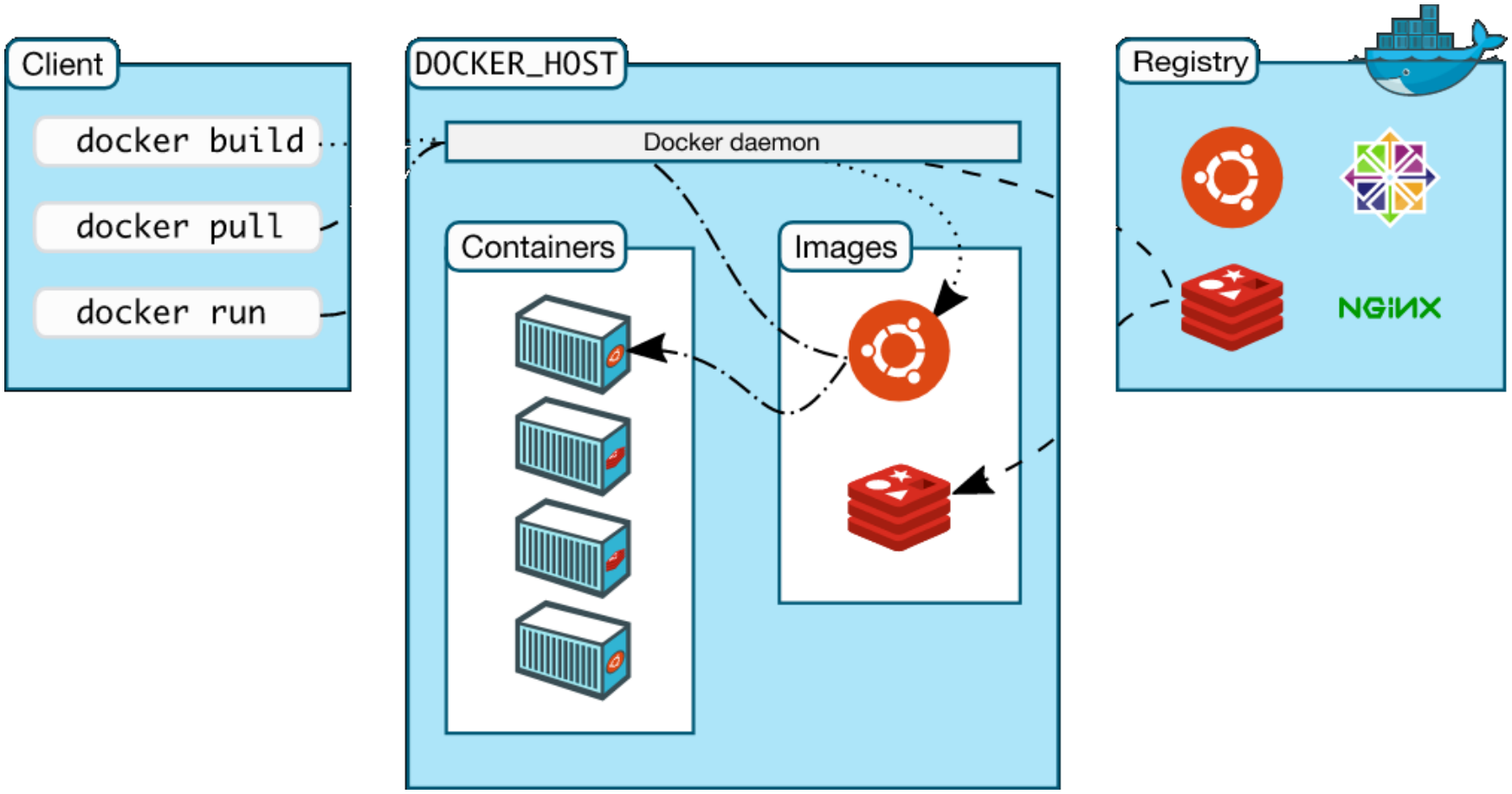
Docker Internals

2020

Conteneurs Linux

Architecture



Conteneurs Linux

- C’est quoi un conteneur ?

  - Les conteneurs partagent le noyau (*kernel*) de l’hôte ;
  - Les conteneurs utilisent la fonctionnalité du noyau qui permet de regrouper des processus pour contrôler les ressources ;
  - Les conteneurs assurent l’isolation via l’utilisation d’espaces de noms différents (*namespaces*) ;
  - Les conteneurs font penser à des VM légères (moins d’utilisation de ressource, plus rapides), mais **ce ne sont pas des machines virtuelles !**

Conteneurs Linux

- Les composants d’un écosystème de conteneurs incluent :
  - Un runtime
  - Un système de mise à disposition d’images
  - De l’outillage
- Mais … il n’y a pas de notion de conteneur dans le noyau Linux, alors, comment ça marche ?

L’histoire des conteneurs

- Chroot, 1982
- FreeBSD Jails, 2000
- Solaris Zones, 2004
- Meiosys – metacluster avec checkpoint/restauration, 2004-2005
- Linux OpenVZ, 2005
- AIX WPARs, 2007
- LXC, 2008
- Systemd-nspawn, 2010-2013
- Docker, 2013
  - Basé sur LXC, puis sur libcontainer puis sur appC, puis sur runC
- …

Par rapport aux …

- Gestionnaires de packages
  - Pas de problème avec les librairies partagées, de gestion des dépendances
- Gestionnaires de configuration logicielle
  - L’image est pré-configurée, pas de variation selon la cible

Synthèse

- Docker met à disposition une image qui comprend tout, et c'est la même image quelque soit le lieu de son instanciation ;

- Docker est (très) rapide, le démarrage d'un conteneur prend à peine quelques secondes. Très peu de surcharge (cpu, mémoire, …), permettant une haute densité (plein de conteneurs sur une machine pas très puissante, …) ;

- Adoption rapide parce que construire une image est très facile, le DSL Dockerfile est très simple et très puissant.

Pourquoi Docker ?

- Le seule système actuellement qui propose l'ensemble des fonctionnalités :
  - Gestionnaire d'image ;
  - Isolation des ressources ;
  - Isolation du système de fichiers ;
  - Isolation du réseau ;
  - Gestion du changement
  - Partage
  - Gestion des processus
  - Découverte des services (DNS depuis la version 1.10)

Comment ?

Kernel Namespaces

- Permettent l’isolation :
  - Des arborescence de processus (PID namespace)
  - Des points de montages (MNT namespace) wc –l /proc/mounts
  - Des réseaux (Net namespace) ip addr
  - Des utilisateurs/UIDs (User namespace)
  - Des noms d’hôtes (UTS namespaces) hostname
  - Des communications iter processus (IPC namespaces) ipcs

Cgroups

- Les groupes de contrôles (*control groups*, *cgroups*) du kernel permettent de gérer les ressources utilisées par les processus, un peu de contrôle d'accès, ainsi que d'autres choses, comme la possibilité de « geler » un groupe de processus.

- Les cgroups consistent en une hiérarchie (arbre) par ressource (cpu, mémoire, …).

Cgroups

- On peut créer des sous-groupes dans chaque hiérarchie.
- Chaque processus sera un nœud pour chaque ressource.

- Mémoire : accounting, limites (soft, hard), notifications
- CPU : poids
- CPUSet : associer un groupe à un CPU
- BlkIO : mesures et limites
- Net-cls et net_prio cgroup
- Devices cgroup

Comment le noyau expose les cgroups ?

- Au moyen d'un pseudo *file system*

/sys/fs/cgroup

- Pour associer un processus à un groupe, envoyer l'ID du processus dans le fichier spécial tasks dans le cgroup :

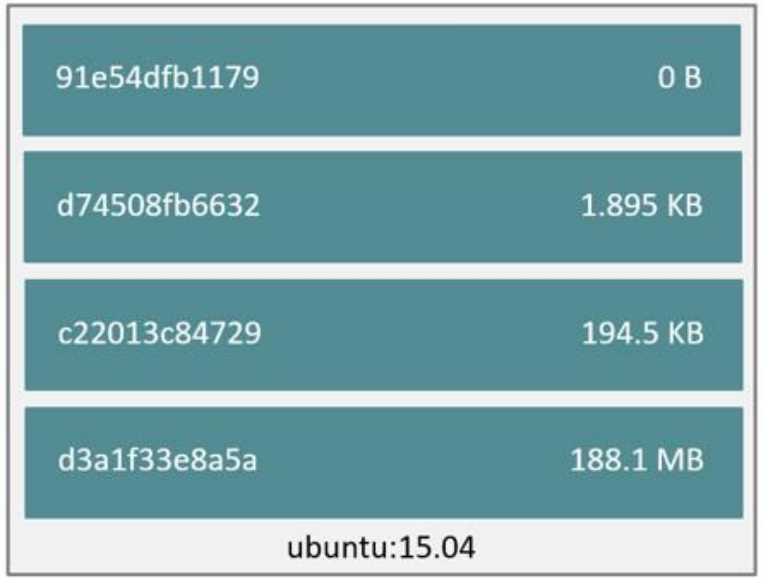
Echo $PID > /sys/fs/cgroup/…/tasks

IPTables (réseau)

- Isolation créée via un commutateur virtuel (*virtual switch*) au niveau du noyau

Images

- Chaque image Docker référence une liste de couches en lecture seules qui représentent les différences entre les filesystems.
- Les couches sont empilées les unes sur les autres et forment la base du file system racine (*rootfs*)



Image

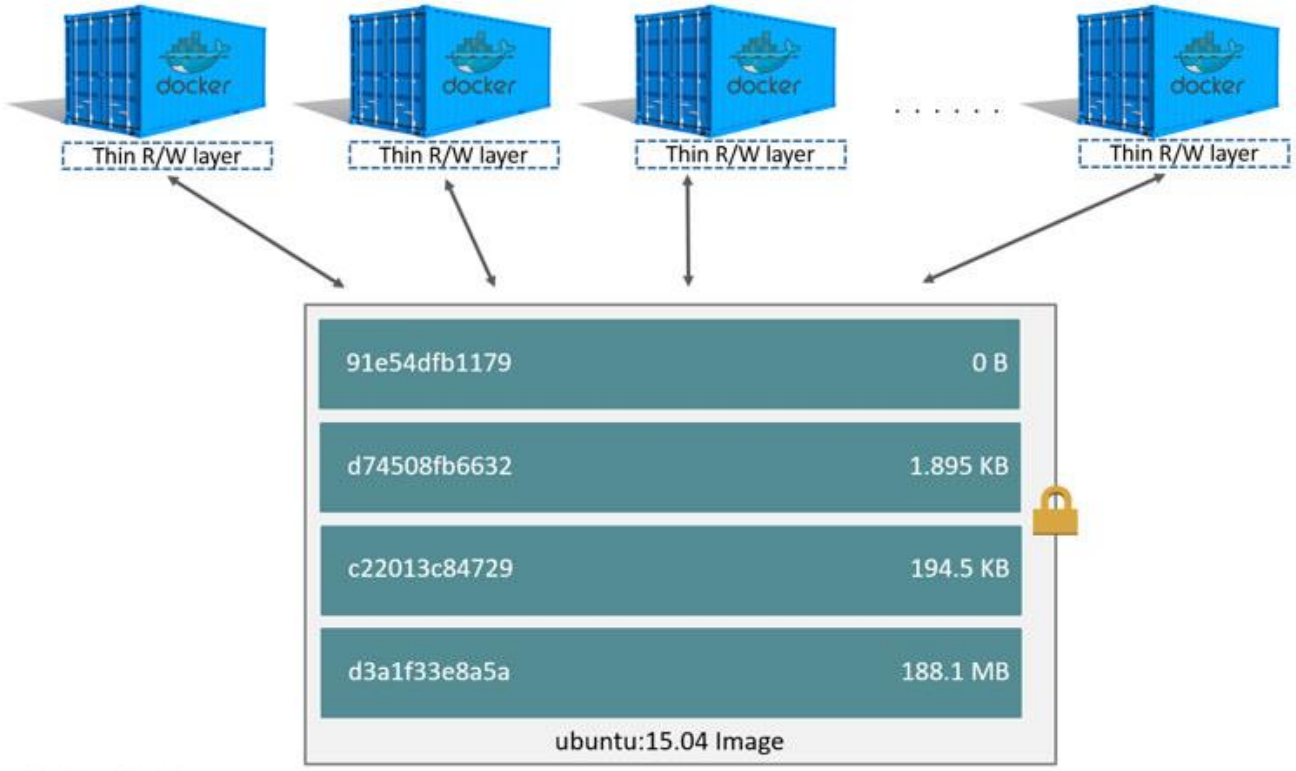
Images

- Lorsqu'un conteneur démarre, Docker prépare le rootfs et utilise chroot pour assurer l'isolation du filesystem (comme LXC).

- (Une des innovations de Docker a été la généralisation du concept de copie sur lecture (*copy on write*) pour accélérer la préparation du rootfs)

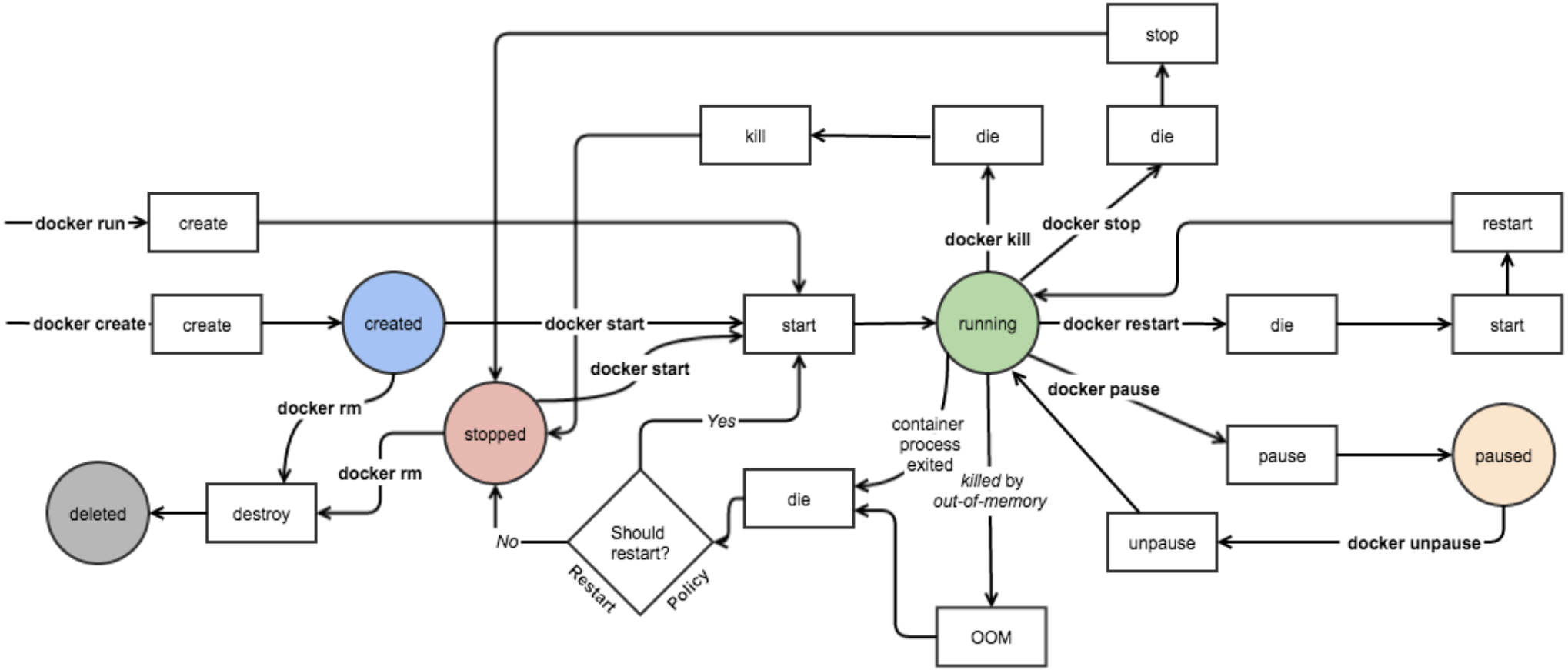
Copy on write

- Lorsque Docker créé un conteneur, il créé une nouvelle fine couche dans laquelle on peut écrire, au dessus de la pile des couches de l'image.



L’API Docker

Docker Engine



Pour aller plus loin

Quelques URL

- https://docs.docker.com/engine/docker-overview/
- http://docker-saigon.github.io/post/Docker-Internals/