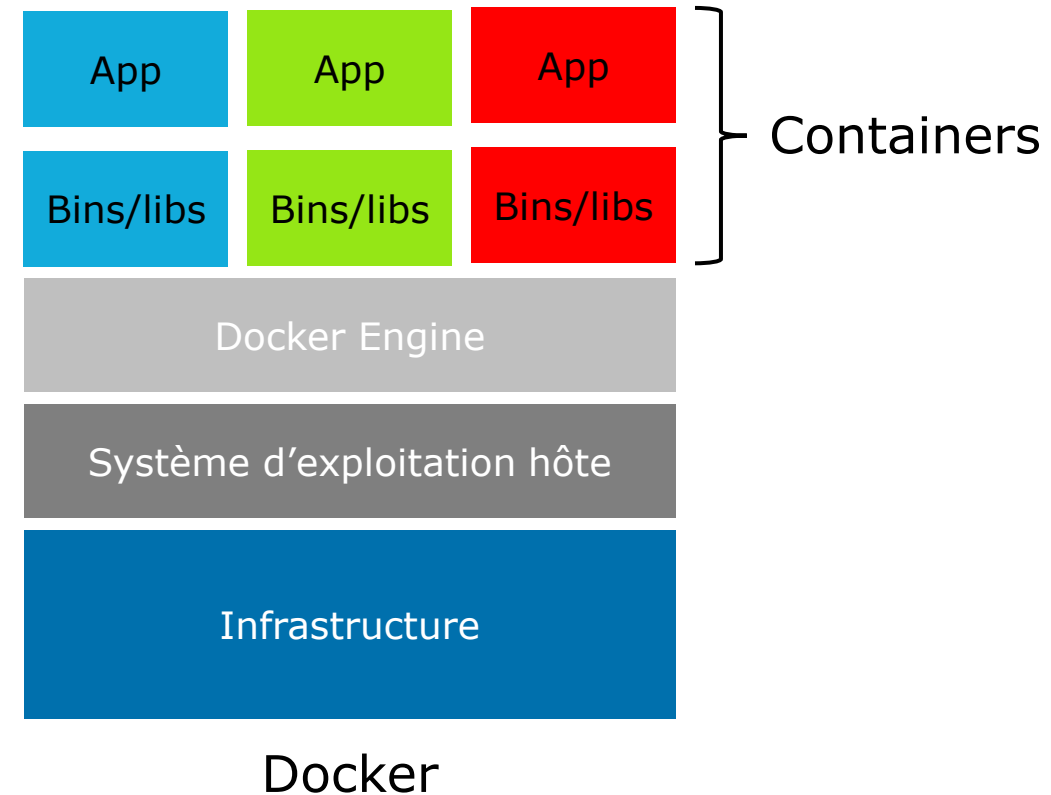
Un container est un environnement qui permet d'isoler une application. Il repose sur LXC et Cgroups => gère les ressources matériels (RAM, CPU).

C'est un ensemble de processus tournant dans un système de fichiers en lecture/écriture.

### Différences VM & containers

VM : isole tout un système, et dispose de ses propres ressources

Container : partage les ressources du systèmes hôte



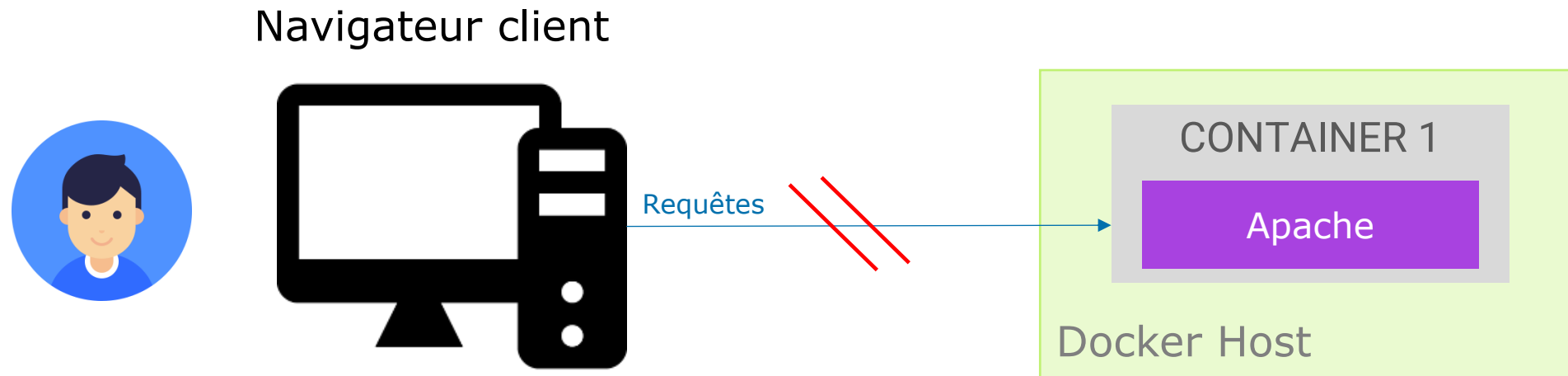


Les différents statuts d'un container :

- Created
- Restarting
- Running
- Removing
- Paused
- Exited
- Dead

Affichage de tous les containers

```
docker ps -a
```



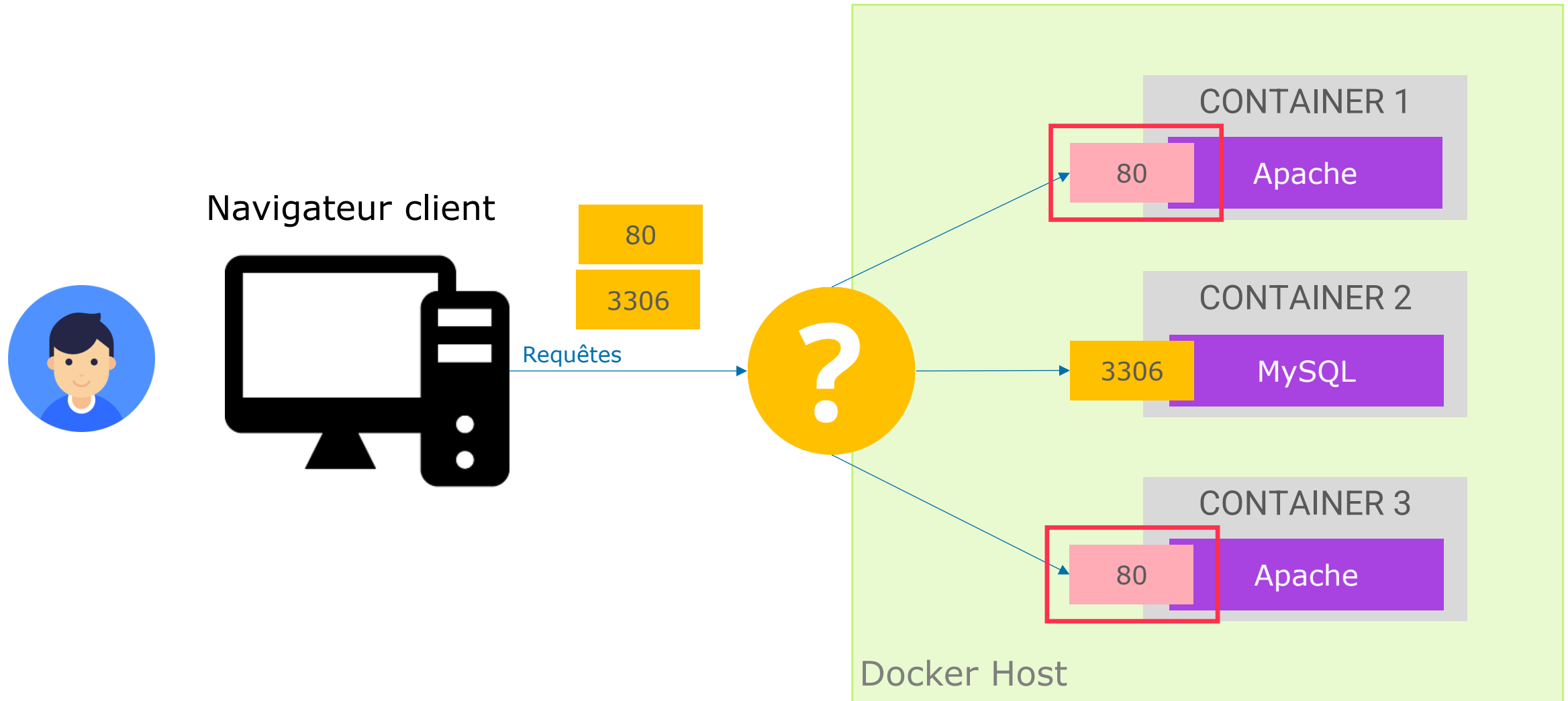
**Par défaut, lors de la création d'un container, les ports ne sont pas ouverts vers l'extérieur (isolation)**

Pour rendre disponible le port du service du container vers l'extérieur de Docker (action : publish), on utilise les flags : `--publish` ou `-p`



# Docker

## Les containers – Réseau : ports

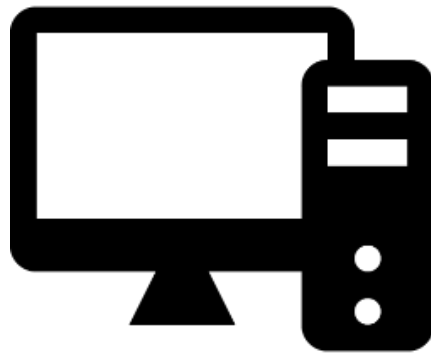


# Docker

## Les containers – Réseau : ports



Navigateur client



Requêtes

Mapper les ports

8081

`-p 8081:80`

80

CONTAINER 1

Apache

3309

`-p 3309:3306`

3306

CONTAINER 2

MySQL

8082

`-p 8082:80`

80

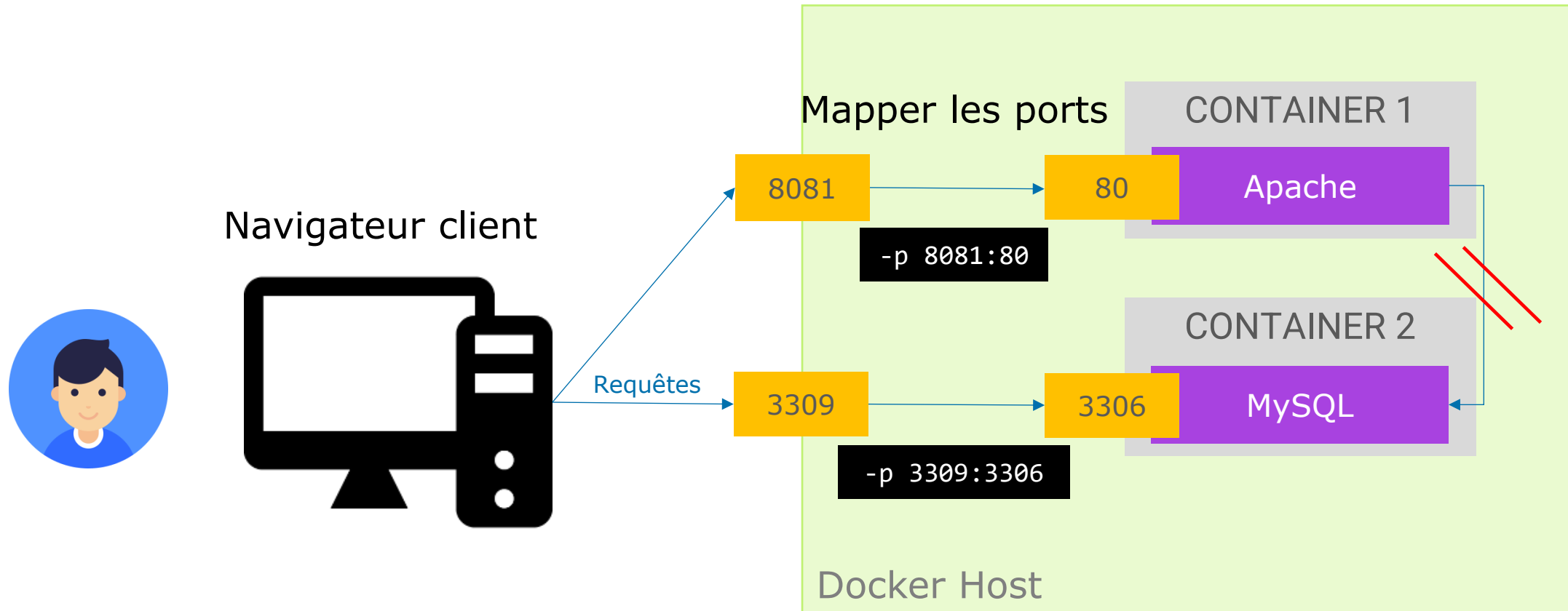
CONTAINER 3

MySQL

Docker Host

# Docker

## Les containers – Réseau : ports



**Par défaut, les containers ne se connaissent pas et ne peuvent donc pas communiquer entre eux**

### Par défaut :

- **Lors de la création d'un container, les ports ne sont pas ouverts vers l'extérieur**
- **les containers ne se connaissent pas et ne peuvent donc pas communiquer entre eux**

```
-p PORT_EXTERNE:PORT_INTERNE
```

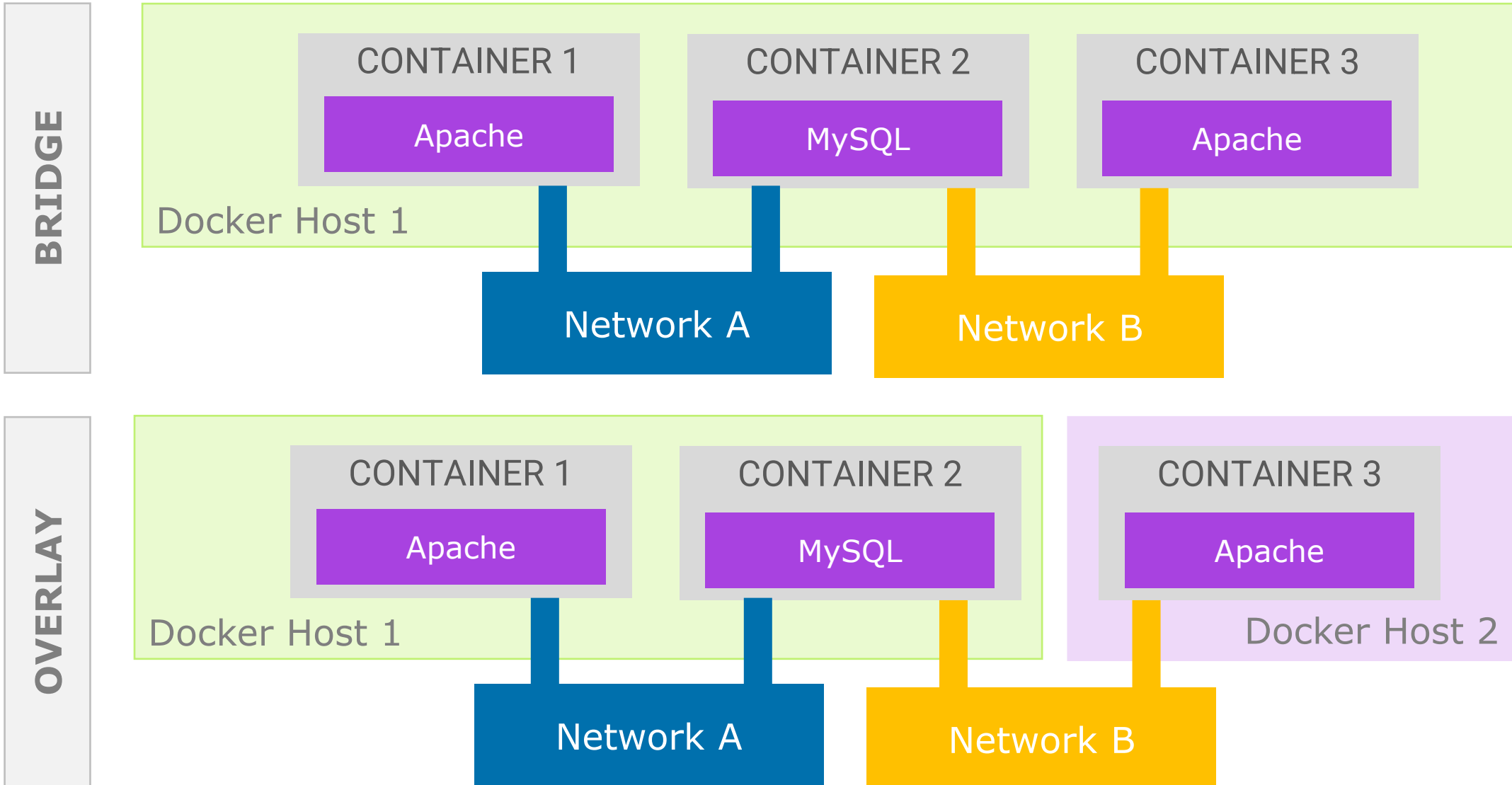


Les différents types de réseaux disponibles :

- Bridge (ou default bridge network) : appliqué aux containers qui utilisent le même processus daemon Docker de l'hôte.
- Overlay
- Macvlan

```
docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
5e1a1805f5d8	bridge	bridge	local
6a726b131ed3	host	host	local
fb6646111a8	none	null	local





### Création d'un bridge

```
docker network create -d bridge my_bridge
```

### Suppression d'un bridge

```
docker network rm my_bridge
```

### Ajouter mes containers à mon réseau « my\_bridge »

```
docker run --name container_1 --net my_bridge image
```

```
docker run --name container_2 --net my_bridge image
```

### Docker via Windows

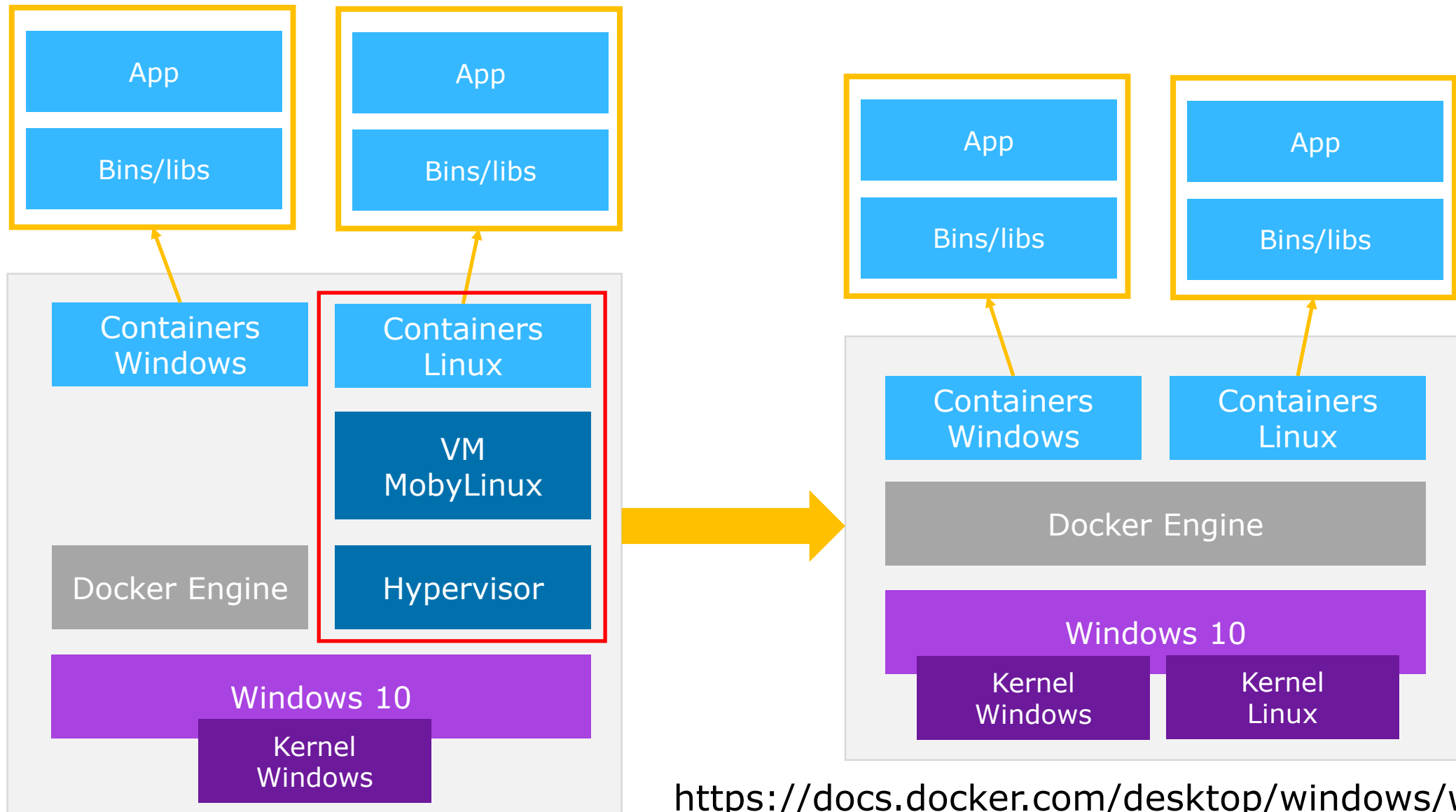
- Containers Windows
- Containers Linux :
  - via la VM MobyLinux,
  - via le kernel Linux dans Windows 10

Exemple d'images Windows :  
[mcr.microsoft.com/windows/servercore/iis](https://mcr.microsoft.com/windows/servercore/iis)  
[mcr.microsoft.com/windows/nanoserver:1903](https://mcr.microsoft.com/windows/nanoserver:1903)

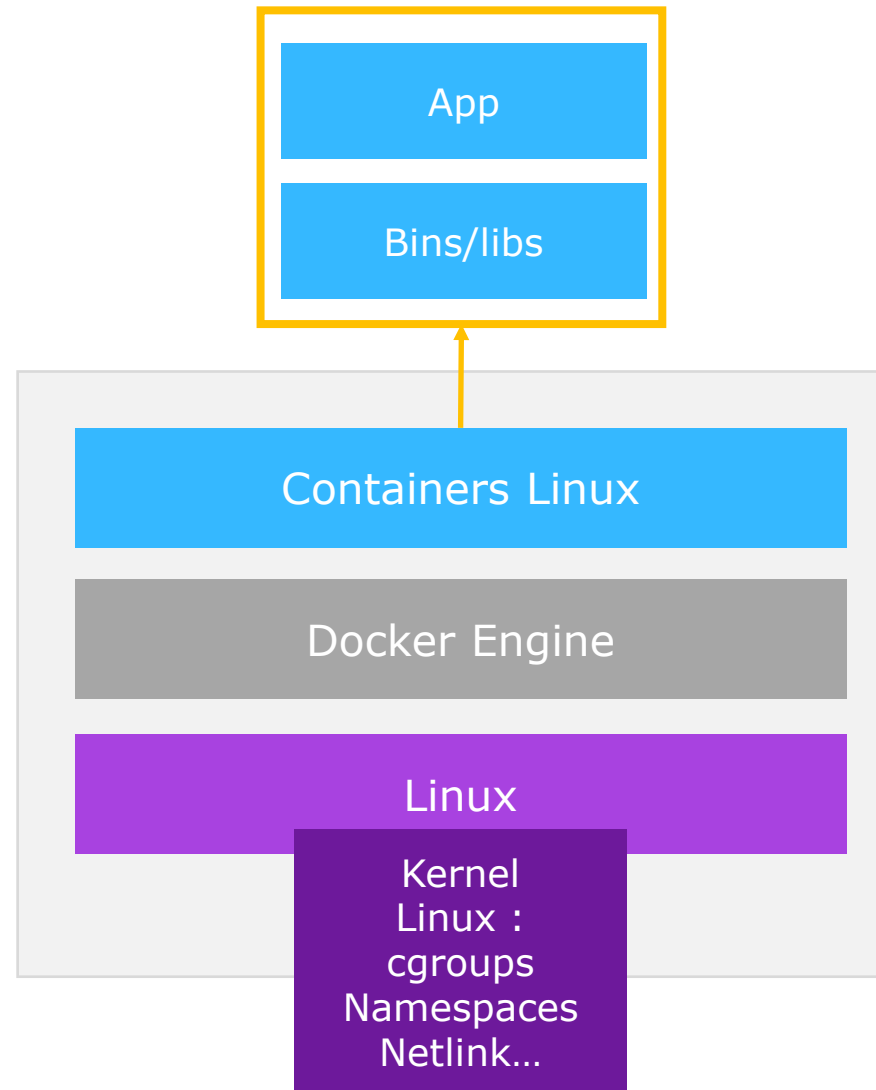
### Docker via Linux

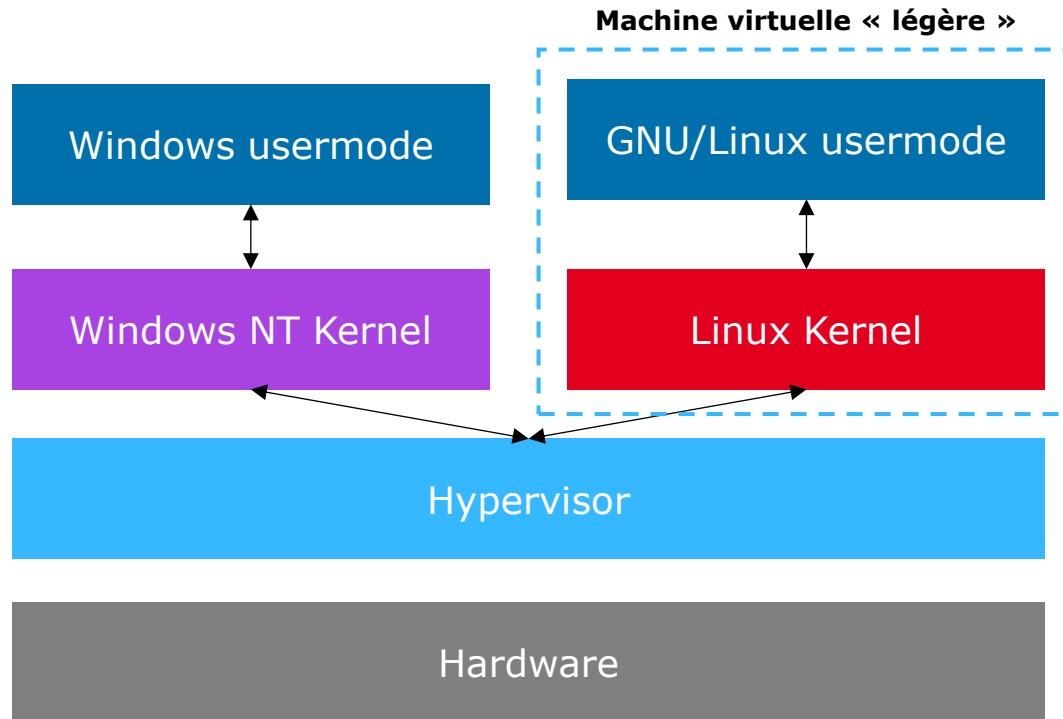
- Container Linux





<https://docs.docker.com/desktop/windows/wsl/>





- Auparavant WSL 1
- Exécution native d'outils linux sous Windows 10
- Intégration d'un noyau linux complet dans Windows 10
- Open source et sous licence GPL



Activation du « Windows Subsystem for Linux »:

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

Activation du «Virtual Machine feature »:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

Téléchargement « Linux kernel update package » :

<https://docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package>

Définition par défaut du WSL 2

```
wsl --set-default-version 2
```



```
docker pull mcr.microsoft.com/windows/servercore/iis
```

```
Using default tag: latest  
latest: Pulling from windows/servercore/iis  
no matching manifest for linux/amd64 in the manifest list entries
```

3 raisons possibles :

- Erreur quand on utilise le tag « Latest »
- Le tag n'est pas disponible
- Image n'est pas compatible avec l'OS/Arch de l'hôte Docker

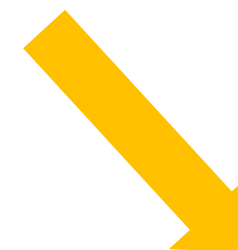
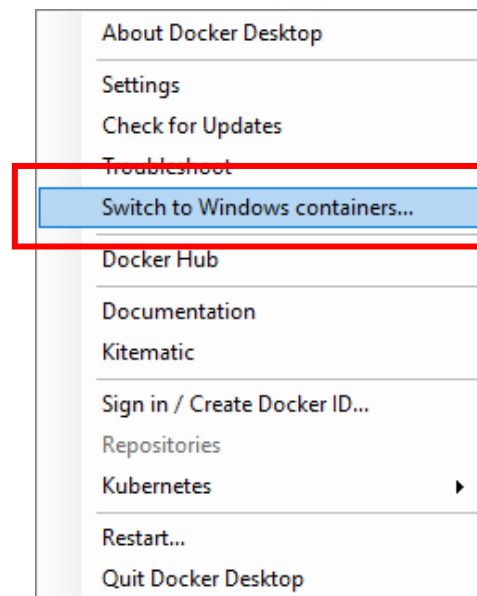


Connaître dans quel mode est Docker

```
docker version
```

```
Client: Docker Engine - Community
Version:      19.03.4
API version:  1.40
Go version:   go1.12.10
Git commit:   9013bf5
Built:        Thu Oct 17 23:44:48 2019
OS/Arch:      windows/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version:      19.03.4
API version:  1.40 (minimum version 1.12)
Go version:   go1.12.10
Git commit:   9013bf5
Built:        Thu Oct 17 23:50:38 2019
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      v1.2.10
GitCommit:    b34a5c8af56e510852c35414db4c1f4fa6172339
runc:
Version:      1.0.0-rc8+dev
GitCommit:    3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
Version:      0.18.0
GitCommit:    fec3683
```



```
OS/Arch:      windows/amd64
```



Démarrer un container

```
docker run -name CONTAINER_NAME -d -it IMAGE_NAME
```

Lister les containers en cours d'exécution

```
docker ps
```

Afficher les logs d'un container

```
docker container logs ID_CONTAINER
```

Se connecter à un container

```
docker exec -it CONTAINER_ID bash
```

Supprimer un container

```
docker rm -f CONTAINER_ID
```

Suppression de tous les containers

```
docker rm $(docker ps -a -q)
```



### 1. Vérification de l'installation

Exécution du container hello-world

```
docker run hello-world
```

Réponse attendue

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

Vérification de l'installation du Powershell

Consultation du Dashboard

### 2. Mon premier container

Déployer un container Debian Jessie (nom : container\_jessie) puis se connecter dessus



# 1. Docker

- Présentation
- Installer Docker
- Les containers

## >\_ Les images

- Le réseau
- Persistance des données

Une image est un système de fichiers en lecture seule. Le Dockerfile est un document texte qui contient toutes les commandes qu'un utilisateur peut appeler.



TAG par défaut : latest

Commande pour construire une image

```
docker build -t IMAGE_NAME .
```

```
docker build -t IMAGE_NAME[:TAG] .
```



### Liste des commandes

FROM  
MAINTAINER -> deprecated : LABEL maintainer  
RUN  
CMD  
EXPOSE  
ADD (Comme COPY + URL)  
COPY  
ENTRYPOINT  
VOLUME  
USER  
WORKDIR  
ONBUILD  
ENV

Documentation

<https://docs.docker.com/engine/reference/builder/>



Syntaxe

Dockerfile

```
# Commentaire INSTRUCTION arguments
```

**Dockerfile**  
Fichier texte



Fichier .txt

Dockerfile

```
FROM Ubuntu
MAINTAINER David Raynaud <david.raynaud@capgemini.com>

COPY php.ini /usr/local/etc/php/

# Ajout des fichiers sources du site
ADD www /var/www/html

# Ajout des fichiers sources du site
ADD main.cf /etc/postfix

RUN a2enmod rewrite
```



Exclusion de fichiers et répertoires suivant des patterns définis dans .dockerignore  
ADD ou COPY

.dockerignore

```
# comment
*/temp*
*/*/temp*
temp?
```

Règle	Exemple
#comment	Ignoré
*/temp*	/somedir/temporary.txt
*/*/temp*	/somedir/subdir/temporary.txt
temp?	/tempa /tempb



Une mauvaise utilisation du fichier peut entraîner une image non conforme.

Documentation : <https://docs.docker.com/engine/reference/builder/#dockerignore-file>



Le Docker Registry ou annuaire permet de stocker et de mettre à disposition des images Docker. Il est open source. Il offre la possibilité d'obtenir un annuaire privé ou public.

Tag une image

```
docker tag my_image[:TAG] $DOCKER_ID_USER/my_image[:TAG]
```

Connexion au Registry

```
docker login
Username : $DOCKER_ID_USER
Password : $DOCKER_PASSWORD_USER
Login Succeeded
```

Envoyer l'image sur le registry

```
docker push $DOCKER_ID_USER/my_image[:TAG]
```

# Docker

## Les images – Docker Registry



### Récupérer une image

```
docker pull $DOCKER_ID_USER/my_image
```

Explicitement : `docker pull`  
Implicitement : `docker run`

### Commande Build

```
docker build -t my_image .
```

### Tag une image

```
docker tag my_image $DOCKER_ID_USER/my_image:BuildID
```

BuildId : numéro de build généré lors de la construction du projet



containers

Container / VM  
Containers Linux/Windows  
Réseau

images

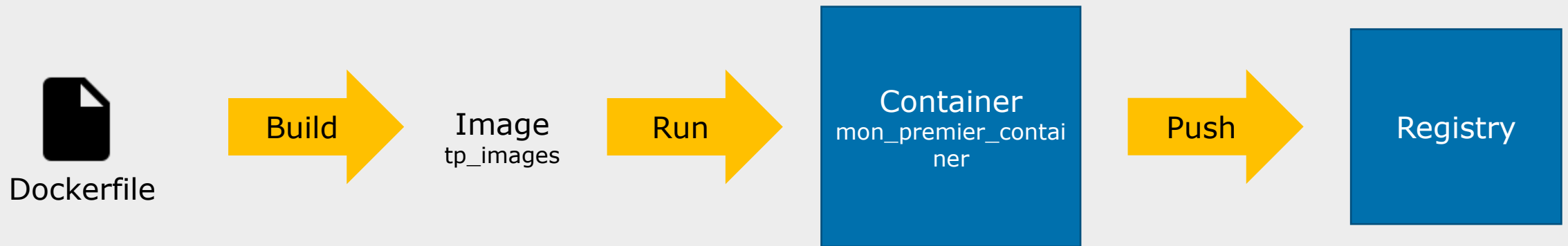
Dockerfile  
Docker registry





Tp : Création d'une image puis de son container et envoi dans le registry

### Etapes

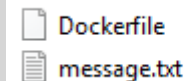


### Informations

Dockerfile :

- Image : debian 8
- Ajouter l'utilisateur « formation »
- Créer et déposer le fichier « message.txt » dans « /home/formation »
- Changer les droits du fichier en 777

#### Arborescence du TP



Dockerfile  
message.txt

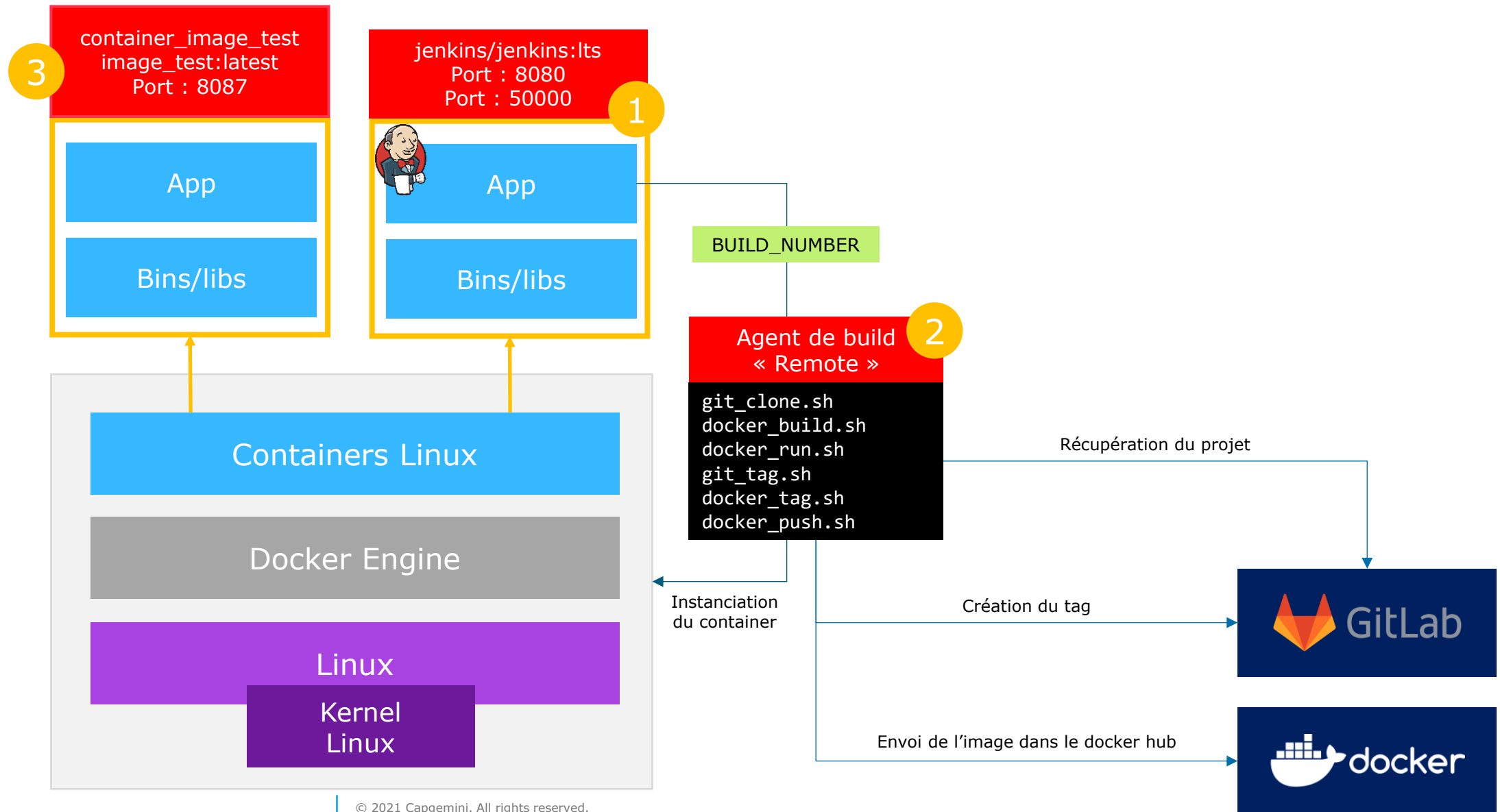
#### DockerHub

Account : formationdocker11  
Password : formation\_cap

Image : debian jessie 8  
Utilisateur : formation

# Docker

## Les images – TP Jenkins





### Jenkins : configuration du job

Saisissez un nom

» Ce champ ne peut pas être vide. Veuillez saisir un nom valide et appuyer sur OK.



#### Construire un projet free-style

Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.

# Docker

## Les images – Tp Jenkins



### Jenkins : création du job

#### Build

##### Exécuter une ligne de commande batch Windows

Commande

```
D:  
cd /ISIMA/jenkins  
git_clone.sh  
docker_build.sh %BUILD_NUMBER%  
docker_run.sh %BUILD_NUMBER%  
git_tag.sh %BUILD_NUMBER%  
docker_tag.sh %BUILD_NUMBER%  
docker_push.sh %BUILD_NUMBER%
```

[Voir la liste des variables d'environnement disponibles](#)

Ajouter une étape au build ▾

BUILD\_NUMBER provient de la liste des variables d'environnement disponibles

- docker\_build.sh
- docker\_push.sh
- docker\_run.sh
- docker\_tag.sh
- git\_clone.sh
- git\_tag.sh

X

?


Avancé...




### Jenkins : ajout de l'agent de build

 Nouveau Item

 Utilisateurs

 Historique des constructions

 Relations entre les builds

 Vérifier les empreintes numériques

 Administrer Jenkins

 Mes vues

 Lockable Resources

 Identifiants

 New View



#### Gérer les nœuds

Ajouter, supprimer, contrôler et monitorer les divers nœuds que Jenkins utilise pour exécuter les jobs.



### Jenkins : configuration de l'agent de build

Nom	<input type="text" value="Agent de build Git et Docker"/>	?
Description	<input type="text"/>	?
Nb d'exécuteurs	<input type="text" value="1"/>	?
Répertoire de travail du système distant	<input type="text" value="C:\Jenkins\"/>	?
Étiquettes	<input type="text"/>	?
Utilisation	<input type="text" value="Utiliser ce noeud autant que possible"/>	?
Méthode de lancement	<input type="text" value="Launch agent by connecting it to the master"/>	?
	<input type="checkbox"/> Disable WorkDir	?
	Custom WorkDir path <input type="text"/>	?
	Internal data directory <input type="text" value="remoting"/>	?
	<input type="checkbox"/> Fail if workspace is missing	?
	<input type="checkbox"/> Use WebSocket	?
	<input type="button" value="Avancé..."/>	
Disponibilité	<input type="text" value="Keep this agent online as much as possible"/>	?

#### Propriétés du nœud

- ☐ Disable deferred wipeout on this node ?
- ☐ Emplacement des outils

Enregistrer



### Jenkins : déploiement de l'agent de build

#### Agent Agent de build Git et Docker

[Marquer ce nœud comme temporairement hors ligne](#)

Exécuter l'agent à partir de l'interprète de commandes

```
curl -sO http://localhost:8080/jnlpJars/agent.jar  
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/Agent%20de%20build%20Git%20et%20Docker/jenkins-agent.jnlp -secret 2a08f26f2f5b90563fb91224bfce9728eb9d42c4ba23de777217611e7fd61555 -workDir "C:\\Jenkins\\"
```

Or run from agent command line, with the secret stored in a file:

```
echo 2a08f26f2f5b90563fb91224bfce9728eb9d42c4ba23de777217611e7fd61555 > secret-file  
curl -sO http://localhost:8080/jnlpJars/agent.jar  
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/Agent%20de%20build%20Git%20et%20Docker/jenkins-agent.jnlp -secret @secret-file -workDir "C:\\Jenkins\\"
```

[Enlever les « »](#)

#### Projets rattachés à Agent de build Git et Docker

Aucun



```
nov. 16, 2022 11:59:34 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connected
```

# 1. Docker

- Présentation
- Installer Docker
- Les containers
- Les images

## >\_ Le réseau

- Persistance des données





Lister les réseaux

```
docker network ls
```

Créer un réseau

```
docker network create <NETWORK>
```

Bridge par défaut

Lors de la création d'un container, le paramètre `--network` permet d'indiquer le réseau à utiliser.

```
docker run --network=<NETWORK>
```

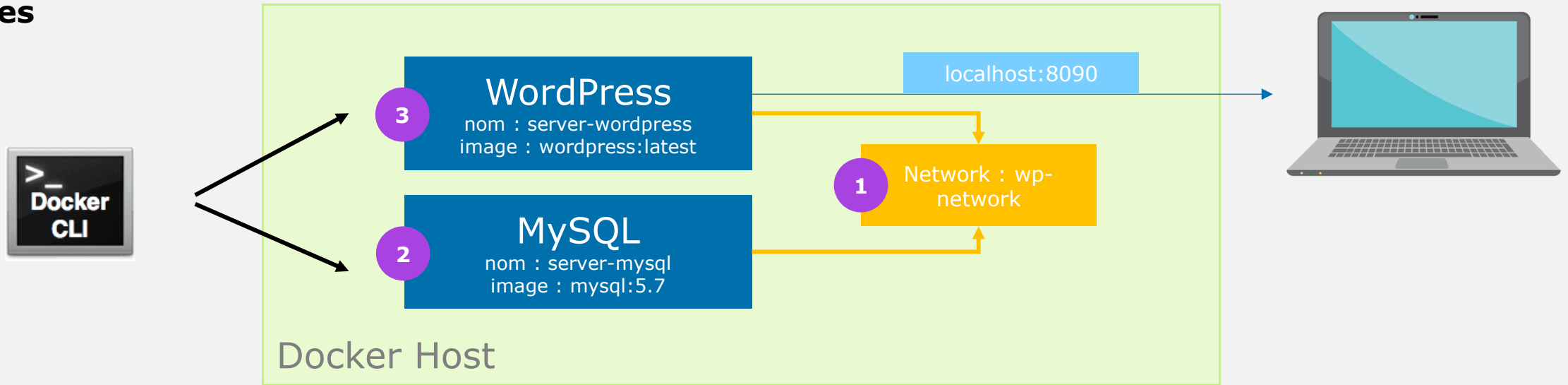
ou

```
docker run --net=<NETWORK>
```



**Objectif :** Déployer la solution WordPress.

### Etapes



### Informations

- 1 - Création du réseau « wp-network »
- 2 - Instanciation du container « server-mysql » : base de données « wordpress »
- 3 - Instanciation du container « server-wordpress »





## Variables d'environnements à paramétrer

```
MYSQL_ROOT_PASSWORD=root  
MYSQL_DATABASE=db_wordpress  
MYSQL_USER=user_wordpress  
MYSQL_PASSWORD=password_wordpress
```

```
WORDPRESS_DB_HOST=server-mysql:3306  
WORDPRESS_DB_NAME=db_wordpress  
WORDPRESS_DB_USER=user_wordpress  
WORDPRESS_DB_PASSWORD=password_wordpress
```



1

```
docker network create wp-network
```

2

```
docker run --name server-mysql -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=db_wordpress -e  
MYSQL_USER=user_wordpress -e MYSQL_PASSWORD=password_wordpress --network=wp-network -d mysql
```

3

```
docker run --name server-wordpress -e WORDPRESS_DB_HOST=server-mysql:3306 -e WORDPRESS_DB_USER=user_wordpress  
-e WORDPRESS_DB_PASSWORD=password_wordpress -e WORDPRESS_DB_NAME=db_wordpress --network=wp-network -p 8090:80  
-d wordpress
```

# 1. Docker

- Présentation
- Installer Docker
- Les containers
- Les images
- Le réseau

**>\_ Persistance des données**



A chaque recreation d'un container les données sont perdues. Comment sauvegarder mes données ?

### Utilisation d'un volume afin d'assurer la persistance des données

- Les volumes fonctionnent aussi bien sur Linux que Windows
- Plusieurs syntaxes sont possibles : -v -volume ou --mount (en dehors de Swarm depuis Docker 17.06)
- Les volumes doivent être supprimés via une action manuelle

Créer un volume

```
docker volume create my-vol --opt device=:<chemin export nfs> <nom du volume NFS Docker>
```

Lister les volumes

```
docker volume ls
```

```
local                my-vol
```

Supprimer un volume

```
docker volume rm my-vol
```

[cloud-infrastructure/docker/understanding-and-managing-docker-container-volumes/](https://cloud-infrastructure/docker/understanding-and-managing-docker-container-volumes/)



Comment supprimer un container et son volume proprement ?

Arrêt du container

```
docker container stop DOCKER_ID_CONTAINER
```

Suppression du container

```
docker container rm DOCKER_ID_CONTAINER
```

Suppression du volume du container

```
docker volume rm DOCKER_NAME_VOLUME
```



### Exemple

#### Création du volume

```
docker volume create my-vol
```

#### Ajout du volume

```
docker run -d --name devtest --mount source=my-vol,target=/app nginx:latest
```

ou

```
docker run -d --name devtest -v my-vol:/app nginx:latest
```



Windows : les disques du Docker Host doivent être accessibles aux containers :

- Docker > Settings > Resources > File Sharing

#### Resources File sharing

These drives (and their subfolders) can be bind mounted into Docker containers. You can check the [documentation](#) for more details.

Select the local drives you want to be available to your containers.

☒ C

☒ D



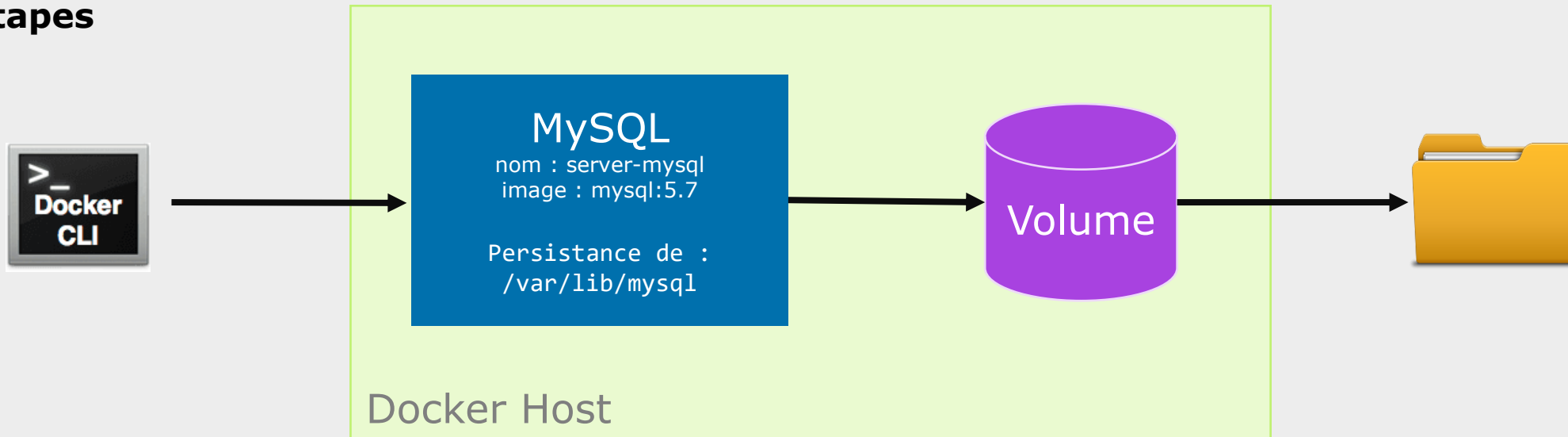
# Docker

## Persistence des données – TP



**Objectif :** Déployer un container MySQL puis assurer la pérennité des données à travers un volume monté.

### Etapes



## 2. DevOps

>\_ Présentation d'une chaîne d'intégration continue

### Principale contrainte

- Garder l'application existante

### Objectifs

- Développer une nouvelle application en microservices via des API REST
- Réduction du temps d'accès aux données et omnicanalité
- Intégration continue



## Front

- Swagger



## Services

- Web Services
- .Net Core
- EF Core
- XUnit
- Docker + Kubernetes



## Data

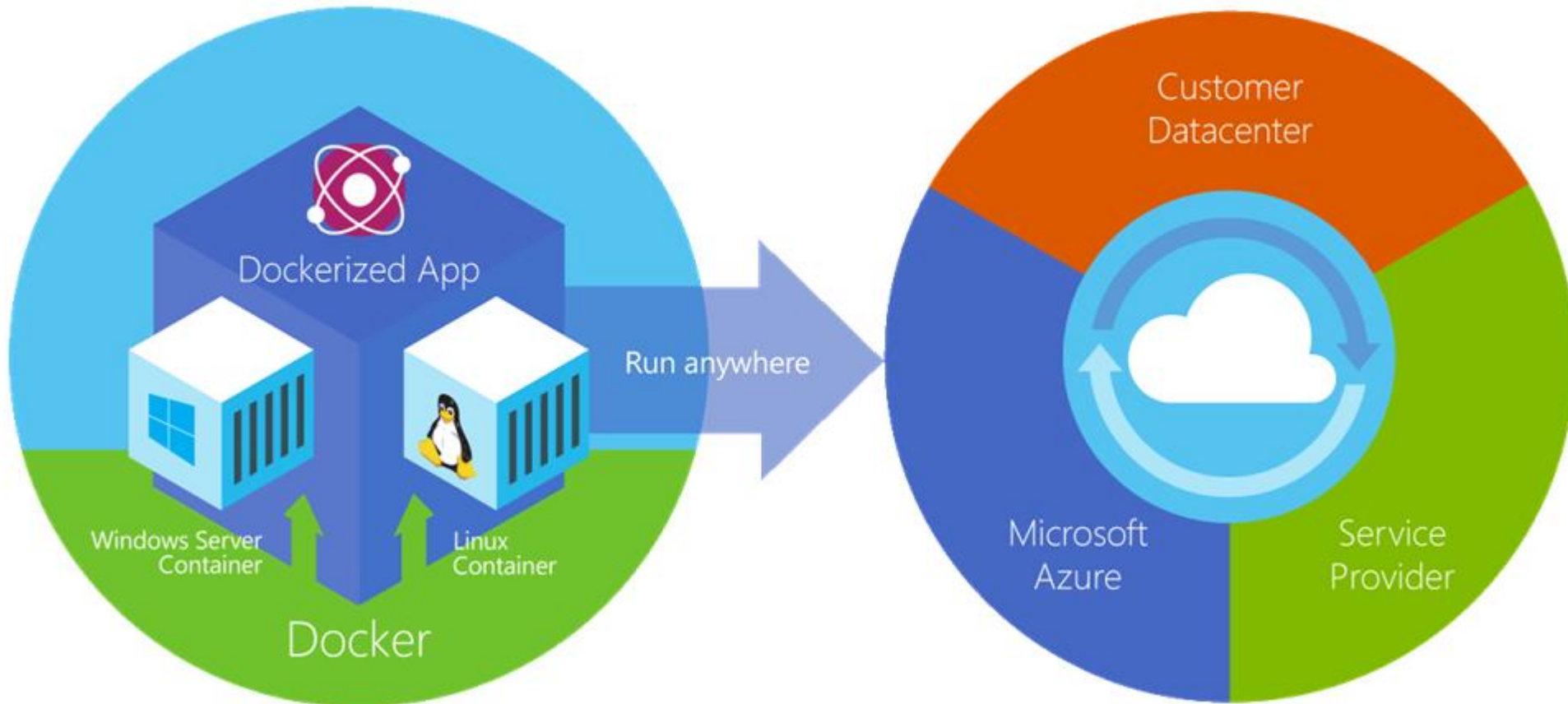
- SQL Server 2016 (azure) & 2014
- MongoDB
- Couchbase

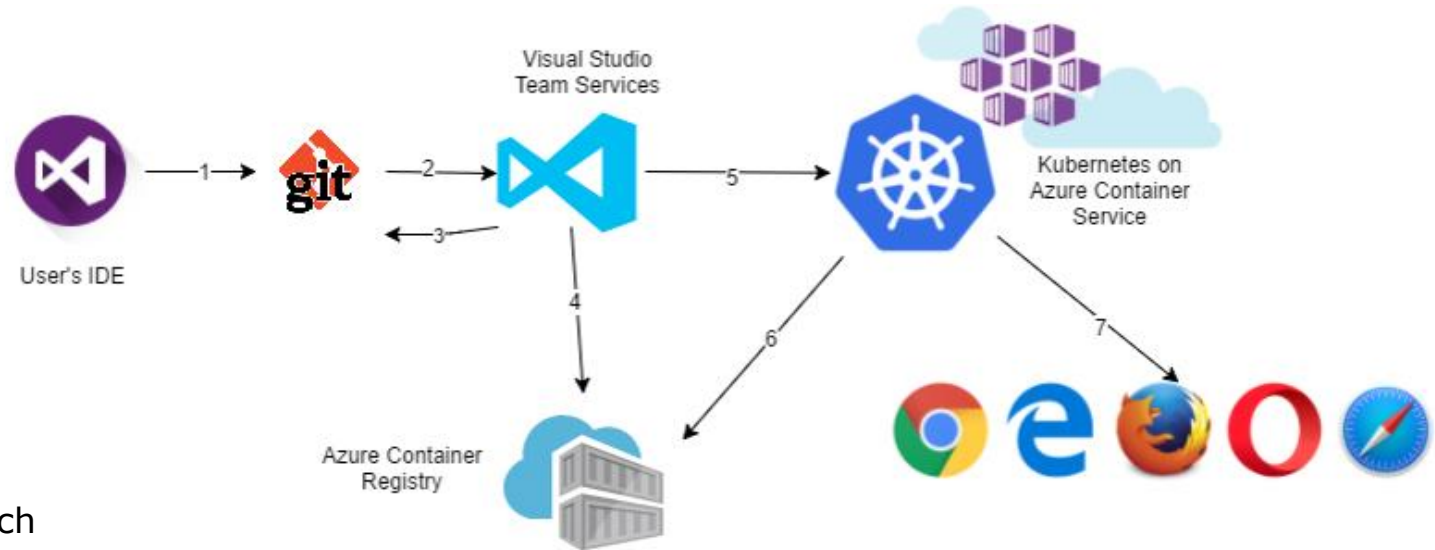
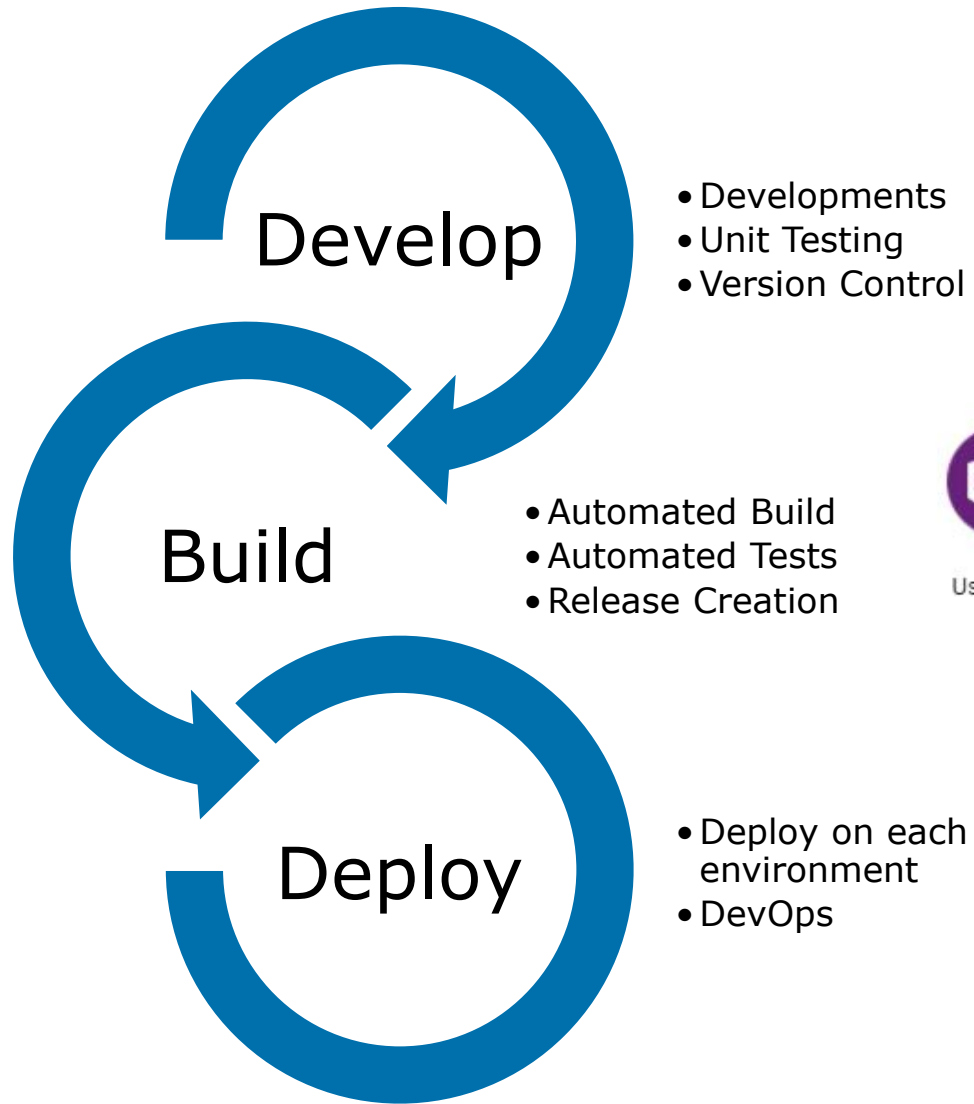


## Plateforme

- Azure
- Azure DevOps (VSTS)







1

### Travail en mode Agile intégré

- Kanban
- Outil de suivi
- Outil de ticketing
- Wiki

2

### Développement

- GitHubFlow depuis VSTS (Git)
- Liaison US - code source
- Gestion des pull-requests pour de la revue de code

3

### Build & déploiement

- Build simple à mettre en place grâce à des templates (Docker, MSBuild, SonarQube, Unit Test)
- Création des environnements cibles dans des définitions de release
- Tests unitaires automatiques par environnement
- Orienté DevOps



**People matter, results count.**

This message contains information that may be privileged or confidential and is the property of the Capgemini Group.  
Copyright © 2020 Capgemini. All rights reserved.

Rightshore® is a trademark belonging to Capgemini.

## About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, [the Collaborative Business Experience™](#), and draws on [Rightshore®](#), its worldwide delivery model.

Learn more about us at

[www.capgemini.com](http://www.capgemini.com)

This message is intended only for the person to whom it is addressed. If you are not the intended recipient, you are not authorized to read, print, retain, copy, disseminate, distribute, or use this message or any part thereof. If you receive this message in error, please notify the sender immediately and delete all copies of this message.



# DevOps

ZZ3 – F2  
2023 - 2024



# PLAN

## 1. Déploiement via docker compose

- Présentation
- Docker compose

## 2. Orchestrateurs

- Présentation
- Swarm by Docker
- Kubernetes by Google
- Mesos

## 3. Tour de table

# 1. Déploiement via docker compose

>\_ Présentation

- Docker compose

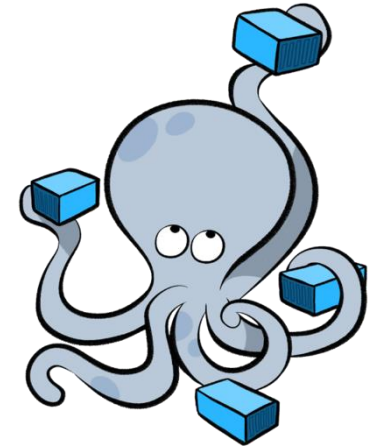
# Déploiement via docker compose

## Présentation



Compose est un outil pour déployer une multitude de containers pour des applications Docker.

- Un fichier YAML (Yet Another Markup Language : extension .yaml ou .yml) permet de décrire la configuration pour déployer les services, ou json (-f docker-compose.json)
- Une seule ligne de commande pour déployer les containers
- Une syntaxe propre à l'outil
- Mise en production



### Installation de l'outil

```
sudo apt install docker compose
```

### Commande Docker classique pour déployer un container

```
docker run
```



### Nouvelle commande pour déployer n containers (Compose V2)

```
docker compose up -d
```

Construit, crée , démarre et attache les containers au service.

# Déploiement via docker compose

## Présentation



docker-compose.yml

```
version: "3.8"

services:
  web:
    build:
      context: .
      dockerfile:
        - dockerfile-alternate
    ports:
      - "5000:5000"
    volumes:
      - logvolume01:/var/log
    image:
      redis
```

**version** : format du fichier, il est dépendant de la version du Docker Engine

**build** : instructions à effectuer lors de la phase de construction de l'image

**ports** : ports à mapper suivant le schéma :  
HOST:CONTAINER

**volumes** : chemins des volumes montés :  
SOURCE:TARGET

**image** : image utilisée par le container (Tag ou Image ID)



Il est impératif de respecter l'indentation. Les tabulations sont interdites.  
Un fichier YAML peut-être validé via : <https://codebeautify.org/yaml-validator>

Ou docker compose -f docker-compose.yml config

# Déploiement via docker compose

## Présentation



Compose file format	Docker Engine release
3.8	19.03.0+
3.7	18.06.0+
3.6	18.02.0+
3.5	17.12.0+
3.4	17.09.0+
3.3	17.06.0+
3.2	17.04.0+
3.1	1.13.1+
3.0	1.13.0+
2.4	17.12.0+
2.3	17.06.0+
2.2	1.13.0+
2.1	1.12.0+
2.0	1.10.0+
1.0	1.9.1.+

# Déploiement via docker compose

## Commandes



Démarrer tous les containers à l'arrêt

```
docker compose start
```

Stop tous les containers en cours d'exécution

```
docker compose stop
```

Validation et affichage de la configuration

```
docker compose config
```

Liste tous les containers en cours d'exécution

```
docker compose ps
```

Stop et supprime tous les containers

```
docker compose down
```

# Déploiement via docker compose

## Docker compose - Installation Linux



Téléchargement de la dernière version de docker compose

```
sudo apt-get update
```

```
sudo apt-get install docker-compose-plugin
```

Vérification de la bonne installation

```
docker compose version
```





# Déploiement via docker compose

## Docker compose - Installation Windows



Docker Desktop intègre nativement l'outil docker compose.

Afficher la version de docker compose

```
docker compose -version
```



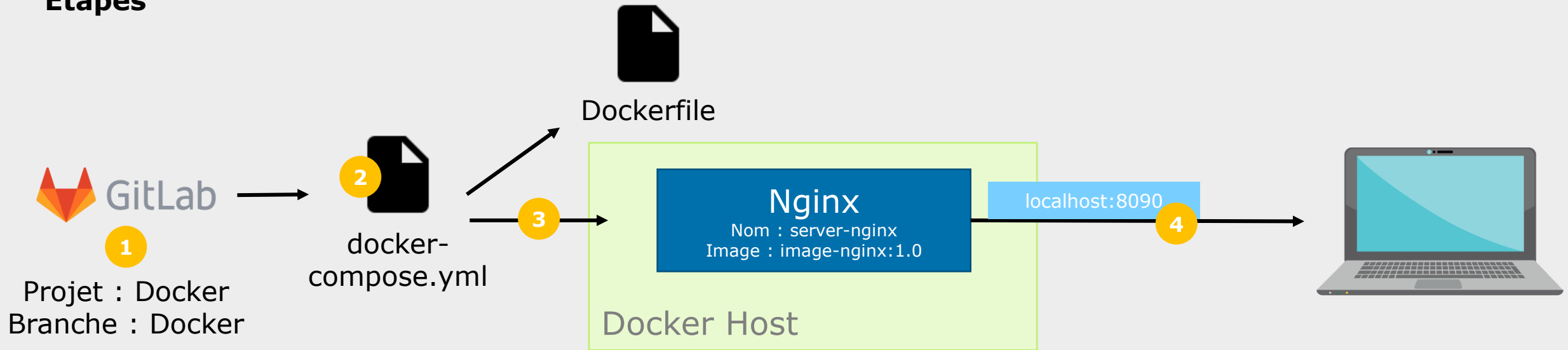
# Déploiement via docker compose

## Docker compose - TP



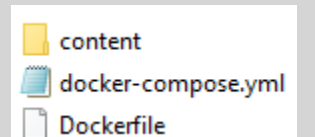
**Objectif :** Déployer le site web du projet Gitlab « Docker » branche « Docker »

### Etapes



- Cloner le projet Gitlab « Docker » branche « Docker »
- Création du docker-compose
- Construction de l'image « image-nginx:1.0 » à partir du Dockerfile

#### Arborescence du Tp

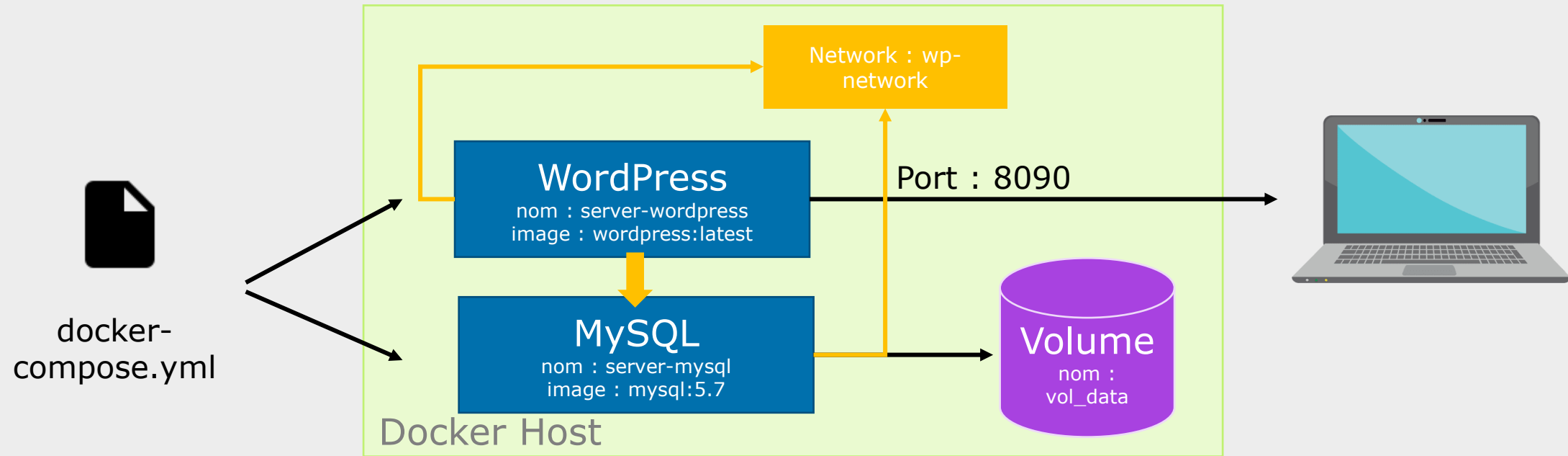


# Déploiement via docker compose

## Docker compose - TP



**Objectif :** Déployer une solution WordPress via l'outil docker-compose 3.8



MySQL :

- Base de données « wordpress »

Utilisation de  
« / » pour le  
chemin déclaré  
dans le volume

Arborescence du Tp

database  
docker-compose.yml

## 2. Orchestrateurs

### >\_ Présentation

- Docker Swarm
- Kubernetes
- Mesos

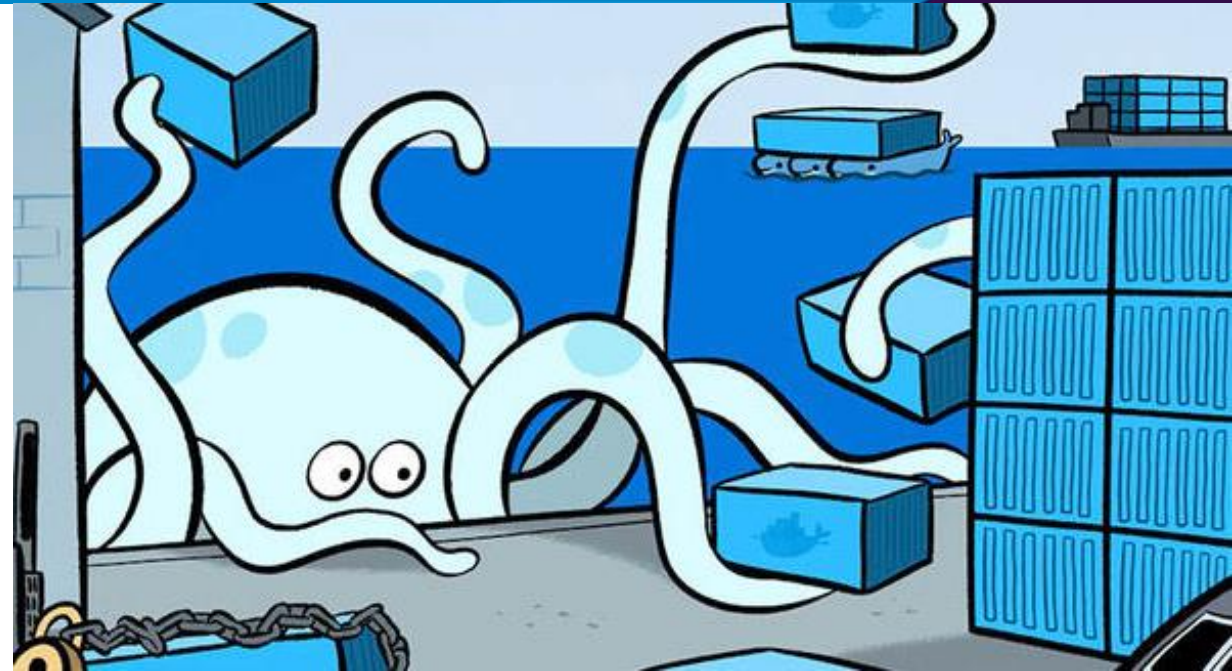
Orchestrer c'est :

- Piloter
- Administrer

**=> Ajout une intelligence au dessus des containers**

**Principales fonctionnalités attendues :**

- Cluster
- Scaling vertical des containers en fonction des charges (RAM et CPU)
- Rollback : restaurer la version précédente du container
- Service de restauration des containers tombés
- Load balancing



## 2. Orchestrateurs

- Présentation

- >\_ Docker Swarm**

- Kubernetes

- Mesos

# Orchestrateurs

## Docker Swarm - Présentation

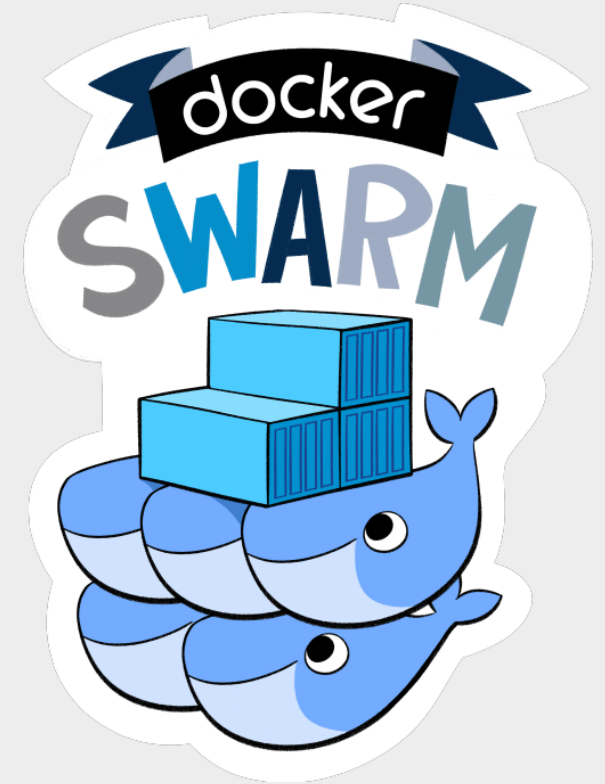


- Créé par la société Docker
- Intégré au Docker Engine depuis la version 1.12
- Utilisation de Docker CLI pour créer un swarm, déployer des services d'applications et manager le cluster.

=> Syntaxe supplémentaire

- Déploiement via le fichier docker-compose.yml

=> Syntaxe supplémentaire



# Orchestrateurs

## Docker Swarm - Fonctionnalités

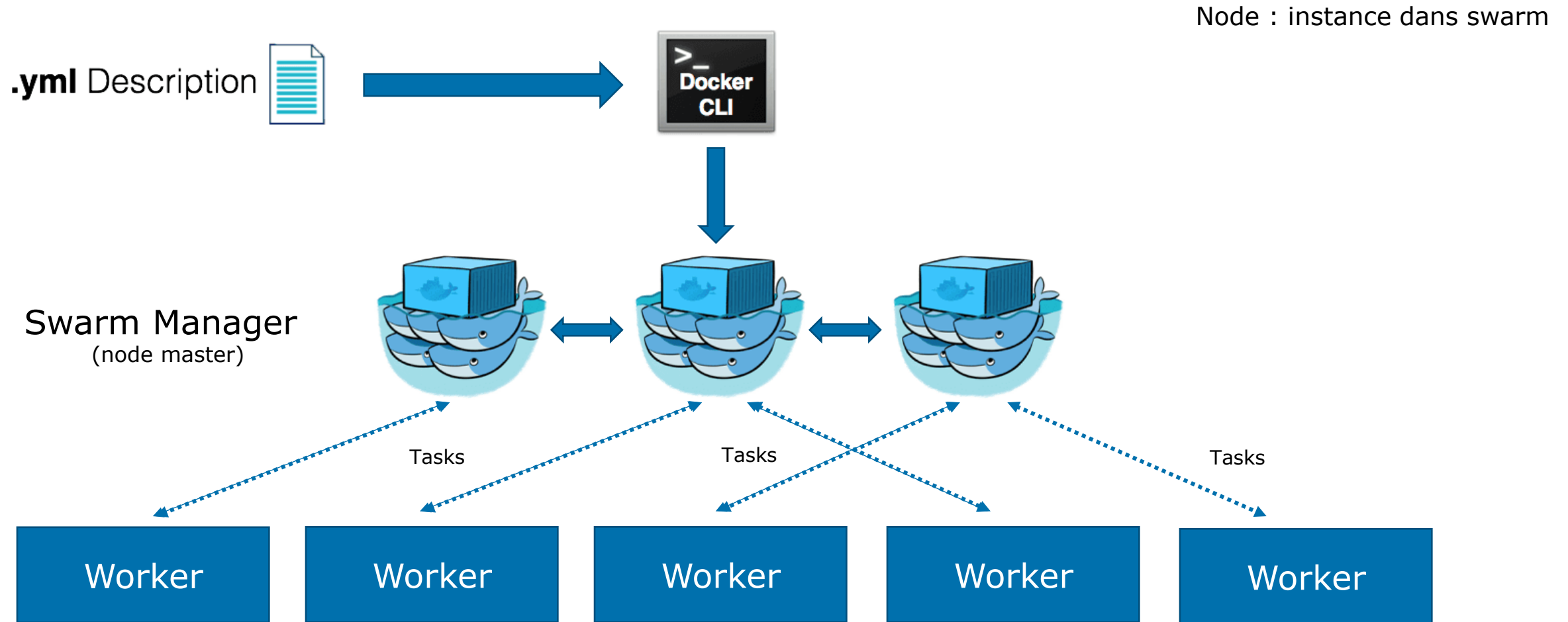


- Déploie des services via un docker compose
- Scaling via des replicas
- Load balacing
- Sécurité
- Rollback



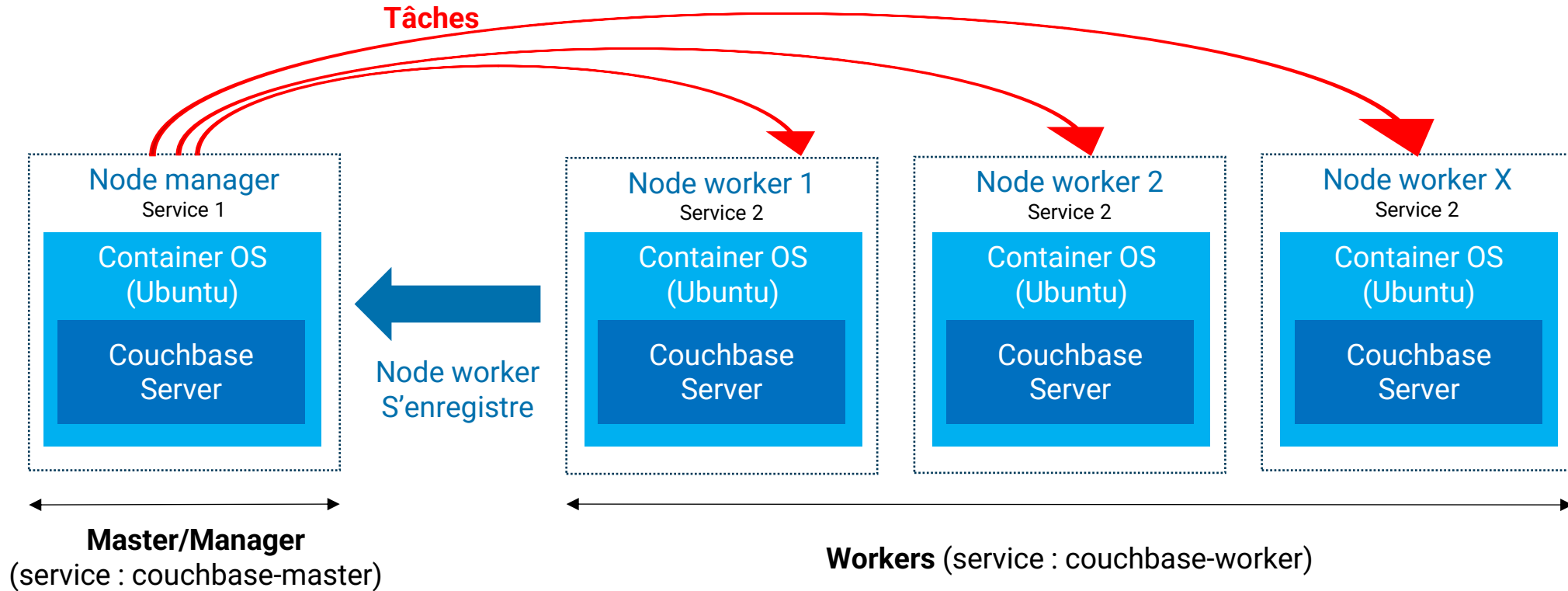
# Orchestrators

## Docker Swarm – Architecture



# Orchestrateurs

## Docker Swarm – Exemple d'architecture



CLUSTER



# Orchestrateurs

## Docker Swarm – Commandes



### Démarrer Docker Swarm

```
docker swarm init
```



Consulter l'état actuel du swarm avec la commande `docker info`

### Quitter Swarm

```
docker swarm leave -f
```

### Nodes

Listez les nodes

```
docker nodes ls
```

### Services

Listez les services

```
docker service ls
```

Déployez des services

```
docker stack deploy --compose-file docker-compose.yml <STACK-NAME>
```

Scale d'un service

```
docker service scale <SERVICE-ID>=<NUMBER-OF-TASKS>
```

Supprimez un service

```
docker service rm <SERVICE-ID>
```

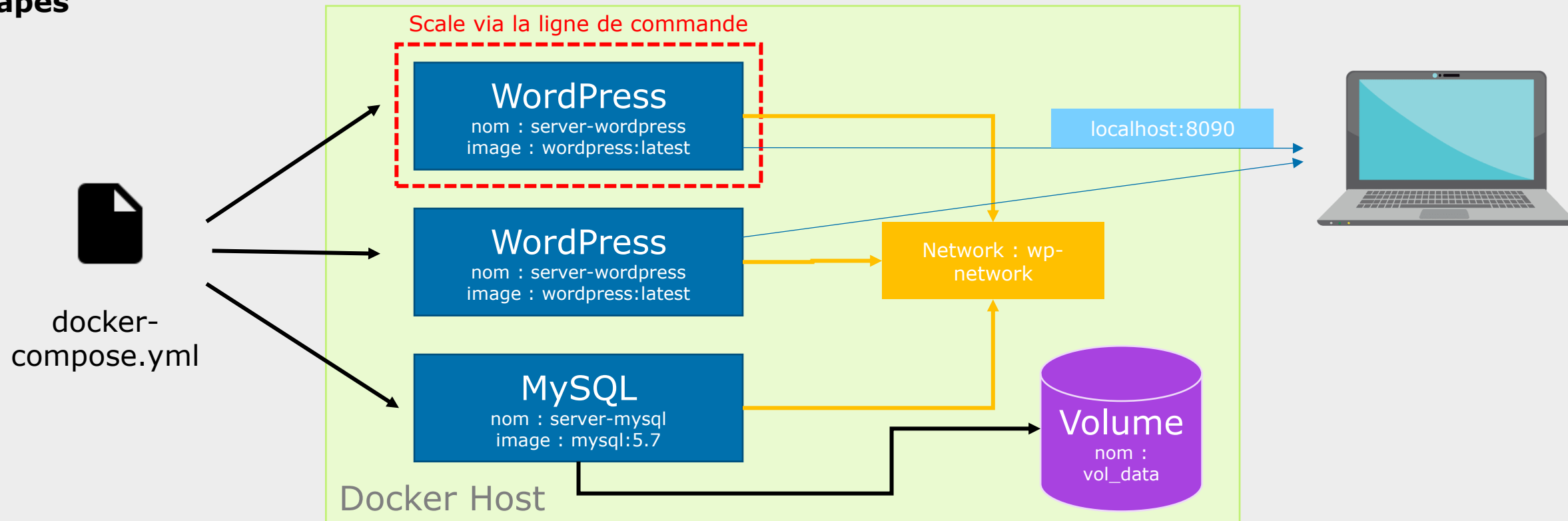
Affichez les logs d'un service

```
docker service logs <SERVICE-ID>
```



**Objectif :** Reprendre le fichier docker-compose.yml du Tp précédent, l'adapter à la syntaxe de Swarm. Déployer les services puis scaler le service « server-wordpress ».

### Etapes



## 2. Orchestrateurs

- Présentation
- Docker Swarm

**>\_ Kubernetes**

- Mesos



# kubernetes

by Google™

- Open source
- Développé par Google
- 7 juin 2014 la première release
- Alias Kube
- Projet à commencé sous l'impulsion de Google
- Contributeurs : Google, CodeOS, RedHat, Microsoft, HP, IBM...

URL roadmap : <https://github.com/kubernetes/kubernetes/milestones/>

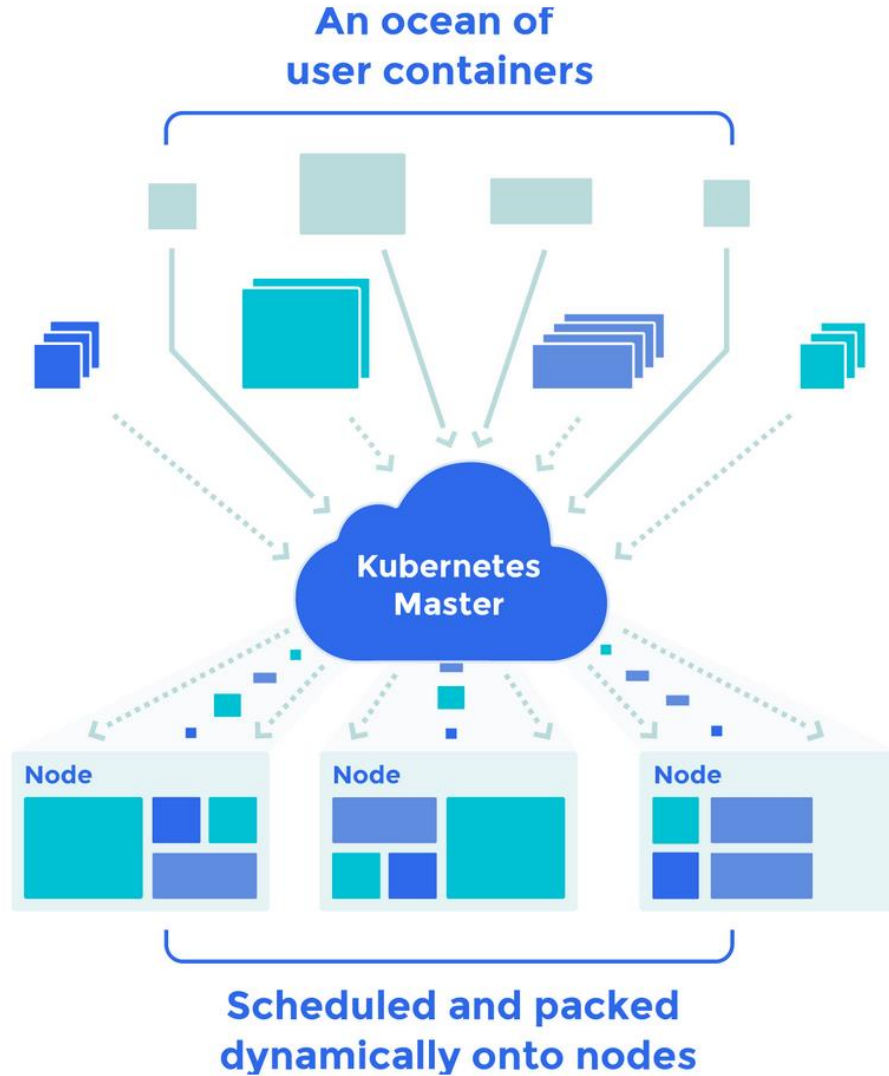


kubernetes  
by Google™

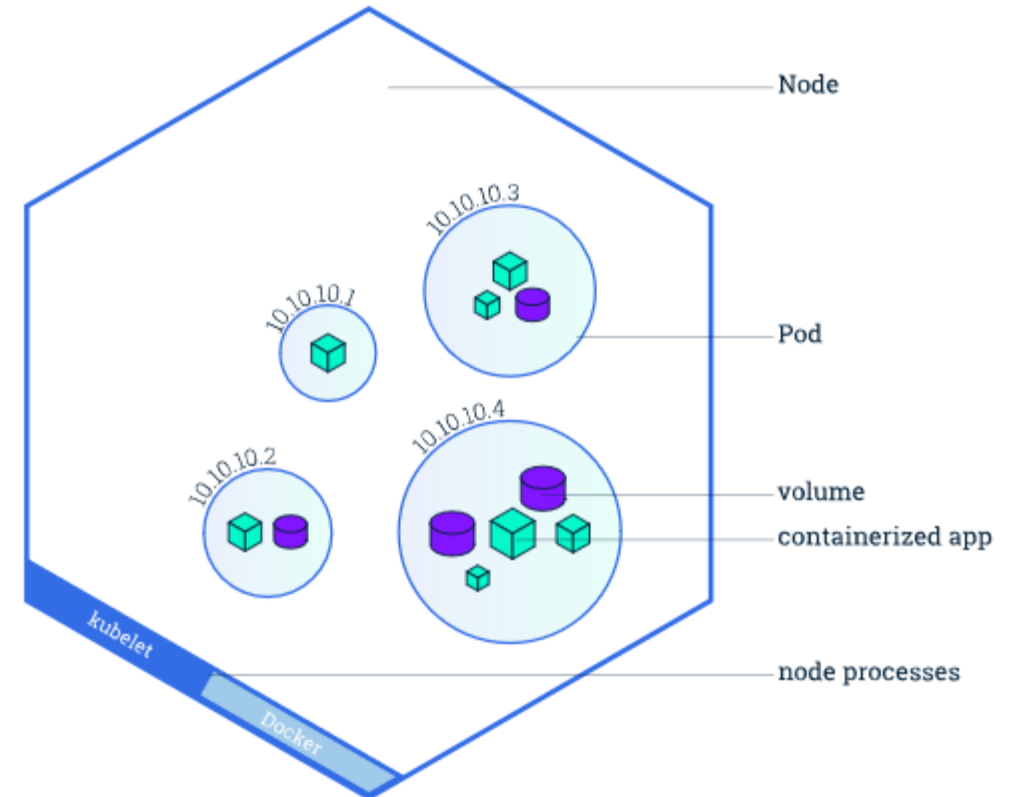
- Auto Scaling horizontale
- Load balancing
- Rollback
- Orchestration du stockage

# Orchestrateurs

## Kubernetes - Architecture



## Structure d'un Node





Kubernetes est compatible avec les Registries suivant :

- Azure container registry (ECR)
- Google Container registry
- AWS (Amazon Web Services)

Une configuration des nodes est nécessaire pour s'authentifier auprès des Registries.





kubectl : interface en ligne de commande, dernière version 1.17

<https://kubernetes.io/docs/user-guide/kubectl-overview/>

Démarrer un container

```
kubectl run
```

Lister les nodes

```
kubectl get nodes
```

Lister les pods

```
kubectl get pod
```

Créer un pod à partir d'un fichier .yaml

```
kubectl create -f FILENAME.yaml
```

Afficher la version de Kubernetes

```
kubectl version
```

# Orchestrateurs

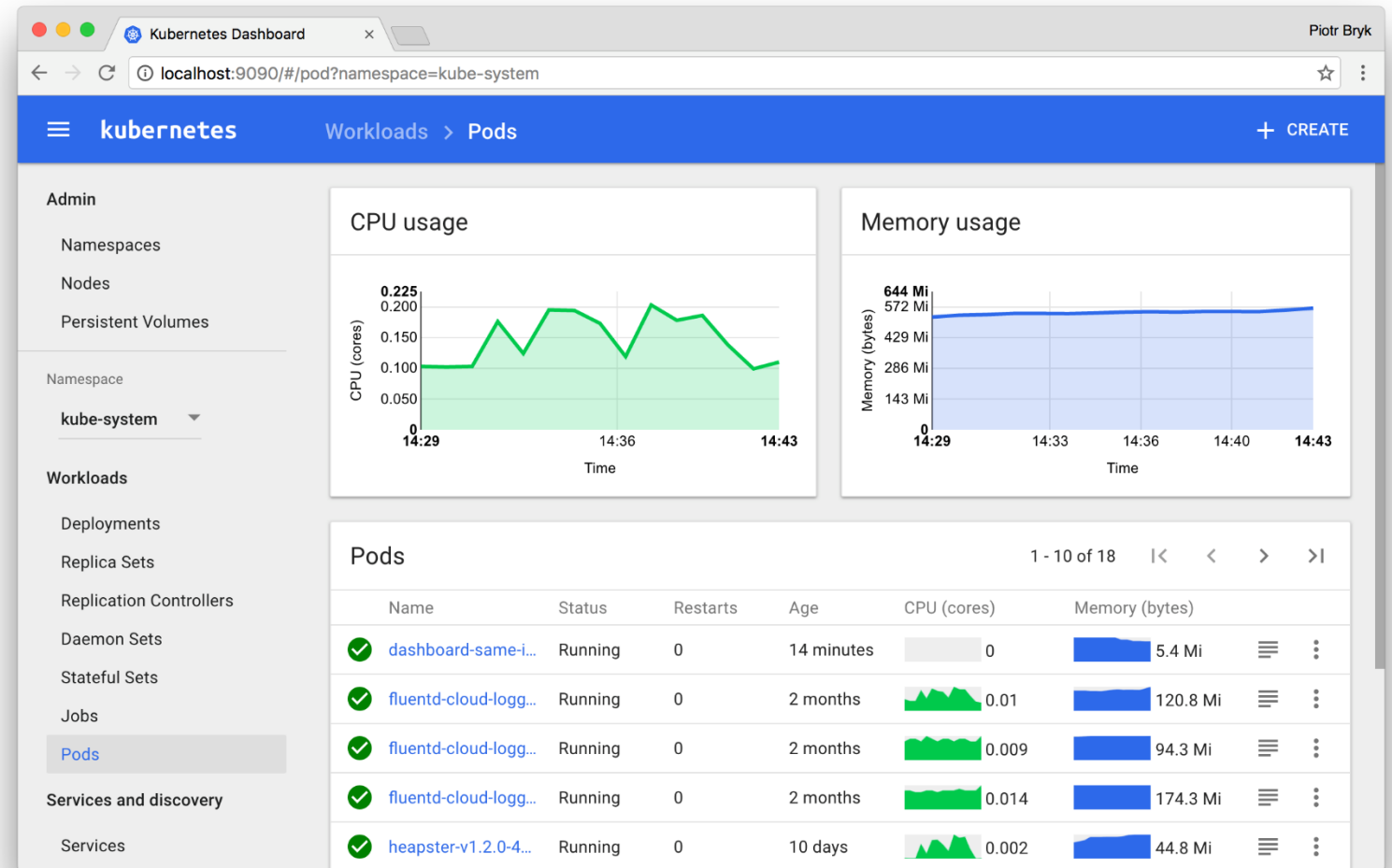
## Kubernetes – Web UI (Dashboard)



Dashboard web pour manager et monitorer les containers dans un cluster Kubernetes

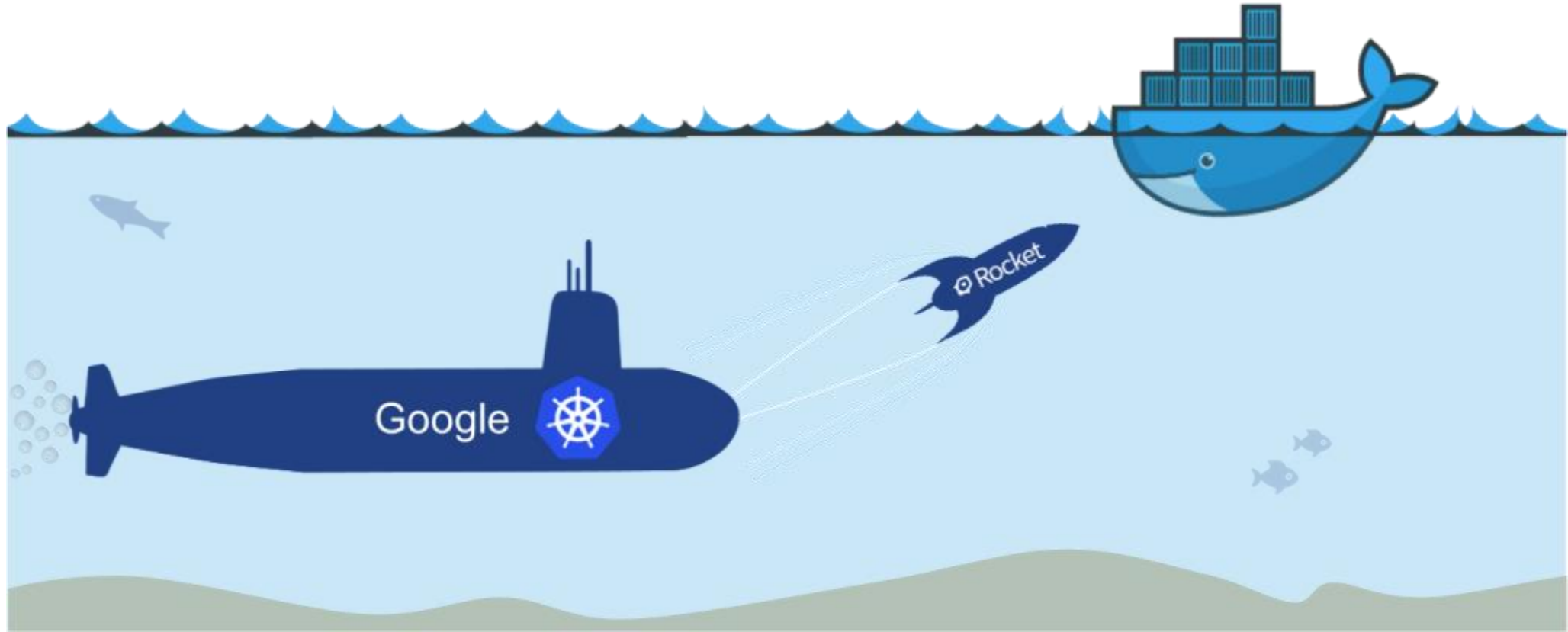
### Déploiement

```
kubectl apply -f  
https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended.yaml
```



# Orchestrateurs

## Kubernetes vs Swarm



## Intégration de Kubernetes officielle dans Docker

## 2. Orchestrateurs

- Présentation
- Docker Swarm
- Kubernetes

>\_ **Apache Mesos**





- Open source
- Cluster Hadoop
- Développé par l'université de Berkeley
- Société commerciale « Mesosphere »



- Relation maître esclaves
- Réplication : tolérance élevée aux pannes
- Container Docker
- Déploiement sur linux
- Déploiement sur environnement Windows non supporté actuellement
- Déploiement milliers de containers
- Allouer des ressources aux clusters



**People matter, results count.**

This message contains information that may be privileged or confidential and is the property of the Capgemini Group.

Copyright © 2020 Capgemini. All rights reserved.

Rightshore® is a trademark belonging to Capgemini.

## About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, [the Collaborative Business Experience™](#), and draws on [Rightshore®](#), its worldwide delivery model.

Learn more about us at

[www.capgemini.com](http://www.capgemini.com)

This message is intended only for the person to whom it is addressed. If you are not the intended recipient, you are not authorized to read, print, retain, copy, disseminate, distribute, or use this message or any part thereof. If you receive this message in error, please notify the sender immediately and delete all copies of this message.