Manual de Docker

Alejo Morell Bethencourt

Índice

Índice de imágenes

Introducción

Lo primero sería explicar, que es Docker y como funciona.

Docker es una plataforma abierta enfocada al desarrollo y despliegue de aplicaciones en sistemas paravirtualizados independientes unos de otros.

Esto se traduce en, tener una infraestructura más flexible, ya que en caso de fallo de una de las aplicaciones no todo el servicio se ve afectado, y a la hora de la administración permite aislar el fallo y solucionarlo de forma mas eficiente.

Docker es una aplicación de cliente-servidor y se compone de 3 niveles:

- El Daemon, que se ejecuta nada más iniciar la máquina, es el proceso que esta pendiente de las peticiones, y es el núcleo del programa.
- La API, que especifica la forma en la que los programas se comunican con el Daemon.
- El CLI, que es la interfaz de línea de comandos con la que nosotros, los usuarios nos comunicamos con el daemon de Docker.

Ahora dejaremos de hablar de Docker únicamente como programa y hablaremos de Docker como ecosistema, el conjunto de programa y los objetos que lo componen:

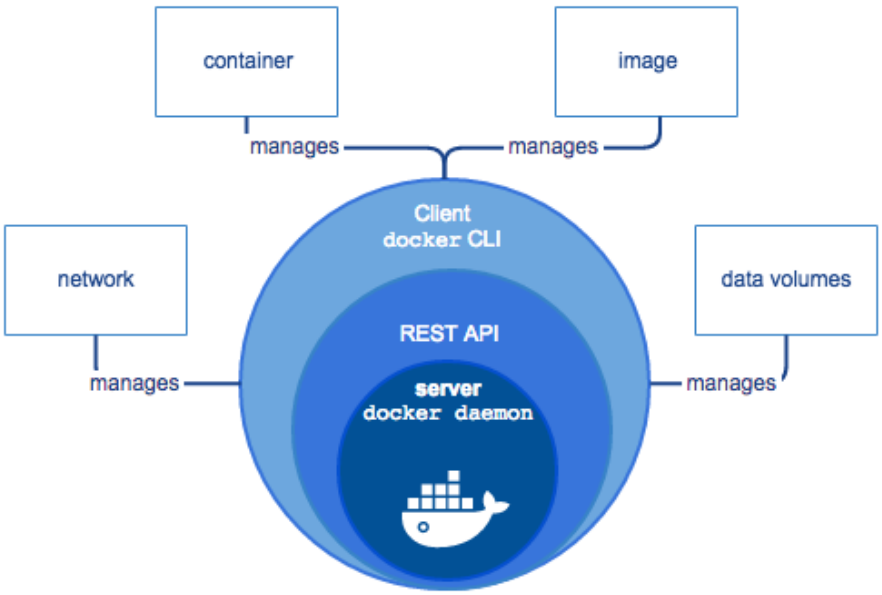

Figura 1: Estructura de Docker

Por un lado tenemos que Docker funciona con elementos llamados contenedores, estos contenedores no son mas que una instancia virtualizada (SO, Aplicación, Programa...).

Y por otro lado tenemos que Docker proporciona la posibilidad de hacer plantillas de esos contenedores, también designadas como imágenes. Estas pueden ser almacenadas en bases de datos como pueden ser docker.hub o los Azure y Google container registries para compartirlas. En ellas encontramos muchísimos servicios preparados para ser importados y configurados en cuestión de segundos.

No solo eso, Docker también ofrece una forma de desplegar varios contenedores a la vez, asignar una relación entre los puertos del host y el contenedor, a la que el host puede acceder, por ejemplo a un servicio Apache a través del navegador. Y relacionar determinados volúmenes de la maquina virtualizada con los del host. Pudiendo por ejemplo tener un contenedor de MySQL en el que "/var/lib/mysql" aparezca como una carpeta con un nombre elegido por nosotros en nuestro sistema anfitrión.

Esto nos posibilita el desplegar aplicaciones de forma rápida y sencilla, fácilmente escalables y administrables, también permite disponer de varias versiones de un mismo contenedor. Por ejemplo tener un contenedor de PHP7 y otro con PHP5, funcionando a la vez. Además de aportar seguridad comparándolo con el modelo de virtualización convencional.

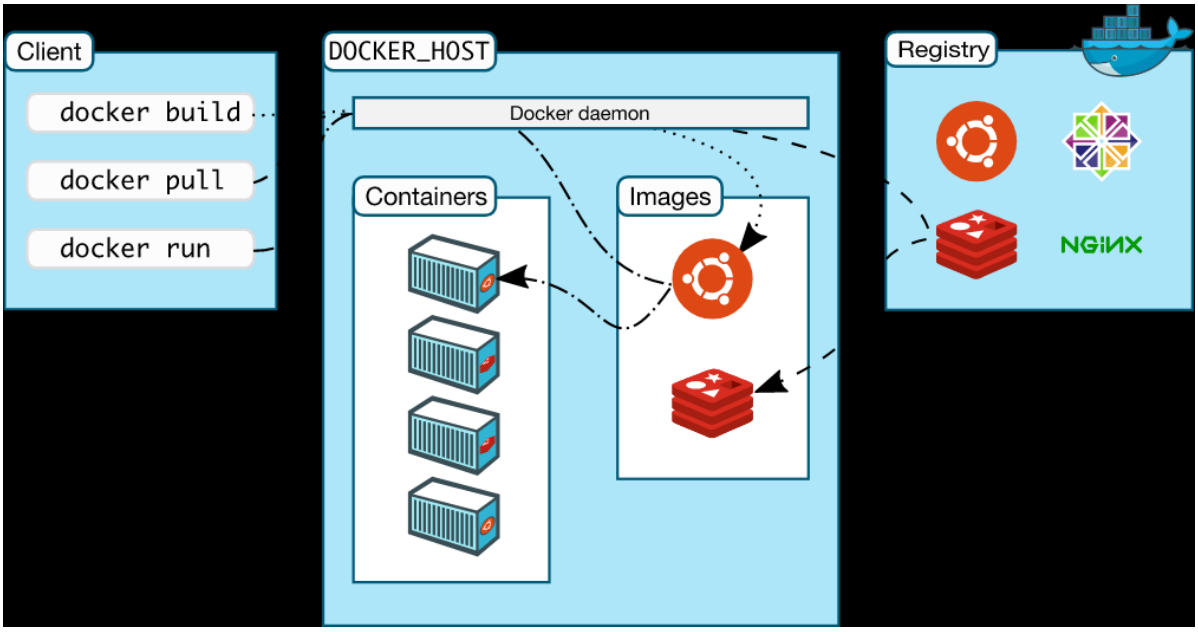

Figura 2: Esquema funcional de Docker

Instalación

Docker puede ser instalado en los 3 sistemas con más cuota en el ecosistema de PC. Windows, Linux y MacOS. A continuación se mostrará como se procede a la instalación en Windows y Ubuntu.

Windows

Requisitos Windows:

- Tener la visualización del procesador activada en la BIOS
- 4 GB de RAM
- Procesador de 64 bits con tecnología SLAT
- Hyper-V y Contenedores activados en características de Windows
- WSL2 (Windows Subsystem Linux), necesario en algunos casos

Proceso de instalación:

1. Descargamos Docker desktop y ejecutamos el instalador. Lo puedes encontrar aquí.
2. Asegurate que cuando aparezca la opción de habilitar Hyper-v esté seleccionada.
3. Sigue las instrucciones del instaladaro hasta finalizar.
4. Cuando el instalador finalice, ciérralo.

Para comprobar que la instalación ha sido correcta trata de buscar Docker desktop en inicio o ejecutando “ `docker -v` ” en la consola.

Ubuntu

Requisitos Ubuntu:

- Versión 16.04 o superior de 64 bits

Proceso de instalación:

1. Eliminar un posible docker instalado previamente, utilizando:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

2. Actualizar la lista de repositorios con:

```
sudo apt update
```

3. Instalar los paquetes que permitan la instalación usando un repositorio mediante HTTPS:

```
sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common
```

4. Agregar la clave GPG de Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

5. Asegurarse de que la salida de el siguiente comando coincide con esta huella “9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88”:

```
sudo apt-key fingerprint 0EBFCD88
```

6. Agregar el repositorio (para la arquitectura amd64, en caso de querer otra remplazar la variable arch, por la nuestra):

```
sudo add-apt-repository \
deb [arch=amd64] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) \
stable"
```

7. Volver a actualizar la lista de repositorios y proceder a la instalación:

```
sudo apt update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

8. Comprobar que la instalación ha sido correcta:

```
docker -v
```

Como utilizar Docker?

Vamos a tratar de entender una serie de conceptos antes de explicar los comandos y para que sirve cada uno.

Como hemos explicado anteriormente Docker funciona con imágenes por lo que el primer paso para comenzar es el importar una, luego definiremos los parámetros necesarios para que las características de la imagen se adecúen a nuestra necesidad específica, por ejemplo, si descargamos una imagen de un programa gestor de bases de datos definir al menos la contraseña de la base de datos.

Esto se puede hacer o bien por medio de la propia linea de comandos o ejecutando Docker a través de un archivo con los parámetros del inicio del contenedor. Una vez el contenedor esta en ejecución, no deja de ser un sistema basado en Linux al cual podemos acceder e interactuar con él. Así que podemos editar archivos de configuración desde el propio contenedor, aunque de querer cambios permanentes no es la opción mas recomendable.

Una vez tenemos el contenedor en funcionamiento podemos, crear una imagen a partir de él, pararlo, crear un commit del contenedor por si vamos a probar algo arriesgado no perder el contenedor original o exportar el contenido del contenedor.

Lo más recomendable a la hora de trabajar con Docker es usar Dockerfiles, en ellas definiremos las variables de entorno de los contenedores, así como las instrucciones, puertos y volúmenes de cada contenedor. Es por así decirlo una especie de script de instalación por llamarlo de forma amigable.

```
# Utiliza la imagen de node con este tag
FROM node:current-slim

# Define el directorio de trabajo
WORKDIR /usr/src/app

# Copia este archivo del host al WORKDIR
COPY package.json .

# Ejecuta este comando dentro del contenedor
RUN npm install

# Define en que puerto esta escuchando el contenedor
EXPOSE 8080

# Ejecuta el comando siguiente
CMD [ "npm", "start" ]

# Copia el codigo de tu aplicación en el WORKDIR
COPY . .
```

Figura 3: Dockerfile, ejemplo de instalación de aplicación Nodejs

La razón por la que queremos utilizar Dockerfile es porque en caso de necesidad de replicar un contenedor, con este archivo y una linea en la consola de comandos tenemos el contenedor funcionando, claro está podemos copiar todo el contenido del contenedor crear un contenedor nuevo y pegarlo en él, pero sabiendo que existen formas mas eficientes es ridículo querer hacer las cosas mas

difíciles de lo que son. Y eso es lo que conseguimos con estos archivos, poder replicar el contenedor sin apenas esfuerzo.

Ahora bien para ejecutarlos tendremos que situarnos en el directorio donde se encuentre el archivo y ejecutar un “ `docker build --tag nombreimagen:1.0` (version) ”, con esto crearemos la imagen con las instrucciones del Dockerfile y para proceder a la ejecución del contenedor utilizaremos algo parecido a “ `docker run --name nombrecontenedor nombreimagen` ”

Desplegar aplicaciones

Ahora que sabemos como crear contenedores de uno en uno, cabe la posibilidad de necesitar crear más de un contenedor. Esto se hace con la instrucción “ `docker-compose` “ y un archivo con el nombre “docker-compose.yml” desde el cual definiremos el nombre de los diversos contenedores, así como sus versiones, asignaciones de puertos y de volúmenes.

Docker compose puede funcionar junto con Dockerfile para, por poner un ejemplo, desplegar un servidor AMP con un único comando en la consola.

Tutorial despliegue AMP

La pila AMP son las siglas de Apache, MySQL y PHP. Estos dos servicios junto con PHP son ampliamente utilizados a lo largo y ancho de internet. Por ejemplo, Wordpress o Moodle, pueden funcionar con este ecosistema. A continuación explicaremos paso a paso el despliegue de una aplicación de forma práctica:

Preparación

Primero tendremos que crear un archivo “docker-compose.yml” y lo editaremos de forma que tengamos 3 imágenes, una de Apache+PHP, una de phpMyAdmin y otra de MySQL conectadas.

Los archivos comienzan por la versión del compose con la sintaxis, “ `version: 'x,x'` “, la versión de docker-compose utilizada y siguen con un “ `services:` “ dentro del cual con tabulaciones definiremos la estructura de nuestros servicios. Como veremos a continuación:

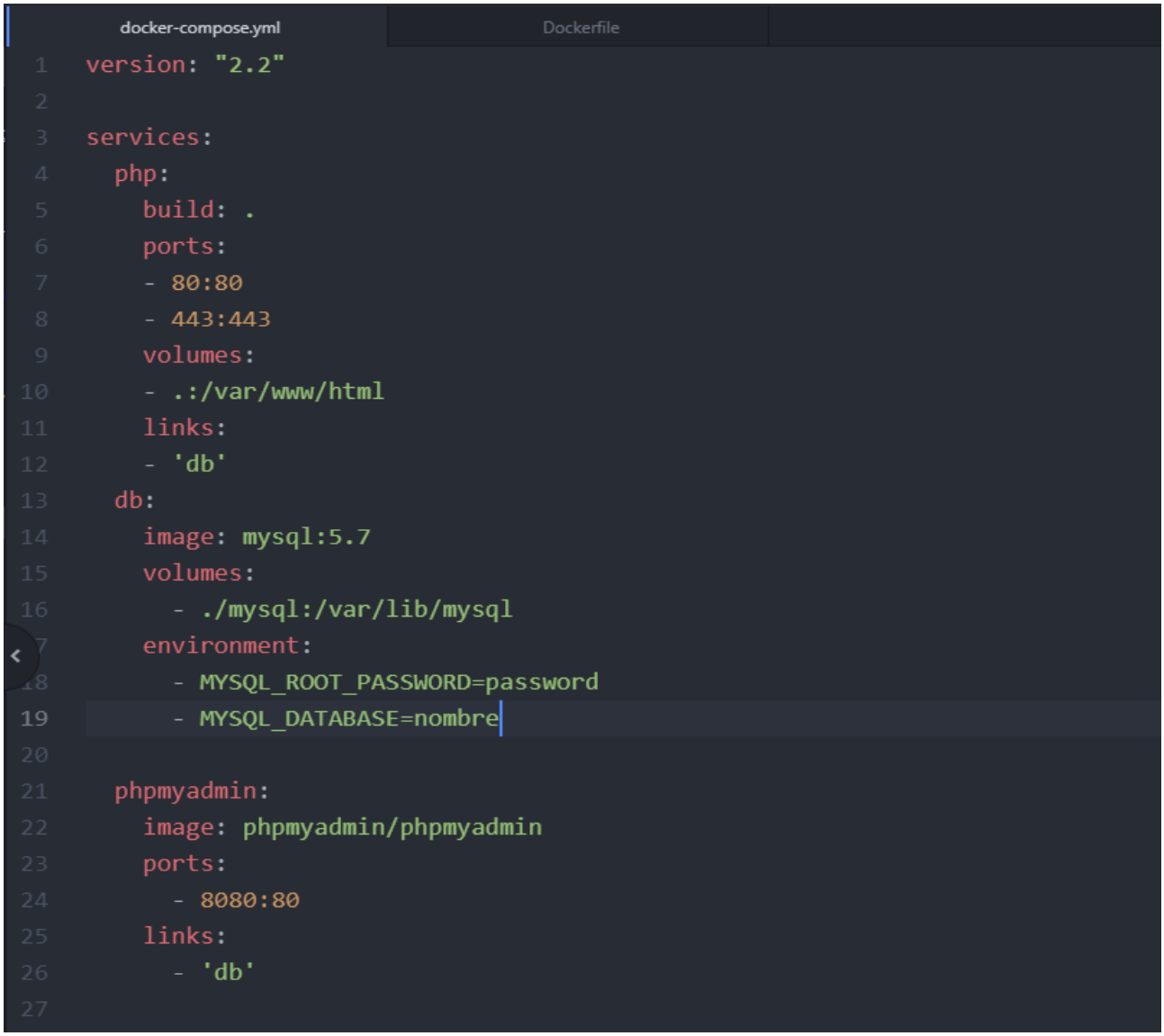

Figura 4: Ejemplo docker-compose.yml

Como podemos observar tras cada salto de línea precedido por un “ : ” al que no le pasamos un parámetro viene una tabulación. Podemos ver que estamos instalando PHP sin definir la versión, MySQL con la versión 5.7 y phpMyAdmin para gestionar la base de datos.

La imagen de PHP viene acompañada de un Dockerfile, en la cual definimos que imagen de PHP queremos, en este caso una con apache incorporado. Por eso en la imagen anterior hemos asignado los puertos web a esta imagen, así como la ruta de publicación web de Apache por defecto, con nuestra ruta actual:

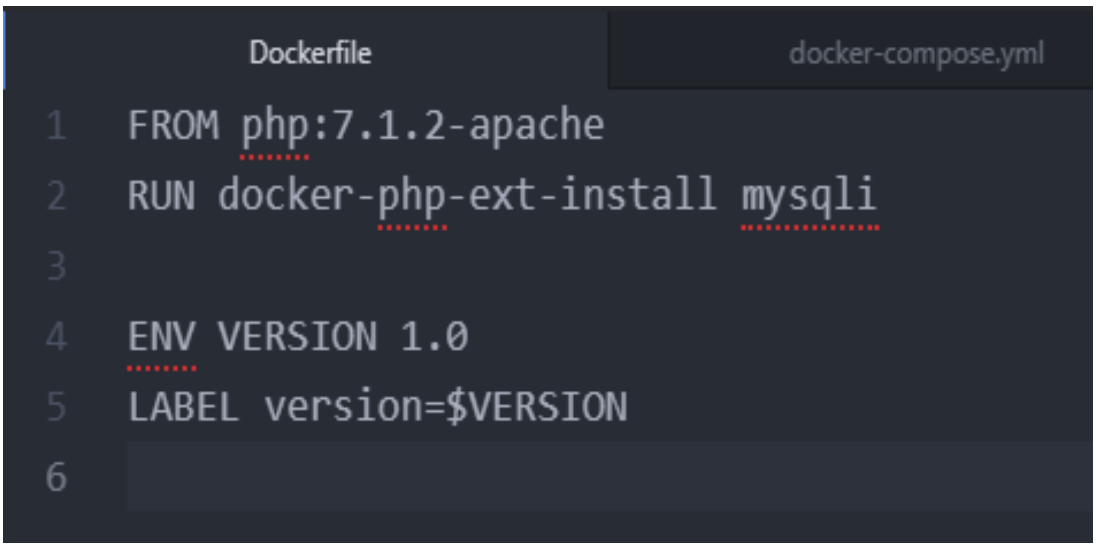

Figura 5: Dockerfile PHP+Apache

Volviendo a la Figura 4, en ella podemos ver que en los parámetros `MYSQL_ROOT_PASSWORD` y `MYSQL_DATABASE` estamos definiendo el nombre de nuestra base de datos, así como la contraseña del root de MySQL. Y relacionando una carpeta llamada mysql que aparecerá en nuestro directorio actual con el directorio donde MySQL almacenará la información de la base de datos.

Y por último phpmyadmin, donde podremos gestionar via interfaz web la base de datos de ahí que asignemos el puerto 8080 al 80 del contenedor.

Despliegue

Como podemos observar tengo aquí mis dos archivos creados con el texto de las imagenes:

```
PS C:\Users\Yuki\Desktop\Manualdocker> ls


    Directorio: C:\Users\Yuki\Desktop\Manualdocker


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        06/11/2020     20:01            475 docker-compose.yml
-a----        06/11/2020     20:00             99 Dockerfile
```

Figura 6: Comprobación archivos creados

Desplegaremos los contenedores tras ejecutar, en la ruta donde estén los archivos docker:

`docker-compose up`

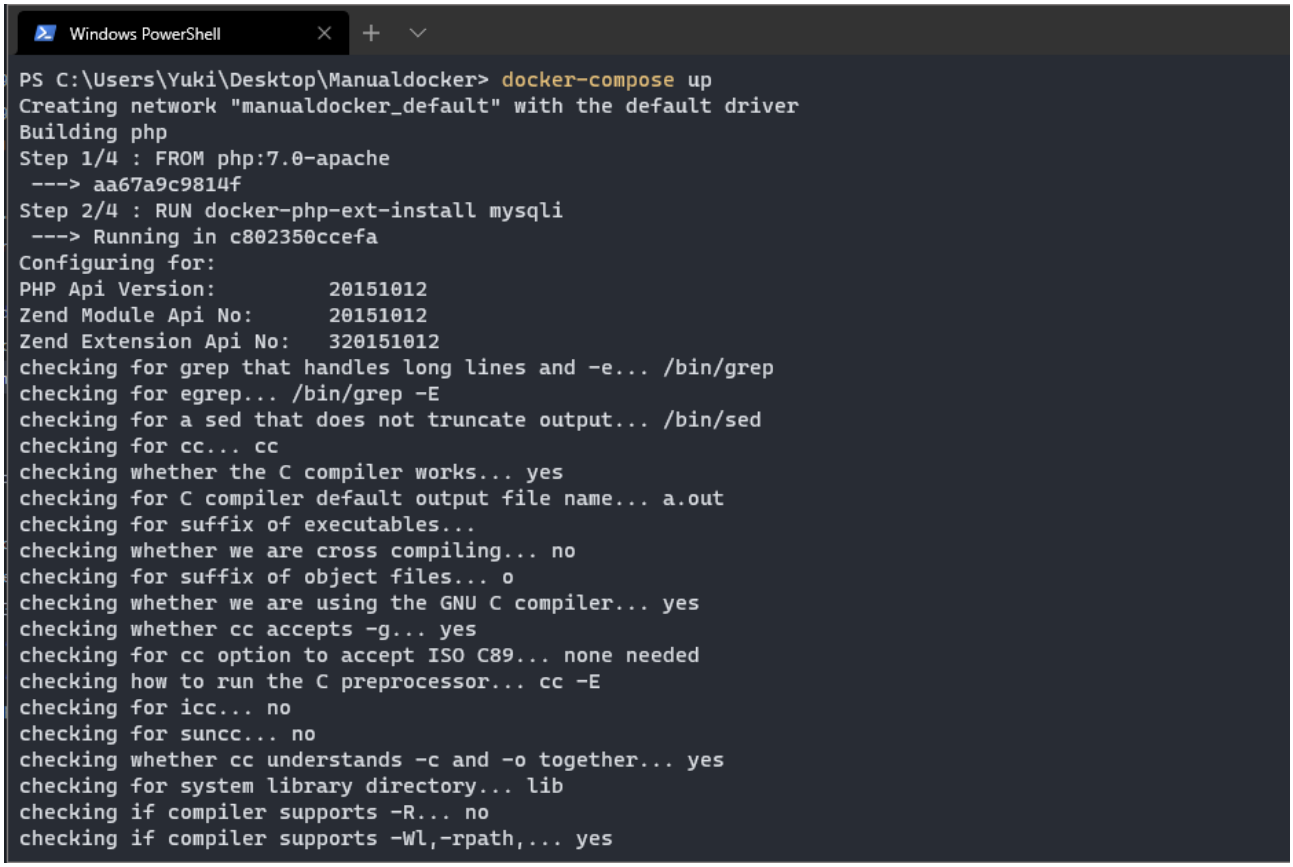

Figura 7: Comando docker-compose up

Utilizando el comando “ `docker ps` ” se mostrará un listado de los contenedores:

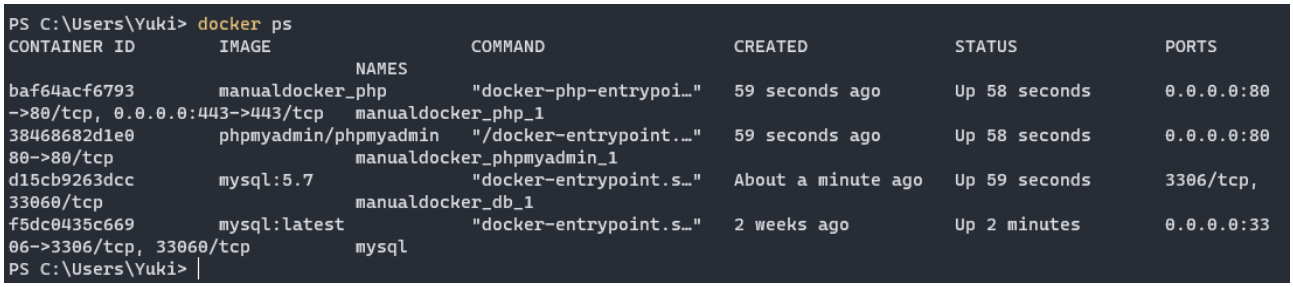

Figura 8: Listar contenedores

Podremos comprobar que funciona intentando acceder por ejemplo a localhost:8080, donde introduciendo el usuario y contraseña de la base de datos

podremos acceder. Y en caso de probar la web, creando un archivo index con extensión php, accediendo por el puerto 80, comprobaremos que funciona.

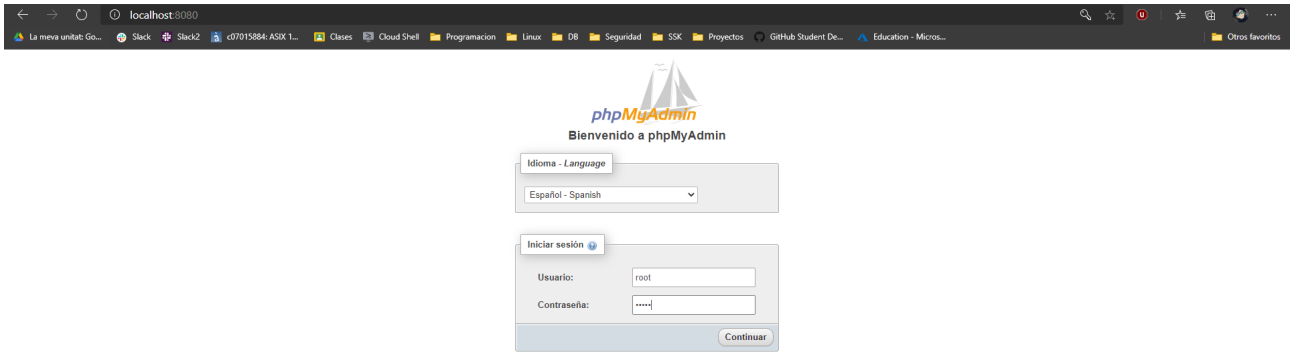

Figura 9: Panel de PhpMyAdmin

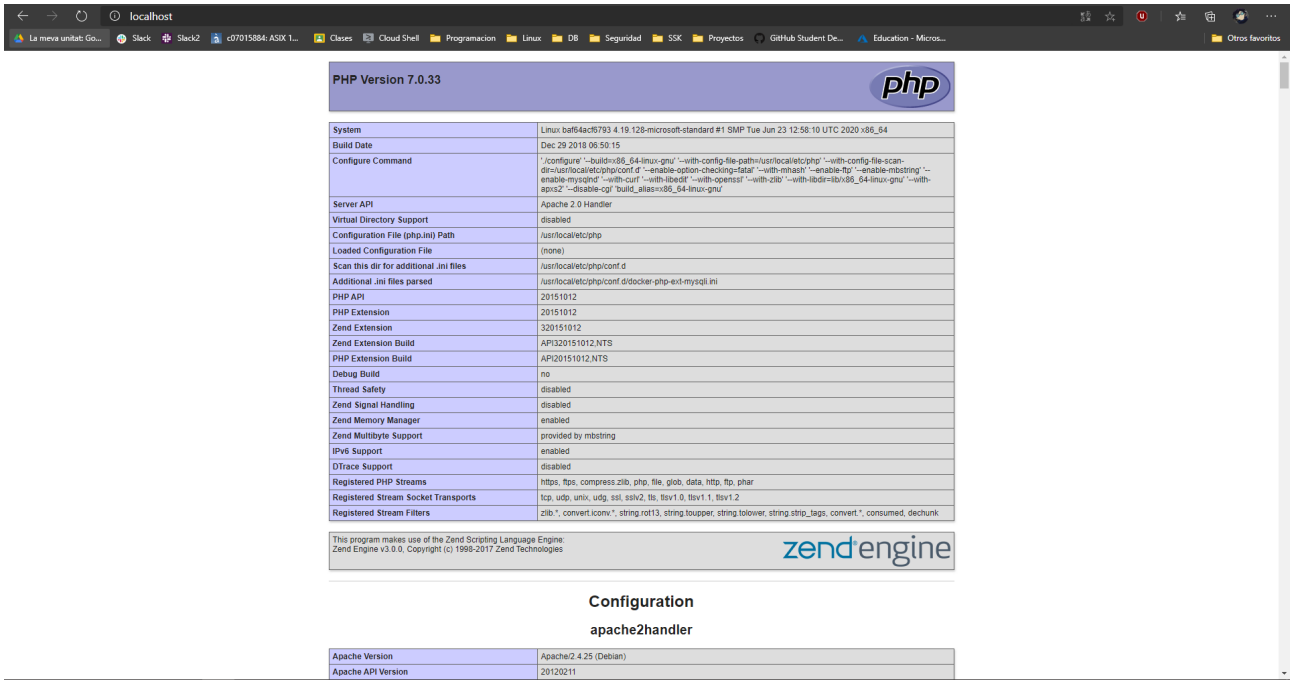

Figura 10: Index.php creado

Chuleta de comandos

Comando	Descripción
docker pull imagen	Descarga la imagen de la BBDD
docker run –name contenedor imagen -p hport:gport -v hroute:groute	Inicia la imagen, asignando relación entre puertos y volúmenes
docker ps	Muestra la lista de contenedores
docker image ls	Muestra la lista de imágenes almacenadas
docker exec -it contenedor comando (ej. bash)	Ejecutar comandos dentro del contenedor
docker build –tag imagen:1.0 .	Crear imagen a partir de Dockerfile en el mismo directorio (el “.” referencia eso, puede ser una ruta)
docker export --output="archivo.tar" contenedor	Exportar el contenido de un contenedor en tar
cat archivo.tar | docker import - imagen:etiqueta	Importar un tar como imagen
docker commit contenedor imagen	Crear una imagen de un contenedor
docker start contenedor	Inicia un contenedor pausado
docker stop contenedor	Detiene un contenedor ejecutándose
docker rm -f contenedor	Borra un contenedor este funcionando o no (-f de force)
docker rename OldName NewName	Renombra un contentenedor
docker diff contenedor	Muestra las modificaciones realizadas en un contenedor

docker port contenedor	Muestra los puertos asignados del contenedor
docker rmi imagen	Borra una imagen
Docker save –-output archivo.tar imagen	Guarda una imagen en un tar
Docker load –-input archivo.tar imagen	Carga una imagen desde un tar
docker-compose up	Levanta un conjunto de contenedores de un archivo docker-compose.yml en esa ruta
Docker-compose down	Borra todo lo relacionado con el archivo docker-compose.yml de la ruta

Bibliografía

https://docs.docker.com/get-started/overview/

https://en.wikipedia.org/wiki/Second_Level_Address_Translation

https://docs.microsoft.com/en-us/windows/wsl/install-win10

https://docs.docker.com/engine/install/ubuntu/

https://dockerlabs.collabnix.com/docker/cheatsheet/

https://www.docker.com/sites/default/files/d8/2019-09/docker-cheat-sheet.pdf

https://medium.com/@lizrice/to-commit-or-not-to-commit-5ab72f9a466e

https://hub.docker.com/_/mysql

https://docs.docker.com/get-started/part2/

https://www.teamnet.com.mx/blog/containers-contenedores-inform%C3%A1ticos

https://mediatemple.net/community/products/dv/204643880/how-can-i-create-a-phpinfo.php-page

https://github.com/matheuspiment/amp