Une introduction à Docker en 15mn

Hervé DARRITCHON, Architecte API
Orange Software Expert, Développeur Sénior
DUT Informatique promo 93

JdA2016

iut BAYONNE PAYS BASQUE

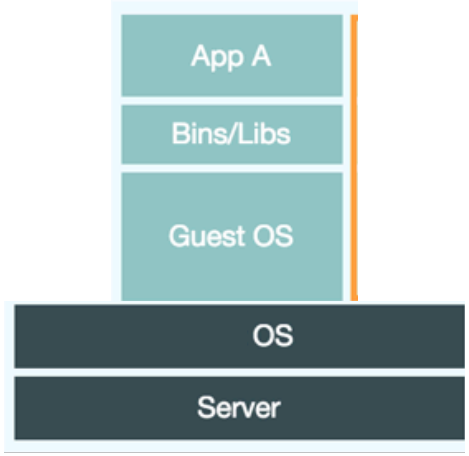
orange™

Agenda :

- Pourquoi les containers légers ?
  - Une évolution naturelle (physique, vm, container)
- Qu'est-ce qu'un container ?
  - Un peu de plomberie
- Pourquoi Docker
- Comment ça marche
- Les usages
  - Développement, intégration, production
- Le Workflow
- Synthèse
- L'éco-système docker
- Conclusion

Evolution des serveurs 1/3 ?

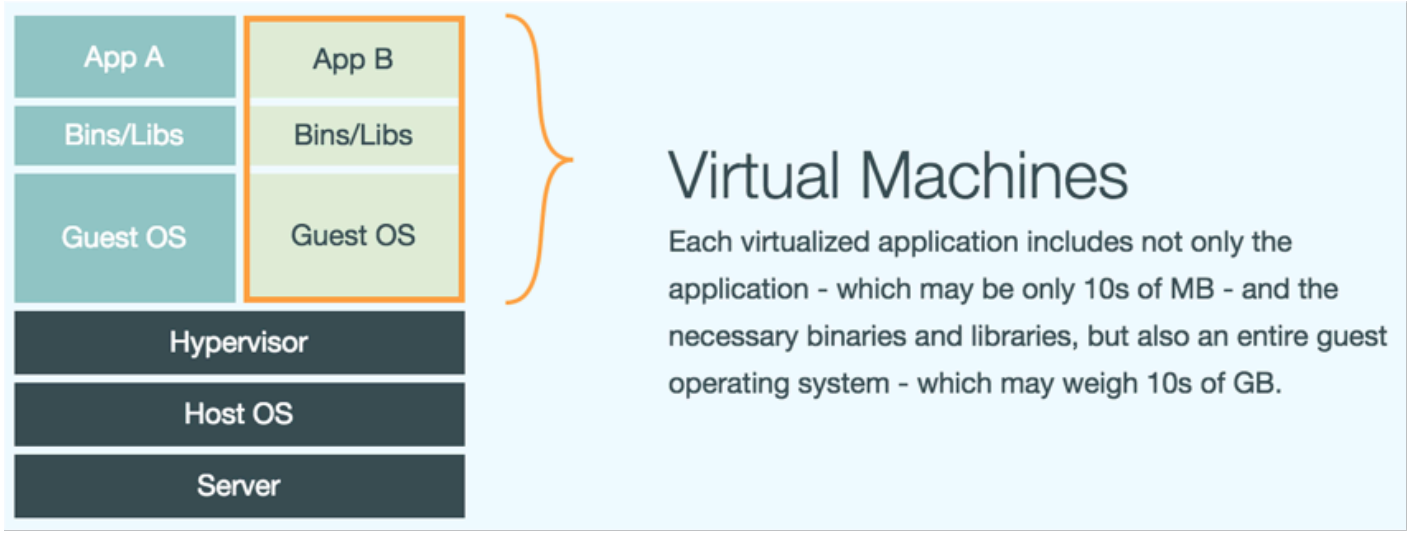
Serveurs physiques



« Partage de l'espace logistique »

Evolution des serveurs 2/3 ?

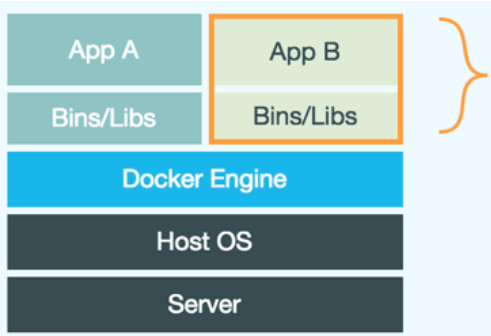
Virtualisations



« Partage des ressources d'une machine »

Evolution des serveurs 2/3 ?

Containeurs légers



Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

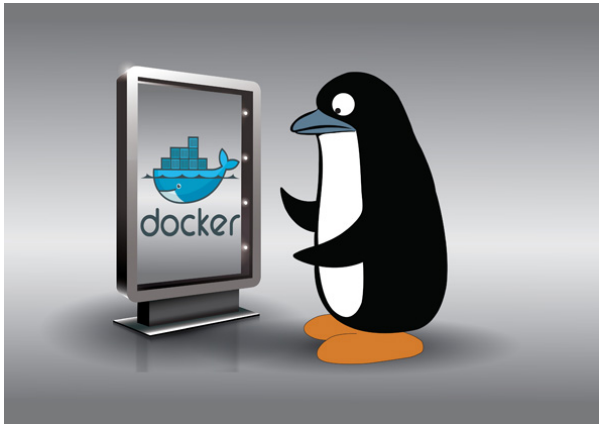
Qu’est-ce qu’un container ?

Un peu de plomberie : aspects théoriques

- De quoi sont faits les containers ? Les briques de bases :
  - Control groups (ou cgroups) => limitent ce qu’on peut consommer
    Permettent de limiter, prioriser, mesurer l’utilisation, **contrôler l usage des ressources** :
    Mémoire, cpu, disk (via blkio), reseau, devices
  - Namespaces => **limitent ce qu’on peut voir**
    pid, net, mnt, ipc, user…
  - Copy-on-write storage
    Au départ, on considère qu il suffit d avoir un pointeur sur le stockage à accéder.
    La copie n’est faite que si le besoin d écriture apparait
- Containers runtimes :

  Sur les mêmes bases
  - LXC
  - rkt, runC

  Sur d’autres bases
  - Jails/Zones



« Docker n’est qu’une solution de containeur parmi d’autres »

Pourquoi docker plutôt qu'un autre :

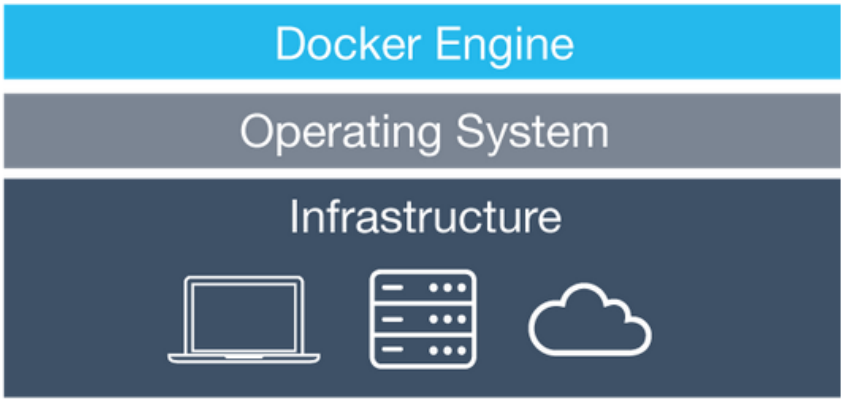
Linux

- Docker compose
- Docker registry

Mac OS/X – Windows Toolbox

- Docker client
- Docker machine
- Docker compose
- Docker kitematic

- Docker swarm



« C'est son écosystème qui fait la force de Docker »

Comment ça marche

Example

- **Dockerfile**

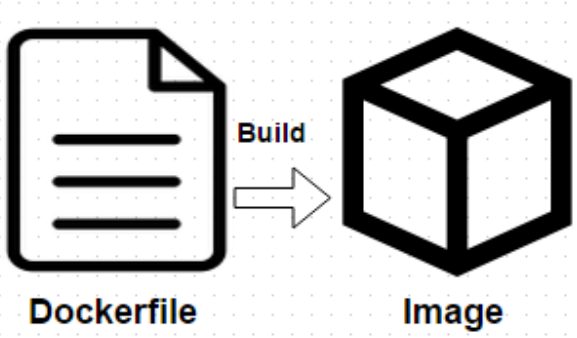
```
FROM ubuntu

RUN apt-get update
RUN apt-get install -y git-core

RUN apt-get install -y python-pip
RUN pip install bottle

RUN git clone https://github.com/waitingkuo/bottle-sample.git

CMD python bottle-sample/app.py
```



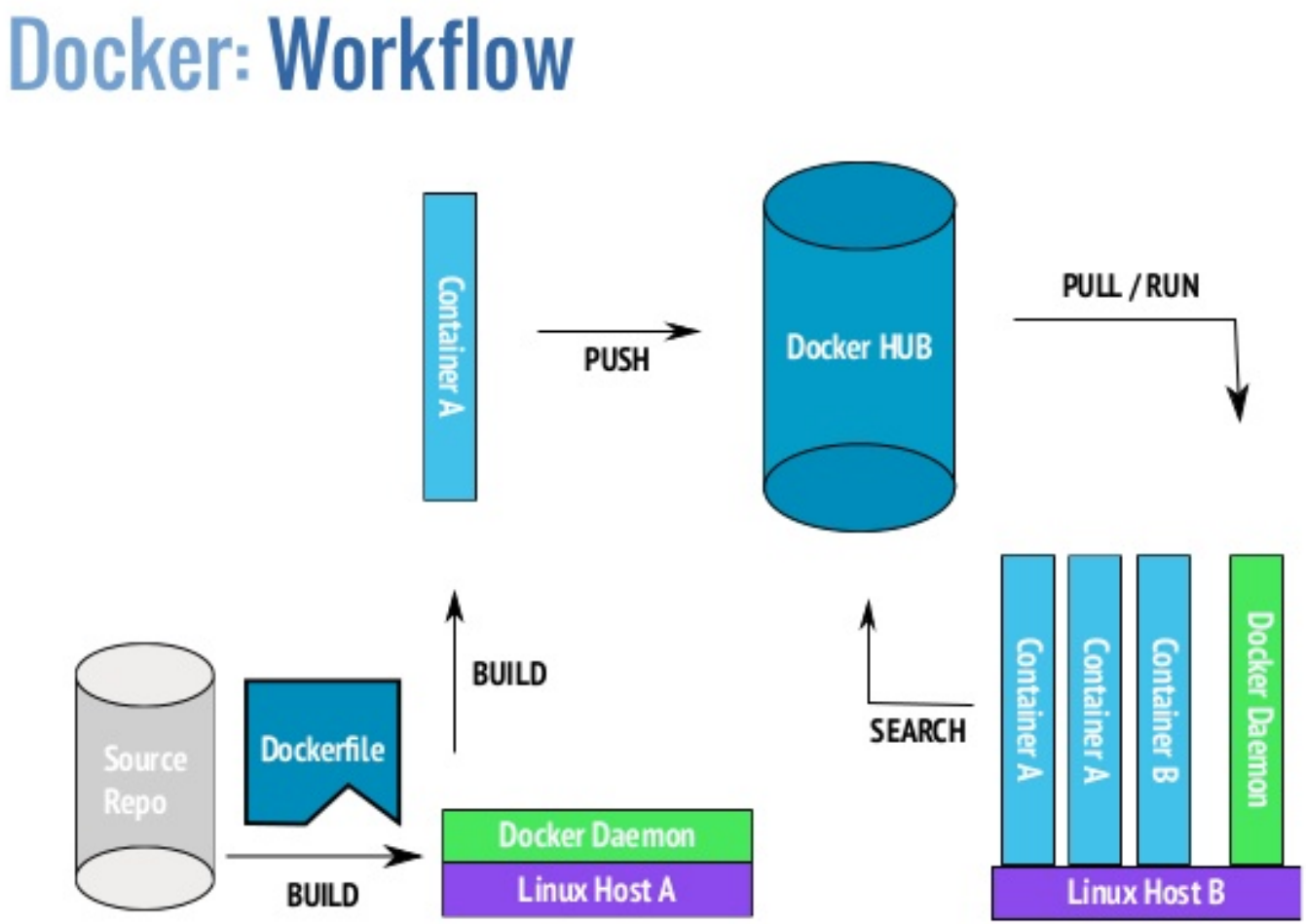
« Une recette de cuisine »

Démo…

Les Usages

- Selon la phase et les tâches dévolues aux acteurs, il peut y avoir un bénéfice
  - En phase de dev
    - facilité la construction d'un environnement iso Prod, test de produits, multi projets, …
  - En phase intégration (Q&A)
    - facilité de livraison et d'adaptation via le paramétrage
    - ré-initialisation d'un environnement (re-jeu de tests)
  - En phase de production
    - le livrable est celui qui a été testé en phase précédente
    - facilité de retour arrière
    - facilité de duplication du service (scalabilité)

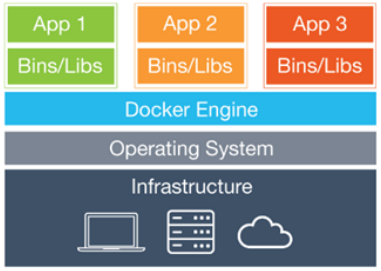
Exemple de Workflow



Synthèse

« **Make Jar not War** »
Scope fonctionnel limité et maîtrisé
Alignement des API

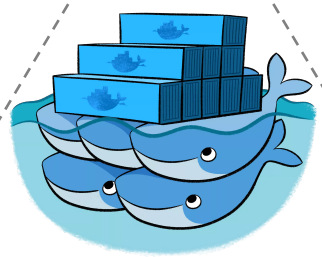
Micro-services



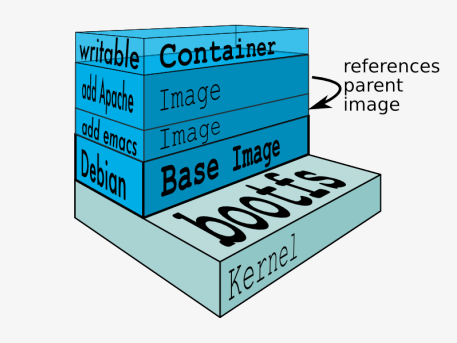
Containers

Chaque app est un Process (java –jar …)

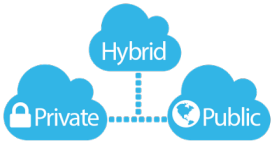
Microservices/Cloud/Devops
- Profiter du Cloud
- Les concepts de microservices existent depuis longtemps mais c'est l'arrivée du Cloud qui lui a donné tout son intérêt.
- Devops sans microservices, c'est une gageure



Container Léger



Run everywhere



What you run is what we test !

Cloud

Provisionning agile
« **Scalabilité** »

Devops

Continuous delivery
Automatisme

iut BAYONNE PAYS BASQUE

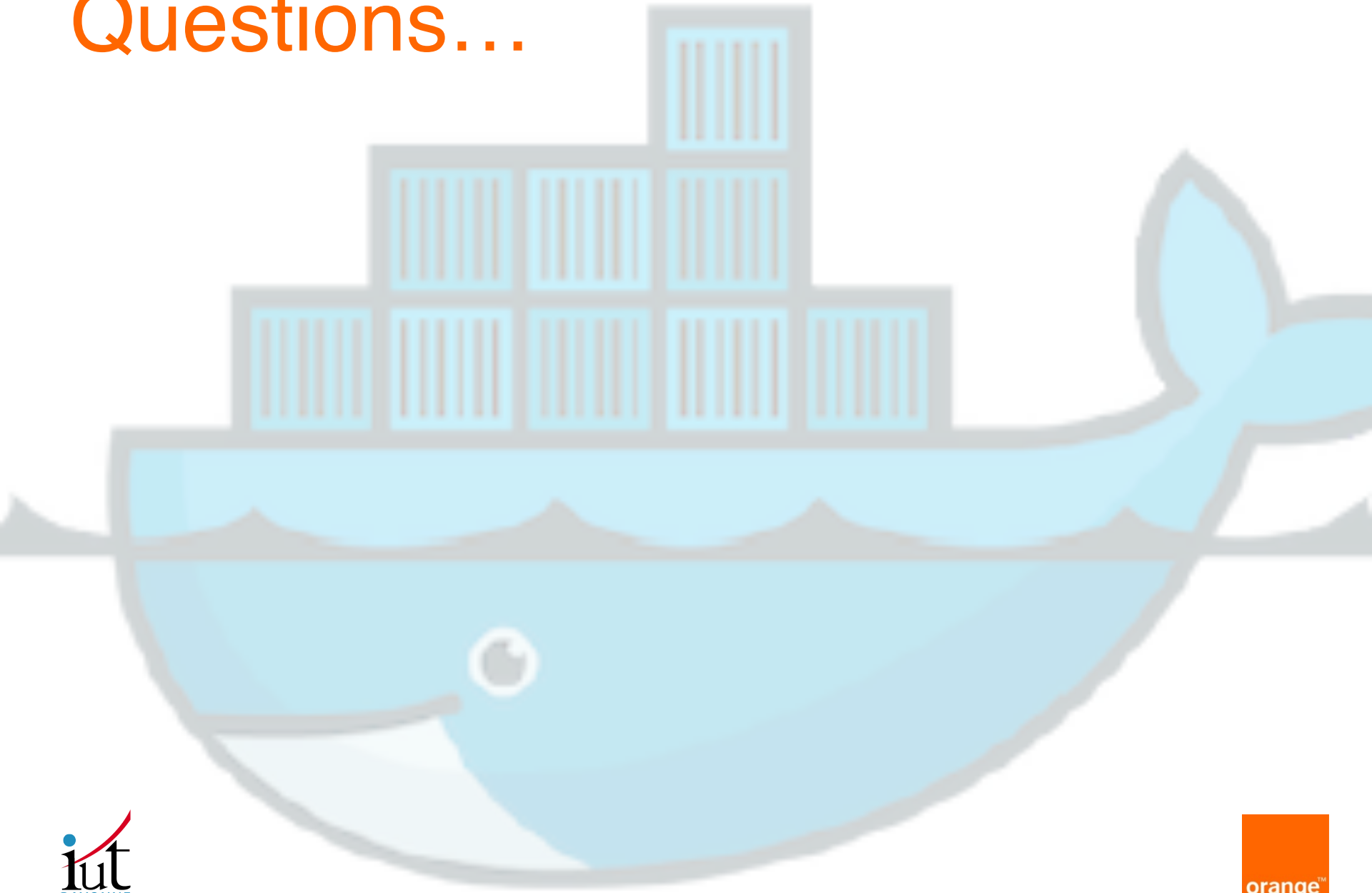
interne Orange

orange™

Conclusion :

- le docker-file comme base pour commencer à travailler sur le mode dev-ops
- Adéquation avec les micros-services
- Apparition de nouvelles problématiques :
  - Découverte dynamique de services
  - Nécessité d'avoir des micros noyaux

Questions…



iut BAYONNE PAYS BASQUE

orange™

Merci à tous

iut BAYONNE PAYS BASQUE

orange™