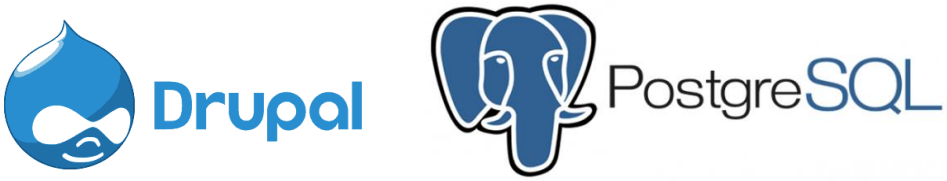
Prácticas Docker

1. Enlazar contenedores. Con --link

- Vamos a montar un enlace entre un Drupal (un gestor de contenidos open-source de los más usados y una Base de datos PosrtgreSQL



- Comprobamos las redes que tenemos en este momento y que vienen de la práctica anterior

```
docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3d8689b8a3ea        bridge              bridge              local
81ce05a3ba16        host                host                local
8e83268b846d        net1                bridge              local
31ed5d426215        net2                bridge              local
17052d6bd175        none                null                local
```

- En este caso vamos a usar la red bridge estándar, porque como hemos visto en el vídeo es necesario usar la forma “legacy” de conexión, con la opción –link.

- Descargamos el contenedor de drupal

```
docker pull drupal
Digest:
sha256:afec4ec454efc0079c00eb3e7ab99972567c15d9577f72377333eeca7e0ac62
5
Status: Downloaded newer image for drupal:latest
```

- Descargamos ahora la de PostgreSQL

```
docker pull postgres
```

- Vamos en primer lugar a arrancar el contenedor Postgresql y crear la Base de datos. Le tenemos que indicar un nombre y luego una variable de entorno para la password, en este caso he puesto “secret”

```
docker run -d --name postgresql1 -e POSTGRES_PASSWORD=secret postgres
08fbb21e7e5cda6a2143f8bc675d63b4435f391a724f19e4f143fae42b3236fd
```

- Debemos tenerlo arrancado.. Podemos ver que la BBDD escucha por el puerto 5432. Como vamos a linkarlo con –link no es necesario “publicar” el puerto, como hemos hecho en el ejercicio anterior.

```
docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                      NAMES
08fbb21e7e5c        postgres            "docker-entrypoint.s…"   3 hours ago
Up 3 hours          5432/tcp                   postgresql1
f03d25490e17        mongo               "docker-entrypoint.s…"   9 hours ago
Up 9 hours          27017/tcp                  mongo3
c1b5a5882d0d        mongo               "docker-entrypoint.s…"   9 hours ago
Up 9 hours          0.0.0.0:27018->27017/tcp   mongo2
e909eff322a2        mongo               "docker-entrypoint.s…"   9 hours ago
Up 9 hours          0.0.0.0:27017->27017/tcp   mongo1
```

- Podemos acceder con una bash para ver que tenemos funcionando la base de datos

```
docker exec -it postgresql1 bash
```

- Nos conectamos a la base de datos (la password es la que hemos puesto al crear el contenedor, “secret” en mi caso). Usamos el comando “psql”. Si todo es correcto tenemos base de datos

```
psql -U postgres -W
Password for user postgres:
psql (10.3 (Debian 10.3-1.pgdg90+1))
Type "help" for help.


postgres=#
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access
privileges


-----------+----------+----------+------------+------------+------------------
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres
  +
           |          |          |            |            |
postgres=CTc/postgr
es
```

```
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres
  +
           |          |          |            |            |
postgres=CTc/postgr
es
(3 rows)
```

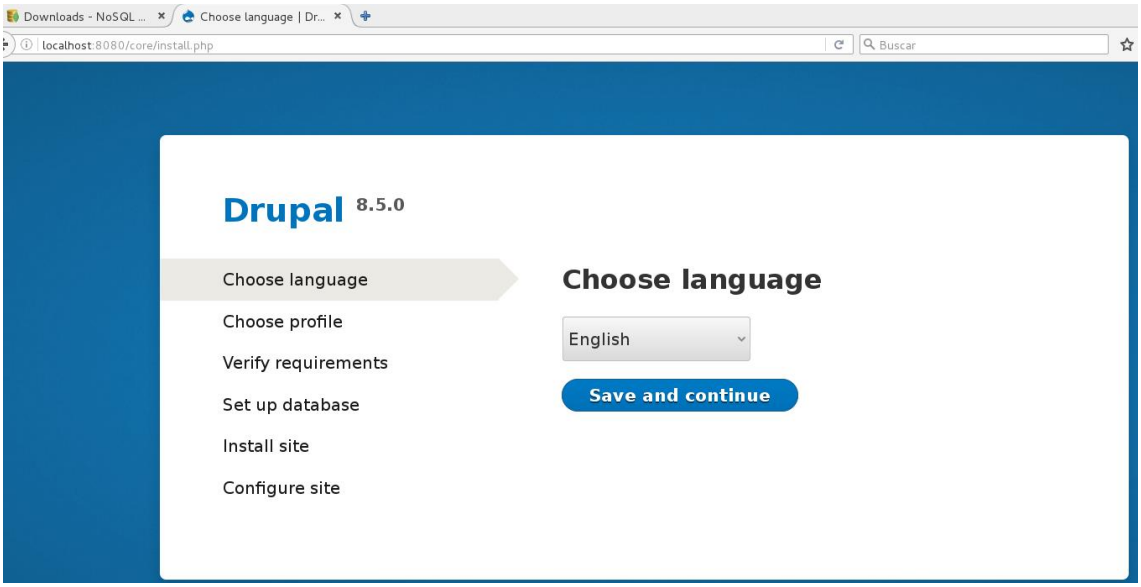
- Para salir de psql ponemos \q
- Ahora arrancamos el contenedor de Drupal y lo linkamos al contenedor de postgresql.
    - La parte más importante es –link. Le tenemos que decir el nombre que hemos dado al contenedor y separado por dos puntos el nombre que Drupal espera encontrar, en este caso “postgres” . Por supuesto es importante leerse la documentación para saber los datos necesarios
    - También le indicamos que para acceder al Drupal vamos a usar el puerto 8080 en la máquina host

```
docker run -d --name drupal1 --link postgresql1:postgres  -p
8080:80 drupal
9df282610690103a6426f076b11857733433094fe65cd2c4a16db4793d20c70e
```

- Podemos ver que están en marcha

```
docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                NAMES
434670127da1        postgres            "docker-entrypoint.s…"   54 seconds
ago      Up 53 seconds       5432/tcp, 0.0.0.0:8080->80/tcp   drupal1
08fbb21e7e5c        postgres            "docker-entrypoint.s…"   3 hours ago
Up 3 hours          5432/tcp                             postgresql1
```

- Si ahora accedemos con el Navegador al puerto 8080, debe aparecer el proceso de instalación de Drupal.



- Si creamos una bash contra Joomla podemos ver lo siguiente

```
docker exec -it drupal1 bash
root@179c7502f29b:/var/www/html#
```

- Si hacemos un cat del /etc/hosts podemos ver que tiene la dirección del otro contenedor (postgres)

```
#cat /etc/hosts
127.0.0.1   localhost
::1   localhost ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
172.17.0.2  postgres b63fbb2d5bbc postgresql1
172.17.0.3  179c7502f29b
```

- Uno de los problemas de usar esta opción de enlace, es que es unidireccional.

- Es decir, si accedemos a la bash de postgres, no podemos acceder a la de drupal por nombre, solo por IP. Tendríamos que hacerlo nosotros manualmente

```
docker exec -it postgresql1 bash
root@b63fbb2d5bbc:/# ping drupal1
```

```
ping: unknown host
```