**Práctica 10. Docker**

Lee detenidamente cada uno de los puntos antes de realizar las tareas solicitadas. Revisa los recursos incluidos.

Al adjuntar ficheros, introdúcelos en carpetas con el nombre de la agrupación a la que corresponde. Revisa el apartado “A entregar” al final del doc.

Parte 1: **Docker (Dockerfile)**

Actividades 1-5 se realizan en conjunto: Crear imagen docker para web y arrancar.

1. Crea un directorio **src/** en tu equipo y dentro coloca un fichero **index.php** de prueba con el siguiente código (y muestra capturas):  
   <?php phpinfo(); ?>
2. Prepara un contenedor con **Apache y PHP** y el programa .php anterior usando **Dockerfile**. Crea el dockerfile en /src (adjunta o copia el contenido de este fichero).
   1. Emplea una imagen oficial de PHP con Apache (ej. php:8.2-apache)
   2. Copia index.php en el contenedor, guarda en /var/www/html
   3. Expón el puerto 80
   4. Recuerda finalizar con un comando de inicio
3. Construye la **imagen** de la aplicación (capturas del docker build y su resultado)
4. Ejecuta el **contenedor** (captura del run). Establece vínculos de puertos y un nombre especifico. Abre http://localhost:8080 y verifica que se muestra la información de PHP (captura).
5. Vía comandos, **muestra los contenedores que tengas en marcha** y **detén el contenedor creado** (capturas de las instrucciones).

Parte 2: **Docker-compose (YAML)**

Actividades 6-8 se realizan en conjunto: Añadir MySQL.

1. Ahora vamos a agregar **MySQL**, pero este se ejecutará desde **otro contenedor** diferente al creado previamente. Para que puedan trabajar en conjunto, crea un archivo **docker-compose.yml**. Establece las *variables de entorno* de MySQL para definir una contraseña al usuario root (a tu elección, pero toma nota), crear una base de datos con tu nombre y \_db al final (formato [nombre\_db]) y un usuario con tu nombre (contraseña también a tu elección).
2. Continuando con el punto anterior, asegúrate de definir un **volumen** para poder tener persistencia a nivel local con el contenedor de MySQL. Para esto, dentro de src/, en local, crea una **subcarpeta llamada “datadir”**, a vincular con la carpeta correspondiente del contenedor (ver documentación – en recomendaciones). Explica cómo has introducido esta configuración.
3. Crea un nuevo fichero de PHP llamado “**db\_test.php**” con el código:  
   <?php  
   $conn = new mysqli("db", "user", "password", "[nombre\_db]");  
   if ($conn->connect\_error) {  
    die("Conexión fallida: " . $conn->connect\_error);  
   }  
   echo "Conexión establecida con MySQL!";  
   ?>  
   Cambia la información de "db" (nombre del servicio), "user", "password", "[nombre\_db]" por los datos definidos en los puntos anteriores para establecer la conexión con MySQL.

Edita **dockerfile** y genera nuevamente la **imagen** de tu aplicación, para incluir db\_test.php.

**Arranca los contenedores** y verifica que <http://localhost:8080/db_test.php> funciona (captura del comando de *docker-compose up*, captura de navegador, y captura de *logs del contenedor de MySQL* donde se vea la conexión realizada desde el navegador).

**Parte 3: Datos persistentes y despliegue de App**

Actividades 9-10 se realizan en conjunto: Añadir PHPMyAdmin y desplegar app web.

1. **Detén** los contenedores con *docker-compose down* (muestra captura). Edita docker-compose.yml para incluir **phpMyAdmin**. Configura las variables de entorno con la contraseña de root que declaraste anteriormente e indicando el nombre del servicio/contenedor de MySQL (ver documentación). Vincula los puertos 8081:80.   
   Una vez configurado, haz docker-compose up y accede a PHPMyAdmin en [http://localhost:8081](http://localhost:8081/) con usuario root y verifica que aparece la base de datos con tu nombre (captura de PHPMyAdmin).
2. Detén los contenedores nuevamente. **Descarga el fichero index.php adjunto** al enunciado. Se debe desplegar con docker como una aplicación web:
   1. Edita el inicio del código PHP según los datos que has definido previamente, concretamente:  
      // **Configuración de la conexión a la base de datos**  
      $host = "localhost:8080"; // A DEFINIR POR EL DESARROLLADOR  
      $user = "nombre"; // A DEFINIR POR EL DESARROLLADOR  
      $pass = "password"; // A DEFINIR POR EL DESARROLLADOR  
      $db = "nombre\_db"; // A DEFINIR POR EL DESARROLLADOR  
      (Adjunta e fichero index.php modificado)
   2. Vuelve a editar el dockerfile para incluir este fichero index.php en la copia (añádelo a la carpeta src) y genera nuevamente la imagen (muestra los cambios y la creación de la imagen).
   3. Edita docker-compose.yml si es necesario (indica si has tenido que cambiar algo y muestra los cambios).
   4. Arranca los contenedores empleando el fichero YAML (captura de la instrucción up y de los contenedores en funcionamiento).
   5. Comprueba si puedes acceder a index.php vía navegador y trabajar con la aplicación (captura del navegador). Introduce mensajes nuevos.
   6. Haz docker-compose down y luego otro up. Comprueba si los mensajes creados previamente siguen existiendo en la aplicación desplegada.

El resultado de esta práctica es una pila LAMP montada con Docker.

**A entregar:**

* Carpeta Actividad 1-5:
  + src/index.php (solo con phpinfo() )
  + src/dockerfile
  + imagen creada (busca en /var/lib/docker)
* Carpeta Actividad 6-8:
  + src/dockerfile
  + src/db\_test.php
  + src/docker-compose.yml (con apache, php, mysql y la app web)
  + imagen creada (busca en /var/lib/docker)
  + Contenido de src/datadir
* Carpeta Actividad 9-10:
  + src/dockerfile
  + src/index.php (adjunto modificado)
  + src/docker-compose.yml (con apache, php, mysql, PHPMyAdmin y la app web)
  + imagen creada (busca en /var/lib/docker)
  + Contenido de src/datadir
* Documento con las capturas y las explicaciones solicitadas, en orden y claramente enumeradas.

**Recursos:**

* <https://docs.docker.com/>
* <https://hub.docker.com/_/mysql>
* <https://hub.docker.com/_/phpmyadmin>

**Recordatorio:**

* Docker image = aplicación y sus recurso preparados para desplegar
* Docker container = imagen desplegada en entorno virtualizado
* Dockerfile = Para poder construir una imagen
* Docker-compose.yaml = para poder arrancar multiples contenedores en conjunto (en una misma red)

**Condiciones de entrega:**

* La práctica se **debe** entregar de forma **individual**, cada uno debe presentar sus propias respuestas. Sin embargo, se puede trabajar en equipo.
* Se debe entregar un documento de texto (.pdf, .docx, .odt, etc.) con los ejercicios correctamente ordenados, identificados y **numerados**.
* En cada página del documento debe aparecer el nombre completo del alumno.
* La nota comprenderá un valor numérico entre 0 y 10.
* **La fecha límite de entrega es la indicada en Google Classroom.**
* **Se podrá entregar hasta 72 horas más tarde de la fecha límite pero con una penalización sobre su puntuación (no será posible aspirar al 10).**