UNIVERSIDAD DE GRANADA

Big Data I

máster ciencia de datos e ingeniería de computadores

Cloud Computing y Big Data

Práctica sobre Contenedores

Autor
Ignacio Vellido Expósito
ignaciove@correo.ugr.es

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2020-2021

Índice

1. Contenedor para actividades de ciencia de datos basado en R

1.1. Descripción

Contenedor partiendo de una imagen base de Ubuntu al que se le añade R con distintos paquetes de ciencia de datos, concretamente:

- tidyverse
- caret
- RSNNS
- frbs
- FSinR
- forecast

1.2. Archivo Dockerfile

```
1  FROM ubuntu
2  LABEL author="Ignacio Vellido Expósito"
3  ENV http_proxy http://stargate.ugr.es:3128
4
5  # To don't get asked about geographic location (disable dpkg interactivity)
6  ENV DEBIAN_FRONTEND=noninteractive
7
8  # Install R and tidyverse
9  RUN apt-get update && \
10     apt-get install -y r-base \
11                        r-cran-tidyverse
12
13 RUN R -e "install.packages(c('caret','RSNNS','frbs','FSinR','forecast'), \
14                           dependencies=TRUE, repos='http://cran.rstudio.com/')"
15
16 # Launch test script
17 COPY testDocker.R /home/testDocker.R
18 RUN cd /home && \
19     Rscript /home/testDocker.R > /home/testOutput.txt
20
21 # Launch R
22 CMD [ "R" ]
```

Figura 1

El paquete "tidyverse" es necesario instalarlo a través de apt-get para evitar errores. Se incluye el proceso de testeo dentro del dockerfile para agilizar las pruebas, y se concluye indicando el comando por defecto de ejecución del script.

1.3. Proceso de construcción

1.3.1. En hadoop.ugr.es

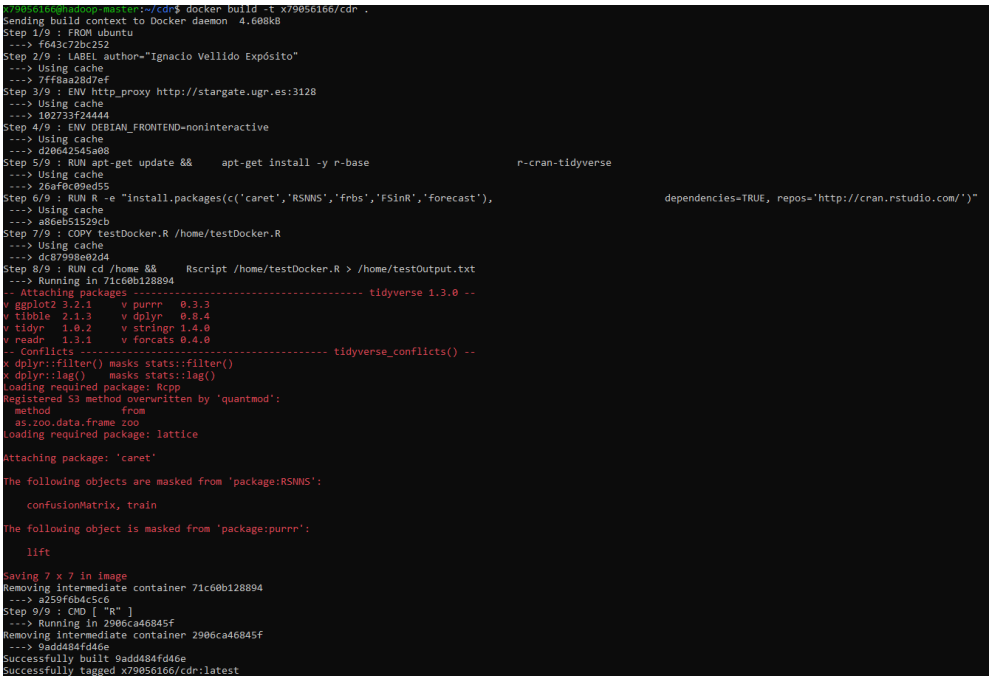

Figura 2: Construcción de la imagen.

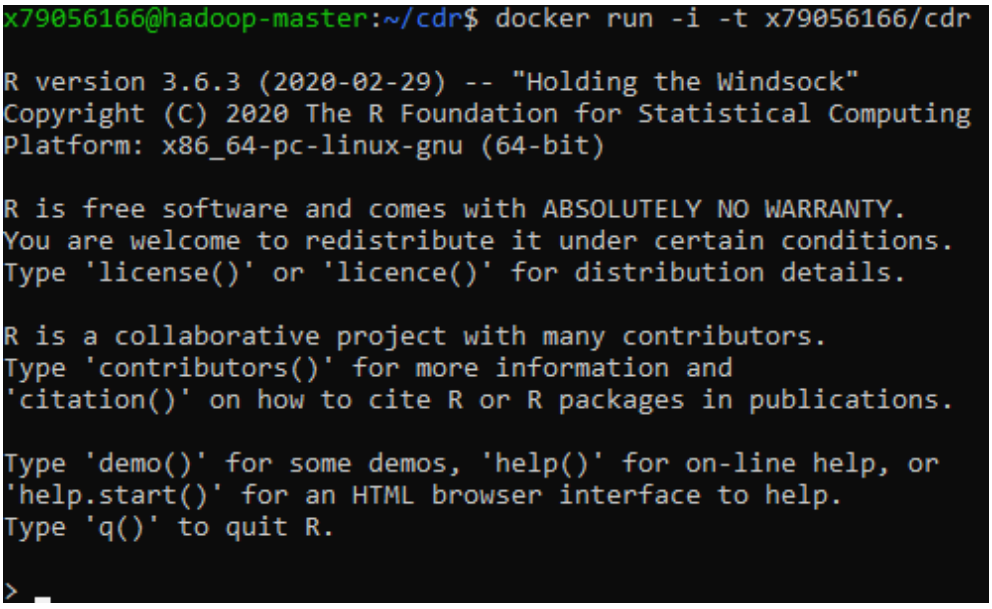

Figura 3: Lanzando la imagen.

```
x79056166@hadoop-master:~/cdr$ docker run -i -t x79056166/cdr /bin/bash
root@5ace4be08114:/# ls /home/
Rplots.pdf  testDocker.R  testGgplot.png  testOutput.txt
root@5ace4be08114:/# cat /home/testOutput.txt
k-Nearest Neighbors

150 samples
  4 predictor
  2 classes: 'negative', 'positive'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 120, 120, 120, 120, 120, 120, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   1  0.9400000  0.9600000  0.9200000
   3  0.9701667  0.9666667  0.9266667
   5  0.9895000  0.9766667  0.9400000
   7  0.9923333  0.9766667  0.9066667
   9  0.9918333  0.9800000  0.9066667
  11  0.9908333  0.9733333  0.9133333

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
```

Figura 4: Contenido de la imagen.

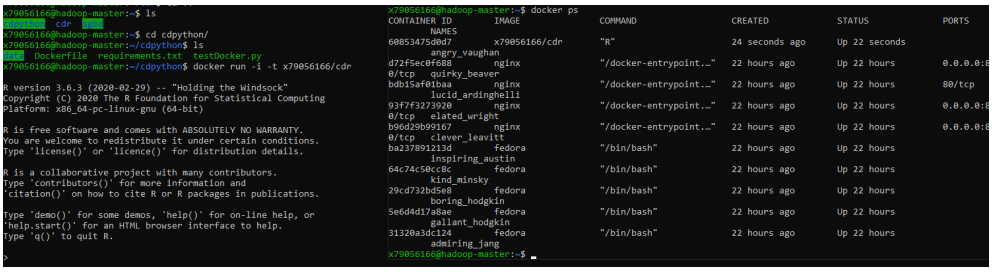

Figura 5: Comprobando ejecución.

1.3.2. En Azure

Tal y como está el archivo Dockerfile, si se desplegara en Azure se crearía una "container instance" que cargaría la imagen ejecutando el script, y se cerraría inmediatamente. Para poder acceder al contenedor antes de que se cierre, es necesario lanzar algún servicio que se mantenga en ejecutando/espera. Una opción sería desplegar una app web como Django o Flask, pero por simplicidad lanzaremos el comando indicado en `https://docs.microsoft.com/en-us/azure/container-instances/container-instances-troubleshooting#issues-during-container-group-runtime`

```
FROM ubuntu
LABEL author="Ignacio Vellido Expósito"
# ENV http_proxy http://stargate.ugr.es:3128

# To don't get asked about geographic location (disable dpkg interactivity)
ENV DEBIAN_FRONTEND=noninteractive

# Install R and tidyverse
RUN apt-get update && \
    apt-get install -y r-base \
                       r-cran-tidyverse

RUN R -e "install.packages(c('caret','RSNNS','frbs','FSinR','forecast'), \
                           dependencies=TRUE, repos='http://cran.rstudio.com/')"

# Launch test script
COPY testDocker.R /home/testDocker.R
RUN cd /home && \
    Rscript /home/testDocker.R > /home/testOutput.txt

# For Azure (to keep container alive)
CMD [ "tail", "-f", "/dev/null" ]
```

Figura 6: Dockerfile modificado.

Para hacer el despliegue, primeramente se crea un repositorio privado en Docker Hub, y se le cambia el nombre a la imagen de hadoop para adaptarla al repositorio.

```
x79056166@hadoop-master:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID,
ub.docker.com to create one.
Username: ignaciove
Password:
WARNING! Your password will be stored unencrypted in /home/x79056166/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
x79056166@hadoop-master:~$ docker push ignaciove/big_data_i:cdr
The push refers to repository [docker.io/ignaciove/big_data_i]
19c81e3a4cbd: Pushed
6cdcaaf11e6c: Pushed
468c259c5bf8: Pushed
0a63c95bf6c7: Pushed
f6253634dc78: Mounted from danijorq/sgbd
9069f84dbbe9: Mounted from danijorq/sgbd
bacd3af13903: Mounted from danijorq/sgbd
cdr: digest: sha256:193718fa7fd7520e7ce6ba2fa8c79c09283a9693948dea4b293be9ac79d9821a size: 1785
```

Figura 7: Subiendo imagen al repositorio.

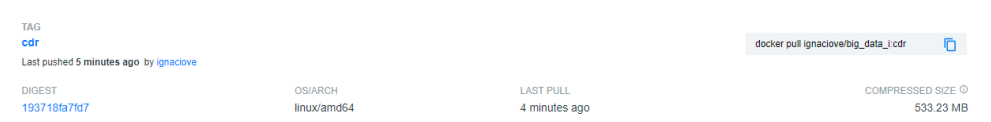

Figura 8: Imagen en Docker Hub.

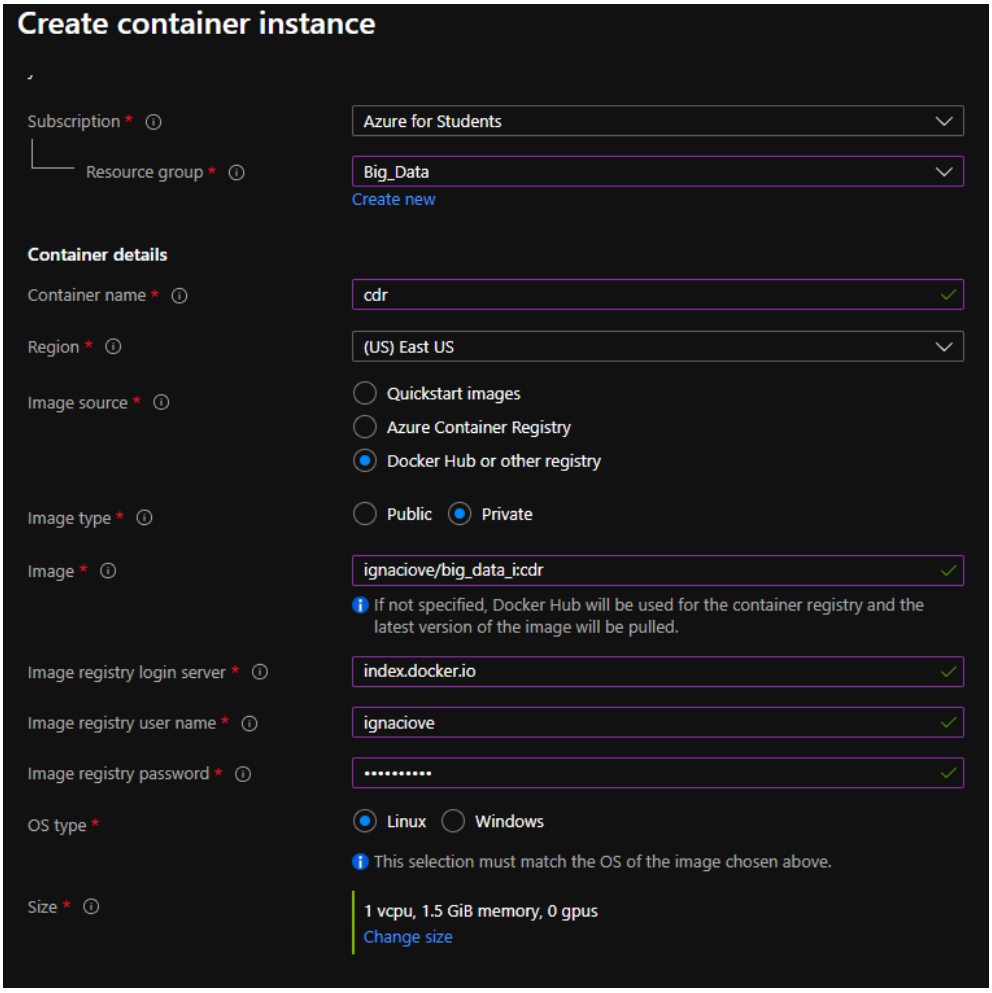

Figura 9: Desplegando el contenedor en Azure.

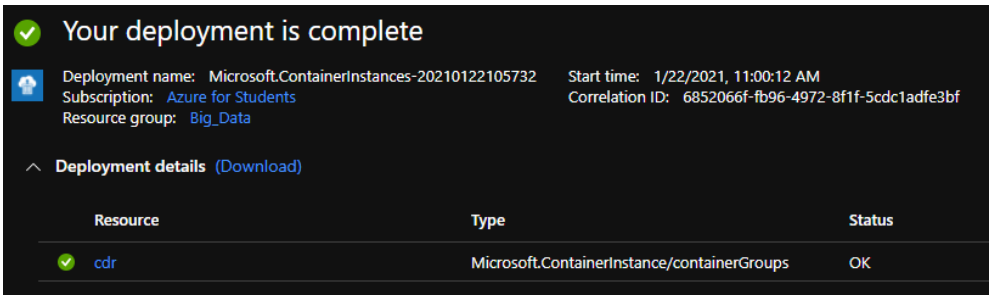

Figura 10: Desplegando el contenedor en Azure.

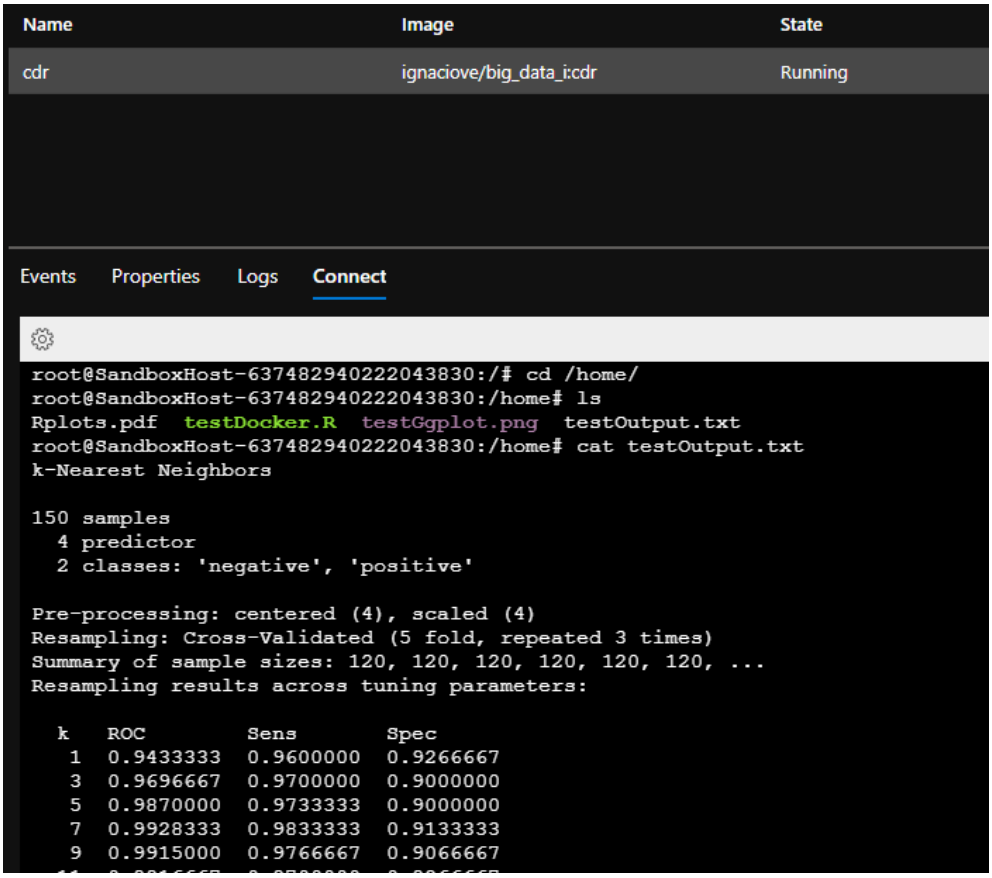

Figura 11: Contenedor en ejecución.

1.4. Evaluación

Para evaluar el contenedor en ambas plataformas se ha comprobado la salida del scripts. En este script se cargan todas las bibliotecas adicionales instaladas y se aplican operaciones con algunas de ellas.

```r
# Test libraries
library(tidyverse)
library(RSNNS)
library(frbs)
library(FSinR)
library(forecast)
library(caret)

# Test ggplot (and tidyverse)
ggplot(iris, aes(x=Sepal.Length, y=Petal.Length))
ggsave("testGgplot.png")

# Test caret
learn_model <-function(dataset, ctrl, message){
  model.fit <- caret::train(Class ~ ., data = dataset, method = "knn",
                            trControl = ctrl, preProcess = c("center","scale"), metric="ROC",
                            tuneGrid = expand.grid(k = c(1,3,5,7,9,11)))
  model.pred <- predict(model.fit,newdata = dataset)
  model.cm <- caret::confusionMatrix(model.pred, dataset$Class,positive = "positive")
  model.probs <- predict(model.fit,newdata = dataset, type="prob")

  return(model.fit)
}

df <- iris
df$Class <- ifelse(df$Species == "virginica", "positive", "negative") %>% as.factor()
df$Species <- NULL

ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary)
model.raw <- learn_model(df, ctrl, "RAW ")

print(model.raw)
```

Figura 12: Script de prueba.

2. Contenedor para actividades de ciencia de datos basado en Python

2.1. Descripción

Contenedor partiendo de una imagen base de Ubuntu al que se le añade Python con distintos paquetes de ciencia de datos, concretamente:

- pandas, numpy, scikit-learn
- seaborn, scipy
- matplotlib, xlrd

2.2. Archivo Dockerfile

Para la construcción del archivo Dockerfile se parte de las recomendaciones de `https://hub.docker.com/_/python` y se adapta para una instalación base de Ubuntu.

```
FROM ubuntu
LABEL author="Ignacio Vellido Expósito"
ENV http_proxy http://stargate.ugr.es:3128

# To don't get asked about geographic location (disable dpkg interactivity)
ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y python3 python3-pip

WORKDIR /usr/src/app

COPY requirements.txt ./

RUN pip3 install --upgrade pip && \
    pip3 install --no-cache-dir -r requirements.txt && \
    rm requirements.txt

# Launch test script
COPY data /home/data
COPY testDocker.py /home/testDocker.py
RUN cd /home && \
    python3 /home/testDocker.py > /home/testOutput.txt

# Launch Python
CMD [ "python3" ]
```

Figura 13

```
cdpython > requirements.txt
1  pandas
2  scikit-learn
3  seaborn
4  scipy
5  numpy
6  matplotlib
7  xlrd
```

Figura 14: Archivo con los paquetes a instalar.

2.3. Proceso de construcción

2.3.1. En hadoop.ugr.es

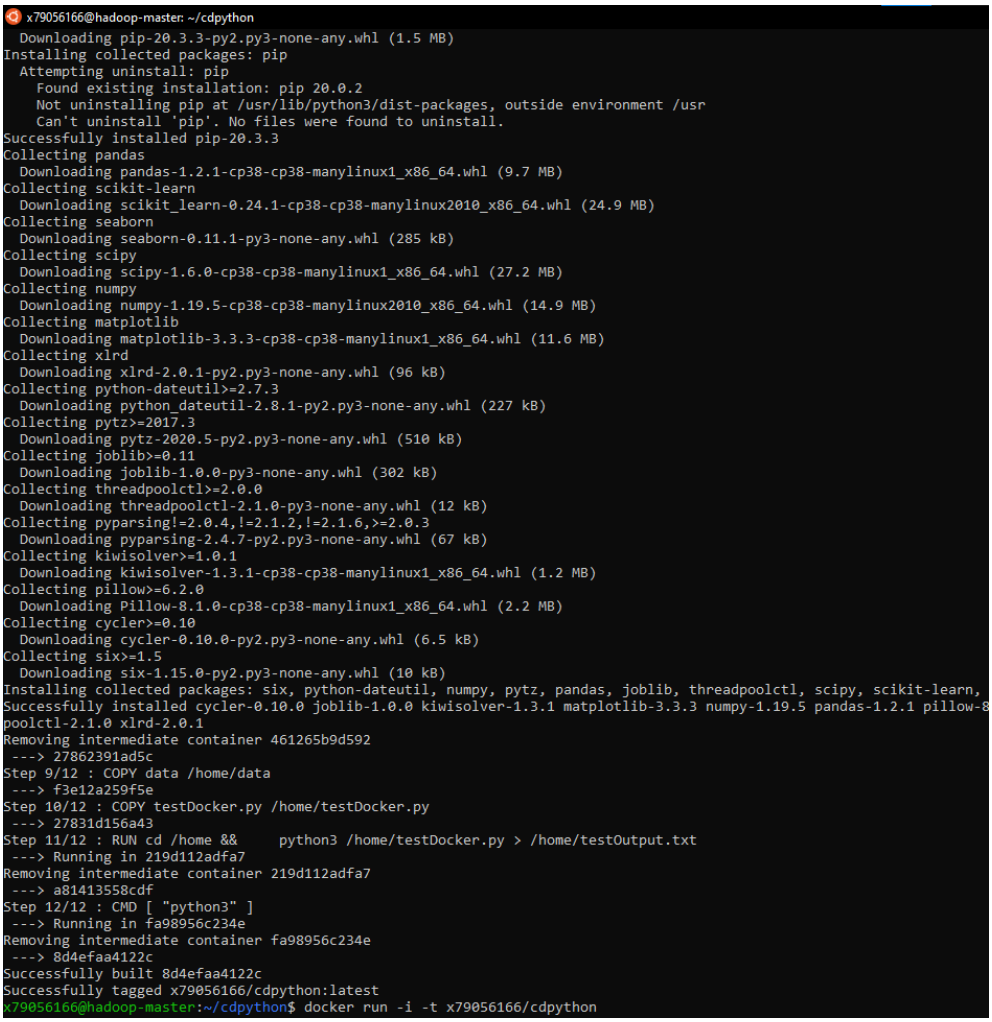

Figura 15: Construcción de la imagen.

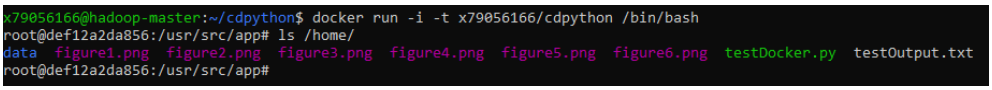

Figura 16: Lanzando la imagen.

```
root@def12a2da856:/usr/src/app# cat /home/testOutput.txt
Datos sin normalizar:
        LBE     LB   AC   FM   UC  ...  Median  Variance  Tendency    E  NSP
1     120.0  120.0  0.0  0.0  0.0  ...   121.0      73.0       1.0  0.0  2.0
2     132.0  132.0  4.0  0.0  4.0  ...   140.0      12.0       0.0  0.0  1.0
3     133.0  133.0  2.0  0.0  5.0  ...   138.0      13.0       0.0  0.0  1.0
4     134.0  134.0  2.0  0.0  6.0  ...   137.0      13.0       1.0  0.0  1.0
5     132.0  132.0  4.0  0.0  5.0  ...   138.0      11.0       1.0  0.0  1.0
...     ...    ...  ...  ...  ...  ...     ...       ...       ...  ...  ...
2122  140.0  140.0  0.0  0.0  6.0  ...   152.0       2.0       0.0  1.0  2.0
2123  140.0  140.0  1.0  0.0  9.0  ...   151.0       3.0       1.0  1.0  2.0
2124  140.0  140.0  1.0  0.0  7.0  ...   152.0       4.0       1.0  1.0  2.0
2125  140.0  140.0  1.0  0.0  9.0  ...   151.0       4.0       1.0  1.0  2.0
2126  142.0  142.0  1.0  1.0  5.0  ...   145.0       1.0       0.0  0.0  1.0

[2126 rows x 25 columns]
----------------------------------------------------------------
Datos normalizados:
        LBE     LB   AC   FM   UC  ...    Mean  Median  Variance  Tendency    E
1     120.0  120.0  0.0  0.0  0.0  ...   137.0   121.0      73.0       1.0  0.0
2     132.0  132.0  4.0  0.0  4.0  ...   136.0   140.0      12.0       0.0  0.0
3     133.0  133.0  2.0  0.0  5.0  ...   135.0   138.0      13.0       0.0  0.0
4     134.0  134.0  2.0  0.0  6.0  ...   134.0   137.0      13.0       1.0  0.0
5     132.0  132.0  4.0  0.0  5.0  ...   136.0   138.0      11.0       1.0  0.0
...     ...    ...  ...  ...  ...  ...     ...     ...       ...       ...  ...
2122  140.0  140.0  0.0  0.0  6.0  ...   150.0   152.0       2.0       0.0  1.0
2123  140.0  140.0  1.0  0.0  9.0  ...   148.0   151.0       3.0       1.0  1.0
2124  140.0  140.0  1.0  0.0  7.0  ...   148.0   152.0       4.0       1.0  1.0
2125  140.0  140.0  1.0  0.0  9.0  ...   147.0   151.0       4.0       1.0  1.0
2126  142.0  142.0  1.0  1.0  5.0  ...   143.0   145.0       1.0       0.0  0.0

[2126 rows x 24 columns]
              LBE           LB  ...     Tendency            E
count  2126.000000  2126.000000  ...  2126.000000  2126.000000
mean    133.303857   133.303857  ...     0.320320     0.033866
std       9.840844     9.840844  ...     0.610829     0.180928
min     106.000000   106.000000  ...    -1.000000     0.000000
25%     126.000000   126.000000  ...     0.000000     0.000000
50%     133.000000   133.000000  ...     0.000000     0.000000
75%     140.000000   140.000000  ...     1.000000     0.000000
max     160.000000   160.000000  ...     1.000000     1.000000

[8 rows x 24 columns]
            counts     freqs
categories
N             1655  0.778457
S              295  0.138758
P              176  0.082785
----------------------------------------------------------------
Clasificando con SVM
----------------------------------------------------------------
Mejores hiperparámetros del modelo:
{'C': 100000.0, 'gamma': 0.001, 'kernel': 'poly'}

Mejor score obtenido:
0.8505882352941176

Resultados de la predicción sobre test:
              precision    recall  f1-score   support

      Normal       0.87      0.96      0.91       325
     Suspect       0.86      0.71      0.78        42
  Pathologic       0.53      0.29      0.37        59
```

Figura 17: Contenido de la imagen.

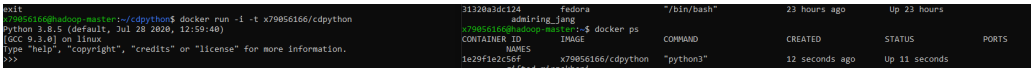

Figura 18: Comprobando ejecución.

2.3.2. En Azure

Los pasos para desplegar la imagen en Azure son los mismos que con el contenedor de R. Es necesario modificar el Dockerfile y subirlo al repositorio, e indicarle a Azure dónde encontrar la imagen.

```
1  FROM ubuntu
2  LABEL author="Ignacio Vellido Expósito"
3  # ENV http_proxy http://stargate.ugr.es:3128
4
5  # To don't get asked about geographic location (disable dpkg interactivity)
6  ENV DEBIAN_FRONTEND=noninteractive
7
8  RUN apt-get update && \
9      apt-get install -y python3 python3-pip
10
11 WORKDIR /usr/src/app
12
13 COPY requirements.txt ./
14
15 RUN pip3 install --upgrade pip && \
16     pip3 install --no-cache-dir -r requirements.txt && \
17     rm requirements.txt
18
19 # Launch test script
20 COPY data /home/data
21 COPY testDocker.py /home/testDocker.py
22 RUN cd /home && \
23     python3 /home/testDocker.py > /home/testOutput.txt
24
25
26 # For Azure (to keep container alive)
27 CMD [ "tail", "-f", "/dev/null" ]
```

Figura 19: Dockerfile desplegado en Azure.

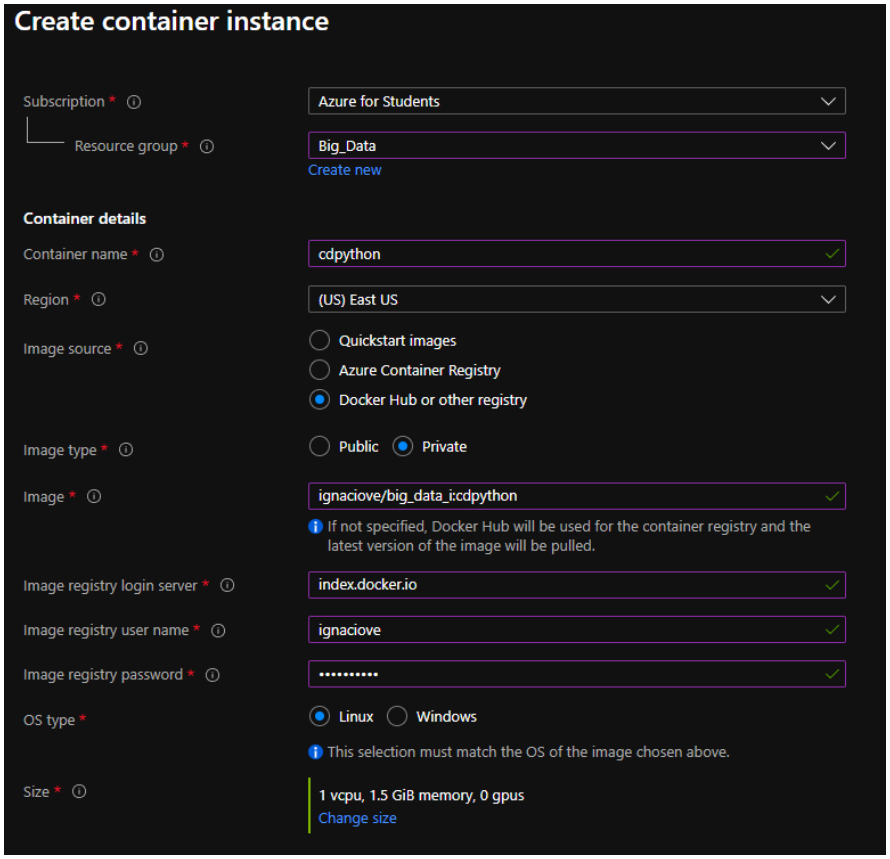

Figura 20: Despliegue del contenedor en Azure.

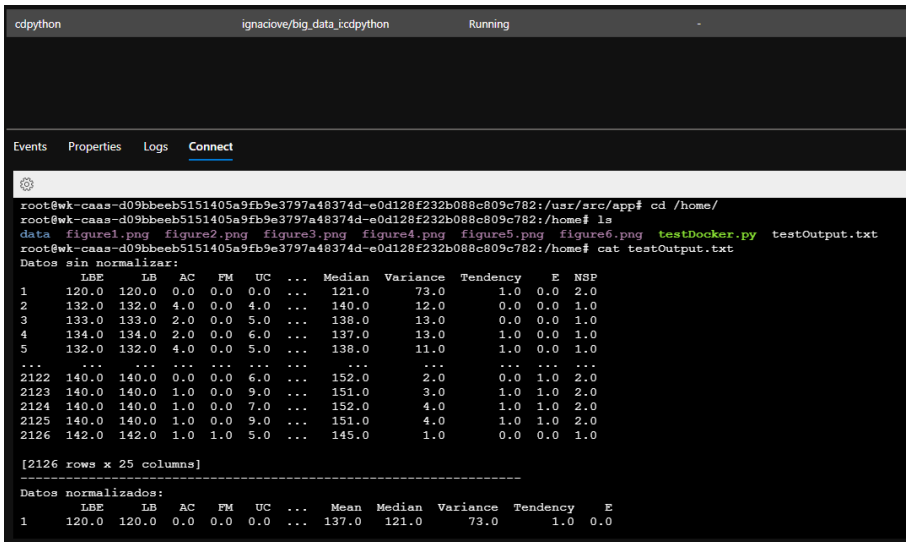

Figura 21: Imagen del contenedor en ejecución.

2.4. Evaluación

Para evaluar el correcto funcionamiento se lanza el siguiente script, que carga los paquetes instalados y realiza un aprendizaje sobre un conjunto de datos con SVM.

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-
##############################################################################
# Librerías
##############################################################################

import random

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Preprocesamiento
from sklearn.preprocessing import Normalizer
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA

# Algoritmos
from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

# Evaluación
from sklearn.metrics import classification_report, \
                            confusion_matrix, \
                            plot_confusion_matrix

##############################################################################
# Lectura
##############################################################################

# Semilla con la que se han analizado los resultados
random.seed(9999)

# Cargamos los datos (sheet Raw Data nos es más cómodo que Data)
data = pd.read_excel("data/CTG.xls", "Raw Data")

# Eliminamos las 3 últimas filas que solo contienen valores de máximos y mínimos
data = data[:-3]

# Eliminamos la primera fila que está vacía
data = data[1:]

# Eliminamos las columnas que no contienen información relevante para la
# clasificación
removed_columns = ["FileName", "Date", "SegFile", "b", "e", "A", "B", "C", "D",
                   "AD", "DE", "LD", "FS", "SUSP", "CLASS"]
```

Figura 22: Parte del script de prueba.

3. Contenedor con SGDB MySQL

3.1. Descripción

Contenedor docker partiendo de una instalación base de MySQL, permitiendo consultas desde el exterior del contenedor.

3.2. Archivo Dockerfile

```
1  FROM mysql
2  LABEL author="Ignacio Vellido Expósito"
3
4  # Upload test DB
5  ENV MYSQL_DATABASE testDB
6  ENV MYSQL_USER testUser
7  ENV MYSQL_PASSWORD testPass
8
9  # All scripts in docker-entrypoint-initdb.d/ are automatically
10 # executed during container startup
11 COPY testDocker.sql /docker-entrypoint-initdb.d/
```

Figura 23

Creamos en el archivo Dockerfile un usuario para realizar las pruebas e incluímos una base de datos a partir de un script.

3.3. Proceso de construcción

3.3.1. En local

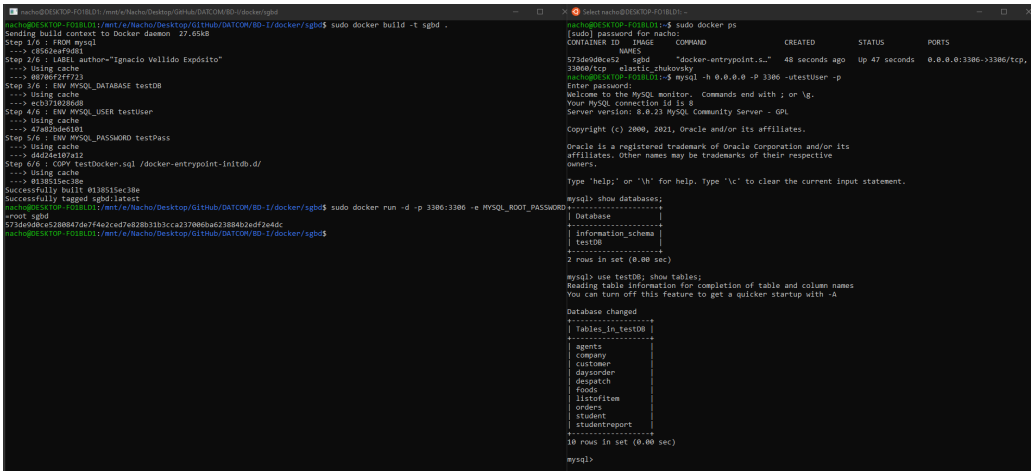

Figura 24: Construcción del contenedor y ejecución desde fuera de él.

```
nacho@DESKTOP-FO1BLD1:~$ mysql -h 0.0.0.0 -P 3306 -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testDB             |
+--------------------+
5 rows in set (0.00 sec)
```

Figura 25: Entrando como usuario root.

3.3.2. En Azure

Una vez más para desplegar el contenedor en Azure subimos la imágen al repositorio y creamos una nueva "container instance".

Para este contenedor es necesario indicarle a Azure el puerto exportado en la imagen, y modificar el Dockerfile para incluir la contraseña del root (puesto que no se le puede indicar a Azure en el momento de ejecución).

```
1  FROM mysql
2  LABEL author="Ignacio Vellido Expósito"
3
4  # Upload test DB
5  ENV MYSQL_DATABASE testDB
6  ENV MYSQL_USER testUser
7  ENV MYSQL_PASSWORD testPass
8  ENV MYSQL_ROOT_PASSWORD root
9
10 # All scripts in docker-entrypoint-initdb.d/ are automatically
11 # executed during container startup
12 COPY testDocker.sql /docker-entrypoint-initdb.d/
```

Figura 26: Dockerfile modificado.

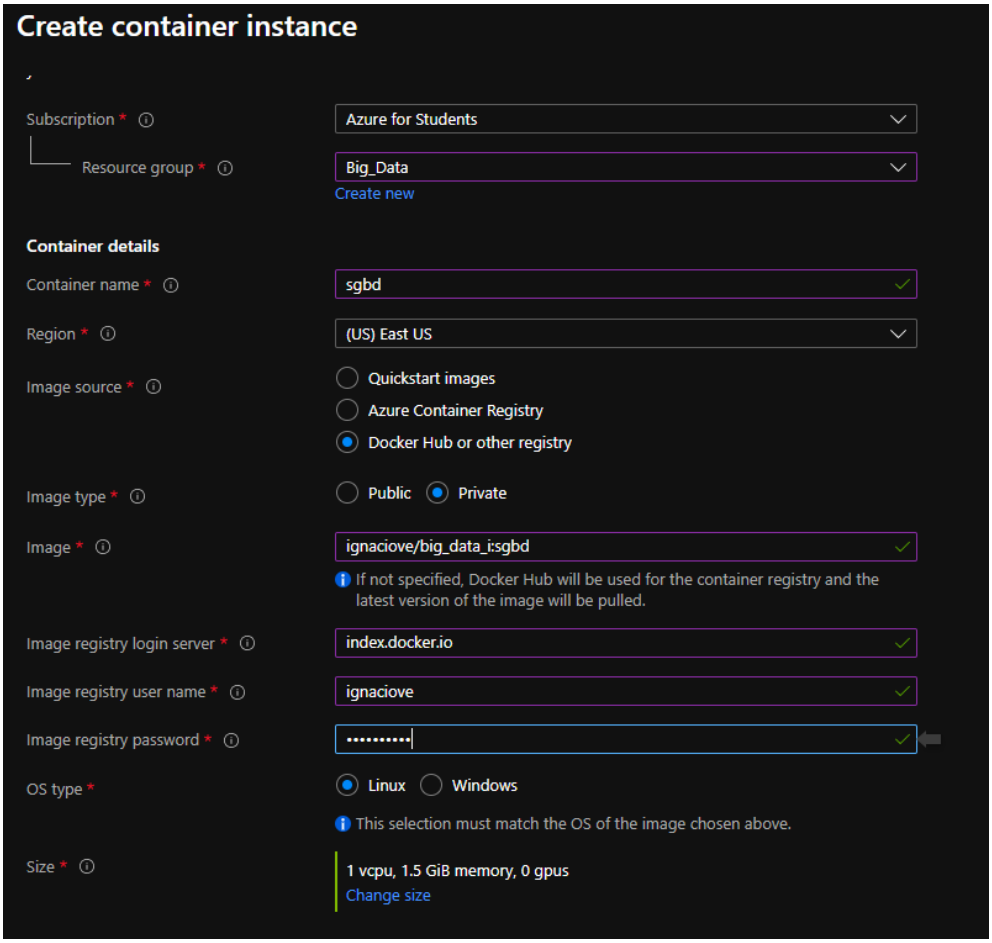

Figura 27: Creando contenedor.

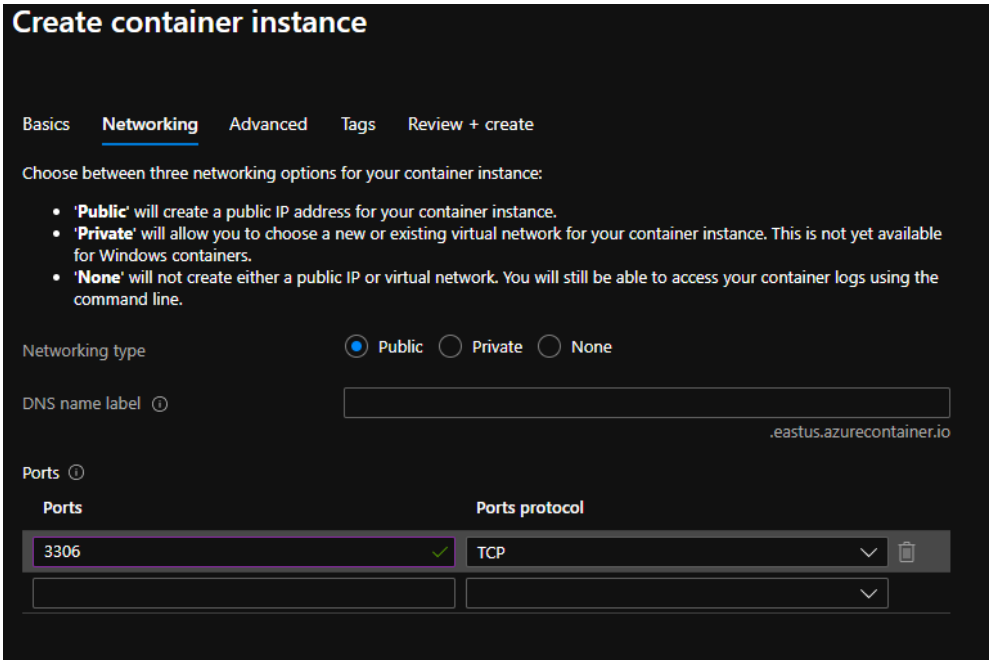

Figura 28: Indicando el puerto exportado.

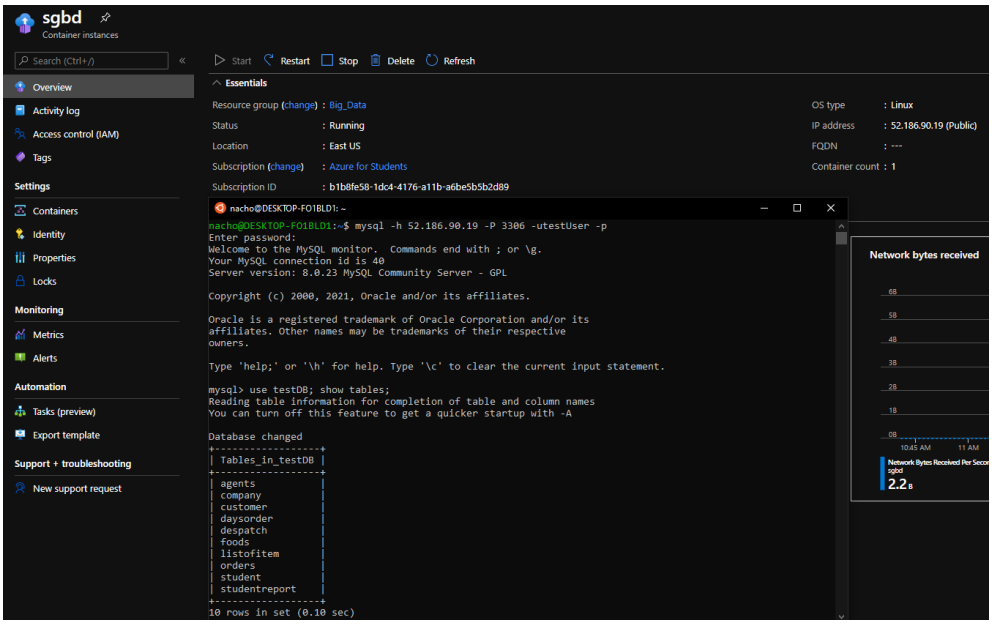

Figura 29: Accediendo a través de la IP pública.

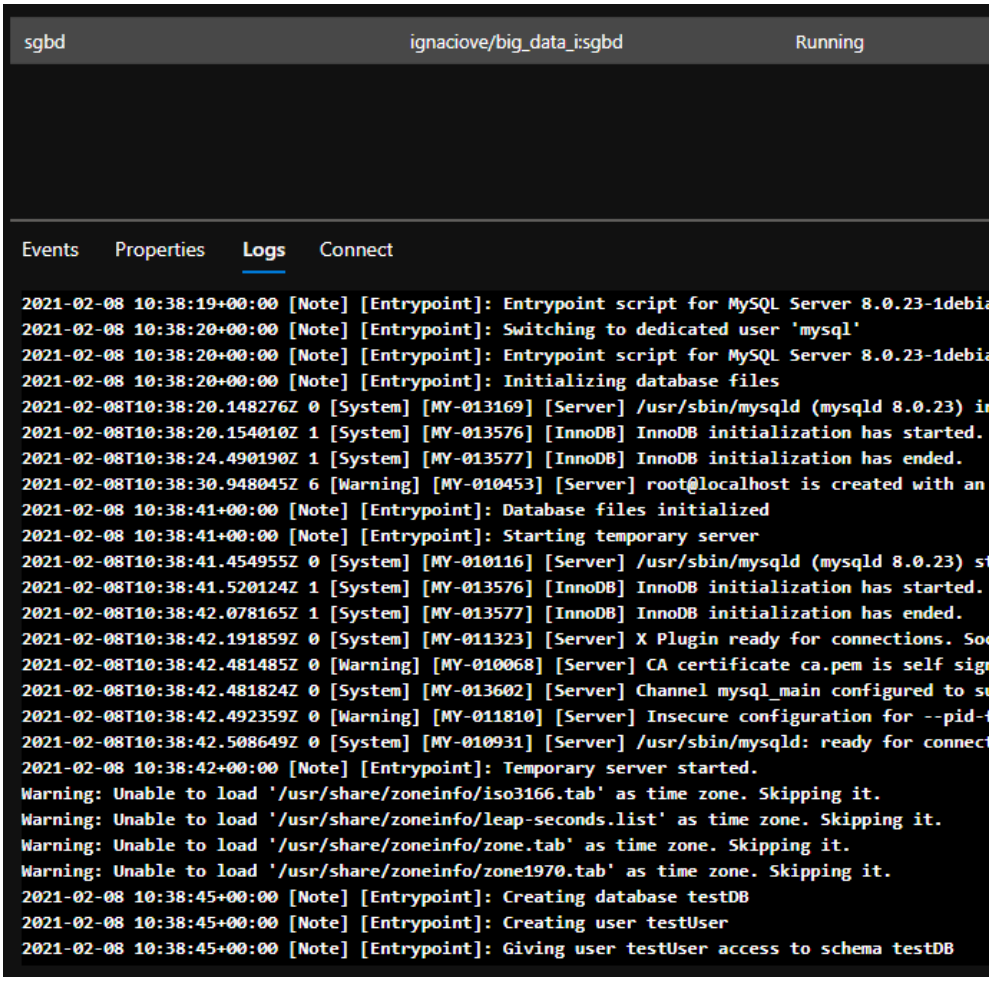

Figura 30: Logs del contenedor.

3.4. Evaluación

La evaluación únicamente se basa en poder acceder al usuario creado en el Dockerfile, corroborando la correcta inclusión de la base de datos de prueba.

```
--
-- Database: `sample`
--

-- --------------------------------------------------------

--
-- Table structure for table `agents`
--

CREATE TABLE IF NOT EXISTS `agents` (
  `AGENT_CODE` varchar(6) NOT NULL DEFAULT '',
  `AGENT_NAME` varchar(40) DEFAULT NULL,
  `WORKING_AREA` varchar(35) DEFAULT NULL,
  `COMMISSION` decimal(10,2) DEFAULT NULL,
  `PHONE_NO` varchar(15) DEFAULT NULL,
  `COUNTRY` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`AGENT_CODE`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `agents`
--

INSERT INTO `agents` (`AGENT_CODE`, `AGENT_NAME`, `WORKING_AREA`, `COMMISSION`, `PHONE_NO`, `COUNTRY`) VALUES
('A007  ', 'Ramasundar                ', 'Bangalore                ', '0.15', '077-25814763   ', '\r'),
('A003  ', 'Alex                      ', 'London                   ', '0.13', '075-12458969   ', '\r'),
('A008  ', 'Alford                    ', 'New York                 ', '0.12', '044-25874365   ', '\r'),
('A011  ', 'Ravi Kumar                ', 'Bangalore                ', '0.15', '077-45625874   ', '\r'),
('A010  ', 'Santakumar                ', 'Chennai                  ', '0.14', '007-22388644   ', '\r'),
('A012  ', 'Lucida                    ', 'San Jose                 ', '0.12', '044-52981425   ', '\r'),
('A005  ', 'Anderson                  ', 'Brisban                  ', '0.13', '045-21447739   ', '\r'),
('A001  ', 'Subbarao                  ', 'Bangalore                ', '0.14', '077-12346674   ', '\r'),
('A002  ', 'Mukesh                    ', 'Mumbai                   ', '0.11', '029-12358964   ', '\r'),
('A006  ', 'McDen                     ', 'London                   ', '0.15', '078-22255588   ', '\r'),
('A004  ', 'Ivan                      ', 'Torento                  ', '0.15', '008-22544166   ', '\r'),
('A009  ', 'Benjamin                  ', 'Hampshair                ', '0.11', '008-22536178   ', '\r');
```

Figura 31: Fragmento de la base de datos insertada.