

Universidad Francisco Marroquín

Data Product

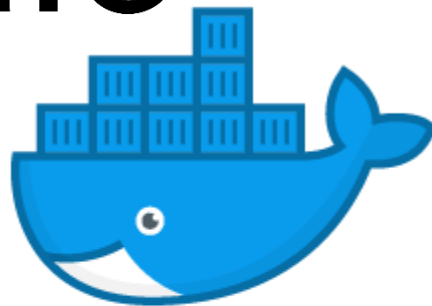
Septiembre del 2020

**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.

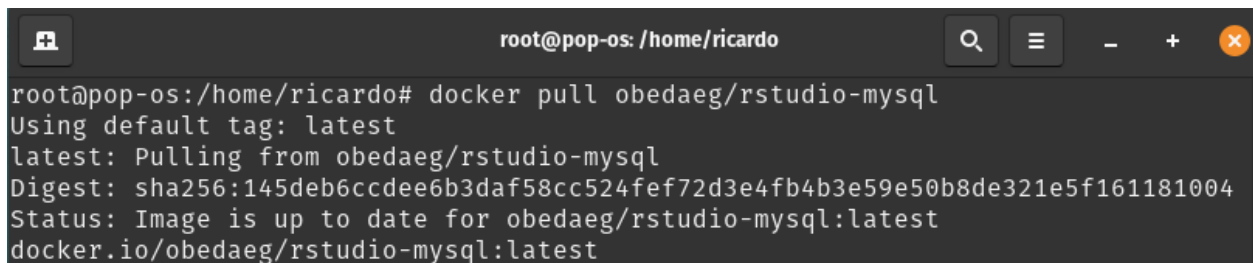
Para lograr esto, de elaborará un contenedor que estará basado en el contenedor de Obed, encontrado en Docker Hub como *obedaeg/rstudio-mysql*.

## 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.

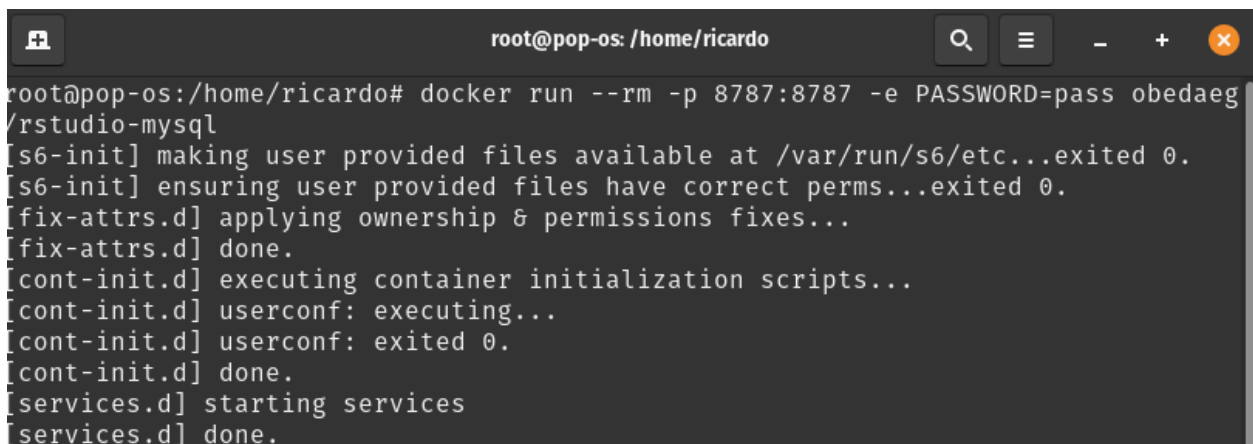
```
root@pop-os:/home/ricardo# docker pull obedaeg/rstudio-mysql
Using default tag: latest
latest: Pulling from obedaeg/rstudio-mysql
Digest: sha256:145deb6ccdee6b3daf58cc524fef72d3e4fb4b3e59e50b8de321e5f161181004
Status: Image is up to date for obedaeg/rstudio-mysql:latest
docker.io/obedaeg/rstudio-mysql:latest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

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:

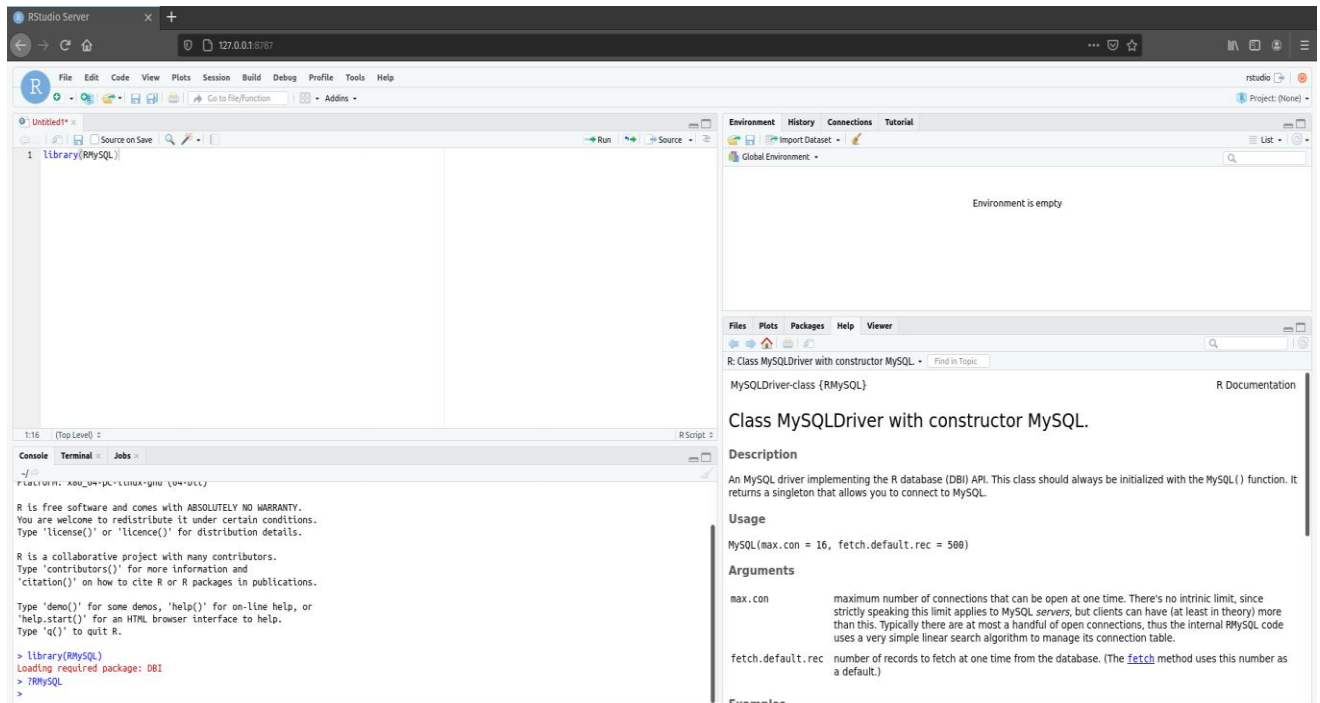
```
docker run --rm -p 8787:8787 -e PASSWORD=yourpasswordhere obedaeg/rstudio-mysql
```

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
```

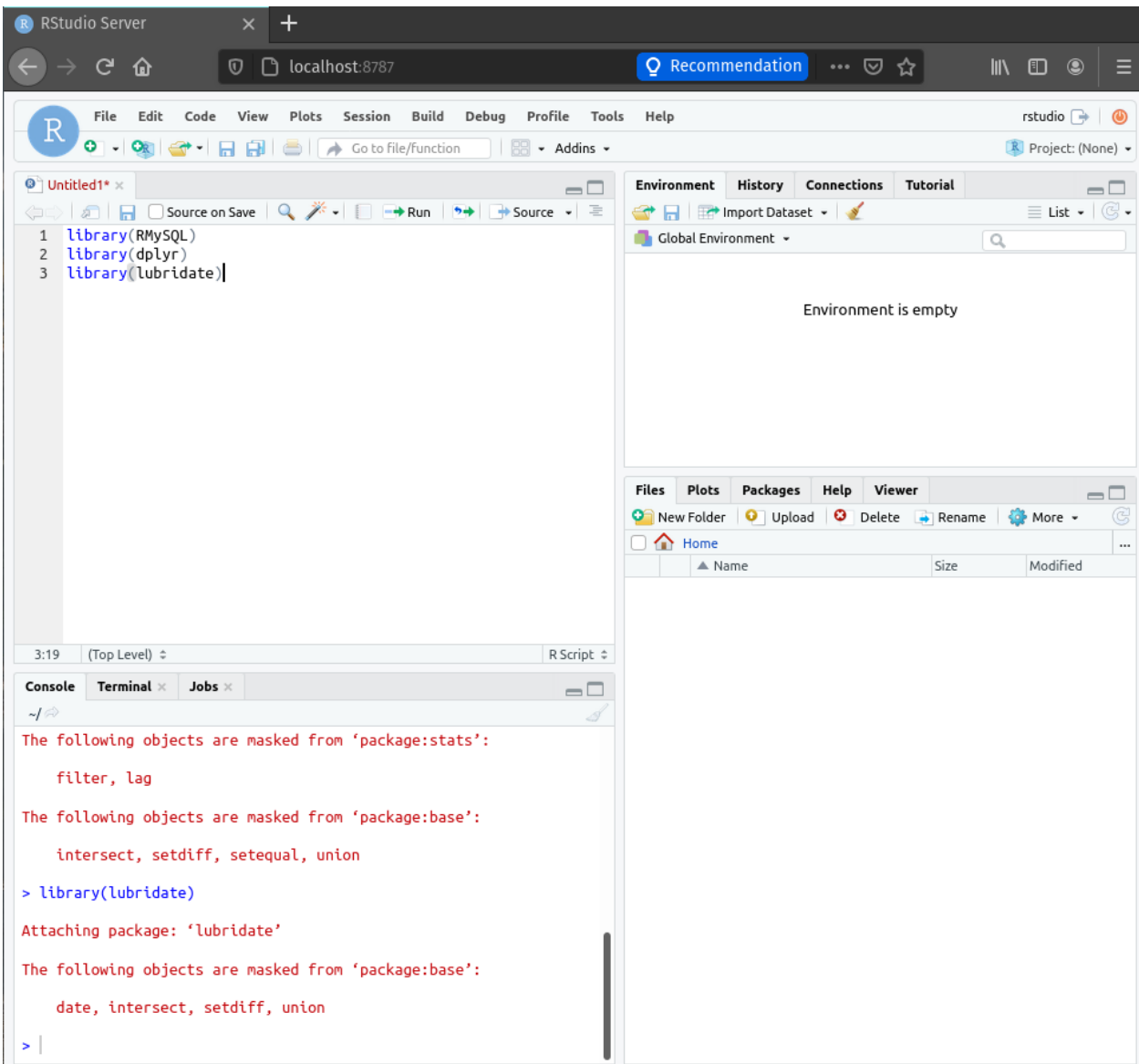
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
parcial_shiny	latest	00783d4ca47f	47 seconds ago	2.24GB

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

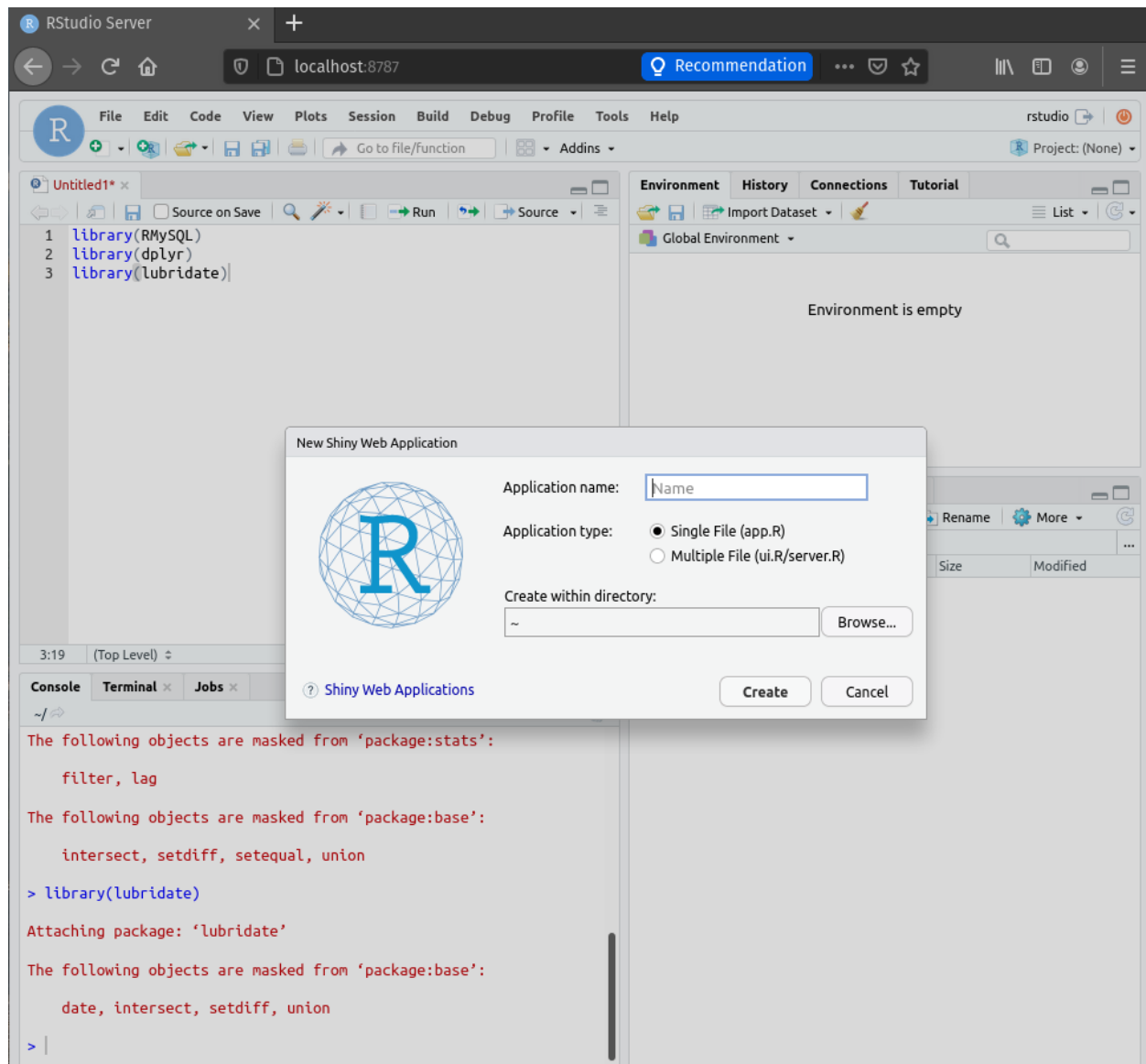
docker run --rm -p 8787:8787 -e PASSWORD=yourpasswordhere [nombre-del-contenedor]

```
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

RUN R -e "install.packages(c('shinydashboard', 'shinyjs', 'V8', 'lubridate', 'shiny', 'readr', 'DT', 'dygraphs', 'stringr', 'parsedate', 'rbokeh'))"

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

# Copiar la data y el código del proyecto

# Código
COPY app.R/ /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. El R script que hace todo el análisis

```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Universidad\ULTIMO\Data Product\Parcial 1\Docker\R-Studio>code .

C:\Universidad\ULTIMO\Data Product\Parcial 1\Docker\R-Studio>docker build -t sebpineda23 -f Dockerfile .
Sending build context to Docker daemon 10.75kB
Step 1/5 : FROM obedaeg/rstudio-mysql
--> 6836db22d1b2
Step 2/5 : RUN R -e "install.packages(c('shinyWidgets', 'shinythemes', 'pool', 'shinydashboard', 'shinyjs', 'V8', 'lubridate', 'shiny', 'readr', 'DT', 'dygraphs', 'stringr', 'parsedate', 'rbokeh'))"
--> Using cache
--> 2b344071061c
Step 3/5 : RUN mkdir -p /home/rstudio/parcial
--> Using cache
--> e02eb2118f15
Step 4/5 : WORKDIR /home/rstudio/parcial
--> Using cache
--> 89189809c8a6
Step 5/5 : COPY app.R/ /home/rstudio/parcial
--> Using cache
--> 05d62fff9cdd
Successfully built 05d62fff9cdd
Successfully tagged sebpineda23:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```



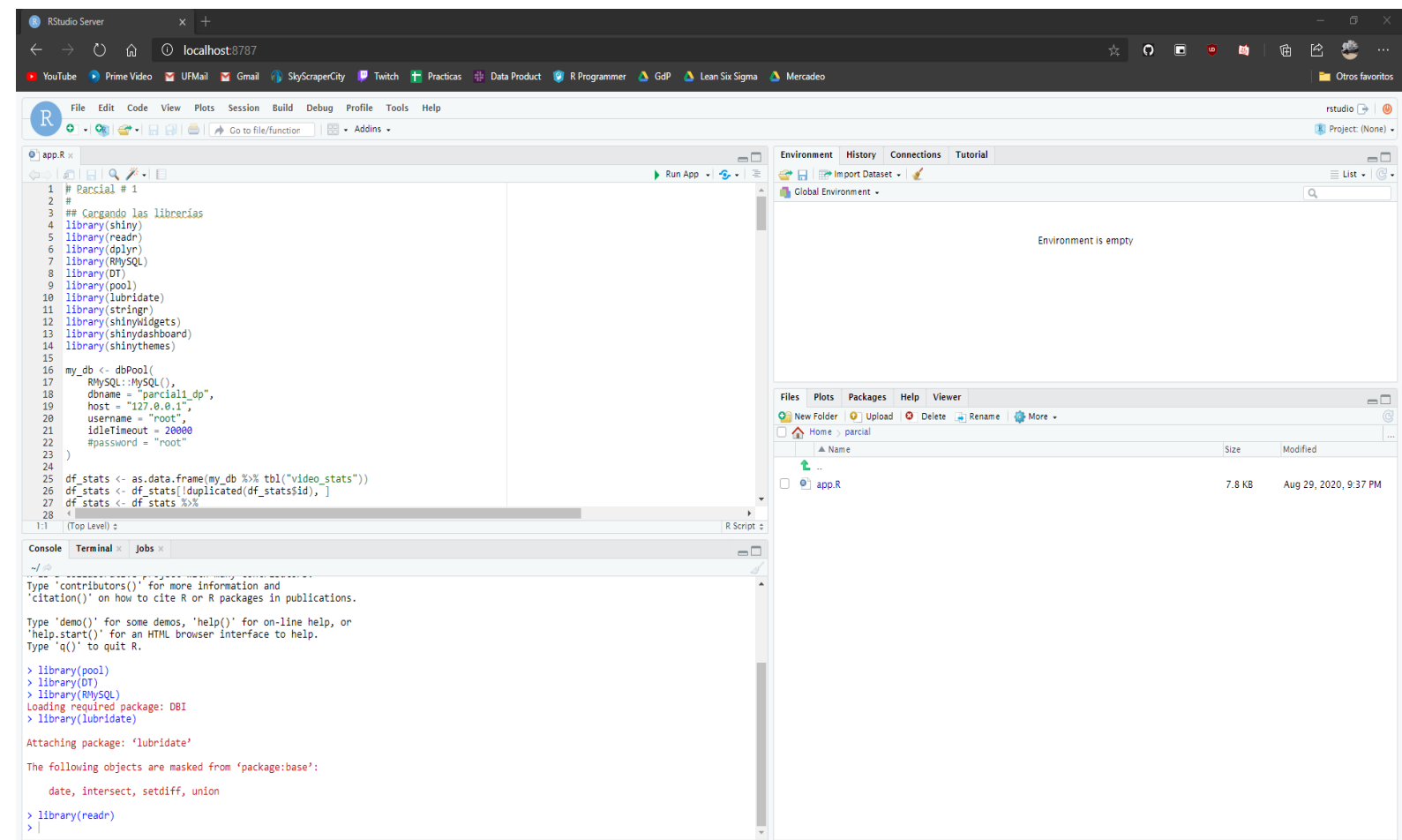
Ahora ya armamos el contenedor. Toca correrlo con el mismo comando de la vez anterior:

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

```
C:\Universidad\ULTIMO\Data Product\Parcial 1\Docke\R-Studio>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]



Efectivamente, fue creado con éxito el directorio que contiene el R Script