Nuages. Conteneurs. Micro-services

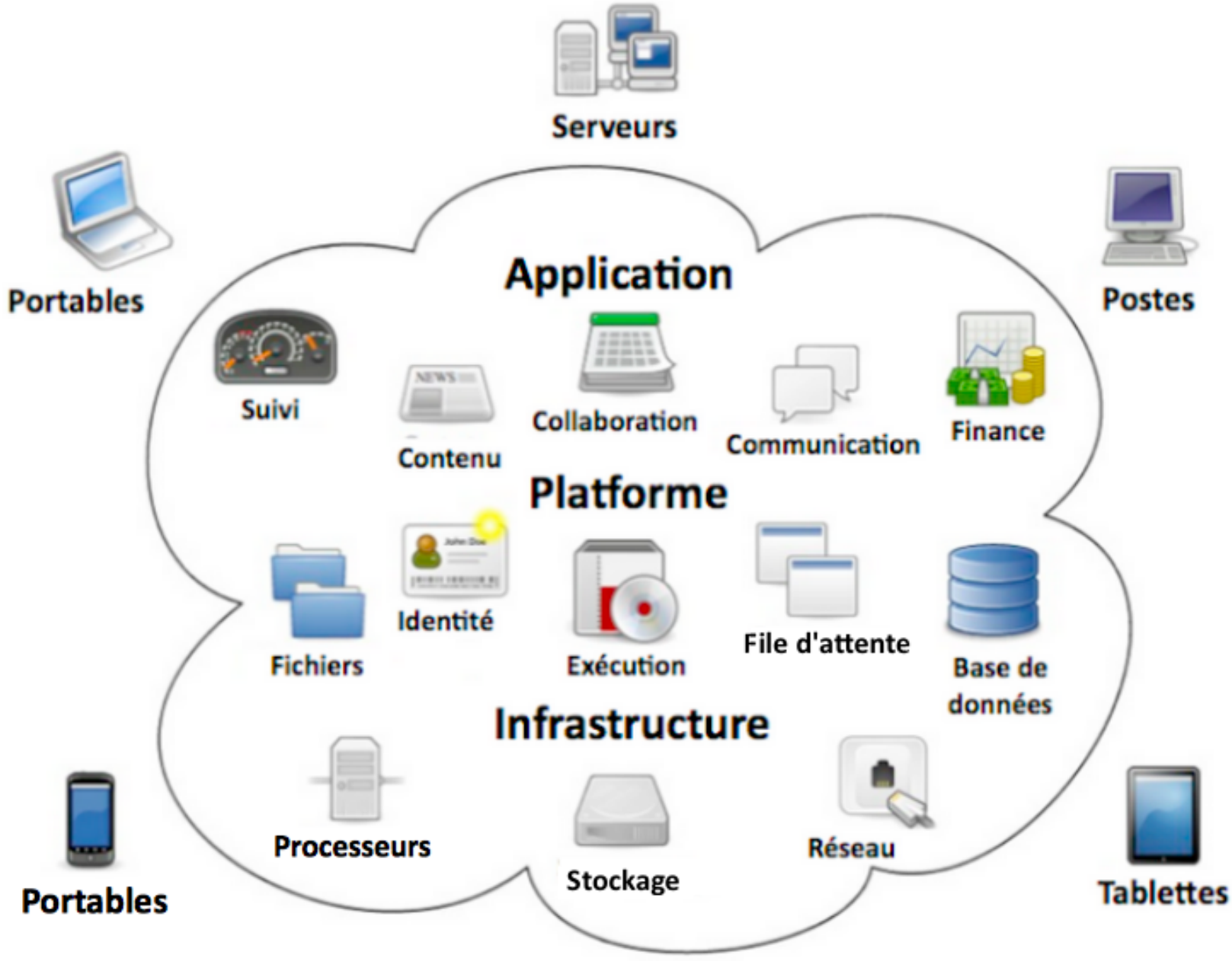
Sergey Kirgizov

ESIREM

Clouds

Le **cloud computing**, *en français* **l'informatique en nuage** *ou* **nuagique** *ou encore* **l'infonuagique** *(au Québec), consiste à exploiter la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement Internet.*

`https://fr.wikipedia.org/wiki/Cloud_computing`


Serveurs
Portables
Postes
Application
Suivi
Contenu
Collaboration
Communication
Finance
Platforme
Identité
Fichiers
Exécution
File d'attente
Base de données
Infrastructure
Processeurs
Stockage
Réseau
Portables
Tablettes

le Nuage

Histoire

1950 Mainframes.

2000 Hébergeurs Web capables d'héberger des applications

2006 Amazon Web Services et Elastic Compute Cloud (EC2)

2008 Google App Engine, NASA's OpenNebula

2010 Microsoft Azure, OpenStack de Rackspace Hosting et NASA

2011 IBM SmartCloud

2012 Oracle Cloud

2012 Google Compute Engine

2013 Docker

2014 Kubernetes

...

La promotion du cloud computing

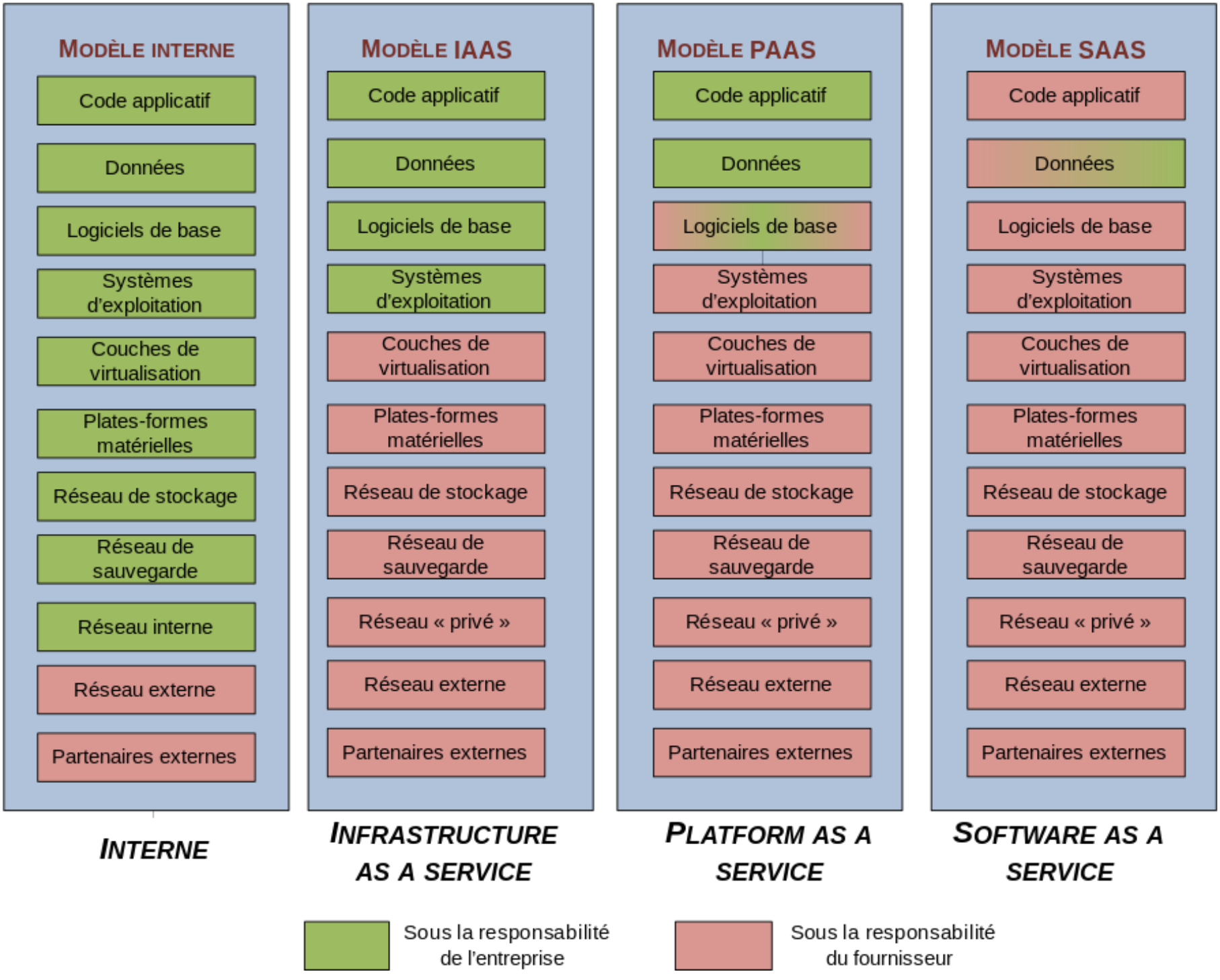
- la généralisation des accès à Internet
- augmentation considérable de puissance des serveurs (la fréquence de fonctionnement des serveurs a été multipliée par un facteur 10, entre 1998 et 2008, les processeurs comportent entre quatre et dix cœurs).
- et de la baisse des coûts de stockage (pour le prix d'un disque dur de 1,2 Go en 2000, on a, en 2013, un disque de 1 000 Go).

Fournisseur de services en nuage

- Amazon Web Service (AWS)
- Microsoft Azure
- Google Cloud Platform
- ...

Construire votre propre cloud

- OpenStack
- OpenShift
- ...


Modèle interne
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau interne
Réseau externe
Partenaires externes
Modèle IAAS
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau « privé »
Réseau externe
Partenaires externes
Modèle PAAS
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau « privé »
Réseau externe
Partenaires externes
Modèle SAAS
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau « privé »
Réseau externe
Partenaires externes
Interne
Infrastructure as a service
Platform as a service
Software as a service
Sous la responsabilité de l'entreprise
Sous la responsabilité du fournisseur

Le monde de logiciel est complexe


Comme un festival d'aventures ?


Des fois c'est un peu chaud

Une solution ?

Mettre les systèmes dans les conteneurs.

Diviser pour régner.

La containérisation. C'est quoi ça ?

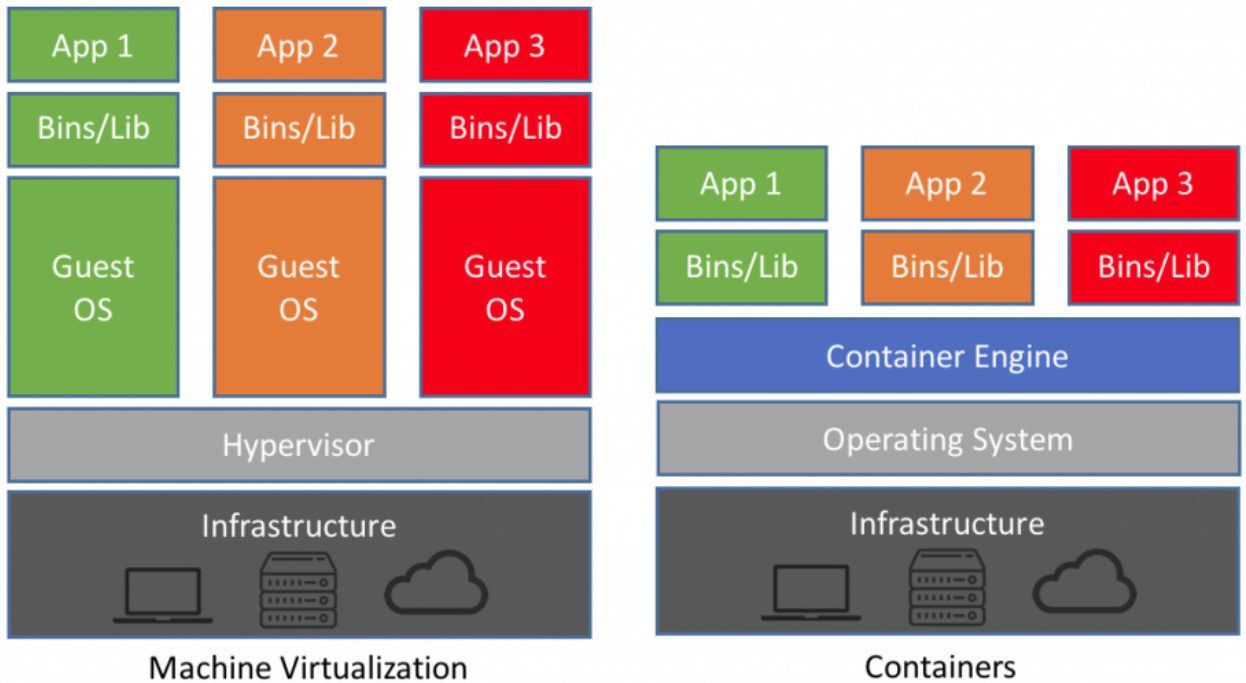
Containérisation

“Comme les .apk mais pour les serveurs.”

- C’est un façon de livrer les applications
  - Sur les serveurs
  - Pour les utilisateurs finaux : Flatpack, Endless OS
- C’est un façon de gérer les app sur les serveurs (mises à jour, rollbacks, backup des version, mises en prodounction)
- Les conteneurs sont des machines virtuelles “bien faites”, simples et faciles, sans perte de performance
- Bootstrap des environments de developpement, de test, des environnement des builds

Conteneurs vs Machine virtuelles

Conteneurs sont des machines virtuelles “bien faites” : simples et faciles, sans perte de performance.



`https://blog.netapp.com/blogs/containers-vs-vms/`

Pourquoi conteneurs ?

Pourquoi conteneurs ?

Déploiement rapide, efficace et reproductible.

Pourquoi conteneurs ?

Déploiement rapide, efficace et reproductible.

- des environnement de test

Pourquoi conteneurs ?

Déploiement rapide, efficace et reproductible.

- des environnement de test
- de recette

Pourquoi conteneurs ?

Déploiement rapide, efficace et reproductible.

- des environnement de test
- de recette
- de produnction

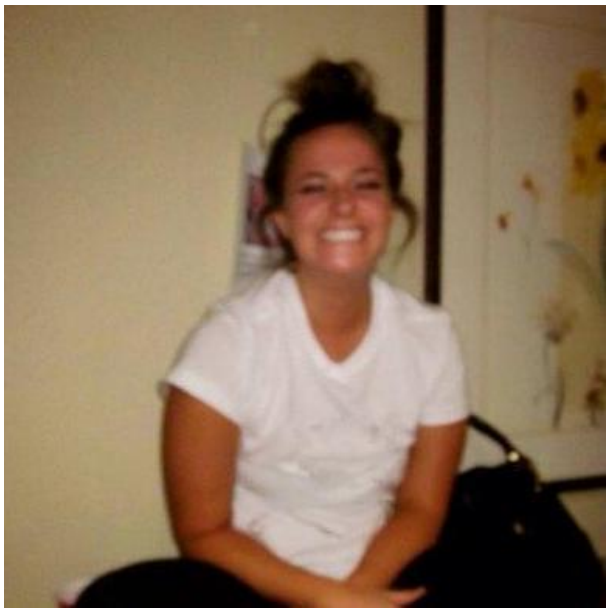
Pourquoi conteneurs ?

Déploiement rapide, efficace et reproductible.

- des environnement de test
- de recette
- de produnction
- des environnements de développement (même local)

“I use Docker to run all the desktop apps on my computers.”
— Jessie Frazelle

`https://github.com/jessfraz/dockerfiles`



jessfraz repo ···

.github	Create FUNDING.yml
ab	Replaced deprecated MAINTAINER with LABEL (#242)
afterthedeadline	switch to openjdk
android-tools	fix android tools
ansible	Update E-Mail to match PGP key
apt-file	change to sid-slim
atom	fix nomad and atom
audacity	change to sid-slim
awscli	Replaced deprecated MAINTAINER with LABEL (#242)
azure-cli	cleanup latest versions
bcc-tools	fix dockerfile builds
beeswithmachineguns	Update Dockerfile (#396)
browsh	update dockerfile versions
buttslock	http:// -> https://
cathode	update
certbot	update versions
cf-reset-cache	Replaced deprecated MAINTAINER with LABEL (#242)
cfssl	update versions
checkup	checkup update
cheese	change to buster slim
chrome	change to sid-slim
chromium	fix chromium
clair	cleanup
cli53	fix py2
clisp	fix dockerfile builds
cloudapp	fix some builds
consul	update consul and rstudio
coredns	update versions
couchpotato	fix foss-heartbeat and couchpotato

Il existe de nombreuses technologies de conteneurisation.

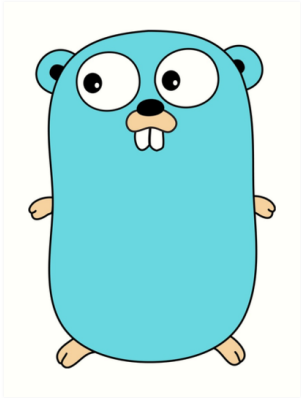
Il existe de nombreuses technologies de conteneurisation.

Docker est l'une des plus populaires.

Mechanism	Operating system	License	Available since or between	Features								
				File system isolation	Copy on Write	Disk quotas	I/O rate limiting	Memory limits	CPU quotas	Network isolation	Nested virtualization	Partiti checkpoi and live mi
chroot	Most UNIX-like operating systems	Varies by operating system	1982	Partial[a]	No	No	No	No	No	No	Yes	No
Docker	Linux,[6] FreeBSD,[7] Windows x64 (Pro, Enterprise and Education)[8] macOS [9]	Apache License 2.0	2013	Yes	Yes	Not directly	Yes (since 1.10)	Yes	Yes	Yes	Yes	Only in Exp Mode with C
Linux-VServer (security context)	Linux, Windows Server 2016	GNU GPLv2	2001	Yes	Yes	Yes	Yes[b]	Yes	Yes	Partial[c]	?	No
lmctfy	Linux	Apache License 2.0	2013	Yes	Yes	Yes	Yes[b]	Yes	Yes	Partial[c]	?	No
LXC	Linux	GNU GPLv2	2008	Yes[11]	Yes	Partial[e]	Partial[f]	Yes	Yes	Yes	Yes	No
Singularity	Linux	BSD Licence	2015[12]	Yes[13]	Yes	Yes	No	No	No	No	No	No
OpenVZ	Linux	GNU GPLv2	2005	Yes	Yes (ZFS)	Yes	Yes[g]	Yes	Yes	Yes[h]	Partial[i]	Yes
Virtuozzo	Linux, Windows	Trialware	2000[18]	Yes	Yes	Yes	Yes[k]	Yes	Yes	Yes[h]	Partial[l]	Yes
Solaris Containers (Zones)	illumos (OpenSolaris), Solaris	CDDL, Proprietary	2004	Yes	Yes (ZFS)	Yes	Partial[m]	Yes	Yes	Yes[n][21][22]	Partial[o]	Partial[
FreeBSD jail	FreeBSD, DragonFly BSD	BSD License	2000[24]	Yes	Yes (ZFS)	Yes[s]	Yes	Yes[25]	Yes	Yes[26]	Yes	Partial[2
sysjail	OpenBSD, NetBSD	BSD License	2006–2009	Yes	No	No	No	No	No	Yes	No	No
WPARs	AIX	Commercial proprietary software	2007	Yes	No	Yes	Yes	Yes	Yes	Yes[t]	No	Yes[3
iCore Virtual Accounts	Windows XP	Freeware	2008	Yes	No	Yes	No	No	No	No	?	No
Sandboxie	Windows	Trialware	2004	Yes	Yes	Partial	No	No	No	Partial	No	No
systemd-nspawn	Linux	GNU LGPLv2.1+	2010	Yes	Yes	Yes[32][33]	Yes[32][33]	Yes[32][33]	Yes[32][33]	Yes	?	?
Turbo	Windows	Freemium	2012	Yes	No	No	No	No	No	Yes	No	No
RKT	Linux	Apache License 2.0	?	?	?	?	?	?	?	?	?	?

https://en.wikipedia.org/wiki/Operating-system-level_virtualization#Implementations

Docker



`https://docker.com`

La grande partie du Docker est écrite en Go.
Code source est ouvert : `https://github.com/docker`

Solomon Hykes, Kamel Founadi, et Sébastien Pahl

En 2008, trois anciens élèves de l'Epitech, Solomon Hykes, Kamel Founadi et Sébastien Pahl créent une SARL à Montrouge, nommée "dotCloud"

twitter.com/solomostre

Kamel Founadi

twitter.com/sebp

`https://fr.wikipedia.org/wiki/Solomon_Hykes`

Brève histoire du Docker

1. En 2010, La société dotCloud postule au “Y Combinator”.

Brève histoire du Docker

1. En 2010, La société dotCloud postule au “Y Combinator”.
2. En janvier 2011, dotCloud s’implante dans la Silicon Valley.

Brève histoire du Docker

1. En 2010, La société dotCloud postule au “Y Combinator”.
2. En janvier 2011, dotCloud s’implante dans la Silicon Valley.
3. Début 2013 — open source.

Brève histoire du Docker

1. En 2010, La société dotCloud postule au “Y Combinator”.
2. En janvier 2011, dotCloud s’implante dans la Silicon Valley.
3. Début 2013 — open source.
4. Tout le monde l’utilise : les startups du “Y Combinator” et puis

Brève histoire du Docker

1. En 2010, La société dotCloud postule au “Y Combinator”.
2. En janvier 2011, dotCloud s’implante dans la Silicon Valley.
3. Début 2013 — open source.
4. Tout le monde l’utilise : les startups du “Y Combinator” et puis ... Microsoft, Google, Amazon, IBM....
5. Solomon Hykes annonce qu’il quitte l’entreprise en 2018. `https://blog.docker.com/2018/03/au-revoir/`

Brève histoire du Docker

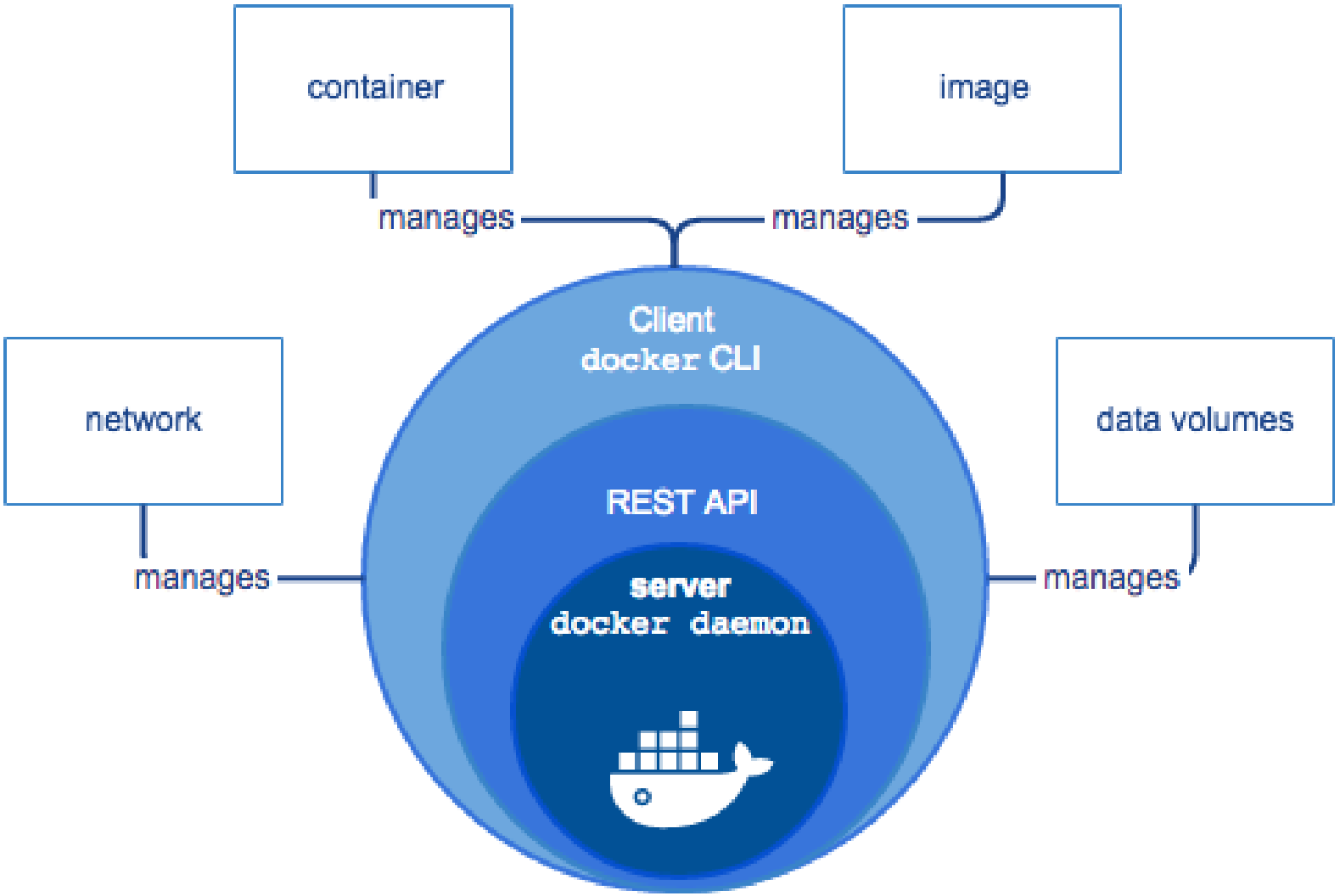
1. En 2010, La société dotCloud postule au “Y Combinator”.
2. En janvier 2011, dotCloud s’implante dans la Silicon Valley.
3. Début 2013 — open source.
4. Tout le monde l’utilise : les startups du “Y Combinator” et puis ... Microsoft, Google, Amazon, IBM....
5. Solomon Hykes annonce qu’il quitte l’entreprise en 2018. `https://blog.docker.com/2018/03/au-revoir/`
6. Sébastien Pahl à quitté Docker en 2011, puis CloudFare, Mesosphere, “Director Of Engineering” chez Red Hat, Entrepreneur

Brève histoire du Docker

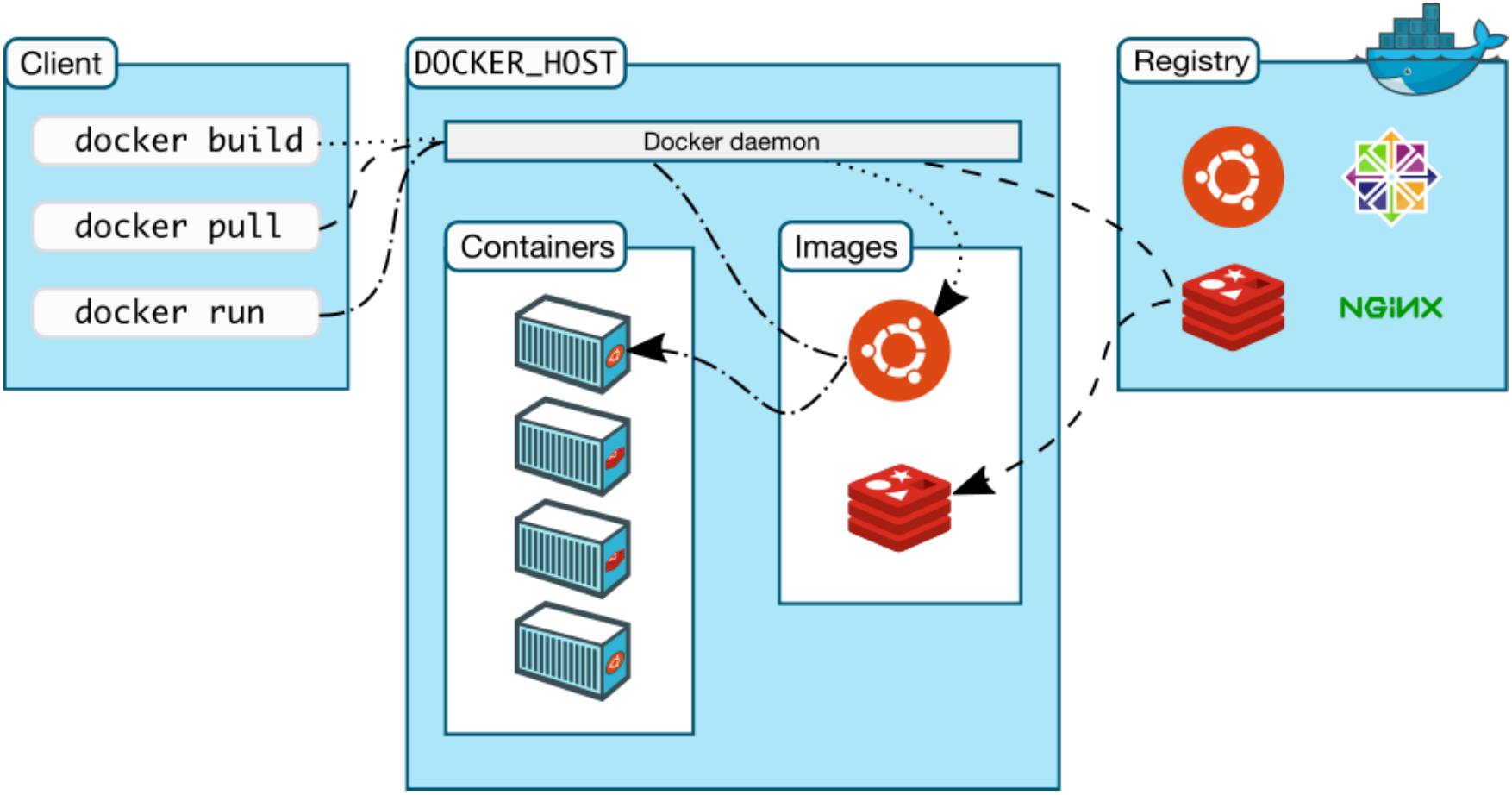
1. En 2010, La société dotCloud postule au “Y Combinator”.
2. En janvier 2011, dotCloud s’implante dans la Silicon Valley.
3. Début 2013 — open source.
4. Tout le monde l’utilise : les startups du “Y Combinator” et puis ... Microsoft, Google, Amazon, IBM....
5. Solomon Hykes annonce qu’il quitte l’entreprise en 2018. `https://blog.docker.com/2018/03/au-revoir/`
6. Sébastien Pahl à quitté Docker en 2011, puis CloudFare, Mesosphere, “Director Of Engineering” chez Red Hat, Entrepreneur
7. Kamel Founadi — ???


En plus, t'avais raison, faire du taysson, ça joue sur la confiance en soi, j'ai l'impression.
Salut tout le monde !
Salut !
Salut Salomon.
Je fais l'échauffement et j'arrive.
On va faire des exercices pour travailler les blocages.
Ah tiens, j'ai télédnangé votre jeu.
Ah ?

Architecture du docker

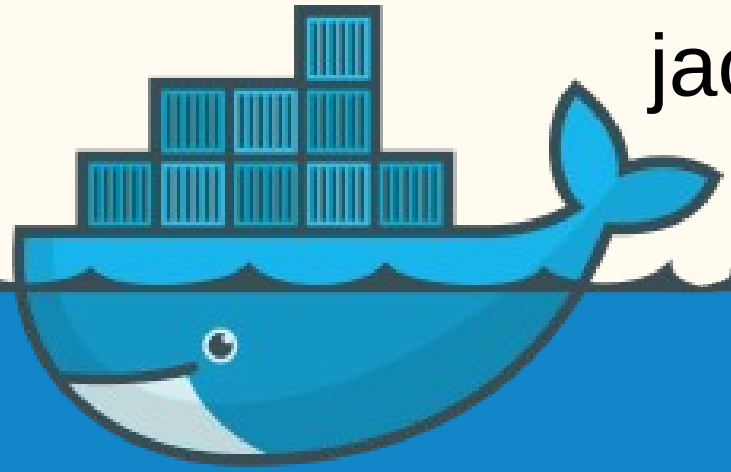


Architecture du docker



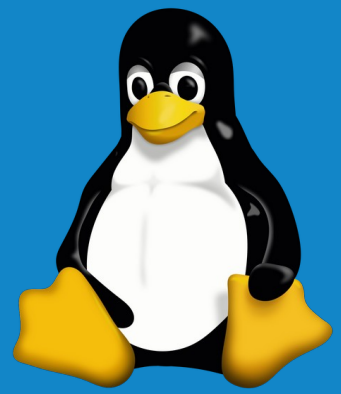
https://docs.docker.com/engine/docker-overview

Océan sous-jacent de Docker



GNU/Linux kernel

namespaces
cgroups Netlink
capabilites
Netfilter



Installation

1. `https://docs.docker.com/install/linux/docker-ce/debian/`
2. Ajoutez l'utilisateur au groupe `docker`, par la commande `sudo gpasswd -a <YOUR-USER-NAME> docker`

Docker images

"A Docker image consists of read-only layers each of which represents a `Dockerfile` *instruction. The layers are stacked and each one is a delta of the changes from the previous layer."*

`https://docs.docker.com/develop/develop-images/dockerfile_best-practices/`

Docker images

```
# Tapez dans votre terminal
docker image
```

- Les images, c'est comme les paquets ".deb" ou les fichier ".apk"
- Les images sont immuables en général
- Souvent une nouvelle image se basée sur une autre image

Créer les images nous mêmes

`docker build`

Où trouver, où stocker ?

- Votre propre Docker registry
- `https://hub.docker.com`
- Catalogues de fournisseurs de logiciels, par exemple `https://access.redhat.com/containers/`

Docker conteneurs

```
# Tapez dans votre terminal
docker container
```

- Conteneur c'est une image lancée sur la machine de Docker

  `docker run <nom d'image>`

- On peut utilise une image pour créer plusieurs conteneurs :
  - Serveurs de dev, recette, prod
  - Mises à l'échelle, "scaling"

- On peut voir l'image comme un fichier exécutable, lors le conteneur un processus créé à partir de cette image

```
aronia-melanocarpa :: ~ » docker container

Usage:  docker container COMMAND

Manage containers

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.
```

Les données des conteneurs ne se sauvegardent pas après le redémarrage !

Docker Volumes

Comment travailler avec des données mutables ?

Docker Volumes

Comment travailler avec des données mutables ?

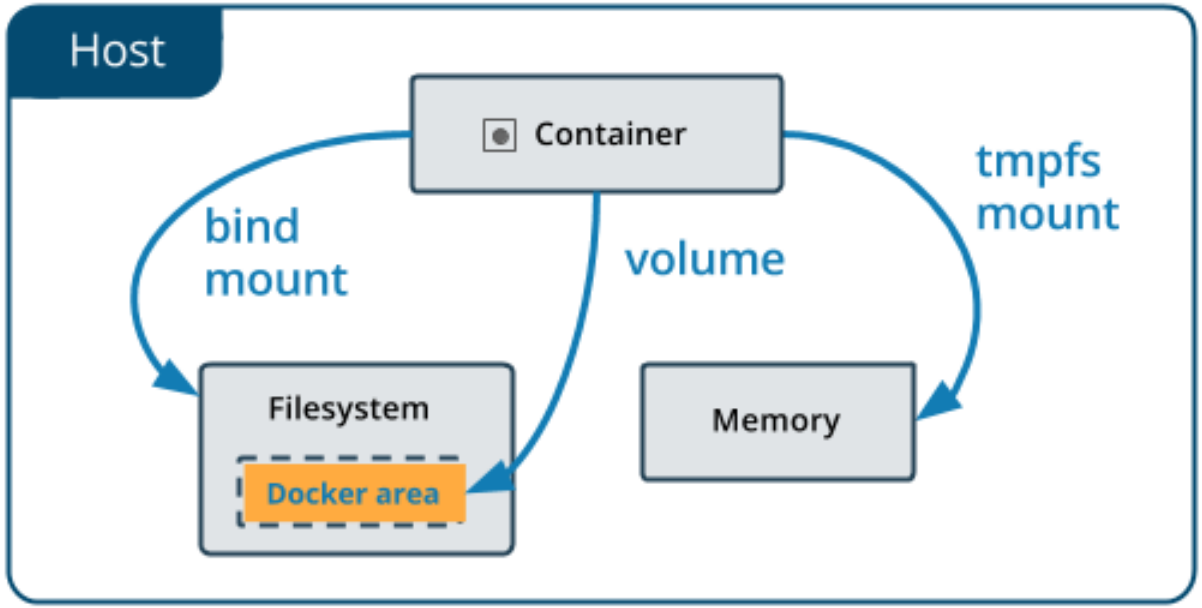
Logs,
DB,
etc...

Docker Volumes

```
# Tapez dans votre terminal
docker volume
```

Comment travailler avec des données mutables ?

Logs,
DB,
etc...



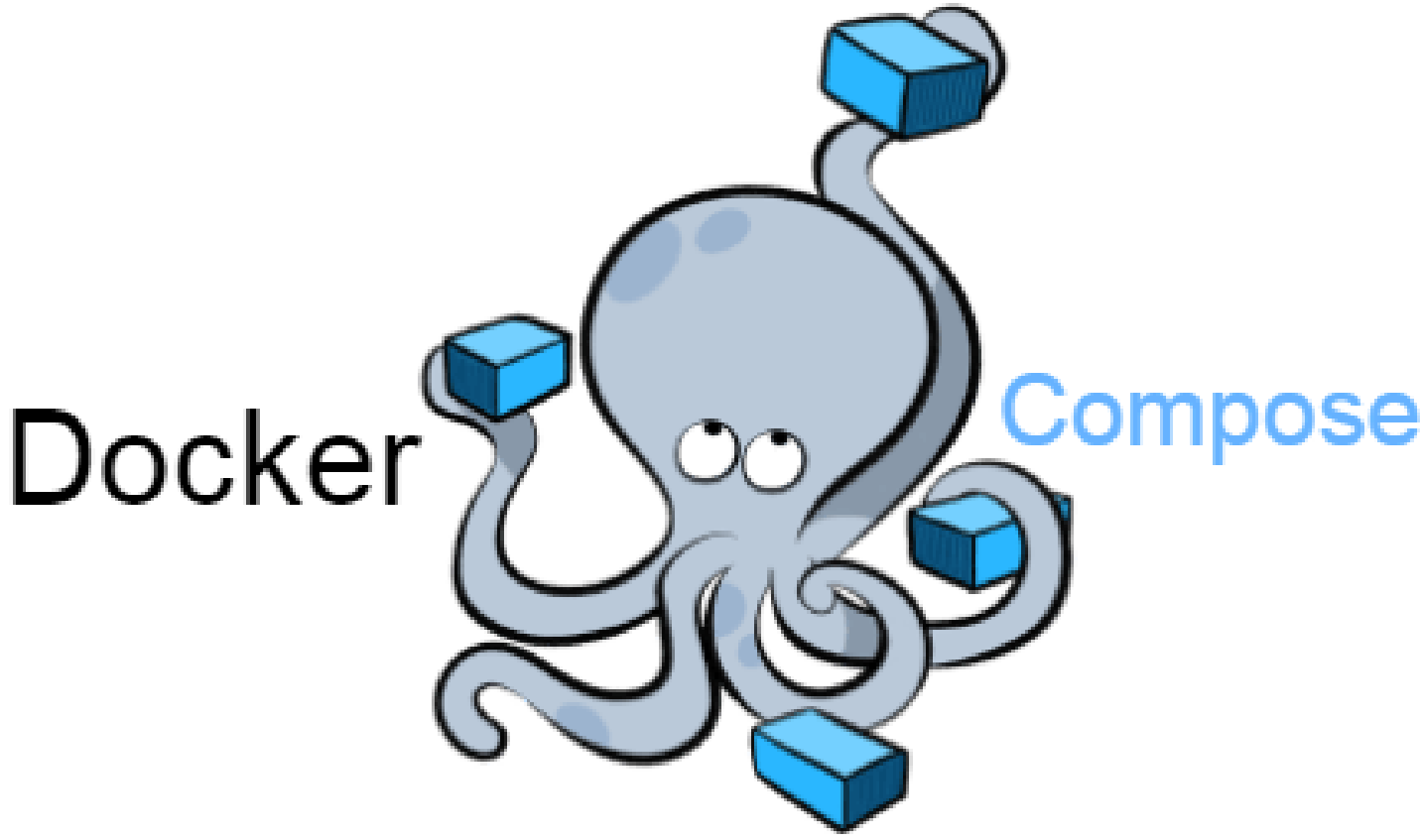
Réseau des conteneurs

```
# Tapez dans votre terminal
docker network
```

Différents types des réseaux virtuelles sont possibles avec docker

- **bridge**
- **host**
- **overlay**
- **macvlan / ipvlan**
- **Rien**
- **plugins des réseau personnalisés**

Un outil permettant d'orchestrer l'ensemble des conteneurs



Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur $\longrightarrow$

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur $\longrightarrow$ juste `docker run`

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur $\longrightarrow$ juste `docker run`

1 machine, un peu de conteneurs $\longrightarrow$

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur $\longrightarrow$ juste `docker run`

1 machine, un peu de conteneurs $\longrightarrow$ Docker compose

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur ⟶ juste `docker run`

1 machine, un peu de conteneurs ⟶ Docker compose

Beaucoup machines, beaucoup de conteneurs ⟶

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur ⟶ juste `docker run`

1 machine, un peu de conteneurs ⟶ Docker compose

Beaucoup machines, beaucoup de conteneurs ⟶ Kubernetes, Docker swarm

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur ⟶ juste `docker run`

1 machine, un peu de conteneurs ⟶ Docker compose

Beaucoup machines, beaucoup de conteneurs ⟶ Kubernetes, Docker swarm

Les environnements encore plus complexes, "Enterprise Support" ⟶
⟶

Gestion des conteneurs

On peut tout faire à la main, mais ce n'est pas bien...

Automatisation est mieux

1 machine, 1 containeur $\longrightarrow$ juste `docker run`

1 machine, un peu de conteneurs $\longrightarrow$ Docker compose

Beaucoup machines, beaucoup de conteneurs $\longrightarrow$ Kubernetes, Docker swarm

Les environnements encore plus complexes, "Enterprise Support" $\longrightarrow$
$\longrightarrow$ Docker Enterprise, Rancher, Red Hat OpenShift Container Platform

Docker compose

Compose est un outil permettant de définir et d'exécuter des multiples conteneurs de Docker.

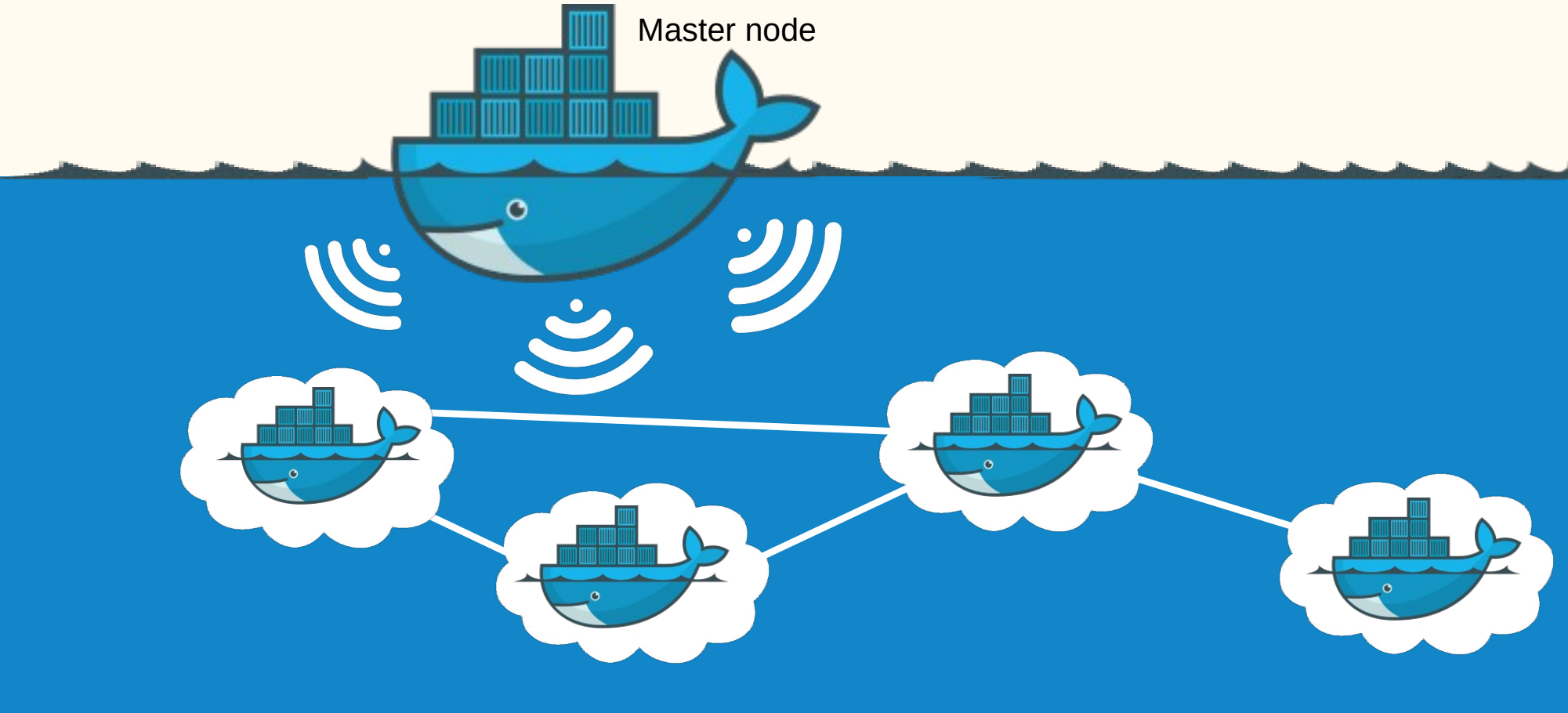
Docker compose

Compose est un outil permettant de définir et d'exécuter des multiples conteneurs de Docker.

Pour Docker Compose, un fichier `docker-compose.yml` doit être créé. Chaque container doit être décrit : image, commande de lancement, volumes, ports...

Docker swarm

```
# Tapez dans votre terminal
docker swarm
```



Kubernetes

`https://kubernetes.io/`

Code source : `https://github.com/kubernetes/kubernetes`

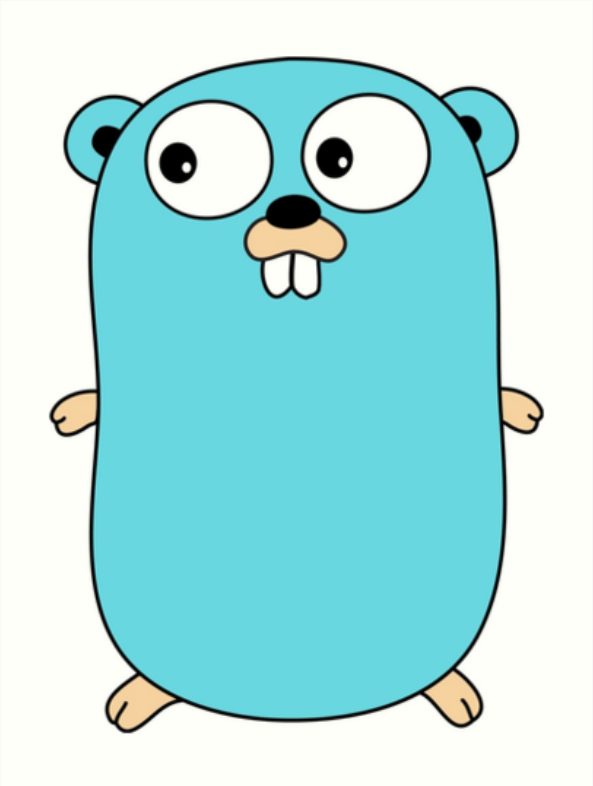
Kubernetes contrôle des Docker-s

Avec Kubernetes...

- Découverte de service et équilibrage de charge
- Orchestration du stockage
- Rollouts et rollbacks automatisés
- Exécution des “batches”
- Mise à l’échelle horizontale
- Gestion de la configuration (dont mots de passe, cléfs, etc)
- Auto-guérison

Language “Go”

La plus grande partie du Docker et Kubernetes est écrite en Go.



`https://golang.org/`

Exemple time

Micro-services

Qu'est ce que c'est ?

Les microservices sont un style d'architecture logicielle à partir duquel un ensemble complexe d'applications est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche. Les processus indépendants communiquent les uns avec les autres en utilisant des API langage-agnostiques.

`https://fr.wikipedia.org/wiki/Microservices`

La philosophie des microservices

La philosophie de l'architecture microservices s'inspire en grande partie de la philosophie UNIX, qui prône "ne faire qu'une seule chose, et la faire bien".

- Les services sont petits, et conçus pour remplir une seule fonction.
- L'organisation du projet doit prendre en compte l'automatisation, le déploiement et les tests.
- Chaque service est élastique (scalable), résilient (tolérant aux pannes), composable, minimal et complet.

Questions ?