

Prácticas Docker

1. Prácticas con Dockerfile. Crear una imagen con la base de datos PostgreSQL 12 en un Ubuntu

- Vamos a crear un dockerfile basado en postgresQL para repasar algunas de las opciones y parámetros usados en esta sección
- Creamos un directorio llamado postgres_imagen
- Creamos el siguiente fichero dockerfile. Lo explicamos en detalle

```
##Si seleccionamos otra versión de Ubuntu o de Postgres,
puede que

##tengamos que modificar el fichero para adaptarlo
FROM ubuntu
LABEL "Creador"="Apasoft Training apasoft.training@gmail.com"

##Actualizamos
RUN apt-get update

##Instalamos WGET para el repositorio
RUN apt-get install -y wget

##Instalamos lsb-release para conocer la releases del
ubuntu
RUN apt-get install -y lsb-release

##Instalamos gpg2 para descargar la clave GPG
RUN apt-get install -y gnupg2

##Instalamos tzdata, porque Ubuntu ahora la pide de forma
interactiva
RUN DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get -y
install tzdata

##Configuramos el repositorio
```

```

RUN echo "deb http://apt.postgresql.org/pub/repos/apt
$(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list

## Descargamos la clave
RUN wget --quiet -O -
https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-
key add -

##Actualizamos el sistema
RUN apt-get update

##Instalamos Postgres
RUN apt-get -y install postgresql-12

##Nos cambiamos al usuario postgres, que se ha creado
##al instalar postgresQL

USER postgres

##Creamos un usuario denominado "pguser" con password
##"secret" y creamos una base de datos llamada "pgdb"
RUN /etc/init.d/postgresql start \
    && psql --command "CREATE USER pguser WITH SUPERUSER
PASSWORD 'secret';" \
    && createdb -O pguser pgdb

##Nos cambiamos a usuario ROOT
USER root

##Permitimos que se puede acceder a PostgreSQL
##desde clientes remotos
RUN echo "host all all 0.0.0.0/0 md5" >>
/etc/postgresql/12/main/pg_hba.conf

##Permitimos que se pueda acceder por cualquier

```

```
##IP que tenga el contenedor
RUN echo "listen_addresses='*'" >>
/etc/postgresql/12/main/postgresql.conf

##Exponemos el Puerto de la Base de Datos
EXPOSE 5432

##Creamos un directorio en /var/run y le damos permisos
##para el usuario postgres
RUN mkdir -p /var/run/postgresql && chown -R postgres
/var/run/postgresql

##Creamos los volúmenes necesarios para guardar
##el backup de la configuración, logs y bases de datos
##y poder acceder desde fuera del contenedor
VOLUME ["/etc/postgresql", "/var/log/postgresql",
"/var/lib/postgresql"]

##Nos cambiamos al usuario postgres
USER postgres

##Indicamos el comando a ejecutar al crear el contenedor
##Básicamente arrancar postgres con la configuración
##adecuada
CMD ["/usr/lib/postgresql/12/bin/postgres", "-D",
"/var/lib/postgresql/12/main", "-c",
"config_file=/etc/postgresql/12/main/postgresql.conf"]
```

- Creamos la imagen. La ponemos ya con el nombre correcto para subirla después a DockerHub. La etiquetamos como v1. Si aparecen mensajes en color rojo es normal, no es un error

```
docker build -t trainingdock/postgres:v1 .
```

- Creamos ahora un contenedor. Lo asociamos a la red net1

```
docker run -d --name postgres1 --network net1
trainingdock/postgres:v1
700f42bd66880a2a6e4de702bc48c743b28858ccc57473f362dae04af7
3141a3

docker ps
```

CONTAINER ID	IMAGE	COMMAND
700f42bd6688	trainingdock/postgres:v1	"/usr/lib/postgresql..."
5432/tcp	postgres1	Up 25 seconds

- Ahora vamos a crear un segundo contenedor pero esta vez arrancando solo la bash. Vamos a usarlo como cliente para conectarse al primer contenedor y comprobar que todo funciona

```
docker run -it --name postgres2 --network net1
trainingdock/postgres:v1 bash
postgres@c22abb78d74d:/$
```

- Lanzamos el comando psql (el cliente de PostgreSQL) para conectarnos al primer servidor. Recordemos que al estar en la misma red personalizada no tenemos que preocuparnos de los nombres de los servidores. Debemos poner "secret" como password y la base de datos "pgdb" que hemos creado con el Dockerfile

```
postgres@c22abb78d74d:/$ psql -h postgres1 -U pguser -W
pgdb
Password for user pguser:
psql (12.10 (Ubuntu 12.10-1.pgdg20.04+1+b1))
SSL connection (protocol: TLSv1.3, cipher:
TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

pgdb=#
```

- Podemos por ejemplo listar las base de datos existentes para comprobar que todo funciona. Usamos el comando \l

pgdb=# \l

List of databases

Name	Owner	Encoding	Collate	Ctype	Access privileges
pgdb	pguser	UTF8	C.UTF-8	C.UTF-8	
postgres	postgres	UTF8	C.UTF-8	C.UTF-8	
template0	postgres	UTF8	C.UTF-8	C.UTF-8	=c/postgres +
					postgres=CTc/postgres
template1	postgres	UTF8	C.UTF-8	C.UTF-8	=c/postgres +
					postgres=CTc/postgres

(4 rows)

pgdb=#

- Para salir del cliente ponemos \q
- Paramos y borramos los dos contenedores

2. Añadir variables y scripts en el CMD

- Como ejemplo adicional, vamos a añadir el uso de variables en la creación del contenedor.
 - USER: usuario a crear, por defecto pguser
 - PASS: La password del usuario. Por defecto será "secret"
 - BBDD: nombre de la Base de datos a crear. Por defecto será "pgdb"
- También vamos a llamar a un script desde el CMD, en vez de invocar directamente el comando. En este script crearemos el usuario y el resto de datos
- Primero creamos un fichero denominado "entrypoint.sh"
- Ponemos el siguiente contenido.

##Arrancamos la Base de Datos para el procedimiento inicial

/etc/init.d/postgresql start

##Creamos el usuario, la pass y la Base de datos, por supuesto con variables

```
psql --command "CREATE USER ${USER} WITH SUPERUSER
PASSWORD '${PASS}';"
createdb -O pguser ${BBDD}

##PAramos la instancia
/etc/init.d/postgresql stop

##Arrancamos de forma normal
exec /usr/lib/postgresql/12/bin/postgres -D
/var/lib/postgresql/12/main -c
config_file=/etc/postgresql/12/main/postgresql.conf
```

- Le ponemos permisos de ejecución

```
chmod +x entrypoint.sh
```

- Modificamos el fichero Dockerfile para reflejar los cambios. Lo que tenemos que hacer es eliminar la parte donde se creaba la parte del usuario y añadir el fichero "entrypoint.sh" y las variables.

```
##Si seleccionamos otra versión de Ubuntu o de Posrtgres,  
puede que  
##tengamos que modificar el fichero para adaptarlo  
FROM ubuntu  
LABEL "Creador"="Apasoft Training <apasoft.training@gmail.com>"  
  
##Actualizamos  
RUN apt-get update  
  
##Instalamos WGET para el repositorio  
RUN apt-get install -y wget  
  
##Instalamos lsb-release para conocer la relebase del  
ubuntu  
RUN apt-get install -y lsb-release  
  
##Instalamos gpg2 para descargar la clave GPG  
RUN apt-get install -y gnupg2
```

##Instalamos tzdata, porque Ubuntu ahora la pide de forma interactiva

```
RUN DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get -y
install tzdata
```

##Configuramos el repositorio.

```
RUN echo "deb http://apt.postgresql.org/pub/repos/apt
$(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list
```

Descargamos la clave

```
RUN wget --quiet -O -
https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-
key add -
```

##Actualizamos el sistema

```
RUN apt-get update
```

##Instalamos Postgres

```
RUN apt-get -y install postgresql-12
```

##Permitimos que se puede acceder a PostgreSQL

##desde clientes remotos

```
RUN echo "host all all 0.0.0.0/0 md5" >>
/etc/postgresql/12/main/pg_hba.conf
```

##Permitimos que se pueda acceder por cualquier

##IP que tenga el contenedor

```
RUN echo "listen_addresses='*'" >>
/etc/postgresql/12/main/postgresql.conf
```

##Exponemos el Puerto de la Base de Datos

```
EXPOSE 5432
```

##Creamos un directorio en /var/run y le damos permisos

##para el usuario postgres

```

RUN mkdir -p /var/run/postgresql && chown -R postgres
/var/run/postgresql

##Creamos los volúmenes necesarios para guardar
##el backup de la configuración, logs y bases de datos
##y poder acceder desde fuera del contenedor
VOLUME ["/etc/postgresql", "/var/log/postgresql",
"/var/lib/postgresql"]

##Copiamos el fichero entrypoint.sh y le ponemos permisos
ADD entrypoint.sh /usr/local/bin
RUN chmod +x /usr/local/bin/entrypoint.sh

##Nos cambiamos al usuario postgres
USER postgres

##Creamos 3 variables para crear el usuario,
##la password y la base de datos
ENV PASS=secret
ENV BBDD=pgdb
ENV USER=pguser

##Ejecutamos el script entrypoint.sh
CMD /usr/local/bin/entrypoint.sh

```

- Creamos la imagen con el tag v2

```
docker build -t trainingdock/postgres:v2 .
```

- Ahora creamos el contenedor. Debemos tener en cuenta pasarle como variables el usuario, la contraseña y el nombre de la Base de datos.
- Por ejemplo

```
docker run -d --name postgres2 --network net1 -e
PASS=password -e BBDD=bd1 -e USER=pguser
trainingdock/postgres:v2
```


- Para probarlo podemos conectarnos en esta ocasión al mismo contenedor e intentar acceder con los datos que hemos puesto en las variables.

```
docker exec -it postgres2 bash
```

- Nos conectamos con la password y base de datos definida

```
sql -h postgres2 -U pguser -W bd1
Password for user pguser:
psql (12.10 (Ubuntu 12.10-1.pgdg20.04+1+b1))
SSL connection (protocol: TLSv1.3, cipher:
TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

bd1=#
bd1=# \l
List of databases
   Name      | Owner   | Encoding | Collate | Ctype  |
Access privileges
-----+-----+-----+-----+-----+-----
bd1          | pguser  | UTF8     | C.UTF-8 | C.UTF-8 |
postgres    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
template0    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
=c/postgres  |         |          |         |         |
postgres=CTc/postgres
template1    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
=c/postgres  |         |          |         |         |
postgres=CTc/postgres
(4 rows)
```

- Salimos y borramos los contenedores