# 04 - Redes Docker

**Conectar contenedores entre sí**

## Objetivo

Aprender a gestionar redes Docker para conectar contenedores entre sí.

## ¿Qué aprenderás?

- Tipos de redes Docker
- Cómo conectar contenedores en la misma red
- Comunicación entre contenedores por nombre

## ¿Por qué Redes en Docker?

Los contenedores necesitan **comunicarse** entre sí:

- Base de datos ↔️ Aplicación
- Frontend ↔️ Backend
- Múltiples servicios trabajando juntos

**Sin red:** Contenedores aislados, no pueden comunicarse

# Tipos de Redes

## 1. Bridge (por defecto)

Red interna de Docker. Los contenedores pueden comunicarse por IP.

- **Uso**: Comunicación entre contenedores
- **Aislamiento**: Separado del host

## 2. Host

El contenedor usa la red del host directamente.

- **Uso**: Máximo rendimiento
- **Limitación**: Solo Linux

## 3. None

Sin conectividad de red.

- **Uso**: Máxima seguridad, sin red

## Red Personalizada

**Ventaja principal:** Los contenedores se comunican por **nombre** en lugar de IP.

**Ejemplo:**

- ❌ Sin red personalizada: `http://172.17.0.2:5432`
- ✅ Con red personalizada: `http://postgres-db:5432`

## Ejercicio 1: Red Bridge por Defecto

```
# Crear dos contenedores
docker run -d --name web-server nginx:latest
docker run -d --name web-server-2 nginx:latest

# Ver en qué red están
docker network inspect bridge
```

**Limitación:** No pueden comunicarse por nombre (solo por IP)

# Ejercicio 2: Red Personalizada

## Crear una red personalizada

```
# Crear una red llamada "mi-red"
docker network create mi-red

# Ver redes disponibles
docker network ls

# Inspeccionar la red
docker network inspect mi-red
```

## Conectar Contenedores a la Red

```
# Ejecutar contenedores en la red personalizada
docker run -d --name app-1 --network mi-red nginx:latest
docker run -d --name app-2 --network mi-red nginx:latest

# Ahora pueden comunicarse por nombre
docker exec app-1 ping -c 3 app-2
docker exec app-2 ping -c 3 app-1
```

**Ventaja:** Comunicación por nombre de contenedor

# Ejercicio 3: Aplicación con Base de Datos

## Crear red para la aplicación

```
docker network create app-network
```

## Ejecutar base de datos

```
docker run -d \
  --name postgres-db \
  --network app-network \
  -e POSTGRES_PASSWORD=password \
  -e POSTGRES_DB=mi_app \
  postgres:15
```

## Aplicación se conecta usando el nombre

```
# app.py
conn = psycopg2.connect(
    host='postgres-db',  # Nombre del contenedor
    database='mi_app',
    user='postgres',
    password='password'
)
```

# Ejercicio 4: Red Host

## Usar red del host

```
# El contenedor usa directamente la red del host
docker run --rm --network host nginx:latest

# Accesible directamente en localhost:80
# (sin necesidad de -p)
```

**Nota:** Solo funciona en Linux. En macOS/Windows usa bridge.

## Gestionar Redes

```
# Listar redes
docker network ls

# Inspeccionar una red
docker network inspect mi-red

# Desconectar un contenedor de una red
docker network disconnect mi-red app-1

# Conectar un contenedor a una red
docker network connect mi-red app-1

# Eliminar una red
docker network rm mi-red

# Eliminar redes no usadas
docker network prune
```

## Comandos Clave

| Comando | Descripción |
| --- | --- |
| `docker network create` | Crear red |
| `docker network ls` | Listar redes |
| `docker network inspect` | Ver detalles |
| `--network` | Especificar red al ejecutar |
| `docker network connect/disconnect` | Conectar/desconectar |

## Mejores Prácticas

✅ Usa redes personalizadas para aislar aplicaciones
✅ Los contenedores se comunican por nombre en la misma red
✅ No expongas puertos innecesarios al host
✅ Usa alias para hacer el código más legible

## Práctica

1. Crea una red personalizada
2. Conecta múltiples contenedores
3. Verifica la comunicación por nombre
4. Gestiona redes

## Siguiente Paso

Aprende Docker Compose para orquestar múltiples contenedores.

**Módulo 05: Docker Compose Básico**

## Preguntas?

¡Tiempo para preguntas y práctica!