



Introducción

Esta sección de la documentación está enfocada sola y específicamente a la conexión entre los dos contenedores armados previamente.

Se mostrará paso a paso como levantar ambos contenedores con la conexión entre si ya establecida, así como el procedimiento para que puedan comunicarse entre sí ya en el script.

Procedimiento

El primer paso es asegurarnos que ninguno de los dos contenedores esté corriendo actualmente.

Para ello, podemos utilizar el siguiente comando:

Docker ps

```
C:\Users\ricar>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

Este es el output esperado, ninguno de los dos contenedores activos.

Si en dado caso estuvieran corriendo, se pueden detener con el siguiente comando:

Docker stop [Nombre del contenedor]

Una vez estemos seguros de que ninguno de los dos está corriendo, podemos realizar la conexión.

Crear el network

El primer paso es crear un network por el cual se conectarán ambos contenedores. Para ello usamos el siguiente comando:

docker network create --driver bridge [Nombre que le quiera dar a la conexión]

```
C:\Users\ricar>docker network create --driver bridge con1  
d96ea03c4e8b374147e31779b48d01704d65be3f858840d331df45fbd716405a
```

Como nos devuelve un Hash, sabemos que fue creada correctamente.

Levantar los contenedores con la conexión que creamos

Contenedor MySQL

`docker run -it --network [Nombre que le haya puesto a la conexión] [Nombre de su contenedor de MySQL] --secure-file-priv=/home/parcial`

```
Símbolo del sistema - docker n × + v
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
2020-08-31 17:45:28+00:00 [Note] [Entrypoint]: Creating database parcial1_dp
2020-08-31 17:45:28+00:00 [Note] [Entrypoint]: Creating user ricardo
2020-08-31 17:45:28+00:00 [Note] [Entrypoint]: Giving user ricardo access to schema parcial1_dp

2020-08-31 17:45:28+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/load_data.sql

2020-08-31 17:45:29+00:00 [Note] [Entrypoint]: Stopping temporary server
2020-08-31 17:45:32+00:00 [Note] [Entrypoint]: Temporary server stopped

2020-08-31 17:45:32+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.

2020-08-31T17:45:32.429350Z 0 [Warning] [MY-010101] [Server] Insecure configuration for --secure-file-priv: Location is accessible to all OS users. Consider choosing a different directory.
2020-08-31T17:45:32.429457Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.21) starting as process 1
2020-08-31T17:45:32.445524Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2020-08-31T17:45:32.783749Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2020-08-31T17:45:33.011339Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqld.sock
2020-08-31T17:45:33.182288Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2020-08-31T17:45:33.182691Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
2020-08-31T17:45:33.189512Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
2020-08-31T17:45:33.231344Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.21' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
```

Contenedor RStudio

Nota: Dado que el contenedor de MySQL se mantendrá activo, hay que correr este comando desde otra terminal:

`docker run --network [Nombre que le haya puesto a la conexión] -p 8888:8888 -e PASSWORD=[Contraseña que quiera] [Nombre de su contenedor de RStudio]`

```
C:\Users\ricar>docker run --network con1 -p 8888:8888 -e PASSWORD=pass sebpineda23/parcial1
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions fixes...
[fix-attrs.d] done.
[cont-init.d] executing container initialization scripts...
[cont-init.d] userconf: executing...
[cont-init.d] userconf: exited 0.
[cont-init.d] done.
[services.d] starting services
[services.d] done.
```

Verificar que ambos estén corriendo correctamente

Desde la terminal se puede hacer con el comando que utilizamos previamente:

Docker ps

```
C:\Users\ricar>docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                               NAMES
80cb18d894e8   sebpineda23/parcial1 "/init"                49 seconds ago Up 47 seconds 8787/tcp, 0.0.0.0:8888->8888/tcp   adoring_turing
dbb4de504ab5   sebpineda23/mysql    "docker-entrypoint.s..." 5 minutes ago  Up 5 minutes  3306/tcp, 33060/tcp              friendly_goldberg
```

En efecto ambos están corriendo correctamente.

Conexión

Ahora que ambos contenedores están activos, hay que inspeccionar la conexión para ver que dirección le asigno de IPV4:

docker network inspect [Nombre que le haya puesto a la conexión]

```
C:\Users\ricar>docker network inspect con1
[
  {
    "Name": "con1",
    "Id": "d96ea03c4e8b374147e31779b48d01704d65be3f858840d331df45fbd716405a",
    "Created": "2020-08-31T17:43:05.3119084Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.20.0.0/16",
          "Gateway": "172.20.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "80cb18d894e81d10dd57118212ea8aae4b03e3b690486120b19e00da261ffccf": {
        "Name": "adoring_turing",
        "EndpointID": "d564b5ddf8db9ceb64f2a2e58b9bceea96c80a1d7b51083fec29cd16be8a9a36",
        "MacAddress": "02:42:ac:14:00:03",
        "IPv4Address": "172.20.0.3/16",
        "IPv6Address": ""
      },
      "dbb4de504ab5743e170df12c0d1d1f11140c5ab0358f609a11e3849e76ba51ea": {
        "Name": "friendly_goldberg",
        "EndpointID": "d8cbccd7fe4f31e4e7e212e26405cb0b832d1bbe96d0bde28fab0b0a776275a9",
        "MacAddress": "02:42:ac:14:00:02",
        "IPv4Address": "172.20.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
```

En mi caso, la dirección que le asignó es la **172.20.0.2**

Hacer la comunicación

Ahora que ya tenemos la dirección asignada, nos vamos a nuestra instancia de RStudio en un browser, en 127.0.0.1:8888 y modificamos la parte de la conexión con la nueva dirección:

```
my_db <- dbPool(  
  RMySQL::MySQL(),  
  dbname = "parcial1_dp",  
  host = "172.20.0.2", ## Cambiar según su conexión  
  username = "ricardo",  
  password = "pass"  
)
```

```
my_db <- dbPool(  
  RMySQL::MySQL(),  
  dbname = "parcial1_dp",  
  host = "172.20.0.2", ## Cambiar según su conexión  
  username = "ricardo",  
  password = "pass"  
)
```

En mi caso así quedó la conexión, pero la dirección IP (host) podría variar según su caso.

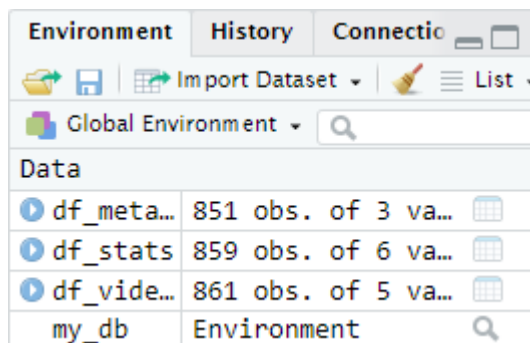
Si probamos jalar las tablas de la base de datos:

```
df_stats <- as.data.frame(my_db %>% tbl("video_stats"))
```

```
df_metadata <- as.data.frame(my_db %>% tbl("video_metadata"))
```

```
df_video_data <- as.data.frame(my_db %>% tbl("videos"))
```

Podemos ver que efectivamente si lo jala:



The screenshot shows the RStudio Environment pane with the following data frames listed:

Object	Details
df_meta...	851 obs. of 3 va...
df_stats	859 obs. of 6 va...
df_vide...	861 obs. of 5 va...
my_db	Environment