Système d'exploitation
Introduction à Docker

Juan Angel Lorenzo del Castillo

2019-2020

Conteneurs

C'est quoi un conteneur ?

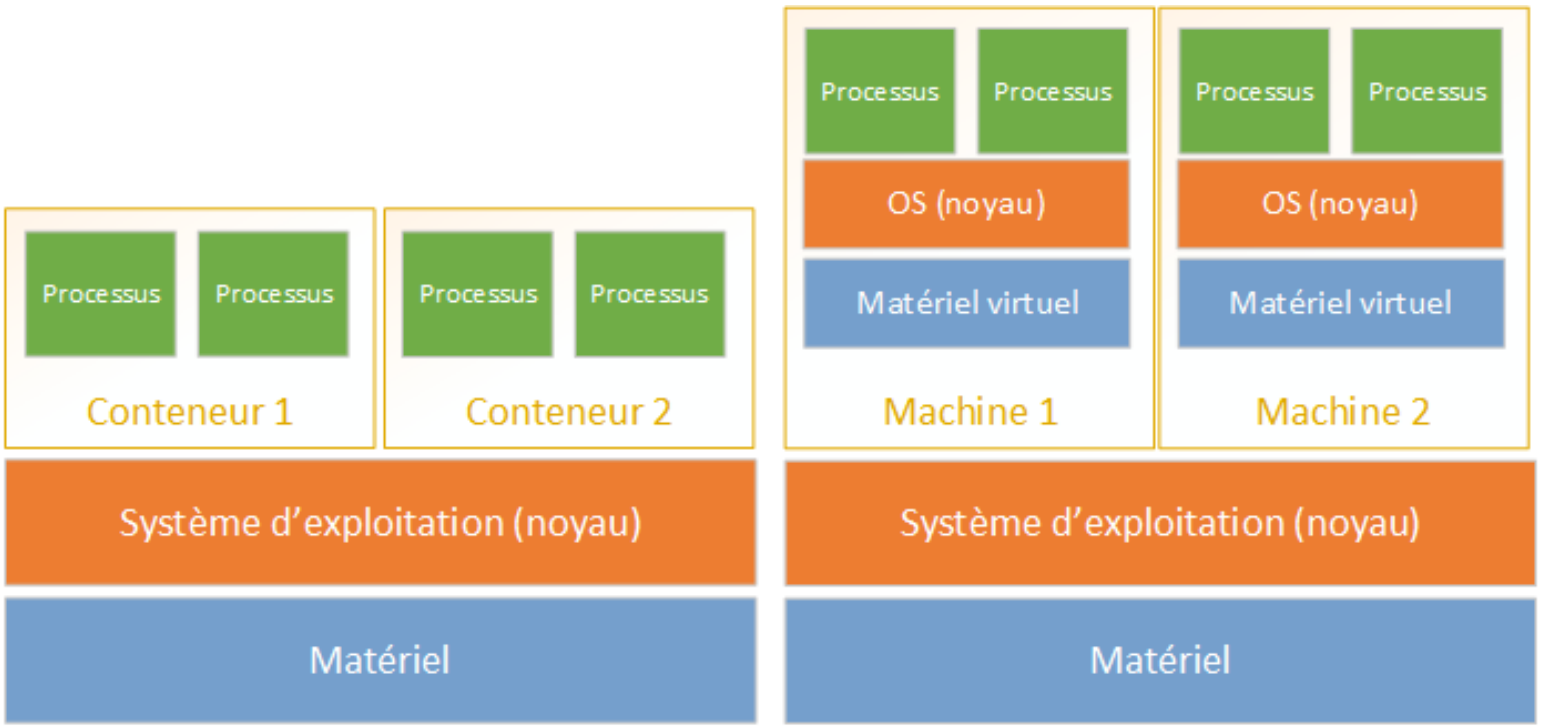
Permet d'isoler l'exécution des applications dans des contextes d'éxécution.

Similaire à la virtualisation, mais sans virtualisation :

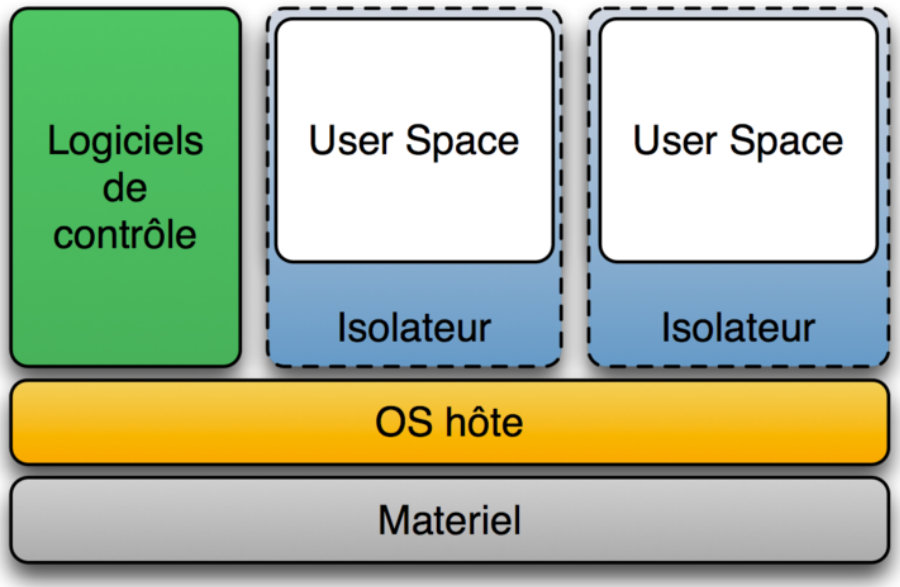
- Agnostique sur le contenu et le transporteur.
- Isolation et automatisation.
- Principe d'infrastructure consistante et répétable.
- Peu d'overhead par rapport à une VM.
- En gros, un super chroot (*chroot on steroids*).
- Un des points forts de Solaris depuis plusieurs années.
- Technologie utilisée chez Google avec son scheduler **Borg** depuis longtemps.

Inspiré par les slides de Johan Moreau (IRCAD/IHU), JDev 2015.

Différences entre VM et conteneur

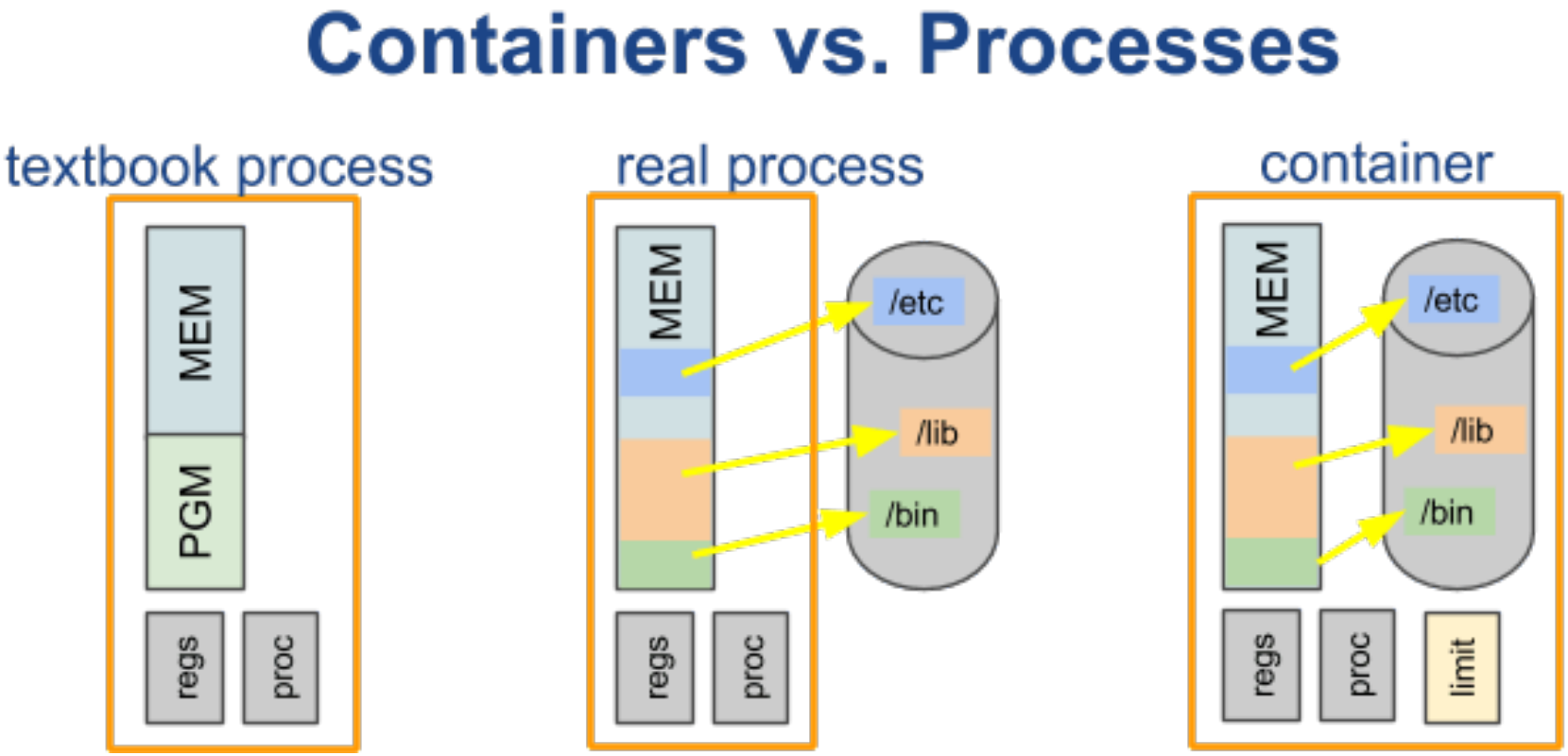


Chroot on steroids



- Avec son propre space de processus
- Avec sa propre interface de réseau
- Sans son propre `/sbin/init`
- Processus isolés
- Kernel partagé avec le host
- Permettent la création de plusieurs environnements similaires, avec des versions du logiciel et configurations identiques

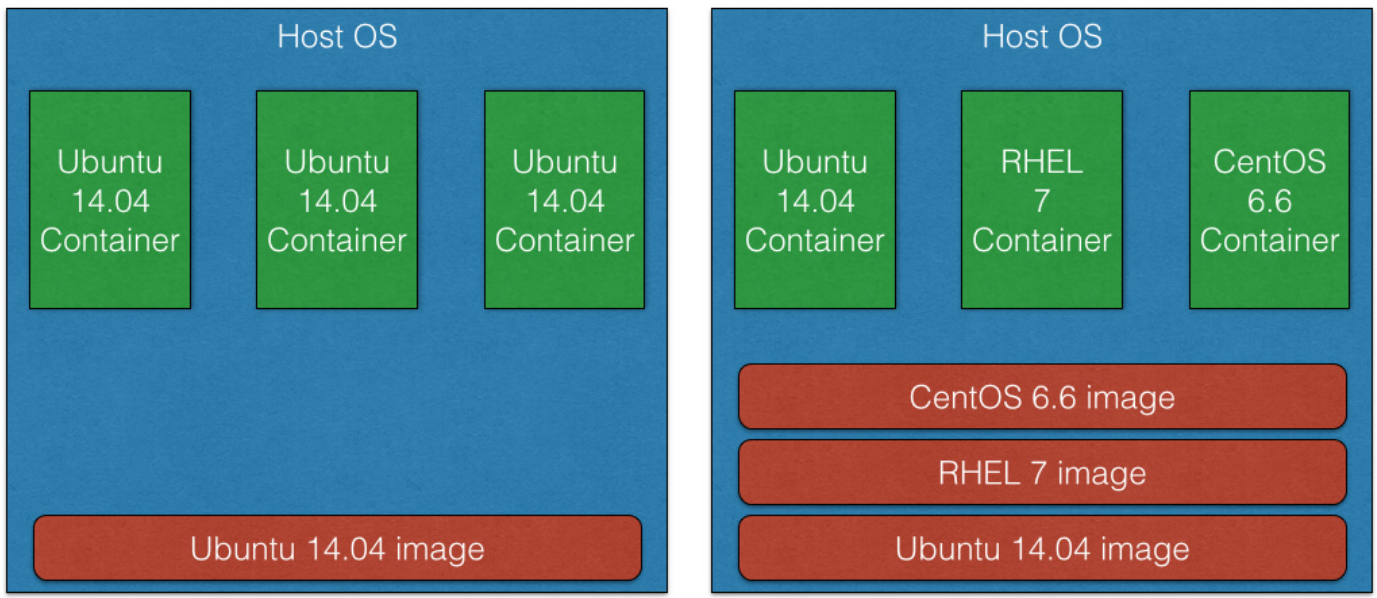
Différences entre processus et conteneur



Source : rightscale.com

Cas d'utilisation des conteneurs

1. Conteneur de Système d'exploitation
   - Environnement virtuel qui partage le kernel avec le SE du host
   - Mais isolé dans l'espace d'utilisateur
   - On exécute plusieurs processus et services
   - Pratique pour exécuter différentes distributions en utilisant des images (modèles)



Identical OS containers Different flavoured OS containers

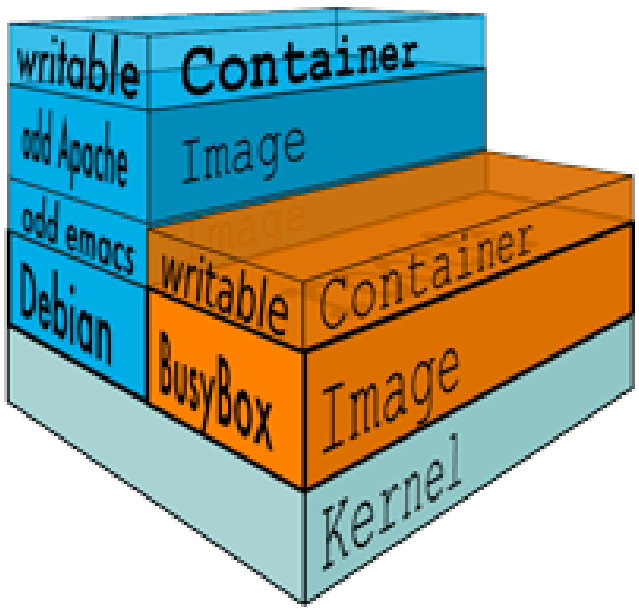
Source : risingstack.com

Cas d'utilisation des conteneurs

2 Conteneur d'application

- Un seul service ou application par conteneur
- **Microservices :** Décomposer une application grande dans plusieurs services petits
- Au lieu de mettre à jour toute une application, on met à jour les services concernés

Images



- Collection de **fichiers** de seule lecture
- **Images base** sur lesquelles on construit les autres images
- Images en **couches**
- Chaque couche est un **différentiel**
- **Conteneur** : Ensemble de processus et fichiers modifiés (lecture-écriture)

Plan

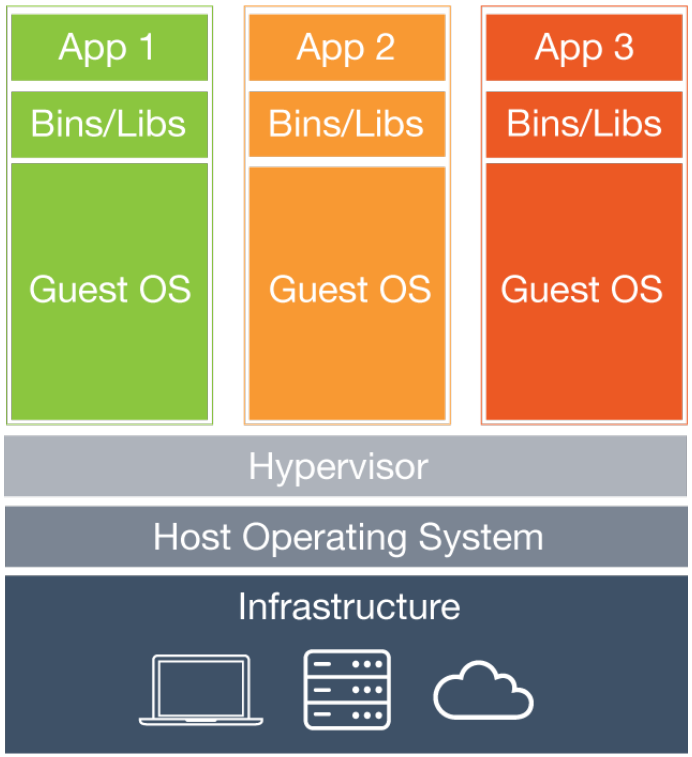
Docker

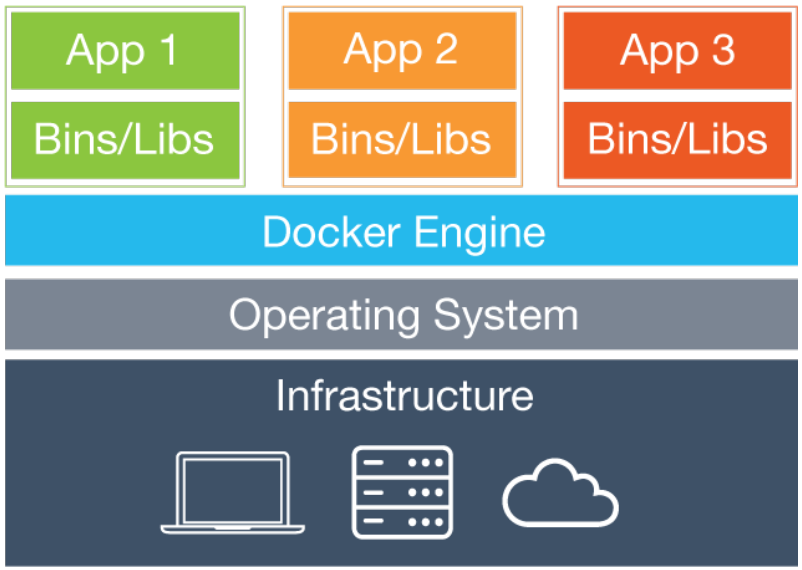
C'est quoi Docker ?

Docker Engine + Docker Hub = Docker Platform

- **Docker Engine** : Exécute les conteneurs
  - Écrit en `Go`
  - API REST
  - Construction des images
  - Partage d'images en utilisant des registres
- **Docker Hub** : Facilite la migration
  - Registres (repos) publiques
  - Registres privés
  - Construction automatique du logiciel

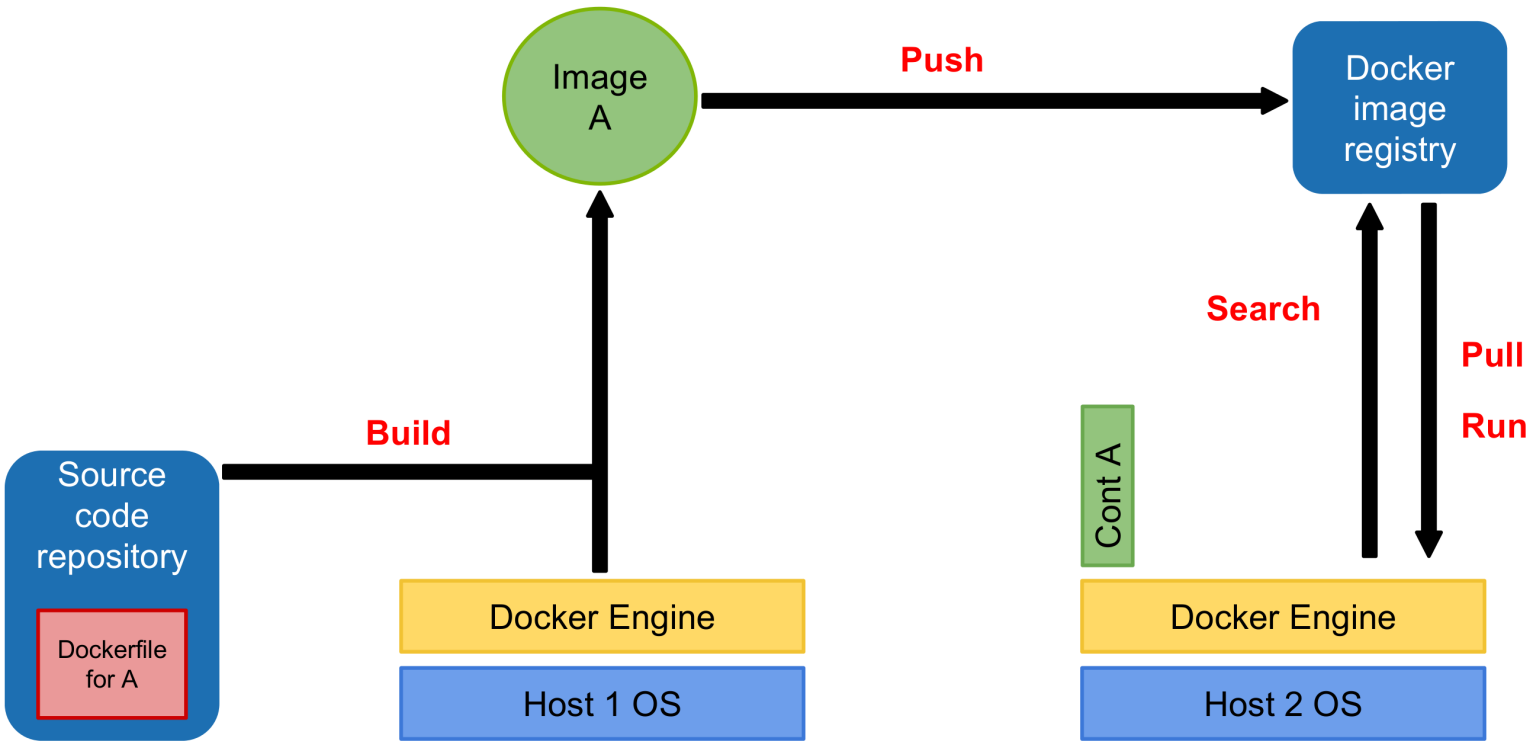
C'est quoi Docker ?





Source : Docker

Docker pour la gestion du logiciel



Source : Treeptik

Terminologie

- ***index*** **:** répertoire publique (https ://index.docker.io/)
- ***image*** **:** conteneur en lecture seule (snapshot)
- ***conteneur*** **:** élément manipulable
- ***dockerfile*** **:** fichier qui permet de construire une image Docker automatiquement
- ***docker-compose*** **:** orchestration des conteneurs (démarrage, arrête, configuration des liens entre conteneurs, etc.)
- ***run*** **:** créer un conteneur (à partir d'une image)

Exemple

```
$ docker run -i -t ubuntu /bin/bash
```

- **run** : lance le conteneur
- **-i -t** : demande un terminal en mode interactif
- **ubuntu** : l'image à utiliser pour ce conteneur
- **/bin/bash** : exécute bash dans le conteneur

```
$ docker run -i -t ubuntu /bin/bash
root@0bc82356b52d9:/# cat /etc/issue
Ubuntu 14.04.2 LTS
root@0bc82356b52d9:/# exit
```

Commandes utiles

- **`docker search ubuntu`** #give ubuntu images from public index (official /trusted)
- **`docker pull stackbrew/ubuntu`** #pull latest stackbrew/ubuntu images
- **`docker history stackbrew/ubuntu`** #view history for this image
- **`docker images`** # show local images
- **`docker run -i -t stackbrew/ubuntu /bin/bash`** #run this image / create container
- **`docker run -t -i -link redis :db -name webapp ubuntu bash`** #link 2 containers
- **`docker ps`** #show active containers (-a to show all containers)
- **`docker logs myUbuntu`**
- **`docker attach myUbuntu`** #retake the hand on the container
- **`docker run -d -p 8888:80 ubuntu`** #export 8888 on master