Práctica 10. Creación de un contenedor Docker con PostgreSQL

Apuntes de BD para DAW, DAM y ASIR

Curso 2026/2027

Índice general

1 Creación de un contenedor Docker con PostgreSQL **1**

1.1 Requisitos 1

1.2 Cómo crear un contenedor sin persistencia de datos 1

1.3 Cómo conectarnos con el contenedor que está ejecutando PostgreSQL usando `psql` 2

1.4 Cómo conectarnos con el contenedor que está ejecutando PostgreSQL usando `adminer` . . . 2

1.5 Cómo comprobar que los contenedores están en ejecución 2

1.6 Cómo detener los contenedores 3

1.7 Referencias 3

2 Licencia **4**

1 Creación de un contenedor Docker con PostgreSQL

1.1. Requisitos

Para poder ejecutar contenedores Docker es necesario tener instalado Docker Community Edition (CE) en nuestro equipo.

En la web oficial encontrará la información necesaria para realizar la instalación de Docker CE sobre Windows, macOS, Ubuntu, Debian, Fedora y CentOS.

1.2. Cómo crear un contenedor sin persistencia de datos

Un contenedor Docker que no tiene persistencia de datos quiere decir que cuando finalice la ejecución perderá todo el contenido que hayamos creado durante la ejecución. Es decir, si durante la ejecución del contenedor hemos creado varias bases de datos en PostgreSQL, éstas se perderán cuando el contenedor se detenga.

El comando que podríamos usar para lanzar nuestro contenedor Docker con PostgreSQL sin persistencia de datos podría ser el siguiente:

```
1  docker run -d --rm --name postgres -e POSTGRES_PASSWORD=mysecretpassword -p
       5432:5432 postgres
```

- `docker run` es el comando que nos permite crear un contenedor a partir de una imagen Docker.
- El parámetro `-d` nos permite ejecutar el contenedor en modo *detached*, es decir, ejecutándose en segundo plano.
- El parámetro `--rm` hace que cuando salgamos del contenedor, éste se elimine y no ocupe espacio en nuestro disco.
- El parámetro `--name` nos permite asignarle un nombre a nuestro contenedor. Si no le asignamos un nombre Docker nos asignará un nombre automáticamente.
- El parámetro `-e` es para pasarle al contenedor variables de entorno. En este caso le estamos pasando la variable de entorno `POSTGRES_PASSWORD`, que contiene el valor de la contraseña del usuario `postgres`.
- El parámetro `-p` nos permite mapear los puertos entre nuestra máquina local y el contenedor. En este caso, estamos mapeando el puerto 5432 de nuestra máquina local con el puerto 5432 del contenedor.

- `postgres` es el nombre de la imagen. Como no hemos indicado la versión que vamos a utilizar para crear el contenedor, utilizará la última versión que esté disponible. Si no se indica lo contrario buscará las imágenes en el repositorio oficial Docker Hub.

1.3. Cómo conectarnos con el contenedor que está ejecutando PostgreSQL usando `psql`

Una vez que hemos creado la instancia del contenedor que está ejecutando PostgrSQL, podemos conectarnos a él para utilizar la herramienta `psql`.

```
1 docker run -it --rm --link postgres:postgres postgres psql -h postgres -U
    postgres
```

1.4. Cómo conectarnos con el contenedor que está ejecutando PostgreSQL usando `adminer`

Podemos crear un contenedor con Adminer para utilizar una interfaz web que nos permite conectar con PostgreSQL.

Para crear un contenedor con Adminer podemos ejecutar el siguiente comando:

```
1 docker run -d --rm  --link postgres:db -p 8080:8080 adminer
```

Una vez hecho esto, podríamos acceder a través de la URL http://127.0.0.1:8080 desde cualquier navegador web.

1.5. Cómo comprobar que los contenedores están en ejecución

Una vez que hemos iniciado los contenedores podemos comprobar que se están ejecutando con el siguiente comando:

```
1 docker ps
```

Deberíamos obtener una salida similar a esta.

```
1 CONTAINER ID        IMAGE               COMMAND                  CREATED
                STATUS              PORTS                    NAMES
2 a1e321c26d33        adminer             "entrypoint.sh …docke"   2 seconds ago
          Up 1 second         0.0.0.0:8080->8080/tcp   admiring_poitras
3 4485390ebc31        postgres            "docker-entrypoint.…s"   59 seconds ago
          Up 58 seconds       0.0.0.0:5432->5432/tcp   postgres
```

1.6. Cómo detener los contenedores

Para detener un contenedor en primer lugar tenemos que conocer cuál es su ID. Para obtenerlo podemos hacer uso del comando `docker ps`.

```
docker ps

CONTAINER ID        IMAGE               COMMAND                  CREATED
                STATUS              PORTS                    NAMES
a1e321c26d33        adminer             "entrypoint.sh …docke"   2 seconds ago
          Up 1 second         0.0.0.0:8080->8080/tcp   admiring_poitras
4485390ebc31        postgres            "docker-entrypoint.…s"   59 seconds ago
          Up 58 seconds       0.0.0.0:5432->5432/tcp   postgres
```

En la primera columna podemos ver cuál es el `CONTAINER ID`. Una vez localizado el identificador ejecutamos el comando `docker stop` y le pasamos como parámetro el identificador del contenedor que queremos detener.

Para el caso anterior deberíamos ejecutar:

```
docker stop a1e321c26d33
docker stop 4485390ebc31
```

1.7. Referencias

- Web oficial de PostgreSQL
- Imagen de PostgreSQL en Docker Hub
- Web oficial de Docker
- Instalación de Docker Community Edition (CE)
- Instalación de Docker Community Edition (CE) en Windows
- Instalación de Docker Community Edition (CE) en macOS
- Instalación de Docker Community Edition (CE) en Ubuntu
- Instalación de Docker Community Edition (CE) en Debian
- Instalación de Docker Community Edition (CE) en Fedora
- Instalación de Docker Community Edition (CE) en CentOS
- Docker Hub

2 Licencia

Esta página forma parte del curso Bases de Datos de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.