

Práctica 1: Imagen MariaDB Personalizada

1. Preparación del proyecto

```
1 mkdir db-docker && cd db-docker
2 mkdir sql
```

Listing 1: Crear estructura

Crea el directorio del proyecto y la carpeta para scripts SQL.

2. Construcción de la imagen

```
1 docker build -t filca/db-docker:v1 .
```

Listing 2: Construir imagen local

Construye la imagen usando el Dockerfile del directorio actual. Etiqueta: `filca/db-docker:v1`

3. Autenticación y publicación

```
1 docker login
```

Listing 3: Login en Docker Hub

Autentica tu usuario (`filca05`) en Docker Hub.

```
1 docker tag filca/db-docker:v1 filca05/db-docker:v1
```

Listing 4: Re-etiquetar para usuario

Cambia el nombre del repositorio al usuario correcto (`filca05`).

```
1 docker push filca05/db-docker:v1
```

Listing 5: Subir imagen

Publica la imagen en Docker Hub.

4. Verificación

```
1 docker run --name db-final-test -p 3306:3306 -d filca05/db-docker:v1
```

Listing 6: Ejecutar contenedor de prueba

Levanta un contenedor en segundo plano exponiendo el puerto 3306.

```
1 docker exec -it db-final-test mariadb -u manager -psecret_manager -e "USE app_data; SELECT * FROM usuarios;"
```

Listing 7: Entrar al contenedor y comprobar datos

Verifica que la base de datos `app_data` y la tabla `usuarios` se hayan creado con los datos iniciales.

Práctica 2: Docker Compose - WordPress + MariaDB

1. Preparación del proyecto

```
1 mkdir practica2-compose && cd practica2-compose
```

Listing 8: Crear carpeta

Crea y entra en el directorio del proyecto Compose.

2. Comandos principales de Docker Compose (versión moderna)

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

Listing 9: Levantar el entorno

Lee `docker-compose.yml`, crea red, volumen y arranca los contenedores en segundo plano.

```
1 docker compose ps
```

Listing 10: Ver estado de contenedores

Muestra los servicios activos y puertos mapeados (ej: 0.0.0.0:8080->80/tcp).

```
1 docker compose logs -f
```

Listing 11: Ver logs

Muestra logs en tiempo real de todos los servicios.

```
1 docker compose exec web bash
```

Listing 12: Entrar al contenedor web

Abre una shell dentro del contenedor WordPress.

```
1 docker compose down
```

Listing 13: Detener y eliminar contenedores + red (conserva datos)

Para el entorno pero mantiene el volumen `data_mysql`.

```
1 docker compose down -v
```

Listing 14: Detener y eliminar TODO (incluye datos)

Elimina también los volúmenes nombrados (borra la BD permanentemente).

3. Comandos útiles de verificación

```
1 docker volume ls
```

Listing 15: Ver volúmenes creados

Muestra volúmenes, incluido `practica2compose_data_mysql`.

```
1 http://localhost:8080
```

Listing 16: Acceso final

Abre el navegador → pantalla de instalación de WordPress.

Resumen rápido de comandos clave (para examen)

Comando	Qué hace
<code>mkdir db-docker; cd db-docker; mkdir sql</code>	Prepara carpeta del proyecto Práctica 1 y subcarpeta sql
<code>docker build -t filca/db-docker:v1 .</code>	Construye la imagen desde el Dockerfile del directorio actual
<code>docker login</code>	Autentica tu usuario en Docker Hub (filca05)
<code>docker tag filca/db-docker:v1 filca05/db-docker:v1</code>	Re-etiqueta la imagen con tu usuario de Docker Hub
<code>docker push filca05/db-docker:v1</code>	Sube la imagen al repositorio público de Docker Hub
<code>docker run --name nombre -p 3306:3306 -d filca05/db-docker:v1</code>	Ejecuta contenedor MariaDB personalizado exponiendo puerto 3306
<code>docker compose up -d</code>	Levanta todo el entorno WordPress + MariaDB (crea red y volumen)
<code>docker compose ps</code>	Muestra estado de los contenedores (nombre, puerto, estado)
<code>docker compose down</code>	Para y elimina contenedores + red (conserva los datos de la BD)
<code>docker compose down -v</code>	Elimina también los volúmenes (borra permanentemente la base de datos)
<code>http://localhost:8080</code>	Acceso al instalador de WordPress en el navegador

Nota importante: En Ubuntu 24.04+ siempre usar `docker compose` (con espacio), NO `docker-compose` (con guion).