

Universidad Francisco Marroquín

Data Product

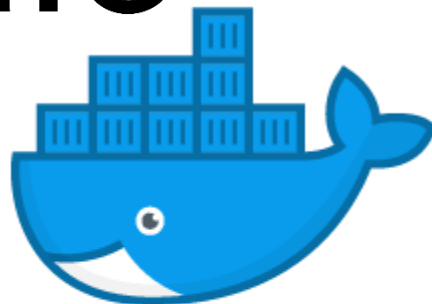
Septiembre del 2020

Parcial # 1

Juan Diego Sique

Ricardo Pineda

Sección "A"



docker

Introducción

Esta sección de la documentación estará enfocada específicamente a la dockerización del R-Studio.

Se demostrará cada paso del proceso para montar un contenedor que contenga las librerías necesarias para el parcial, tomando en cuenta que desarrollamos el análisis tanto con flexdashboards, como con Shiny.

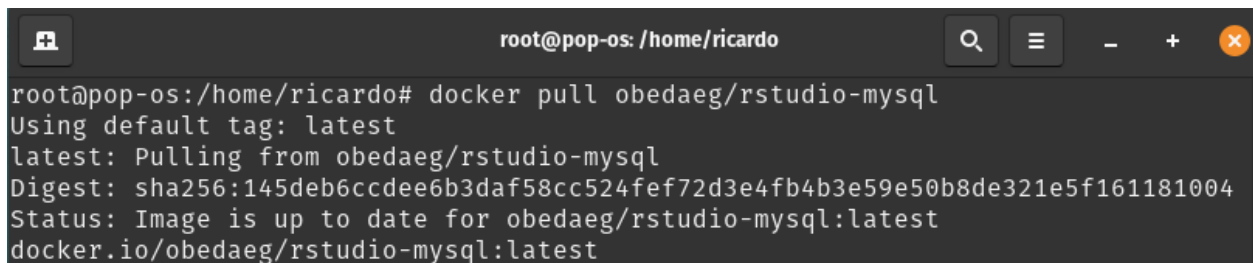
Para lograr esto, de elaborará un contenedor que estará basado en el contenedor de Obed, encontrado en Docker Hub como *obedaeg/rstudio-mysql*.

Procedimiento

Descargar la imagen

El primer paso por tomar es jalar el contenedor que nos dio Obed para tenerlo localmente en nuestra máquina. Para eso se utiliza el siguiente comando desde la terminal:

```
docker pull obedaeg/rstudio-mysql
```

A terminal window titled 'root@pop-os: /home/ricardo' showing the command 'docker pull obedaeg/rstudio-mysql' and its output. The output indicates that the image is already up to date.

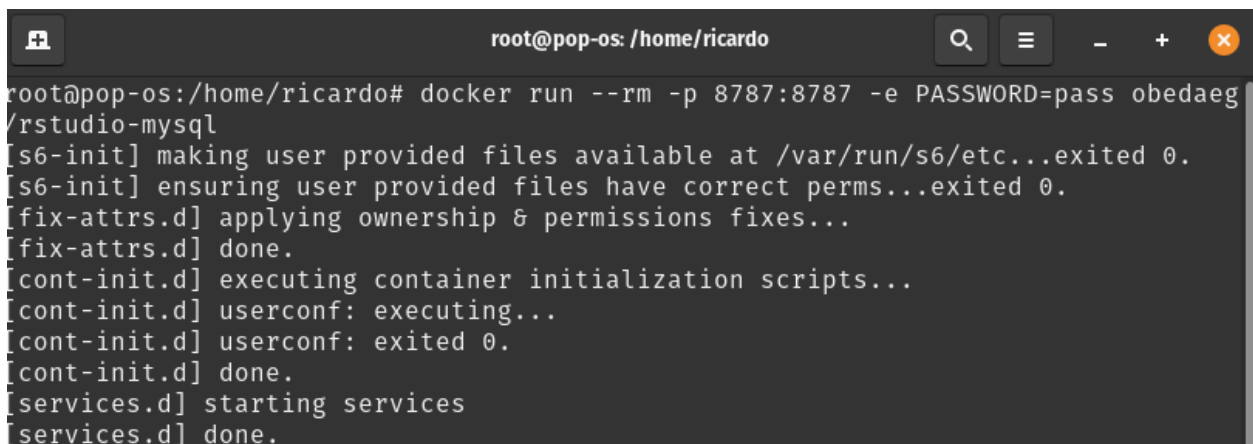
```
root@pop-os:/home/ricardo# docker pull obedaeg/rstudio-mysql
Using default tag: latest
latest: Pulling from obedaeg/rstudio-mysql
Digest: sha256:145deb6ccdee6b3daf58cc524fef72d3e4fb4b3e59e50b8de321e5f161181004
Status: Image is up to date for obedaeg/rstudio-mysql:latest
docker.io/obedaeg/rstudio-mysql:latest
```

En mi caso ya estaba instalada, pero de no ser así, se comenzará a descargar y tomará unos minutos.

Probar la imagen

Ahora que la imagen ha sido descargada, nos toca probar para verificar que esté correctamente instalada. Para eso se utiliza el siguiente comando:

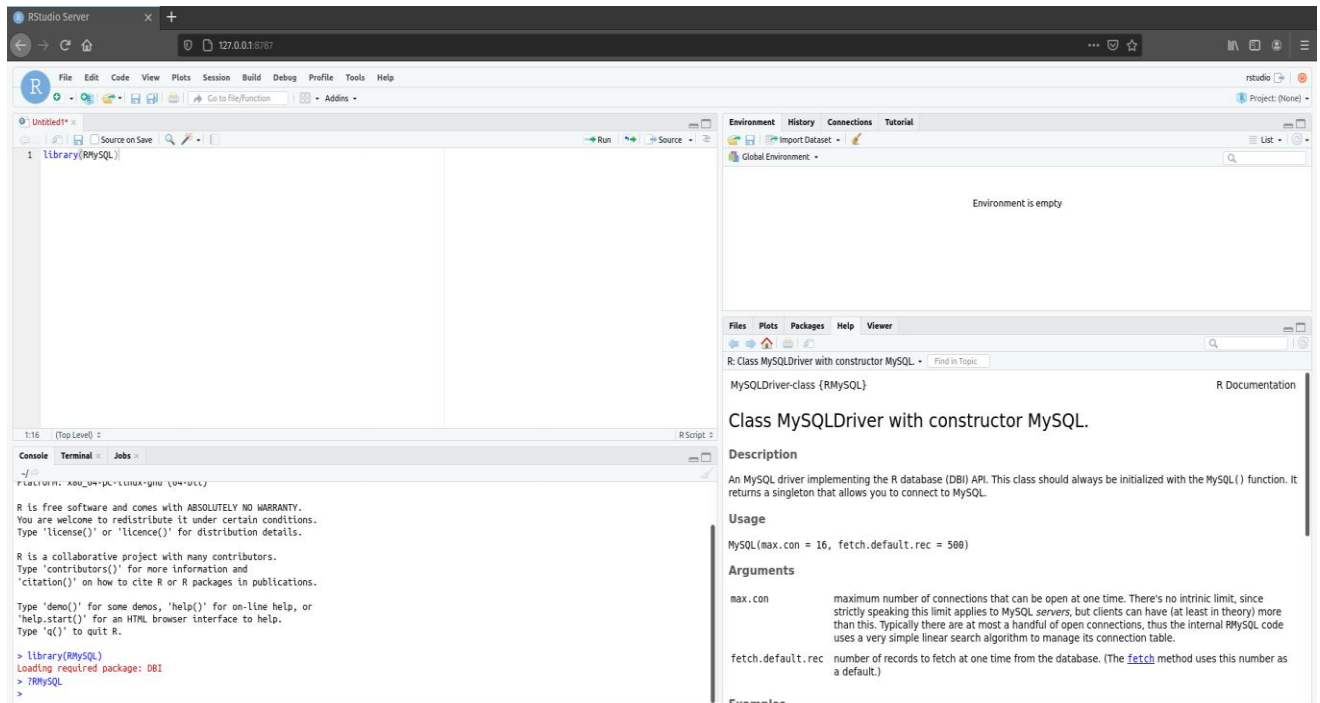
```
docker run --rm -p 8787:8787 -e PASSWORD=yourpasswordhere obedaeg/rstudio-mysql
```

A terminal window titled 'root@pop-os: /home/ricardo' showing the command 'docker run --rm -p 8787:8787 -e PASSWORD=pass obedaeg/rstudio-mysql' and its output. The output shows the container initialization process, including setting up user files and starting services.

```
root@pop-os:/home/ricardo# docker run --rm -p 8787:8787 -e PASSWORD=pass obedaeg/rstudio-mysql
[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.
```

Eso es lo que nos tiene que devolver la terminal. Si ese fue nuestro output, podemos ir al browser en: 127.0.0.1:8787 para usar la instancia de R Studio.

NOTA: El usuario default para la instancia es Rstudio. El password es el que definimos en el comando.



Podemos ver que tenemos una sesión de RStudio con la funcionalidad completa dentro del contenedor, y que la librería de RMySQL funciona correctamente.

Sin embargo, nosotros utilizamos varias librerías extras, así como también desarrollamos una parte del análisis utilizando Shiny, por lo que optamos por crear nuestro propio contenedor que tome en cuenta estas consideraciones.

Crear el contenedor

Para crear nuestro contenedor, nos basamos en la imagen de Obed y construimos sobre ella. Para hacer eso, creamos un Dockerfile con las especificaciones de nuestro proyecto.

El Dockerfile se puede crear en cualquier editor de texto, pero personalmente recomiendo Visual Studio Code pues trae varios plug ins que pueden ayudar a escribir un Dockerfile.

Procedimiento

1. (Opcional pero recomendado) Crear un directorio específico para los archivos
2. Crear el Dockerfile que tiene los comandos necesarios para el proyecto

Dockerfile:

```
FROM obedaeg/rstudio-mysql
# Instalar las librerías
RUN R -e "install.packages(c('shinydashboard', 'shinyjs', 'V8', 'lubridate', 'shiny', 'readr', 'DT', 'dygraphs', 'stringr', 'parsedate', 'rbokeh'))"
```

3. Ejecutar el siguiente comando:

Sudo Docker build -t [NOMBRE] . [Path/al/Dockerfile] (No es necesario si ya estamos en el directorio)

```
root@pop-os:/home/ricardo/Parcial# docker build -t parcial_shiny .
Sending build context to Docker daemon 5.632kB
Step 1/2 : FROM obedaeg/rstudio-mysql
--> 6836db22d1b2
Step 2/2 : RUN R -e "install.packages(c('shinydashboard', 'shinyjs', 'V8', 'lubridate', 'shiny', 'readr', 'DT', 'dygraphs', 'stringr', 'parsedate', 'rbokeh'))"
--> Using cache
--> 00783d4ca47f
Successfully built 00783d4ca47f
Successfully tagged parcial_shiny:latest
```

4. Esperar a que el comando ejecute e instale todo lo necesario. Debido a que en este paso se están instalando todas las librerías y se están creando los directorios necesarios, tomará unos minutos
5. Para verificar que esté instalada correctamente, podemos poner Docker images en la terminal y ver que efectivamente esté nuestra imagen:

```
root@pop-os:/home/ricardo/Parcial# docker images
```

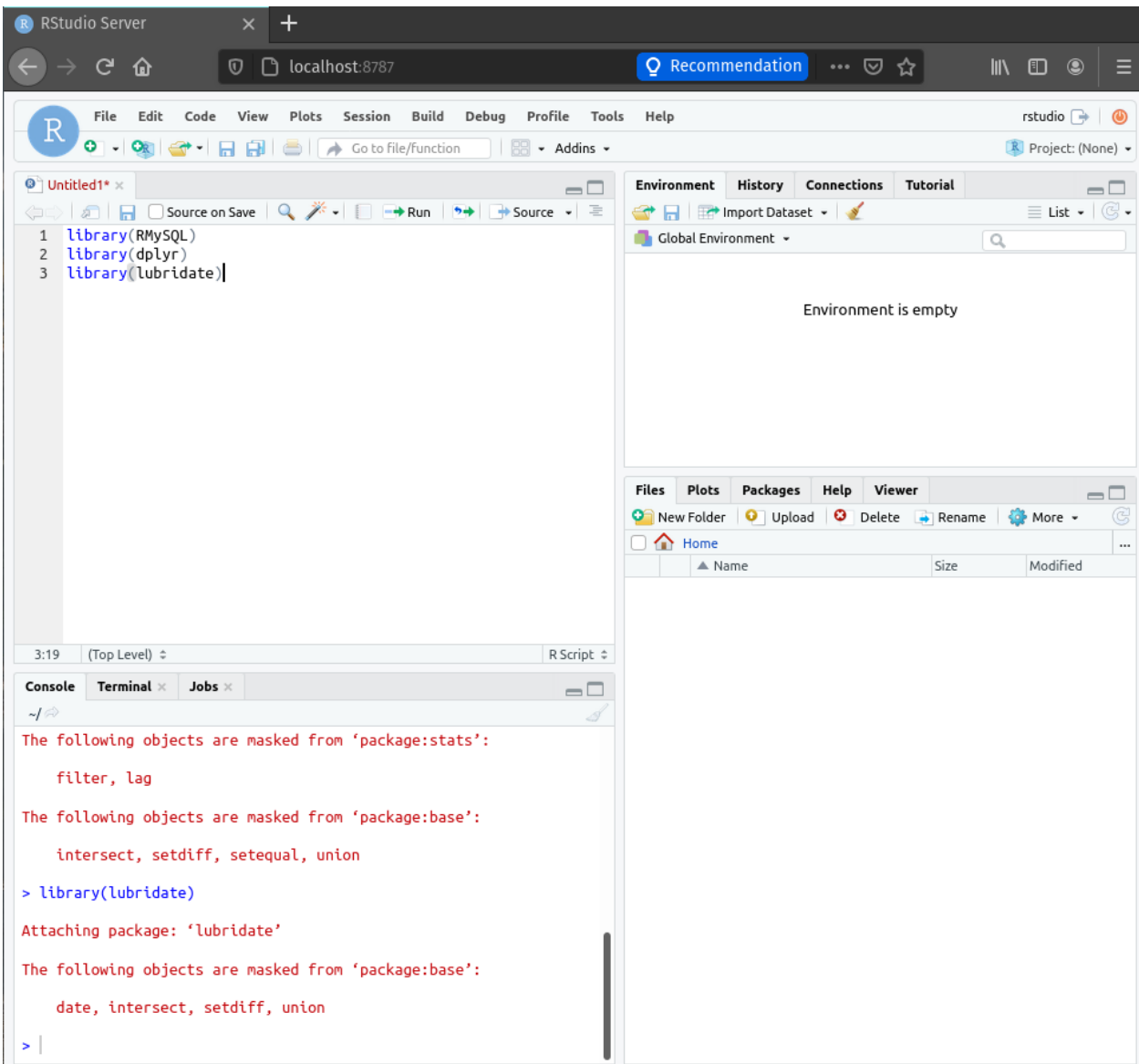
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
parcial_shiny	latest	00783d4ca47f	47 seconds ago	2.24GB

6. Ahora, para correr el contenedor utilizamos el siguiente comando:

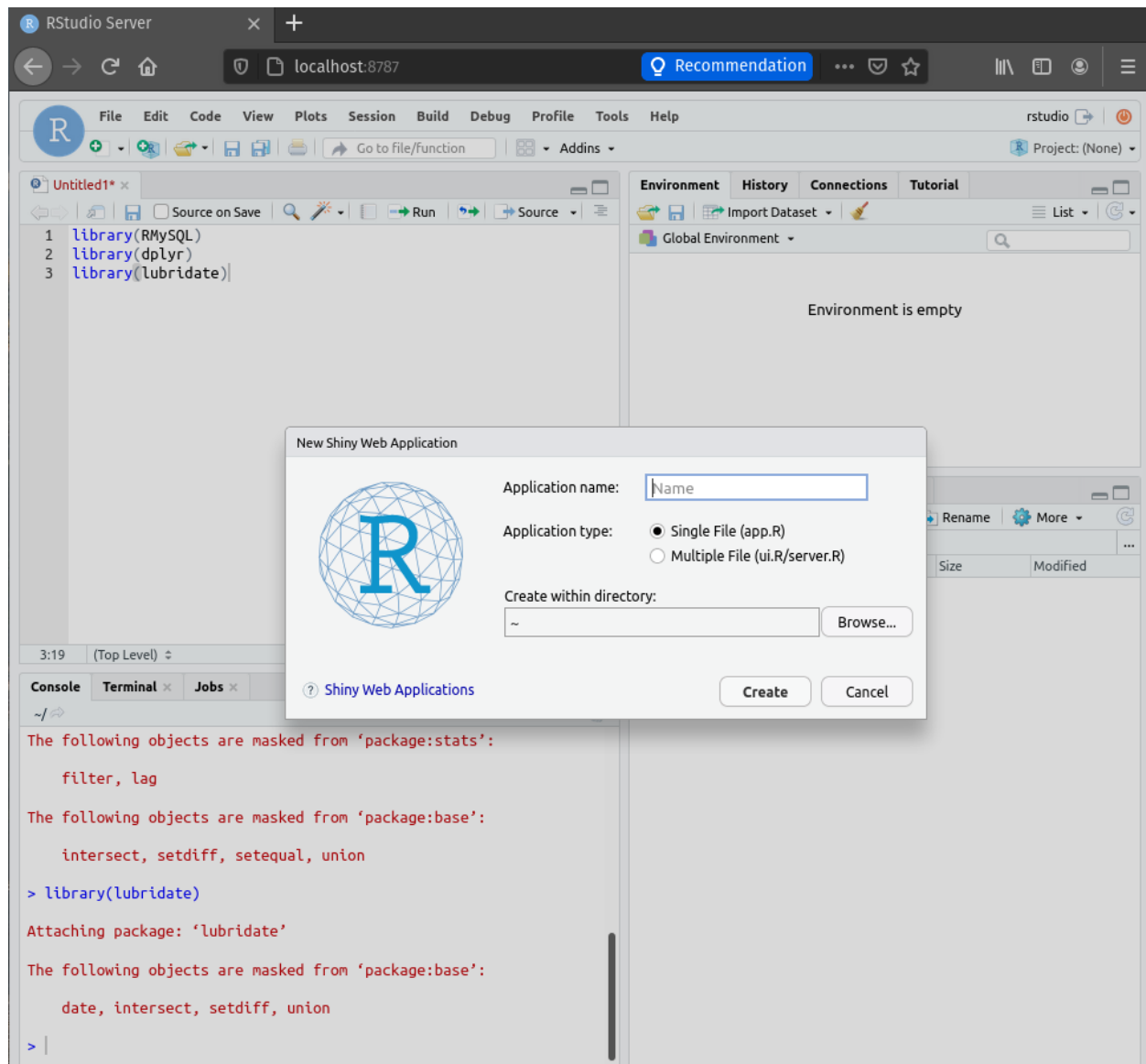
docker run --rm -p 8787:8787 -e PASSWORD=yourpasswordhere [nombre-del-contenedor]

```
root@pop-os:/home/ricardo/Parcial# docker run --rm -p 8787:8787 -e PASSWORD=pass parcial_shiny
[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.
```

- a. Al igual que con el contenedor de Obed, el usuario default es Rstudio, y el password es el que especificamos en el comando al levantar el contenedor.
- b. Para probar que este correcto, ir a 127.0.0.1:[Puerto que especificamos]



Como podemos ver, la instancia de Rstudio funciona correctamente, y trae automáticamente instaladas las librerías que necesitaremos en el parcial, así como la posibilidad de crear un Shiny App:



Scripts y Data

Por como está configurado el contenedor, actualmente cada vez que volvamos a levantar la imagen tendremos solamente una instancia de RStudio vacía con las librerías instaladas. Para resolver eso, haremos nuevos directorios y copiaremos la información necesaria.

Para eso actualizaremos nuestro Dockerfile para tomar eso en cuenta:

Dockerfile

```
FROM obedaeg/rstudio-mysql

# instala los paquetes necesarios
RUN R -e "install.packages(c('flexdashboard', 'shinyWidgets', 'shinythemes', 'pool', 'shinydashboard', 'shinyjs', 'V8', 'lubridate', 'shiny', 'readr', 'DT', 'dygraphs', 'stringr', 'parsedate', 'rbokeh'))"

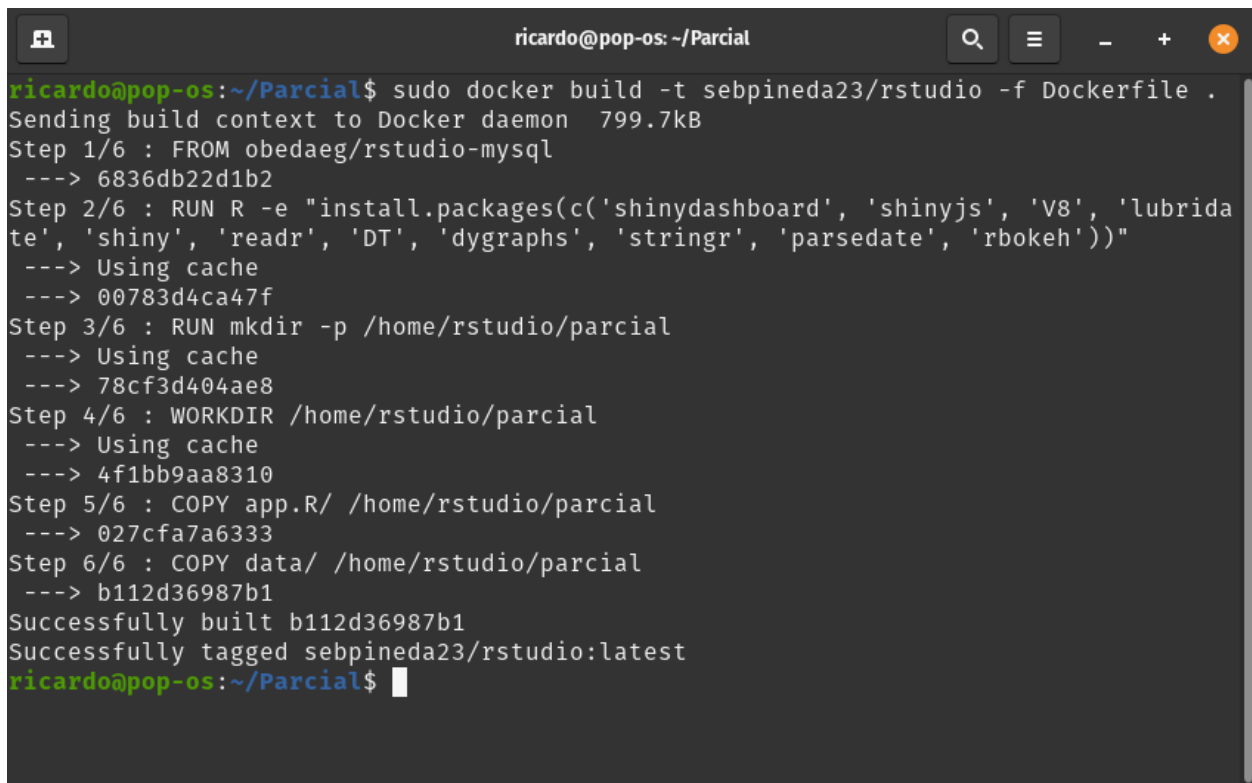
# Crea un directorio para el parcial
RUN mkdir -p /home/rstudio/parcial
WORKDIR /home/rstudio/parcial

RUN chmod -R 777 /home/rstudio/parcial/

# Copia los scripts necesarios
COPY app.R/ /home/rstudio/parcial
COPY dashboards.Rmd/ /home/rstudio/parcial
```

El Dockerfile ahora hace varias cosas:

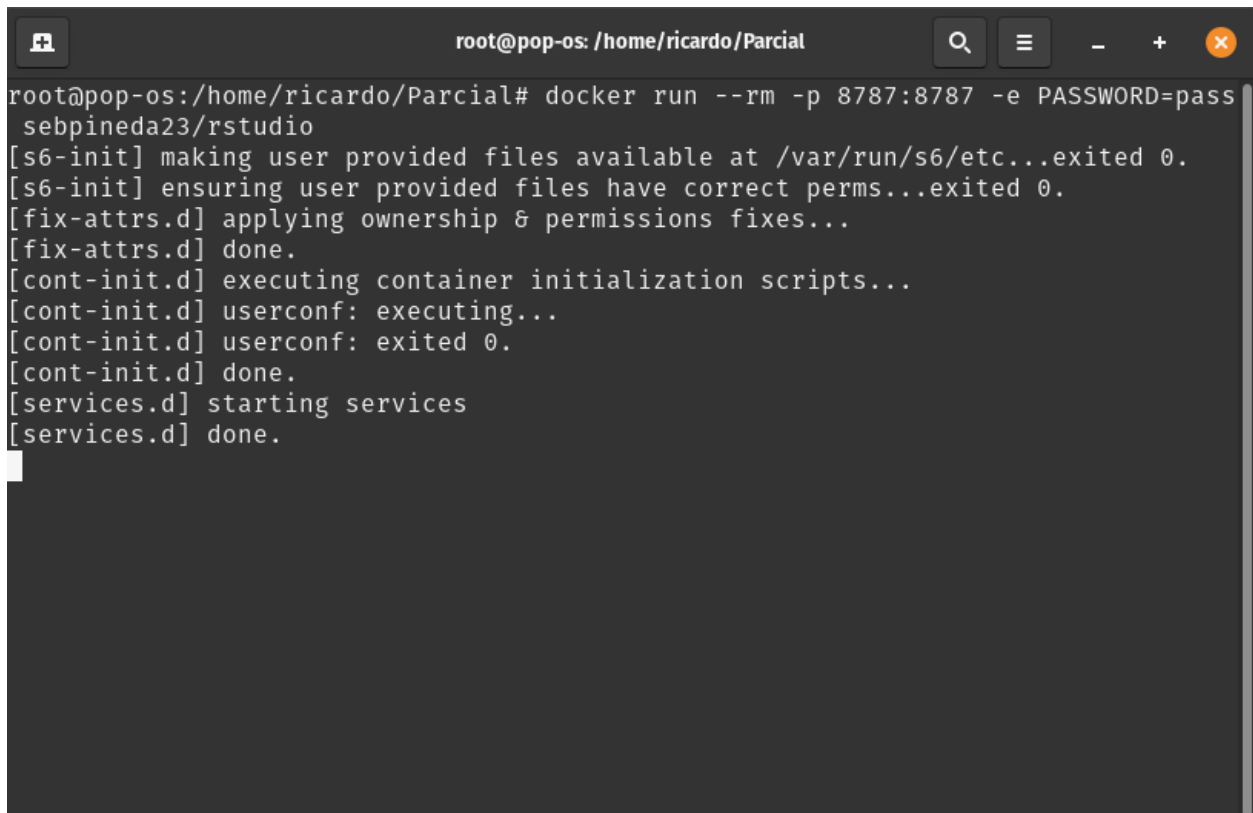
1. Instala las librerías necesarias para el parcial
2. Crea un directorio llamado “parcial”
3. Copia a ese directorio:
 - a. La data necesaria para el análisis
 - b. El R script que hace todo el análisis



```
ricardo@pop-os: ~/Parcial
ricardo@pop-os:~/Parcial$ sudo docker build -t sebpineda23/rstudio -f Dockerfile .
Sending build context to Docker daemon 799.7kB
Step 1/6 : FROM obedaeg/rstudio-mysql
--> 6836db22d1b2
Step 2/6 : RUN R -e "install.packages(c('shinydashboard', 'shinyjs', 'V8', 'lubridate', 'shiny', 'readr', 'DT', 'dygraphs', 'stringr', 'parsedate', 'rbokeh'))"
--> Using cache
--> 00783d4ca47f
Step 3/6 : RUN mkdir -p /home/rstudio/parcial
--> Using cache
--> 78cf3d404ae8
Step 4/6 : WORKDIR /home/rstudio/parcial
--> Using cache
--> 4f1bb9aa8310
Step 5/6 : COPY app.R/ /home/rstudio/parcial
--> 027cfa7a6333
Step 6/6 : COPY data/ /home/rstudio/parcial
--> b112d36987b1
Successfully built b112d36987b1
Successfully tagged sebpineda23/rstudio:latest
ricardo@pop-os:~/Parcial$
```


Ahora ya armamos el contenedor. Toca correrlo con el mismo comando de la vez anterior:

```
docker run --rm -p 8787:8787 -e PASSWORD=yourpasswordhere [nombre-del-contenedor]
```



```
root@pop-os: /home/ricardo/Parcial
root@pop-os:/home/ricardo/Parcial# docker run --rm -p 8787:8787 -e PASSWORD=pass
sebpineda23/rstudio
[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.
```

Podemos ver que ya está levantado correctamente, ahora solo tenemos que ir al browser y poner:

127.0.0.1:[Puerto que especificamos]

The screenshot displays the RStudio IDE interface. The top toolbar includes navigation icons and the address bar shows the URL `127.0.0.1:8787`. Below the toolbar is the menu bar with options: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main editor window shows an R script with the following code:

```
1 # Parcial # 1
2 #
3 ## Cargando las librerías
4 library(shiny)
5 library(readr)
6 library(dplyr)
7 library(RMySQL)
8 library(DT)
9 library(pool)
10 library(lubridate)
11 library(stringr)
12 library(shinyWidgets)
13 library(shinydashboard)
14 library(shinythemes)
15
16 my_db <- dbPool(
17   RMySQL::MySQL(),
18   dbname = "parcial1_dp",
19   host = "172.20.0.2",
20   username = "root",
21   idleTimeout = 20000
22   #password = "root"
23 )
24
25 df_stats <- as.data.frame(my_db %>% tbl("video_stats"))
26 df_stats <- df_stats[!duplicated(df_stats$id), ]
27 df_stats <- df_stats %>%
28   rename(
29     Vistas = viewCount,
30     Likes = likeCount,
31     'No me gusta' = dislikeCount,
32     Favoritos = favoriteCount,
33     Comentarios = commentCount
34   )
35 df_metadata <- as.data.frame(my_db %>% tbl("video_metadata"))
36 df_metadata <- df_metadata %>%
37   rename(
38     id = video_id,
39     'Título' = title,
40     'Descripción' = description,
41     'Código' = iframe,
42     Enlace = link
43   )
44 df_metadata <- df_metadata[!duplicated(df_metadata$id), ]
45 df_stats$id <- substr(df_stats$id, 1, 10)
46 df_videos <- merge(df_metadata, df_stats, by = "id")
47
```

The right-hand pane contains the Environment, History, and Connections tabs. The Environment tab shows the following data objects:

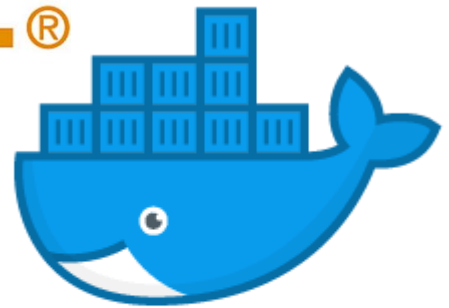
Object	Size
df_meta...	851 obs. of 3 va...
df_stats	859 obs. of 6 va...
df_vide...	861 obs. of 5 va...
my_db	Environment

The Values section shows the following structure:

cols	num	[1:9]	1	2	5	6	...

The bottom pane shows the Files tab with a file explorer view. It displays a folder named `parcial` under the `Home` directory.

Efectivamente, fue creado con éxito el directorio que contiene el Directorio del parcial con los Scripts dentro. El `app.R` es el script de Shiny. El `dashboards.Rmd` es el script de los flexdashboards



docker

Introducción

Esta sección de la documentación estará enfocada específicamente a la dockerización del R-Studio.

Se mostrará una guía paso a paso de como se ejecutó el contenedor de MySQL para el parcial # 1 de Data Product.

Para lograr esto, nos basamos en la imagen oficial de MySQL encontrada en Docker hub.

Procedimiento

Jalar la imagen oficial

El primer paso es jalar la imagen oficial de MySQL desde Docker hub con el siguiente comando:

Docker pull mysql:latest

```
root@pop-os:/home/ricardo# docker pull mysql:latest
latest: Pulling from library/mysql
Digest: sha256:c358e72e100ab493a0304bda35e6f239db2ec8c9bb836d8a427ac34307d074e
d
Status: Image is up to date for mysql:latest
docker.io/library/mysql:latest
```

En mi caso ya estaba descargada, pero si no fuese así, habría que esperar a que se descarge y tomaría unos minutos.

Probar la imagen

Ahora que la imagen fue descargada correctamente, toca probarla. Para ello se utiliza el siguiente comando:

Docker run --name [nombre_libre] -e MYSQL_ROOT_PASSWORD=[Su_contraseña] -d mysql:latest

```
ricardo@pop-os:~$ sudo docker run --name dbtest -e MYSQL_ROOT_PASSWORD=pass -d
mysql:latest
[sudo] password for ricardo:
04d9ac3c33a88380fffe8f5390043187af4d00bd6b94e28c4e7b37bb0eaed408
```

Nos devuelve un hash, lo que quiere decir que está corriendo correctamente. De igual forma lo podemos probar de la siguiente forma:

Docker exec -it [Nombre que le pusimos] "bash"

```
root@pop-os:/home/ricardo# docker exec -it dbtest "bash"
root@04d9ac3c33a8:/# ls
bin    docker-entrypoint-initdb.d  home    media  proc  sbin  tmp
boot  entrypoint.sh              lib     mnt    root  srv   usr
dev    etc                       lib64   opt    run   sys   var
root@04d9ac3c33a8:/#
```

Podemos ver que está corriendo correctamente pues, si ponemos “ls”, nos devuelve los directorios del contenedor

Armar nuestra propia imagen

Ahora que ya instalamos y ejecutamos correctamente la imagen oficial de MySQL, nos toca armar nuestra propia imagen por diversas razones:

1. Para indicarle que jale automaticamente los datos .csv que se utilizarán para poblar las tablas.
2. Para poder especificar nuestras credenciales en el dockerfile.
3. Para poder crear una conexión entre ambos contenedores con más facilidad.

Como hacerlo

Lo primero que hay que hacer, es crear una carpeta con un nombre de nuestra preferencia. En mi caso la carpeta se llamará “SQL”.

Dentro de esa carpeta, tenemos que hacer otra carpeta que se llame “data”, y dentro de ella ubicar los tres archivos .csv.

De vuelta en la carpeta “SQL”, tenemos que hacer dos archivos. Se pueden hacer con cualquier editor de texto, pero yo recomiendo Visual Studio Code que tiene plug ins que nos pueden ayudar en ambos scripts. Esos scripts serán:

1. Dockerfile
2. Load_data.sql

Dockerfile

Este archivo nos sirve para darle instrucciones al contenedor de que acciones ejecutar al ser armado.

```
FROM mysql:latest

# Crea un directorio específico para el parcial
# En el estará la data
RUN mkdir -p /home/parcial/
WORKDIR /home/parcial

# Se definen las credenciales
ENV MYSQL_ROOT_PASSWORD pass
ENV MYSQL_DATABASE parcial1_dp
ENV MYSQL_USER ricardo
ENV MYSQL_PASSWORD pass

# Copia los tres archivos .csv desde "data", al directorio
# que creamos anteriormente
COPY data/ /home/parcial

# Añade nuestro script load_data.sql al entrypoint
```

```
# El entrypoint es lo que se corre de primero al llamar el contenedor
# En el caso de MySQL, este es el directorio default
ADD load_data.sql /docker-entrypoint-initdb.d

# Puerto
EXPOSE 3306
```

Load_data.sql

Este archivo es el que irá dentro del entrypoint del contenedor, es decir, lo que se correrá de primero.

Se pone ahí pues de esta forma las tablas serán pobladas automáticamente, en vez de nosotros tener que poblarlas cada vez que levantamos el contenedor.

```
USE parcial1_dp;

CREATE TABLE video_metadata (
  video_id VARCHAR(20) DEFAULT NULL,
  title VARCHAR(140) DEFAULT NULL,
  link VARCHAR(500) DEFAULT NULL
);

CREATE TABLE video_stats (
  id varchar(20) DEFAULT NULL,
  viewCount int(11) DEFAULT NULL,
  likeCount int(11) DEFAULT NULL,
  dislikeCount int(11) DEFAULT NULL,
  favoriteCount int(11) DEFAULT NULL,
  commentCount int(11) DEFAULT NULL
);

CREATE TABLE videos (
  kind varchar(100) DEFAULT NULL,
  etag varchar(500) DEFAULT NULL,
  id text DEFAULT NULL,
  content_video_id varchar(45) DEFAULT NULL,
  date varchar(45) DEFAULT NULL
);

LOAD DATA INFILE '/home/parcial/academica_videos_metadata.csv'
INTO TABLE video_metadata
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
LOAD DATA INFILE '/home/parcial/academática_video_stats.csv'
INTO TABLE video_stats
FIELDS TERMINATED BY '\,'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE '/home/parcial/academática_videos.csv'
INTO TABLE videos
FIELDS TERMINATED BY '\,'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Este archivo hace varias cosas:

1. Primero crea una base de datos específica para nosotros que se llama parcial1_dp
2. Luego crea tres tablas, una por cada archivo
3. Luego de eso, va al directorio que creamos en el Dockerfile, en el que se encuentran los tres archivos .csv, y va poblando las tablas una por una. Ignora la primera fila por los títulos.

Creación del contenedor

Ahora que ya tenemos los dos scripts, podemos armar nuestro propio contenedor basándonos en ellos.

Nos dirigimos a la carpeta “SQL” en donde están los dos scripts y nuestra carpeta que contiene los tres .csv y ejecutamos el siguiente comando:

Docker build -t [Nombre libre] .

```

root@pop-os:/home/ricardo/Downloads/SQL# docker build -t mysql-parcial .
Sending build context to Docker daemon 219.6kB
Step 1/10 : FROM mysql:latest
---> 0d64f46acfd1
Step 2/10 : RUN mkdir -p /home/parcial/
---> Running in 022d7d807101
Removing intermediate container 022d7d807101
---> f62f35525b38
Step 3/10 : WORKDIR /home/parcial
---> Running in 7a74d14ffa9e
Removing intermediate container 7a74d14ffa9e
---> 37e5350de9d8
Step 4/10 : ENV MYSQL_ROOT_PASSWORD pass
---> Running in 79904d072a5b
Removing intermediate container 79904d072a5b
---> c62eaa9791a4
Step 5/10 : ENV MYSQL_DATABASE parcial1_dp
---> Running in e2c790d29f29
Removing intermediate container e2c790d29f29
---> c0b2ea2118cd
Step 6/10 : ENV MYSQL_USER ricardo
---> Running in 949f5428c533
Removing intermediate container 949f5428c533
---> 0bfb03fdae54
Step 7/10 : ENV MYSQL_PASSWORD pass
---> Running in c864f94239c5
Removing intermediate container c864f94239c5
---> ec8ccb3871b4
Step 8/10 : COPY data/ /home/parcial
---> 781a8fb3c844
Step 9/10 : ADD load_data.sql /docker-entrypoint-initdb.d
---> 070a1421f1ae
Step 10/10 : EXPOSE 3306
---> Running in 40989bac859b
Removing intermediate container 40989bac859b
---> 60d7c128d778
Successfully built 60d7c128d778
Successfully tagged mysql-parcial:latest

```

Podemos ver que se ejecutaron todos los pasos con éxito. Ahora solo nos queda correrlo, para ello utilizamos el siguiente comando:

```
docker run --name db -p 3306:3306 -d [Nombre que pusimos] --secure-file-priv=/home/parcial
```

Nota: Es muy importante especificar el parámetro de `--secure-file-priv=/home/parcial`, pues si no el contenedor no tendrá acceso a ese directorio, y no podrá poblar las tablas.

```

root@pop-os:/home/ricardo/Downloads# docker run --name dbparcial -p 3306:3306
-d mysql-parcial --secure-file-priv=/home/parcial
f2839accef16ae5bc1a1eb770cab6191be91eae38a5d2f060a8e24a3cbc4ba20

```


Efectivamente nos devuelve un hash, lo que quiere decir que está corriendo correctamente. Lo podemos probar de la siguiente forma:

```
Docker exec -it [Nombre que le pusimos] "bash"
```

```
root@pop-os:/home/ricardo/Downloads# docker exec -it dbparcial "bash"
root@f2839accef16:/home/parcial# ls
academática_video_stats.csv  académica_videos_metadata.csv
academática_videos.csv
```

Efectivamente corre, y podemos ver que el directorio default es /home/parcial, justo como lo especificamos en el Dockerfile. Adicionalmente, podemos ver que los tres archivos csv se encuentran dentro del contenedor.

Pero vayamos mas allá, chequeemos que efectivamente la base de datos exista y las tablas estén pobladas:

Para ello, podemos ejecutar el siguiente comando siempre dentro del bash:

```
Mysql -uricardo -ppass
```

El -u indica el usuario, en este caso es "Ricardo" pues ese especificamos dentro del Dockerfile.

El -p es la contraseña, e igualmente en este caso es pass pues ese especificamos en el Dockerfile.

```
root@f2839accef16:/home/parcial# mysql -uricardo -ppass
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.21 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Si logramos acceder correctamente, este debería de ser el output.

Para ver si esta la base de datos, colocamos el siguiente comando:

```
Show databases;
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| parcial1_dp |
+-----+
```

Efectivamente podemos ver que la base de datos “parcial1_dp” existe.

Para utilizarla ponemos:

Use parcial1_dp;

```
mysql> use parcial1_dp;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed
```

Y para listar las tablas dentro de la base de datos, ponemos:

Show tables;

```
mysql> show tables;  
+-----+  
| Tables_in_parcial1_dp |  
+-----+  
| video_metadata        |  
| video_stats           |  
| videos                |  
+-----+  
3 rows in set (0.00 sec)
```

Efectivamente están las tres tablas. Podemos ver los datos de una si utilizamos un select convencional de SQL:

```
mysql> SELECT COUNT(*) FROM videos;  
+-----+  
| COUNT(*) |  
+-----+  
|      861 |  
+-----+  
1 row in set (0.02 sec)
```

Por ejemplo, la tabla de videos. Ahora que ya obtuvimos ese output, estamos listos para hacer la conexión entre ambos contenedores.



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:

Environment	History	Connectio
		
Global Environment		
Data		
	df_meta...	851 obs. of 3 va...
	df_stats	859 obs. of 6 va...
	df_vide...	861 obs. of 5 va...
	my_db	Environment