

Docker

Pour éviter le célèbre

“Ca marche chez moi pourtant !!”



Qui sommes nous ?

Nicolas Lescure : @nlescure

Expert technique et chef de projet chez Tata Consultancy Services depuis 2016;

Travaille dans les technologies web (dont Php) depuis 2006;

“Docker-Fan” depuis 2 ans

Patrick Joubert : @patjoub

Développeur web freelance depuis 2004

Utilisateur quotidien de Docker depuis 2 ans

Docker : Pourquoi ?

- Des environnements de production hétérogènes
 - OS différents : debian, centos
 - Versions de php allant de 5.2 à 5.5, 5.6, 7.*
 - Mysql (5.0 → 5.5) / MariaDB (5.5 / 10)

Bref.... Une vraie cacophonie de configurations !

Docker : Pourquoi ?

Des applications utilisant des socles différents :

Drupal (drush)

Symfony (console symfony)

Cakephp (console cakephp)

Reprise de vieux sites hébergés sur des mutualisés différents

Docker : Pourquoi ?

Pour éviter le classique

“Ca fonctionne chez moi et pas sur le serveur”

Pour mieux maîtriser mon application (devops)

L'application nécessite un minimum de prérequis techniques.

La configuration ad-hoc permise par Docker permet aussi de mieux comprendre l'interaction entre l'OS et l'application.

Et vice versa

Docker : Pourquoi ?



Docker : Pourquoi ?

Pour installer plus facilement (rapidement ?) un nouveau socle / nouvelle version

Pour imaginer et mettre en place de nouvelles architectures difficiles à concrétiser avec lamp, mamp, wamp

Ouverture sur d'autres technologies (Mongo, NodeJs, TrucXPress, Anguract)

Et Vagrant alors ?

En complément de Docker.

On peut aussi intégrer Docker dans Vagrant.

(certains mêmes préconisent cette configuration pour rendre le développement complètement indépendant de la configuration utilisateur)

The logo features the text "Vagrant + Docker" in a bold, blue, sans-serif font. The "+" symbol is a simple plus sign. The background of the logo is a light blue gradient.

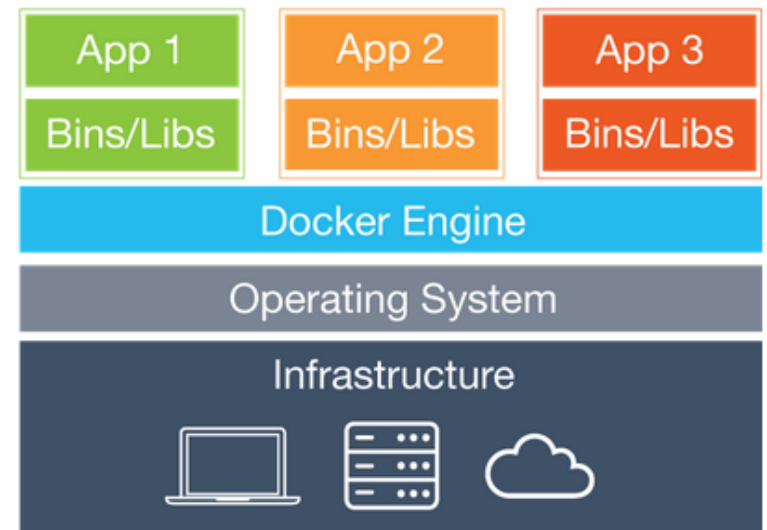
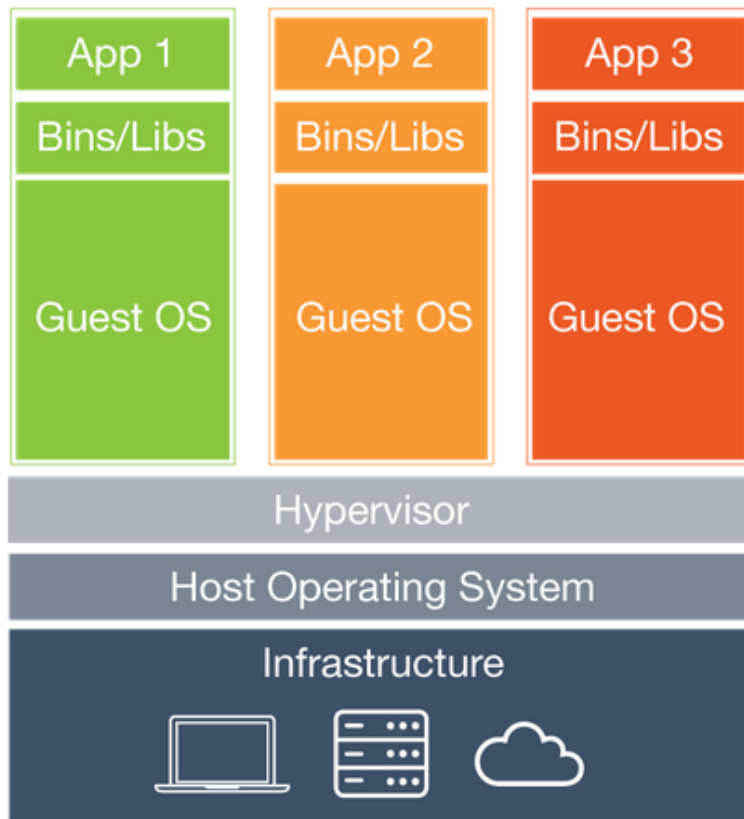
Vagrant + Docker

building portable environments

David Giordano
@3dgiordano

Clip slide

Docker : Comment ça marche ?



Docker : Comment ca marche ?

Des **images** => briques logicielles (php, mysql, mongo, machin) / ro

Des **containers** => instance d'une image / rw

Docker : Comment ca marche ?

Comment se procurer des images ?








<https://hub.docker.com> : répertoire d'images à la fois officielles et personnelles

`docker pull nom-image`

Il faut être vigilant quant aux images “non-officielles”. Elles peuvent parfois faire référence à d’autres images contenant du code dont vous ne connaissez pas la provenance. Il vaut donc mieux se construire soi même son dockerfile ;)

Docker : Comment ça marche ?

Extrait de dockerHub

Explore Official Repositories			
 busybox official	479 STARS	62.7 M PULLS	> DETAILS
 ubuntu official	3.1 K STARS	39.5 M PULLS	> DETAILS
 swarm official	144 STARS	23.4 M PULLS	> DETAILS
 nginx official	2.1 K STARS	20.9 M PULLS	> DETAILS
 registry official	573 STARS	18.4 M PULLS	> DETAILS
 redis official	1.6 K STARS	17.6 M PULLS	> DETAILS
 mysql official	1.7 K STARS	12.0 M PULLS	> DETAILS
 mongo official	1.3 K STARS	7.5 M PULLS	> DETAILS
 node official	1.6 K STARS	6.6 M PULLS	> DETAILS
 debian official	1.1 K STARS	6.5 M PULLS	> DETAILS

Docker : création d'image

Souvent il est nécessaire de créer une image personnalisée.

Bibliothèques php

Outils : drush, composer

D'autres choses (vim, etc...)

Docker : le Dockerfile

Le DockerFile permet cette configuration

Docker : Un Dockerfile

Exemple de DockerFile

```
RUN apt-get install -y curl

RUN mkdir -p /usr/src/drush
WORKDIR /usr/src/drush
RUN curl -OL https://github.com/drush-ops/drush/archive/6.6.0.tar.gz
RUN tar -xvf 6.6.0.tar.gz --strip-components=1
RUN rm 6.6.0.tar.gz
RUN chmod u+x ./drush
RUN ln -s /usr/src/drush/drush /usr/bin/drush

#### VIM
RUN apt-get -y install vim

RUN php -r "readfile('https://getcomposer.org/installer');" | php -- --install-c
    && chmod +x /usr/local/bin/composer

RUN docker-php-ext-install curl ftp gettext hash iconv json mcrypt mbstring mysq
    && docker-php-ext-install session simplexml tokenizer xml xmlrpc zip \
    && docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-
    && docker-php-ext-install gd

###OPCACHE
RUN docker-php-ext-install opcache

RUN apt-get install -y rsyslog
```

Docker : Un Dockerfile

Autre exemple de DockerFile

```
RUN apt-get -y install git

RUN apt-get install -y libicu-dev
RUN pecl install intl
RUN docker-php-ext-install intl

RUN php -r "readfile('https://getcomposer.org/installer');" | php -- --install-dir=/usr/local/bin --filename=composer && chmod +x /usr/local/bin/composer

RUN docker-php-ext-install curl ftp gettext hash iconv json mcrypt mbstring mysqli pdo pdo_mysql pdo_sqlite \
    && docker-php-ext-install session simplexml tokenizer xml xmlrpc zip \
    && docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-dir=/usr/include/ \
    && docker-php-ext-install gd

#RUN docker-php-ext-install opcache

#### Symfony
RUN curl -Ls https://symfony.com/installer -o /usr/local/bin/symfony
RUN chmod a+x /usr/local/bin/symfony

RUN a2enmod rewrite
RUN service apache2 restart
```


Docker : le Dockerfile

Pour créer une image, une seule commande :

```
Docker build -t "nomimage" .
```

Docker : création d'une image

On peut créer une image à partir d'un container existant

`docker commit "idcontainer" "nomimage"`

Docker : gestion des images

C'est véritablement là que se situe la puissance de Docker en permettant une configuration ad-hoc.

Docker : les images #résumé

3 façons de se procurer une image pour une application

#1 : Téléchargement d'une image de DockerHub

#2 : Création à partir d'un Dockerfile



#2 : Commit d'un container

Docker : Comment ca marche ?

Un container :

- * C'est une instance d'une image

- * Est exécuté :

`docker exec/run "options" nom-image`

`docker exec -it php:7-apache /bin/bash`

`docker run -d -p 7080:80 php:7-apache`

Docker : Comment ca marche ?

Un container :

* A des interactions avec le système hôte :

- un ou plusieurs ports :

```
docker run -p 7080:80
```

- un ou plusieurs répertoires

```
docker run -v $PWD:/var/www/html
```

- avec d'autres containers

```
docker --links nom-container:mysql
```

Docker : Comment ca marche ?

Et pour de vrai....

Docker : En résumé #1

Un container est exécuté à partir d'une image.

- * directement (`docker exec`)

- * en mode 'daemon' (`docker run -d`)

Il est en interaction avec le système hôte via :

- des ports
- des répertoires
- d'autres containers

Docker : Les données persistantes

Quand un container est arrêté, les données disparaissent.

Comment assurer la persistance ?

- 1 - Associer un répertoire physique via l'option “-v”
- 2 - Utiliser un “data volume” : Container ne contenant que des données

Docker : Les données persistantes

Les données contenues dans les “data volumes” restent accessibles.

Concrètement cela complique un peu l'architecture mais est vraiment nécessaire pour une plus grande souplesse.

ie : données identiques visibles par un container mysql5-6, mysql5-5, etc...

Docker : au quotidien

Une architecture minimale web est composée :

- php/apache
- mysql (mariadb)
- phpmyadmin (ou pas)

Pour lancer ces 3 containers il faudrait alors exécuter les 3 commandes suivantes

Docker : au quotidien

```
(mysql) docker -run -d -p 3306:3306 --name "db" -v  
$PWD/sql-data:/var/lib/mysql -e  
MYSQL_ROOT_PASSWORD=supersecret -e MYSQL_USER=c_est -e  
MYSQL_PASSWORD=_moi mysql:5.6
```

```
(php) docker -run -d -p 8080:80 --name "php" -v  
$PWD:/var/www/html --links db php:7.1.4-apache
```

```
(phpmyadmin) docker -run -p 8180:80 --name "phpmyadmin"  
--links db:mysql php:7.1.4-apache
```

Inenvisageable tous les jours plusieurs fois par jour.

Docker : au quotidien

Heureusement il y a

(roulements de tambour.... cymbale)

Docker-compose !

Docker : docker-compose

Docker-compose est un utilitaire permettant de lancer automatiquement des containers dont le comportement est décrit dans un fichier de configuration:

docker-compose.yml

- Un exemple (avec ssmtp configurable à la volée)

```
site:
  #image : assurback-5-6
  image: assurback_ssmtp
  ports :
    - "9670:80"
  volumes:
    - ./www:/var/www/html/
    - /etc/hosts:/etc/hosts
    - ./build_image/config/apache/apache-config.conf:/etc/apache2/sites-available/000-default.conf
    - ./build_image/config/apache/apache-config.conf:/etc/apache2/sites-enabled/000-default.conf
    - ./build_image/config/ssmtp.conf:/etc/ssmtp/ssmtp.conf:ro
    - ./build_image/config/revaliasess:/etc/ssmtp/revaliasess
    - ./build_image/config/php/conf.d/php-mail.conf:/usr/local/etc/php/conf.d/mail.ini:ro

external_links:
  - dockmysql:dbhost
```

- un autre exemple (configuration spéciale pour Drush)

```
site7:
  image : cellf_php7
  ports :
    - "9487:80"
  volumes:
    - ./www:/var/www/html/
    - /etc/hosts:/etc/hosts
    - ./build_image/config/apache/apache-config.conf:/etc/apache2/sites-available
    - ./build_image/config/apache/apache-config.conf:/etc/apache2/sites-enabled
    - ./www/Console_Table-1.3.0:/usr/src/drush/lib/Console_Table-1.1.3
  external_links:
    - dockmysql:dbhost
```


- avec limitation de mémoire et lien vers un container maria exécuté en permanence

```
site:
  image : actiparc-7-1-4
  ports :
    - "10080:80"
  volumes:
    - /home/patjoub/Sites/actiparc/www:/var/www/html/
    - /etc/hosts:/etc/hosts
    - ./config/apache/apache-config.conf:/etc/apache2/sites-available/000-de
    - ./config/apache/apache-config.conf:/etc/apache2/sites-enabled/000-de
    - ./config/opcache.ini:/usr/local/etc/php/conf.d/999-opcache.ini

#links:
# - database_pep86:dbhost
external_links:
  - dockmaria:dbhost
mem_limit: 1G
```

Docker : docker-compose

Un exemple de 'docker-compose.yml'

```
site:
  build: build/php

  ports :
    - "37080:80"
    - "37022:22"
  volumes:
    - ./www:/var/www/html/
  links:
    - database

phpmyadmin:
  image: corbinu/docker-phpmyadmin
  ports :
    - "37280:80"
  environment:
    - MYSQL_USERNAME=root
    - MYSQL_PASSWORD=password
  links:
    - database:mysql

database:
  image: mysql:5.6
  ports:
    - "3306:3306"
  environment:
    - MYSQL_ROOT_PASSWORD=p4TJ0u5
    - MYSQL_DATABASE=sed_site
    - MYSQL_USER=admin_sed_site
    - MYSQL_PASSWORD=trop_secret
```

Docker : docker-compose (php)

Explication du fichier :

site:

build: build/php

Construit l'image lors de la première exécution avec un ensemble de configurations personnalisées.

ports :

- "37080:80"

- "37022:22"

Fixe la communication entre le système hôte et le container

volumes:

- ./www:/var/www/html/

Indique le répertoire ./www comme étant le répertoire /var/www/html du container

links:

- database

Lie ce container au container "database"

Docker : docker-compose (phpmyadmin)

phpmyadmin:
image: corbinu/docker-phpmyadmin

Cette fois-ci le container est instanciée à partir d'une image existante

ports :
- "37280:80"

Communication entre le port 37280 de l'hôte et le port 80 du container

environment:
- MYSQL_USERNAME=root
- MYSQL_PASSWORD=password

Définit des variables d'environnement (ici pour mysql)

links:
- database:mysql

Lie le container "database" à la variable d'environnement mysql

Docker : docker-compose (bd)

database:
image: mysql:5.6

Container instancié à partir de l'image officielle de mysql

ports:
- "3306:3306"

La communication entre ports

environment:
- MYSQL_ROOT_PASSWORD=top_secret
- MYSQL_DATABASE=mon_site
- MYSQL_USER=admin_mon_site
- MYSQL_PASSWORD=euh...euh

Les variables d'environnement

Docker : docker-compose

Les commandes

Lancer le tout

docker-compose up -d

Arrêter

docker-compose stop

Effacer les containers

docker-compose rm

Effacer les containers

Et les volumes physiques sur le disque

docker-compose rm -v

Docker : au quotidien

Se connecter à un container (pour faire des trucs)

```
docker exec -it nomcontainer /bin/bash
```

Utilisation de la mémoire

```
docker stats $(docker ps -q)
```

Analyser un container

```
docker inspect nomcontainer
```

Docker : un bilan

Une meilleure compréhension du système (le côté “devops”)

Création d'environnement pour chaque projet. Ce qui permet d'avoir des configurations ajustées.

L'isolation des processus permet d'éviter les conflits de configuration

Comme la machine hôte n'a plus besoin d'avoir les différents composants (php/mysql,etc..)

==> les configurations deviennent “portables” et facilement répliquables sur une autre machine.

Docker : versus Vagrant

Docker nécessite moins de place

=> Toutes les images, containers et volumes : 4,9 G

Vagrant : centos + php 5.5 + mysql 5.5 (bdd : 500 Mo)+
phpmyadmin = 11 Go

License VmWare : 251,95 €

Plugin Vagrant-VmWare : 79 \$

Temps de lancement

Vagrant : environ 1 minute

Docker : quelques secondes

Docker : un bilan

Des architectures peuvent être plus facilement mises en place (php / mysql / mongo / node js) suivant les projets

==> Plus de limitation à un seul écosystème (lamp)

On peut maintenant déposer les images et container sur un hébergeur et ainsi reproduire fidèlement l'environnement technique et l'application (Aws, Microsoft Azure, Digital Ocean, etc.... mais aussi un hébergeur de poitiers ;))

La compréhension de Docker fut très enrichissante et me fait gagner maintenant pas mal de temps

Et je peux désormais tester des technos très rapidement sans me soucier des problématiques d'installation. Ainsi j'enrichi mon savoir-faire

Docker

<https://www.docker.com/>

<https://hub.docker.com/>

James Turnbull : The Docker Book → <http://dockerbook.com/>

Docker for php developpers :

<http://www.newmediacampaigns.com/blog/docker-for-php-developers>

Day 21: Docker–The Missing Tutorial

<https://blog.openshift.com/day-21-docker-the-missing-tutorial/>

Series: How to create your own website based on Docker (Part 1 - Introduction)

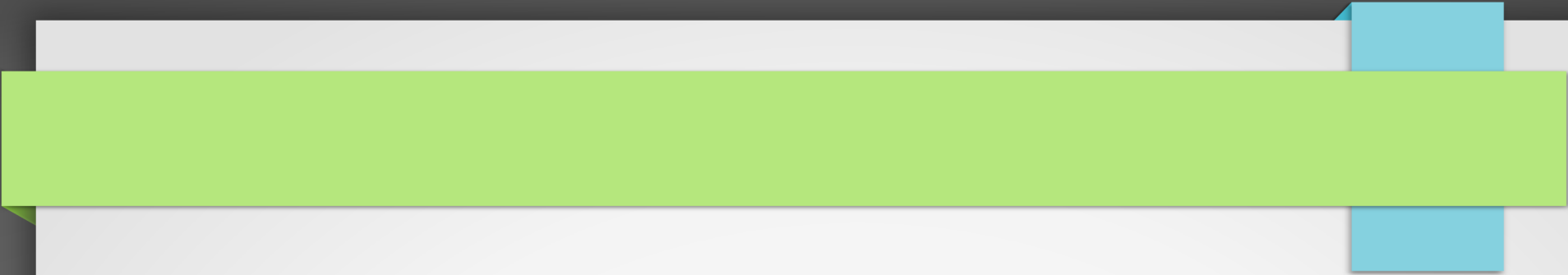
<http://project-webdev.blogspot.de/2015/05/create-site-based-on-docker-part1.html>

Dive into docker : <https://diveintodocker.com/> (environ une trentaine d'heures de vidéos pour apprendre docker.... très très bien fait et surtout très récent ... juin 2017)

Et mettre les mains dans le cambouis



Merci
pour votre
attention



Une petite démo avant
de laisser la place à
Nicolas pour
RANCHER