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

- **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`.

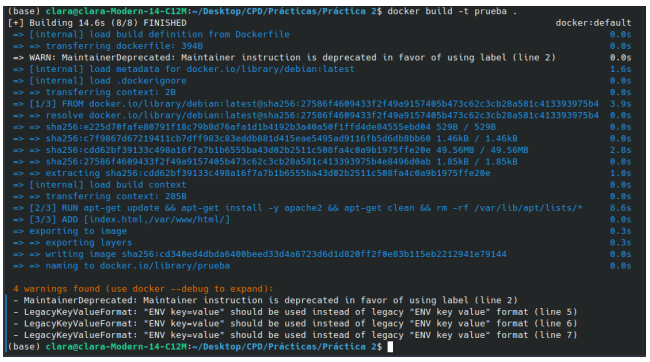
- **Crear el dockerfile:**
  Dentro de la carpeta que ya creamos para esta práctica creamos el dockerfile, que es un archivo que define los pasos para construir la imagen de Docker.

```
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 lacarpeta que creamos y ejecutamos el siguiente comando:

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



  `docker build` construye la imagen a partir del Dockerfile.

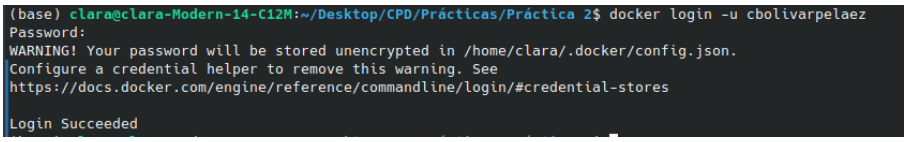
- **Ejecutar la imagen:**
  Creamos el contenedor

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

- **Comprobar:**
  Accedemos a `localhost:8888` y obtenemos

2. Subir la imagen a Docker Hub

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

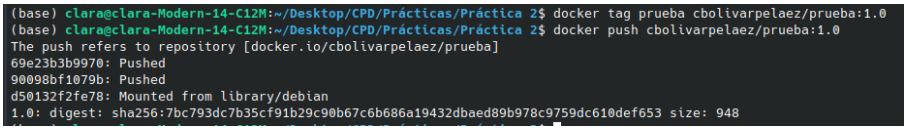


- **Subir la imagen a Docker Hub**

```
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
```



3. Configurar Wordpress con Docker Compose

- **Crear el archivo `docker-compose.ysl`**

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

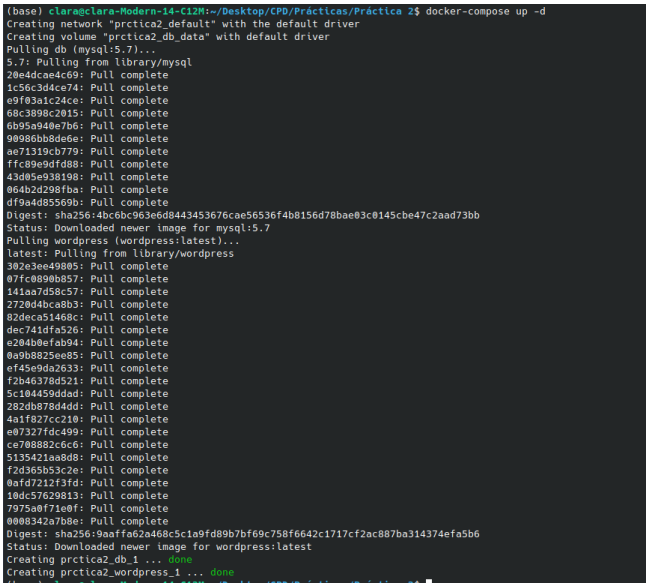
```
version: '3.3'

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**

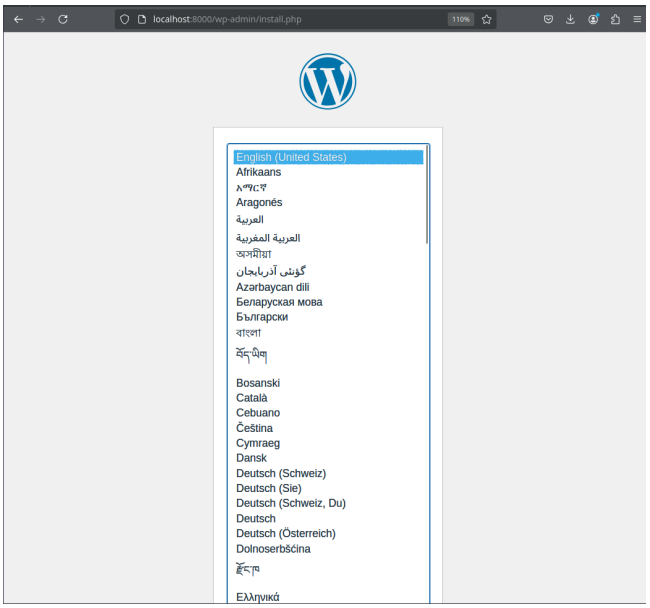
```
docker-compose up -d
```

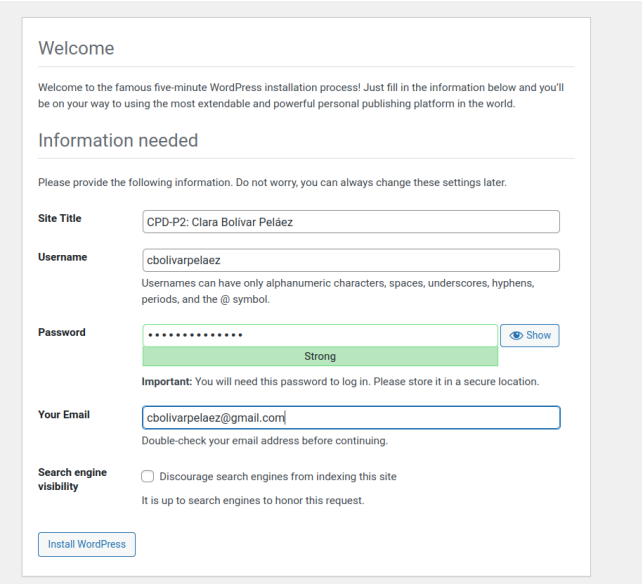


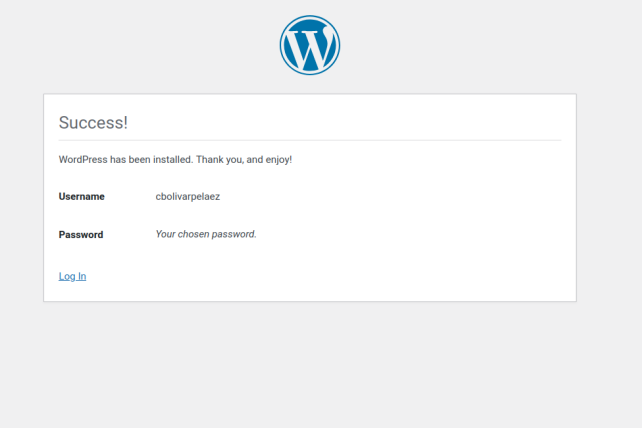
Hemos creado dos contenedores Docker que incluyen una base de datos MySQL y una aplicación Wordpress para lo que Docker Compose usa el archivo `docker-compose.yml`

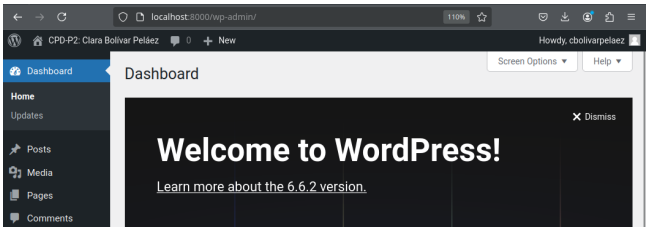
- **Acceder a Wordpress**

Accedemos a `localhost:8080` y configuramos Wordpress.









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`**

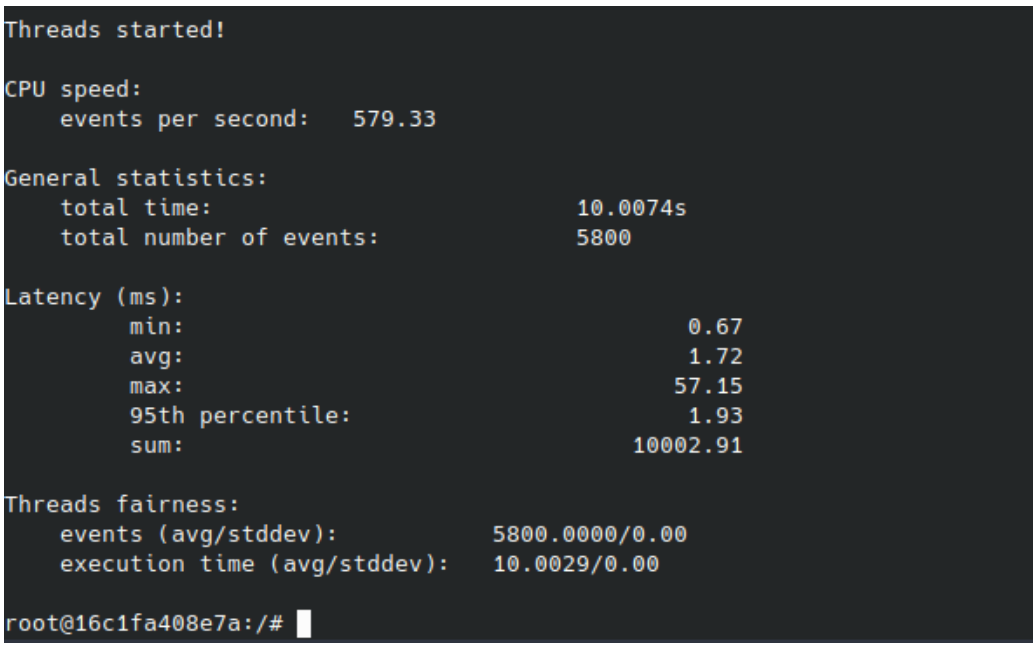
```
apt-get update
apt-get install -y sysbench
```

- **Ejecutar la prueba de CPU**

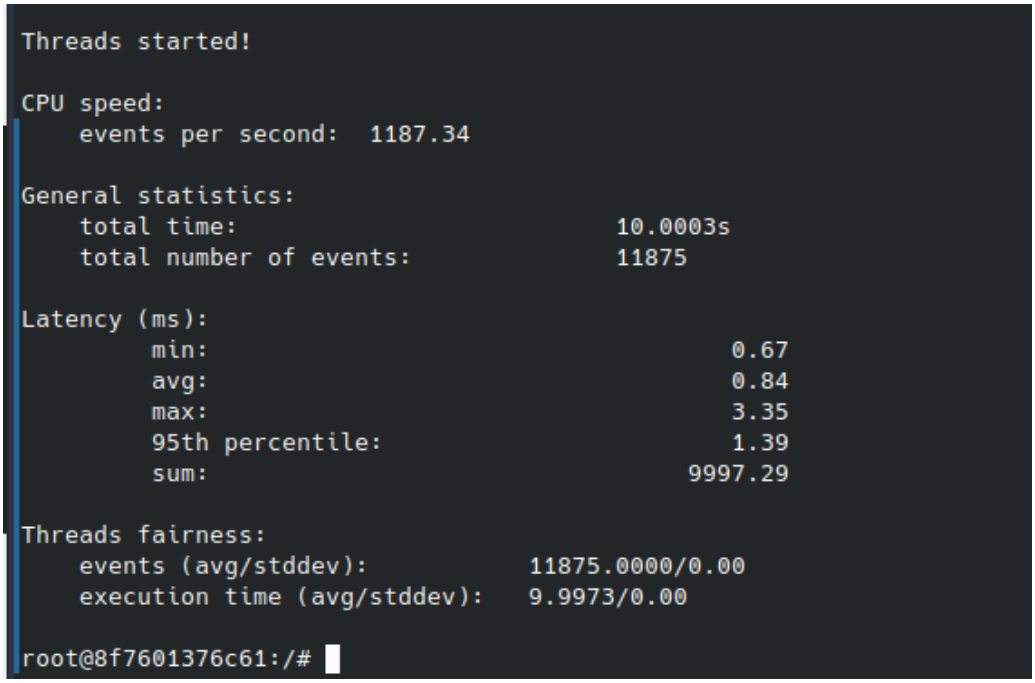
```
sysbench --test=cpu --cpu-max-prime=20000 run
```

4.1. Pruebas

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



2. **Contenedor con una CPU completa (1)**



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:* $\frac{5800}{10,0074} = 579,5711 op/s$
  b) *Prueba 2:* $\frac{11875}{10,003} = 1187,1438 op/s$
  c) *Prueba 3:* $\frac{14108}{10,004} = 1410,2359 op/s$

  Podemos ver que a más CPU, menos tiempo de ejecucuión 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 oepraciones 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 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"]
```

- *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.