PRÁCTICA DOCKER

# Instalación Docker

* Deberéis instalar en vuestro equipo la versión adecuada de docker.
* Una vez hayáis hecho esta instalación deberéis configurar vuestro sistema para poder ejecutar docker sin permisos de administrador (en caso de ser necesario).
* Posteriormente desde consola deberéis ejecutar el comando docker que me permite obtener la versión instalada y el comando docker para obtener el "HOLA MUNDO" de esta tecnología.
* Crea una cuenta en Dockerhub.
* Deberéis capturar tres pantallazos, uno para la versión, otro para Hola Mundo y otro para Dockerhub con vuestra cuenta, comprimirlos en un único fichero Ejercicio1.zip

# Ejecutando y gestionando contenedores

* Debéis descargar en vuestro equipo las imágenes de Ubuntu:20.04 y mariadb:latest
* Ejecuta un contenedor de la imagen de Ubuntu. Páralo y vuelve a arrancarlo.
* Crea una carpeta llamada myapp en tu pc. Colocar dentro un fichero llamado index.html con el siguiente contenido: <h1>HOLA SOY ALUMNO</h1>
* Colocar en ese mismo directorio un archivo llamado index.php con el siguiente contenido: <?php phpinfo(); ?>
* Arranca un contenedor que ejecute una instancia de la imagen php:7.4-apache, que se llame web y que sea accesible desde tu equipo en el puerto 8181. Debe hacer un bindmount de tu carpeta myapp a la carpeta /var/www/html del contenedor.
* Arrancar un contenedor que se llame bbdd y que ejecute una instancia de la imagen mariadb para que sea accesible desde el puerto 3336.
* Antes de arrancarlo visitar la página del contenedor en Docker Hub (https://hub.docker.com/\_/mariadb) y establecer las variables de entorno necesarias para que la contraseña de root sea root, crear una base de datos automáticamente al arrancar que se llame prueba, crear el usuario invitado con la contraseña invitado.
* Comprime en un fichero Ejercicio2.zip capturas de pantalla del fichero index.html cargado en tu navegador en el puerto 8181, y de index.php. También entrando en el contenedor con mariadb y mostrando las bases de datos que tiene (ejecutar mysql –u xxxx –p y dentro show databases).

# Creación de imágenes con Dockerfile

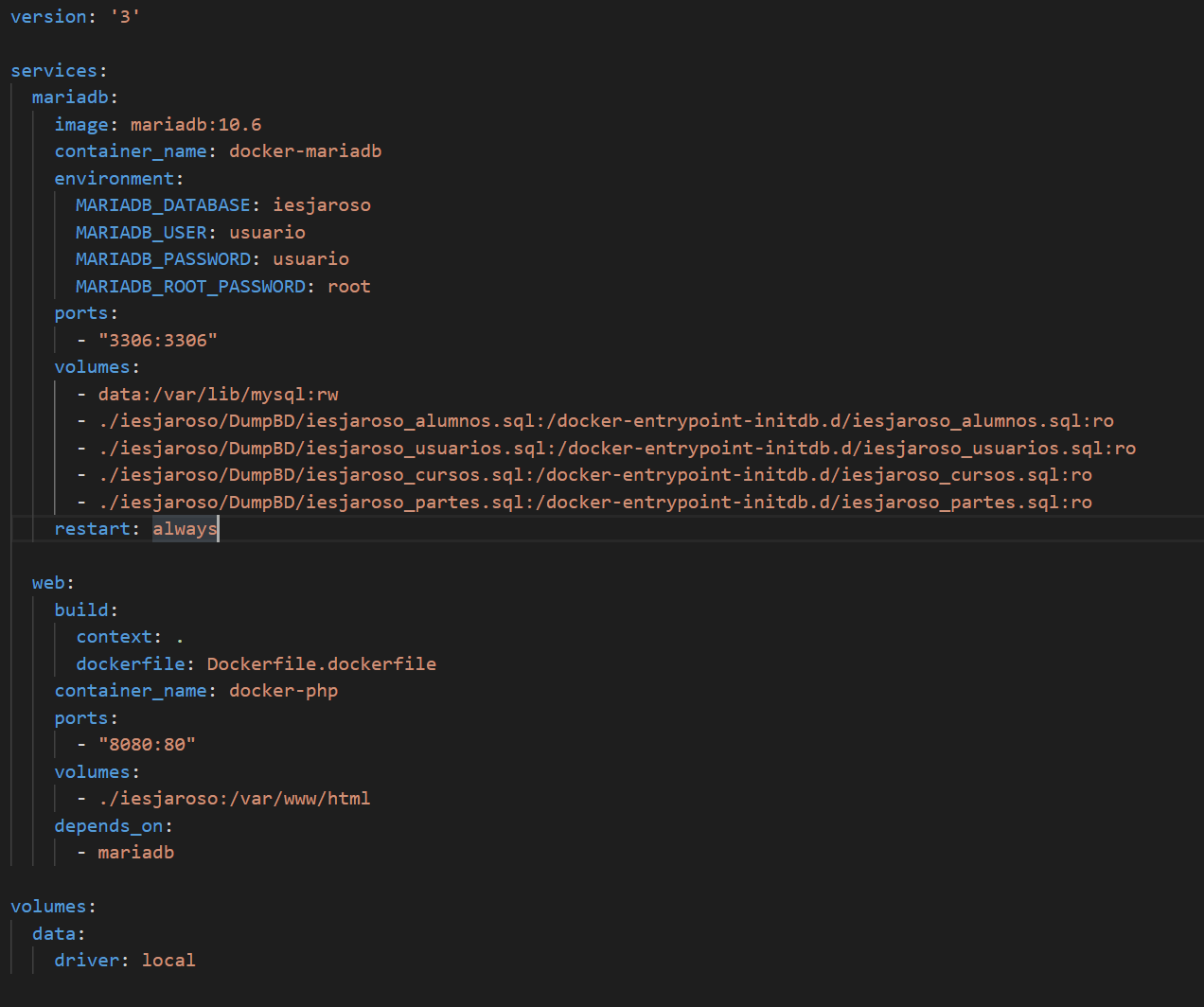
* Partiendo de la imagen php:8.0-apache construye un Dockerfile que incluya:
  1. Actualice repositorios (apt update –y)
  2. Instalar nano (apt install –y nano)
  3. Instalar extensión php para Postgresql (apt install –y php8-pdo-pgsql)
  4. Instalar extensión php para mbstring (apt install –y php8-mbstring)
  5. Coloca en el directorio raíz de apache (/var/www/html) una de las prácticas hechas en clase sin base de datos (todoist, agenda, ahorcado, …)
* Construye la imagen a partir del Dockerfile que se llame TuNombreDockerHub/web
* Entrega Ejercicio3.zip que incluya pantallazo mostrando la aplicación corriendo en tu imagen Docker, otra con el resultado de ‘docker images’ para que se vea que se ha creado la imagen, y otra con tu página de Dockerhub donde se vea la has subido la imagen a tu repositorio (docker push).

# Aplicaciones con docker-compose

* Es este ejercicio vamos a poner en marcha una tienda virtual realizada con Prestashop. Utilizaremos el fichero docker-compose.yml de Bitnami que podemos encontrar en la siguiente URL <https://hub.docker.com/r/bitnami/prestashop/>
* Una vez hemos descargado el fichero docker-compose.yml asociado deberemos modificarlo de la siguiente manera. Modificar los valores de las variables de entorno para conseguir lo siguiente:
  1. El usuario de prestashop para conectarse a la base de datos deberá ser admin y su contraseña nimda. Investigar en la página de Dockerhub cuál es el nombre de las variables de entorno que debo modificar y/o añadir.
  2. Modificar el nombre de la base de datos de prestashop para que se llame mitienda. Debéis de modificar esos valores en los dos servicios. Investigar en la página de Dockerhub cuál es el nombre de las variables de entorno que debo modificar.
* Entregar Ejercicio4.zip con captura de pantalla de vuestro fichero docker-compose.yml y de la ejecución en vuestra máquina de la tienda Prestashop corriendo a través de vuestro docker-compose modificado.

# Aplicación web PHP + Mysql

* En este ejercicio debéis desplegar una aplicación web php con base de datos, cualquiera de las vistas en el tema 5 del curso de Servidor, puede ser vuestra o de las hechas en clase.
* Para ellos deberéis crear un fichero docker-compose.yml que tenga dos servicios uno para php con apache y otro servicio con mariadb:10.6 Deberéis conectarlos (depends on). Además, los ficheros irán en un volumen de vuestro equipo y se copiarán en el servicio adecuado. La base de datos también llevará persistencia en vuestro equipo local, de modo que los cambios que se hagan en la base de datos se vean desde fuera de los contenedores. Os dejo un ejemplo de docker-compose.yml pero debéis cambiar las variables de entorno correspondientes y la versión de mariadb. El nombre de la bd, usuario y password será el que haya en el código de la app php que vayas a desplegar.



* Cuando lo pruebes verás que hay un problema pues en la imagen por defecto de php no viene instalada la extensión PDO. Para solucionarlo deberás crear un Dockerfile desde la imagen php:8.0-apache pero instalando la extensión. Y cambiando en el docker-compose.yml que no coja la imagen php:8.0-apache sino la de tu Dockerfile.

FROM php:8.0-apache

RUN docker-php-ext-install pdo pdo\_mysql

* Habrá otro problema. En el modelo.php la conexión a la base de datos la hacemos a “localhost”. **Deberás cambiarlo y poner la ip del contenedor de mariadb** (172.20.0.2, o la que sea, puede cambiar).
* El último problema es que las tablas no están creadas en la base de datos, con lo cual deberás crearlas al iniciar el contenedor mariadb. Para ello si te fijas hay creado un volumen indicando donde están los ficheros dump de la base de datos para que los importe al arrancar. Pero a veces no funciona. En el peor de los casos te conectas a la consola del contenedor mariadb, ejecutas mysql –uroot –p, creas la base de datos si no está creada. La usas (use basededatos;). Por último, copias el código de los dumps hechos de las tablas y los pegas, deberían crearse las tablas.
* Muestra la aplicación corriendo a través de los contenedores docker. Capturas de la app funcionando en local en un navegador, pero a través de docker, de los contenedores docker arrancados y de tu docker-compose.yml.