Containers: docker

Balbuceos sobre una tecnología de moda.

Giménez Silva Germán Alberto
Programador para EosWeb

Manuel Garcia
Director Altoros Argentina
www.altoros.com

Links - Acerca de la presentación:

Docker | https://www.docker.com/ (https://www.docker.com/)

Vagrant | https://www.vagrantup.com/ (https://www.vagrantup.com/)

```
No se si entra:
```

Docker Machine | https://docs.docker.com/machine/ (https://docs.docker.com/machine/)

```
Puede ser de interés pero NO entra en la presentación:
```

Swarm (https://docs.docker.com/swarm/install-w-machine/)

Giant Swarm | https://giantswarm.io/ (https://giantswarm.io/)

chef | https://www.chef.io/chef/ (https://www.chef.io/chef/)

puppets | https://puppetlabs.com/ (https://puppetlabs.com/)

docker vs puppet vs vagrant vs chef (http://www.javiergarzas.com/2015/08/docker-vs-puppet-vs-vagrant-vs-chef.html)

Licencias

La mayoría de las opciones para elegir en esta tecnología tienen licencias libres. Aunque, como todo corre en un servidor, terminaremos pagando para que nuestros containers corran en un infrestructura que es complicado conocer sus licencias. En general, serán gnu-linux por economía.

Licencias

Apache License | Version 2.0, January 2004 | https://www.apache.org/licenses/ [FREE]

The MIT License [FREE]

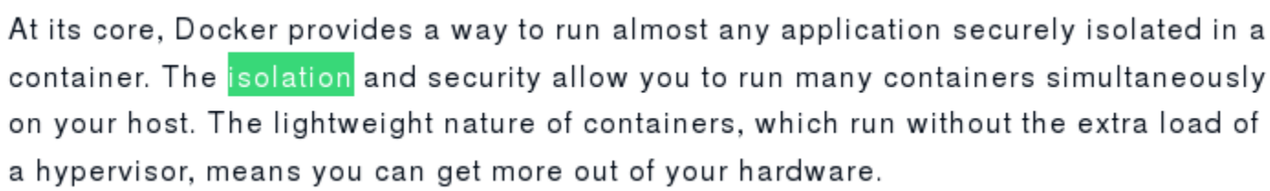
GNU General Public License V2 [FREE]

boot2docker

es un linuxito que sirve para correr containers Docker dentro de plataformas no Linux y también corre en Linux.

http://boot2docker.io/ (http://boot2docker.io/)

Es un desarrollo libre que nos permite correr la infraestructura de containers dentro de plataformas Windows / MacOs / Linux.

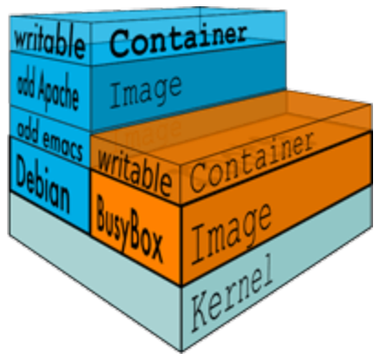


Acerca de la charla

- 1. Principalmente Docker
- 1.1 Algunos conceptos
- 1.2 Creando un proyecto en docker
- 2. Comparativa con vagrant

Docker ¿Qué es docker?

Docker nos permite empaquetar una aplicación con todas sus dependencias en un contenedor.



Los contenedores Docker son un sistema de archivos completo donde podemos colocar nuestras aplicaciones con todas sus librerías. Aqui podemos instalar todo lo que necesitamos en un servidor y esto nos garantiza que estaremos ejecutando el mismo código con las mismas librerías, independientemente del entorno.

Docker - Isolation

Dockerfile best practices (https://docs.docker.com/articles/dockerfile_best-practices/)

"La buena práctica" - existe la idea de que en cada container, por cuestiones de seguridad y menor promiscuidad, debe estar corriendo sólo una proceso y no más que eso. A esto se le llama aislamiento y es algo que es muy valorado dentro de las comunidades de creadores de containers.

NOTA: es algo que esta bueno saberlo, pero a la hora de SHIP (enviar la aplicacion a un amigo que no es experto) no es tan útil, ya que es mas sencillo hacer sólo un container con todo adentro y listo.

Docker - Usando

En las charlas de Docker que he visto, hay una fuerte tendencia a crear una infrastructura, con fines de empresa (seguridad, escalabilidad, etc ...)

Pero como programador también veo muchos beneficios que a veces no son tan parecidos a un deber ser de infraestructura empresarial. Y esto es, que en tiempo de desarrollo, meter toda una aplicación con sus dependencias en un solo container (app, database, etc...), es algo que puede ser útil y hacernos ganar tiempo.

```
En este caso dejaríamos las buenas prácticas para el deploy
y nuestro equipo sólo deberia conocer el comando.
# docker import app.tar.gz
```

Creando un container - Instalación

```
Instalar docker
```

install | https://docs.docker.com/installation/debian/ (https://docs.docker.com/installation/debian/)

Adding a nonroot user to administer Docker

For ease of use, we can allow a nonroot user to administer Docker by adding them to a Docker group.

Getting ready

1. Create the Docker group if it is not there already:

   ```
   $ sudo group add docker
   ```

2. Create the user to whom you want to give permission to administer Docker:

   ```
   $ useradd dockertest
   ```

How to do it...

Run the following command to allow the newly created user to administer Docker:

```
$ sudo  gpasswd -a dockertest docker
```

Creando un container - search

```
Buscar una imagen:
# docker search mongo
```

```
[ggerman@ggermanlp:~]$ docker search mongo
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
mongo                             MongoDB document databases provide high av...   910       [OK]
tutum/mongodb                     MongoDB Docker image – listens in port 2...     47                   [OK]
torusware/speedus-mongo           Always updated official MongoDB docker ima...   8                    [OK]
jacksoncage/mongo                 Instant MongoDB sharded cluster                 6                    [OK]
knickers/mongo-express            Graphically manage your mongoDB container ...   5                    [OK]
mongooseim/mongooseim-docker      MongooseIM server the latest stable version     3                    [OK]
leportlabs/mongo-k8s-sidecar      Kubernetes side car to setup and maintain ...   2                    [OK]
pablosan/mongo                                                                    1                    [OK]
lingya/mongo                      Test build mongo,Tibco CDC                      1                    [OK]
19hz/mongo-container              Mongodb replicaset for coreos                   1                    [OK]
longieirl/mongo                                                                   0                    [OK]
wegenenverkeer/mongo                                                              0                    [OK]
criptext/mongo                    mongo for boot2docker                           0                    [OK]
ogolosovskiy/docker-coturn-mongo  Mongo Modified version coTurn                   0                    [OK]
totem/docker-mongo-es             Docker based wrapper for mongo connector t...   0                    [OK]
davidsblog/node-mongo             A container which runs node.js and mongoDB      0                    [OK]
modli/mongo                       Init Mongo for Modli testing                    0                    [OK]
brownman/mongo                                                                    0                    [OK]
yeasy/mongo-connector             pipeline from a MongoDB cluster to other s...   0                    [OK]
smartprocure/mongo                                                                0                    [OK]
asteris/apache-php-mongo          Apache2.4 + PHP + Mongo + mod_rewrite           0                    [OK]
twistedogic/mongo                 Mongo with bind_ip option                       0                    [OK]
appertly/hhvm-mongo               Docker image of HHVM with compiled Mongofi...   0                    [OK]
akilli/mongo                                                                      0                    [OK]
ulexus/mongo                      DEPRECATED - please use the official mongo...   0                    [OK]
[ggerman@ggermanlp:~]$
```

... Profundizar en esto ...

`Como almacenar registros en docker mongoDB "-v"`

The Docker documentation is a good starting point for understanding the different storage options and variations, and there are multiple blogs and forum postings that discuss and give advice in this area. We will simply show the basic procedure here for the latter option above:

1. Create a data directory on a suitable volume on your host system, e.g. `/my/own/datadir`.
2. Start your `mongo` container like this:

```
$ docker run --name some-mongo -v /my/own/datadir:/data/db -d mongo:tag
```

The `-v /my/own/datadir:/data/db` part of the command mounts the `/my/own/datadir` directory from the underlying host system as `/data/db` inside the container, where MongoDB by default will write its data files.

Note that users on host systems with SELinux enabled may see issues with this. The current workaround is to assign the relevant SELinux policy type to the new data directory so that the container will be allowed to access it:

Creando un container - run / build

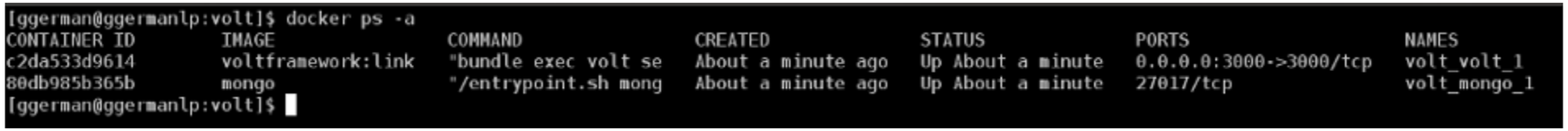
```
En este caso run y build funcionan de la misma manera
```

Cuando corremos (run) o creamos una imagen (build) docker primero busca si no se encuentra en nuestro host local y si no está allí, la descarga de docker hub.

```
Descarga los layers que le faltan para completar nuestra imagen.
```

Creando un container - run / build -> ps -a

```
Conocer los containers que estan corriendo en nuestro host.
# docker ps -a
```




```
Matar todos los containers corriendo en nuestro host
# docker rm $(docker ps -a -q)

Listar todas las imagenes alojadas en nuestro host
# docker images
Borrar una imagen:
# docker rmi -f 865a6d4ec0ed
```

```
[ggerman@ggermanlp:clojure]$ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
presentation            latest              29124ebaf800        6 days ago          731 MB
clojure                 latest              276f186b8d71        7 days ago          833.2 MB
rubylit                 latest              865a6d4ec0ed        2 weeks ago         951.1 MB
ggerman/rubylit         v1                  bfa3a818e7a2        2 weeks ago         674.3 MB
rails.pg.ssh/rubylit    version04           ac959702881a        2 weeks ago         689.6 MB
rails.pg.ssh/rubylit    version03           6f7040f06429        2 weeks ago         689.3 MB
ggerman-rubylit/rails   version02           a16ec4e41331        4 weeks ago         674.3 MB
ggerman-rubylit/rails   latest              f62ae91e45e8        4 weeks ago         674.3 MB
swarm                   latest              207e8b983242        4 weeks ago         10.2 MB
ggerman/games           latest              8785050fb427        5 weeks ago         666.8 MB
postgres                latest              730d1d72bda2        6 weeks ago         265.3 MB
ruby                    2.1                 c0767f27abaf        6 weeks ago         713.6 MB
debian/rail             latest              cccf79f2dcc7        6 weeks ago         581.1 MB
debian/postgresql       latest              150468d28cd9        6 weeks ago         501.4 MB
rails                   onbuild             ebbe866cb7bd        8 weeks ago         757.6 MB
rails                   latest              722c7f393bc4        8 weeks ago         810.8 MB
debian                  stable              e8a26d5f0bf7        8 weeks ago         125.2 MB
debian                  jessie              9a61b6b1315e        8 weeks ago         125.2 MB
tutum/hello-world       latest              c833a1892a15        3 months ago        17.32 MB
ruby                    2.2.0               51473a2975de        7 months ago        774.6 MB
```

Creando un container - hagamos correr volt con mongodb (1)

Dockerfile

```
FROM debian
MAINTAINER Giménez Silva Germán Alberto <ggerman@gmail.com>

RUN apt-get update && apt-get install -y patch ruby ruby-dev gcc g++ make zlib1g-dev vim

RUN gem install mini_portile --no-rdoc --no-ri
RUN gem install rake --no-rdoc --no-ri
RUN gem install patch --no-rdoc --no-ri
RUN gem install bundler pry --no-rdoc --no-ri
RUN gem install volt --no-rdoc --no-ri
RUN gem install nokogiri -v '1.6.6.2' --no-rdoc --no-ri

ENV APP /app
WORKDIR $APP

RUN volt new . && bundle install
RUN rm config/app.rb
COPY app.rb config/app.rb
ENTRYPOINT ["bundle", "exec", "volt", "server"]
```

VOLT | http://voltframework.com/ (http://voltframework.com/)

Creando un container - hagamos correr volt con mongodb (2)

```
# docker build -t voltframework .
```

```
root@ggermanlp:/home/ggerman/docker/volt# docker build -t voltframework:presentation .
Sending build context to Docker daemon 9.728 kB
Sending build context to Docker daemon
Step 0 : FROM debian
 ---> 8c00acfb0175
Step 1 : MAINTAINER Giménez Silva Germán Alberto <ggerman@gmail.com>
 ---> Using cache
 ---> 8524fcdf4613
Step 2 : RUN apt-get update && apt-get install -y patch ruby ruby-dev gcc g++ make zlib1g-dev vim
 ---> Using cache
 ---> 42f75629aab6
Step 3 : RUN gem install mini_portile --no-rdoc --no-ri
 ---> Using cache
 ---> c42cd6a25eca
Step 4 : RUN gem install rake --no-rdoc --no-ri
 ---> Using cache
 ---> 45bae973b263
Step 5 : RUN gem install patch --no-rdoc --no-ri
 ---> Using cache
 ---> 548e59020019
Step 6 : RUN gem install bundler pry --no-rdoc --no-ri
 ---> Using cache
 ---> a59bcb2ddaf7
Step 7 : RUN gem install volt --no-rdoc --no-ri
 ---> Using cache
 ---> feb802286c8d
Step 8 : RUN gem install nokogiri -v '1.6.6.2' --no-rdoc --no-ri
 ---> Using cache
 ---> d0cdd71612f7
Step 9 : ENV APP /app
 ---> Using cache
 ---> 3f1a7d506f6c
```

Containers: docker

Creando un container - RUN (1)

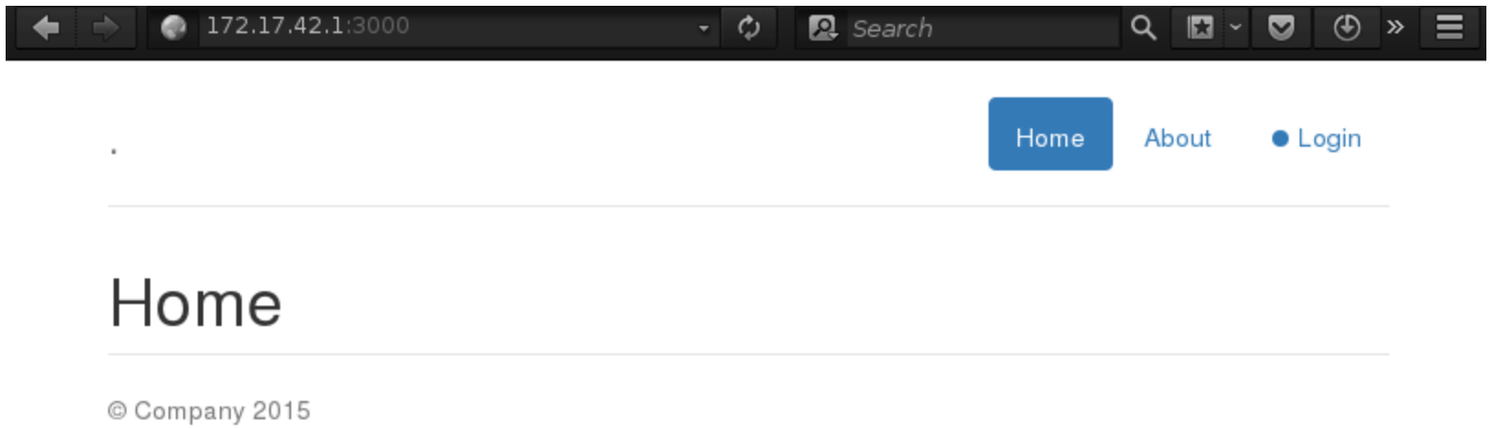
```
# docker run -p 3000:3000 --name volt voltframework
```

Aqui estamos corriendo una instancia de nuestra aplicación (container).
El parámetro -d hace correr la instancia RUN como demonio.

```
# docker run -d -p 3000:3000 --name volt voltframework
```

```
root@ggermanlp:/home/ggerman/docker/volt# docker run -d --name volt -p 3000:3000 voltframework
4602ce309ddc11abcfc83488107660e0865f6c44db4be78c5e014cd58d97885e
```

Creando un container - RUN (2)



Volt: conectando a la base de datos.

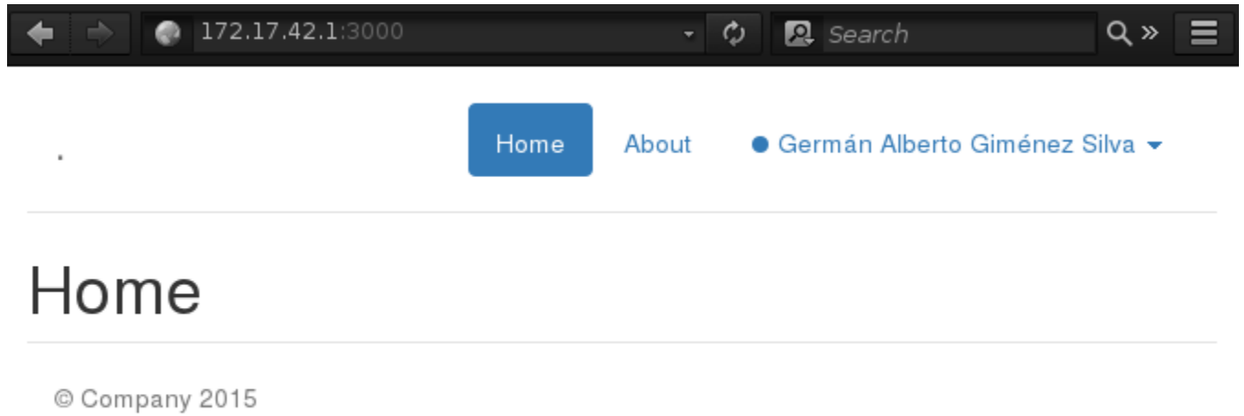
Volt contiene un archivo de configuración config/app.rb. Que a los fines de esta presentación voy a eliminar y copiar uno con mi configuración para que funcione con una conexión a otro container donde tendremos mongoDB.

```
##########
# Database
##########
# Database config all start with db_ and can be set either in the config
# file or with an environment variable (DB_NAME for example).

config.db_driver = 'mongo'
config.db_name = (config.app_name + '_' + Volt.env.to_s)
config.db_host = '172.17.42.1'
config.db_port = 27017
```

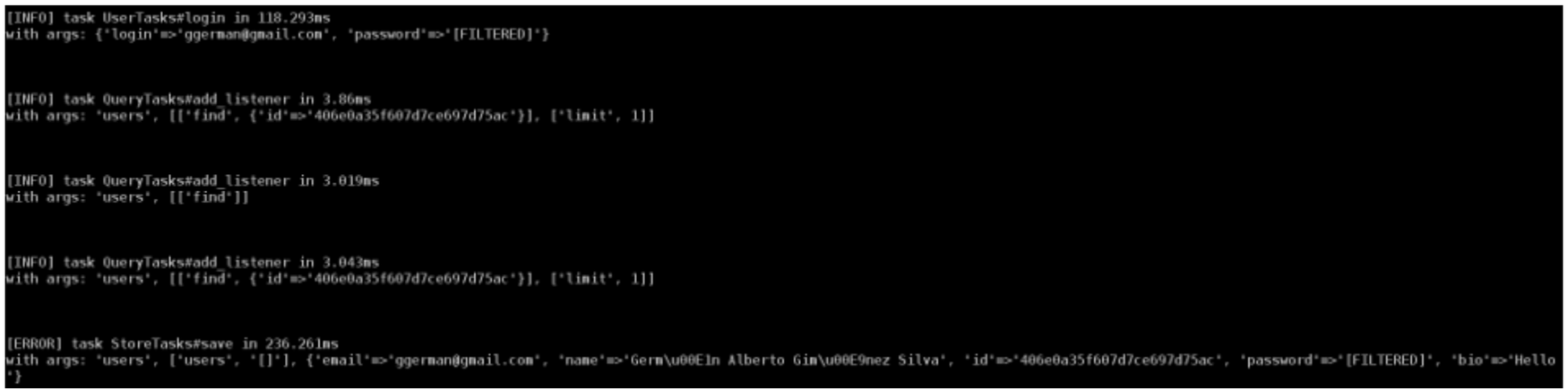
MongoDB: puesta en marcha de la base de datos.

```
# docker run -d --name db --port 27017:27017 mongo
```



MongoDB: puesta en marcha de la base de datos. (2)

```
# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
91531052333f        mongo               "/entrypoint.sh mong   16 minutes ago      Up 16 minutes       0.0.0.0:2701
4602ce309ddc        voltframework       "bundle exec volt se   35 minutes ago      Up 35 minutes       0.0.0.0:3000
# docker logs 4602ce309ddc
```




```
(divagando) >> Ejecutar un comando en un container que ya esta corriendo.
# docker exec -it 4602ce309ddc vim config/app.rb
```

```
root@ggermanlp:/home/ggerman/docker/volt# docker ps -a
  NTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                      NAMES
91531052333f        mongo               "/entrypoint.sh mong   16 minutes ago      Up 16 minutes       0.0.0.0:27017->27017/tcp   db
4602ce309ddc        voltframework       "bundle exec volt se   35 minutes ago      Up 35 minutes       0.0.0.0:3000->3000/tcp     volt
root@ggermanlp:/home/ggerman/docker/volt# docker exec -it 4602ce309ddc vim config/app.rb
```

¡ Listo ! Ya tenemos todo corriendo en containers

DOCKER: Esta cosa tiene commits y layers

Cuando modificamos un container este es tan efímero como lo es nuestro entorno, para que los cambios persistan debemos hacer commits.

```
# docker commit 4602ce309ddc voltframework:version2
```

LAYERS: ¿a qué nos referimos con esto?

Un container no es más que un conjunto de layers agrupados.

```
Los layers son etapas que se generan cuando creamos un container
lo podemos visualizar con el comando build:

Step 1 : FROM busybox
Pulling repository busybox
 ---> e9aa60c60128MB/2.284 MB (100%) endpoint: https://cdn-registry-1.docker.io/v1/
```

Docker tiene la capacidad de reutilizar esos layers todo el tiempo.

```
- Reciclaje de paquetes instalados
- Reciclaje de tiempo de configuración
```

docker-compose: haciendo las cosas más automáticas

Compose es una aplicación que nos permite automatizar toda la tarea de conexión entre containers. Los links son generados por dentro del MAIN HOST de Docker si necesidad de publicar los puertos. De esta manera, lo único que debemos mostrar es el puerto por el que queremos acceder a nuestra aplicación, dejando que docker orqueste las conexiones entre nuestros containers.

docker-compose.yml

```
volt:
  image: voltframework:link
  ports:
    - "3000:3000"
  links:
    - mongo
mongo:
  volumes:
    - /home/myuser/docker/volt/db:/data/db
  image: mongo
```

docker-compose: database link

Para crear la conexión entre los containers al igual que manualmente publicamos el puerto de la base de datos.

```
ports:
  - "27017:27017"
```

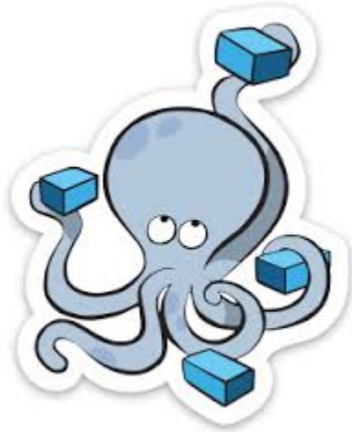
Y dejamos que docker haga el link entre VOLT y Mongo, para esto debemos modificar la configuración de VOLT linkeando con el nuevo nombre.

```
  ##########
  # Database
  ##########
  # Database config all start with db_ and can be set either in the config
  # file or with an environment variable (DB_NAME for example).

  config.db_driver = 'mongo'
  config.db_name = (config.app_name + '_' + Volt.env.to_s)
  config.db_host = 'mongo'
  config.db_port = 27017
```

docker-compose: UP

```
# docker-compose up
```



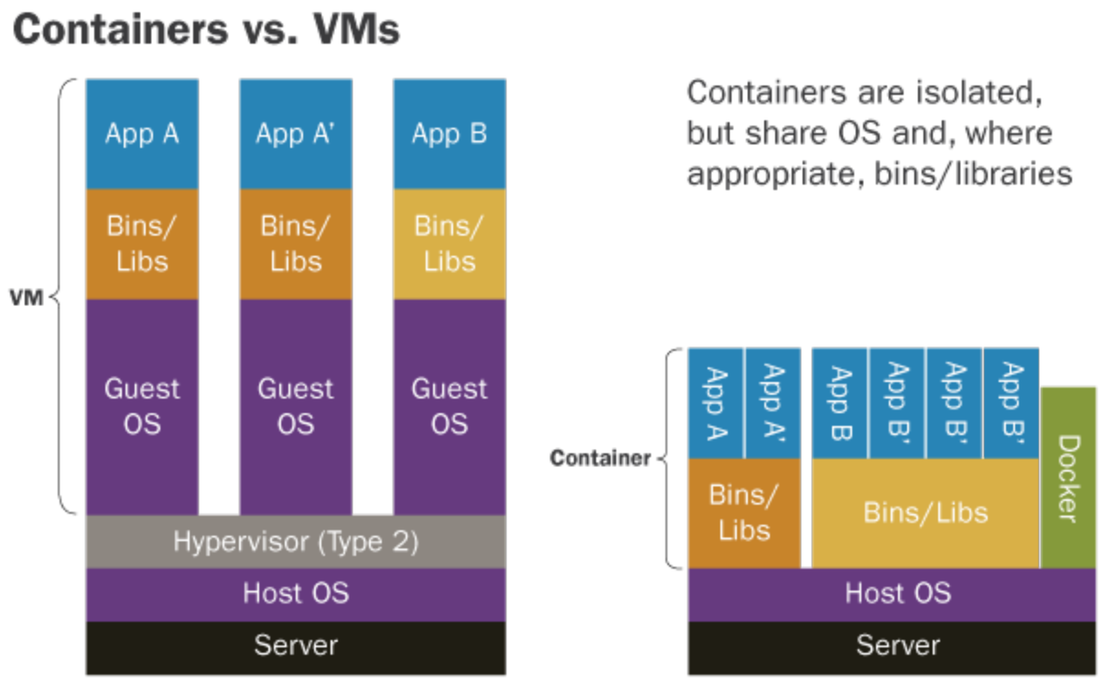
```
[ggerman@ggermanlp:volt(master|5756m)]$ docker-compose up
Creating volt_mongo_1...
Creating volt_volt_1...
Attaching to
```

Vagrant

De www.vagrantup.com: "Create and configure lightweight, reproducible, and portable development environments"

	Docker	Vagrant
A qué nivel se encuentra?	Container	Virtual Machine
Host SO	Linux	Mac, Wondows y Linux (DEB/RPM)
Guest SO	Linux	Lo que sea que corra en Virtualbox/VMware
Donde lo ejecuto?	Linux c/Docker daemon o plataformas DockeraaS (ej.: AWS ECS)	Local, AWS, OpenStack, DigitalOcean y otros
Definicion de 'Imagenes'	Dockerfile	Vagrantfile
Provisioning	Dockerfile DSL	Shell, Chef, Puppet, Ansible, otros
Spin up	**ms**	**segs a mins**

Vagrant (VMs) versus Docker



docker-machine

Miren la hora y pregúntense

28.1. ¿Seguimos con docker-machine?

Preguntas

```
[ggerman@ggermanlp:~]$ docker run busybox /bin/echo '¿Preguntas?'
¿Preguntas?
[ggerman@ggermanlp:~]$
```

Agradecimientos:

- Redacción / Ortografía: María Jimena Franco
- Iniciativa: Manuel Garcia
- Logística: Manuel Garcia, Altoros, EosWeb
- Aguante: a los Presentes

Thank you

Giménez Silva Germán Alberto
Programador para EosWeb
http://www.eosweb.info (http://www.eosweb.info)
ggerman@gmail.com (mailto:ggerman@gmail.com)
@gsgerman (http://twitter.com/gsgerman)

Manuel Garcia
Director Altoros Argentina
www.altoros.com
manuel.garcia@altoros.com (mailto:manuel.garcia@altoros.com)
@rmgarciap (http://twitter.com/rmgarciap)