

# Docker Básico

---

**Arturo Silvelo**

Try New Roads

# Docker Compose

## ¿Qué es Docker Compose?

---

Es una herramienta para definir y ejecutar aplicaciones de múltiples contenedores a partir de un único archivo de configuración YAML.

- **Ventajas:** Simplifica la gestión de servicios, redes y volúmenes en un solo archivo, facilitando el despliegue de tu stack de aplicaciones de forma eficiente.

- **Uso:** Con un solo comando, puedes crear y arrancar todos los servicios configurados en el archivo `docker-compose.yml`, ideal para entornos de producción, desarrollo, testing y CI/CD.

## ¿Cómo Funciona Docker Compose?

---

- Docker Compose utiliza un archivo YAML (`compose.yaml`) para definir la configuración de tu aplicación y sus servicios.
- Sigue las reglas establecidas por la *Compose Specification* (**especificación completa**).
- También se soportan archivos `docker-compose.yml` para compatibilidad con versiones anteriores.

## Configuración compose

---

En un archivo `compose.yaml`, las claves principales son:

- **services:** Define los servicios (contenedores) que forman tu aplicación. Cada servicio puede tener su propia imagen, variables de entorno, puertos, volúmenes, etc.
- **image:** Especifica la imagen de Docker que se usará para crear el contenedor del servicio. Puede ser una imagen pública, privada o una construida localmente.
- **environment:** Permite establecer variables de entorno para los servicios.

- **container\_name**: Permite asignar un nombre personalizado al contenedor que se crea para el servicio, facilitando
- **ports**: Expone y mapea puertos del contenedor al host.
- **depends\_on**: Indica dependencias de arranque entre servicios.
- **volumes**: Define volúmenes persistentes para almacenar datos fuera del ciclo de vida de los contenedores.
- **networks**: Permite definir redes personalizadas para que los servicios se comuniquen entre sí de forma aislada o con el exterior.

# Comandos Comunes de Docker Compose

---

`docker compose` ([referencia](#)) para gestionar aplicaciones multicontenedor.

- `docker compose up`: Inicia todos los servicios definidos en el archivo.
- `docker compose down`: Detiene y elimina todos los contenedores, redes y volúmenes creados por `up`.
- `docker compose logs`: Muestra los logs de todos los servicios.
- `docker compose ps`: Lista todos los contenedores que se están ejecutando.

Puedes invocar contenedores de manera individual usando `docker compose` `<command> <nombre del servicio>`.



# Creando un archivo `compose.yaml` básico

Creación de un contenedor con `docker run`

```
docker run \
-d \
-p 3000:3000 \
--network course-network \
--hostname course-backend \
--name cb \
-e USE_DB=true \
-e DB_HOST=course-database \
-e DB_PORT=5432 \
-e DB_USER=postgres \
-e DB_PASS=12345678 \
-e DB_NAME=postgres \
course-backend
```

## Convertir la configuración a un compose

```
services:
  course-backend:
    image: course-backend
    container_name: cb
    hostname: course-backend
    networks:
      - course-network
    ports:
      - "3000:3000"
    environment:
      - USE_DB=true
      - DB_HOST=course-database
      - DB_PORT=5432
      - DB_USER=postgres
      - DB_PASS=12345678
      - DB_NAME=postgres

networks:
  course-network:
    external: true
```