

# Introduction à la conteneurisation

*Par Maxime Cottret - Consultant Cloud @ Objectif Libre*



4 Juin 2018

[www.objectif-libre.com](http://www.objectif-libre.com)  
@objectiflibre

# Sommaire

- ✓ **Objectif Libre en 3 slides**
- ✓ Conteneur 101
- ✓ Le cas Docker
- ✓ Open Container Initiative
- ✓ Moby Project
- ✓ Cloud Native Computing Foundation
- ✓ Ecosystème opensource
- ✓ Q/R



# Une société de services 100% Open Source

- ✓ Société de Services en Logiciels Libres **par conviction**
- ✓ En **croissance continue**
  - Création en 2009
  - +30 à 40% de croissance annuelle depuis 2014
  - 21 salariés... et bientôt + !
  - 14% du CA ré-investi en R&D
  - + de 3500 personnes formées
  - 2 agences : Toulouse | Paris



- ✓ Une entreprise **engagée**



Christophe Sauthier, CEO et fondateur  
Lauréat du prix  
'Entrepreneur Open Source de l'année 2016'



# Objectif Libre, c'est...

Une société de  
services 100%  
Open Source

We ♥ Linux

Experts en  
technologies  
d'infrastructures  
innovantes

#Cloud  
# Conteneurs

#DevOps  
#Automatisation



Audit / Conseil  
Formation  
Intégration  
Développement  
Support

*'Packs Starters' OpenStack  
Catalogue complet de formations*

Accompagner nos clients  
dans leurs projets Cloud



# Ils nous font confiance



Nous sommes partenaires / contributeurs

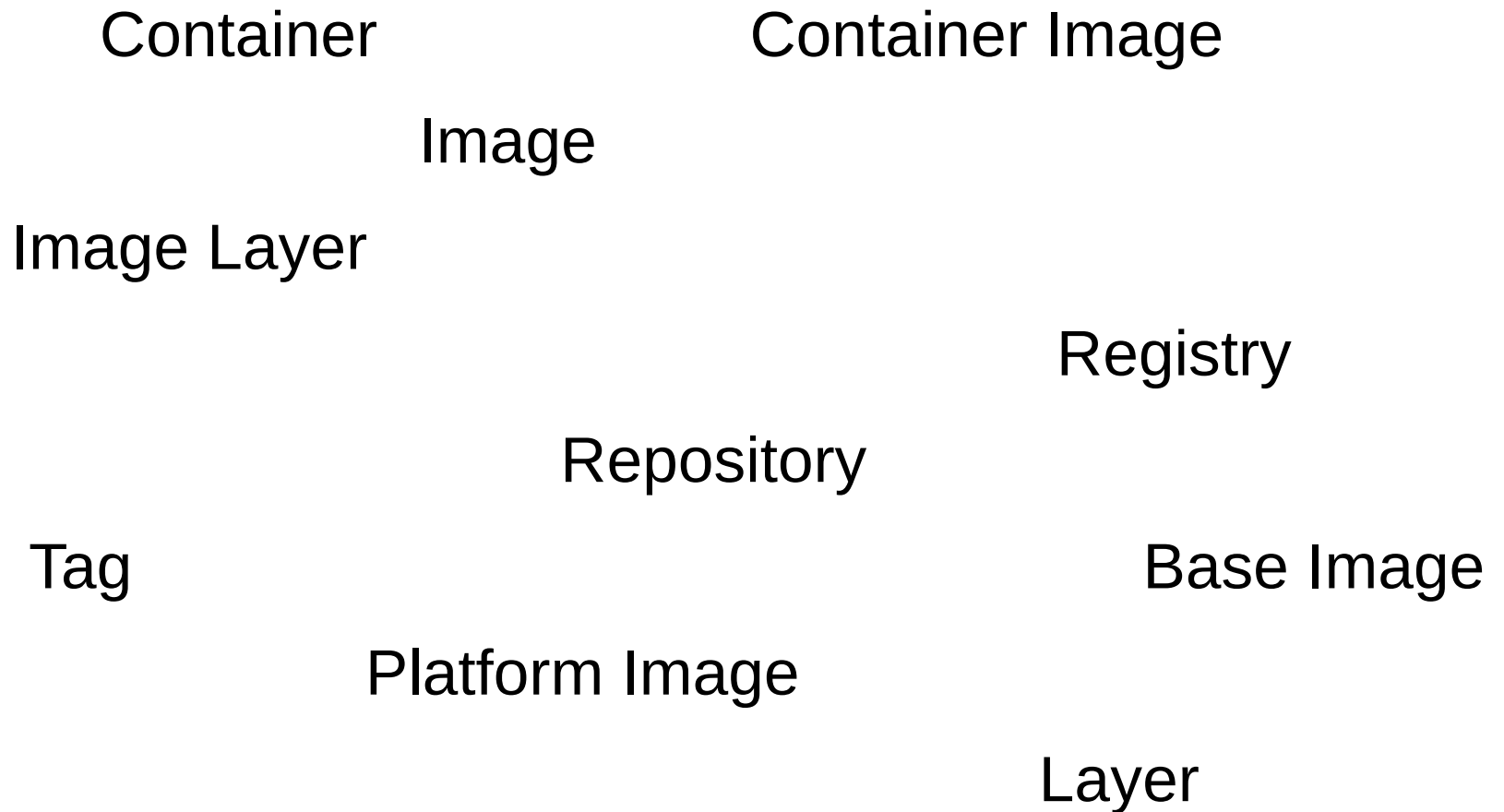


# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ **Conteneur 101**
- ✓ Le cas Docker
- ✓ Open Container Initiative
- ✓ Moby Project
- ✓ Cloud Native Computing Foundation
- ✓ Ecosystème opensource
- ✓ Q/R



# Qu'est qu'un conteneur ?



# Qu'est qu'un conteneur ?

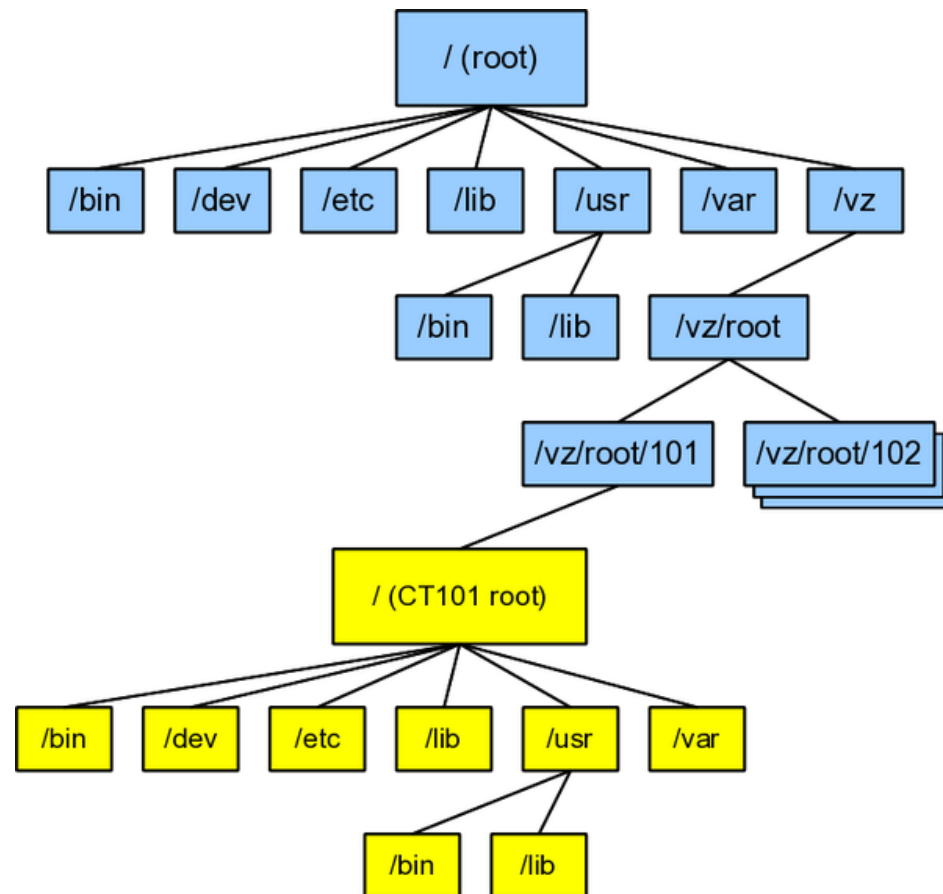
- ✓ Un conteneur est un **concept** : le principe est d'isoler l'exécution d'ensembles de processus sur un même hôte.
- ✓ Le noyau et l'accès au matériel reste partagé (à la différence d'une VM)
- ✓ La création d'un « conteneur » résulte de l'**utilisation conjointe de différentes technologies**
- ✓ Une notion commune est l'utilisation d'un **jeu de bibliothèques séparé** de l'hôte (rootfs)





# Qu'est qu'un conteneur ?

## Rootfs



# Qu'est qu'un conteneur ?

## Historique

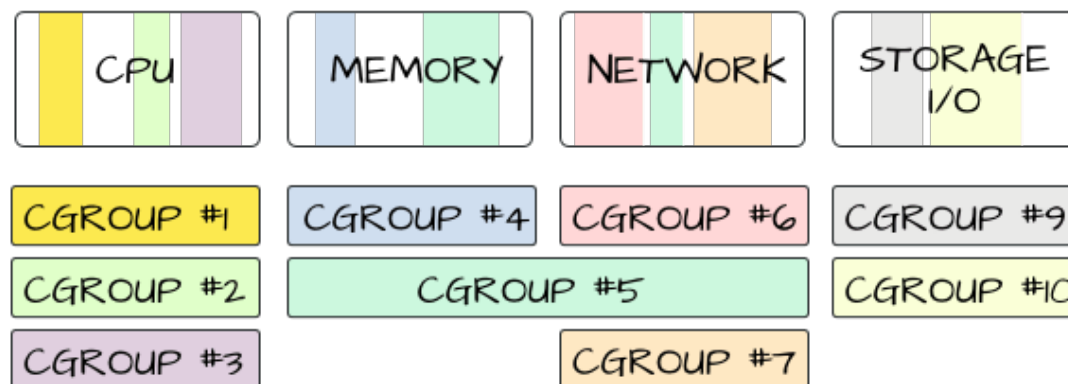
- 1979 : Unix V7 chroot
- 2000 : FreeBSD Jails
- 2001 : LinuxVServer
- 2004 : Solaris Containers
- 2005 : OpenVZ
- 2006 : **cgroups**
- 2008 : **namespaces**
- 2008 : LXC
- 2011 : Warden (Cloudfoundry)
- 2013 : Lmctfy (google)
- 2013 : **Docker**



# Qu'est qu'un conteneur ?

## C(ontrol) groups

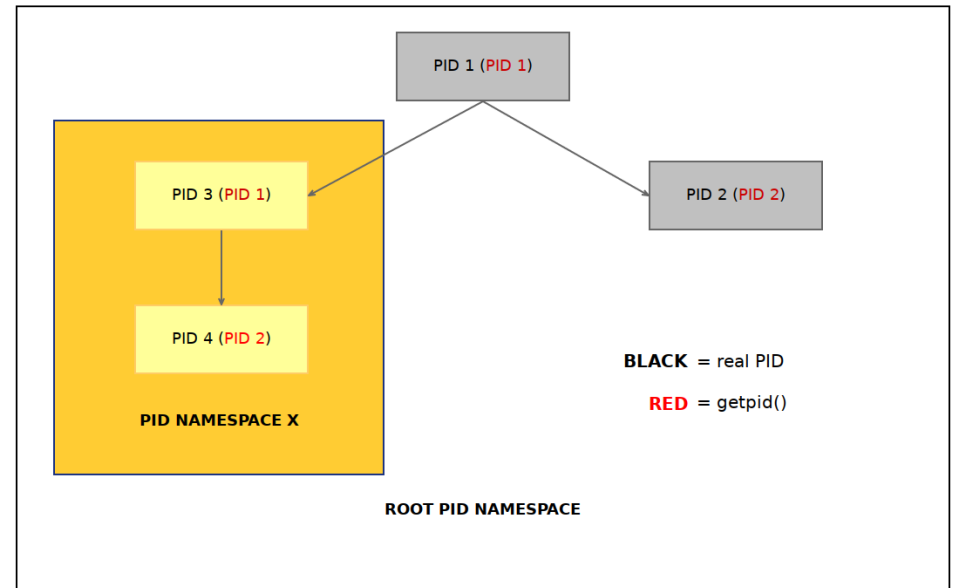
- Développé à l'origine par Google.
- Disponible depuis le noyau 2.6.24
- Cgroups permet la limitation de ressources, la comptabilité, le contrôle, la priorisation



# Qu'est qu'un conteneur ?

## Namespaces :

- Développé à l'origine par Bell Lab et disponible depuis le noyau 2.4.19
- Permettent de disposer d'un **environnement distinct** pour PID, réseau, points de montage, utilisateurs et groupes, etc.



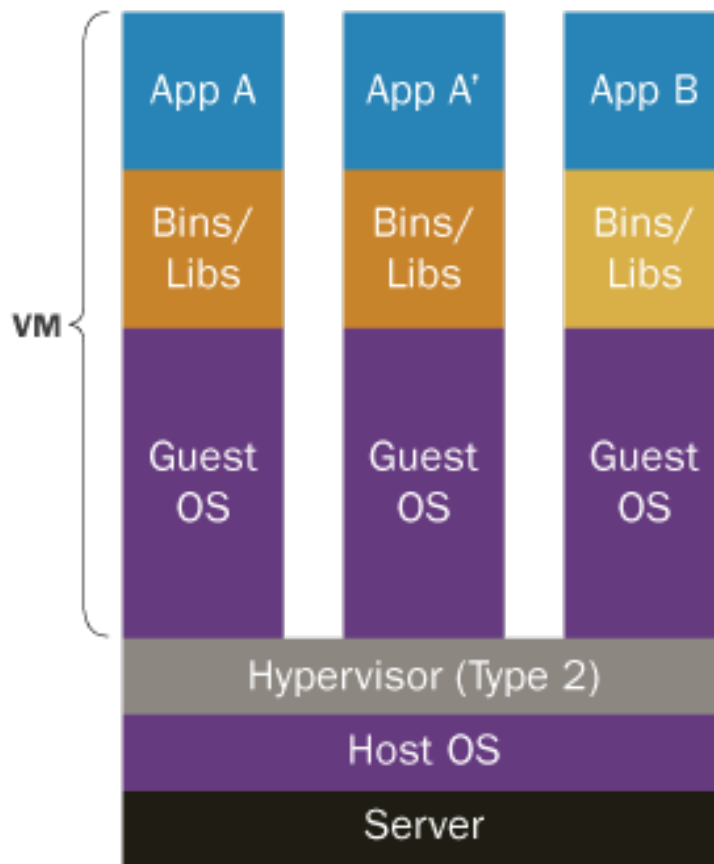
# Qu'est qu'un conteneur ?

- ✓ En bref, un **conteneur linux** est :
  - Un *RootFS*
  - Une *configuration Cgroups*
  - Un *ensemble de namespaces*
- ✓ Pour plus de sécurité et d'isolation :
  - Selinux, apparmor, secomp, capabilities, iptables, etc
  - Configuration réseau dédié (via bridge/overlay et veth)

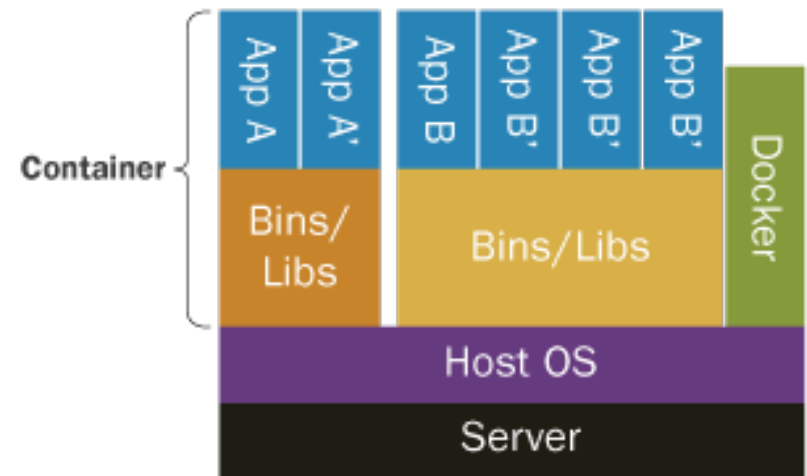


# VM vs Conteneurs

## Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



# Qu'est qu'un conteneur ?

DEMO

[https://github.com/ObjectifLibre/  
container-from-scratch](https://github.com/ObjectifLibre/container-from-scratch)



# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ Conteneur 101
- ✓ **Le cas Docker**
- ✓ Open Container Initiative
- ✓ Moby Project
- ✓ Cloud Native Computing Foundation
- ✓ Ecosystème opensource
- ✓ Q/R





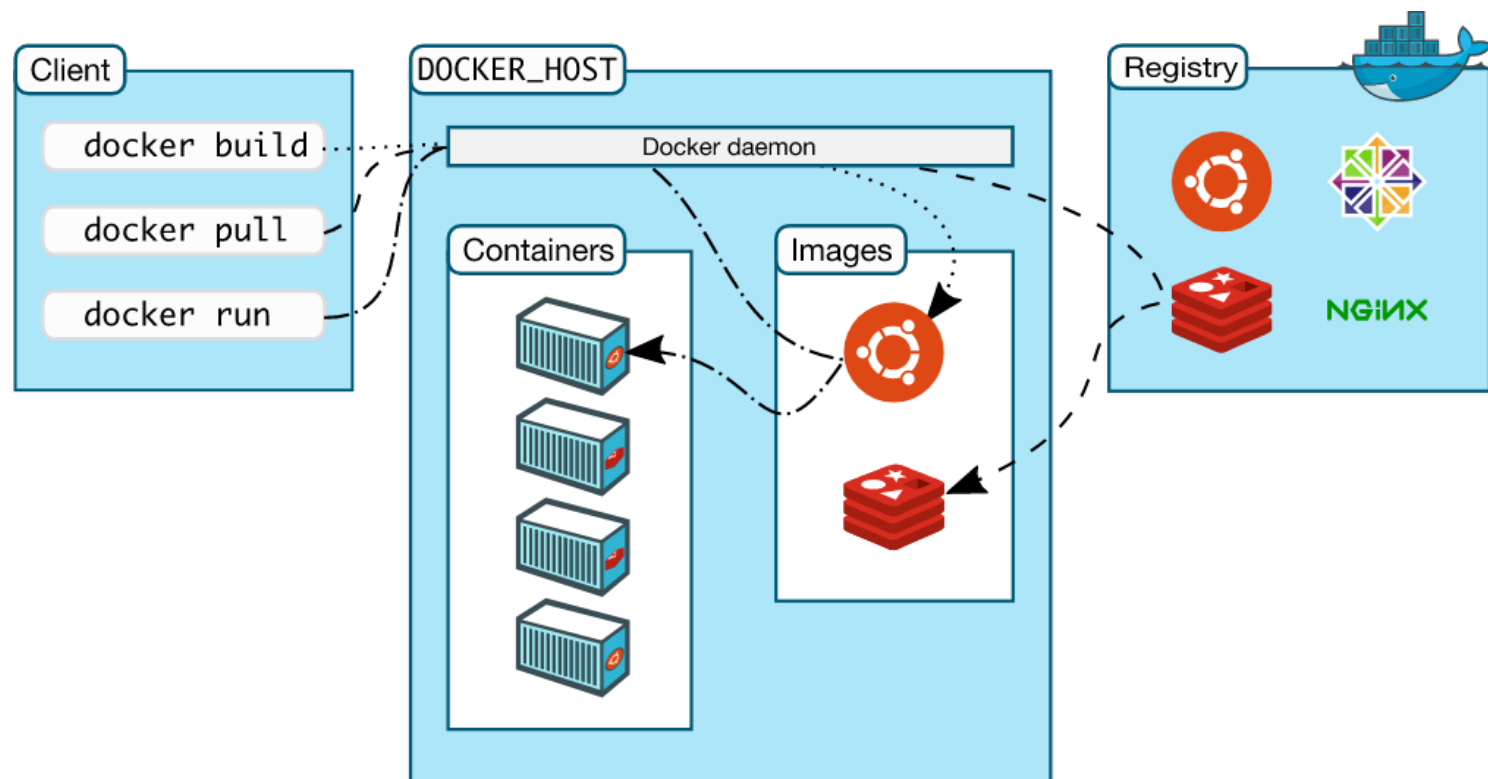
# Projet Docker



- ✓ 2013 – Arrivé de l'outil Docker
  - Outil pour simplifier la création de conteneurs linux en exploitant des technologies noyaux existantes
  - **Outil de création et distribution d'application portable via le concept d'image**
- ✓ Développement interne reversé en opensource dans le **projet Docker**
- ✓ Technologie en **très forte adoption** dès le début
- ✓ Docker Inc. abandonne l'activité de PaaS historique pour supporter le développement de la technologie et de la communauté



# Projet Docker



**Build – Ship – Run**



# Projet Docker



- ✓ Conteneur selon Docker
  - Mono-processus (approche micro-service)
  - Isolé via cgroups/namespaces
  - Isolé niveau réseau par défaut
  - Stockage éphémère par défaut
  - Sécurisé par défaut : seccomp, apparmor, capacités réduite, ...
- ✓ CLI simplifié pour la gestion d'un conteneur :
  - *docker container run nginx*



# Projet Docker



## ✓ Image

- rootfs redistribuable sous forme de couches binaire multiples + manifest (commande, user, volumes, etc)
- Dédicée à l'exécution d'une application en particulier

## ✓ Dockerfile

- format de description de la construction d'un rootfs

## ✓ CLI de gestion relativement simple :

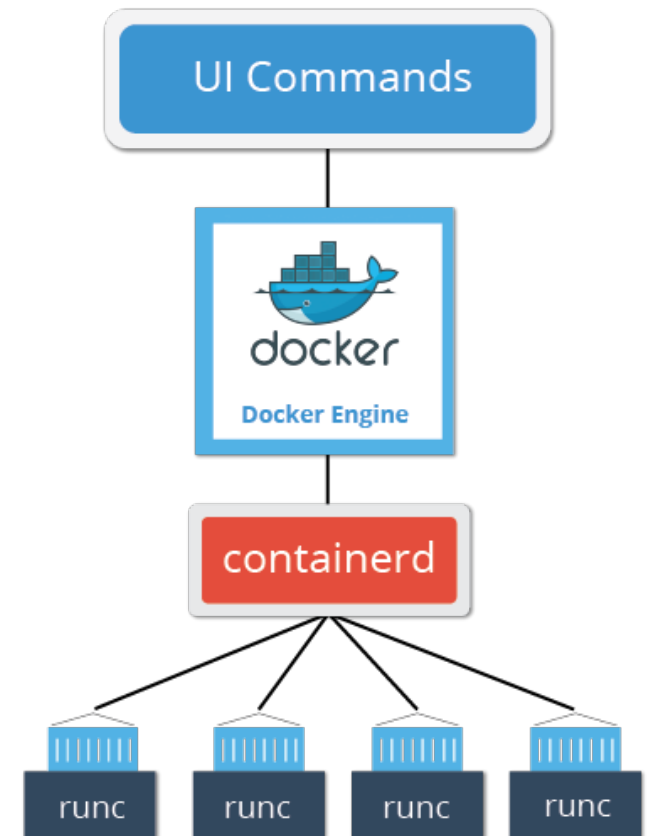
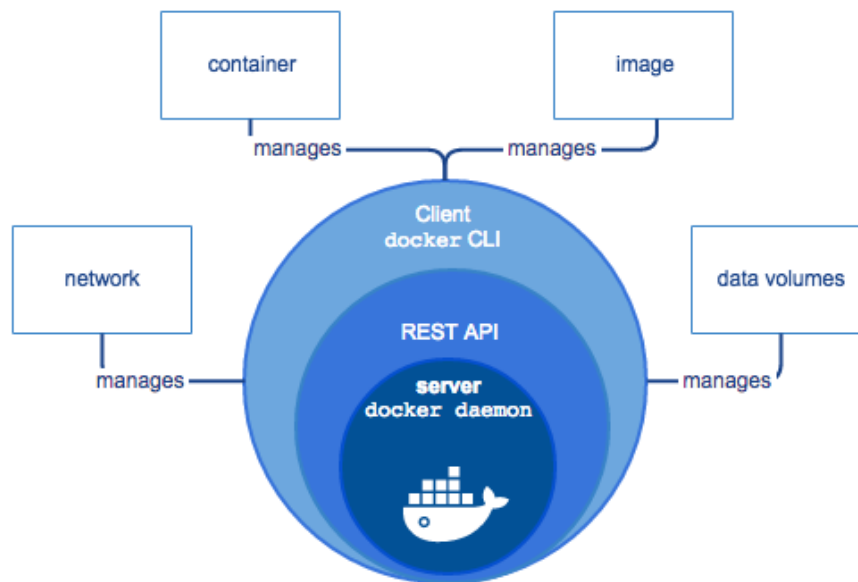
- *docker image build/tag/push/pull*



# Projet Docker



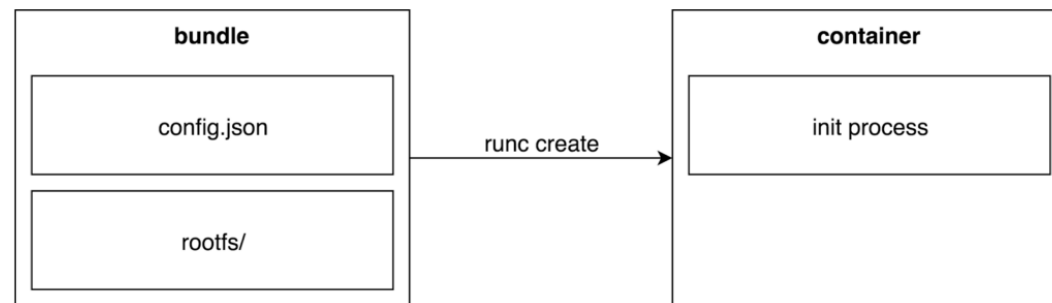
## ✓ Architecture



# RunC



- ✓ Implémentation de référence d'un moteur OCI
- ✓ Composants très bas niveau
  - Bibliothèque en Go (libcontainer)
  - CLI
- ✓ Travail en cours pour créer des conteneurs non privilégiés
- ✓ **1 seule fonction** : créer (au sens cgroups/namespaces/mounts) un conteneur et gérer l'exécution du processus défini dans la spec OCI



- ✓ Moteur de gestion de conteneurs mono-hôte
  - Exécution et supervision de conteneurs
    - Basé sur RunC
  - Distribution et stockage local des images OCI
  - Support multi-tenants
  - API gRPC – Démon + librairie cliente en Go
- ✓ Périmètre fonctionnel volontairement limité
  - Approche par composant
  - Focus sur performances, stabilité et sécurité



# Projet Docker



- ✓ Docker Community Edition
  - API REST + CLI
  - Gestion du cycles de vies des conteneurs
  - Gestion de réseau (plugin)
  - Gestion de stockage (plugin)
  - Orchestration intégrée (Swarm) pour la distribution de conteneurs sur un cluster
- ✓ Docker Enterprise Edition : extension de Docker CE
  - GUI
  - ACL
  - Registry sécurisée avec scan de vulnérabilité, politique de gestion des images





# Projet Docker



- ✓ Autres produits Docker
  - Docker-machine
  - Docker-compose
  - Registry
  - Docker4Mac / Docker4Win
    - Client natif
    - Serveur exécuté dans une VM



# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ Conteneur 101
- ✓ Le cas Docker
- ✓ **Open Container Initiative**
- ✓ Moby Project
- ✓ Cloud Native Computing Foundation
- ✓ Ecosystème opensource
- ✓ Q/R



# Open Container Initiative



- ✓ 2014 : CoreOS crée un runtime et un format alternatif (Rkt et AppC)
- ✓ 2015 : pour éviter la multiplication des formats, Docker et CoreOS (entre autre) décide de s'entendre au travers de la création d'une fondation en charge de gérer la standardisation des modèle de conteneurs : **Open Container initiative**



# Open Container Initiative



## ✓ Spécifications

- **Image** : spécification du format d'image multi-couche, basé sur AppC et Docker Image v2
  - **Runtime** : spécification de la configuration d'un conteneur, basée principalement sur libcontainer.
  - **Distribution** : réflexion en cours sur la spécification d'un API pour la distribution des images
- ✓ Cela doit permettre de changer simplement le moteur bas-niveau (OCI) au sein d'un moteur de conteneur complet comme Docker Engine par exemple.



# Open Container Initiative



- ✓ Implémentation officielle : runC
- ✓ Alternatives :
  - Nvidia-oci : moteur permettant la mise à disposition simplifiée des GPU à un conteneur
  - Katacontainer : exécution du conteneur dans un noyau ultra-léger dédié → simplicité du conteneur avec l'isolation de la VM
  - Railcar : implémentation en RUST par Oracle





Demo (en mode non privilégié)

<https://github.com/aolwas/rootless-ctr>



# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ Conteneur 101
- ✓ Le cas Docker
- ✓ Open Container Initiative
- ✓ **Moby Project**
- ✓ Cloud Native Computing Foundation
- ✓ Ecosystème opensource
- ✓ Q/R



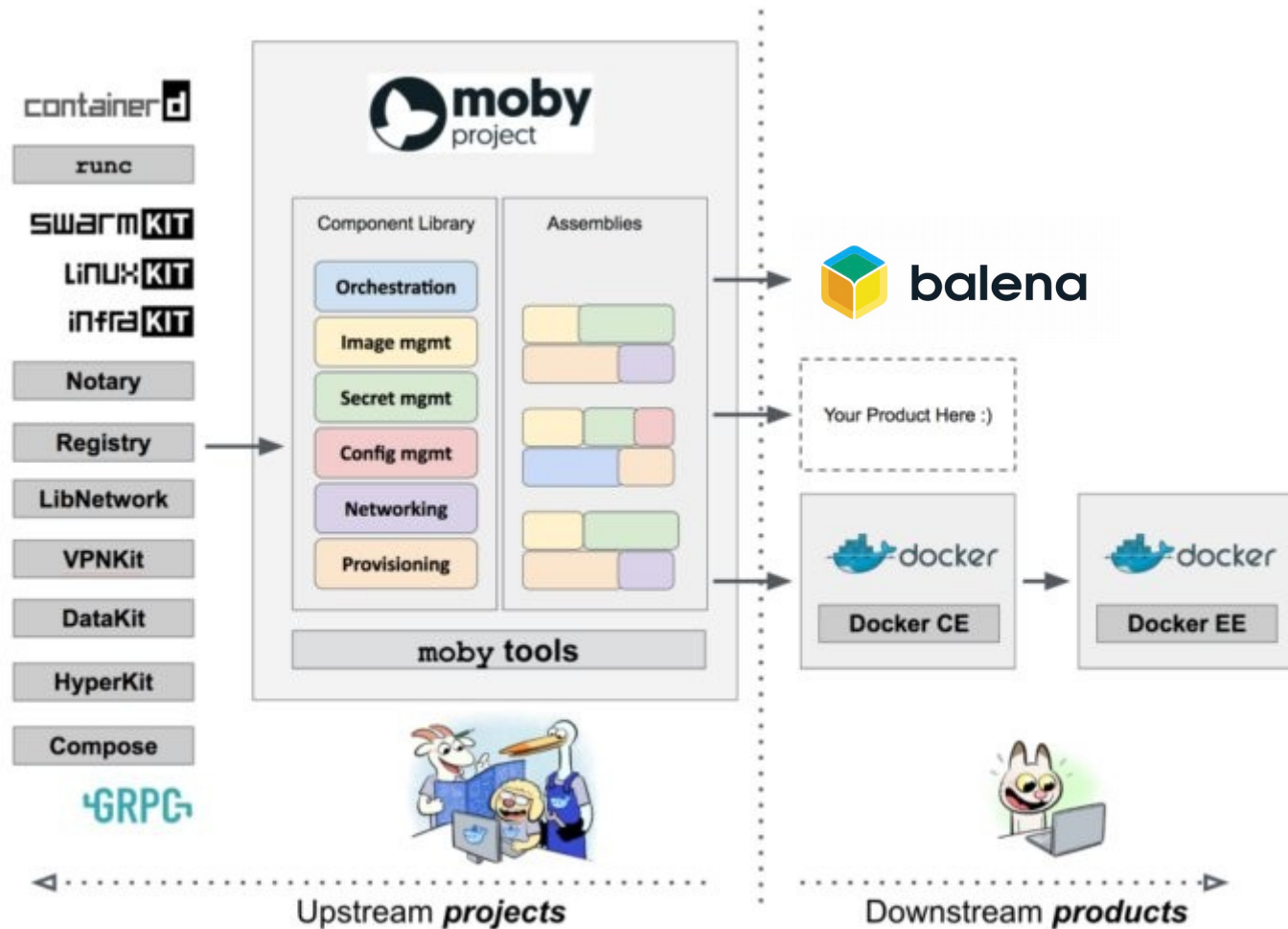
# Projet Docker vs Docker Inc

- ✓ Gouvernance du projet par Docker Inc uniquement
  - Divergences de vision technologique → Fork (Rkt et le format d'image Appc)
  - Introduction de fonctionnalités concurrentes d'autres projets opensources (swarmkit vs k8s)
- ✓ Commercialisation de produits Docker Inc estampillés « Docker ».
  - Divergence des roadmaps Docker Inc et autres contributeurs
  - Ambiguïté projet opensource/projet commerciale





# Projet Moby

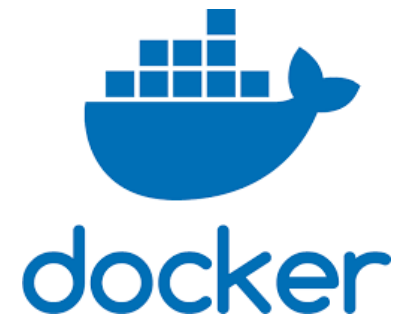


# Projet Moby / Docker Inc.

- ✓ Le démon *dockerd* est développé par Moby
  - Bientôt renommé *moby-engine*



- 
- ✓ Le client *docker* est développé par Docker Inc.
    - Reste Opensource
  - ✓ Les instaleurs *Docker4xxx* et la plateforme Entreprise sont développés par Docker Inc.
    - ⇒ Pas d'impacts pour les utilisateurs actuels de la plateforme Docker



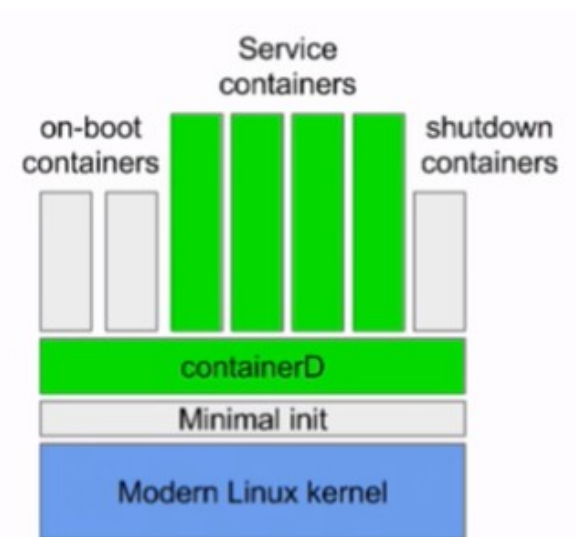
- ✓ Ensemble d'outils pour la création d'OS Linux spécifique, léger, immuable et basé exclusivement sur les conteneurs
- ✓ Description de l'OS via fichier de configuration YAML
- ✓ Système de packages (disponible sous forme d'images)
- ✓ Architecture ouverte et extensible
- ✓ Docker's motto : *Build, Push, Run*
- ✓ Plusieurs formats de sorties (raw, qcow, vmdk, iso, etc) et services de cloud (gce, azure, openstack, etc)



# Linuxkit



```
# connect: nc localhost 6379
kernel:
  image: linuxkit/kernel:4.9.74
  cmdline: "console=tty0 console=ttyS0 console=ttyAMA0"
init:
  - linuxkit/init:9250948d0de494df8a811edb3242b4584057cfe4
  - linuxkit/runc:abc3f292653e64a2fd488e9675ace19a55ec7023
  - linuxkit/containerd:e58a382c33bb509ba3e0e8170dfaa5a100504c5b
onboot:
  - name: dhcpdd
    image: linuxkit/dhcpdd:0d59a6cc03412289ef4313f2491ec666c1715cc9
    command: ["/sbin/dhcpdd", "--nobackground", "-f", "/dhcpdd.conf", "-1"]
services:
  - name: getty
    image: linuxkit/getty:22e27189b6b354e1d5d38fc0536a5af3f2adb79f
  env:
    - INSECURE=true
# Currently redis:4.0.6-alpine has trust issue with multi-arch
# https://github.com/docker-library/official-images/issues/3794
  - name: redis
    image: redis:4.0.5-alpine
  capabilities:
    - CAP_NET_BIND_SERVICE
    - CAP_CHOWN
    - CAP_SETUID
    - CAP_SETGID
    - CAP_DAC_OVERRIDE
  net: host
trust:
  org:
    - linuxkit
    - library
```



- ✓ Utilisé dans :
  - Docker4Desktop (Mac, Win)
  - Docker4Cloud (AWS, Azure)
  - IBM Bluemix container platform
- ✓ Image Linuxkit LDAP (qcow) ~ 80 Mo



# Autres projets

- ✓ **Infrakit** : bibliothèque pour la construction déclarative d'infrastructures auto-cicatrisantes
- ✓ **Buildkit** : bibliothèque pour la construction de chaîne de « construction » d'artéfacts (réimplémentation de *docker build*)
- ✓ **Notary** (CNCF) : outil permettant de garantir la provenance et/ou la validité d'une collection de contenus
- ✓ **Swarmkit** : librairie pour la création de systèmes distribués
- ✓ **Distribution** : ensemble de composants pour créer, distribuer et stocker du contenu

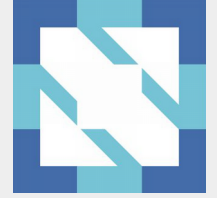


# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ Conteneur 101
- ✓ Le cas Docker
- ✓ Open Container Initiative
- ✓ Moby Project
- ✓ **Cloud Native Computing Foundation**
- ✓ Ecosystème opensource
- ✓ Q/R



# Cloud Native Computing Foundation



- ✓ Encadre le développement des briques logicielles pour la mise en œuvre de plateformes natives cloud
  - Conteneur, orchestration, tracing, monitoring, stockage, database, etc
- ✓ Créé en 2015 par Google pour l'encadrement du développement de Kubernetes, parrainée par la fondation Linux
- ✓ 23 projets dont Containerd et Rkt





# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ Conteneur 101
- ✓ Le cas Docker
- ✓ Open Container Initiative
- ✓ Moby Project
- ✓ Cloud Native Computing Foundation
- ✓ **Ecosystème opensource**
- ✓ Q/R



# Engines (alternatives)

- ✓ CRI-O (compatible OCI)
- ✓ Garden (compatible OCI)
- ✓ LXC (orienté système)
- ✓ Systemd-nspawn (orienté système)
- ✓ OpenVZ (orienté système, nécessite un noyau kernel custom)
- ✓ **Singularity, shifter, Charliecloud (pour utilisation HPC)**



# Orchestrateurs

- ✓ Docker (via swarmkit)
- ✓ Kubernetes
- ✓ Mesos/Marathon
- ✓ Nomad



# PaaS

- ✓ Cloudfoundry
- ✓ Openshift (k8s)
- ✓ *Gitlab Autodevops (k8s)*
- ✓ *Jenkins X*
- ✓ *Docker EE*



# OCI Image tools

- ✓ Docker
- ✓ **Img** : générateur d'image basé sur buildkit et Dockerfile
- ✓ **Skopeo** : outil de gestion de registry
- ✓ **Umoci** : outil de manipulation d'images au format OCI
- ✓ Buildah : outil de création et de manipulation d'images au format OCI
- ✓ Ansible-container : construction d'image OCI à l'aide de rôle ansible (nécessite Docker)
- ✓ Packer : outil générique de création d'images virtuelles



# Sommaire

- ✓ Objectif Libre en 3 slides
- ✓ Conteneur 101
- ✓ Le cas Docker
- ✓ Open Container Initiative
- ✓ Moby Project
- ✓ Cloud Native Computing Foundation
- ✓ Ecosystème opensource
- ✓ Q/R



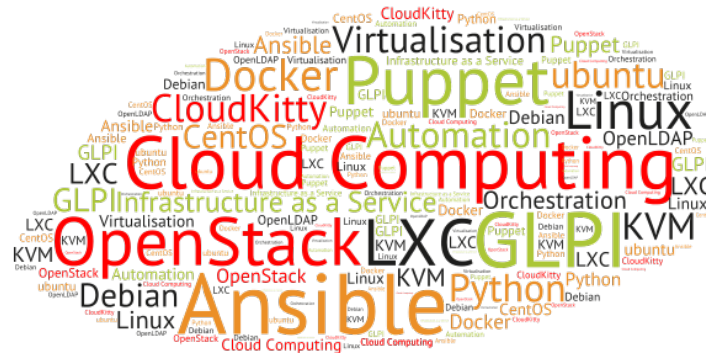
# Conclusion


Q / R



[www.objectif-libre.com](http://www.objectif-libre.com)

## Restons en contact



 @objectiflibre

 Objectif-libre

 ObjectifLibre

 notre newsletter

L'essentiel Cloud&DevOps  
[olib.re/abo-pause](http://olib.re/abo-pause)



Envie d'en savoir plus ? Besoin d'étudier un projet sur mesure ?  
Contactez-nous au 05.82.95.65.36 ou [contact@objectif-libre.com](mailto:contact@objectif-libre.com)

