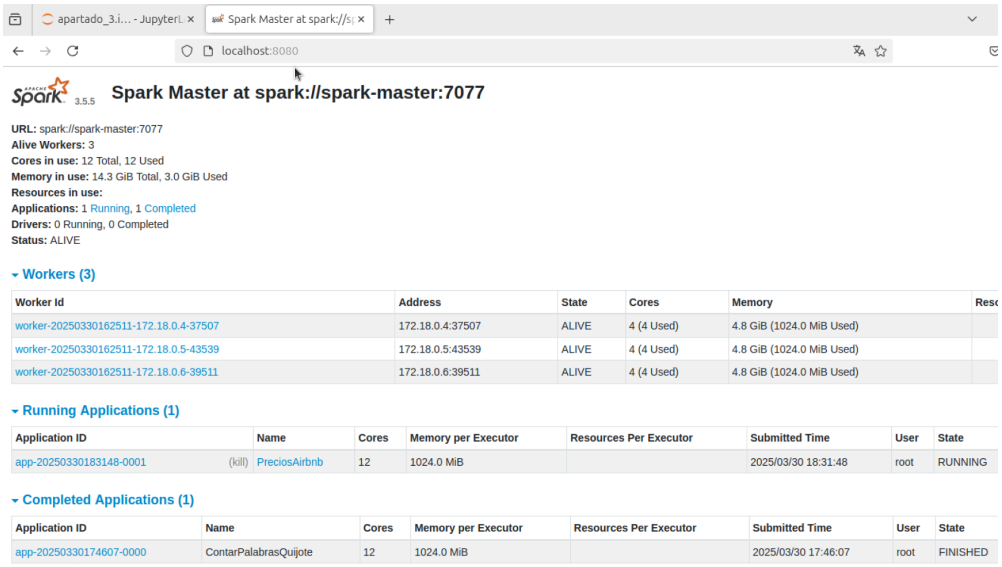
Tarea evaluable CE_5075 7.1

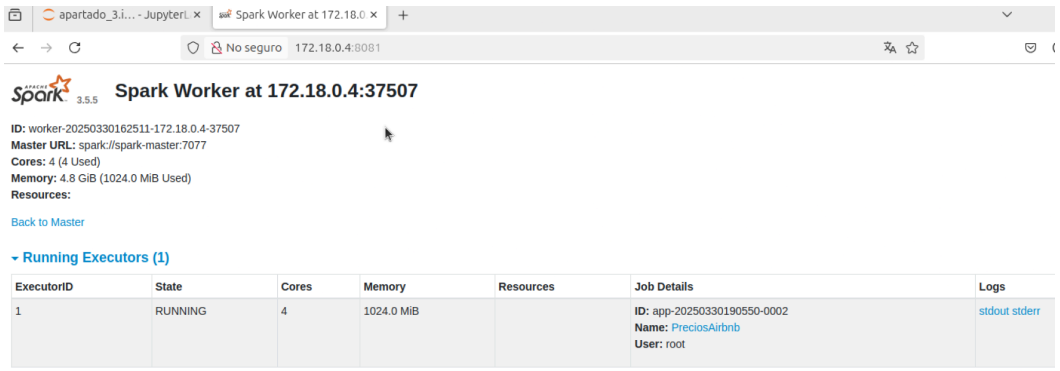
Big data aplicado

David Ramírez Ruiz
30/03/25

APARTADO 3

En este enlace puedes encontrar un archivo con los precios de todos los alojamientos de Airbnb en la isla de Menorca, con datos actualizados al 31 de diciembre de 2024.





Al igual que en el apartado anterior dejaré el cuaderno de Jupyter que contiene el código ejecutado entre los archivos de la entrega. Aún así aquí van algunas capturas del cuaderno

```
from pyspark import SparkContext

# Inicializar SparkContext
sc = SparkContext("spark://spark-master:7077", "PreciosAirbnb")
sc
```

```
/usr/local/lib/python3.9/dist-packages/pyspark/bin/load-spark-env.sh: line 68: ps: command not found
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/03/30 18:31:47 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
ere applicable
```

SparkContext

Spark UI

Version	`v3.5.5`
Master	`spark://spark-master:7077`
AppName	`PreciosAirbnb`

Crea otro cuaderno de Jupyter en el nodo con JupyterLab de tu clúster, donde, utilizando PySpark y RDDs (no puedes usar DataFrames de Spark SQL), debes calcular:

1. El número de alojamientos (número de valores)

```
# Cargar datos desde el archivo
rdd = sc.textFile('preus.txt')
# Parsear los datos a flotante y eliminar encabezados y valores no numéricos
prices_rdd = rdd.filter(lambda x: x.strip().isdigit()).map(lambda x: float(x.strip()))
```

```
# Número de alojamientos
num_accommodations = prices_rdd.count()
print(f'Número de alojamientos: {num_accommodations}')
```

```
[Stage 0:>                                                          (0 + 2) / 2]
Número de alojamientos: 1048253
```

2. Los valores mínimo y máximo

```
# Valor mínimo y máximo de los precios
min_price = prices_rdd.min()
max_price = prices_rdd.max()
print(f'Mínimo: {min_price} - Máximo: {max_price}')
```

```
Mínimo: 10.0 - Máximo: 23229.0
```

3. La media y la desviación estándar

```
# Media y desviación estándar
mean_price = prices_rdd.mean()
stdev_price = prices_rdd.stdev()
print(f'Media: {mean_price} - Desviación estándar: {stdev_price}')
```

```
[Stage 4:>                                                          (0 + 2) / 2]
Media: 436.0350125399137 - Desviación estándar: 1188.3904253821981
```

4. Se nos indica que todos los alojamientos con un precio inferior a 1000 euros lo aumentarán en un 2%, mientras que los de 1000 euros o más lo harán en un 3%. Calcula la nueva media y desviación estándar después de aplicar estos aumentos.

```
# Actualizar valores de los precios
new_price_rdd = prices_rdd.map(lambda x: x * 1.02 if x < 1000 else x * 1.03)
```

```
# Nueva media y desviación estándar
new_mean_price = new_price_rdd.mean()
new_stdev_price = new_price_rdd.stdev()
print(f'Nueva media: {new_mean_price} - Nueva desviación estándar: {new_stdev_price}')
```

```
[Stage 6:>                                                          (0 + 2) / 2]
Nueva media: 446.8178788517616 - Nueva desviación estándar: 1224.097335293833
```

5. Ahora queremos definir los siguientes intervalos de precios:
    - Grupo 1: Hasta 150 euros (incluido)
    - Grupo 2: Desde 151 euros hasta 300 euros, ambos incluidos
    - Grupo 3: Desde 301 euros hasta 500 euros, ambos incluidos
    - Grupo 4: Más de 500 euros

    Debes calcular cuántos alojamientos hay en cada grupo.

```
price_groups = {
    'Grupo 1': new_price_rdd.filter(lambda x: x <= 150).count(),
    'Grupo 2': new_price_rdd.filter(lambda x: 151 <= x <= 300).count(),
    'Grupo 3': new_price_rdd.filter(lambda x: 301 <= x <= 500).count(),
    'Grupo 4': new_price_rdd.filter(lambda x: x > 500).count()
}
for group, count in price_groups.items():
    print(f'{group}: {count}')
```

```
Grupo 1: 396986
Grupo 2: 300880
Grupo 3: 145719
Grupo 4: 199829
```

```
# Detener SparkContext
```