

Introducción a Docker

UD 06. Caso práctico 07 - Desplegando Odoo



Autor: Sergi García Barea

Actualizado Febrero 2025

Licencia





Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 **Importante**

 **Atención**

 **Interesante**

1. Introducción	3
2. Diferencias clave entre modo producción y modo desarrollo	3
3. Fichero "docker-compose.yml" para desarrollo	4
4. Fichero "docker-compose.yml" para producción	5
5. Poniendo en marcha el sistema (para cualquiera de las dos configuraciones)	6
6. Bibliografía	6

UD06. CASO PRÁCTICO 07

1. INTRODUCCIÓN

Odoo es un sistema de gestión empresarial (ERP) y CRM de código abierto que proporciona una amplia gama de aplicaciones para la gestión de negocios, incluyendo contabilidad, ventas, inventario, fabricación, recursos humanos y más. Su flexibilidad y modularidad permiten a las empresas adaptar su funcionamiento a sus necesidades específicas.

En este caso práctico, trabajaremos con **Odoo Community Edition**, la versión gratuita y de código abierto de Odoo, que ofrece muchas funcionalidades clave sin necesidad de una suscripción de pago. Más información y documentación oficial se puede encontrar en su página web: <https://www.odoo.com/>.

Vamos a desplegar Odoo utilizando Docker Compose con dos configuraciones distintas:

- **Modo Desarrollo:** Permite la modificación de archivos en caliente, lo que implica que cualquier cambio en los archivos del sistema se refleja de inmediato. Sin embargo, este modo es más lento debido a la necesidad de verificar constantemente los cambios.
- **Modo Producción:** Optimizado para el rendimiento, carga los archivos en memoria y no revisa cambios en disco hasta que se reinicia el servicio.

Ambos despliegues contarán con un servicio de base de datos PostgreSQL.

! Atención: para realizar el caso práctico, con que pongáis en marcha uno de los dos supuestos, es suficiente.

2. DIFERENCIAS CLAVE ENTRE MODO PRODUCCIÓN Y MODO DESARROLLO

Característica	Desarrollo	Producción
Permite cambios en caliente	✓ Sí	✗ No
Rendimiento optimizado	✗ No	✓ Sí
Uso de volúmenes de configuración	✓ Sí	✗ No
Recarga automática de archivos	✓ Sí	✗ No

Interesante: elegir el modo correcto depende del entorno en el que se despliegue Odoo. Para desarrollo, la flexibilidad es clave, mientras que en producción se prioriza estabilidad y rendimiento.

3. FICHERO "DOCKER-COMPOSE.YML" PARA DESARROLLO

A continuación mostramos el fichero "docker-compose.yml" comentado para hacer un despliegue de Odoo en modo desarrollo:

```
version: '3.3'

services:
  #Definimos el servicio Web, en este caso Odoo
  web:
    #Indicamos que imagen de Docker Hub utilizaremos
    image: odoo:17
    #Indicamos que depende de "db", por lo cual debe ser procesada primero "db"
    depends_on:
      - db

    # Port Mapping: indicamos que el puerto 8069 del contenedor se mapeara con el
    # mismo puerto en el anfitrión
    # Permitiendo acceder a Odoo mediante http://localhost:8069
    ports:
      - 8069:8069

    # Mapeamos el directorio de los contenedores (como por ejemplo"
    # /mnt/extra-addons" )
    # en un directorio local (como por ejemplo en un directorio
    # "./volumesOdoo/addons")
    # situado en el lugar donde ejecutemos "Docker compose"
    volumes:
      - ./volumesOdoo/addons:/mnt/extra-addons
      - ./volumesOdoo/odoo/filestore:/var/lib/odoo/filestore
      - ./volumesOdoo/odoo/sessions:/var/lib/odoo/sessions
    #Indicamos que el contenedor funcionara con usuario root y no con usuario odoo
    user: root
    # Definimos variables de entorno de Odoo
    environment:
      - HOST=db
      - USER=odoo
      - PASSWORD=odoo
    # Indica que pasa ese parametro al arrancar el servicio Odoo
    command: --dev=all
  #Definimos el servicio de la base de datos
  db:
    image: postgres:15

    # Definimos variables de entorno de PostgreSQL
    environment:
      - POSTGRES_PASSWORD=odoo
      - POSTGRES_USER=odoo
      - POSTGRES_DB=postgres
    # Mapeamos el directorio del contenedor "var/lib/postgresql/data" en un
    # directorio "./volumesOdoo/dataPostgreSQL"
    # situado en el lugar donde ejecutemos "Docker compose"
```

```
volumes:  
- ./volumesOdoo/dataPostgreSQL:/var/lib/postgresql/data
```

4. FICHERO "DOCKER-COMPOSE.YML" PARA PRODUCCIÓN

A continuación mostramos el fichero "docker-compose.yml" comentado para hacer un despliegue de Odoo en modo producción:

```
version: '3.3'  
  
services:  
#Definimos el servicio Web, en este caso Odoo  
  web:  
    #Indicamos que imagen de Docker Hub utilizaremos  
    image: odoo:17  
    #Indicamos que depende de "db", por lo cual debe ser procesada primero "db"  
    depends_on:  
      - db  
  
    # Port Mapping: indicamos que el puerto 8069 del contenedor se mapeara con el  
    # mismo puerto en el anfitrión  
    # Permitiendo acceder a Odoo mediante http://localhost:8069  
    ports:  
      - 8069:8069  
  
    # Mapeamos el directorio de los contenedores (como por ejemplo "  
    # /mnt/extra-addons" )  
    # en un directorio local (como por ejemplo en un directorio  
    # "./volumesOdoo/addons")  
    # situado en el lugar donde ejecutemos "Docker compose"  
    volumes:  
      - ./volumesOdoo/addons:/mnt/extra-addons  
    #Indicamos que el contenedor funcionara con usuario root y no con usuario odoo  
    user: root  
    # Definimos variables de entorno de Odoo  
    environment:  
      - HOST=db  
      - USER=odoo  
      - PASSWORD=odoo  
#Definimos el servicio de la base de datos  
  db:  
    image: postgres:15  
  
    # Definimos variables de entorno de PostgreSQL  
    environment:  
      - POSTGRES_PASSWORD=odoo  
      - POSTGRES_USER=odoo  
      - POSTGRES_DB=postgres  
    # Mapeamos el directorio del contenedor "var/lib/postgresql/data" en un  
    # directorio "./volumesOdoo/dataPostgreSQL"
```

```
# situado en el lugar donde ejecutemos "Docker compose"
volumes:
- ./volumesOdoo/dataPostgreSQL:/var/lib/postgresql/data
```

5. PONIENDO EN MARCHA EL SISTEMA (PARA CUALQUIERA DE LAS DOS CONFIGURACIONES)

Para iniciar Odoo en cualquiera de los dos modos, nos situamos en el directorio donde se encuentra el fichero docker-compose.yml y ejecutamos:

```
docker compose up -d
```

Esto descargará las imágenes (si es necesario) y lanzará los contenedores. Una vez iniciado, podemos acceder a Odoo desde <http://localhost:8069>.

Para detener y eliminar los contenedores sin afectar a los volúmenes de datos:

```
docker compose stop
```

! Atención: usamos “docker compose stop” porque solo para los contenedores. Un comando similar pero no igual, “docker compose down” o que hace además de parar los contenedores, los borra, aunque no borra la persistencia.

Para reiniciar Odoo tras haberlo detenido:

```
docker compose up -d
```

o si ya están creados

```
docker compose start
```

Si se han hecho cambios en la configuración del modo desarrollo, estos se aplicarán directamente sin necesidad de reconstruir la imagen, aunque si da problemas o estás en modo producción siempre puedes pararlo y ponerlo en marcha con

```
docker compose stop && docker compose start
```

6. BIBLIOGRAFÍA

- [1] Docker Docs <https://docs.docker.com/>
- [2] Docker Compose Docs <https://docs.docker.com/compose/>
- [3] Materiales libres del módulo SGE donde se utiliza Odoo
<https://github.com/sergarb1/ApuntesSistemasGestionEmpresarial>