

DAY-12 (Spark Assignment - 2)

Mitushi Vishwakarma

Creating Compute in DataBricks :

Databricks is a unified, open analytics platform for building, deploying, sharing, and maintaining enterprise-grade data, analytics, and AI solutions at scale.

Databricks compute refers to the selection of computing resources you can provision in your Databricks workspace. We need compute to run data engineering, data science, and data analytics workloads

The screenshot shows the 'New compute' page in the Databricks UI. At the top, there's a breadcrumb 'Compute > New compute >'. Below it, the title 'New compute' is followed by 'Cancel' and 'Create compute' buttons. To the right, resource allocation is shown: '0 Workers: 0 GB Memory, 0 Cores, 0 DBU' and '1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU'. The 'Compute name' field contains 'TEST'. The 'Databricks runtime version' dropdown is set to 'Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)'. Under the 'Instance' section, a message states: 'Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).' At the bottom, there are tabs for 'Instances' (selected) and 'Spark'.

Compute > New compute >

New compute

Cancel

Create compute

0 Workers: 0 GB Memory, 0 Cores, 0 DBU

1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU

Compute name

TEST

Databricks runtime version

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)

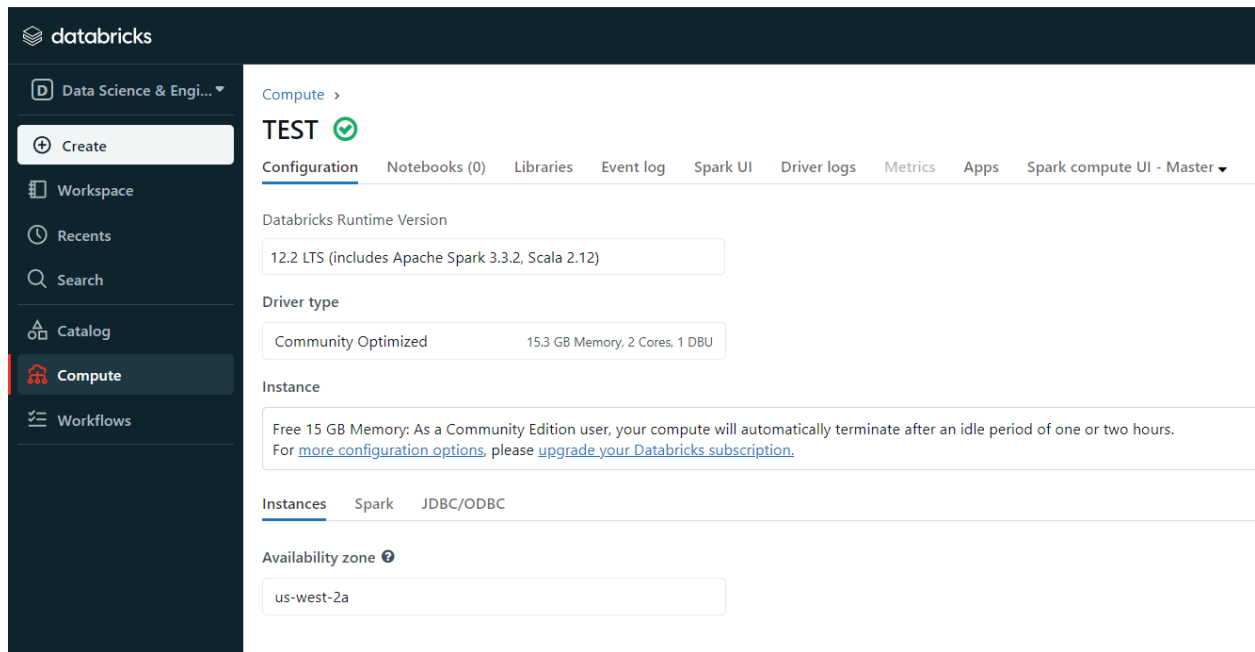
Instance

Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).

Instances

Spark

Compute created successfully.



Before using pyspark in jupyter notebook, We need to run the following command

Pip install pyspark

Then from pyspark.sql module import SparkSession method to create a session as a variable. Then using spark variable we read csv file using read.csv() and displayed data using .show ().

```
[2]: import pyspark

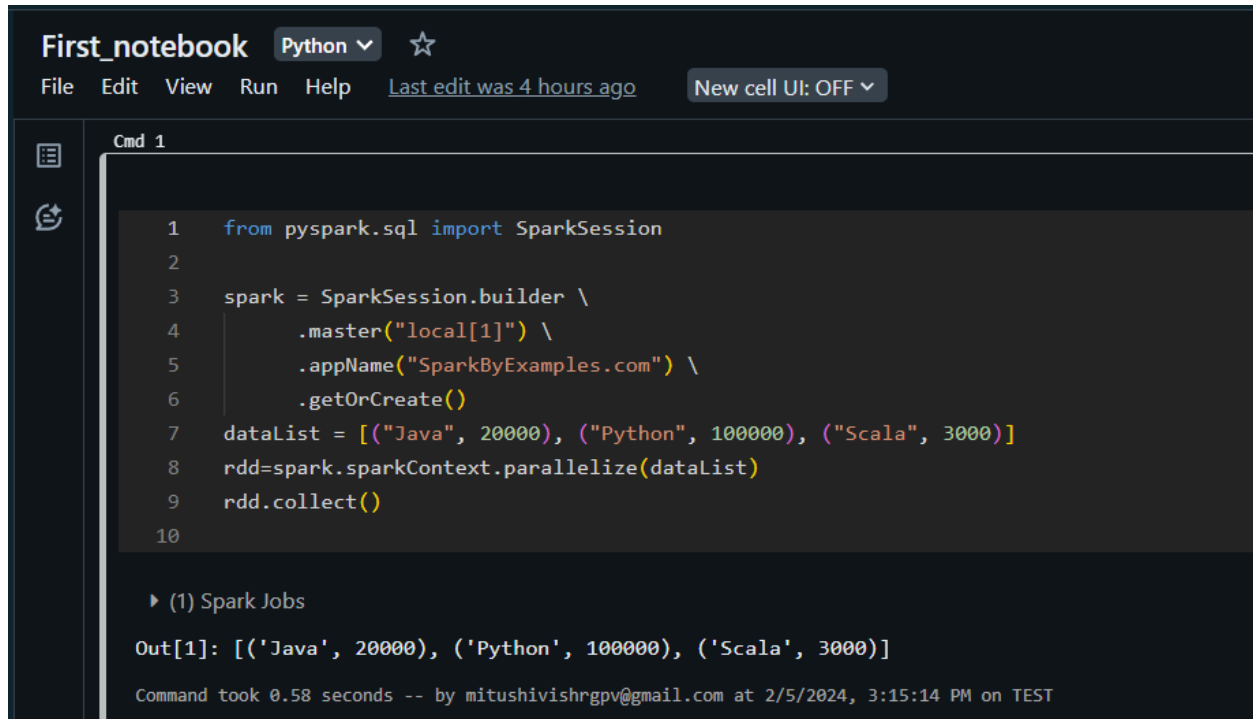
[22]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("sampleData").getOrCreate()
data = spark.read.csv("Employees.csv")
data.show()

+-----+-----+-----+
| _c0 | _c1 | _c2 |
+-----+-----+-----+
| EmpID | Name | Salary |
| e101 | Pramod | 1200000 |
| e120 | Dinesh | 2200000 |
| e205 | Sabesta | 1500000 |
| e331 | Harry | 1700000 |
| e421 | Avinash | 1300000 |
| e231 | Joy | 2300000 |
| e222 | Smith | 2100000 |
| e339 | Khan | 1800000 |
| e150 | Dilip | 1900000 |
| e131 | Kiran | 800000 |
+-----+-----+-----+

[23]: data

[23]: DataFrame[_c0: string, _c1: string, _c2: string]
```

Here we created spark session then using in code collection of objects and passing that collection into SparkContext.parallelize and using .collect() to display data.



The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook is titled "First_notebook" and has a "Python" language selector. The menu bar includes "File", "Edit", "View", "Run", "Help", "Last edit was 4 hours ago", and "New cell UI: OFF". The notebook contains a single code cell labeled "Cmd 1". The code in the cell creates a SparkSession, builds a SparkContext with a local master and an application name, and then creates an RDD from a list of tuples. The output of the cell shows the RDD contents as a list of tuples. The command took 0.58 seconds to execute.

```
1 from pyspark.sql import SparkSession
2
3 spark = SparkSession.builder \
4     .master("local[1]") \
5     .appName("SparkByExamples.com") \
6     .getOrCreate()
7 dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]
8 rdd=spark.sparkContext.parallelize(dataList)
9 rdd.collect()
10
```

► (1) Spark Jobs

Out[1]: [('Java', 20000), ('Python', 100000), ('Scala', 3000)]

Command took 0.58 seconds -- by mitushivishrgpv@gmail.com at 2/5/2024, 3:15:14 PM on TEST

Reading csv file using textFile() method :



The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook contains a single code cell labeled "Cmd 2". The code in the cell creates an RDD from a CSV file using the textFile() method and then collects the data. The output of the cell shows the RDD contents as a list of strings, each representing a row in the CSV file. The command took 1.68 seconds to execute.

```
1 rdd2 = spark.sparkContext.textFile("/FileStore/tables/employee/Employees.csv")
2 rdd2.collect()
```

► (1) Spark Jobs

Out[2]: ['EmpID,Name,Salary',
'e101,Pramod,1200000',
'e120,Dinesh,2200000',
'e205,Sabesta,1500000',
'e331,Harry,1700000',
'e421,Avinash,1300000',
'e231,Joy,2300000',
'e222,Smith,2100000',
'e339,Khan,1800000',
'e150,Dilip,1900000',
'e131,Kiran,800000']

Command took 1.68 seconds -- by mitushivishrgpv@gmail.com at 2/5/2024, 3:21:03 PM on TEST

Reading text file using textFile() method in pyspark.

```
1
2 rdd3 = spark.sparkContext.textFile("/FileStore/tables/employee/Testfile-1.txt")
3 rdd3.count()
```

► (2) Spark Jobs

Out[1]: 2

Command took 10.46 seconds -- by mitushivishrgpv@gmail.com at 2/5/2024, 9:58:22 PM on TEST

Cmd 4

```
1 rdd3.collect()
```

► (1) Spark Jobs

Out[2]: ['Hello Everyone! ', 'Wwlcome to pyspark programming']

Command took 0.89 seconds -- by mitushivishrgpv@gmail.com at 2/5/2024, 9:59:32 PM on TEST

Day - 2

Agenda:-

1. Intro. to PySpark and setting up env.
2. Hands-on → Basic PySpark
3. ————— → analyze sample dataset
4. RDDs and Transformations
5. Hands-on → Transforming data with PySpark RDDs.

PySpark → Apache Spark library is written in python.

to run Python applications using Apache Spark capabilities.

- Python API
- analytical processing engine for large scale data processing and ML applications.

Spark → data analytics application
can run on → single node machines → one worker
multi-node machines → multiple workers.

→ created to ~~reduce~~ address the limitation of mapReduce.

→ in-memory processing.

Pyspark :-

- efficient processing of large datasets
- can be used in Anaconda, Spyder, Jupyter.

Features -

- in-memory computation.
- parallel processing
- used with yarn, Mesos
- fault tolerant -
- immutable
- Lazy evaluation
- cache & persistence.
- inbuilt optimization

App. run on Pyspark is 100x faster than traditional system.

- Using pyspark, we can process data from Hadoop, HDFS, AWS S3.
- used to process real-time data using Kafka.

Apache Kafka.

- event streaming platform
- high-performance data pipelines.

PySpark supports :-

Python 3.8

Java 8, 11, 13, 17

Scala 2.12 and 2.13

R 3.5

Modules & Packages

PySpark, RDD

pyspark.sql (Dataframe and sql)

pyspark-streaming

pyspark.ml / mllib

pyspark.GraphFrames

pyspark.resource

PySpark commands

Read data of a file.

→ Create a txt file.

→ val filePath = "path"

→ val textRDD = sc.textFile(filePath)

→ textRDD.collect().foreach(println)

Create RDD

2 ways

→ loading an external dataset

→ distributing a set of collection of objects

2023
MONDAY
WK 52 • DAY 359-006
DECEMBER

25-12-2023

25

using parallelize function.

RDD is an immutable distributed collection of object sets.

→ data structure in ~~py~~spark

parallelize() takes collection of objects and pass it to Spark Context

Spark Context is the entry point to Spark cluster manager.

SC → Spark Context in spark-shell

•

pyspark.sql module → SparkSession method

getOrCreate() → initialize the session

• show() will display the data

• toDF() is used to create DataFrame with specified column.

spark.sparkContext.parallelize([column])

] .toDF()

Spark.stop → terminates the SparkSession.

Spark.createDataFrame (data[☞], schema=)

First step :-

1. import SparkSession
2. create SparkSession
3. add data
4. display
5. stop session.

Create RDD

1. First create SparkSession

→ using builder ()
or
→ newSession()

We can create multiple SparkSession
but one SparkContext.