4. Implantación de Odoo

Cómo empezar a trabajar con docker y odoo

Para la realización de esta práctica en la máquina virtual con Odoo y docker instalado, lo mejor es parar el servicio de odoo que arranca por defecto (*sudo service odoo stop*).

1. Descarga y arranque de los servicios básicos

- **docker run**: crea un contenedor a partir de una imagen y lo arranca. Si haces varios *docker run*, estás creando varios contenedores, no arrancando varias veces un contenedor. Para crear un contenedor sin arrancarlo existe el comando **docker create**

Enlace a más información de docker run

- **Base de datos postgres**:

```
docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres:15
```

- **Servicio de odoo**

```
docker run -d -p 18069:8069 --name odoo --link db:db -t odoo
```

- Opciones de docker run utilizadas:
  - -d: segundo plano
  - -e: asigna variables de entorno
  - * -name: nombre del contenedor
  - -p: mapeo de puertos **host_port:container_port**
  - * -link: conexión con otro contenedor
  - -t: uso para un terminal

2. Utilidades para gestión de los contenedores

- **docker ps**: permite ver los contenedores activos. Con la opción **-a** lista también los inactivos

```
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
                                                    NAMES
1af4611db689   odoo           "/entrypoint.sh odoo"    4 minutes ago    Up 4 minutes    8069/tcp,
8071-8072/tcp, 0.0.0.0:8069->18069/tcp, :::18069->8069/tcp   odoo
a4ce6bb84c53   postgres:15    "docker-entrypoint.s..."   20 minutes ago   Up 20 minutes   5432/tcp
                                                          db
```

- **docker start/stop/restart**: inicia, para o reinicia el contenedor, indicando el id o el nombre del contenedor.

```
docker stop 1af4611db689
docker start odoo
```

- **docker exec**: sirve para ejecutar comandos en un contenedor en ejecución. La opción **-it** enlaza el terminal que se está usando.

Para salir usamos el comando *exit*

```
$ docker exec -it odoo bash
odoo@1af4611db689:/$
odoo@1af4611db689:/$ ls /var/lib/odoo
addons
```

- **docker cp**: permite copiar archivos y directorios del anfitrión al contenedor.

Para realizar una copia usamos un módulo simple ya desarrollado y descomprimido y lo copiamos en la carpeta */mnt/extra-addons* que es la ruta configurada para módulos auxiliares. Usamos rutas absolutas para evitar errores de rutas.

```
# Sintaxis: docker cp origen destino
# copia del directorio kleenex (anfitrión) a addons (contenedor)
docker cp /home/alumno/Documentos/modulos/kleenex odoo:/mnt/extra-addons
```

Comprobamos el resultado con *docker exec*

- **docker rm**: borra el contenedor (no la imagen)

```
docker rm odoo
docker rm 1af4611db689
```

3. Pasos iniciales para configurar Odoo

Al iniciar un navegador con la url *localhost:18069* tenemos la pantalla inicial que permite crear una base de datos para nuestra plataforma.

Introducimos los datos:

- Database Name: ejemploDocker
- email: admin@odoo.com
- password: odoo
- Language: Spanish / Español
- Country: Spain
- Demo Data: NO

y creamos la base de datos.

Aparece la pantalla inicial que permite arrancar con Odoo tal como lo conocemos. Se pueden ir instalando las aplicaciones que necesitemos.

Para poder instalar y trabajar con nuestros módulos, instalamos uno de ejemplo, como **Conversaciones** (productividad). Posteriormente, copiamos el directorio del módulo descomprimido, e instalamos tras actualizar las aplicaciones.

Los logs los podemos ver en un nuevo terminal con la orden **docker logs -f odoo**

4. Persistencia de los datos

Mientras el contenedor del servidor de Odoo y el de la base de datos de Postgres no se destruyan se mantienen las aplicaciones instaladas y los datos generados. Se puede parar y reiniciar el contenedor.

Sin embargo, al eliminar el contenedor, se destruyen los datos que se hayan creados en él. Para evitar la pérdida de las configuraciones y los datos lo que se hace es asignar directorios del anfitrión al contenedor en la creación y arranque del contenedor.

Para evitar ésto, se asocian **volúmenes** a los contenedores, que no son más que directorios del anfitrión montados en los directorios adecuados en el contenedor.

- **Paso 1**: Paramos los contenedores que tengamos arrancados

```
docker stop odoo
docker stop db
```

- **Paso 2**: Para no interferir entre contenedores, renombramos los anteriores. Básicamente se trata de que las variables de entorno (db) permanezcan igual al ser utilizadas.

```
docker rename odoo odoo_v0
docker rename db db_v0
```

- **Paso 3**: Crear carpetas que serán las que guarden los datos en el anfitrión. Una para la base de datos *odoo-db* y otra para la plataforma de Odoo *odoo-data*. La ruta quedará parecida a */home/alumno/Documentos/volumenes/odoo-db*.
- **Paso 4**: Iniciar nuevos contenedores con **volúmenes** asignados. Primero el de la base de datos y posteriormente el de odoo.

```
docker run -d -v /home/alumno/Documentos/volumenes/odoo-db:/var/lib/postgresql/data
-e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres:15

(respuesta: id del contenedor) d58c60aed02a02f577829f6d27b8751eff2e3b4bf1581236d592a869cb43ecaf
```

```
docker run -v /home/alumno/Documentos/volumenes/odoo-data/firestore:/var/lib/odoo/filestore
-v /home/alumno/Documentos/volumenes/odoo-data/sessions:/var/lib/odoo/sessions
-d -p 18069:8069 --name odoo --user="root" --link db:db -t odoo --dev=all

(respuesta: id del contenedor) 05c5af5f6ad38b3d6df4f76770f28228bf9506a83a3f2f96b38944f2c484c273
```

Hay que montar los directorios **filestore** y **sessions**, y dar acceso con el usuario *root* para poder guardar los datos que se vayan generando.

- **Paso 5**: Crear la base de datos y empezar a configurar nuestra empresa.

Comprobación del funcionamiento

- Crear una nueva base de datos con datos de demostración
- Instalar una aplicación de ejemplo
- Parar los contenedores
- Renombrar los contenedores
- Volver a lanzar nuevos contenedores con las instrucciones anteriores

Hay que ser "paciente" en la parada y arranque de los contenedores ya que la base de datos se puede quedar inestable. (mirar los logs de los contenedores)

- **Paso 6**: Añadir módulos adicionales

En este paso montamos el directorio /mnt/extra-addons a la carpeta donde tengamos alojados nuestros módulos y aplicaciones. Se añade una nueva opción -v en la instrucción de arranque.

- Parar sólo el contenedor *odoo*
- Renombrarlo para que no haya conflictos de nombre (para no borrar)
- Iniciar con la instrucción:

```
docker run -v /home/alumno/Documentos/modulos:/mnt/extra-addons
  -v /home/alumno/Documentos/volumenes/odoo-data/firestore:/var/lib/odoo/filestore
  -v /home/alumno/Documentos/volumenes/odoo-data/sessions:/var/lib/odoo/sessions
  -d -p 18069:8069 --name odoo --user="root" --link db:db -t odoo --dev=all
```

5. Creación de un fichero de configuración (Docker Compose)

Para la realización de este apartado es necesario instalar la utilidad **docker-compose**

```
sudo apt install docker-compose
```

Todas las tareas anteriores se pueden reunir en un fichero de configuración para docker llamado **docker-compose.yml**

Como paso previo, creamos una carpeta **odoo-docker-compose** donde vamos a alojar el fichero *yml* y los volúmenes para la base de datos y los datos de odoo.

Creamos el fichero **docker-compose.yml**

```
version: '3.3'

services:

#Define el servicio Web, en este caso Odoo

  odoo:
    #Indicamos que imagen de Docker Hub utilizaremos
    image: odoo:17.0

    # Nombre del contenedor
    container_name: odoo

    # Reinicio automático (por si falla la conexión)
    restart: unless-stopped

    # Depende de "db", por lo cual debe ser arrancado primero "db"
    links:
      - db:db
    depends_on:
      - db

    # Port Mapping: indicamos que el puerto 8069 del contenedor se mapeara con el
    # puerto 18070 en el anfritrion
    ports:
      - "18070:8069"

    # Definimos variables de entorno de Odoo
    environment:
      - HOST=db
      - USER=odoo
      - PASSWORD=odoo

    # Mapeamos el directorio de los contenedores /mnt/extra-addons
    # en un directorio local /home/alumno/Documentos/modulos (como ejemplo)
    # verificar las rutas relativas al directorio donde se ejecute Docker Compose
    volumes:
      - odoo-data:/var/lib/odoo
      - /home/alumno/Documentos/modulos:/mnt/extra-addons


# Definimos el servicio de la base de datos
  db:
    image: postgres:15
    container_name: db
    restart: unless-stopped
```

```
    # Definimos variables de entorno de PostgreSQL
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_PASSWORD=odoo
      - POSTGRES_USER=odoo

    # Mapeamos el directorio del contenedor "var/lib/postgresql/data"
    # en el directorio "odoo-db"
    # situado donde se ejecuta Docker Compose
    volumes:
      - odoo-db:/var/lib/postgresql/data

# Declaración de los volúmenes que se utilizan
volumes:
  odoo-data:
  odoo-db:
```

Y ejecutamos la instrucción (para ir viendo los logs quitamos la opción -d)

```
docker-compose up -d
```

Para parar la ejecución y eliminar los contenedores se utiliza **docker-compose down**