

# Utilisation de Docker dans le cadre d'un Hébergement Web associatif

Jean-Paul GARRIGOS

le-pic.org

28/03/2017



- 1 Pourquoi ?
  - CMS exotiques
  - Accès SSH
  - Différentes technologies
- 2 Docker Quésako ?
  - Positionnement
  - Principe de fonctionnement
  - Le fichier dockerfile : création d'une image Docker
  - Les dépôts d'images : le hub Docker
- 3 Automatisation de conteneurs Docker
  - Pourquoi plusieurs conteneurs ?
  - Docker Compose
  - Le fichier docker-compose.yml
- 4 Exemple d'hébergement
  - Un site Wordpress et phpmySQL avec Docker
- 5 Perspectives
  - Héberger plusieurs sites web



# Sommaire

- 1 Pourquoi ?
  - CMS exotiques
  - Accès SSH
  - Différentes technologies
- 2 Docker Quésako ?
- 3 Automatisation de conteneurs Docker
- 4 Exemple d'hébergement
- 5 Perspectives



# Pourquoi ?

## CMS exotiques

Certaines associations ont demandé si le PIC est en mesure d'héberger certains CMS que nous ne connaissons pas. Installer ces CMS n'est pas chose aisée, en assurer la maintenance encore moins. Quelques exemples :

- Wordpress : prisé des web designer ;
- joola ! : bon à tout faire ;
- drupal : idem mais davantage communautaire ;
- etc ...



# Pourquoi ?

## Accès SSH

Certaines associations ont demandé si le PIC est en mesure de mettre à disposition un accès SSH sur le serveur. Que permet notamment de réaliser un accès SSH pour une association :

- Automatiser des tâches ;
- lancer des scripts ;
- personnaliser son espace LAMP ;
- utiliser un dépôt GIT distant pour versionner son site web ;
- etc ...



# Pourquoi ?

## Différentes technologies d'hébergements

Certains outils nécessitent une base logicielle différente de celle basé sur PHP, mysql et Apache :

- un autre serveur Web (NGINX par exemple) ;
- d'autres bases de données (postgreSQL, mariaDB, mongoDB, etc.) ;
- d'autres moteurs (node.js, python entre autres) ;
- certaines application web "spéciales" (Cagette par exemple) ;
- etc ...



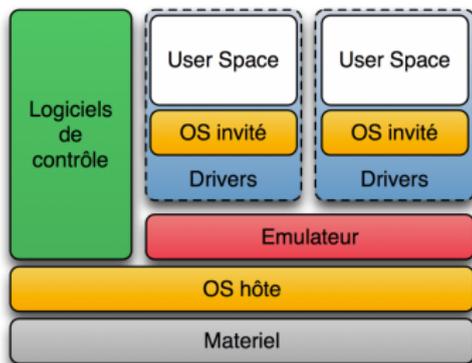
# Sommaire

- 1 Pourquoi ?
- 2 **Docker Quésako ?**
  - Positionnement
  - Principe de fonctionnement
  - Le fichier dockerfile : création d'une image Docker
  - Les dépôts d'images : le hub Docker
- 3 Automatisation de conteneurs Docker
- 4 Exemple d'hébergement
- 5 Perspectives



# Docker Quésako ? I

## Positionnement



## Virtualisation :

- Adapter les ressources (mémoire, CPU, etc.) au besoin de l'application ;
- gestion simple des VM (sauvegardes, copies, etc.) ;
- économie sur le matériel hôte par optimisation des ressources.

Exemple : Les VPS de Gandi que nous utilisons sous l'hyperviseur Xen.



# Docker Quésako ? II

## Positionnement

Il existe plusieurs technologies de virtualisation (hyperviseur, émulateur, etc.). Inconvénients de la virtualisation :

- Il est nécessaire de passer par une couche d'abstraction plus ou moins lourde (très lourde en mode émulation) ;
- chaque machine virtuelle dispose de son système d'exploitation complet ;
- pas de mutualisation des bibliothèques utilisées par plusieurs applications.



# Docker Quésako ? III

## Positionnement

Docker est un projet créé par Solomon Hykes (ingénieur Epitech) et de sa société DotCloud dont l'objectif est de créer des solutions de PaaS (Platform as a Service). L'adoption rapide de l'industrie (motivée par le besoin de flexibilité) incite DotCloud à rendre la plateforme open source et à Solomon Hykes de créer Docker inc. aux Etats Unis.

### A retenir

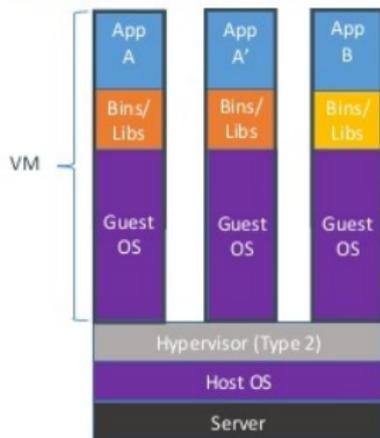
- Docker est le projet d'une plateforme servant à l'automatisation du déploiement d'application sous forme de conteneurs.
- Docker permet de faire abstraction de l'OS tant que les ressources nécessaires sont disponibles.



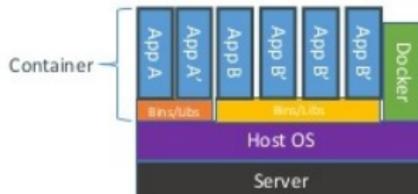
# Docker Quésako ? IV

## Positionnement

### Containers vs. VMs



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



# Docker Quésako ? V

## Positionnement

Les conteneurs légers :

- Conteneur "sur mesure" n'embarquant que l'essentiel nécessaire à l'application (application et dépendances) ;
- une seul système d'exploitation (l'hôte), ce qui demande moins de ressources ;
- il est possible de faire tourner plusieurs instances de la même application (différentes versions à tester) ;
- simplicité de gestion en terme de sécurité et de droits car ils ne s'appliquent que sur la machine hôte ;
- conteneurs de tailles plus réduites qu'une VM (facilite la portabilité);
- temps d'exécution essentiellement identique à celui de l'application.



# Docker Quésako ? VI

## Positionnement

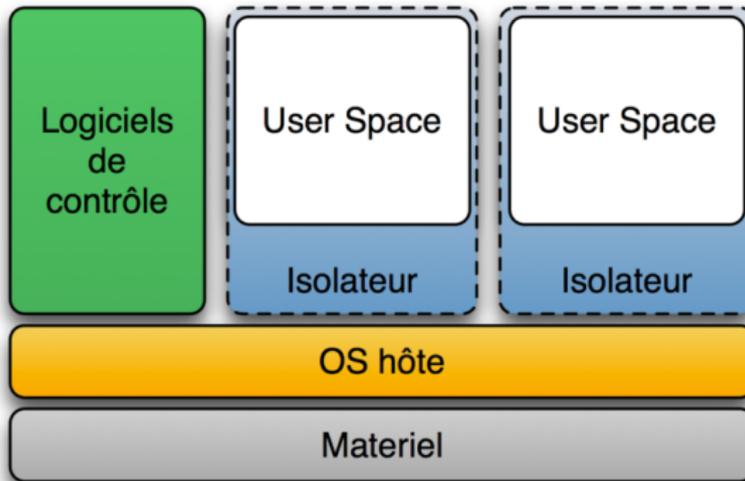
### Docker dans un VPS ?

- Avantages des VM (allocation de ressources, redimensionnement des disques, snapshot, etc.) ;
- Avantages des conteneurs (souplesse d'utilisation, isolation fonctionnelle, portabilité, etc.).



# Docker Quésako ?

## Principe de fonctionnement - Linux Containers

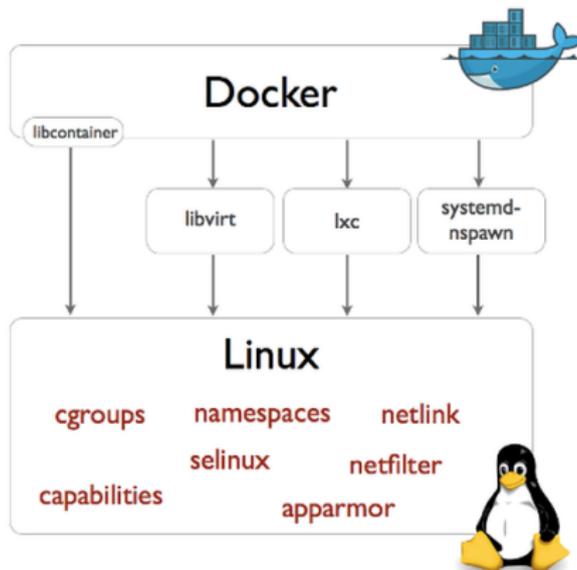


LXC - Linux Containers



# Docker Quésako ?

## Principe de fonctionnement - LibContainer



Libcontainer bibliothèque écrite en Go



# Docker Quésako ?

## Le fichier dockerfile

```
FROM eboras/debian:stable
MAINTAINER Ed Boraas <ed@boras.ca>

RUN apt-get update && apt-get -y install apache2 &&
apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

RUN /usr/sbin/a2ensite default-ssl
RUN /usr/sbin/a2enmod ssl
EXPOSE 80
EXPOSE 443
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```



# Docker Quésako ?

## Les dépôts d'images

Les dépôts d'images peuvent être publiques ou privés.

Le hub Docker est un dépôt public. [▶ Visiter le hub Docker](#)

- la Registry contient des images de conteneurs (qui seront téléchargés) ;
- l'index public : les différentes images sont classées et notées (pertinence, nombre de téléchargements, etc.) ;
- la création d'un compte sur le hub permet d'y stocker ses propres images ;
- certaines entreprise (Canonical, Red Hat, etc.) dispose d'un compte officiel ;
- build automatique d'une application partir d'un dépôt Git (GitHub, GitLab ou BitBucket) et création de l'image Docker de déploiement.



# Sommaire

- 1 Pourquoi ?
- 2 Docker Quésako ?
- 3 Automatisation de conteneurs Docker**
  - Pourquoi plusieurs conteneurs ?
  - Docker Compose
  - Le fichier docker-compose.yml
- 4 Exemple d'hébergement
- 5 Perspectives



# Automatisation de conteneurs Docker

## Pourquoi plusieurs conteneurs ?

- l'Esprit de Docker est basé sur le principe des micro services (une image pour le frontal, une autre pour la base de données, etc.) ;
- permet de changer facilement de technologie (remplacer PHP-FPM par HHVM par exemple) ;
- permet d'utiliser directement les images du hub docker sans devoir les personnaliser ;
- les images vont devoir se lier entre elles pour pouvoir communiquer : la solution Docker Compose !



# Automatisation de conteneurs Docker

## Principes de Docker Compose

Docker Compose est un outil dont l'objectif est de définir et lancer des applications Docker Multi Conteneurs. Cet utilitaire se base sur un fichier de description des services fournis par l'application à mettre en place.

L'outil respecte trois étapes :

- 1 Définition de l'environnement applicatif à l'aide d'un fichier `dockerfile` ;
- 2 définition des services que propose l'application dans le `docker-compose.yml` afin qu'ils puissent fonctionner ensemble dans un environnement isolé ;
- 3 finalement, exécuter `docker-compose up` afin que Compose démarre et exécute l'ensemble de l'application complète.



# Automatisation de conteneurs Docker

## Fonctionnalités de Docker Compose

Quelques fonctionnalités de Compose les plus pertinentes :

- Plusieurs environnements isolés sur un seul hôte ;
- conserver les données de volume lorsque des conteneurs sont créés ;
- recréer uniquement les conteneurs qui ont changé ;
- utilisation de variables et déplacement d'une composition entre environnements.



# Automatisation de conteneurs Docker

## Le fichier docker-compose.yml I

```
site:
  build: site
  ports :
    - "80:80"
  volumes:
    - /home/www/comet:/var/www/html/
  links:
    - database
phpmyadmin:
  image: corbinu/docker-phpmyadmin
  ports :
    - "8080:80"
  environment:
    - MYSQL_USERNAME=root
    - MYSQL_PASSWORD=<password>
```



# Automatisation de conteneurs Docker

## Le fichier docker-compose.yml II

```
links:
  - database:mysql
database:
image: mysql:5.5.43
ports:
  - "3306:3306"
environment:
  - MYSQL_ROOT_PASSWORD=<password>
  - MYSQL_DATABASE=orsicomezqcomet
  - MYSQL_USER=root
  - MYSQL_PASSWORD=<password>
```



# Sommaire

- 1 Pourquoi ?
- 2 Docker Quésako ?
- 3 Automatisation de conteneurs Docker
- 4 Exemple d'hébergement**
  - Un site Wordpress et phpmysql avec Docker
- 5 Perspectives



# Exemple d'hébergement

## Un site Wordpress et phpmysql avec Docker

▶ Visiter mon site wordpress propulsé par Docker

- Un conteneur frontal (Apache et PHP) ;
- un conteneur pour stocker les données (mysql) ;
- un conteneur pour gérer la base de données (phpmyadmin).



# Sommaire

- 1 Pourquoi ?
- 2 Docker Quésako ?
- 3 Automatisation de conteneurs Docker
- 4 Exemple d'hébergement
- 5 Perspectives
  - Héberger plusieurs sites web



# Perspectives

## Héberger plusieurs sites web

Un problème évident : chaque conteneur expose son propre socket !

Une association serait donc en `http://machin.le-pic.org` et la suivante en `http://bidule.le-pic.org:81 ...`

La solution : un Reverse Proxy comme NGINX qui sera chargé de rediriger les flux http vers le bon conteneur.

