Contenedores Docker. Docker-compose

1. Apuntes previos:

   a) Ejemplo de estructura del fichero **docker-compose.yml**

```yaml
version: '3.1'
services:
  app:
    container_name: guestbook
    image: iesgn/guestbook
    restart: always
    ports:
      - 8080:5000
  db:
    container_name: redis
    image: redis
    restart: always
    volumes:
      - redis:/data
volumes:
    redis:
```

   En este fichero YAML se definen los datos relativos a los contenedores de la aplicación como **servicios** (**app** y **db**) y dentro de cada servicio los datos del nombre del **contenedor**, la **imagen** a descargar, la opción de **reiniciarse** si se apaga, las **variables de entorno**, los **puertos** que se exponen y los **volúmenes** de datos persistentes que se van a emplear

   Este fichero debe guardarse en una carpeta de nuestro sistema y a continuación desde la consola del sistema, se debe acceder a dicha carpeta para ejecutar el comando que crea los contenedores definidos en el archivo:

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

   b) Opciones habituales de docker-compose (en **negrita** las más típicas):

   - `docker-compose up`: Crea los contenedores (servicios) que están descritos en el docker-compose.yml.
   - **`docker-compose up -d`**: Crea en modo detach los contenedores (servicios) que están descritos en el
   - **`docker-compose stop`**: Detiene los contenedores que previamente se han lanzado con docker-compose up.
   - **`docker-compose run`**: Inicia los contenedores descritos en el docker-compose.yml que estén parados.
   - `docker-compose rm`: Borra los contenedores parados del escenario. Con las opción -f elimina también los contenedores en ejecución.

- **`docker-compose restart`**: Reinicia los contenedores. Orden ideal para reiniciar servicios con nuevas configuraciones.
- `docker-compose down`: Para los contenedores, los borra y también borra las redes que se han creado con docker-compose up (en caso de haberse creado).
- **`docker-compose down -v`**: Para los contenedores y borra contenedores, redes y volúmenes.
- `docker-compose exec servicio1 /bin/bash`: Ejecuta una orden, en este caso /bin/bash en un contenedor llamado servicio1 que estaba descrito en el docker-compose.yml
- `docker-compose build`: Ejecuta, si está indicado, el proceso de construcción de una imagen que va a ser usado en el docker-compose.yml a partir de los ficheros Dockerfile que se indican.

c) Volúmenes con docker-compose → Ejemplo de volumen para mariadb:

```
version: '3.1'
services:
  db:
    container_name: contenedor_mariadb
    image: mariadb
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: asdasd
    volumes:
      - mariadb_data:/var/lib/mysql
volumes:
    mariadb_data:
```

Se definen en el bloque ultimo volumes: pero además hay que indicar que volumen usara cada contenedor como se indica que se va a crear un volumen mariadb_data apuntando a la carpeta /var/lib/mysql del contenedor

También es posible crear volúmenes en carpetas concretas dentro de nuestro sistema como se muestra:

```
volumes:
    - c:\users\avelino\mariadb_data:/var/lib/mysql
```

2. Se pide ahora desplegar la aplicación Guestbook a partir del fichero `docker-compose.yml` que se muestra en el apartado 1. a). Este fichero lo guardaremos en una carpeta de nuestro usuario de Windows 10 y una vez en la consola accedemos a la carpeta para desplegar la aplicación con el comando `docker compose up -d`. Verificar el acceso a la aplicación con http://localhost:8080

Comprobar también desde la vista de los volúmenes de Docker Desktop que el almacenamiento persistente para la base de datos está creado

3. Crear ahora un fichero **docker-compose.yml** para el despliegue de la aplicación de Temperaturas que se realizó en la práctica anterior, con los siguientes datos:

    a) versión 3.1
    b) los servicios serán frontend y backend
    c) para el servicio frontent:
        a. el contenedor se llamará temperaturas-frontend
        b. la imagen para usar es avelinopef/temperaturas_front
        c. se debe reiniciar automáticamente
        d. El mapeo de puertos es 80:3000
        e. Este servicio depende de backend (usaremos depends-on)
    d) Para el servicio backend:
        a. el contenedor se llamará temperaturas-backend
        b. la imagen para usar es avelinopef/temperaturas_back
        c. se debe reiniciar automáticamente

    Una vez creado, como en el caso anterior debemos guardarlo en una carpeta particular y desplegarlo con `docker-compose up -d`
    Verificar desde un navegador que se accede a la aplicación con http://localhost

4. Se pide ahora desplegar la aplicación de Wordpress con docker-compose a partir del fichero wp_mariadb.yml que se entrega.

    Observa los valores de las variables de entorno y ajusta los valores de modo que:

    a) El nombre del contenedor de wordpress será contenedor_wp
    b) El nombre del contenedor de mariadb será contenedor_mariadb
    c) El usuario para wp en mariadb será admin
    d) La contraseña del usuario anterior será naranco23
    e) La base de datos de wp será bbdd_wp
    f) El volumen de datos para wordpress será datos_wp
    g) El volumen de datos para mariadb será datos_mariadb

    Ya que el fichero no cuenta con el nombre habitual por defecto (docker-compose.yml), la composición debe ser lanzada con la orden:

    ```
    docker-compose -f wp_mariadb.yml up -d
    ```

    Accede al http://localhost y completa la instalación de wp

5. Ahora desplegaremos una aplicación java sobre un servidor Tomcat y con proxy inverso nginx que permita el acceso a la web de la aplicación. Para ello:

    a) Vamos a utilizar los volúmenes para copiar el archivo sample.war al directorio de publicación de tomcat y el fichero default.conf para copiar una versión modificada

del fichero por defecto de nginx. Estos dos archivos deben copiarse una carpeta particular de nuestro sistema (por ejemplo c:\users\usuario\tomcat)

b) Copiamos a la misma carpeta el fichero tomcat-nginx.yml

c) Desplegamos como siempre con `docker-compose`

d) Verificamos el acceso a la aplicación con http://localhost

6. Se pide ahora realizar el despliegue con docker-compose de la aplicación phpmyadmin que hará uso de un servidor de bases de datos mariadb. Para ello:

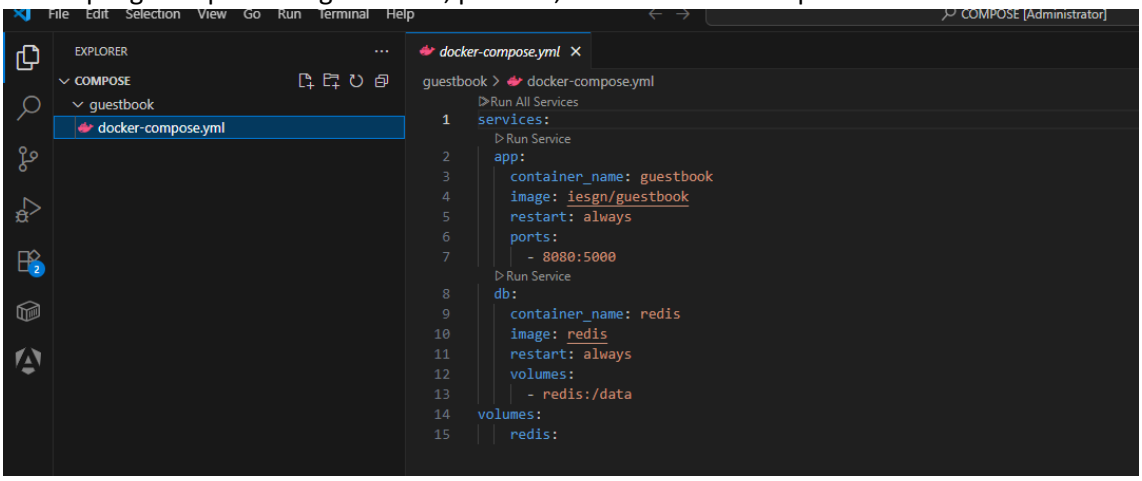
   a) Debe crearse un fichero Docker-compose.yml que permita el desplegar los dos servicios
   b) Se crearán también volúmenes de persistencia de datos para los dos contenedores
   c) Se verificará el acceso a phpmyadmin desde la URL cargada en un navegador

7. Se pide ahora desplegar la tienda virtual Prestashop. Para ello se entrega el fichero prestashop.yml y se deben realizar los siguientes cambios:

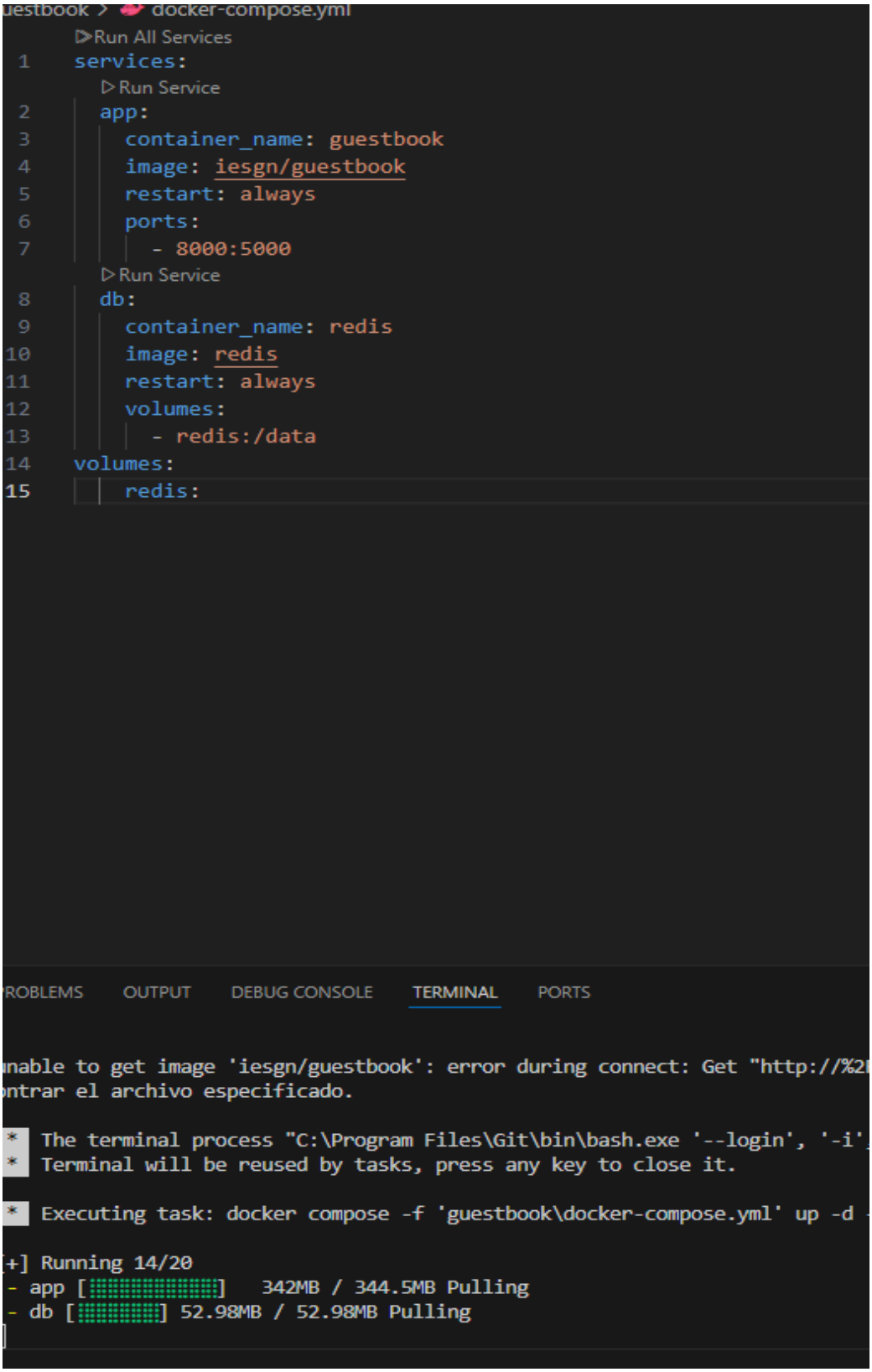
   a) El usuario de la base de datos para prestashop será user_prestashop
   b) El nombre de la base de datos será mitienda

   Se debe acceder a la URL de la tienda y mostrar la aplicación online en el navegador

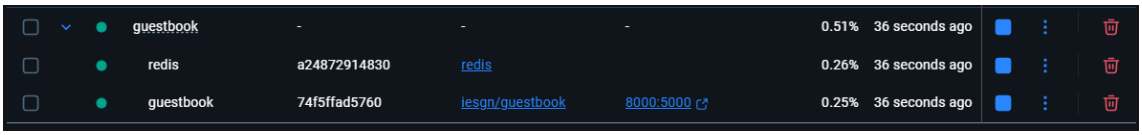
1.- Despliego la aplicación guestbook, primero, creo el docker-compose:



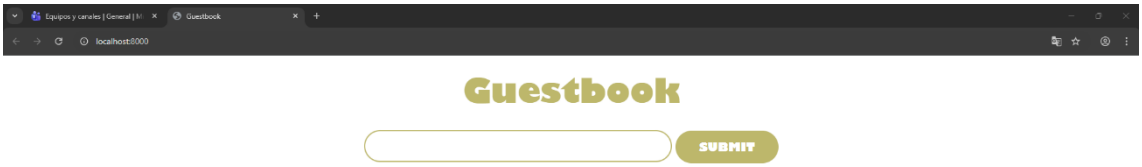
2.- Tras esto lo ejecuto, esta vez desde el visual studio, después serán todas desde la consola.



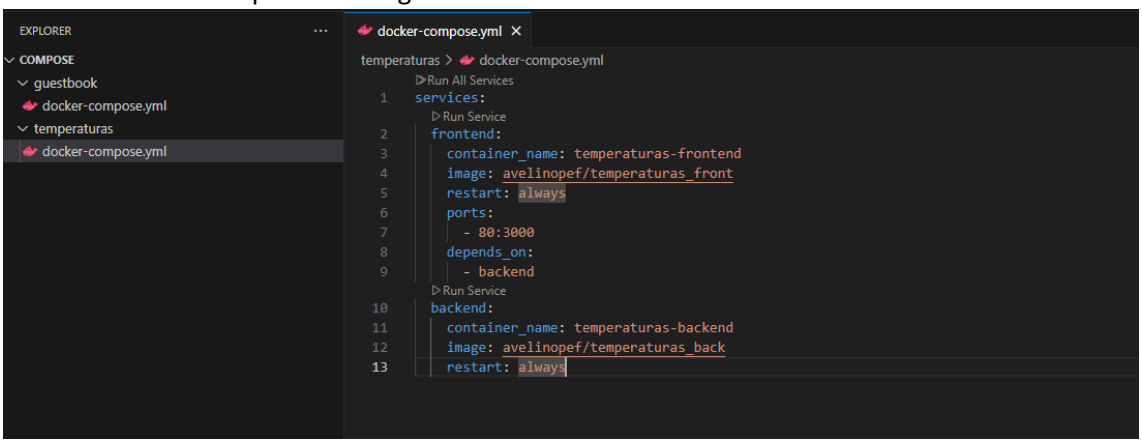
3.- Como se puede observar ya están funcionando los dos contenedores.



4.- Ahora si vamos a localhost:8000, podrémos ver la página funcionando.



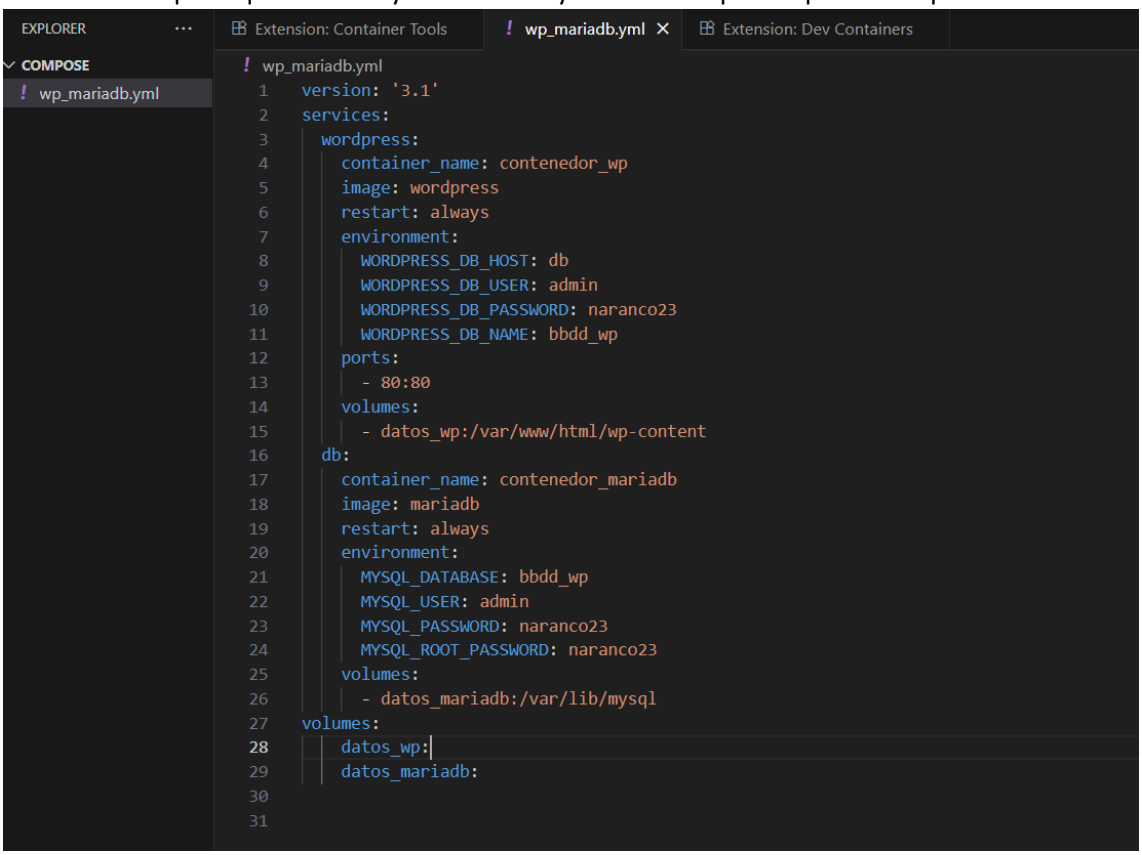
5.- Ahora vamos a desplegar la página de temperaturas de avelino, asique volvemos a crear un docker-compose de la siguiente manera.



6.- Ejecutamos el comando docker-compose up -d

Y ya la tendríamos funcionando

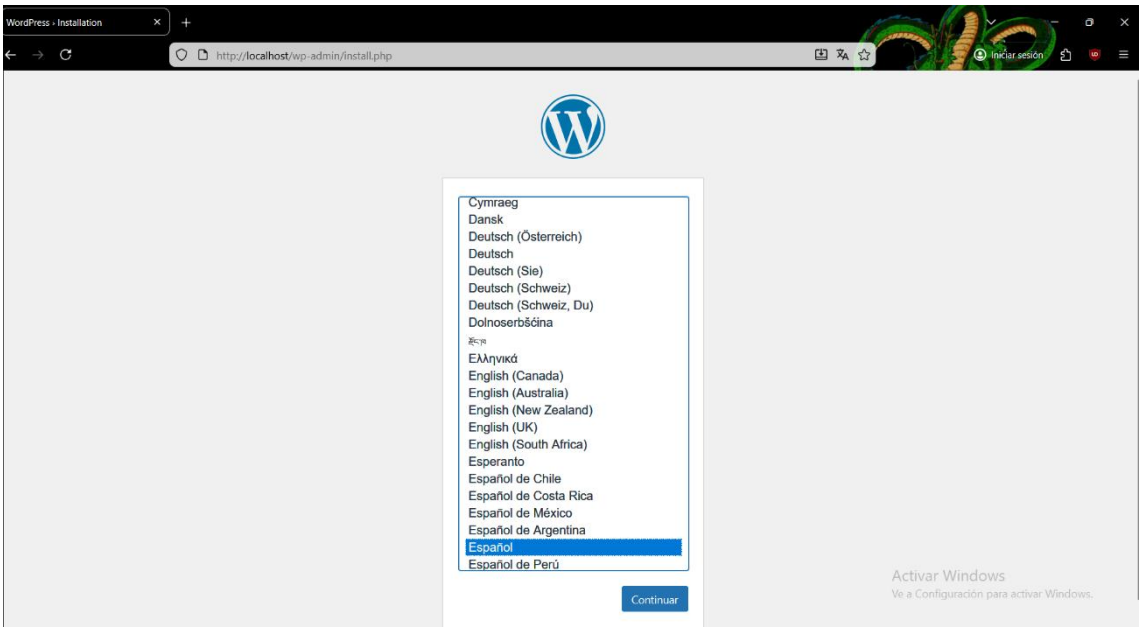
7.- Ahora vamos a hacer lo mismo para la página de wordpress, para ello descargamos el docker-compose que nos dan y lo editamos y cambiamos por lo que se nos pide:



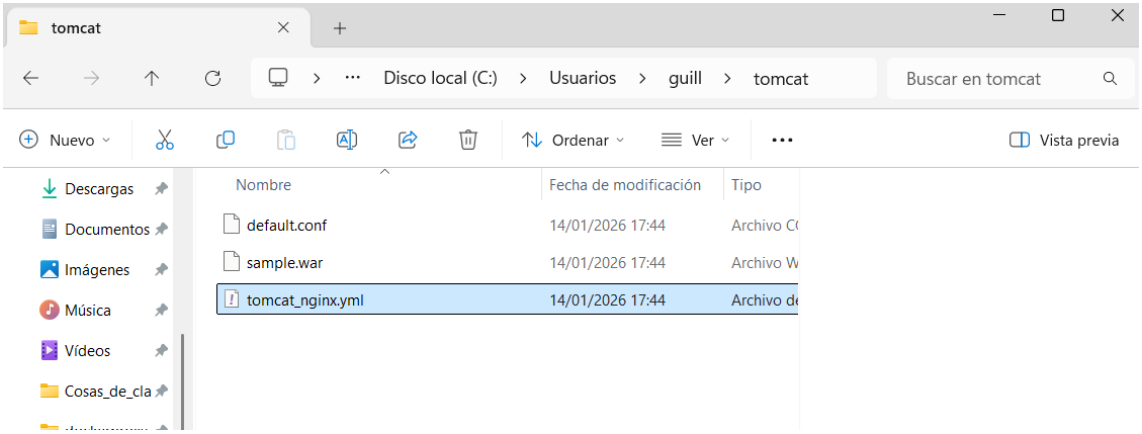
8.- después de esto debemos ejecutar el comando, en este caso es con -f ya que el archivo adjunto no se llama docker-compose.

```
C:\Users\guill\Documents\2DAWB\COMPOSE>docker-compose -f wp_mariadb.yml up -d
```

9.- Ahora si vamos al localhost ya tenemos la página de wordpress funcionando.



10.- Para continuar, creamos una carpeta en c\users\tomcat y metemos ahí los archivos que nos dan con el ejercicio.

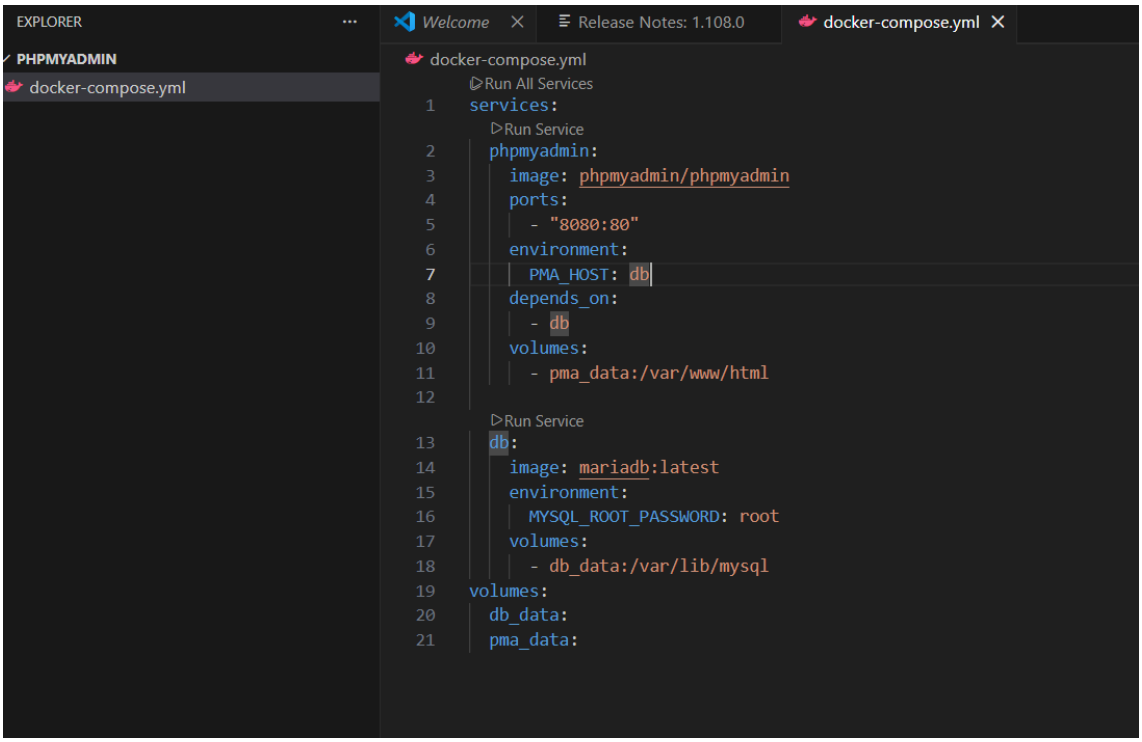


11.- Volvemos a hacer lo mismo con el docker-compose -f

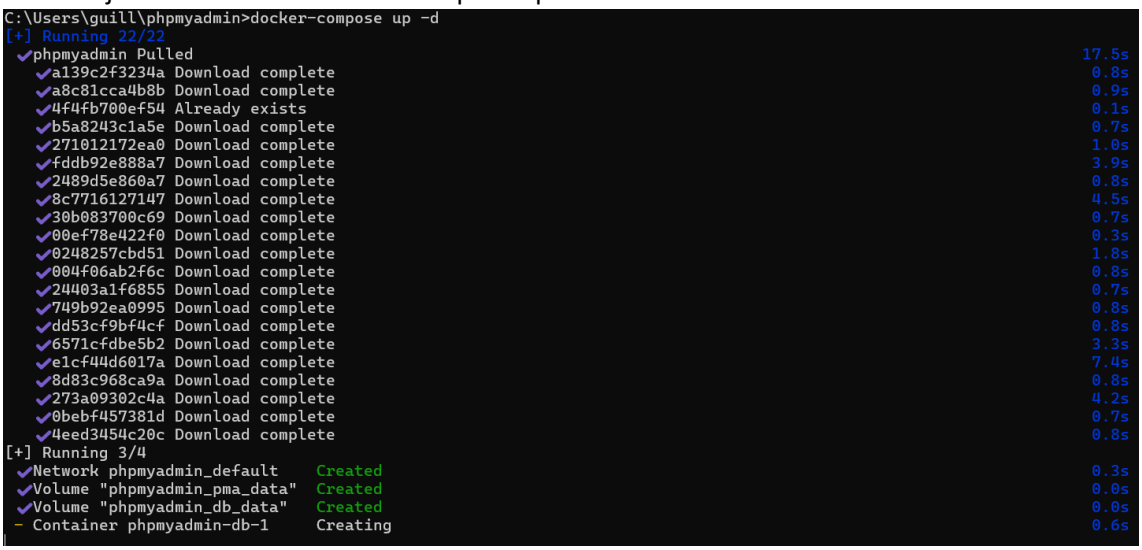
```
C:\Users\guill\tomcat>docker-compose -f tomcat_nginx.yml up -d
```

12.- Ahora si entramos a localhost nos saldrá el sample.

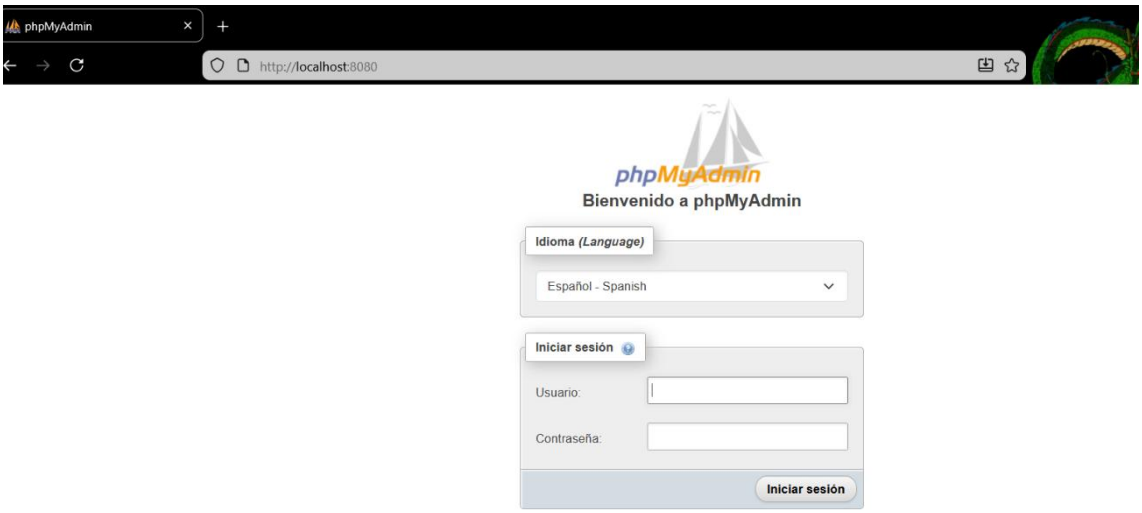
13.- Ahora crearemos un contenedor para phpmyadmin con mariadb. Primero creamos el docker-compose de la siguiente manera.



14.- Lo ejecutamos con el docker-compose up -d



15.- Y si ahora vamos a localhost:8080 nos aparecerá la interfaz de phpMyAdmin, donde usuario será root y contraseña root.



16.- Este es el .yml del último ejercicio, no sé si he hecho algo mal, pero a la hora de hacer el docker-compose up me da un error, estuve buscando en internet y parece que han puesto algunos repositorios privados.