Grado en Ingeniería Informática - 4º Curso

Sistemas de Negocio

Práctica 2: Docker y Odoo

Guillermo Blanco González

guillermo.blanco.gonzalez@uvigo.gal

Virtualización vs Contenedores

En virtualización:

- El hypervisor crea una abstracción de los recursos hardware disponibles → Máquina virtual (MV)
- Cada MV tiene instalado su propio SO completo y está completamente aislada del resto → Cada MV puede tener un SO diferente.

Virtualización vs Contenedores

La tecnología de contenedores:

- Aísla el SO de las aplicaciones → Crea un entorno ligero (contenedor) que permite ejecutar aplicaciones en el mismo SO del host.
- Los contenedores comparten el kernel con el host y añaden las aplicaciones del usuario →Aislamiento ligero.
- Todos los contenedores deben usar el mismo SO que el host.
- Requieren menos recursos (menos sobrecarga) que las máquinas virtuales.

Docker

Docker es una tecnología de contenedores para Linux.

Conceptos básicos:

- **Imagen**: plantilla (de solo lectura) que contiene las aplicaciones y librerías deseadas.
  - Se crean a partir de un fichero Dockerfile.

- **Contenedor**: instancia o ejecución de una imagen concreta.
  - Se pueden instanciar tantos contenedores como se quiera a partir de una imagen.

Docker - Imágenes

Docker Hub: repositorio de imágenes Docker (https://hub.docker.com/).

Descargar una imagen de Docker Hub:

```
docker pull ubuntu:20.04
```

Docker - Comandos básicos

Obtener un listado de imágenes

```
docker image ls
```

Ejecutar una imagen, es decir, crea un contenedor y lo ejecuta

```
docker run -ti <ubuntu> /bin/bash
```

Listar contenedores en ejecución

```
docker ps
```

Listar contenedores:

```
docker ps -a
```

Docker - Comandos de gestión

Eliminar un contenedor:

```
docker rm <container_id>
```

Eliminar una imagen:

```
docker rmi <image name>
```

Para que los contenedores se eliminen automáticamente y no se acumulen, añade el parámetro --rm al comando docker run. Con -ti dejamos activada la interfaz gráfica, sin ella el contenedor se ejecutaría en segundo plano

```
docker run -ti --rm ubuntu:20.04  /bin/bash
```

Docker - Comandos de gestión

Arrancar un contenedor, ya existente:

```
docker start <container_id> - i
```

Eliminar todos los contenedores parados:

```
docker container prune
```

Volúmenes

Una imagen Docker consta de varias capas de solo lectura. Cuando se inicia una imagen desde un contenedor, Docker añade una nueva capa editable en la parte superior del sistema. Si se elimina un contenedor, se pierde la capa más alta, la que tiene permisos de escritura. Esto significa que todos los cambios realizados después de iniciar el contenedor se pierden.

Un volumen de contenedor permite conservar los datos, aunque se elimine el Docker container. Los volúmenes también permiten un intercambio práctico de datos entre el host y el container.

Crear un volumen de Docker es una buena solución para poder:

- Transferir datos a un contenedor de Docker
- Guardar los datos de un contenedor de Docker
- Intercambiar datos entre contenedores de Docker

Volúmenes - Comandos de gestión

Crear un volumen:

```
docker volume create <volume-name>
```

Muestra todos los volúmenes creados:

```
docker volume rm <volume-name>
```

Instalación de Postges-Odoo

Para poder realizar la práctica en Odoo necesitamos tener una base de datos, operativa, como no queremos persistir los datos, necesitamos crear un volumen.

```
docker volume create postgres-data
```

Instalación de Postges-Odoo

Ahora vamos a arrancar el contenedor, estas son las opciones utilizadas:

- **-d**: con esta opción el contenedor se arrancará en segundo plano.
- **-v postgres-data:/var/lib/postgresql/data**: esta opción se utiliza para configurar un volumen. En este caso, se está conectando el contenedor al volumen "postgres-data" y se está montando en el directorio "/var/lib/postgresql/data" dentro del contenedor. Esto permite que los datos de PostgreSQL sean persistentes incluso si el contenedor se detiene o se elimina.
- **-e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres**: son variables propias del contenedor, en este caso para configurar la base de datos.

Instalación de Postges-Odoo

Opciones utilizadas:

- **--name db:** establece un nombre personalizado para el contenedor, en este caso “db”
- **postgres:** Este es el nombre de la imagen de Docker que se utilizará para crear el contenedor. Si no existe, Docker la buscará en el hub y la descargará.

Ejecutamos:

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

Instalación de Postges-Odoo

Con esto ya tendríamos un contenedor con una base de datos Postgres, ahora queremos un entorno, con Odoo instalado, el cual conectaremos a la base de datos del otro contenedor.

Ahora arrancaremos el contenedor de Odoo con las siguientes opciones:

- **-p 8069:8069**: esta opción se utiliza para mapear un puerto en el host a un puerto en el contenedor. En este caso, el puerto 8069 del host se mapea al puerto 8069 del contenedor.
- **--name odoo**: esta opción establece un nombre personalizado para el contenedor, en este caso, "odoo".
- **--link db:db**: Esta opción se utiliza para vincular el contenedor "odoo" a nuestro otro contenedor llamado "db".

Instalación de Postges-Odoo

Opciones utilizadas:

- **-t odoo**: La opción -t se utiliza para que puedas interactuar con la terminal del contenedor si es necesario. Esta opción habilita la interfaz, ya que, no nos interesa tenerla en segundo plano.

Ejecución:

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

Resultado

Creamos y comprobamos el volumen:

```
C:\Users\Guillermo>docker volume create postgres-data
postgres-data

C:\Users\Guillermo>docker volume ls
DRIVER    VOLUME NAME
local     postgres-data

C:\Users\Guillermo>
```

Resultado

Creación y ejecución de imagen y contenedor de Postgres:

```
C:\Users\Guillermo>docker run -d -v postgres-data:/var/lib/postgresql/data -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
a378f10b3218: Pull complete
2ebc5690e391: Pull complete
8fe57f734687: Pull complete
a2ddbb09cd9a: Pull complete
5a2499e87ab8: Pull complete
a45f5c4adf1b: Pull complete
178017fd978e: Pull complete
428dff1cb77d: Pull complete
4667364adfc4: Pull complete
4eea1f5281a9: Pull complete
369467411787: Pull complete
51184495a2bc: Pull complete
d3e246f01410: Pull complete
Digest: sha256:3d9ed832906091d609cfd6f283e79492ace01ba15866b21d8a262e8fd1cdfb55
Status: Downloaded newer image for postgres:latest
a137160168e0745e6f69738f51cc5d9fa7b2708ffdd565949ed186b2734764c2

C:\Users\Guillermo>docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
postgres     latest    f7d9a0d4223b   6 weeks ago   417MB

C:\Users\Guillermo>docker container ls
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS      NAMES
a137160168e0   postgres   "docker-entrypoint.s…"   15 seconds ago   Up 13 seconds   5432/tcp   db
```

Resultado

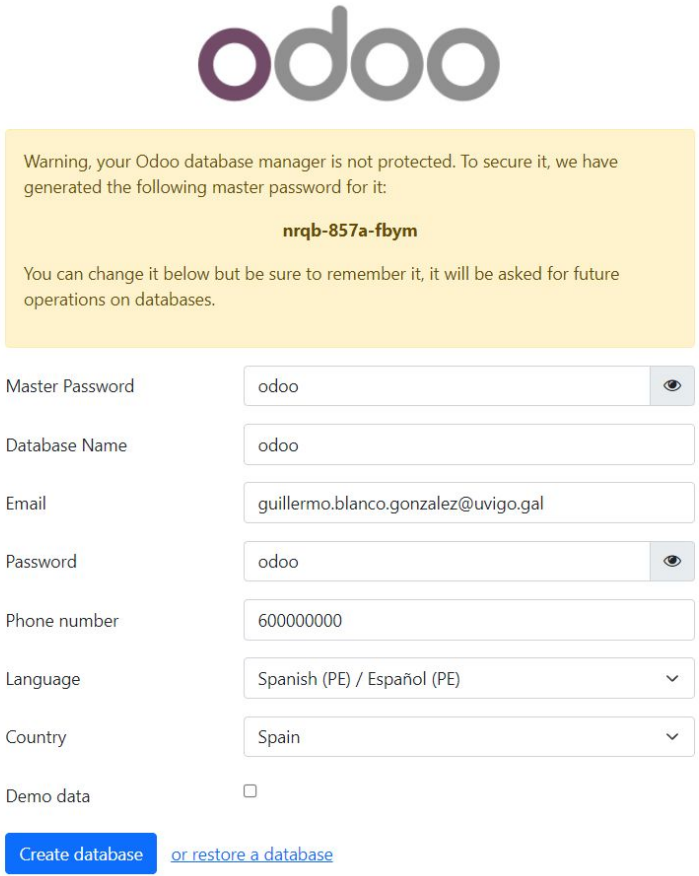
Creamos y ejecutamos contenedor de Odoo:

```
C:\Users\Guillermo>docker run -p 8069:8069 --name odoo --link db:db -t odoo
Unable to find image 'odoo:latest' locally
latest: Pulling from library/odoo
e67fdae35593: Pull complete
d361645e6e1c: Pull complete
d5302d28554b: Pull complete
79d21d45cc4a: Pull complete
bef9674e5578: Pull complete
c89b0de0a7ef: Pull complete
3f2f302451ec: Pull complete
da0c51ea4877: Pull complete
3fc2566bf1e4: Pull complete
Digest: sha256:01697e5fbab426474004fa6b8c49f0af7554b1fafe2d5c10d68f52461fccf490
Status: Downloaded newer image for odoo:latest
2023-10-31 12:09:19,372 1 INFO ? odoo: Odoo version 16.0-20230925
2023-10-31 12:09:19,372 1 INFO ? odoo: Using configuration file at /etc/odoo/odoo.conf
2023-10-31 12:09:19,372 1 INFO ? odoo: addons paths: ['/usr/lib/python3/dist-packages/odoo/addons', '/var/lib/odoo/addons/16.0', '/mnt/extra-addons']
2023-10-31 12:09:19,373 1 INFO ? odoo: database: odoo@172.17.0.2:5432
2023-10-31 12:09:19,680 1 INFO ? odoo.addons.base.models.ir_actions_report: Will use the Wkhtmltopdf binary at /usr/local/bin/wkhtmltopdf
2023-10-31 12:09:20,147 1 INFO ? odoo.service.server: HTTP service (werkzeug) running on cfe9147b91f1:8069
2023-10-31 12:10:14,729 1 INFO ? werkzeug: 172.17.0.1 - - [31/Oct/2023 12:10:14] "GET /websocket HTTP/1.1" 404 - 1 0.005 0.089
```

Configurando Odoo

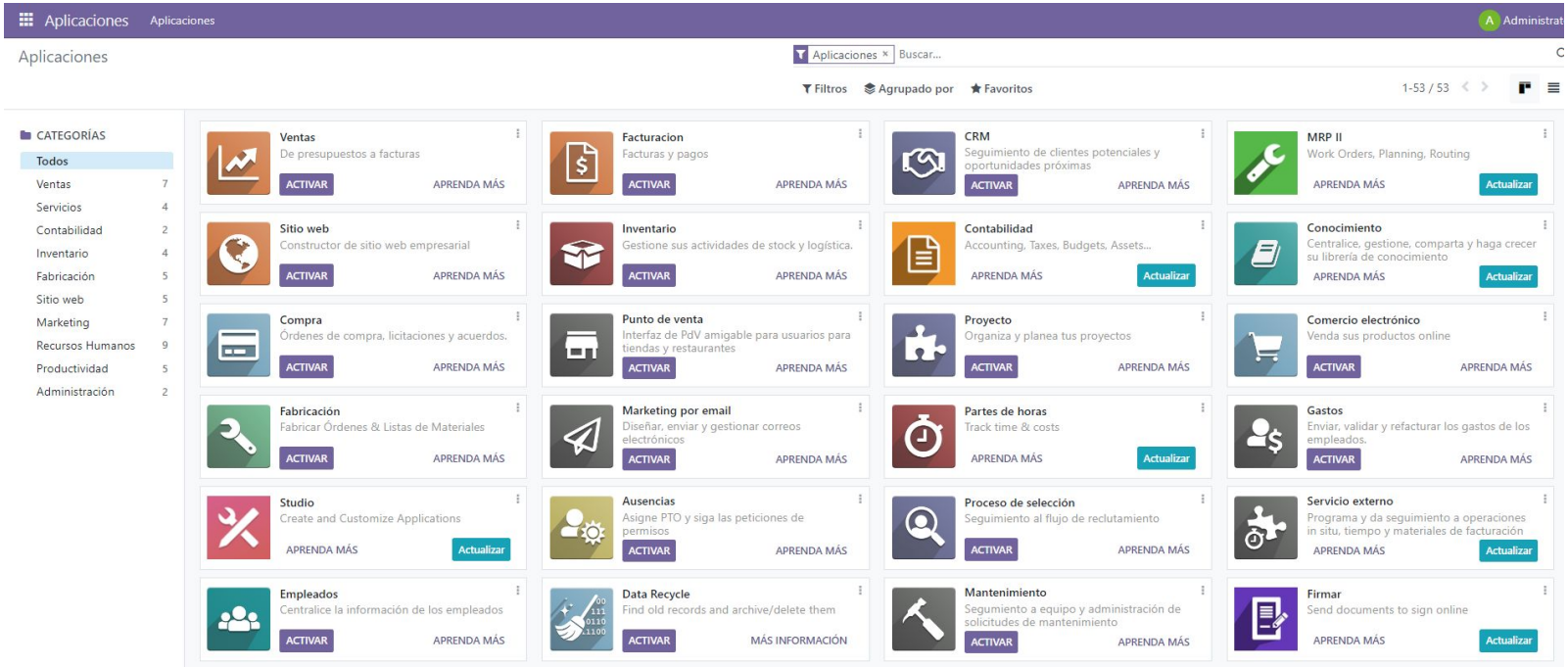
Si todo ha ido bien, deberíamos poder acceder a http://localhost:8069/

Configuramos la conexión a la base de datos y la información básica del administrador.



Configurando Odoo

Una vez dentro debería salir el siguiente panel:



Arrancar de nuevo el contenedor

Parar contenedor

```
docker stop <container_id>
```

Arrancar contenedor sin interfaz:

```
docker start <container_id>
```

Arrancar contenedor con interfaz:

```
docker exec -it <container_id> -i
```

Configurando Odoo

Para nuestras prácticas necesitamos las siguientes aplicaciones:

- Fabricación
- Facturación
- Compra
- Ventas

Backup y restauración de datos

Para realizar un backup debemos irnos a la siguiente URL:

http://localhost:8069/web/database/manager

Backup y restauración de datos

A la hora de restaurar los datos en el login, usamos la opción “restore database”, y cargamos el zip que descargamos en el backup:

http://localhost:8069/web/database/manager