



UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingeniería Informática y
Telecomunicaciones

Centro de Procesamiento de Datos

PRÁCTICA 2: CONTENEDORES DOCKER (II)

Doble Grado Ingeniería Informática y Matemáticas

Autor: Clara Bolívar Peláez

Septiembre 2024

Índice

1. Crear una imagen de Docker personalizada con Apache	2
2. Subir la imagen a Docker Hub	3
3. Configurar Wordpress con Docker Compose	3
4. Limitar el uso de CPU y realizar un benchmark	5
4.1. Pruebas	6
4.2. Análisis	7
5. Cuestiones a destacar de la P2	7

1. Crear una imagen de Docker personalizada con Apache

■ Crear el archivo index.html:

Creamos una nueva carpeta para la práctica 2 en la que alojaremos nuestro fichero index.html.

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <title>Práctica2 CPD</title>
  </head>
  <body>
    <header></header>
    <main></main>
    <h1>CPD.P2: Clara Bolivar Pelaez</h1>
    <footer></footer>
  </body>
</html>
```

Este archivo será accedido cuando accedamos a `http://localhost:8888`.

■ Crear el dockerfile:

Dentro de la carpeta que ya creamos para esta práctica creamos el dockerfile, que es un archivo que define los pasos para construir la imagen de Docker.

```
FROM debian
MAINTAINER Usuario CPD "cbolivarpelaez@ugr.es"
RUN apt-get update && apt-get install -y apache2 && apt-get clean
&& rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
EXPOSE 80
ADD ["index.html", "/var/www/html/"]
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

- FROM debian: Se basa en la imagen oficial de Debian.
- MAINTAINER: Define el autor de la imagen.
- RUN apt-get: Instala Apache en el contenedor.
- ADD: Copia tu archivo index.html en el servidor web.
- ENTRYPOINT: Configura Apache para que se ejecute en primer plano.

■ Construir la imagen:

Para construir la imagen nos situamos en la carpeta que creamos y ejecutamos el siguiente comando:

```
docker build -t prueba .
```

```
(base) clara@clara-Modern-14-C12R:~/Desktop/CPD/Prácticas/Práctica 2$ docker build -t prueba .
[+] Building 14.6s (8/8) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> transferring dockerfile: 34B                                  0.0s
=> WARN: MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label (line 2)
=> [internal] load metadata for docker.io/library/debian:latest 1.6s
=> [internal] load .dockerignore                                  0.0s
=> transferring context: 2B                                       0.0s
=> [1/3] FROM docker.io/library/debian:latest@sha256:27586f4609433f2f49a9157485b473c62c3cb28a581c413393975b4 3.9s
=> resolve docker.io/library/debian:latest@sha256:27586f4609433f2f49a9157485b473c62c3cb28a581c413393975b4 0.0s
=> sha256:c225d78f8e80911f18c798d76af1d1b41102b0ab591ff6d40a455c8d84 520B / 520B 0.0s
=> sha256:c7f9867667219411cb7d7ff983c3eadd881d415eae5495ad0116fb5d6db8bb68 1.46kB / 1.46kB 0.0s
=> sha256:cd62b739133c498a16f7a7b1b0555ba43d02b2511c508fa4ca9b1975ffe28e 49.56MB / 49.56MB 2.8s
=> sha256:27586f4609433f2f49a9157485b473c62c3cb28a581c413393975b4a8496d8ab 1.85kB / 1.85kB 0.0s
=> extracting sha256:cd62b739133c498a16f7a7b1b0555ba43d02b2511c508fa4ca9b1975ffe28e 1.8s
=> [internal] load build context                                  0.0s
=> transferring context: 285B                                     0.0s
=> [2/3] RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/* 5.6s
=> [3/3] ADD [index.html,/var/www/html/]                        0.0s
=> exporting image                                              0.3s
=> writing image sha256:cd348ed4ddda6488beed33d4a6723d6d1d828ff2f6e83b115eb2212941e79144 0.0s
=> naming to docker.io/library/prueba                           0.0s

4 warnings found (use docker --debug to expand):
- MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label (line 2)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 5)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 6)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 7)
(base) clara@clara-Modern-14-C12R:~/Desktop/CPD/Prácticas/Práctica 2$
```

docker build construye la imagen a partir del Dockerfile.

■ Ejecutar la imagen:

Creamos el contenedor

```
docker run --name contenedor1 -p 8888:80 -d prueba
```

- **Comprobar:**

Accedemos a localhost:8888 y obtenemos



CPD_P2: Clara Bolívar Peláez

2. Subir la imagen a Docker Hub

- **Iniciar sesión en Docker Hub desde la consola**

```
(base) clara@clara-Modern-14-C12M:~/Desktop/CPD/Prácticas/Práctica 2$ docker login -u cbolivarpelaez
Password:
WARNING! Your password will be stored unencrypted in /home/clara/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
```

- **Subir la imagen a Docker Hub**

```
docker tag prueba cbolivarpelaez/prueba:1.0
```

donde `docker tag` es el comando que se utiliza para asignar una etiqueta a una imagen existente.

```
docker push cbolivarpelaez/prueba:1.0
```

```
(base) clara@clara-Modern-14-C12M:~/Desktop/CPD/Prácticas/Práctica 2$ docker tag prueba cbolivarpelaez/prueba:1.0
(base) clara@clara-Modern-14-C12M:~/Desktop/CPD/Prácticas/Práctica 2$ docker push cbolivarpelaez/prueba:1.0
The push refers to repository [docker.io/cbolivarpelaez/prueba]
69e23b3b9970: Pushed
90098bf1079b: Pushed
d50132f2fe78: Mounted from library/debian
1.0: digest: sha256:7bc793dc7b35cf91b29c90b67c6b686a19432dbaed89b978c9759dc610def653 size: 948
```

3. Configurar Wordpress con Docker Compose

- **Crear el archivo docker-compose.yml**

Creamos dicho archivo y le copiamos el contenido que aparece en el archivo de swad.

```
version: '3.3'

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
```

```
WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
```

El archivo `docker-compose.yml` es un archivo de configuración que permite definir y gestionar aplicaciones compuestas por múltiples contenedores Docker.

- Levantar el entorno

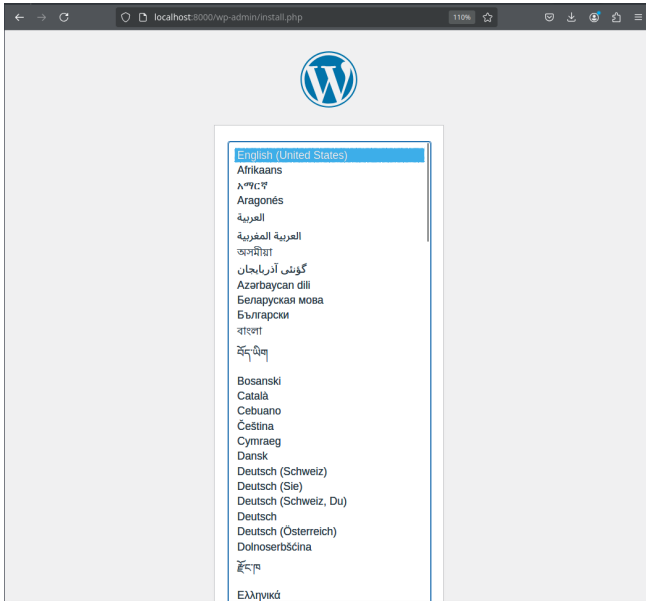
```
docker-compose up -d
```

```
(base) clarc@practica-modern-14-12M:~/desktop/CPD/Practicas/Practica 25 docker-compose up -d
Creating network "practica25_default" with default driver
Creating volume "practica25_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
28e4dc4dc69: Pull complete
1c5de3dc4c74: Pull complete
e9f031c124cc: Pull complete
68c3898c2015: Pull complete
6b95a948e7b6: Pull complete
99f086bb8d6e: Pull complete
e071319c3779: Pull complete
ff08e9df4d8b: Pull complete
43d05e938198: Pull complete
064b2d298fba: Pull complete
df9a4d85569b: Pull complete
Digest: sha256:64db0b36e36ed43453367cae56536f4b8156d78bae830145cbe47c2aad73bb
Status: Downloaded newer image for mysql:5.7
Pulling wordpress (wordpress:latest)...
latest: Pulling from library/wordpress
382e3ee49b05: Pull complete
07fca89b0857: Pull complete
141aa7d58c57: Pull complete
2720d4bca8b3: Pull complete
82deca51468b: Pull complete
dec741df4526: Pull complete
a284b0efab94: Pull complete
0a9b825ee085: Pull complete
ef45e9da2633: Pull complete
f2b46378d521: Pull complete
5c184459ddad: Pull complete
282d0878d4dd: Pull complete
43a1f027cc210: Pull complete
e07327fd499: Pull complete
ce708882c6c6: Pull complete
5135421aa0d8: Pull complete
f2d363b83c2e: Pull complete
0a6df7212f3fd: Pull complete
10dc57629813: Pull complete
7975a0f71e0f: Pull complete
008834247b8e: Pull complete
Digest: sha256:3baff6a246dc1e9fd89b7bf69c758f66421717cf2ac8b7ba314374fa5b6
Status: Downloaded newer image for wordpress:latest
Creating practica25_db_1 ... done
Creating practica25_wordpress_1 ... done
```

Hemos creado dos contenedores Docker que incluyen una base de datos MySQL y una aplicación Wordpress para lo que Docker Compose usa el archivo `docker-compose.yml`

■ Acceder a Wordpress

Accedemos a `localhost:8080` y configuramos Wordpress.



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title

Username
 Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

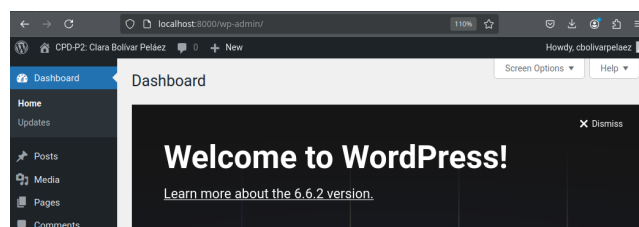
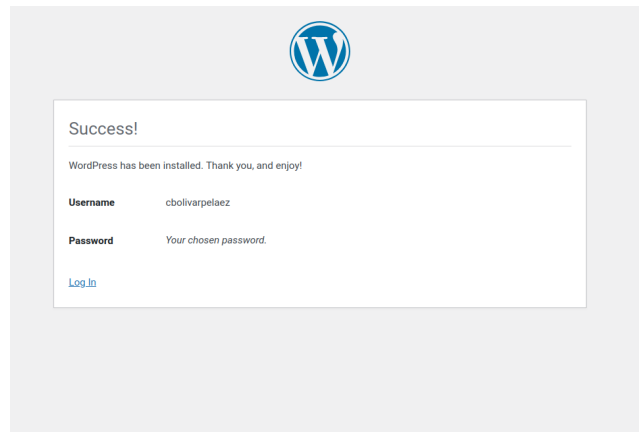
Password [Show](#)
 Strong

Important: You will need this password to log in. Please store it in a secure location.

Your Email
 Double-check your email address before continuing.

Search engine visibility ☐ Discourage search engines from indexing this site
 It is up to search engines to honor this request.

[Install WordPress](#)



4. Limitar el uso de CPU y realizar un benchmark

Los siguientes pasos los haremos en cada una de las pruebas:

- Crear un contenedor de Ubuntu

```
docker run -it --cpus="numero" ubuntu /bin/bash
```

donde **numero** hace referencia a las CPUs.

- Instalar sysbench

```
apt-get update
apt-get install -y sysbench
```

- Ejecutar la prueba de CPU

```
sysbench --test=cpu --cpu-max-prime=20000 run
```

4.1. Pruebas

1. Contenedor con un límite de CPU de la mitad (0.5)

```
Threads started!

CPU speed:
  events per second:   579.33

General statistics:
  total time:          10.0074s
  total number of events: 5800

Latency (ms):
  min:                 0.67
  avg:                 1.72
  max:                 57.15
  95th percentile:    1.93
  sum:                 10002.91

Threads fairness:
  events (avg/stddev):  5800.0000/0.00
  execution time (avg/stddev): 10.0029/0.00

root@16c1fa408e7a:/#
```

2. Contenedor con una CPU completa (1)

```
Threads started!

CPU speed:
  events per second:  1187.34

General statistics:
  total time:          10.0003s
  total number of events: 11875

Latency (ms):
  min:                 0.67
  avg:                 0.84
  max:                 3.35
  95th percentile:    1.39
  sum:                 9997.29

Threads fairness:
  events (avg/stddev):  11875.0000/0.00
  execution time (avg/stddev): 9.9973/0.00

root@8f7601376c61:/#
```

3. Contenedor con un límite de dos CPUs (2)

```
Threads started!

CPU speed:
  events per second: 1410.61

General statistics:
  total time:          10.0004s
  total number of events: 14108

Latency (ms):
  min:                0.64
  avg:                0.71
  max:                2.29
  95th percentile:   0.80
  sum:                9997.37

Threads fairness:
  events (avg/stddev): 14108.0000/0.00
  execution time (avg/stddev): 9.9974/0.00

root@3d2bd41cf121:/#
```

4.2. Análisis

- **Tiempo total de ejecución:**
 - a) *Prueba 1:* 10.0074s
 - b) *Prueba 2:* 10.003s
 - c) *Prueba 3:* 10.004s
- **Tasa de operaciones por segundo:**
 - a) *Prueba 1:* $\frac{5800}{10,0074} = 579,5711op/s$
 - b) *Prueba 2:* $\frac{11875}{10,003} = 1187,1438op/s$
 - c) *Prueba 3:* $\frac{14108}{10,004} = 1410,2359op/s$

Podemos ver que a más CPU, menos tiempo de ejecución y más operaciones por segundo.

Aunque en nuestro caso no disminuyamos el tiempo de ejecución al usar 2 CPUs en lugar de una, si que conseguimos aumentar el número de operaciones por segundo.

5. Cuestiones a destacar de la P2

1. Qué es dockerfile y para qué sirve

El Dockerfile es un archivo que contiene un conjunto de instrucciones para crear una imagen de Docker personalizada de manera automatizada. Un ejemplo:

```
FROM debian
MAINTAINER Usuario CPD "usuario@ugr.es"
RUN apt-get update && apt-get install -y apache2
&& apt-get clean && rm -rf /var/lib/apt/lists/*
ENV APACHERUN_USER www-data
ENV APACHERUN_GROUP www-data
ENV APACHELOG_DIR /var/log/apache2
EXPOSE 80
ADD ["index.html", "/var/www/html/"]
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

- **Diferencia entre *add* y *copy*:**
Ambos copian archivos al sistema de archivos del contenedor, pero ADD puede descomprimir archivos automáticamente y permite especificar URL, mientras que COPY solo copia archivos locales sin descomprimir.

- *Qué hace el comando **expose**:*
Declara el puerto en el que el contenedor escuchará por defecto (aunque necesitas también el -p al ejecutar el contenedor).
- *Qué hace **entrypoint** y como difiere de **cmd**:*
ENTRYPOINT define el proceso principal que el contenedor ejecutará, mientras que CMD especifica argumentos por defecto para dicho proceso. CMD puede ser sobrescrito en la línea de comandos, pero ENTRYPOINT no.

2. Qué hace docker build

Se utiliza para crear una imagen de Docker a partir de un archivo Dockerfile. Esta imagen se puede utilizar posteriormente para crear y ejecutar contenedores. El comando procesa las instrucciones definidas en el Dockerfile y genera una imagen que contiene todo lo necesario para ejecutar una aplicación o servicio en un contenedor.

```
docker build -t nombre_imagen .
```

3. Pasos para subir una imagen personalizada a Docker Hub

Creas una cuenta en Docker Hub, usas docker tag para asociar la imagen con tu repositorio, haces docker login y finalmente subes la imagen con docker push.

- *Cuál es la función de **docker tag**:*
docker tag asigna una etiqueta que incluye el nombre del usuario y la imagen, lo que es necesario para poder subir la imagen a Docker Hub.

```
docker tag [imagen-origen] [nombre_destino:tag]
docker tag prueba cbolivarpelaez/prueba:1.0
```

- *Qué ocurre si queremos subir una nueva versión de nuestra imagen a Docker Hub:*
Utilizas etiquetas como 1.0, 1.1, latest, etc., para identificar las distintas versiones de la imagen. Cada versión puede subirse con un comando docker push después de etiquetar la imagen.

```
docker push cbolivarpelaez/prueba:1.0
```

- *Cómo gestionarías la actualización de una aplicación en Docker utilizando Docker Hub y Docker Compose:*
Subirías una nueva versión de la imagen a Docker Hub utilizando una nueva etiqueta (por ejemplo, 1.1), luego actualizarías el archivo docker-compose.yml con esa nueva versión y ejecutarías docker-compose up -d para actualizar los contenedores.

4. Qué es docker compose

Docker Compose permite definir y gestionar múltiples contenedores de manera conjunta utilizando un archivo docker-compose.yml, lo que facilita la creación de entornos complejos que involucren varias aplicaciones.

- *Qué servicios se configuraron en nuestro archivo **docker-compose.yml**:*

```
version: '3.3'

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
```

```
WORDPRESS_DB_USER: wordpress
WORDPRESS_DB_PASSWORD: wordpress
WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
```

Configuramos db (MySQL) y wordpress, que dependen entre sí. El servicio db es la base de datos MySQL y wordpress es el servidor Wordpress que depende de la base de datos.

- *Qué comando utilizas para levantar todos los contenedores definidos en un archivo **docker-compose.yml**:*

```
docker-compose up -d
```

- *Cómo se levantan servicios con **docker-compose**:*

Para levantar servicios con Docker Compose, se utiliza el archivo docker-compose.yml para definir múltiples contenedores o servicios que se ejecutarán juntos. Una vez que el archivo está configurado, puedes levantar los servicios utilizando el comando docker-compose up.

5. Limitando el uso de CPU de los contenedores

- *Qué comando utilizas para limitar el uso de CPU en un contenedor de Docker:*

Utilizas la opción `-cpus`, por ejemplo `docker run --cpus=".5"` para limitar el contenedor a usar un máximo del 50 % de una CPU.

- *Cómo impacta el límite de CPU en los resultados del benchmark de **sysbench**:*

Al reducir la cantidad de CPU disponible, el tiempo de ejecución aumenta y la tasa de operaciones por segundo disminuye, demostrando cómo los recursos limitados afectan el rendimiento del contenedor.