

# Docker Básico

---

**Arturo Silvelo**

Try New Roads

# Ejercicios Contenedores

## Objetivo de la práctica

---

Aprender a gestionar contenedores Docker mediante la creación, modificación y manipulación de un contenedor NGINX.

## Pasos a seguir

---

1. Limpiar cualquier recurso previo (contenedores, imágenes, volúmenes, redes).
2. Crear un contenedor NGINX que sirva contenido web en un puerto aleatorio.
3. Acceder al contenedor para modificar el archivo `index.html` y personalizar el contenido.
4. Copiar el contenido de la carpeta HTML del contenedor al sistema host.
5. Crear un nuevo contenedor NGINX utilizando la carpeta copiada como su contenido web.

## Notas

---

- Si estás utilizando PowerShell en Windows, el comando `pwd` no funcionará como en sistemas basados en Unix. En PowerShell, debes usar `${PWD}` para obtener la ruta del directorio actual.
- También puedes usar una ruta relativa o absoluta en lugar de `pwd` para especificar la ubicación de la carpeta que deseas montar en el contenedor.

## Recursos adicionales

---

Si tienes alguna pregunta o necesitas más información sobre cómo trabajar con contenedores en Docker, consulta la **documentación oficial de Docker**.

# Ejercicios Redes

---

## Enunciado:

Tienes dos imágenes Docker disponibles en Docker Hub:

- `ghcr.io/trynewroads/course-frontend:latest`
- `ghcr.io/trynewroads/course-backend:latest`

El objetivo de este ejercicio es desplegar ambos contenedores y configurarlos para que se comuniquen entre sí utilizando una red personalizada en Docker.



1. Crea una red personalizada llamada `course-network` que permita la comunicación entre los contenedores.
2. Inicia un contenedor basado en la imagen `course-backend`, conéctalo a la red `course-network`, dale un nombre de host y al contenedor.
3. Inicia un contenedor basado en la imagen `course-frontend` y conéctalo también a la red `course-network`.
  - Cambiar la configuración del nginx para que se conecte al servidor. [Mirar notas](#)
4. Asegúrate de que la aplicación frontend pueda acceder al backend a través del nombre del servicio (`course-backend`) en lugar de una dirección IP.

## Notas

---

- Usuario de la interfaz es `admin` y su contraseña es `12345678`
- El fichero de configuración de nginx puedes obtenerlo de `nginx.conf`
- Al montar directorios pueden aparecer problemas de permisos de ficheros, para ello se puede usar `--user $(id -u):$(id -g)` para que el contenedor utilice el mismo usuario del host.

# Ejercicio: Volúmenes

Cada uno de los ejercicios parten del ejercicio de redes donde se conectan las aplicaciones `course-backend` y `course-frontend`,

## Ejercicio 1: Anonymous

---

1. Crea una nueva instancia de postgres con un volumen anónimo e inspecciona la información de los volúmenes.
2. Inserta algunas entradas de ejemplo en la base de datos de postgres.
3. Elimina el contenedor de postgres y crea uno nuevo.

## Ejercicio 2: Named Volumes

---

1. Crea un nuevo contenedor de postgres usando un volumen de nombrado.
2. Inserta algunas entradas de ejemplo en la base de datos de postgres.
3. Elimina el contenedor de postgres y crea uno nuevo.

## Ejercicio 3: Bind Mounts

1. Haz una copia del volumen en un directorio local.
2. Creamos una nueva máquina de postgres con un volumen de tipo `bind`.
3. Verifica que ambas bases de datos tengan los mismos datos.

# Notas

---

- El volume de datos para postgres se encuentra en el siguiente directorio:

```
/var/lib/postgresql/data
```

- Para la realización de estos ejercicios es necesario modificar el backend para que use una base de datos. Para esto tenemos que configurar las siguiente variables al iniciar el contenedor.

```
USE_DB=false  
DB_HOST=localhost  
DB_PORT=5432  
DB_USER=todo_user  
DB_PASS=todo_pass  
DB_NAME=todo_db
```



```
docker run
-d # Segundo plano
-p 3000:3000 # Conectar el puerto 3000 del host con el 3000 del contenedor
--network course-network # Asignar la red
--hostname course-backend # Nombre en la red
--name cb # Nombre del contenedor
-e USE_DB=true # Variables de entorno
-e DB_HOST=course-database
-e DB_PORT=5432
-e DB_USER=postgres
-e DB_PASS=12345678
-e DB_NAME=postgres
ghcr.io/trynewroads/course-backend:1.0.0 # imagen usada
```

Al finalizar los ejercicios es recomendable volver al estado del ejercicio 2, con la base de datos en un disco nombrado.

# Ejercicio 1: Crear una Imagen Docker para producción

Crear una imagen Docker para la aplicación `backend`

1. Descargar el repositorio

```
git clone git@github.com:trynewroads/course-backend.git
```

2. Crear un archivo `Dockerfile`

3. Usar `node:20` como imagen base

4. Establecer un directorio de trabajo (`/app`)

5. Copiar fichero de dependencias (`package.json`)

6. Instalar las dependencias (`npm install`)

7. Definir **variables** y configurar puerto (3000)
8. Copiar todo el contenido
9. Compilar la aplicación (`npm run build`)
10. Comando para iniciar la aplicación (`node dist/main.js`)
11. Crear la imagen (`docker build`)
12. Probar la imagen (`docker run`)

## Ejercicio 2: Crea una imagen de docker para desarrollo

Crear una imagen Docker para la aplicación `backend`

1. Descargar el repositorio

```
git clone git@github.com:trynewroads/course-backend.git
```

2. Crear un archivo `Dockerfile.dev`

3. Usar `node:20` como imagen base

4. Establecer un directorio de trabajo (`/app`)

5. Copiar fichero de dependencias (`package.json`)

6. Instalar las dependencias (`npm install`)

7. Definir **variables** y configurar puerto (3000)
8. Copiar todo el contenido
9. Definir volume (en la carpeta `app`)
10. Comando para iniciar la aplicación (`npm run start:dev`)
11. Crear la imagen (`docker build`)
12. Probar la imagen (`docker run`)

# Ejercicio Docker Compose

Crear la estructura que llevamos usando hasta ahora pero en formato `compose` .



## Ejercicio 1: Crear Compose de backend

---

1. Crear un fichero `docker-compose.yml` ó `compose.yml`

## 2. Configurar el servicio backend

```
docker run -d \  
-p 3005:3000 \  
--network course-network \  
--hostname course-compose-backend \  
--name ccb \  
-e USE_DB=false \  
ghcr.io/trynewroads/course-backend:1.0.0
```

3. Levantar el `compose`
4. Comprobar que el servicio.

## Ejercicio 2: Crear Compose del frontend

1. Editar el fichero `docker-compose.yml` ó `compose.yml`

## 2. Configurar el servicio frontend

```
docker run \  
-d \  
-p 8080:80 \  
--network course-network \  
--hostname course-compose-frontend \  
-v ./nginx/default.conf.template:/etc/nginx/templates/default.conf.template:ro \  
--name ccf \  
ghcr.io/trynewroads/course-frontend:1.0.0
```

3. Levantar el `compose`

4. Comprobar que el servicio.

## Ejercicio 3: Crear Compose de la base de datos

1. Editar el fichero `docker-compose.yml` ó `compose.yml`

## 2. Configurar el servicio postgres

```
docker run
-d
--name cdb
-e POSTGRES_PASSWORD=12345678
--hostname course-database
--network course-network
-v postgres_data:/var/lib/postgresql/data postgres
```



3. Levantar el `compose`

4. Comprobar que el servicio.

## Ejercicio 4: Mejorar compose

1. Establecer redes independientes para aislar los servicios que se comuniquen entre ellos. ( Crear dos redes `course-compose-public` y `course-compose-private` )
2. Establecer dependencia entre los servicios para que se inicien en orden ( `depends_on` )
3. Eliminar la exposición de puertos no necesarios ( `puerto de backend` )
4. Limpiar sistema

