05 - Docker Compose Básico

Orquestar múltiples contenedores

Objetivo

Aprender a usar Docker Compose para orquestar múltiples contenedores con un solo archivo.

¿Qué es Docker Compose?

Herramienta para definir y ejecutar aplicaciones **multi-contenedor**.

Usa un archivo **YAML** para configurar los servicios.

Problema que resuelve: En lugar de ejecutar múltiples comandos `docker run`, defines todo en un archivo.

Ventajas de Docker Compose

✅ **Un solo comando**: `docker compose up`
✅ **Configuración centralizada**: Todo en un archivo
✅ **Dependencias automáticas**: Ordena el inicio
✅ **Reproducible**: Mismo entorno siempre

Antes:

```
docker run -d --name db postgres
docker run -d --name web --link db nginx
```

Ahora:

```
docker compose up
```

¿Qué aprenderás?

- Sintaxis de `compose.yml` o `docker-compose.yml`
- Cómo definir servicios, volúmenes y redes
- Comandos básicos de Docker Compose

Estructura del Proyecto

```
05-docker-compose-basico/
├── docker-compose.yml
├── app/
│   ├── Dockerfile
│   ├── app.py
│   └── requirements.txt
```

docker-compose.yml

```yaml
services:
  web:
    build: ./app
    ports:
      - "5000:5000"
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
      POSTGRES_DB: mi_app
    volumes:
      - postgres-data:/var/lib/postgresql/data

volumes:
  postgres-data:
```

Explicación del Archivo

- **services**: Define los contenedores
- **web**: Servicio de la aplicación
  - `build`: Construye desde Dockerfile
  - `ports`: Mapeo de puertos
  - `depends_on`: Dependencias
- **db**: Servicio de base de datos
- **volumes**: Volúmenes nombrados

Comandos Básicos

Iniciar servicios

```
docker compose up
docker compose up -d  # Segundo plano
```

Ver estado

```
docker compose ps
docker compose logs
docker compose logs web
```

Detener servicios

```
docker compose stop
docker compose down
docker compose down -v  # Eliminar volúmenes
```

Variables de Entorno

Crear archivo .env

```
POSTGRES_USER=postgres
POSTGRES_PASSWORD=password
POSTGRES_DB=mi_app
```

Usar en docker-compose.yml

```yaml
services:
  db:
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
```

Healthchecks

```yaml
services:
  db:
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 10s
      timeout: 5s
      retries: 5

  web:
    depends_on:
      db:
        condition: service_healthy
```

Comandos Útiles

```
# Ejecutar comando en servicio
docker compose exec web python manage.py migrate

# Reconstruir servicio
docker compose build web

# Reiniciar servicio
docker compose restart web

# Validar configuración
docker compose config
```

Buenas Prácticas

- ✅ Usa `depends_on` para ordenar inicio
- ✅ Usa healthchecks para dependencias críticas
- ✅ Separa archivos para dev/prod
- ✅ Usa variables de entorno
- ✅ Define redes y volúmenes explícitamente

Práctica

1. Crea un `docker-compose.yml`
2. Define servicios web y base de datos
3. Inicia los servicios
4. Verifica la comunicación
5. Detén y elimina los servicios

Siguiente Paso

Aprende sobre multi-stage builds para optimizar tus imágenes.

Módulo 06: Multi-Stage Builds

Preguntas?

¡Tiempo para preguntas y práctica!