Práctica Docker: Aplicación Multi-Contenedor

Objetivos de la Práctica

Al finalizar esta práctica serás capaz de:

- Instalar Docker en tu sistema operativo
- Entender la estructura de un proyecto Docker
- Lanzar contenedores de forma individual
- Usar Docker Compose para gestionar múltiples contenedores
- Verificar que los servicios funcionan correctamente

Parte 1: Instalación de Docker

1.1 Instalación en Windows

Requisitos previos

- Windows 10/11 64-bit (Pro, Enterprise o Education)
- Virtualización habilitada en BIOS
- Mínimo 4GB de RAM

Paso 1: Instalar WSL 2

Abre **PowerShell como Administrador** y ejecuta:

```
wsl --install
```

Reinicia el equipo cuando se solicite.

Paso 2: Instalar Docker Desktop

1. Descarga Docker Desktop desde: https://www.docker.com/products/docker-desktop/
2. Ejecuta el instalador
3. Marca la opción **"Use WSL 2 instead of Hyper-V"**
4. Completa la instalación y reinicia

Paso 3: Verificar la instalación

```
docker --version

docker compose version

docker run hello-world
```

1.2 Instalación en Linux (Ubuntu/Debian)

Paso 1: Instalar dependencias

sudo apt update

sudo apt install -y ca-certificates curl gnupg lsb-release

Paso 2: Añadir repositorio de Docker

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Paso 3: Instalar Docker

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

Paso 4: Configurar permisos

sudo usermod -aG docker $USER

newgrp docker

Paso 5: Verificar la instalación

docker --version

docker compose version

docker run hello-world

📝 Ejercicios - Instalación

1. ¿Qué versión de Docker tienes instalada?
2. Ejecuta `docker run hello-world`. ¿Qué mensaje aparece?

Parte 2: Comandos Básicos de Docker

2.1 Gestión de Imágenes

Operación	**Comando Docker**

Descargar una imagen	`docker pull nginx:alpine`
Listar las imágenes descargadas	`docker images`
Eliminar una imagen	`docker rmi nginx:alpine`

2.2 Gestión de Contenedores

Acción	**Comando**
Crear y ejecutar un contenedor	`docker run nginx`
Ejecutar en segundo plano	`docker run -d --name mi-web nginx`
Ejecutar con **mapeo de puertos**	`docker run -d -p 8080:80 --name mi-web nginx`
Listar contenedores **activos**	`docker ps`
Listar todos los contenedores	`docker ps -a`
Ver **logs**	`docker logs mi-web`
Parar	`docker stop mi-web`
Iniciar (uno parado)	`docker start mi-web`
Eliminar	`docker rm mi-web`
Acceder al interior con shell sh	`docker exec -it mi-web sh`

2.3 Limpieza

Acción	**Comando**
Comprobar Uso de espacio	`docker system df`
Liberar Recursos no utilizados (contenedores, imágenes, volúmenes y	`docker system prune`

redes sin referencia).	

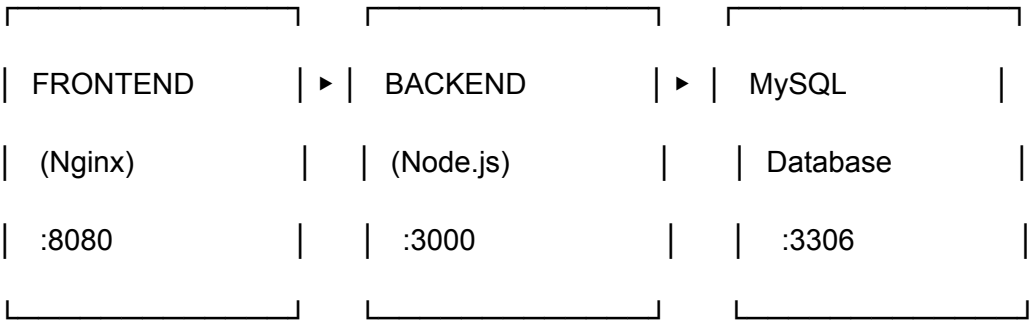
📝 Ejercicios - Comandos Básicos

1. Descarga la imagen `nginx:alpine` y comprueba que está descargada.
2. Crea un contenedor llamado `prueba-nginx` que mapee el puerto 9090 al 80.
3. Accede a `http://localhost:9090` en tu navegador. ¿Qué ves?
4. Muestra los logs del contenedor.
5. Para y elimina el contenedor.

Parte 3: Estructura del Proyecto

3.1 Arquitectura de la Aplicación

Nuestra aplicación tiene tres componentes:



- **Frontend**: Página web servida por Nginx (puerto 8080)
- **Backend**: API REST con Node.js y Express (puerto 3000)
- **Database**: Base de datos MySQL (puerto 3306)

3.2 Estructura de Carpetas

```
docker-practica/
├── docker-compose.yml      # Configuración de todos los servicios
├── frontend/
│   ├── Dockerfile          # Instrucciones para crear la imagen
│   ├── nginx.conf          # Configuración del servidor web
│   ├── index.html          # Página principal
│   ├── styles.css          # Estilos
│   └── app.js              # Lógica JavaScript
├── backend/
│   ├── Dockerfile          # Instrucciones para crear la imagen
│   ├── package.json        # Dependencias de Node.js
```

```
│   └── server.js              # Código del servidor API
└── database/
    └── init.sql               # Script para crear tablas y datos
```

3.3 Preparar el Proyecto

1. Descarga y descomprime el archivo `docker-practica-dam.zip`
2. Abre una terminal en la carpeta del proyecto

cd docker-practica-dam

ls -la

📝 Ejercicios - Estructura

1. ¿Cuántos servicios tiene la aplicación?
2. ¿Qué archivo define cómo se construye cada imagen?
3. ¿En qué puerto estará disponible el frontend?

Parte 4: Lanzar la Base de Datos

Vamos a lanzar primero solo el contenedor de MySQL.

Recuerda que el VM Docker daemon (la aplicación que te has descargado e instalado) debe de estar en ejecución. También necesitarás internet para poder descargar las imágenes de DockerHub.

4.1 Crear y Ejecutar el Contenedor

```
# Lanzar MySQL con las configuraciones necesarias

docker run -d \
  --name tienda-db \
  -e MYSQL_ROOT_PASSWORD=rootpassword \
  -e MYSQL_DATABASE=tienda \
  -p 3306:3306 \
  -v $(pwd)/database/init.sql:/docker-entrypoint-initdb.d/init.sql \
  mysql:8.0
```

4.2 Verificar que Funciona

```
# Ver que el contenedor está corriendo
docker ps
# Ver los logs (esperar a "ready for connections")
docker logs -f tienda-db
# Pulsa Ctrl+C para salir
```

4.3 Conectarse a MySQL

```
# Entrar al cliente MySQL
docker exec -it tienda-db mysql -u root -prootpassword
```

Una vez dentro, ejecuta estas consultas:

```
-- Ver bases de datos
SHOW DATABASES;

-- Usar la base de datos tienda
USE tienda;

-- Ver tablas
SHOW TABLES;

-- Ver productos
SELECT * FROM productos;

-- Contar productos
SELECT COUNT(*) FROM productos;

-- Salir
EXIT;
```

4.4 Ejecutar Consultas Directamente

```
# Ejecutar una consulta sin entrar al contenedor
docker exec tienda-db mysql -u root -prootpassword -e "SELECT * FROM tienda.productos;"
```

4.5 Limpiar

```
# Parar y eliminar el contenedor
docker rm -f tienda-db
```

📝 Ejercicios - Base de Datos

1. Lanza el contenedor de MySQL y verifica que está corriendo.
2. Conéctate y cuenta cuántos productos hay en la tabla.
3. Ejecuta una consulta que muestre solo el nombre y precio de los productos.
4. Para y elimina el contenedor.

Parte 5: Lanzar el Backend

Ahora lanzaremos MySQL y el Backend juntos, conectados por una red.

5.1 Crear una Red

```
# Crear red para conectar los contenedores
docker network create tienda-network
```

5.2 Lanzar MySQL en la Red

```
docker run -d \
  --name tienda-db \
  --network tienda-network \
  -e MYSQL_ROOT_PASSWORD=rootpassword \
  -e MYSQL_DATABASE=tienda \
  -v $(pwd)/database/init.sql:/docker-entrypoint-initdb.d/init.sql \
  mysql:8.0

# Esperar a que MySQL esté listo
docker logs -f tienda-db
```

5.3 Construir la Imagen del Backend

```
# Construir la imagen desde el Dockerfile
docker build -t tienda-backend ./backend
```

5.4 Lanzar el Backend

```
docker run -d \
  --name tienda-api \
  --network tienda-network \
  -e DB_HOST=tienda-db \
  -e DB_USER=root \
  -e DB_PASSWORD=rootpassword \
  -e DB_NAME=tienda \
  -p 3000:3000 \
  tienda-backend

# Ver los logs
docker logs tienda-api
```

5.5 Probar la API

```
# Ver información de la API
curl http://localhost:3000/

# Verificar conexión con la base de datos
curl http://localhost:3000/api/health

# Obtener todos los productos
curl http://localhost:3000/api/productos
```

```
# Obtener un producto específico
curl http://localhost:3000/api/productos/1

# Crear un producto nuevo

curl -X POST http://localhost:3000/api/productos \
  -H "Content-Type: application/json" \
  -d '{"nombre": "Producto Test", "precio": 99.99, "stock": 10}'

# Eliminar un producto
curl -X DELETE http://localhost:3000/api/productos/1
```

5.6 Limpiar

```
# Eliminar contenedores
docker rm -f tienda-api tienda-db

# Eliminar red
docker network rm tienda-network

# Eliminar imagen
docker rmi tienda-backend
```

📝 Ejercicios - Backend

1. Construye la imagen del backend. ¿Cuánto ocupa? (usa `docker images`)
2. Lanza MySQL y el backend conectados a una red.
3. Usa curl para obtener la lista de productos.
4. Crea un producto nuevo y verifica que aparece en la lista.
5. Limpia todos los contenedores y la red.

Parte 6: Lanzar Todo con Docker Compose

Docker Compose permite lanzar todos los servicios con un solo comando.

6.1 Construir y Levantar

```
# Construir las imágenes y levantar los servicios
docker compose up --build -d
```

6.2 Verificar el Estado

```
# Ver los servicios corriendo
docker compose ps

# Ver los logs de todos los servicios
docker compose logs
```

```
# Ver logs en tiempo real
docker compose logs -f

# Ver logs de un servicio específico
docker compose logs backend
```

6.3 Probar la Aplicación

1. **Frontend**: Abre `http://localhost:8080` en tu navegador
2. **Backend**: Abre `http://localhost:3000/api/productos`
3. Pulsa "Cargar Productos" en el frontend

6.4 Comandos Útiles

```
# Parar los servicios (sin eliminar)
docker compose stop

# Iniciar los servicios
docker compose start

# Reiniciar un servicio
docker compose restart backend

# Entrar a un contenedor
docker compose exec backend sh
docker compose exec database mysql -u root -prootpassword

# Ver uso de recursos
docker compose stats
```

6.5 Parar y Limpiar

```
# Parar y eliminar contenedores
docker compose down

# Parar, eliminar contenedores Y volúmenes (borra datos)
docker compose down -v
```

📝 Ejercicios - Docker Compose

1. Levanta toda la aplicación con `docker compose up --build -d`.
2. ¿Cuántos contenedores se han creado?
3. Accede al frontend y carga los productos. ¿Cuántos hay?
4. Añade un producto nuevo desde el frontend.
5. Entra al contenedor de la base de datos y verifica que el producto existe.
6. Para el contenedor de solo el backend. ¿Qué pasa en el frontend?

7. Vuelve a iniciar el backend.
8. Elimina todo con `docker compose down -v` y vuelve a levantar. ¿Siguen los productos? ¿Por qué?

Resumen de Comandos

Comando	**Descripción**
`docker pull imagen`	Descargar imagen
`docker images`	Listar imágenes
`docker run -d --name X imagen`	Crear contenedor
`docker ps`	Ver contenedores activos
`docker logs nombre`	Ver logs
`docker exec -it nombre sh`	Entrar al contenedor
`docker stop nombre`	Parar contenedor
`docker rm nombre`	Eliminar contenedor
`docker build -t nombre ./carpeta`	Construir imagen
`docker network create nombre`	Crear red
`docker compose up -d`	Levantar servicios
`docker compose down`	Parar y eliminar
`docker compose logs`	Ver logs

Entregables

1. **Capturas de pantalla**:

   - `docker --version` funcionando
   - `docker compose ps` con los 3 contenedores corriendo
   - Frontend funcionando en el navegador
   - Resultado de `curl http://localhost:3000/api/productos`

2. **Respuestas a los ejercicios** de cada sección en texto o captura de pantalla si procede.
3. Guarda el documento con respuestas en PDF y entrega.