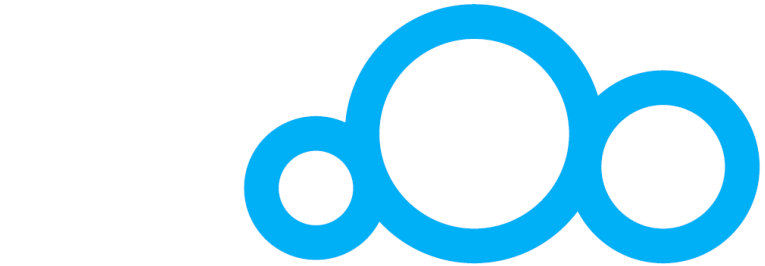


clouding.io

Alojando con Docker en el cloud

Sobre mi

```
# whoami | dig
```

Arnau Marcé

Higher than average Linux user
Black belt Shell Scripting fu
Microsoft Certified Systems Engineer

clouding.io

Cloud servers for everything and everyone
Openstack made easy

clouding.io

Sobre “nosotros”

```
# head -3 /etc/group | dig
```

Xavier Trilla

Openstack and ceph hacker
Master of puppet
Entropy and randominess eliminator

Patricia Armesto

Google algorithm fighter
IT to human language translator
Happy and friendly girl

Jesús Camacho

Ticket squashing master
Customer to IT language translator
Ex digitalocean addict (Now he is addicting to clouding.io)

clouding.io

Objetivo del taller

Desplegar una VM (2 Cores | 2GB RAM | 25 GB SSD) con docker preinstalado en clouding.io y levantar los siguientes servicios:

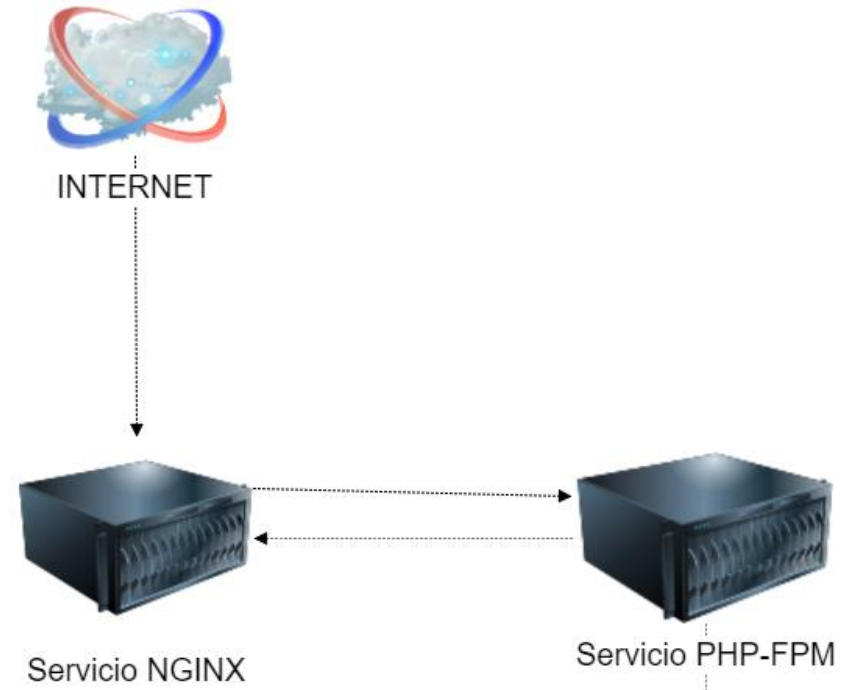
1x Servidor NGINX
1x Servidor PHP-FPM
1x Servidor MariaDB

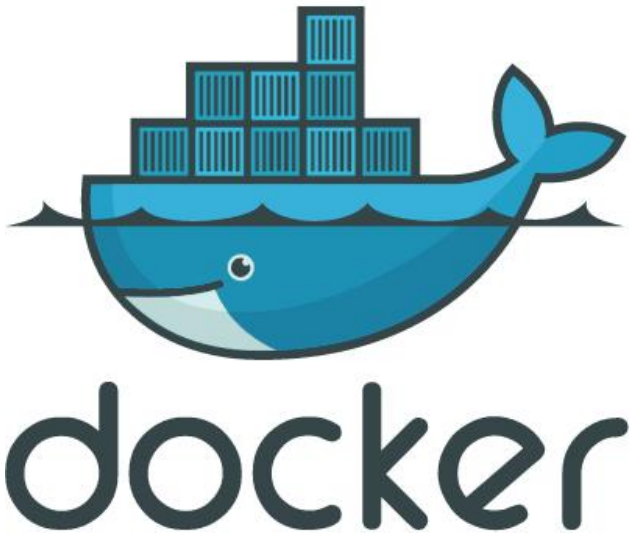
Sobre los que ejecutaremos una instalación de wordpress para simular un caso práctico.

URL presentación: http://docker.amarce.me/taller-docker-nginx.pdf
Repositorio GitHUB: https://github.com/cloudingVPS/taller-docker

clouding.io

Servicios NGINX, PHP FPM y MariaDB

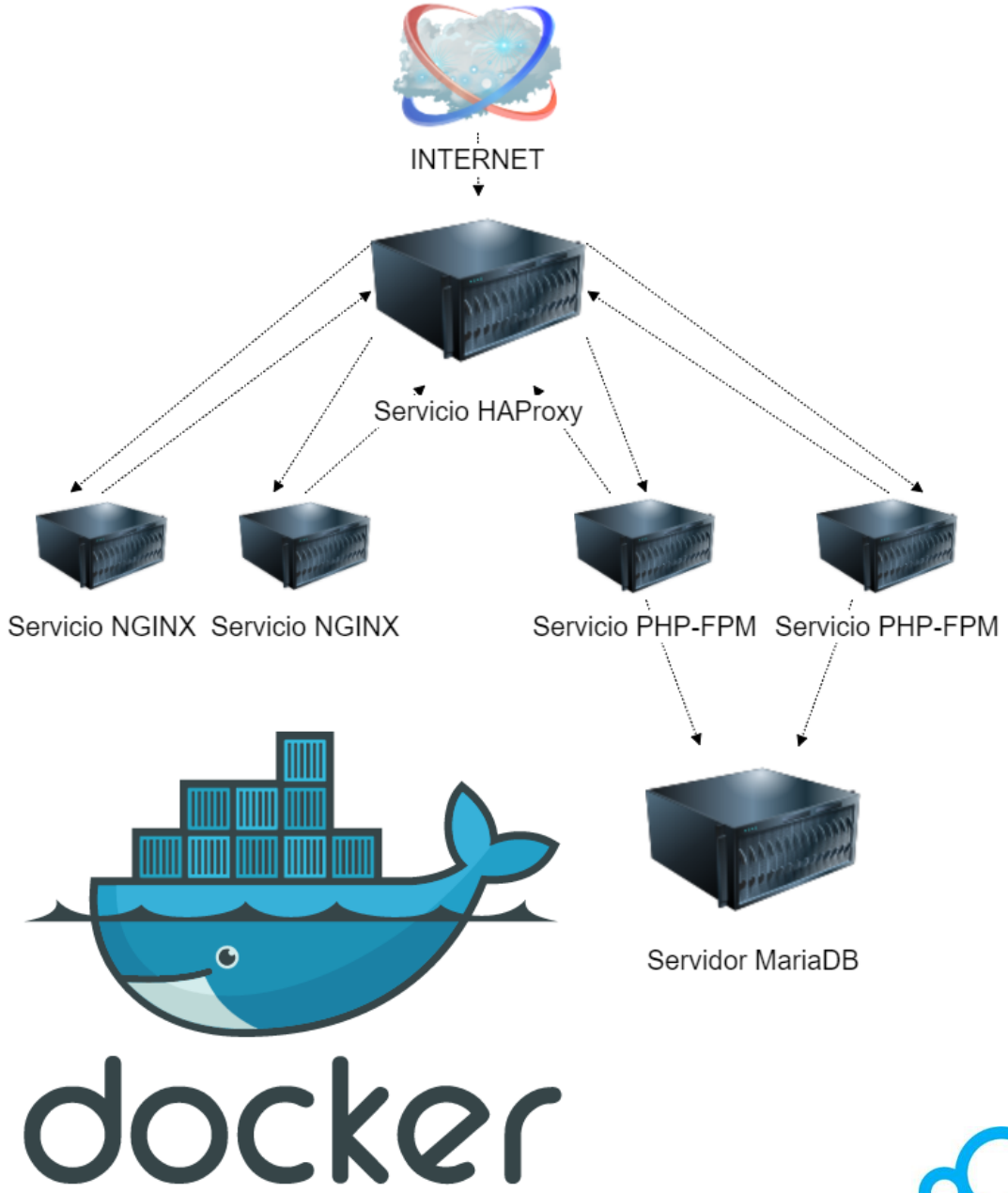




clouding.io

Multiples servicios

NGINX + HAProxy



clouding.io

Actualización de la versión de docker preinstalada

Primeramente, actualizaremos el paquete de docker a la última versión para evitar posibles problemas con docker hub ya que el paquete de docker está “pineado” y no se actualiza con apt-get upgrade o dist-upgrade:

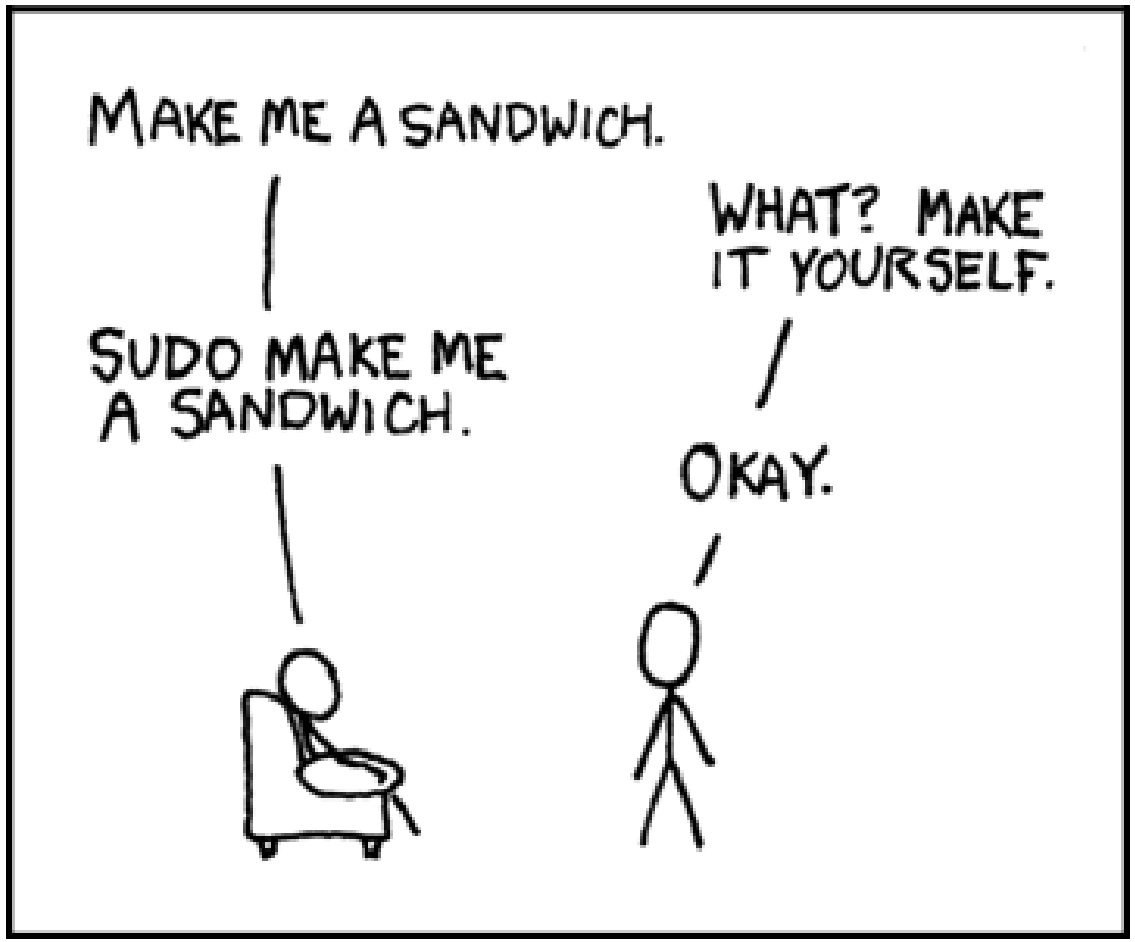
```
#!/Cheatsheet
# apt install lxc-docker

# reboot
```

clouding.io

¿Espera, estás seguro?

```
# whereis sudo
```

El prompt es “#” ¿Estamos ejecutando todo como root?



clouding.io

Distribución de los datos

Los datos estarán en las siguientes rutas:

- Ruta base donde residirá todo: /srv/dockerexports

a) Configuraciones: /srv/dockerexports/etc
   - ✓ Docker: /srv/docker/etc
   - ✓ NGINX: /srv/dockerexports/etc/nginx
   - ✓ PHP: /srv/dockerexports/etc/php
   - ✓ MariaDB: /srv/dockerexports/etc/mysql

b) Datos: /srv/dockerexports/var
   - ✓ NGINX: /srv/dockerexports/var/www
   - ✓ MariaDB: /srv/dockerexports/var/lib/mysql

c) Logs: /srv/dockerexports/var/log
   - ✓ NGINX: /srv/dockerexports/var/log/nginx
   - ✓ PHP: /srv/dockerexports/var/log/php
   - ✓ MariaDB: /srv/dockerexports/var/log/mysql

clouding.io

Creación de directorios

#!/Cheatsheet

```
# mkdir -p /srv/docker/etc/php-7.0.16

# mkdir -p /srv/dockerexports/etc/nginx
# mkdir /srv/dockerexports/etc/php
# mkdir /srv/dockerexports/etc/mysql


# mkdir -p /srv/dockerexports/var/www
# mkdir -p /srv/dockerexports/var/lib/mysql

# mkdir -p /srv/dockerexports/var/log/nginx
# mkdir /srv/dockerexports/var/log/php
# mkdir /srv/dockerexports/var/log/mysql
```

clouding.io

Levantando el servicio PHP-FPM

a) En Docker hub, buscamos php, y apuntamos la 7.0 oficial mas nueva que contenga fpm (En este caso es la 7.0.16-fpm).

b) Creamos un *dockerfile* en la carpeta de configuración de docker php-7.0.16 mediante el cual descargaremos la imagen de docker y activaremos los módulos de PHP necesarios para nuestro wordpress (php5-mysql, php5-gd).

c) Creamos un fichero php.ini con los valores necesarios para que wordpress funcione correctamente (Activamos la extensión mysql, la extensión xmlrpc, la extensión GD y cambiamos upload_max_filesize, post_max_size, memory_limit, file_uploads y max_execution_time).

d) Creamos la imagen de docker utilizando el dockerfile con el modulo de MySQL instalado.

e) Arrancaremos el nuevo contenedor con la imagen que hemos creado, montando nuestro directorio de configuración de PHP y nuestro directorio de datos.

clouding.io

Contenedor PHP-FPM

#!/Cheatsheet

```
# cat /srv/docker/etc/php-7.0.16/Dockerfile
FROM php:7.0.16-fpm
RUN apt-get update && apt-get install -y libxml2-dev libpng-
dev && docker-php-ext-install mysqli pdo pdo_mysql gd xmlrpc

# cat /srv/dockerexports/etc/php/php.ini
upload_max_filesize = 1000M
post_max_size = 2000M
memory_limit = 3000M
file_uploads = On
max_execution_time = 0
extension=mysqli.so
extension=pdo_mysql.so
extension=xmlrpc.so
extension=gd.so

# docker build -t docker/php-fpm:7.0.16 /srv/docker/etc/php-
7.0.16/.

# docker run -d --name php-fpm-01 -v
/srv/dockerexports/etc/php:/usr/local/etc/php -v
/srv/dockerexports/var/www:/var/www/html docker/php-
fpm:7.0.16
```

clouding.io

Levantando el servicio NGINX

a) En Docker hub, buscamos nginx, y vemos la stable mas nueva oficial (En este caso es la 1.10.3).

b) Creamos un fichero de configuración de NGINX que vaya a buscar los ficheros web a nuestra carpeta de datos web, guarde los logs en la carpeta de logs y utilice para PHP nuestro contenedor de FPM.

c) Arrancaremos el nuevo contenedor, haciendo un NAT del puerto 80 hacia el puerto 80 de la máquina, para que todo el tráfico HTTP que llegue al servidor virtual se redirija a la instancia de NGINX de este contenedor. También vincularemos el contenedor de FPM a este para que tengan conexión LAN entre ellos por nombre de host.

clouding.io

Contenedor NGNIX

#!/Cheatsheet

```
# cat /srv/dockerexports/etc/nginx/www.conf
server {
    index index.php index.html;
    server_name _;

    error_log  /var/log/nginx/www_error.log;
    access_log /var/log/nginx/www_access.log;

    root /var/www/html;

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;

        fastcgi_pass php-fpm-01:9000;

        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}

# docker run -d --name nginx-01 -v
/srv/dockerexports/etc/nginx:/etc/nginx/conf.d -v
/srv/dockerexports/var/www:/var/www/html -v
/srv/dockerexports/var/log/nginx:/var/log/nginx -p 80:80 --link php-fpm-01
nginx:stable
```

clouding.io

Probando NGINX y PHP

Crearemos un fichero index.php en nuestra raíz web y probaremos que nos responde todo correctamente:

```
#!/Cheatsheet
```

```
# cat /srv/dockerexports/var/www/index.php
<? phpinfo(); ?>
```

clouding.io

Levantando el servicio MariaDB

a) En Docker hub, buscamos mariadb, y vemos la 10.0 mas nueva oficial (En este caso es la 10.0.29).

b) Arrancaremos el nuevo contenedor, especificando el password para el usuario root, la ruta al directorio de datos de MySQL, la base de datos que queremos crear/utilizar y el puerto del cual queremos hacer NAT a MySQL (En este caso el 3306).

c) Arrancaremos “de nuevo” el contenedor para comprobar que MySQL está funcionando y se ha creado la base de datos correctamente.

clouding.io

Contenedor MariaDB

#!/Cheatsheet

```
# docker run -d --name mysql-01 -p 3306:3306 -e
MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=wordpress
-v /srv/dockerexports/var/lib/mysql:/var/lib/mysql
mariadb:10.0

# docker run -it --link mysql-01 --rm mariadb:10.0 sh -c
'exec mysql -hmysql-01 -P3306 -uroot -ppassword‘

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| wordpress          |
+--------------------+
4 rows in set (0.00 sec)
```

clouding.io

Juntando todos los contenedores

Detenemos todos los contenedores y procedemos a:

a) Eliminar los contenedores de PHP y NGINX.

b) Recrear el contenedor de PHP relinkandolo con el de MariaDB además del de NGINX.

c) Recrear el contenedor de NGINX linkandolo con el de PHP igual que antes ya que al recrearlo ha cambiado el ID y el link antiguo ya no es válido.

d) Probar si al detener e inciar de nuevo los contenedores todo sigue funcionando igual que ahora.

Poniendo todo en marcha

#!/Cheatsheet

```
# docker stop mysql-01 php-fpm-01 nginx-01

# docker rm php-fpm-01 nginx-01

# docker start mysql-01

# docker run -d --name php-fpm-01 --link mysql-01 -v
/srv/dockerexports/etc/php:/usr/local/etc/php -v
/srv/dockerexports/var/www:/var/www/html docker/php-fpm:7.0.16

# docker run -d --name nginx-01 -v
/srv/dockerexports/etc/nginx:/etc/nginx/conf.d -v
/srv/dockerexports/var/www:/var/www/html -v
/srv/dockerexports/var/log/nginx:/var/log/nginx -p 80:80 --link
php-fpm-01 nginx:stable

# docker ps

# docker stop mysql-01 php-fpm-01 nginx-01

# docker start mysql-01 php-fpm-01 nginx-01
```

clouding.io

Instalando wordpress

a) Descargamos la última versión de wordpress y la descomprimimos en la raíz de nuestra carpeta de datos web.

b) Miramos el UID y GID del usuario NGINX del contenedor de NGINX y aplicamos ese owner a los ficheros.

c) Por motivos de seguridad quitamos la escritura a todos de las carpetas y ficheros.

d) Abrimos un navegador apuntando a la IP del servidor y realizamos la primera parte de la instalación de wordpress, pegamos el resultado obtenido en el fichero wp-config.php

e) Volvemos a la ventana del navegador y terminamos la instalación de wordpress.

Instalación de wordpress

#!/Cheatsheet

```
# cd /srv/dockerexports/var/www/

# wget https://wordpress.org/latest.tar.gz

# tar zxvf latest.tar.gz

# mv wordpress/* ./

# rmdir wordpress

# docker exec -it nginx-01 grep nginx /etc/passwd
nginx:x:104:107:nginx user,,,:/nonexistent:/bin/false

# chown -R 104:107 /srv/dockerexports/var/www/*

# chown -R 104:107 /srv/dockerexports/var/www/.

# chmod -R a-w /srv/dockerexports/var/www/*

# chmod -R a-w /srv/dockerexports/var/www/.

ABRIMOS CON UN NAVEGADOR LA IP DEL SERVIDOR Y REALIZAMOS LA CONFIGURACIÓN.
PEGAMOS EL RESULTADO AL FICHERO wp-config.php

# vi /srv/dockerexports/var/www/wp-config.php

FINALIZAMOS LA INSTALACIÓN DE WORDPRESS
```

clouding.io

Bonus – Docker compose

Para arrancar todos los contenedores con un solo comando, usaremos docker compose; es un “pequeño” binario que permite controlar los contenedores linkados y arrancarlos en el orden correcto.

```
#!/Cheatsheet
# apt -y install python-pip

# pip install docker-compose
```

clouding.io

docker-compose.yml

#!/Cheatsheet

```
# apt -y install python-pip
# pip install docker-compose

# cat /srv/docker/etc/docker-compose.yml
php-fpm-01:
  container_name: php-fpm-01
  build: /srv/docker/etc/php-7.0.16
  volumes:
    - /srv/dockerexports/etc/php:/usr/local/etc/php
    - /srv/dockerexports/var/www:/var/www/html
  links:
    - mysql-01

nginx-01:
  container_name: nginx-01
  image: nginx:stable
  ports:
    - 80:80
  volumes:
    - /srv/dockerexports/etc/nginx:/etc/nginx/conf.d
    - /srv/dockerexports/var/www:/var/www/html
    - /srv/dockerexports/var/log/nginx:/var/log/nginx
  links:
    - php-fpm-01

mysql-01:
  container_name: mysql-01
  image: mariadb:10.0
  ports:
    - 3306:3306
  volumes:
    - /srv/dockerexports/var/lib/mysql:/var/lib/mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
    - MYSQL_DATABASE=wordpress
```

clouding.io

/usr/local/bin/docker-compose

```
# man docker-compose
```

Eliminaremos todos los contenedores con la API de docker estándar y arrancaremos con la de docker-compose

Para crear los contenedores:
docker-compose up

Para arrancar los contenedores una vez que están creados:
> docker-compose start

Para parar contenedores:
> docker-compose stop

Para parar contenedores y eliminarlos:
> docker-compose down

```
#!/Cheatsheet
```

```
# docker rm nginx-01 php-fpm-01 mysql-01

# docker-compose up –d
```

clouding.io

Auto-arranque de contenedores

~~# update-rc.d docker enable~~

Cuando arranca el servicio de docker, únicamente se inicializa la API, pero no arrancan los contenedores; para que los contenedores arranquen al arrancar la máquina, la forma mas fácil es añadiendo al fichero /etc/rc.local la siguiente línea (Una vez ejecutado “docker-compose up”):

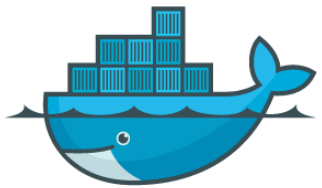
```
#!/Cheatsheet
/usr/local/bin/docker-compose -f
/srv/docker/etc/docker-compose.yml start
```

clouding.io

clouding.io

¡ Muchas gracias por haber participado !

```
# /sbin/poweroff
```



docker NGINX php fpm

MariaDB