Cosas de docker

Peleando con docker prefiero tener a mano unos tips

- Información de los contenedores docker
- Wordpress en docker para desarrollo.
- Proyectos con path public, public_html, app en Docker con Apache+PHP-FPM

Información de los contenedores docker

Introduccion

Aunque usemos herramientas de desktop, etc, muchos comandos son necesarios, ya no solo por su eficacia, sino porque muchas veces las herramientas no cubren todos los aspectos, y porque otras muchas, los manuales y la información que tenemos a mano, habla de ellos, y no de las herramientas de trabajo.

Contenedores

Muchas, muchismas veces es necesario entrar o conocer datos de los contendores.

Listado

```
docker container ls -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS                    PORTS                                                  NAMES
b52d46f0eaa1   sail-8.1/app           "start-container"        4 minutes ago   Up 4 minutes              0.0.0.0:80->80/tcp, 0.0.0.0:5173->5173/tcp, 8000/tcp   sitelight-laravel.test-1
46868801fab5   kibana:8.4.3           "/bin/tini -- /usr/l…"   4 minutes ago   Up 4 minutes              0.0.0.0:5601->5601/tcp                                 sitelight-kibana
be49291ac94e   mysql/mysql-server:8.0   "/entrypoint.sh mysq…"   4 minutes ago   Up 4 minutes (healthy)   0.0.0.0:3306->3306/tcp, 33060-33061/tcp                sitelight-mysql-1
2cbdb7150093   elasticsearch:8.4.3    "/bin/tini -- /usr/l…"   4 minutes ago   Up 4 minutes              0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp         sitelight-es01
3b17bd32cff8   redis:alpine           "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes (healthy)   0.0.0.0:6379->6379/tcp                                 sitelight-redis-1
01eecfe2c5e1   mysql:latest           "docker-entrypoint.s…"   2 weeks ago     Exited (0) 2 weeks ago                                                           mysql
```

Ejecutar un comando en un contenedor activo

```
docker exec -it 2cbdb7150093 /usr/share/elasticsearch/bin/elasticsearch-reset-password -a -u elastic
WARNING: Owner of file [/usr/share/elasticsearch/config/users] used to be [root], but now is [elasticsearch]
WARNING: Owner of file [/usr/share/elasticsearch/config/users_roles] used to be [root], but now is [elasticsearch]
This tool will reset the password of the [elastic] user to an autogenerated value.
The password will be printed in the console.
Please confirm that you would like to continue [y/N]y
```

> NOTA: este comando ya no es necesario en un despligue con docker de elasticsearch y kibana. Entra sin seguridad activa, ni login al cluster.

Aviso

Wordpress en docker para desarrollo.

Cómo usar docker para desarrollo o gestión de incidentes con Wordpress

A veces es necesario tener un entorno de desarrollo para realizar determinadas acciones, como puede ser una actualización conflictiva, un cambio de versión u otras que no se van a necesitar de un entorno que no sea el de producción.

Docker+, es la elección perfecta, al menos para mí, acostumbrado a trabajar para todo con él, acompañandome todo mi desarrollo en mi portatil, como buen nómada que soy.

En el caso descrito habitual relativo a WordPress, muy, muy, muy antiguos, que usan determinadas versiones de PHP, y que al intentar actualizarlas en producción, puede ser un completo desastre, y aunque pese a tener un sistema basado en **rsync** y **mysqldump** en la comnsola de comandos, la web que tenía que actualizar no tenía esa posibilidad de poder estar perdiendo 45 minutos en producción así qué la dockerize.

> Este tip es valido, para cualquier sitio web que queramos actualizar con un minimo de profesionalidad y seguridad

Dockerización de Wordpress

Habitualmente uso Laravel Sail, pero sin embargo, he preferido hacer el tema de WordPress con una instalación independiente y usando Docker composer.

>

> Los datos son específicos a mi entorno, pero si no eres de copiar y pegar y tratas de comprender cada paso, podrás trabajar donde quieras y como quieras que se adapte a tus necesidades. Esto supone diertas configuraciones de docker que pueden hacer que en otros escenearios o configuraciones no funcione lo que digo, sobre todo si la confioguración especifica afecta al modo de red. Ver
>
> Docker Tips :: Algos sobre redes

> Se recomienda leer los tips

Creación de del entorno de trabajo

Directorio de trabajo

```
mkdir ~/Sites/myweb
cd $_
```

Creación del docker-compose.yml

La creación la haremos en un principio ajustándose al escenario de funcionamiento, para luego progresivamente ir subiendo a la versión que queremos, para lidiar con todos los problemas de plugins, templates desactualizados.

Por tanto lo suyo no es copiar, por ejemplo las imágenes de docker que se indican, sino las que necesitamos obteniendo la información de las tags, en el hub de docker.

La imagen de Wordpress deberá ser la adecuada al escenario de nuestro sitio con problema en producción.

Versión de PHP, apache o nginx, fpm,...

Más adelante iremos subiendo a la que queremos.

> Uso docker desktop por muchas razones, y lo recomiendo.

Editamos el fichero `editor ~/Sites/myweb/docker-compose.yml`

```
services:
  db:
    # We use a mariadb image which supports both amd64 & arm64 architecture
    # image: mariadb
    # If you really want to use MySQL, uncomment the following line
    image: mysql:8
    command: '--default-authentication-plugin=mysql_native_password'
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=la_misma_que_en_producción
      - MYSQL_USER=el_mismo_que_en_producción
      - MYSQL_PASSWORD=PasSW0rD_que_en_producción
    ports:
      - "3306:3306"
  wordpress:
    image: wordpress:php7.4-fpm-alpine
    volumes:
      - ./wp_data:/var/www/html
    ports:
      - 80:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=el_mismo_que_en_producción
      - WORDPRESS_DB_PASSWORD=PasSW0rD_que_en_producción
      - WORDPRESS_DB_NAME=la_misma_que_en_producción
volumes:
  db_data:
  wp_data:
```

Explicación

- **Mysql** lo montamos con un volumen, para hacer permanente sus datos.
- La instancia de **WordPress** lo mismo, poniendo el directorio de montaje en el directorio en el que vamos a clonar el sitio, apuntando al directorio `/var/www/html` del volumen.

Copia del contenido de producción

La mejor opción es `rsync` una herramienta **imprescindible** no sólo para los administradores de sistemas, sino para los webmaster o mantenedores de sitios web.

```
$ rsync -avzzz --progress -e "ssh" user_remoto@dominio_remoto.tld:/path/al/sitio/ ~/Sites/myweb/wp_data/
receiving file list ...
33791 files to consider
./
license.txt
      19915 100%   18.99MB/s    0:00:00 (xfer#1, to-check=33778/33791)
readme.html
       7402 100%    7.06MB/s    0:00:00 (xfer#2, to-check=33774/33791)
wp-config-sample.php
       3013 100%    2.87MB/s    0:00:00 (xfer#3, to-check=33766/33791)
wp-config.php
       3601 100%    3.43MB/s    0:00:00 (xfer#4, to-check=33765/33791)
...
...
wp-includes/widgets/class-wp-widget-text.php
      21342 100%   20.99kB/s    0:00:00 (xfer#2301, to-check=0/33791)
sent 479944 bytes  received 860743 bytes  536274.80 bytes/sec
total size is 1363857930  speedup is 1017.28
```

Copia o dump de la base de datos de producción

Podemos usar nuestra herramienta preferida, como TablePlus, Navicat, phpMyAdmin... o el propio shell.

El caso es hacer una copia completa.

Me gusta usar el shell.

Como no tengo acceso mysql remoto, y no me apetece montar un túnel, hago el backup en remoto y luego lo bajo con `rsync`

Servidor remoto

```
mysqldump --no-tablespaces --opt -u <wp_user> -p  <database>  > ~/<database>.sql
Enter password:
```

Servidor local

Ahora traemos el fichero de dump

```
rsync -avzz --progress -e "ssh -p 2244" user_remoto@remoto.tld:/home/myuser/<database>.sql .
receiving file list ...
1 file to consider
<database>.sql
  130317359 100%    7.80MB/s    0:00:15 (xfer#1, to-check=0/1)
sent 25352 bytes  received 35045636 bytes  2004056.46 bytes/sec
total size is 130317359  speedup is 3.72
```

> Atención al `.` final del comando que quiere decir `cópiame aquí`

Restaurar la copia

Obtener información del container

```
docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                    NAMES
9b780079bbeb   wordpress:latest   "docker-entrypoint.s…"   40 minutes ago   Up 40 minutes   0.0.0.0:80->80/tcp               myweb-wordpress-1
c591fdcae17a   mysql:8            "docker-entrypoint.s…"   40 minutes ago   Up 40 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp   myweb-db-1
```

Volcar el backup

Usaremos el **id** del contenedor de mysql `c591fdcae17a` para ejecutar un comando en él.

```
cat <database>.sql | docker exec -i c591fdcae17a /usr/bin/mysql -u root --password=somewordpress <database>
mysql: [Warning] Using a password on the command line interface can be insecure.
```

> En mi caso por vaquería y comodidad, uso TablePlus para muchas cosas, y tiene un importador. Pero siempre hay una vez para fallar, y esta vez, fue al importar una bd con 45Mb, (he importado en table plus bd de 6 GB sin problemas). Se terminaba siempre en una de las tablas. Fue hacerlo en el el shell, y funcionó sin problemas.

Levantamos la imagen

```
docker compose up -d
```

Para probar que esta todo correcto, solo tendríamos que ir a nuestro navegador y solicitar localhost

Devolver a su sitio.

Tras actualziar y ajustar todo a las últimas versiones tanto de wordpress, como de PHP, Mysql o MariaDb, deberemos subirlo a producción.

Para devolver a su sitio la actualización completa es sencillo. Usamos la misma técnica de `rsync` + `mysqldump` pero al revés.

Mysqldump

> “ Esta vez usaremos el **name** del container en lugar del **ID**

```
docker exec -i myweb-db-1 /usr/bin/mysqldump --opt -u root --password=somewordpress --databases <database> --skip-comments > dump-updated.sql
```

Subir los cambios a producción

```
 ~/Sites/myweb $ rsync -avzz --progress --delete -e "ssh -p 2244" ~/Sites/myweb/wp_data/ user_remoto@dominio_remoto.tld:/path/al/sitio/
~/Sites/myweb $  rsync -avzz --progress --delete -e "ssh -p 2244" ~/Sites/myweb/dump-updated.sql user_remoto@dominio_remoto.tld:/path/al/user/
```

Restaurar la base de datos en el servidor de producción

```
Mysql -u <user_de_mysql> -p <bd_produccion>  < ~/dump-updated.sql
```

Tips.

- Editar `/etc/hosts` para que apunte a a local (127.0.0.1) y evitar líos de redirecciones. En el caso de MacOsx, sería en `/private/etc/hosts`

```
127.0.0.1  myweb.com
```

- Usar un navegador permisivo o configurado para no usar https (safari, opera...). Si tenemos algun plugin que fuerce la redireción `http` a `https` deberías desactivarlo.
- Usar una página `info.php` para tener las cosas claras. Simplemente debe tener `<?php phpinfo();` y ponerla en el `public` de nuestra web.
- Restaurar el backup (mejor desde shell como siempre)
- Los datos de creación de mysql, user y password deben ser los mismos en el par, producción-desarrollo.
- El parámetro del fichero `wp-config.php` `define('DB_HOST', 'localhost');` debe estar definido a 'localhost'.
- **editor** es el alias de mi editor preferido.
- `$` indica que estamos en una cuenta de usuario y su path y `#` es root
- `<variable>` indica que debe ser sustituido por los valores apropiados incluyendo en el reemplazo los `<>`
- Atención a la barra `/` al final de los parámetros de `rsync` porque no es lo mismo con ella que sin ella.

A trabajar con el tema.

Tips globales

- Una vez que ya hemos conseguido la actualización menor necesaria para poder pasar a una versión más alta de php sin problemas, simplemente debemos editar el fichero `docker-compose.yml` y solicitar la nueva versión de php, y reconstruir el volumen de Wordpress. (Borrar container y volumen wordpress.
- Siempre debemos trabajar inicialmente con las mismas versiones y software que como está en producción. Hay que recordar que existe, versiones de Mysql, MariaDb (no son iguales no... ojo a ese datos que más de un quebradero de cabeza os podéis llevar un

día), versiones de PHP, con FPM, mod, con Nginx, con Apache, etc.

- Si no te sientes cómodo con el shell, al menos elige un hosting que tenga una herramienta de bacjup como Jetbackup que te permite crear snapshots antes de liarte con cambios.

Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de Castris Si necesitas soporte profesional puedes contratar con Castris soporte profesional.

Proyectos con path public, public_html, app en Docker con Apache+PHP-FPM

Introducción

Existen desarrollos en software web, que siguen una pauta moderna y más segura, consistente en exponer solamente al público el fichero index.php dejando en el `doc_root` de **Apache** o **Nginx** el directorio que lo contiene, y no todos el desarrollo. Ejemplos clásicos son, Laravel, Symfony, Slim, y otros muchos. aunque si no eres amigo de los marcos de trabajo (framework), bueno es que también uses esta formula de trabajo y no meter todo en el directorio del `doc_root` del servidor Web.

En mi último trabajo he tenido que lidiar con una aplicación obsoleta, en PHP 5.6 desarrollada con el marco Zend Framework, que estaba en un cPanel pero con las estructura correcta por debajo del `public_html`. (Lastima que encima puso un blog con Wordpress, y se le olvido actualizarlo)

Esta guía te mostrara el camino de la creación de los contenedores necesarios para Apache, PHP-Fpm y MySQL para ejecutar tu desarrollo, usando **Docker compose**

Docker Apache, PHP FPM setup with docker compose y MySQL

Cuando uno llega aquí, lo único que queda es dockerizar nuestro area de trabajo, ya que desarrollar en el servidor está... **prohibido**

El proceso esta indicado para un escenario concreto, pero con los ajustes necesarios, puedes elegir la versión de PHP, la de MySQL/MariaDb/Percona que necesites (siempre trabaja con el mismo escenario que en producción).

Requisitos

Necesitamos tener instalado en nuestro equipo Docker.

Generaremos la siguiente estructura en nuestro proyecto

```
proyecto
├── Dockerfile
├── docker-compose.yml
├── apache/
│ │ ├── apache.vhost.conf
│ │ └── Dockerfile
├── public/
│    └── index.php
├── vendor/
├── ...
```

Algo así quedaría

Image not found or type unknown

- **Dockerfile** global que contiene la configuración necesaria para crear y preparar el escenario para un PHP con PHP-FPM y sus extensiones necesarias en las imágenes necesarias.
- **docker-compose.yml** que crear los contenedores necesarios y los unirá.
- **apache/apache.vhost.conf** que contendrá la información necesaria para que Apache sea el servidor web de nuestro proyecto.
- **apache/Dockerfile**
- **public** que será el directorio root de nuestro proyecto, expuesto a internet por el par Apache/PHP-FPM

Crear la imagen PHP FPM

Te damos las instrucciones necesarias a docker para construir la imagen. El ejemplo es el básico que me permite usar y trabajar en esta dockerización, con el proyecto Zend en un PHP 5.6. Si necesitas más extensiones, o algún cambio necesitaras un trabajo extra para añadirlo.

Consulta siempre en Hub Docker las posibilidad que tienes, y si quieres un consejo, usa siempre imágenes oficiales. Te ahorraras muchos disgustos.

> Los comentarios indican modificación sobre la imagen oficial. Si usas una no oficial, o versiones oficiales basadas en otro so, como alpine, etc. los paths indicados pueden sufrir variaciones

```
FROM php:5.6-fpm

RUN docker-php-ext-install mysqli pdo pdo_mysql

## Esta linea la puedes comentar si tu proyecto tiene un estructura plana, como puede ser un Wordpress.
Donde todo está en el mismo nivel del doc_root
RUN echo 'php_admin_value[doc_root] = /var/www/html/public' >> /usr/local/etc/php-fpm.d/www.conf
```

- Docker hará un pull **php:5.6-fpm**
- Si quieres o necesitas instalar una extensión adicional con Docker usa el commando **docker-php-ext-install** dentro de doc usando **RUN**

Crear Apache

Necesitamos crear la imagen con el servidor web Apache, atendiendo a que usaremos PHP-FPM como manejador de PHP. Así que usaremos un fichero de configuración del host virtual, y usaremos **apache/apache.vhost.conf** para este propósito.

```
# La configuracion esta adaptada al escenario indicado public/
# Si no es así deberá modificarse al típico /var/www/html/ allí donde este usándose public/
# Establezca el nombre del servidor en localhost
ServerName localhost

# Configure un VirtualHost para manejar solicitudes en el puerto 80
<VirtualHost *:80>
    # Solicitudes de proxy PHP para portar el contenedor PHP-FPM 9000
    ProxyPassMatch ^/(.*\.php(/.*)?)$ fcgi://php-fpm:9000/var/www/html/public/$1

    # Set the DocumentRoot for the virtual host
    DocumentRoot /var/www/html/public

    # Directory configuration for the DocumentRoot
    <Directory /var/www/html/public>
        DirectoryIndex index.php
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    # Definir los destinos CustomLog y ErrorLog
    CustomLog /proc/self/fd/1 common
```

```
    ErrorLog /proc/self/fd/2
</VirtualHost>
```

- La aplicación se ejecutara en modo localhost y responderá a las peticiones que se hagan la puerto 80.
- Apache reenviará las peticiones php al container PHP-FPM, cuyo nombre es 'php-fpm'. Este nombre será usado en el `docker-compose.yml` del proyecto.
- `/var/www/html/public/` será el directorio por defecto de tu sitio virtual de Apache. Por tanto sería el index o fichero que este allí.
- Si deseas depurar tu aplicación, cree destinos CustomLog y ErrorLog para administrar los registros de la aplicación y los de Docker.

Ahora construiremos nuestro **Dockerfile** para construir la imagen de nuestro servidor web Apache

```
# httpd base image
FROM httpd:2.4

# Copy the Apache virtual host configuration file to the container
COPY ./apache/apache.vhost.conf /usr/local/apache2/conf/extra/apache.vhost.conf

# Activar módulos de Apache para garantizar una funcionalidad adecuada.
RUN sed -i \
    # Descomente la configuración de mod_rewrite para habilitar el control de reescritura contenido.
    -e '/#LoadModule rewrite_module/s/^#//g' \
    # Descomente la configuración de mod_rewrite para habilitar el control de caducidad del contenido.
    -e '/#LoadModule expires_module/s/^#//g' \
    # Descomente la configuración de mod_deflate para habilitar la compreión
    -e '/#LoadModule deflate_module/s/^#//g' \
    # Descomente la configuración del poryx para habilitar su uso.
    -e '/#LoadModule proxy_module/s/^#//g' \
    # Descomente la configuración de mod_proxy_fcgi para habilitar el módulo proxy FastCGI
    -e '/#LoadModule proxy_fcgi_module/s/^#//g' \
    /usr/local/apache2/conf/httpd.conf

# Incluir el archivo de configuración del host virtual en la configuración principal de Apache.
RUN echo "Include /usr/local/apache2/conf/extra/apache.vhost.conf" >> /usr/local/apache2/conf/httpd.conf
```

Crear el container

Bien es hora de armarlo todo.

> En mi caso, no uso PHPMyAdmin así que no instalaremos nada adicional. Nunca lo he usado porque nunca me ha parecido correcto tenerlo instalado. En su lugar uso un programa de consola, y me conecto via tunnel SSH. Pero os dejo comentada la opción. Gracias TablePlus

```
version: '3.9'
services:
  php-fpm:
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
      - .:/var/www/html
    depends_on:
      - mysql-db

  apache-httpd:
    build:
      context: .
      dockerfile: ./apache/Dockerfile
    volumes:
      - .:/var/www/html
    ports:
      - "8080:80"
    depends_on:
      - php-fpm
      - mysql-db
  mysql-db:
    image: mariadb:10.1
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: base_de_datos
      MYSQL_USER: user_datos
      MYSQL_PASSWORD: contraseña
    ports:
      - "3336:3306"
    volumes:
      - mysql-data:/var/lib/mysql
# Mac not working
```

```
#  phpmyadmin:
#    image: phpmyadmin/phpmyadmin:latest
#    links:
#      - mysql-db
#    ports:
#      - "8081:80"
#    environment:
#      PMA_HOST: mysql-db
#      MYSQL_ROOT_PASSWORD: password
volumes:
  mysql-data:
```

- `.:/var/www/html` esto indica que crearemos un **volumen** que montara en la raíz de nuestro proyecto `.` en `/var/www/html/` en el container.
- El puerto expuesto para acceder será el 8080
- Crearemos un container para **MySQL** que tendrá persistencia de datos, en un volumen **mysql-data** que permitirá tener persistencia de datos.
- Mi consejo es que uses los mismo datos para el usuario, base de datos, y contraseña relativos a tu proyecto, que los que usas en producción, por si realizas sincronizaciones. El root no lo toques.
- El puerto expuesto es el 3336 ya que tengo en local, una instalación **Herd** que es la que uso habitualmente y no me apetece apagarla.

Testing

Para evitar perdidas de tiempo, es bueno hacer dos comprobaciones.

- Que el container esta preparado para trabajar con PHP en el directorio `public`
- Que la conexión con Mysql funciona.

info.php

Crea un fichero `public/info.php`

```
<?php phpinfo();
```

index.php

Si ya tienes uno renombrarlo, para más tarde y crea un nuevo en `public/index.php`

> También pues añadir al código al principio del index.php y salir con un `exit();`

```
<?php
$host = 'mysql-db';
$user = 'root';
$pass = 'password';
$db = 'mysql';



$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

echo "Connected to MySQL successfully";

$conn->close();
?>
```

Construir los contenedores

Bueno, ya esta todo.

```
docker-compose up --build -d
docker-compose up --build -d
[+] Building 2.7s (16/16)
FINISHED
              docker:desktop-linux
 => [php-fpm internal] load build definition from
Dockerfile
     0.0s
 => => transferring dockerfile:
204B
                      0.0s
 => [php-fpm internal] load metadata for docker.io/library/php:5.6-
fpm                                                                    1.6s
 => [php-fpm internal] load
.dockerignore
                              0.0s
 => => transferring context:
```

2B
0.0s
=> [php-fpm 1/3] FROM docker.io/library/php:5.6-
fpm@sha256:4f070f1b7b93cc5ab364839b79a5b26f38d5f89461f7bc0cd4bab4a3ad7d67d7
0.0s
=> CACHED [php-fpm 2/3] RUN docker-php-ext-install mysqli pdo
pdo_mysql
0.0s
=> CACHED [php-fpm 3/3] RUN echo 'php_admin_value[doc_root] = /var/www/html/public' >>
/usr/local/etc/php-fpm.d/www.conf 0.0s
=> [php-fpm] exporting to
image
0.0s
=> => exporting
layers
0.0s
=> => writing image
sha256:552b673bf2989a93258340ee86ae7c31477c946f95aa73a63af49cede70235a6
0.0s
=> => naming to docker.io/library/staybarcelona-php-
fpm
0.0s
=> [apache-httpd internal] load build definition from
Dockerfile
0.0s
=> => transferring dockerfile:
1.17kB
0.0s
=> [apache-httpd internal] load metadata for
docker.io/library/httpd:2.4
0.9s
=> [apache-httpd internal] load
.dockerignore
0.0s
=> => transferring context:
2B
0.0s
=> [apache-httpd internal] load build
context

```
              0.0s
 => => transferring context:
72B
                     0.0s
 => [apache-httpd 1/4] FROM
docker.io/library/httpd:2.4@sha256:104f07de17ee186c8f37b9f561e04fbfe4cf080d78c6e5f3802fd08fd118c3da
                                                                  0.0s
 => CACHED [apache-httpd 2/4] COPY ./apache/apache.vhost.conf
/usr/local/apache2/conf/extra/apache.vhost.conf
              0.0s
 => CACHED [apache-httpd 3/4] RUN sed -i             -e '/#LoadModule rewrite_module/s/^#//g'         -e
'/#LoadModule expires_module/s/^#//g'     -e '/#LoadModule deflate_module/s/^#//g'     -e '/#LoadModule
proxy_module/s/^#  0.0s
 => CACHED [apache-httpd 4/4] RUN echo "Include /usr/local/apache2/conf/extra/apache.vhost.conf" >>
/usr/local/apache2/conf/httpd.conf                                                  0.0s
 => [apache-httpd] exporting to
image
                   0.0s
 => => exporting
layers
                            0.0s
 => => writing image
sha256:7428ace356414dd78d15c185ffa4137e7e3806018ce31eac61be778c773738f5
                                                          0.0s
 => => naming to docker.io/library/staybarcelona-apache-
httpd
0.0s
[+] Running 3/3
 ✔ Container staybarcelona-mysql-db-1
Started
        0.0s
 ✔ Container staybarcelona-php-fpm-1
Started
        0.6s
 ✔ Container staybarcelona-apache-httpd-1  Started
```

Comprobaciones

En el navegador probaremos la información del `phpinfo()` en la ruta `localhost:8080`

Info
Image not found or type unknown

Ahora la conexión **mySQL**

Info
Image not found or type unknown

Conclusión

Bien, esto es un esbozo de como podemos adecuarnos con Docker Composer, en un escenario como el descrito además de aprender ciertas metodologías con Docker.

Esto es muy básico y dependerá de la instalación y trabajos que tienes en tu máquina. Hay muchos caminos y formas de trabajar con Docker. Esta es una rápida para un escenario particular.

Aviso