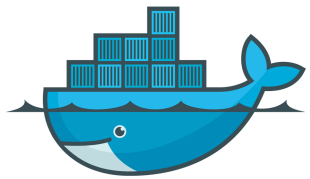
Développer et packager votre application Symfony2 avec Docker et Vagrant

Symfony



VAGRANT

https://joind.in/11240

(@thierrymarianne | thierrym@theodo[point]fr)

Proposition

À partir d'une application Symfony2 ayant fait ses preuves :

- Utilisateurs

- Intégration continue / déploiements automatisés

=> Passer à Docker (sans Puppet / Chef / Ansible ...)

https://github.com/thierrymarianne/symfony2-docker-vagrant

(@thierrymarianne | thierrym@theodo[point]fr)

Un peu d'histoire

Projet interne chez DotCloud, Solomon Hykes

Janvier 2014 : Levée de fond de 15M$

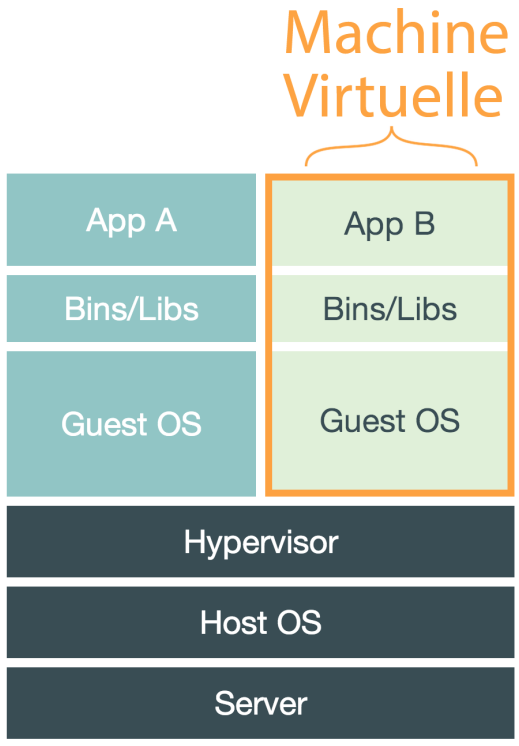
9 Juin 2014 : Sortie de la 1.0 à la DockerCon 2014

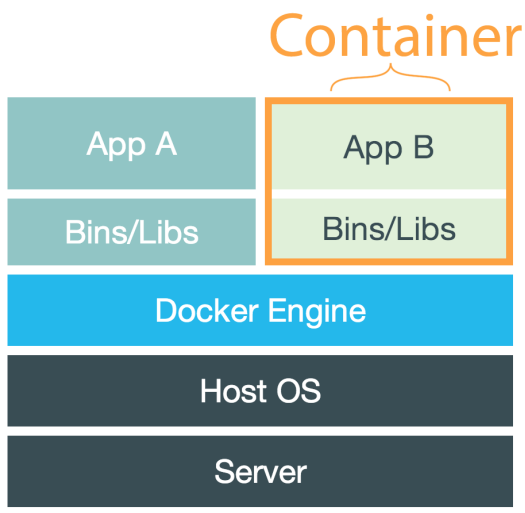
10 Juin 2014 : Intégration de Docker dans Google App Engine

Tu vois, le monde se divise en deux catégories :

les machines virtuelles et les containers

Machines virtuelles !== Containers





Dans quels contextes utiliser Docker ?

Développement

Intégration continue

Déploiement

Pour le développeur

Davantage d'indépendance vis à vis des Ops

Meilleure sensation d'être en production sur son poste de travail

Pour l'opérationnel

Davantage d'indépendance vis à vis des Devs

Faciliter la surcharge des configurations utilisées en développement

(variables d'environnements, fichiers de configuration, persistence)

En pratique

Container Machine ("machine virtuelle" rapide)

Container Application (rôle au sens gestion de configuration)

Container Volume de Données (point de montage)

Intégration avec Vagrant

En mode provider

Rôle tenu de manière traditionnel par VirtualBox / VMware

=> Container Machine

```
01. $ vagrant init
02.
03. [...]
04. config.vm.provider "docker" do |d|
05.   d.image = "afup/nginx"
06. end
07. [...]
```

"Machine virtuelle" légère ?

Utilisation des images de base Phusion

/sbin/my_init (collecte des processus orphelins)

Runit à la place d'upstart (supervision de service)

Syslog-ng / logrotate

ssh

/!\ Non-recommandé par Michael Crosby

Intégration avec Vagrant

- Provisioning

```
01. Vagrant.configure("2") do |config|
02.   config.vm.provision "docker",
03.   images: ["afup/symfony2"]
04. end
```

- Rôle aussi tenu par Shell / Puppet / Chef

=> Container Application

Selon le fondateur de Docker

Solomon Hykes
@solomonstre

Follow

@amaltson @drnic @phusion_nl both are legitimate use cases. The more legacy you need to support, the fatter your container.

Reply Retweet Favorite More

4:55 AM - 19 Feb 2014

Ou alors, Boot2Docker!

Linux très léger optimisé pour lancer des containers rapidement

Installation pour OSX / Windows

Comment utiliser Docker?

Ligne de commande / Dockerfile

https://docs.docker.com/installation/

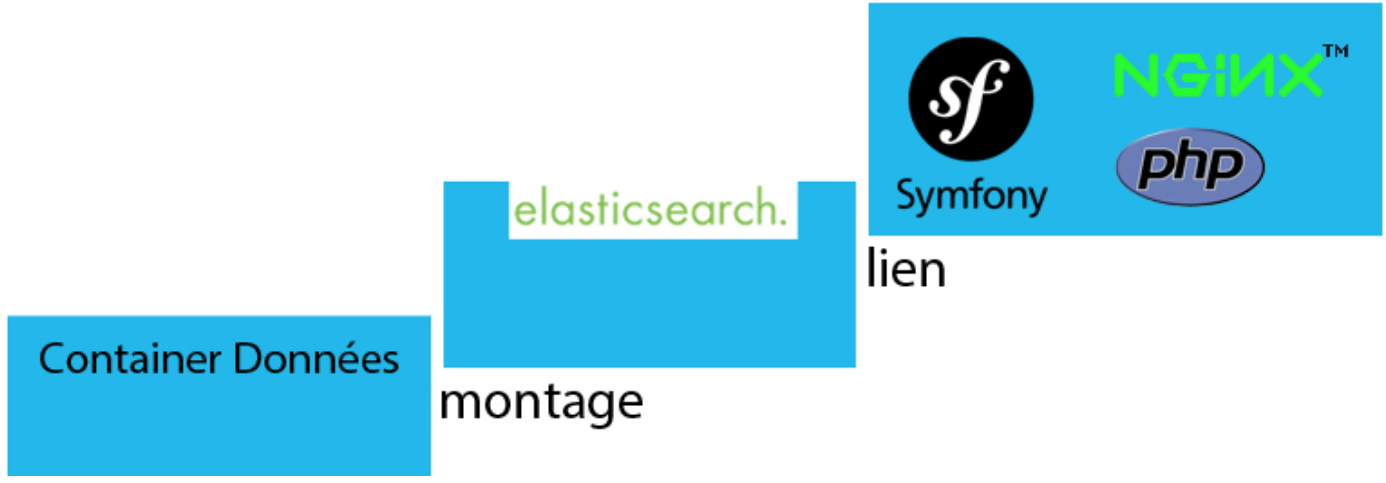
```
$ docker -h
```

Outils de configuration / orchestration

```
# pip install -U fig
```


(@thierrymarianne | thierrym@theodo[point]fr)

Application Symfony2 associée à un index de recherche



Ne pas réinventer la roue

Le contenu de notre Dockerfile pour notre container Elasticsearch :

```
01. $ vi Dockerfile
02.
03. FROM tutum/elasticsearch
04. EXPOSE 9200
05. VOLUME ['/var/lib/elasticsearch']
06. ENTRYPOINT ["/usr/share/elasticsearch/bin/elasticsearch"]
```

Contruire une image Elasticsearch

```
01. docker build -t afup/elasticsearch:0.1 .
02. # => nouvelle image créée pour le référentiel
03. # afup/elasticsearch
```

```
01. # Envoi sur index public
02. docker push afup/elasticsearch:0.1
```

```
01. # Récupération à partir de l'index public
02. docker pull afup/elasticsearch:0.1
```

Démarrer notre container à partir de l'image

```
01. docker run -d -p :9200 \
02. --name elasticsearch-server \
03. -v `pwd`/elasticsearch:/var/lib/elasticsearch \
04. afup/elasticsearch:0.1
```

Tagger notre container Elasticsearch

```
01. docker run -d -p :9200 \
02. --name elasticsearch-server \
03. --volumes-from elasticsearch-data-volume \
04. -v `pwd`/elasticsearch:/var/lib/elasticsearch \
05. afup/elasticsearch:0.1
```

Déclarer un Container Volume de Données

```
01. docker run -d -p :9200 \
02. --name elasticsearch-server \
03. --volumes-from elasticsearch-data-volume \
04. -v `pwd`/elasticsearch:/var/lib/elasticsearch \
05. afup/elasticsearch:0.1
```

Lancer un Container Application (nginx)

Avec montage de notre application Symfony2

```
01. docker run -t -i -p 80:80 \
02. --name php-nginx-server \
03. --link elasticsearch-server:symfony__elasticsearch_ \
04. -v symfony2:/var/www/symfony2 \
05. -v `pwd`/nginx/sites-enabled:/etc/nginx/sites-enabled \
06. # [...]
07. afup/nginx:0.1
```

Lier nos containers Elasticsearch et nginx

```
01. docker run -t -i -p 80:80 \
02. --name php-nginx-server \
03. --link elasticsearch-server:symfony__elasticsearch_ \
04. -v symfony2:/var/www/symfony2 \
05. -v `pwd`/nginx/sites-enabled:/etc/nginx/sites-enabled \
06. afup/nginx:0.1
```

Configuration de notre application

```
01. # Définition de paramètres de configuration en yaml
02. parameters:
03.     elasticsearch_host:     %elasticsearch.host%
04.     elasticsearch_port:     %elasticsearch.port%
05.     gmail_search_index:     gmail
06.     twitter_search_index:   twitter
```

Variables d'environnement (1/2)

Injection d'une variable au démarrage d'un container

```
01. docker run -t -i -p 80:80 \
02. --name php-nginx-server \
03. --e SYMFONY__ELASTICSEARCH__HOST=127.0.0.1
04. -v symfony2:/var/www/symfony2 \
05. -v `pwd`/nginx/sites-enabled:/etc/nginx/sites-enabled \
06. afup/nginx:0.1
```

```
01. root@ad09c8a66671:# dans mon container Elasticsearch
02. root@ad09c8a66671:env
03. root@ad09c8a66671: SYMFONY__ELASTICSEARCH__HOST=127.0.0.1
```

Variables d'environnement (2/2)

```
01. docker run -t -i -p 80:80 \
02. --name php-nginx-server \
03. --link elasticsearch-server:symfony__elasticsearch_ \
04. -v symfony2:/var/www/symfony2 \
05. -v `pwd`/nginx/sites-enabled:/etc/nginx/sites-enabled \
06. afup/nginx:0.1
```

SYMFONY__ELASTICSEARCH__HOST <=> %elasticsearch.host%

Lier deux containers transfère les variables d'environnement

Pro Tip: Abuser des images officielles

```
01. FROM ubuntu:14.04
02. ENV DEBIAN_FRONTEND noninteractive
03.
04. # RUN apt-get upgrade  à bannir
05.
06. RUN apt-get update
07. RUN apt-get install -y --force-yes software-properties-common
```

Pro Tip: Proxy Packages Debian (1/2)

Proxy pour gestionnaire de paquets sur la machine hôte

```
01.
02. sudo apt-get install squid-deb-proxy avahi-utils
03.
```

Peut être même dans son propre container ?

```
01.
02. https://github.com/yasn77/docker-squid-repo-cache.git
03.
```

Pro Tip: Proxy Packages Debian (2/2)

Dans mon Dockerfile, avant l'installation de paquets :

```
01. RUN route -n | awk '/^0.0.0.0/ {print $2}' > /tmp/host_ip.txt
02. RUN echo "HEAD /" | nc `cat /tmp/host_ip.txt` 8000 | \
03. grep squid-deb-proxy \
04.   && (echo "Acquire::http::Proxy \"http://$(cat /tmp/host_ip.txt):8000\
05.   /etc/apt/apt.conf.d/30proxy) \
06.   && (echo "Acquire::http::Proxy::ppa.launchpad.net DIRECT;" >> \
07.   /etc/apt/apt.conf.d/30proxy) \
08.   || echo "No squid-deb-proxy detected on docker host"
```

Pro Tip: Tirer parti du cache de Docker

Dans vos Dockerfiles :

Conserver au maximum l'ordre des instructions!

```
01. RUN apt-get install -y --force-yes nginx
02. RUN apt-get install -y --force-yes php5-fpm
```

Différent de

```
01. RUN apt-get install -y --force-yes php5-fpm
02. RUN apt-get install -y --force-yes nginx
```

Pro Tip: Proxy Toran pour composer

Par l'un des lead développeurs de Composer (Jordi Boggiano)

Accélérer l'installation des vendors

Sert de point de redondance avec github / référentiels privés

=> Container Application

Et si on se débarrassait (en partie) du shell ?

```
01. # fig.yml
02. nginx:
03.   build: /home/afup/php-nginx
04.   links:
05.    - elasticsearch:symfony__elasticsearch_ [...]
06.   ports:
07.    - "8081:80"
08.   expose:
09.    - "80"
10.
```


(@thierrymarianne | thierrym@theodo[point]fr)

Fig

```
01. elasticsearch:
02.   image: afup/elasticsearch:0.1
03.   volumes_from: elasticsearch-data-volume
04.   expose:
05.    - "9200"
06.   ports:
07.    - ":9200"
```

Où se cache notre application Symfony2 cette fois ?

Fig

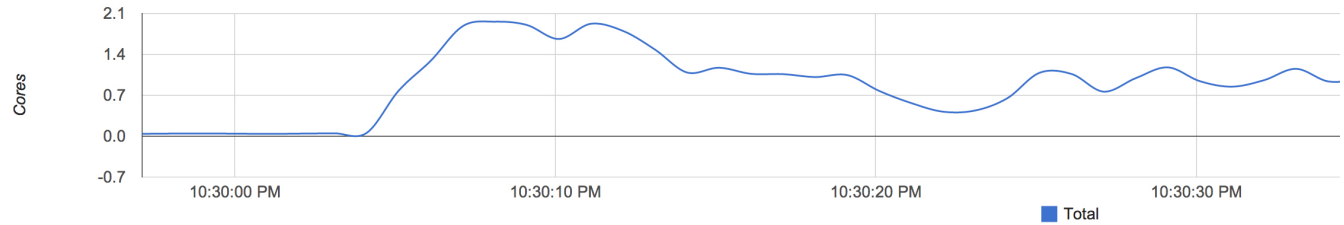
```
01. nginx:
02.   [...]
03.   volumes:
04.     - nginx/sites-enabled:/etc/nginx/sites-enabled
05.     - symfony2:/var/www/symfony2
06.
07. $ fig up
```

La vie des containers avec Cadvisor

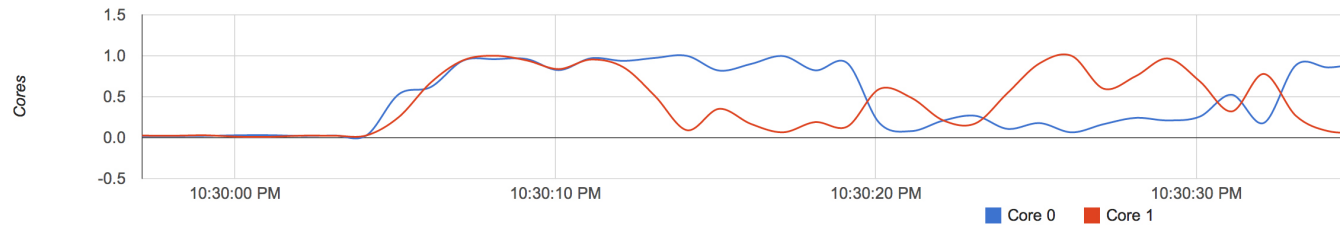
cAdvisor

CPU

Total Usage

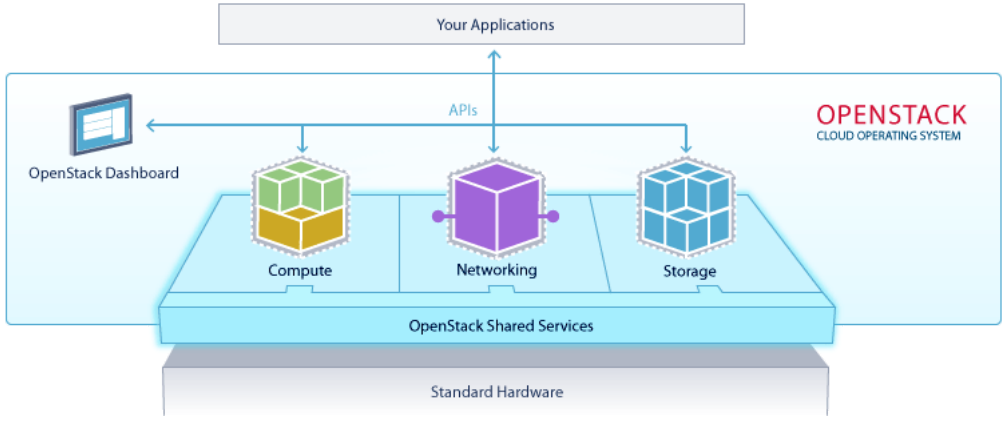


Usage per Core



Aller encore plus loin avec OpenStack

Avec OpenStack (cloud manager)



Démarrer une "Stack" avec Heat

```
01. description: >
02.   Stack Heat (Orchestration)
03. resources:
04.   nginx:
05.     type: DockerInc::Docker::Container
06.     properties:
07.       image: afup/php-nginx:0.1
08.   elasticsearch:
09.     type: DockerInc::Docker::Container
10.     [...]
```

```
heat stack-create nginx -f ../openstack/php-nginx.yml
```

Et ce n'est que le début !

Orchestration avec Maestro-Ng

Configuration visuelle avec Gaudi

Déploiement avec Deis / CoreOS

Support de Docker avec AppEngine

Showtime

- /!\ Effet démo en perspective !!

Questions ?

Sources

- https://linuxcontainers.org
- http://en.wikipedia.org/wiki/Chroot
- http://en.wikipedia.org/wiki/Cgroups
- https://www.docker.io
- https://hub.docker.com/
- https://speakerdeck.com/ubermuda/a-multi-container-symfony2-setup-with-docker
- https://wiki.openstack.org/wiki/Solum
- https://speakerdeck.com/ubermuda/a-multi-container-symfony2-setup-with-docker
- http://blog.docker.com/2014/04/openstack-update-icehouse-release-update/
- https://github.com/phusion/baseimage-docker
- http://crosbymichael.com/dockerfile-best-practices.html
- https://toranproxy.com/
- http://openstack.redhat.com/Quickstart/