Contenedores con Docker. Descarga de imágenes de aplicaciones

- Se trata de descargar la imagen del servicio mysql preconfigurado desde el Docker Hub (en lugar de descargar una imagen de Ubuntu y dentro de ella descargar mysql) para ello:

  a) Descarga la imagen con docker pull mysql
  b) Ejecuta el comando:

```
docker run --name mysql -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306 mysql
```

En el comando anterior la opción -e es de "entorno" y sirve para establecer variables para el contenedor y el -d sirve para ejecutar el contenedor y dejarlo en corriendo en background ya que se trata de un servicio.

c) Ejecuta el comando `docker ps` para comprobar que el contenedor está en ejecución
d) Vamos a probar el acceso al servidor MySQL y para ello en la maquina anfitrión instalamos el cliente de mysql con:

- Caso de maquina Debian: `apt-get install mariadb-client`
- Caso de maquina Windows: se debe instalar un cliente mysql como Mysql Workbench o bien phpmyadmin (accesible desde XAMPP)

e) Desde la maquina anfitrión accedemos al servicio MySQL del contenedor con:

- Caso de maquina Debian: `mysql -h localhost -u root -p`
- Caso de maquina Windows: acceso desde el cliente grafico Workbench (si se usa phpmyadmin debe ajustarse la contraseña del root de MySQL en el fichero config.inc.php)

Nos pedirá la contraseña de ROOT que establecimos en el apartado b) y ya estamos en la gestion de MySQL desde donde podemos teclear sentencias SQL como `create database pruebas; use pruebas; create table..,` etc.. y cuando terminemos salimos con `quit` a la consola del sistema anfitrión

f) Se pide ahora crear un nuevo contenedor de mysql a partir de la imagen descargada anteriormente, pero esta vez debemos lanzarlo con un volumen gestionado por Docker que permita guardar los datos del directorio /var/lib/mysql

g) Una vez creado el contendor se debe comprobar desde Docker Desktop que el volumen esta creado y en uso. Visualizar los datos inicialmente creados en el volumen

h) Ahora, empleando la aplicación Mysql Workbench se debe ejecutar el script SQL Neptuno.sql que se entrega (este script crea la BBDD Neptuno, las tablas y varios datos de ejemplo)

i) Borramos ahora el contenedor y creamos un nuevo contenedor enlazándolo con el mismo volumen anterior

j) Nos conectamos con la aplicación y verificamos que los datos de la BBDD Neptuno siguen estando disponibles

1.- Descargamos la imagen de mysql.

```
C:\>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
7a5e1e917526: Downloading [============================>                      ]  27.26MB/47.31MB
98d18218e1bb: Downloading [=======>                                           ]  26.76MB/169.1MB
aff72f8f4e98: Downloading [=========================>                         ]  28.31MB/51.34MB
55f85a7d691e: Download complete
57bd38a2d740: Download complete
b843491434c1: Download complete
e4f376e797b1: Download complete
f6a972aa365b: Download complete
ac5a3aa7003a: Download complete
ba8d858f8b56: Download complete
```

Con docker images comprobamos que la tenemos descargada.

```
C:\>docker images
REPOSITORY                   TAG       IMAGE ID       CREATED         SIZE
mysql                        latest    fe036967257b   2 days ago      1.27GB
guillermosr93/ubuntu_daw2b   latest    6e1e1724daaa   7 days ago      365MB
ubuntu                       latest    c35e29c94501   7 weeks ago     117MB
hello-world                  latest    f7931603f70e   3 months ago    20.3kB
mariadb                      11.2      ff87d49107a1   13 months ago   551MB

C:\>
```

2.- Ahora vamos a lanzar el contenedor con docker run, donde Naranco será mi contraseña para la base de datos y el puerto será 3320.

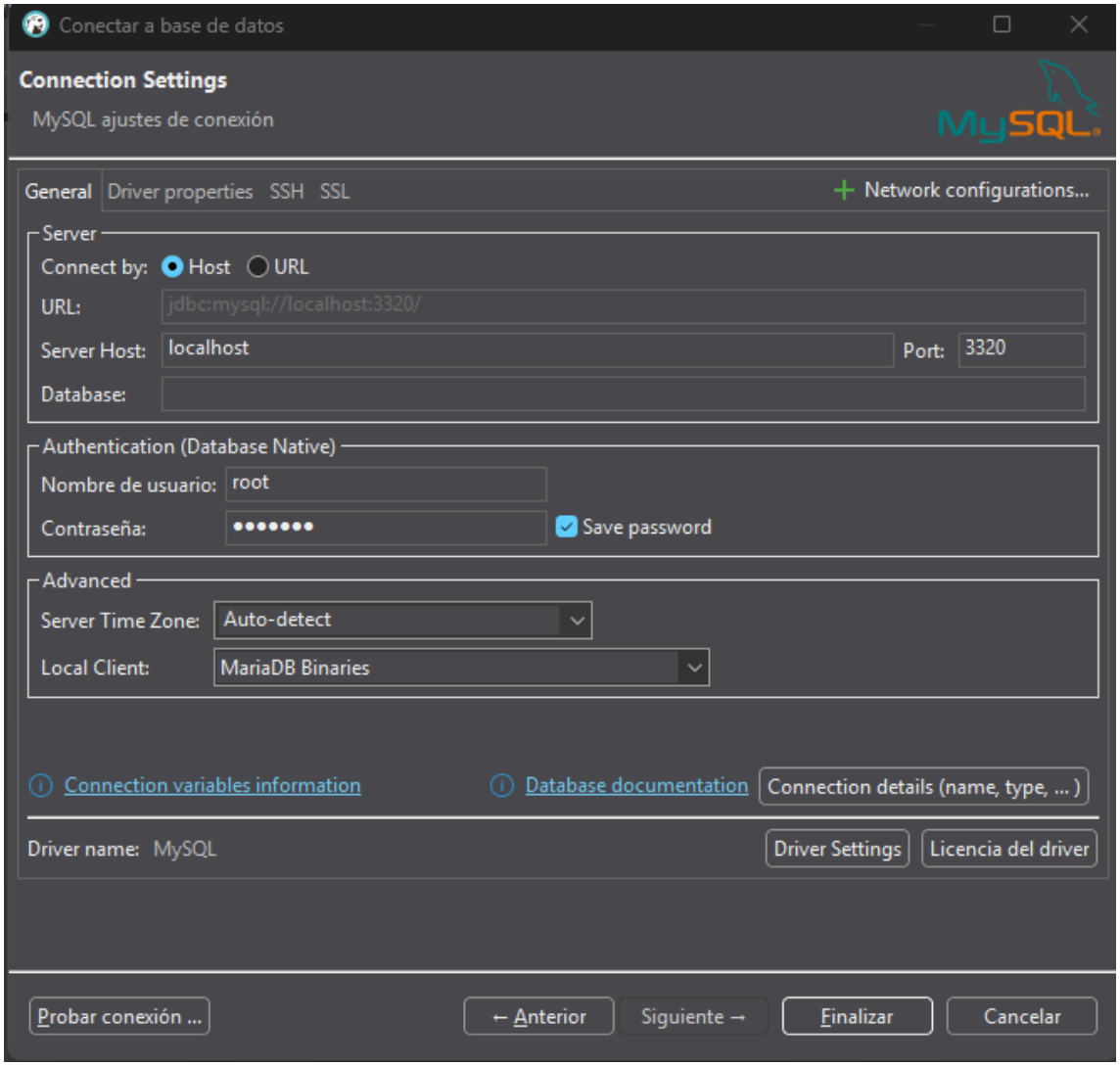
```
C:\>docker run --name mysql -e MYSQL_ROOT_PASSWORD=naranco -d -p 3320:3306 mysql
cc60500bdac0efe6cd699a1f0fed0c3e438367381d908251b51f676278aa2b9c
```

Ahora con docker ps comprobamos que está funcionando.

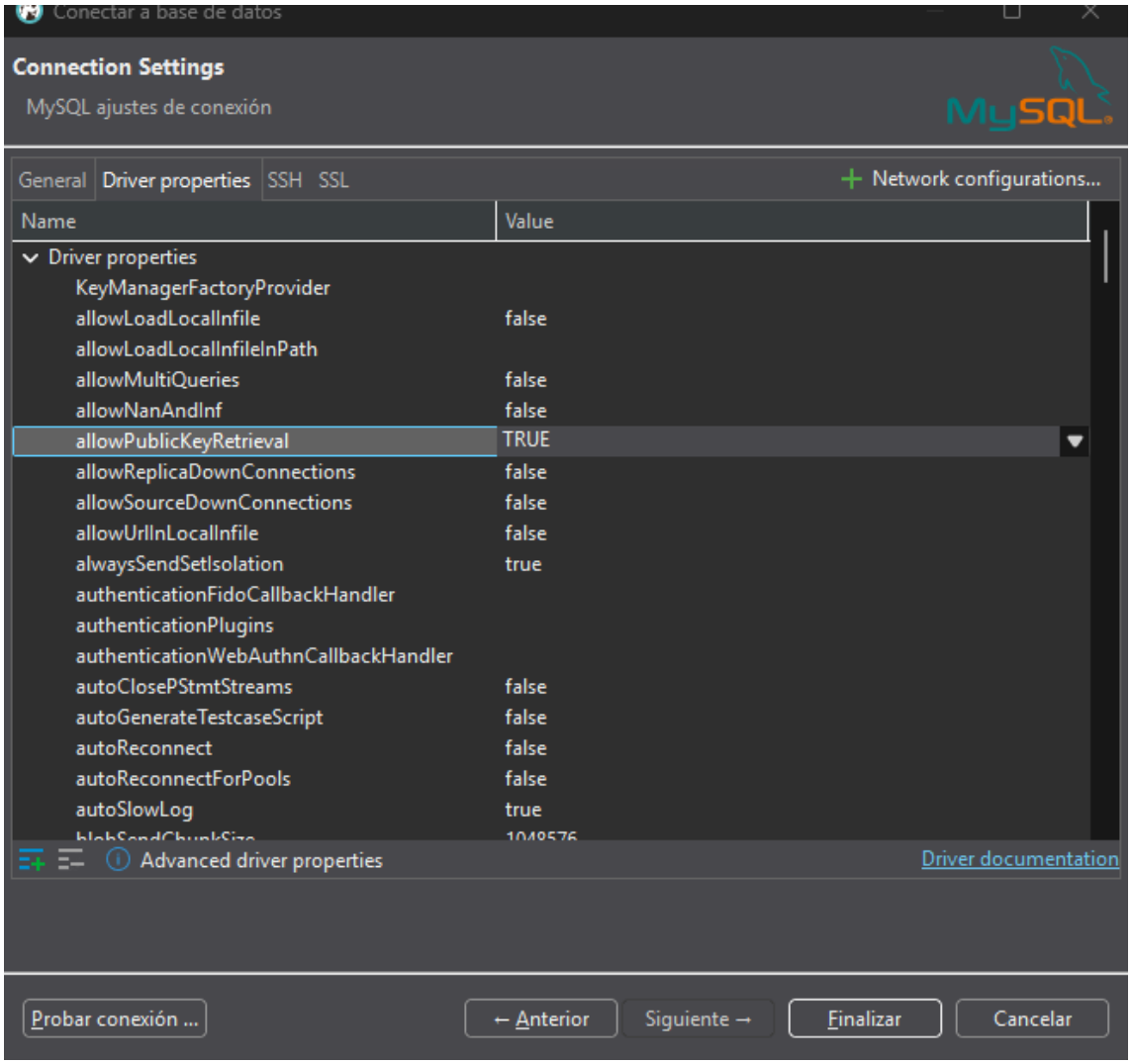
```
C:\>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS
       NAMES
cc60500bdac0   mysql     "docker-entrypoint.s…"   About a minute ago   Up About a minute   33060/tcp, 0.0.0.0:3320->3306
/tcp   mysql

C:\>
```

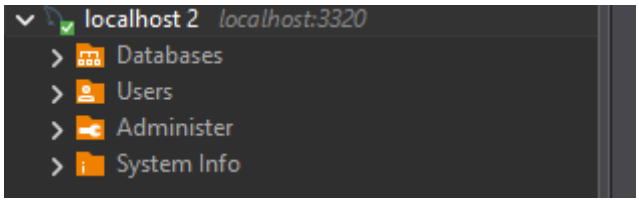
3.- Ahora nos conectaremos a la base de datos, en mi caso con DBeaver, con el puerto que hemos puesto y la contraseña.



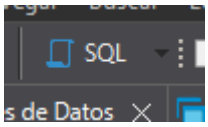
Debemos poner allowPublicKeyRetrieval a TRUE.



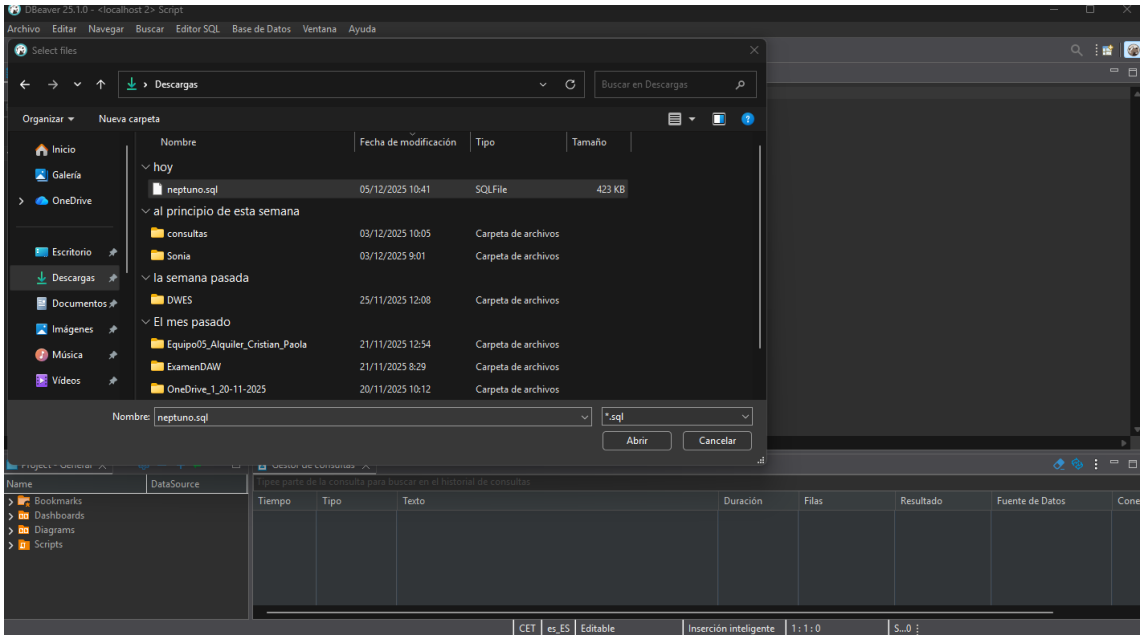
Finalmente, se nos habrá conectado a la base de datos.

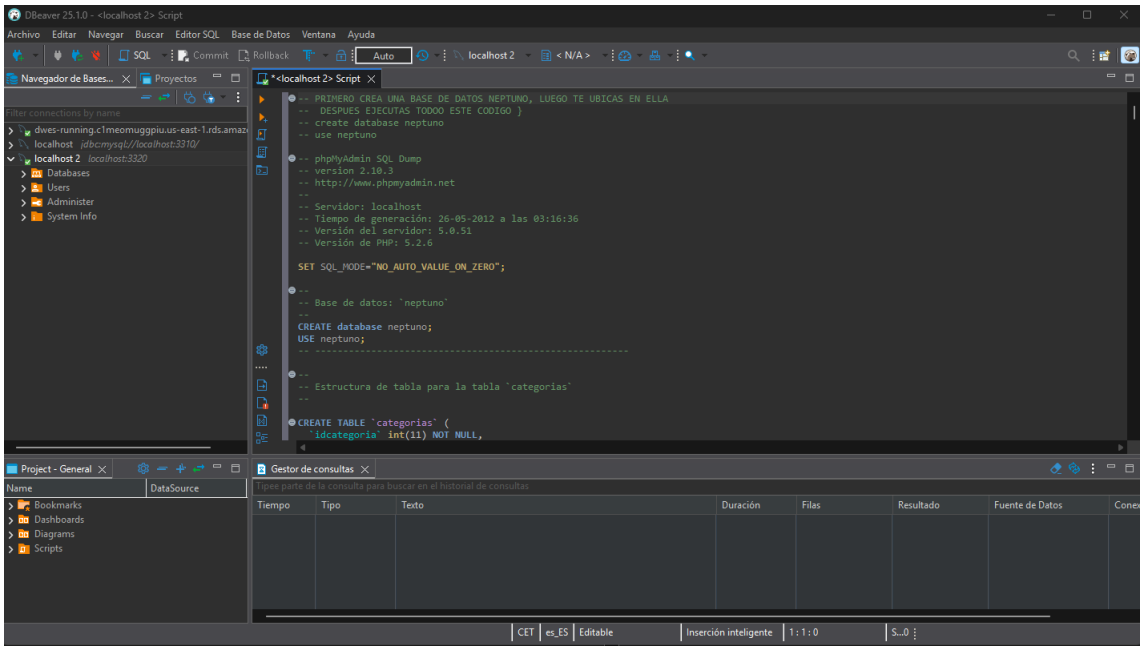


4.- Ahora ejecutaremos el script Neptuno.sql, para ello iremos a la ventana de SQL.

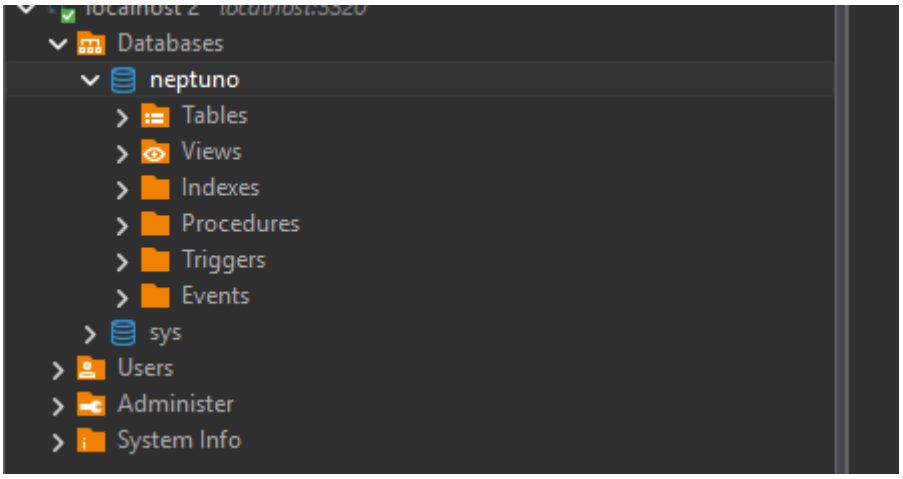


Abriremos el archivo



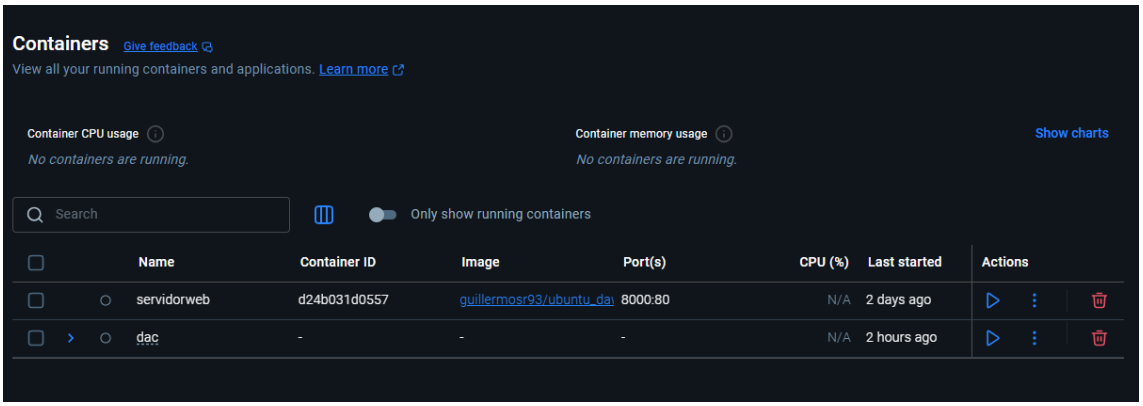


Y en el apartado de la izquierda ejecutaremos el script.



Ya estará listo.

5.- Ahora borraremos el container y lo volveremos a lanzar pero creando un volumen.



Con el comando -v creamos el volumen.

```
C:\>docker run --name basedatos -v bbdd:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=naranco -d -p 3320:3306 mysql
```

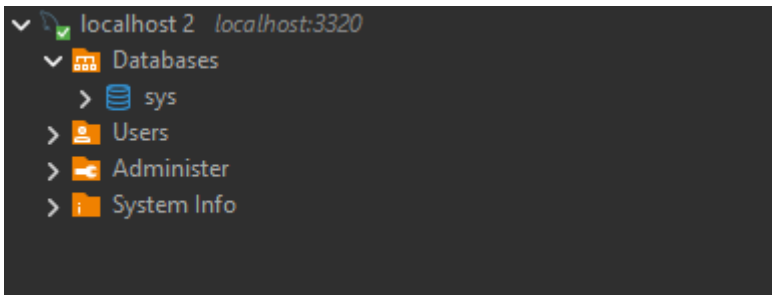
Comprobamos que este funcionando con docker ps

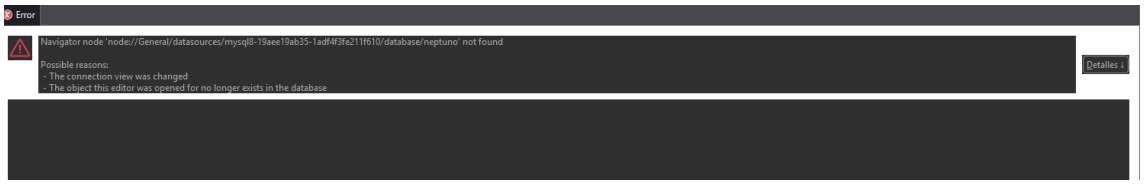
```
C:\>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                               NAM
ES
f9714b72560b   mysql     "docker-entrypoint.s…"   2 seconds ago   Up 2 seconds   33060/tcp, 0.0.0.0:3320->3306/tcp   bas
edatos

C:\>
```

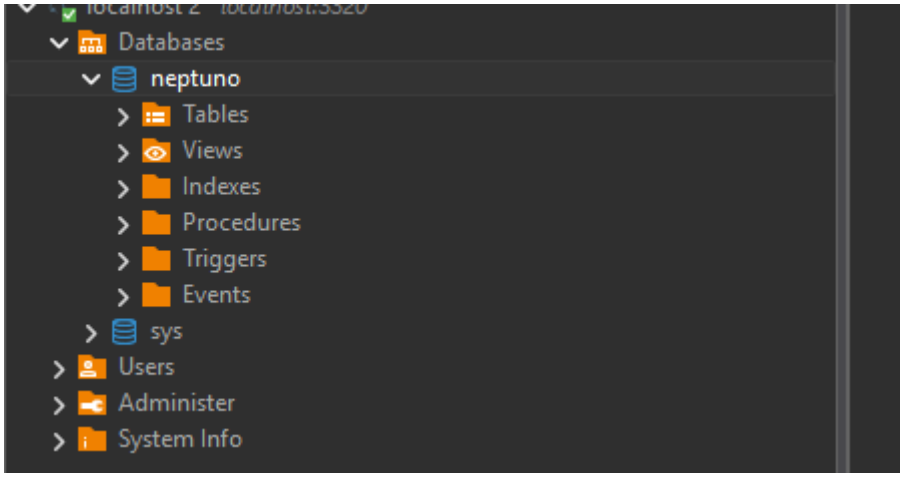
Y comprobamos también que se nos ha creado el volumen.

Como podemos observar, antes no se nos han guardado los datos.

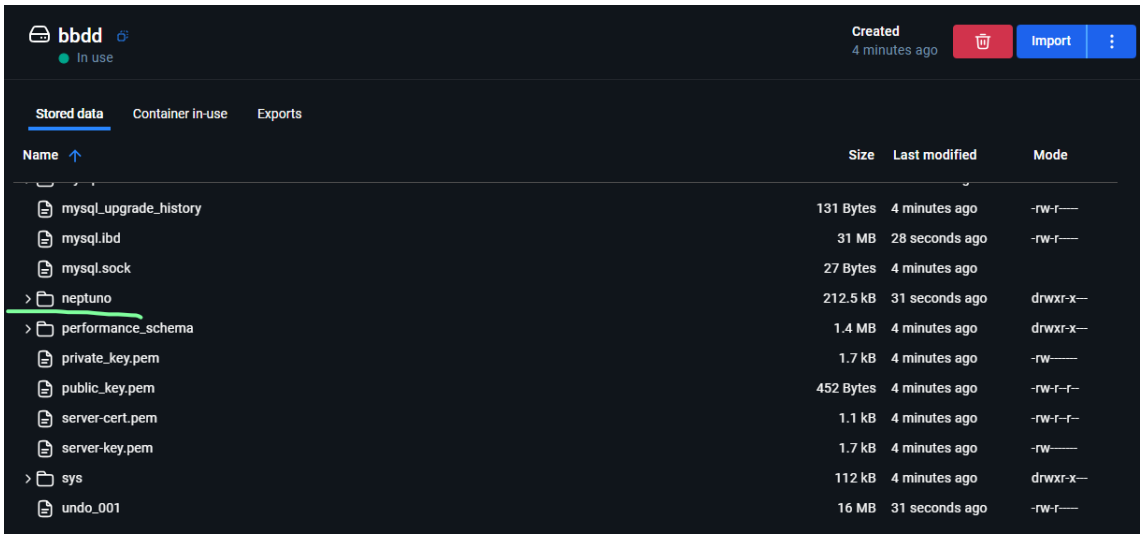




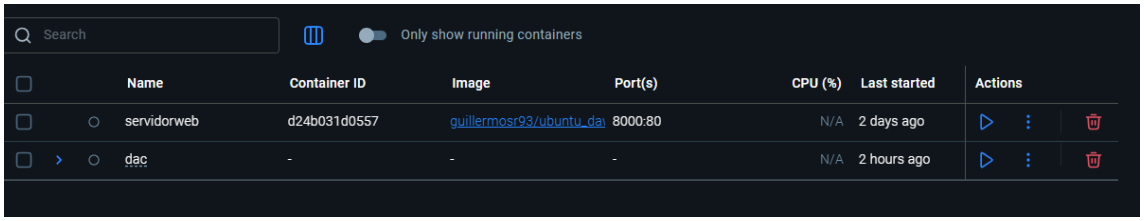
Volveremos a realizar los pasos anteriores para volver a crear la base de datos.



6.- Ahora en el apartado de volúmenes, en los datos, nos debería de salir nuestra base de datos.



7.- Si volvemos a lanzar el contenedor después de haberlo borrado, nos debería de haber guardado los datos. Primero lo borramos.



Y ahora con el run -v lo volvemos a lanzar.

```
C:\>docker run --name basedatos2 -v bbdd:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=naranco -d -p 3320:3306 mysql
```

Y como podemos comprobar se nos ha guardado la base de datos creada previamente.