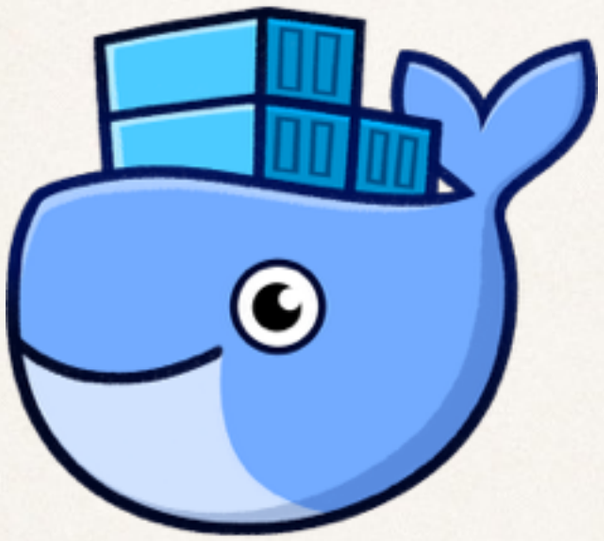
WordCamp Sevilla 16



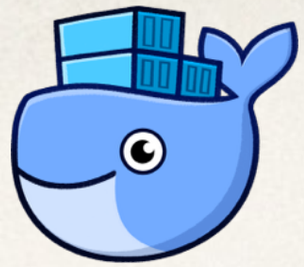
&

Docker y WordPress

Eric Zeidan

@ericjanzei

Docker y WordPress



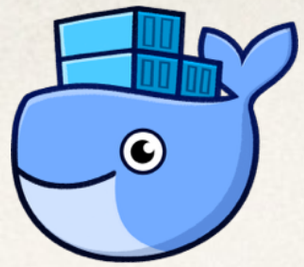
&

- ¿Que es Docker?

Docker es un proyecto de código abierto con el que fácilmente podremos crear "contenedores". Los contenedores de Docker podríamos definirlos como máquinas virtuales ligeras, menos exigentes con nuestros equipos.

Las características principales de los contenedores son: portabilidad, ligereza y autosuficiencia.

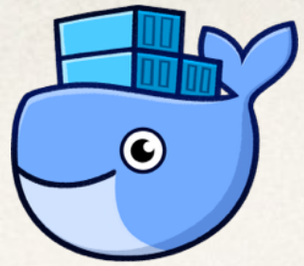
Docker y WordPress


&


- Portabilidad.

El contenedor Docker podremos desplegarlo en cualquier otro sistema (que soporte esta tecnología), con lo que nos ahorraremos tener que instalar en éste nuevo entorno todas aquellas aplicaciones que normalmente usamos para desarrollar.

Docker y WordPress

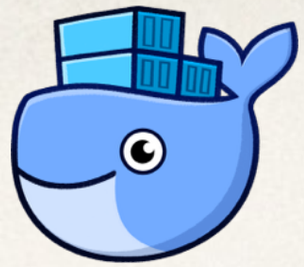


&

- Ligereza.

  El peso de éste sistema no tiene comparación con cualquier otro sistema de virtualización más convencional que estemos acostumbrados a usar. Por ejemplo, una de las herramientas de virtualización más extendida es VirtualBox, y cualquier imagen de Ubuntu que queramos usar en otro equipo pesará mas de 1Gb si contamos únicamente con la instalación limpia del sistema. En cambio, un Ubuntu con Apache y una aplicación web en Docker, pesa alrededor de 180Mb, lo que nos demuestra un significativo ahorro a la hora de almacenar diversos contenedores que podamos desplegar con posterioridad.

Docker y WordPress



&

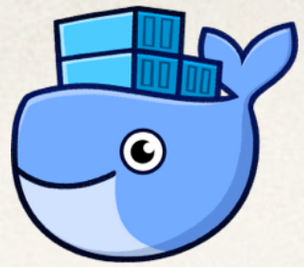
- Autosuficiencia.

Un contenedor Docker no contiene todo un sistema completo, sino únicamente aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. Asimismo Docker se encarga de la gestión del contenedor y de las aplicaciones en él.

Además, su ligereza es lo que más gusta, puesto que incluso en equipos antiguos se desenvuelve prácticamente igual que el sistema anfitrión. A parte, nos ofrece un entorno similar a Git para, a base de "capas", controlar cada cambio que se haga en la máquina virtual o contenedor.

Para obtener esta fluidez Docker extiende LXC (LinuX Containers), un sistema de virtualización ligero que permite crear múltiples sistemas totalmente aislados entre sí sobre la misma máquina o sistema anfitrión. Y todo dado que no se emula un sistema operativo completo, sólo las librerías y sistemas de archivos necesarios para la utilización de las aplicaciones que tengamos instaladas en cada contenedor.

Docker y WordPress



&

Docker se compone de tres elementos fundamentales:

- Los Contenedores Docker:

  Son como un directorio, contienen todo lo necesario para que una aplicación pueda funcionar sin necesidad de acceder a un repositorio externo al contenedor. Cada uno de éstos es una plataforma de aplicaciones segura y aislada del resto que podamos encontrar o desplegar en la misma máquina host.

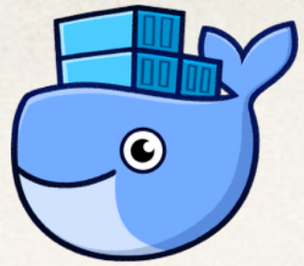
- Las Imágenes Docker:

  La imagen Docker podríamos entenderla como un SO con aplicaciones instaladas (Por ejemplo un OpenSUSE con un paquete ofimático). Sobre la base podremos empezar a añadir aplicaciones que vayamos a necesitar en otro equipo donde tengamos intención de usar la imagen. Docker también nos ofrece una forma muy sencilla de actualizar las imágenes que tengamos creadas, así como un sencillo método para crear nuevas imágenes.

- Y el Repositorio Docker:

  Contienen imágenes creadas por los usuarios y puestas a disposición del público. Podemos encontrar repositorios públicos y totalmente gratuitos o repositorios privados donde tendremos que comprar las imágenes que necesitemos. Éstos registros permiten desarrollar o desplegar aplicaciones de forma simple y rápida en base a plantillas, reduciendo el tiempo de creación o implementación de aplicaciones o sistemas.

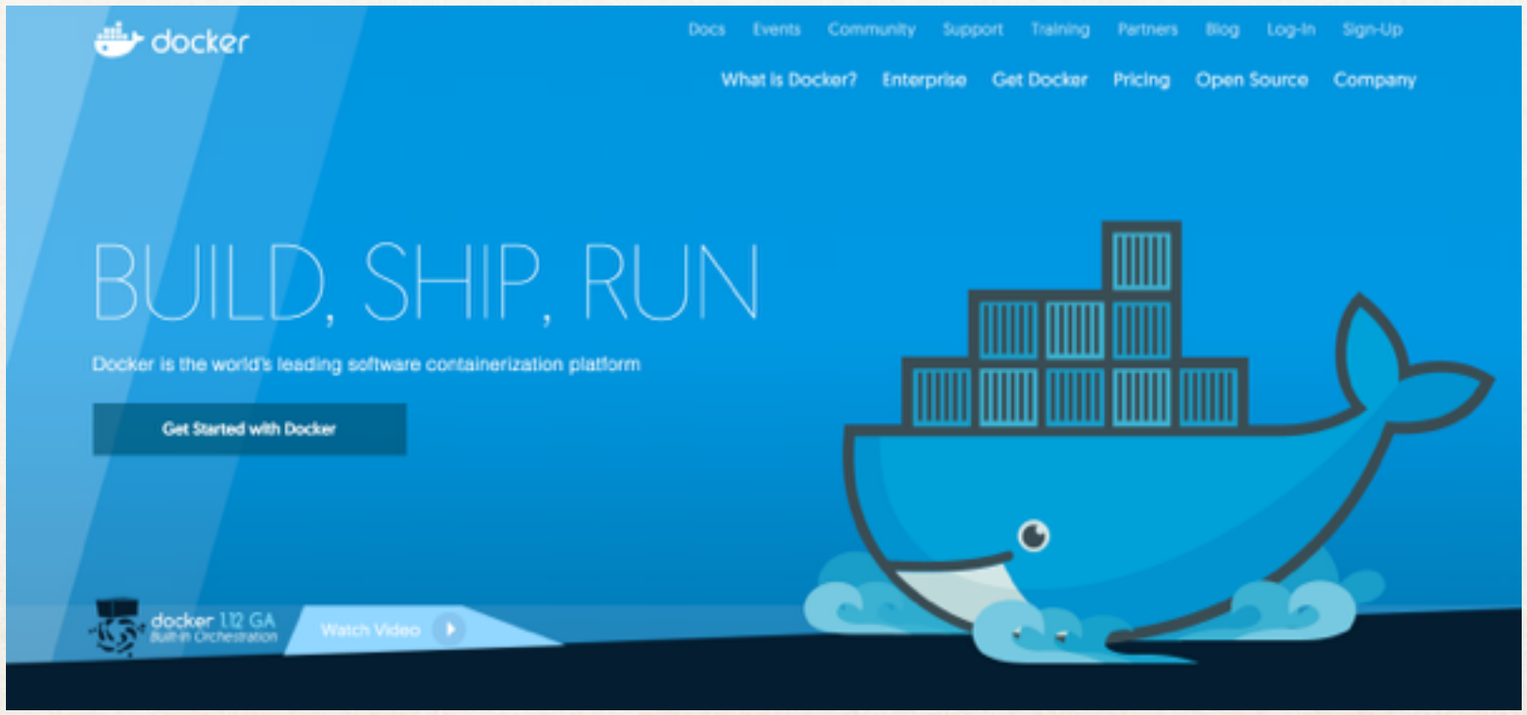
Docker y WordPress



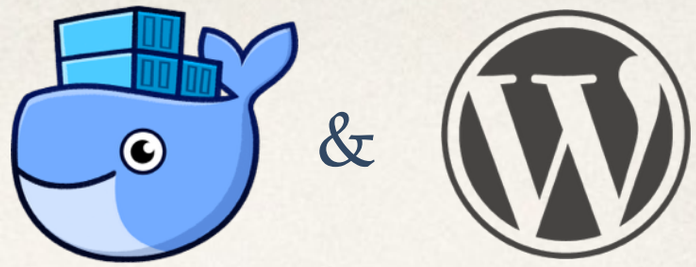
&

- Primeros Pasos

  Primero debemos instalar Dockers en nuestro equipo, para ello podemos buscar los métodos de instalación en su página web https://www.docker.com/. En ella encontraremos toda la información sobre como instalarlo dependiendo de nuestro entorno ya sea, Linux, Mac o Windows.



Docker y WordPress



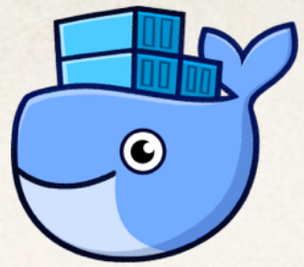
Una vez instalado podemos probar montar un entorno para WordPress ejecutando este simple comando desde nuestro Terminal.

$ docker run --name algun-wordpress --link algun-mysql:mysql -d wordpress

y podríamos usar todas estas variables

- -e WORDPRESS_DB_HOST=... (especificaremos la IP y el puerto del mysql)
- -e WORDPRESS_DB_USER=... (el usuario de nuestra BBDD, por defecto es "root")
- -e WORDPRESS_DB_PASSWORD=... (la contraseña del usuario por defecto root MYSQL_ROOT_PASSWORD o la que coloquemos a nuestro usuario)
- -e WORDPRESS_DB_NAME=... (el nombre de nuestra BBDD por defecto se usa "wordpress")
- -e WORDPRESS_TABLE_PREFIX=... (el prefijo de la tabla, por defecto es "", solo se usa si necesitas sobreescribir la que viene por defecto en el wp-config.php)
- -e WORDPRESS_AUTH_KEY=..., -e WORDPRESS_SECURE_AUTH_KEY=..., -e WORDPRESS_LOGGED_IN_KEY=..., -e WORDPRESS_NONCE_KEY=..., -e WORDPRESS_AUTH_SALT=..., -e WORDPRESS_SECURE_AUTH_SALT=..., -e WORDPRESS_LOGGED_IN_SALT=..., -e WORDPRESS_NONCE_SALT=... (y estos están por defecto a SHA1s)

Docker y WordPress



&

- A continuación veamos un ejemplo en vivo de como funciona con el siguiente comando

  primero debemos crear una imagen para nuestro mysql

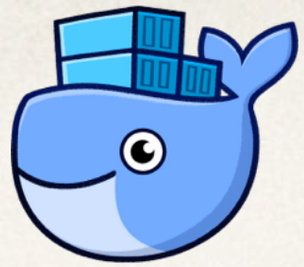
  ```
  $ docker run --name wordpressdb -e MYSQL_ROOT_PASSWORD=password -e
  MYSQL_DATABASE=wordpress -d mysql:5.7
  ```

  luego creamos el contenedor para WordPress

  ```
  $ docker run --name wordpress --link wordpressdb:mysql -p 8080:80 -d wordpres
  ```

  Si ejecutamos el comando $docker ps podremos ver cómo nuestros dos contenedores están activos y corriendo en nuestro entorno, y para ver nuestro WordPress sólo debemos ir en nuestro navegador a http://localhost:8080 y veremos nuestro nuevo WordPress.

Docker y WordPress

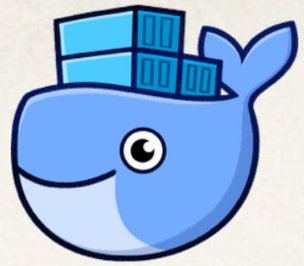

&


- Ahora bien, ya sabemos cómo ejecutar éstos entornos directamente con imágenes de Docker, ahora crearemos un entorno para nuestro desarrollo, para ello usaremos el docker-compose.

- ¿Qué es el docker-compose?

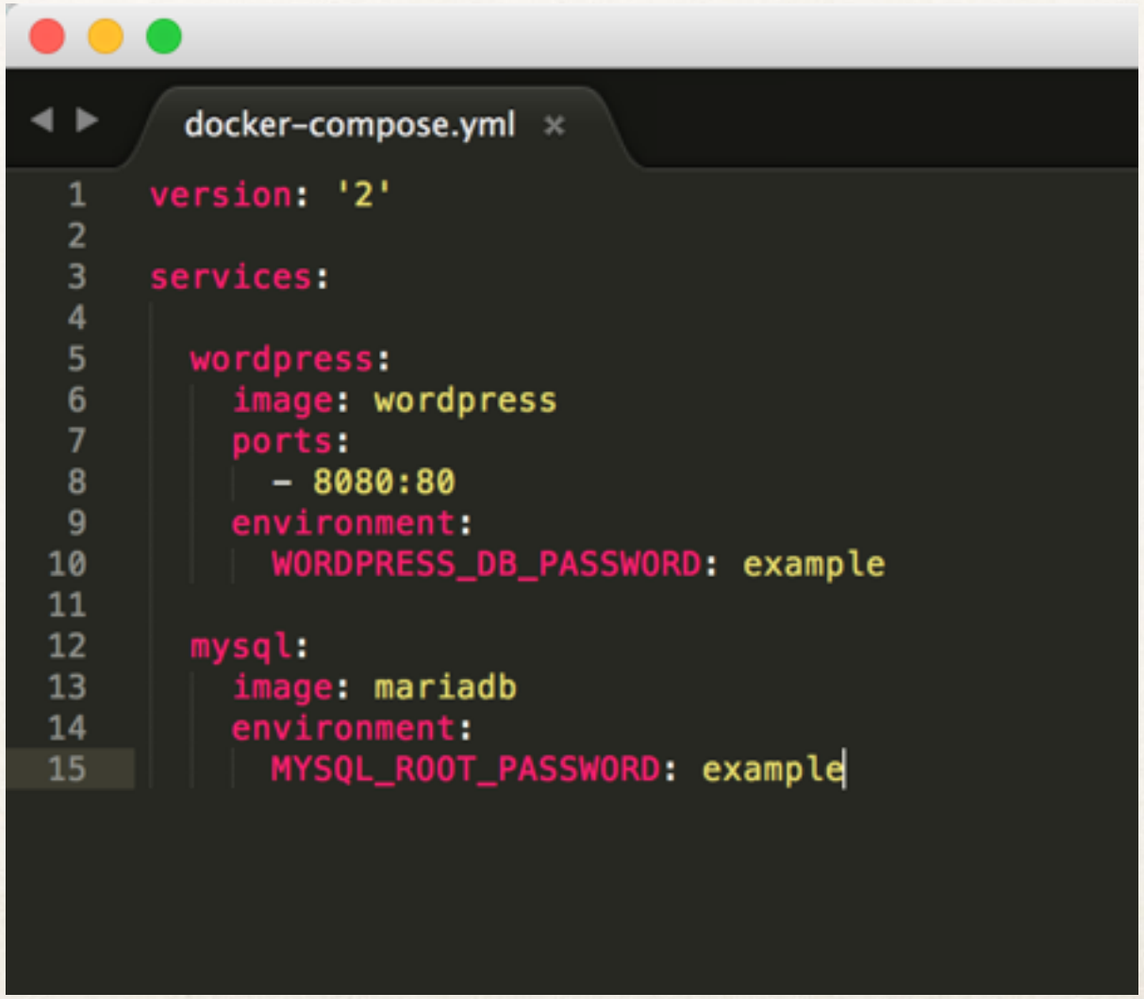
  Compose es una herramienta para la definición y ejecución de aplicaciones multi-contenedores de Docker.

Docker y WordPress



&

- Con el docker-compose podremos crear un entorno adecuado a nuestro desarrollo WordPress, para ello tenemos que crear una carpeta donde guardaremos nuestro archivos y ejecutar el archivo docker-compose.yml donde escribiremos los comandos necesarios para ejecutar las imágenes que necesitamos.



docker-compose.yml

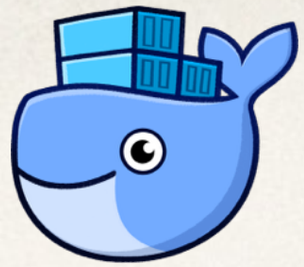
```yaml
version: '2'

services:

  wordpress:
    image: wordpress
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_PASSWORD: example

  mysql:
    image: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: example
```

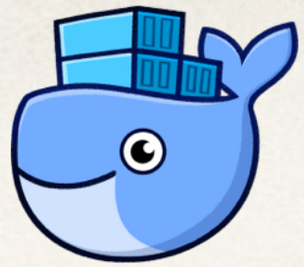
Docker y WordPress



&

- Veamos qué pasa al ejecutar $ docker-compose up

Docker y WordPress



&

- Ahora crearemos un docker-compose.yml más extenso, agregando los directorios que necesitamos para trabajar y datos que necesitaríamos en nuestra BBDD. Para ello editamos el archivo yml de la siguiente manera.

docker-compose.yml

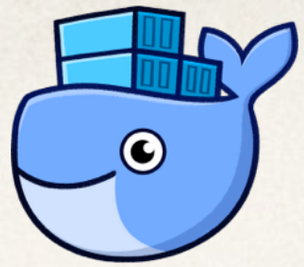
```yaml
version: '2'

services:

  wordpress:
    image: wordpress
    ports:
      - 8080:80
    working_dir: /var/www/html
    volumes:
      - ./wp-content/uploads:/var/www/html/wp-content/uploads
      - ./wp-content/themes/eric:/var/www/html/wp-content/themes/eric
      - ./wp-content/plugins:/var/www/html/wp-content/plugins
    environment:
      WORDPRESS_DB_NAME: nombredelabd
      WORDPRESS_DB_PASSWORD: example

  mysql:
    image: mariadb
    mem_limit: 256m
    container_name: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: nombredeladb
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: example
```

Docker y WordPress



&

- Ya tenemos unos directorios para nuestros archivos uploads, nuestro temas y nuestros plugins

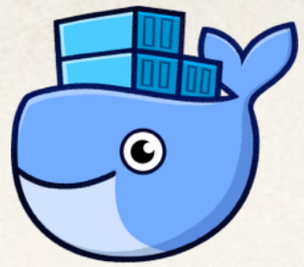
- Ahora veamos como trabajar con la BBDD, si queremos importarla o exportarla, es realmente muy sencillo.

Podemos hacer un dump o montar nuestros datos directamente a la imagen con unos simples códigos de ejecución

```
$ docker exec -i mariadb mysqldump -u root -p'example' nombredelabd > destinationfile.sql
```

```
$ docker exec -i mariadb mysql -u root -p'example' nombredelabd < destinationfile.sql
```

Docker y WordPress



&

- ¿Cómo creamos nuestro Primer ISO para WordPress?

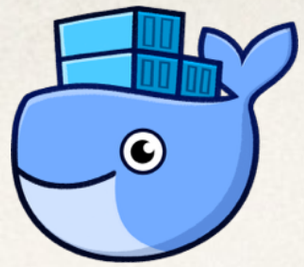
  Algo muy diferente a nuestro entorno de desarrollo es crear nuestra imagen ISO para colocar por ejemplo nuestra web en un entorno asegurado. ¿Cuando haremos esto?. Cuando ya queramos colocar nuestra web en producción en un alojamiento web para Dockers, o cuando queramos imitar un entorno virtual de un servidor.

  Lo primero que debemos hacer es correr nuestro Dockers en el entorno que queramos usar, ya sea Linux o Debian. En nuestro caso usaremos Debian

  $ docker run -i -t debian /bin/bash

  Con éste comando le estamos diciendo a Dockers que nos abra un entorno con Debian para trabajar.

Docker y WordPress



&

- Ya dentro de nuestro entorno con Debian, podremos instalar la configuración de Apache y PHP que mas nos convenga. Een nuestro caso apache2 con php5 y mysql.

  $ apt-get update && apt-get install apache2 -y && apt-get install nano -y && apt-get install mysql-server -y && apt-get install php5 php-pear php5-mysql php5-gd -y

  Lanzamos nuestro servicio mysql con

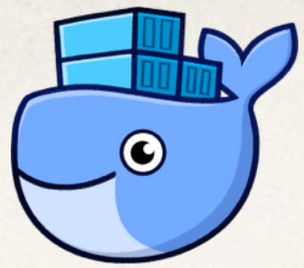
  # service mysql start

  Una vez lanzado procedemos a entrar en el

  # mysql -u root

  Nos debe aparecer el prompt de mysql >

Docker y WordPress



&

Creamos nuestra BBDD para nuestro Wordpress un Usuario y le asignamos un password; luego le damos los permisos necesarios

mysql > CREATE DATABASE wpdb;

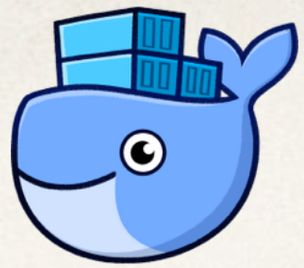
mysql > CREATE USER wpuser@localhost;

mysql > SET PASSWORD FOR wpuser@localhost= PASSWORD('wppass');

mysql > GRANT ALL PRIVILEGES ON wpdb.* TO wpuser@localhost IDENTIFIED BY ‘wppass’;

mysql > FLUSH PRIVILEGES;

Docker y WordPress



&

Creamos nuestra BBDD para nuestro Wordpress un Usuario y le asignamos un password; luego le damos los permisos necesarios

mysql > CREATE DATABASE wpdb;

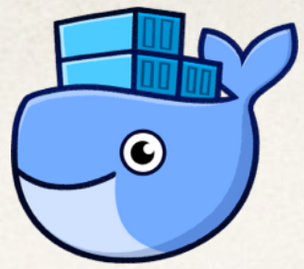
mysql > CREATE USER wpuser@localhost;

mysql > SET PASSWORD FOR wpuser@localhost= PASSWORD('wppass');

mysql > GRANT ALL PRIVILEGES ON wpdb.* TO wpuser@localhost IDENTIFIED BY ‘wppass’;

mysql > FLUSH PRIVILEGES;

Docker y WordPress



&

Una vez realizados éstos pasos tendremos nuestra BBDD llamada wpdb con nuestro usuario wpuser y su password wppass

Bajamos nuestro Wordpress, para ello primero nos instalamos el wget

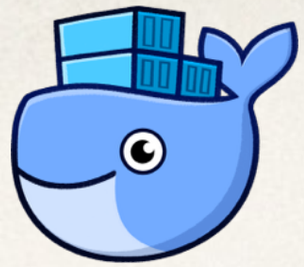
```
# apt-get install wget -y
```

y una vez instalado nos bajamos la ultima versión que queramos instalar

```
# wget http://wordpress.org/latest.tar.gz –no-check-certificate
```

Importante al bajar el WordPress colocar –no-check-certificate de lo contrario no realizará el wget

Docker y WordPress



&

Una vez lanzado procedemos a descomprimir el archivo

tar -xzvf latest.tar.gz

Se nos debe haber creado una carpeta llamada wordpress

cd /wordpress y copiamos # cp wp-config-sample.php wp-config.php

Editamos los datos de la BBDD, luego guardamos y nos vamos a la raíz

Una vez en la raíz copiamos el contenido de la carpeta a /var/www/html

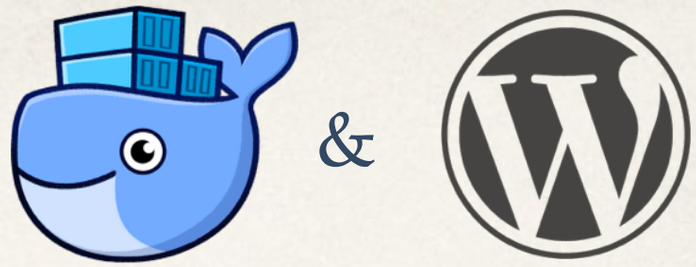
cp -r wordpress/* /var/www/html/

Le asignamos el usuario www-data y el grupo www-data

chown www-data:www-data /var/www/html/* -R

usermod -a -G www-data root

Docker y WordPress



Por último cambiamos o eliminamos el archivo de apache por defecto index.html

mv /var/www/html/index.html /var/www/html/index.html.orig

Eliminamos los archivos que ya no necesitamos para evitar que nuestro contenedor pese mucho

cd /

rm latest.tar.gz

rm -r wordpress

Ahora como debemos lanzar dos servicios Apache2 y mysql, debemos crear un script que nos ejecute los dos y un supervisor; nos instalamos el supervisor

apt-get install supervisor -y

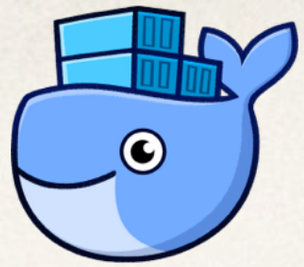
Nos vamos a la raíz y escribimos

nano run.sh una vez dentro agregamos las siguientes lineas y guardamos

#!/bin/bash

exec supervisord -n

Docker y WordPress



&

nano start-apache2.sh una vez dentro agregamos las siguientes lineas y guardamos

```
#!/bin/bash

source /etc/apache2/envvars
exec apache2 -D FOREGROUND
```

nano start-mysqld.sh una vez dentro agregamos las siguientes líneas y guardamos

```
#!/bin/bash

exec mysqld_safe
```

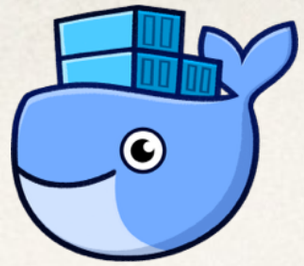
Luego damos permisos a estos tres archivos

```
# chmod 777 run.sh

# chmod 777 start-apache2.sh

# chmod 777 start-mysqld.sh
```

Docker y WordPress



&

Nos vamos a la configuración del supervisor

cd /etc/supervisor/conf.d/

nano supervisor-apache2.conf

[program:apache2]

command=/start-apache2.sh

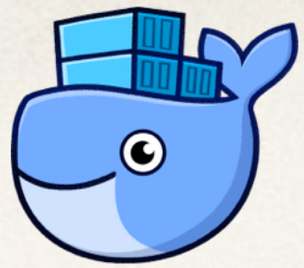
numprocs=1

autostart=true

autorestart=true

autorestart=true

Docker y WordPress



&

nano supervisord-mysqld.conf

[program:mysqld]

command=/start-mysqld.sh

numprocs=1

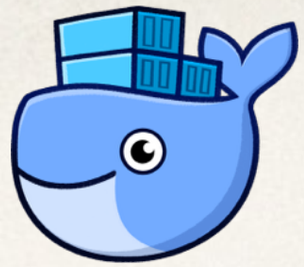
autostart=true

autorestart=true

Damos permisos de ejecución a estos archivos con chmod +x

Listo ahora salimos del contenedor con ctrl p + q

Docker y WordPress



&

Con el comando docker ps veremos nuestro contenedor y el nombre que Dockers le ha asignado, este es el nombre que necesitaremos para hacer el commit del contenedor

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

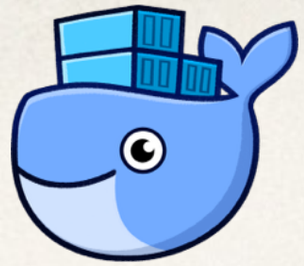
Con este nombre procedemos a realizar nuestro commit y asignaremos un nombre al mismo

$ docker commit nombre_asignado wordpress_01

Ya podemos proceder a detener el contenedor actual y ejecutar el de nuestro commit asignando el puerto donde veremos nuestra web

$ docker run -d -p 8080:80 wordpress_01 /run.sh

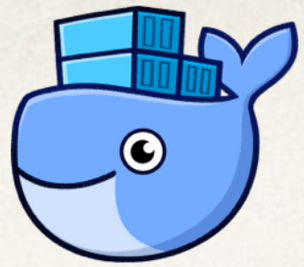
Docker y WordPress


&


- En Conclusión:

  Dockers nos da un mundo de posibilidades, podemos crear entornos a nuestra medida, modificarlos y trabajar con ellos usando los mínimos recursos necesarios para nuestras webs. Lo que nos queda, sería comenzar a jugar con los contenedores, crear imágenes y desarrollar; existe un mundo de información en la web y cada vez son mas las empresas que usan docker por la seguridad que ofrece y la facilidad de crear máquinas virtuales para trabajar o bien para montar el proyecto en línea.

Docker y WordPress



&

GRACIAS

@ericjanzei

http://zeidan.info