

PySpark on DataBricks - Community Edition

First Draft of this knowledge sharing document

Table of contents

1. Create a spark cluster
2. Upload a large csv file into the spark cluster
3. Upload a large JSON file into the spark cluster
4. Creating SQL tables from these files uploaded earlier
5. Creating DELTA Lake .
6. Azure Data Bricks - DELTA Engine
- 7.
8. Hosting RStudio Server Pro - On DataBricks
9. Hosting R Server Pro (RStudio Workbench) - On Local Ubuntu and Connecting to DataBricks - Remote Connect and ODBC
10. Databricks AutoML
11. RStats → Final Model saved as .rds files . Sample R Code and process discussed
12. RStat → Save the coefficients from a Linear Model as .rds File

As an initial step we go to the Databricks Community Edition page the URL of which is given below

On the Community Edition page you click on the **get started** and you can create a Community Edition account using your personal email ID , you need not use your **Black Straw** official email ID.

<https://databricks.com/try-platform>

DATABRICKS PLATFORM - FREE TRIAL

CLOUD

For businesses

- Collaborative environment for Data teams to build solutions together
- Unlimited clusters that can scale to any size, processing data in your own account
- Job scheduler to execute jobs for production pipelines
- Fully collaborative notebooks with multi-language support, dashboards, REST APIs
- Native integration with the most popular ML frameworks (scikit-learn, TensorFlow, Keras,...), Apache Spark™, Delta Lake, and MLflow
- Advanced security, role-based access controls, and audit logs
- Single Sign On support
- Integration with BI tools such as Tableau, Qlik, and Looker
- 14-day full feature trial (excludes cloud charges)

For students and educational institutions

- Single cluster limited to 15GB and no worker nodes
- Basic notebooks without collaboration
- Limited to 3 max users
- Public environment to share your work

[GET STARTED](#)

By clicking "Get Started" for the Community Edition, you agree to the [Databricks Community Edition Terms of Service](#).

CHOOSE YOUR CLOUD



Please note that Azure



By clicking on the "AWS"



By clicking on the



<https://community.cloud.databricks.com/?o=6471035763102896>

Welcome to databricks

[Explore the Quickstart Tutorial](#)

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

[Import & Explore Data](#)

Quickly import data, preview its schema, create a table, and query it in a notebook.

[Create a Blank Notebook](#)

Create a notebook to start querying, visualizing, and modeling your data.

Common Tasks
Recents
What's new in v3.48

Recent files appear here as you work.

[Databricks Status](#)
[View latest release notes](#)

You will now see the welcome page , so the first thing we want to do is create a new cluster

Creating a new cluster

The screenshot shows the 'Create Cluster' page on the Databricks website. On the left is a sidebar with icons for Home, Clusters, Databricks Runtime, Jobs, Data, and Help. The main area has a title 'Create Cluster' and a sub-section 'New Cluster'. It includes a 'Cancel' button and a 'Create Cluster' button. At the top right are UI and JSON tabs. Below these are fields for 'Cluster Name' (with a placeholder 'Please enter a cluster name'), 'Databricks Runtime Version' (set to 'Runtime: 8.2 (Scala 2.12, Spark 3.1.1)'), and 'Instance' (with a note about free memory and termination after two hours). A note at the bottom says 'Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.' There are tabs for 'Instances' (selected) and 'Spark'.

First we name our cluster , we select a **databricks runtime** version

The runtime version has two parameters one the Scala version and the Spark version

the latest **Scala** version is 2.12

the latest **Spark** version is 3.11

You will see a note which states that databricks version 8.x that is 8.0 onwards supports **Delta Lakes** as the default table schema.

The bottom most drop down that you see on this page is the **availability zone**

it will make sense to choose something besides **auto** , if you want to choose a nearby Geographic zone for your AWS storage which will be linked with your Databricks Spark cluster DataBricks Runtime's Further reading -

(<https://docs.microsoft.com/en-us/azure/databricks/clusters/configure#--databricks-runtime>)

Please enter a cluster name

Databricks Runtime Version

Runtime: 8.2 (Scala 2.12, Spark 3.1.1)

Databricks runtime

Standard > 8.3 ML GPU, Scala 2.12, Spark 3.1.1
ML > 8.3 ML Scala 2.12, Spark 3.1.1
8.2 ML GPU, Scala 2.12, Spark 3.1.1
8.2 ML Scala 2.12, Spark 3.1.1
8.1 ML GPU, Scala 2.12, Spark 3.1.1
8.1 ML Scala 2.12, Spark 3.1.1
8.0 ML Scala 2.12, Spark 3.1.1
7.6 ML GPU, Scala 2.12, Spark 3.0.1
7.6 ML Scala 2.12, Spark 3.0.1
7.3 LTS ML GPU, Scala 2.12, Spark 3.0.1

For more configuration options, please upgrade your Databricks subscription.

Within the dropdown we can see many options for the ML cluster the machine learning cluster these are of two basic types one's with graphical processors or the GPU's And the others with cpus with no graphical processing units

for the Data Bricks Community Edition I think it's best to use one without a GPU otherwise we will get a warning for a Nvidia end user licence agreement (NVIDIA - EULA), we will look into this later for now I have chosen a non GPU runtime

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours.
For more configuration options, please upgrade your Databricks subscription.

Instances Spark

Spark Config

Enter your Spark configuration options here. Provide only one key-value pair per line. Example:
spark.speculation true spark.kryo.registrator my.package.MyRegistrar

Environment Variables

```
SPARK_NICENESS=0
JAVA_OPTS="-D... -D... -XX:MaxPermSize=256m ..."
JAVA_OPTS="$JAVA_OPTS -D..."
```

On this same page one tab has instances and the other tab is called Spark within the spark tab we have Spark config and **Environment variables for Spark**

if we were to compare this with a standalone spark installation on a physical server we would end up having these environment variables in a local config-file something like the **.bashrc** on a nix OS (Ubuntu)

the new cluster takes some time to get configured as it is getting configured you can utilise this time to view the various tabs on the configuration screen

The screenshot shows the 'Create Cluster' interface on the Databricks web portal. The URL is <https://community.cloud.databricks.com/?o=6471035763102896#create/cluster>. The page title is 'New Cluster'. Configuration settings include:

- Runtime:** 8.1 ML (GPU, Scala 2.12, Spark 3.1.1)
- Driver Type:** Community Optimized (15.3 GB Memory, 2 Cores)
- Environment Variables:** SPARK_NICENESS=0

A tooltip for the runtime dropdown states: "By using this version of Databricks Runtime, you agree to the terms and conditions outlined in [NVIDIA End User License Agreement \(EULA\)](#) with respect to CUDA, cuDNN and Tesla libraries, and [NVIDIA End User License Agreement \(EULA\)](#) with [NCCL Supplement](#) for the NCCL library."

The screenshot shows the 'Clusters' interface on the Databricks web portal. The URL is <https://community.cloud.databricks.com/?o=6471035763102896#setting/clusters/0624-062359-tinny727>. The cluster name is 'ml_83_sp_311'. Configuration settings include:

- Runtime:** 8.3 ML (includes Apache Spark 3.1.1, Scala 2.12)
- Driver Type:** Community Optimized (15.3 GB Memory, 2 Cores)
- Environment Variables:** SPARK_NICENESS=0

A tooltip for the runtime dropdown states: "By using this version of Databricks Runtime, you agree to the terms and conditions outlined in [NVIDIA End User License Agreement \(EULA\)](#) with respect to CUDA, cuDNN and Tesla libraries, and [NVIDIA End User License Agreement \(EULA\)](#) with [NCCL Supplement](#) for the NCCL library."

I have named the cluster **m83_sp_311** , which basically means **ml 8.3 and Spark 3.11**
JSON of the Cluster Configuration is as seen below ---

```
{  
  "num_workers": 0,  
  "cluster_name": "ml_83_sp_311",  
  "spark_version": "8.3.x-cpu-ml-scala2.12",  
  "spark_conf": {  
    "spark.databricks.delta.preview.enabled": "true"  
  },  
  "aws_attributes": {  
    "first_on_demand": 0,  
    "availability": "ON_DEMAND",  
    "zone_id": "us-west-2b",  
    "spot_bid_price_percent": 100,  
    "ebs_volume_count": 0  
  },  
  "node_type_id": "dev-tier-node",  
  "driver_node_type_id": "dev-tier-node",  
  "ssh_public_keys": [],  
  "custom_tags": {},  
  "spark_env_vars": {},  
  "autotermination_minutes": 120,  
  "enable_elastic_disk": false,  
  "cluster_source": "UI",  
  "init_scripts": [],  
  "cluster_id": "0624-062359-tinny727"  
}
```

Various tabs seen on this page are **Configuration , Notebooks , Libraries , Event log , Spark UI , Driver logs , Metrics**

The screenshot shows the Databricks Cluster UI for the cluster **ml_83_sp_311**. The top navigation bar includes links for Configuration, Notebooks, Libraries, Event Log, Spark UI, Driver Logs, Metrics, Apps, and Spark Cluster UI - Master. The **Spark UI** tab is currently selected. On the left, there is a sidebar with icons for Clusters, Databricks Runtime, Storage, Environment, Executors, SQL, JDBC/ODBC Server, and Structured Streaming. The main content area displays the cluster's configuration details, including its hostname (`ec2-54-188-20-235.us-west-2.compute.amazonaws.com`), Spark version (`8.3.x-cpu-ml-scala2.12`), and various metrics like Data Read from External Filesystem, Data Read from IO Cache, Data Written to IO Cache, Estimated Size of Repeatedly Read Data, and Cache Metadata Manager Peak Disk Usage.

The screenshot shows the Databricks Cluster UI for cluster **ml_83_sp_311**. The **Executors** tab is selected. At the top, it displays the hostname `ec2-54-188-20-235.us-west-2.compute.amazonaws.com` and Spark Version `8.3.x-cpu-ml-scala2.12`. Below this, there are tabs for Jobs, Stages, Storage, Environment, Executors (which is active), SQL, JDBC/ODBC Server, and Structured Streaming. The Executors section includes a summary table and a detailed Executors view. The summary table shows the following data:

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(1)	0	0.0 B / 3.9 GiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	0.0 B / 3.9 GiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0

The Executors view shows a grid of executor status icons. Below the table, there are filters for "Show 20 entries" and a search bar.

The Event log

As seen below the event log will track all the **instances / events** which happen with the current Cluster

The screenshot shows the Databricks Cluster UI for cluster **ml_83_sp_311**. The **Event Log** tab is selected. At the top, it displays the same cluster information as the previous screenshot. Below this, there is a filter bar labeled "Filter by Event Type..." and a refresh button. The main area is a table showing event logs:

Event Type	Time	Message
DRIVER_HEALTHY	2021-06-24 12:02:51 IST	Driver is healthy.
RUNNING	2021-06-24 11:59:46 IST	Cluster is running.
CREATING	2021-06-24 11:54:00 IST	Cluster creation requested by rohit.dhankar@strategic-leadership-llc-india.com.

The screenshot shows the Databricks Cluster UI for cluster 'ml_83_sp_311'. The top navigation bar includes links for Configuration, Notebooks, Libraries, Event Log, Spark UI (selected), Driver Logs, Metrics, Apps, and Spark Cluster UI - Master. Below the navigation is a header with Hostname: ec2-54-188-20-235.us-west-2.compute.amazonaws.com and Spark Version: 8.3.x-cpu-ml-scala2.12. A sub-header lists Jobs, Stages, Storage, Environment, Executors, SQL, JDBC/ODBC Server, and Structured Streaming. The main content area is titled 'Spark Jobs' with a question mark icon. It shows user: root, total uptime: 6.2 min, and scheduling mode: FAIR. A section titled 'Event Timeline' has an 'Enable zooming' checkbox. Below it, two tables show 'Executors' and 'Jobs' status over time (days 25, 26, 27, 28, 29). The Executors table shows no changes (Added: 0, Removed: 0). The Jobs table shows one job: Succeeded (1), Failed (0), and Running (0).

Time	Succeeded	Failed	Running
25	1	0	0
26	1	0	0
27	1	0	0
28	1	0	0
29	1	0	0

Spark UI

The Spark UI gives us more details than the **event log** it will show us who the user is , in this case it is **root** as of now .

Also it will show us which particular job started at what time and what is the status of the job .

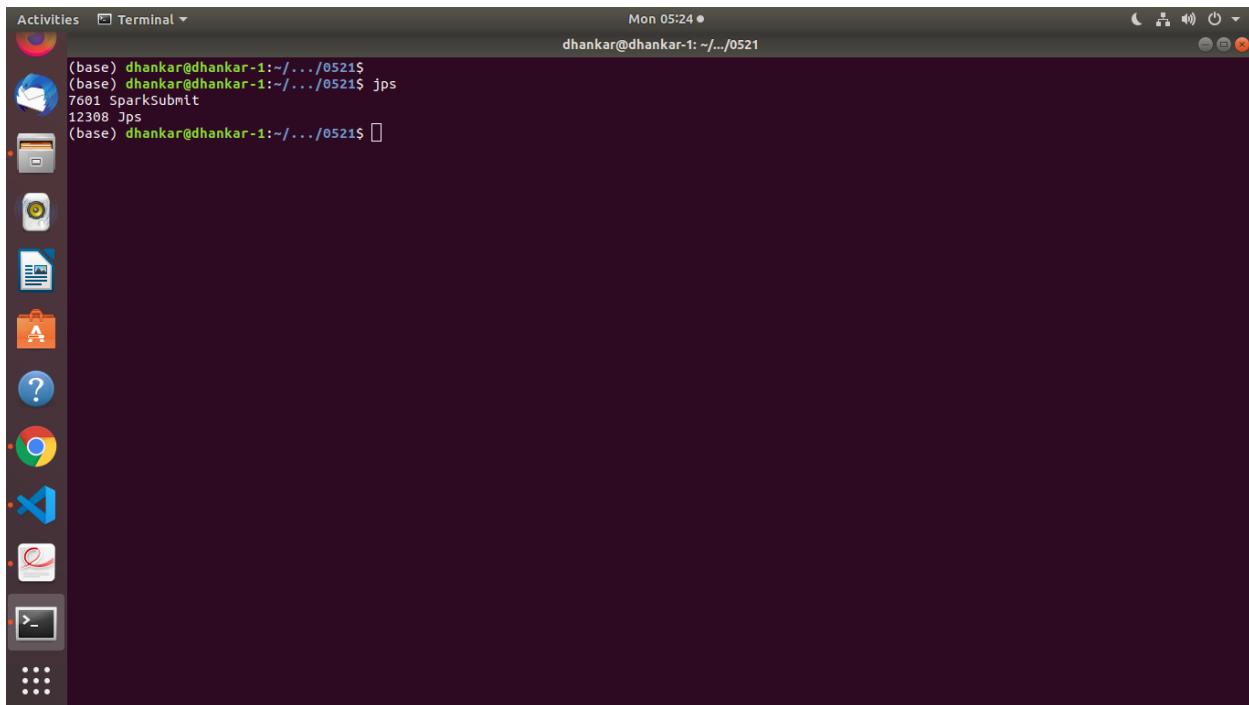
The **Scheduling Mode** is shown as **FAIR** , the default mode is usually **FIFO (First in First Out)** (<https://spark.apache.org/docs/latest/job-scheduling.html>)



```
# at this stage we should look at the DAG and details for various JOBS in
the UI
#http://localhost:4040/jobs/job/?id=1
#http://localhost:4040/jobs/job/?id=2
#http://localhost:4040/jobs/job/?id=3
```

Viewing the SPARK PROCESSES - JVM Process Status Tool (jps command)

The jps command outputs PID s and the names of JVM processes running on the machine



Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
30	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:28	23 ms	1/1			2.0 KiB	
29	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:25	3 s	6/6	69.8 MiB			2.0 KiB
28	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:25	23 ms	1/1			2.0 KiB	
27	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:21	3 s	6/6	69.8 MiB			2.0 KiB
26	head at <ipython-input-76-7c71b80c26cd>:23	+details 2021/07/05 11:35:21	4 ms	1/1	64.0 KiB			
25	count at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:21	3 ms	1/1			354.0 B	
24	count at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:21	0.1 s	6/6	69.8 MiB			354.0 B
23	csv at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:21	0.6 s	6/6	69.8 MiB			
22	csv at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:35:21	3 ms	1/1	64.0 KiB			
21	sum at <ipython-input-74-9c49a30d6b9a>:7	+details 2021/07/05 11:34:12	19 ms	6/6				
20	sum at <ipython-input-73-d759e121e92f>:7	+details 2021/07/05 11:33:52	36 ms	6/6				
19	sum at <ipython-input-72-e684e79cf2d6>:4	+details 2021/07/05 11:30:56	2 s	6/6				
18	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:22:13	24 ms	1/1			2.0 KiB	
17	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:22:09	3 s	6/6	69.8 MiB			2.0 KiB
16	describe at NativeMethodAccessImpl.java:0	+details 2021/07/05 11:22:09	26 ms	1/1			2.0 KiB	

Create a Table

Upload a CSV file to the DBFS (Data Bricks File System)

This file was by default uploaded to this path - dbfs:/FileStore/tables/walmart_stock.csv

Create New Table

Data source [?](#)

Upload File [S3](#) [DBFS](#) [Other Data Sources](#) [Partner Integrations](#)

DBFS Target Directory [?](#)
/FileStore/tables/ [\(optional\)](#) [Select](#)

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files [?](#)

walmart_stock.csv
90.3 KB [Remove file](#)

✓ File uploaded to /FileStore/tables/walmart_stock.csv

[Create Table with UI](#) [Create Table in Notebook](#) [?](#)

Two options to create a new table

- Create Table with UI
- **Create Table in Notebook**

We start with - Create Table in Notebook

As seen below - we get a new notebook with a Default Name and default code populated in the first few cells

```
1 # File location and type
2 file_location = "/FileStore/tables/walmart_stock.csv"
3 file_type = "csv"
4
5 # CSV options
6 infer_schema = "false"
7 first_row_is_header = "false"
8 delimiter = ","
9
10 # The applied options are for CSV files. For other file types, these will be ignored.
11 df = spark.read.format(file_type) \
12   .option("inferSchema", infer_schema) \
13   .option("header", first_row_is_header) \
14   .option("sep", delimiter) \
15   .load(file_location)
16
17 display(df)
```

```
1 ml_83_sp_311
2 delimiter = ","
3
4 # The applied options are for CSV files. For other file types, these will be ignored.
5 df = spark.read.format(file_type) \
6   .option("inferSchema", infer_schema) \
7   .option("header", first_row_is_header) \
8   .option("sep", delimiter) \
9   .load(file_location)
10
11 display(df)
```

(3) Spark Jobs

- Job 0 View (Stages: 1/1)
Stage 0: 1/1
- Job 1 View (Stages: 1/1)
- Job 2 View (Stages: 1/1)
Stage 2: 1/1

df: pyspark.sql.dataframe.DataFrame = [Date: string, Open: double ... 5 more fields]

	Date	Open	High	Low	Close	Volume	Adj Close
1	2012-01-03	59.970001	61.060001	59.869999	60.330002	12668800	52.61923499999996
2	2012-01-04	60.20999899999996	60.349998	59.470001	59.70999899999996	9593300	52.078475
3	2012-01-05	59.349998	59.619999	58.369999	59.419998	12768200	51.825539
4	2012-01-06	59.419998	59.450001	58.869999	59	8069400	51.45922
5	2012-01-09	59.029999	59.549999	58.919998	59.18	6679300	51.61621500000004
6	2012-01-10	59.43	59.70999899999996	58.98	59.04000100000004	6907300	51.494109
7	2012-01-11	59.000001	59.500001	58.000001	59.000001	6679300	51.494109

Looking at the **DTYPES** of the DataFrame
We have **string** , **double** , and **integer**

2021-06-24 - DBFS Example (Python)

ml_83_sp_311 | 17 | display(df)

(3) Spark Jobs

- Job 0 View (Stages: 1/1)
Stage 0: 1/1
- Job 1 View (Stages: 1/1)
- Job 2 View (Stages: 1/1)
Stage 2: 1/1

df: pyspark.sql.dataframe.DataFrame

	Date	Open	High	Low	Close	Volume	Adj Close
1	2012-01-03	59.970001	61.060001	59.869999	60.330002	12668800	52.61923499999996
2	2012-01-04	60.20999899999996	60.349998	59.470001	59.70999899999996	9593300	52.078475
3	2012-01-05	59.349998	59.619999	58.369999	59.419998	12768200	51.825539
4	2012-01-06	59.419998	59.450001	58.869999	59	8069400	51.45922
5	2012-01-09	59.029999	59.549999	58.919998	59.18	6679300	51.61621500000004
6	2012-01-10	59.43	59.70999899999996	58.98	59.04000100000004	6907300	51.494109
7	2012-01-11	59.000001	59.500001	59.01000100000001	59.100001	6000000	51.494109

Truncated results, showing first 1000 rows.

```
# Create a view or table
temp_table_name = "listingsAndReviews_json"
df.createOrReplaceTempView(temp_table_name)
```

```
1 # With this registered as a temp view, it will only be available to this particular notebook. If you'd like other users to be able to query this
2 # table, you can also create a table from the DataFrame.
3 # Once saved, this table will persist across cluster restarts as well as allow various users across different notebooks to query this data.
4 #
5 permanent_table_name = "walmart_stock"
6
7 df.write.format("parquet").saveAsTable(permanent_table_name)

AnalysisException: Attribute name "Adj Close" contains invalid character(s) among " ;{}()\n\t=". Please use alias to rename it.

-----  

AnalysisException Traceback (most recent call last)
<command-1291429090299065> in <module>
      5 permanent_table_name = "walmart_stock"
      6
----> 7 df.write.format("parquet").saveAsTable(permanent_table_name)

/databricks/spark/python/pyspark/sql/readwriter.py in saveAsTable(self, name, format, mode, partitionBy, **options)
  1183         if format is not None:
  1184             self.format(format)
-> 1185         self._jwrite.saveAsTable(name)
  1186
  1187     def json(self, path, mode=None, compression=None, dateFormat=None, timestampFormat=None,
/databricks/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in __call__(self, *args)
  1302
  1303     answer = self.gateway_client.send_command(command)
https://community.cloud.databricks.com/?o=6471035763102896#lue
```

Seen above - we get an **ERROR**, the Column Name / Col Attribute - "Adj Close" has a space between two strings .

```
df1 = df.withColumnRenamed( "Adj Close" , "Adj_Close")
This creates a new - df1:pyspark.sql.dataframe.DataFrame
Date:string
Open:double
High:double
Low:double
Close:double
Volume:integer
Adj_Close:double
```

The screenshot shows the Databricks Table View interface. At the top, the URL is https://community.cloud.databricks.com/?o=6471035763102896#table/default/walmart_stock. The left sidebar has icons for File, Notebook, Cluster, Job, Pipeline, ML, and Databricks. The main area shows the table name 'default.walmart_stock' and a 'Refresh' button. A dropdown menu shows 'ml_83_sp_311'. Below is the 'Schema:' section with a table:

	col_name	data_type	comment
1	Date	string	null
2	Open	double	null
3	High	double	null
4	Low	double	null
5	Close	double	null
6	Volume	int	null
7	Adj Close	double	null

Showing all 7 rows.

Below the schema is the 'Sample Data:' section with a table:

	Date	Open	High	Low	Close	Volume	Adj Close
1	2012-01-03	59.970001	61.060001	59.869999	60.330002	12668800	52.61923499999996
2	2012-01-04	60.20999899999996	60.349998	59.470001	59.70999899999996	9593300	52.078475
3	2012-01-05	59.349998	59.619999	58.369999	59.419998	12768200	51.825539
4	2012-01-06	59.419998	59.450001	58.869999	59	8069400	51.45922
5	2012-01-09	59.029999	59.549999	58.919998	59.18	6679300	51.61621500000004
6	2012-01-10	59.12	59.70000000000006	59.00	59.04000100000004	8007200	51.404100

Code used to correct this error and create new Column names - is as seen in the attached Ipython Notebooks

As a Community Edition user, your cluster will automatically terminate after **an idle period of two hours**. When not used the cluster will get **Terminated**, you can **START** it again by clicking on the **PLAY ICON**

The screenshot shows the Databricks Compute settings page at https://community.cloud.databricks.com/?o=6471035763102896#setting/clusters. The left sidebar has icons for File, Notebook, Cluster, Job, Pipeline, ML, and Databricks. The main area shows 'Compute' and 'Job Clusters'. A table lists clusters:

	State	Nodes	Runtime	Driver	Worker	Creator	Actions
1	Terminated	-	8.3 ML (includes Apache Spark 3.1...)	Comm...	Comm...	rohit.dha...	0 Libraries / Spark UI / Logs ▶ Start

Page navigation: 1 - 1 of 1 | < > | 20 | Start

```

2021-06-24 - DBFS Example (1) (Python)
ml_83_sp_311
%fs ls /FileStore/tables/

```

path	name	size
1 dbfs:/FileStore/tables/listingsAndReviews.json	listingsAndReviews.json	111371457
2 dbfs:/FileStore/tables/walmart_stock.csv	walmart_stock.csv	90266

Showing all 2 rows.

Command took 13.31 seconds -- by rohit.dhankar@strategic-leadership-llc-india.com at 24/6/2021, 2:38:28 pm on ml_83_sp_311

Cmd 4

```

1 file_location = "/FileStore/tables/listingsAndReviews.json"
2 imdb_json = spark.read.json(file_location)
3 display(imdb_json)

```

▶ (1) Spark Jobs

AnalysisException: Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the referenced columns only include the internal corrupt record column (named `_corrupt_record` by default). For example:
`spark.read.schema(schema).json(file).filter("_corrupt_record".isNotNull).count()`
and `spark.read.schema(schema).json(file).select("_corrupt_record").show()`. Instead, you can cache or save the parsed results and then send the same query. For example, `val df = spark.read.schema(schema).json(file).cache()` and then `df.filter("_corrupt_record".isNotNull).count()`.

Command took 12.28 seconds -- by rohit.dhankar@strategic-leadership-llc-india.com at 24/6/2021, 2:41:52 pm on ml_83_sp_311

Cmd 5

READING A MULTI LINE JSON FILE

To read a Multi Line JSON file - we need to specify Multi Line as seen in code and snapshot below .

Seen below - the NESTED JSON or NESTED DICT Structure is called a - STRUCT , the other DTYPES are seen as STRING and DOUBLE .

```

2021-06-24 - DBFS Example (1) (Python)
ml_83_sp_311

```

```

1 #file_location = "/FileStore/tables/listingsAndReviews.json"
2 # File location and type
3 file_location = "/FileStore/tables/weather_data.json"
4 #imdb_json = spark.read.json(file_location)
5 df_json = spark.read.option("multiline","true").json(file_location)
6 #display(imdb_json)

```

▶ (1) Spark Jobs

df_json: pyspark.sql.dataframe.DataFrame

- _id: struct
 - \$oid: string
- airTemperature: struct
 - quality: string
 - value: double
- atmosphericPressureChange: struct
 - quantity24Hours: struct
 - quality: string
 - value: double
 - quantity3Hours: struct
 - quality: string
 - value: double
 - tendency: struct
 - code: string
 - quality: string
- atmosphericPressureObservation: struct
 - alimeterSetting: struct
 - quality: string
 - value: double

LISTING FILES

We can list all the Files with the below mentioned list command

```
# List all the Files in the -- file_location = "/FileStore/tables/
```

```
# use the Magic Command - %fs == File System
```

```
%fs ls /FileStore/tables/
```

This will show in a tabular format the path the name and the size of the files stored in the dbfs

The screenshot shows a Databricks notebook interface. At the top, there's a browser header with the URL <https://community.cloud.databricks.com/>. Below it, the notebook title is "2021-06-24 - DBFS Example (1) (Python)". On the left, there's a sidebar with various icons for file operations like copy, move, and delete. The main area has two sections: a table view and a code editor.

Table View:

	path	name	size
1	dbfs:/FileStore/tables/listingsAndReviews.json	listingsAndReviews.json	111371457
2	dbfs:/FileStore/tables/walmart_stock.csv	walmart_stock.csv	90266
3	dbfs:/FileStore/tables/weather_data.json	weather_data.json	23801818

Below the table, a message says "Showing all 3 rows."

Code Editor (Cmd 4):

```
1 file_location = "/FileStore/tables/listingsAndReviews.json"
2 imbd_json = spark.read.option("multiline","true").json(file_location)
3
4 file_location_1 = "/FileStore/tables/weather_data.json"
5 weather_json = spark.read.option("multiline","true").json(file_location_1)
6
7 display(imbd_json)
8 display(weather_json)
```

Below the code, there are two expanded sections:

- (4) Spark Jobs
 - imbd_json: pyspark.sql.dataframe.DataFrame = [__id: string, access: string ... 39 more fields]
 - weather_json: pyspark.sql.dataframe.DataFrame = [__id: struct, airTemperature: struct ... 24 more fields]

At the bottom, there's a preview of a DataFrame with columns: _id, access, accommodates, address. One row is shown with _id 10006546 and address "We are always available to help guests. The house is fully available to guests. We are always ready to assist guests. when 8".

PRINT SCHEMA of a JSON FILE

```
weather_json.printSchema()
```

(see the SCHEMA in the attached - .ipynb file)

```
root
| -- value: string (nullable = true)
```

```
weather_json.printSchema()
```

```
root
| -- _id: struct (nullable = true)
| | -- $oid: string (nullable = true)
| -- airTemperature: struct (nullable = true)
| | -- quality: string (nullable = true)
| | -- value: double (nullable = true)
| -- atmosphericPressureChange: struct (nullable = true)
| | -- quantity24Hours: struct (nullable = true)
| | | -- quality: string (nullable = true)
| | | -- value: double (nullable = true)
| | -- quantity3Hours: struct (nullable = true)
| | | -- quality: string (nullable = true)
| | | -- value: double (nullable = true)
| | -- tendency: struct (nullable = true)
| | | -- code: string (nullable = true)
| | | -- quality: string (nullable = true)
| -- atmosphericPressureObservation: struct (nullable = true)
| | -- altimeterSetting: struct (nullable = true)
| | | -- quality: string (nullable = true)
| | | -- value: double (nullable = true)
| | -- stationPressure: struct (nullable = true)
```

```
#RStudio Server Installation
```

DB UTILITIES - dbutils

Usage of DataBricks Utilities - from within the iPython Notebooks

File system utility (dbutils.fs) -

<https://docs.microsoft.com/en-us/azure/databricks/dev-tools/databricks-utils#--file-system-utility-dbutilsfs>

An example of the - **dbutils.fs.put** - can be seen below , we create and PUT a small csv file at the **temp** location

The screenshot shows a Databricks notebook interface with the following command history:

- Cmd 8:** dbutils.fs.put('dbfs:/tmp/sample_csv.csv', """text
some text sample text values rohit dhankar
""")
Wrote 49 bytes.
Out[16]: True
- Cmd 9:** %fs ls /tmp/
The output shows a table with two rows:

	path	name	size
1	dbfs:/tmp/file1.csv	file1.csv	298
2	dbfs:/tmp/sample_csv.csv	sample_csv.csv	49

Showing all 2 rows.
- Cmd 10:** # Create a view or table

SPARK CLI -- PERSONAL ACCESS TOKEN

<https://docs.databricks.com/dev-tools/api/latest/authentication.html>

Its not possible to get a PERSONAL ACCESS TOKEN from a Community Account . Thus had to create a new DataBricks Account . Had to register with a credit card and make a token payment .

My new - workspace URL is --

<https://adb-8792613572312780.0.azuredatabricks.net>

<https://adb-8792613572312780.0.azuredatabricks.net/?o=8792613572312780#setting/account>

Configure the Personal Access Token --

databricks configure --token

(dbfs_env) dhankar@dhankar-1:~/.../cli_code\$ [databricks workspace ls](#)

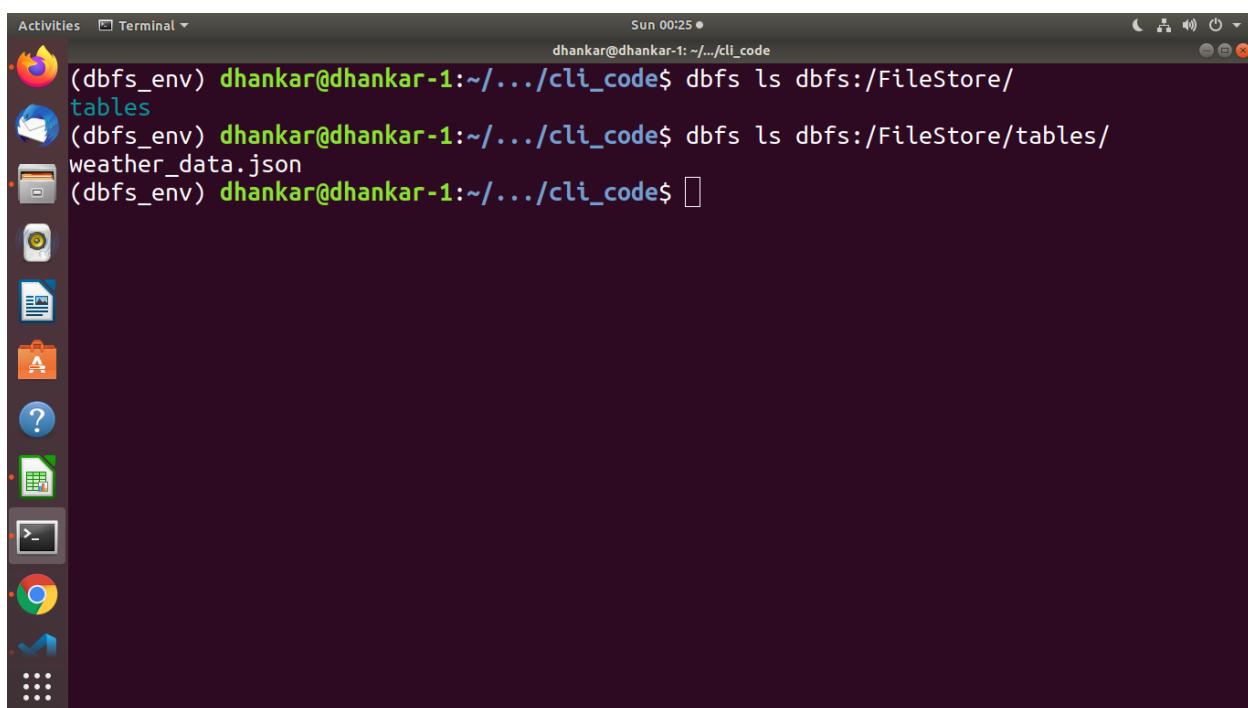
Users

Shared

Repos

(dbfs_env) dhankar@dhankar-1:~/.../cli_code\$

#



The screenshot shows a Linux desktop environment with a terminal window open in the foreground. The terminal window has a dark background and displays the command history and output of the 'databricks workspace ls' command. In the background, there is a file manager window showing a file tree with a 'FileStore' directory and its sub-directories 'tables' and 'weather_data.json'. The desktop interface includes a dock on the left with various icons and a system tray at the top.

```
dbfs_ls_dbfs:/FileStore/
tables
weather_data.json
```

```
%sh mkdir -p /dbfs/rohit_test_dir
%sh ls -ltr
%sh ls -ltr /dbfs/databricks-datasets
total 2935
-rwxrwxrwx 1 root root      0 Sep 25  2015 airlines_$folder$
-rwxrwxrwx 1 root root      0 Nov 18  2015 sms_spam_collection_$folder$
-rwxrwxrwx 1 root root    3359 Feb   9  2016 SPARK_README.md
-rwxrwxrwx 1 root root      0 Feb 11  2016 cs100_$folder$
-rwxrwxrwx 1 root root     976 Jul 24  2018 README.md
drwxrwxrwx 2 root root 2803754 Nov   6  2018 overlap-join
drwxrwxrwx 2 root root    4096 Jun 26 18:29 wine-quality
drwxrwxrwx 2 root root    4096 Jun 26 18:29 wikipedia-datasets
```

```
drwxrwxrwx 2 root root 4096 Jun 26 18:29 wiki
drwxrwxrwx 2 root root 4096 Jun 26 18:29 weather
drwxrwxrwx 2 root root 4096 Jun 26 18:29 tpch
drwxrwxrwx 2 root root 4096 Jun 26 18:29 timeseries
drwxrwxrwx 2 root root 4096 Jun 26 18:29 structured-streaming
drwxrwxrwx 2 root root 4096 Jun 26 18:29 songs
drwxrwxrwx 2 root root 4096 Jun 26 18:29 sms_spam_collection
drwxrwxrwx 2 root root 4096 Jun 26 18:29 sfo_customer_survey
drwxrwxrwx 2 root root 4096 Jun 26 18:29 samples
drwxrwxrwx 2 root root 4096 Jun 26 18:29 sample_logs
drwxrwxrwx 2 root root 4096 Jun 26 18:29 sai-summit-2019-sf
drwxrwxrwx 2 root root 4096 Jun 26 18:29 rwe
#
%sh ls -ltr /dbfs/databricks
total 8
drwxrwxrwx 2 root root 4096 Jun 26 18:29 mlflow-tracking
drwxrwxrwx 2 root root 4096 Jun 26 18:29 mlflow-registry
```

FLIGHTS DATA SET - EDA and FLIGHT CLASSIFIER

Microsoft Azure | Databricks

2021-06-26 - DBFS Example (1) (Python)

Free trial ends in 14 days. Upgrade to Premium in Azure Portal

Portal infobot2016@gmail.com

pyspark

```
1 # COMMAND -----
2
3 splits = df_flights.randomSplit([0.7, 0.3])
4 train = splits[0]
5 test = splits[1].withColumnRenamed("label", "trueLabel")
6 train_rows = train.count()
7 test_rows = test.count()
8 print("Training Rows:", train_rows, " Testing Rows:", test_rows)
9 #Training Rows: 1904896 Testing Rows: 814522
```

(4) Spark Jobs

train: pyspark.sql.dataframe.DataFrame

- DayofMonth: string
- DayOfWeek: string
- Carrier: string
- OriginAirportID: string
- DestAirportID: string
- DepDelay: string
- ArrDelay: string

test: pyspark.sql.dataframe.DataFrame = [DayofMonth: string, DayOfWeek: string ... 5 more fields]

Training Rows: 1904896 Testing Rows: 814522

Command took 9.18 seconds -- by infobot2016@gmail.com at 27/6/2021, 2:31:23 am on pyspark_8_311

Cmd 5

```
1 # Create a view or table
```

Clusters /

ml_83_sp_311

Edit Clone Restart Terminate Delete

Configuration Notebooks Libraries Event Log Spark UI Driver Logs Metrics Apps Spark Cluster UI - Master

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours.
For more configuration options, please upgrade your Databricks subscription.

Instances Spark JDBC/ODBC Permissions

Server Hostname: community.cloud.databricks.com

Port: 443

Protocol: HTTPS

HTTP Path: sql/protocolv1/o/6471035763102896/0624-062359-tinny727

JDBC URL: jdbc:spark://community.cloud.databricks.com:443/default;transportMode=http;ssl=1;httpPath=/sql/protocolv1/o/6471035763102896/0624-062359-tinny727;AuthMech=3;UID=token;PWD=<personal-access-token>

JDBC / ODBC URL .

Required to connect BI tools to PySpark clusters on the Azure / DataBricks - we connect business intelligence (BI) tools to Azure Databricks clusters and SQL endpoints to query data in tables.

More info see here - (<https://docs.microsoft.com/en-us/azure/databricks/integrations/bi/jdbc-odbc-bi>)

Example JDBC URL ==

```
jdbc:spark://adb-2951675697717654.14.azuredatabricks.net:443/default;transportMode=http;ssl=1;httpPath$sql/protocolv1/o/2951675697717654/0629-072014-finch221;AuthMech=3;UID=tok  
en;PWD=<personal-access-token>
```

```
jdbc:spark://adb-2951675697717654.14.azuredatabricks.net:443/default;  
transportMode=http;  
ssl=1;  
httpPath$sql/protocolv1/o/2951675697717654/0629-072014-finch221;  
AuthMech=3;  
UID=tok  
en;  
PWD=<personal-access-token>
```

STORAGE - PARQUET FILES -

Seen below the tab -- **Spark UI >> Storage >> Parquet IO Cache** , this is the status before we have loaded any data , various metrics are shown - **Data read from external file system** , **Data read from IO cache** , **Estimated size of repeatedly read data** , **Cache Metadata manager peak disk usage**.

The screenshot shows the Databricks Cluster UI for cluster 'ml_83_sp_311'. The 'Storage' tab is selected. The top navigation bar includes 'Edit', 'Clone', 'Restart', 'Terminate', and 'Delete' buttons. Below the tabs, it displays 'Hostname: ec2-54-188-20-235.us-west-2.compute.amazonaws.com' and 'Spark Version:8.3.x-cpu-ml-scala2.12'. Under the 'Storage' section, there's a 'Parquet IO Cache' table:

Data Read from External Filesystem	Data Read from IO Cache	Data Written to IO Cache	Estimated Size of Repeatedly Read Data	Cache Metadata Manager Peak Disk Usage
0.0 B	0.0 B	0.0 B	0.0 B (0%) - 0.0 B (0%)	0.0 B

Seen below the tab -- **Spark UI >> Environment >> Runtime Information**
Spark UI >> Environment >> Spark Properties

Seen below we have the tab -- **Spark UI >> Executors**

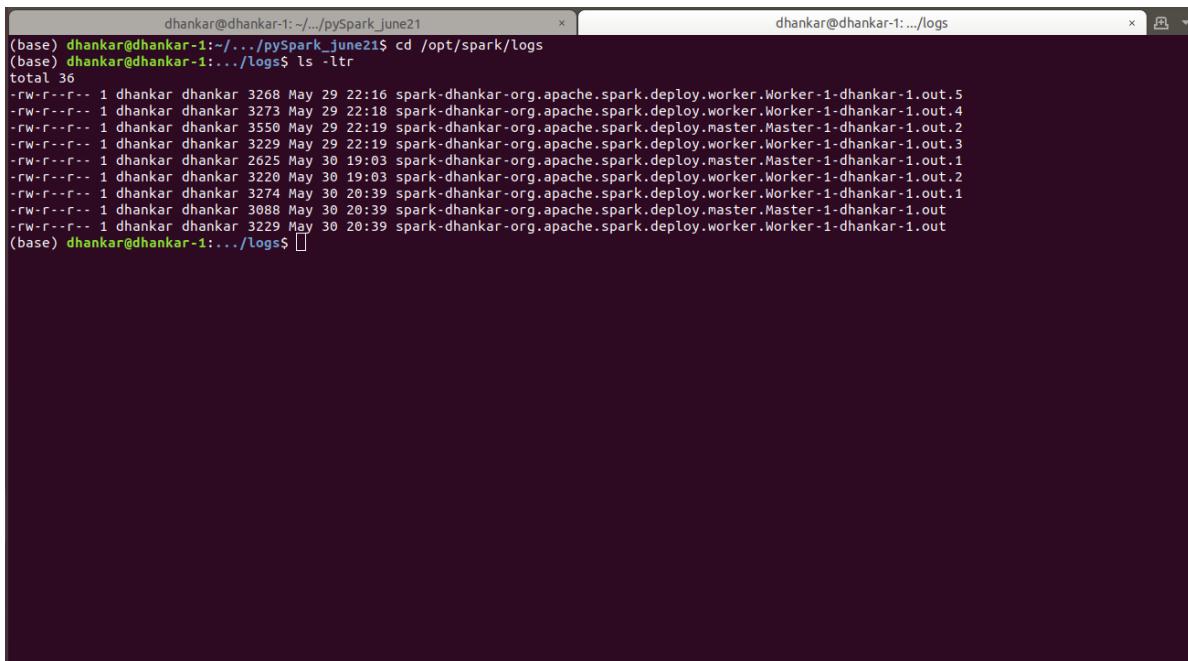
The screenshot shows the Databricks Cluster UI for cluster 'ml_83_sp_311'. The 'Executors' tab is selected. The top navigation bar includes 'Edit', 'Clone', 'Restart', 'Terminate', and 'Delete' buttons. Below the tabs, it displays 'Hostname: ec2-54-188-20-235.us-west-2.compute.amazonaws.com' and 'Spark Version:8.3.x-cpu-ml-scala2.12'. Under the 'Executors' section, there's a 'Summary' table:

RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(1)	0.0 B / 3.9 GiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0.0 B / 3.9 GiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0

Below the summary, there's a table for 'Executors' showing 20 entries. The columns include Task Time, which is currently empty.

PySpark - On Local Standalone Cluster

Installed the latest version of PySpark on a local system running Ubuntu
Checking Logs being created in the DIR = **/opt/spark/logs**



The screenshot shows two terminal windows side-by-side. The left window has the title 'dhankar@dhankar-1:~/.../pySpark_june21\$' and contains a command to change directory to '/opt/spark/logs'. The right window has the title 'dhankar@dhankar-1:~/logs\$' and contains a command to list files in that directory with '-ltr'. Both windows show a list of log files with timestamps ranging from May 29 to May 30, 2020.

```
(base) dhankar@dhankar-1:~/.../pySpark_june21$ cd /opt/spark/logs
(base) dhankar@dhankar-1:~/logs$ ls -ltr
total 36
-rw-r--r-- 1 dhankar dhankar 3268 May 29 22:16 spark-dhankar-org.apache.spark.deploy.worker.Worker-1-dhankar-1.out.5
-rw-r--r-- 1 dhankar dhankar 3273 May 29 22:18 spark-dhankar-org.apache.spark.deploy.worker.Worker-1-dhankar-1.out.4
-rw-r--r-- 1 dhankar dhankar 3550 May 29 22:19 spark-dhankar-org.apache.spark.deploy.master.Master-1-dhankar-1.out.2
-rw-r--r-- 1 dhankar dhankar 3229 May 29 22:19 spark-dhankar-org.apache.spark.deploy.worker.Worker-1-dhankar-1.out.3
-rw-r--r-- 1 dhankar dhankar 2625 May 30 19:03 spark-dhankar-org.apache.spark.deploy.master.Master-1-dhankar-1.out.1
-rw-r--r-- 1 dhankar dhankar 3220 May 30 19:03 spark-dhankar-org.apache.spark.deploy.worker.Worker-1-dhankar-1.out.2
-rw-r--r-- 1 dhankar dhankar 3274 May 30 20:39 spark-dhankar-org.apache.spark.deploy.worker.Worker-1-dhankar-1.out.1
-rw-r--r-- 1 dhankar dhankar 3088 May 30 20:39 spark-dhankar-org.apache.spark.deploy.master.Master-1-dhankar-1.out
-rw-r--r-- 1 dhankar dhankar 3229 May 30 20:39 spark-dhankar-org.apache.spark.deploy.worker.Worker-1-dhankar-1.out
(base) dhankar@dhankar-1:~/logs$
```

Getting some warnings after starting PySpark - this not being looked into now .

```
$ pyspark
Python 3.8.2 (default, Mar 26 2020, 15:53:00)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
21/06/27 20:35:28 WARN Utils: Your hostname, dhankar-1 resolves to a
loopback address: 127.0.1.1; using 192.168.1.2 instead (on interface
enp2s0)

21/06/27 20:35:28 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to
another address

WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform
(file:/opt/spark/jars/spark-unsafe_2.12-3.1.1.jar) to constructor
java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of
org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal
reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/06/27 20:35:28 WARN NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```

Using Spark's default log4j profile:
org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).

21/06/27 20:35:30 WARN Utils: Service 'SparkUI' could not bind on port
4040. Attempting port 4041.

Welcome to

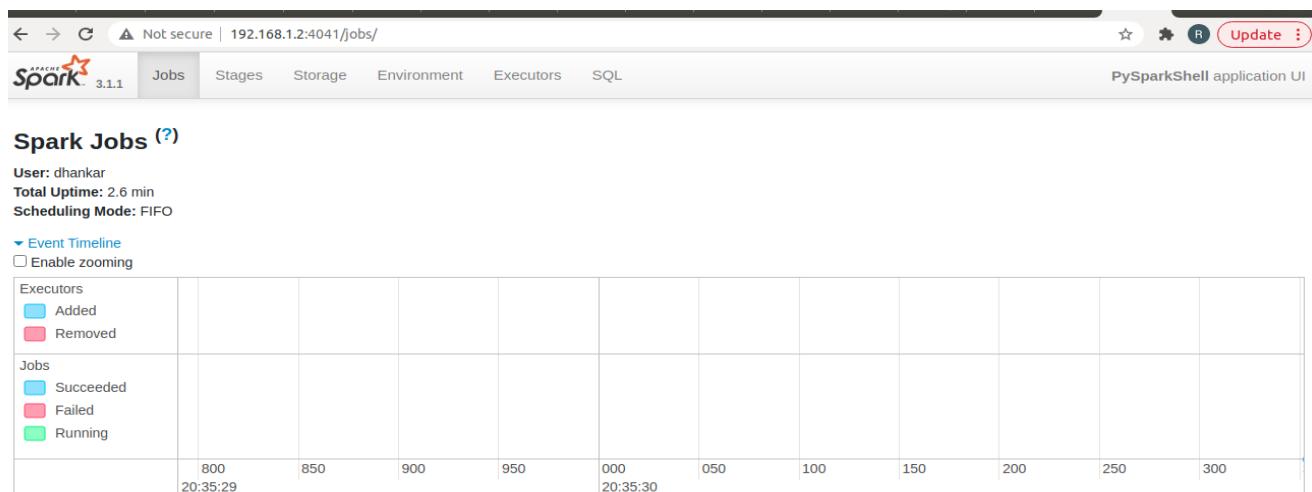
    _/_____
   /  \ /  \
  _\  \|  _ \  \
 /_  / . _/ \_ , _/ /  /_/\_ \
 /_/
   version 3.1.1

Using Python version 3.8.2 (default, Mar 26 2020 15:53:00)
Spark context Web UI available at http://192.168.1.2:4041
Spark context available as 'sc' (master = local[*], app id =
local-1624806330290).
SparkSession available as 'spark'.
>>>

```

Looking at the Web UI on Start-Up ..

Spark context Web UI available at <http://192.168.1.2:4041>



Not secure | 192.168.1.2:4041/executors/ Update

Apache Spark 3.1.1 Jobs Stages Storage Environment Executors SQL PySparkShell application UI

Executors

Show Additional Metrics

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(1)	0	0.0 B / 434.4 MiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	0.0 B / 434.4 MiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0

Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	192.168.1.2:37599	Active	0	0.0 B / 434.4 MiB	0.0 B	8	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	Thread Dump

Showing 1 to 1 of 1 entries Previous **1** Next

Started a local Jupyter Notebook and imported `findspark`, loaded the **WalmartStock.csv**

localhost:8888/notebooks/GitUp_PySpark_June21/GitUp/pySpark_jun21/pySpark_walmartData.ipynb 90% Logout

jupyter pySpark_walmartData Last Checkpoint: 05/30/2021 (unsaved changes) Python 3

In [6]: `import findspark
findspark.init()

import pyspark
import random`

In [2]: `from pyspark.sql import SparkSession`

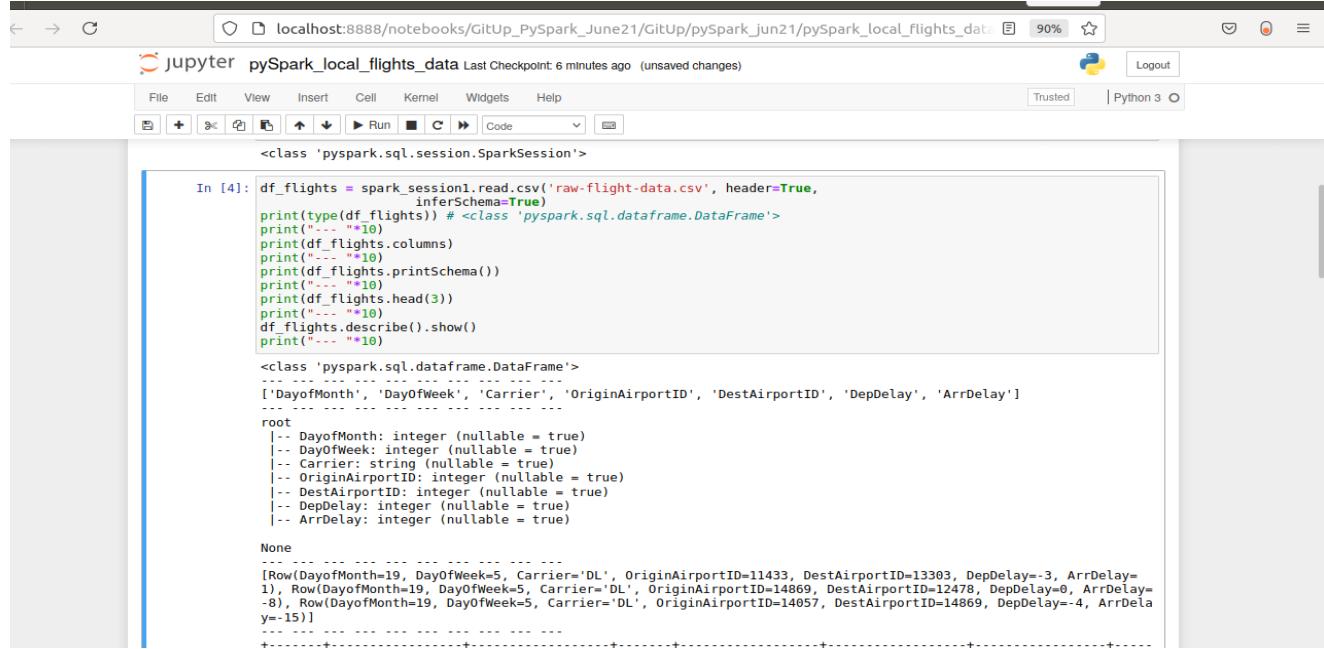
In [3]: `spark_session1 = SparkSession.builder.appName('solution').getOrCreate()
print(type(spark_session1)) #<class 'pyspark.sql.session.SparkSession'>
<class 'pyspark.sql.session.SparkSession'>`

In [4]: `df_walmart = spark_session1.read.csv('walmart_stock.csv', header=True,
 inferSchema=True)
print(type(df_walmart)) # <class 'pyspark.sql.dataframe.DataFrame'>
print("... *10")
print(df_walmart.columns)
print("... *10")
print(df_walmart.printSchema())
print("... *10")
print(df_walmart.head(3))
print("... *10")
df_walmart.describe().show()
print("... *10")

<class 'pyspark.sql.dataframe.DataFrame'>
...
['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
...
root
|-- Date: string (nullable = true)
| |-- Open: double (nullable = true)`

FLIGHTS DATA SET - EDA (local machine Standalone Spark Cluster)

Started another local Jupyter Notebook loaded the **raw-flight-data.csv**



The screenshot shows a Jupyter Notebook window titled "jupyter pySpark_local_flights_data". The code cell In [4] contains Python code to read a CSV file and print its schema and first few rows. The output shows the DataFrame schema and a sample of five rows from the flight data.

```
In [4]: df_flights = spark_session1.read.csv('raw-flight-data.csv', header=True, inferSchema=True)
print(type(df_flights)) # <class 'pyspark.sql.dataframe.DataFrame'
print("---- *10")
print(df_flights.columns)
print("---- *10")
print(df_flights.printSchema())
print("---- *10")
print(df_flights.head(3))
print("---- *10")
df_flights.describe().show()
print("---- *10")
```

```
<class 'pyspark.sql.session.SparkSession'>
<class 'pyspark.sql.session.SparkSession'>
[DayofMonth', 'DayofWeek', 'Carrier', 'OriginAirportID', 'DestAirportID', 'DepDelay', 'ArrDelay']
root
-- DayofMonth: integer (nullable = true)
-- DayofWeek: integer (nullable = true)
-- Carrier: string (nullable = true)
-- OriginAirportID: integer (nullable = true)
-- DestAirportID: integer (nullable = true)
-- DepDelay: integer (nullable = true)
-- ArrDelay: integer (nullable = true)

None
[Row(DayofMonth=19, DayofWeek=5, Carrier='DL', OriginAirportID=11433, DestAirportID=13303, DepDelay=-3, ArrDelay=-1), Row(DayofMonth=19, DayofWeek=5, Carrier='DL', OriginAirportID=14869, DestAirportID=12478, DepDelay=0, ArrDelay=-8), Row(DayofMonth=19, DayofWeek=5, Carrier='DL', OriginAirportID=14057, DestAirportID=14869, DepDelay=-4, ArrDelay=-15)]
```

```
# Display One Col -- Carrier
df_flights.select("Carrier").show(5)
```

```
#
```

Changing the iPython Notebook Kernel to point to the correct Virtual Environment -

```
(dbfs_env) dhankar@dhankar-1:~/.../pySpark_june21$ python -m ipykernel install
    --user --name dbfs_env --display-name "pySpark_dbfs_env"
Installed kernelspec dbfs_env in
    /home/dhankar/.local/share/jupyter/kernels/dbfs_env
```

https://ipython.readthedocs.io/en/stable/install/kernel_install.html#kernels-for-different-environments

jupyter pySpark_local_flights_data Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

In [1]:

```
import findspark
findspark.init()

import pyspark
import random
```

ModuleNotFoundError: No module named 'findspark'

In [2]:

In [3]:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *

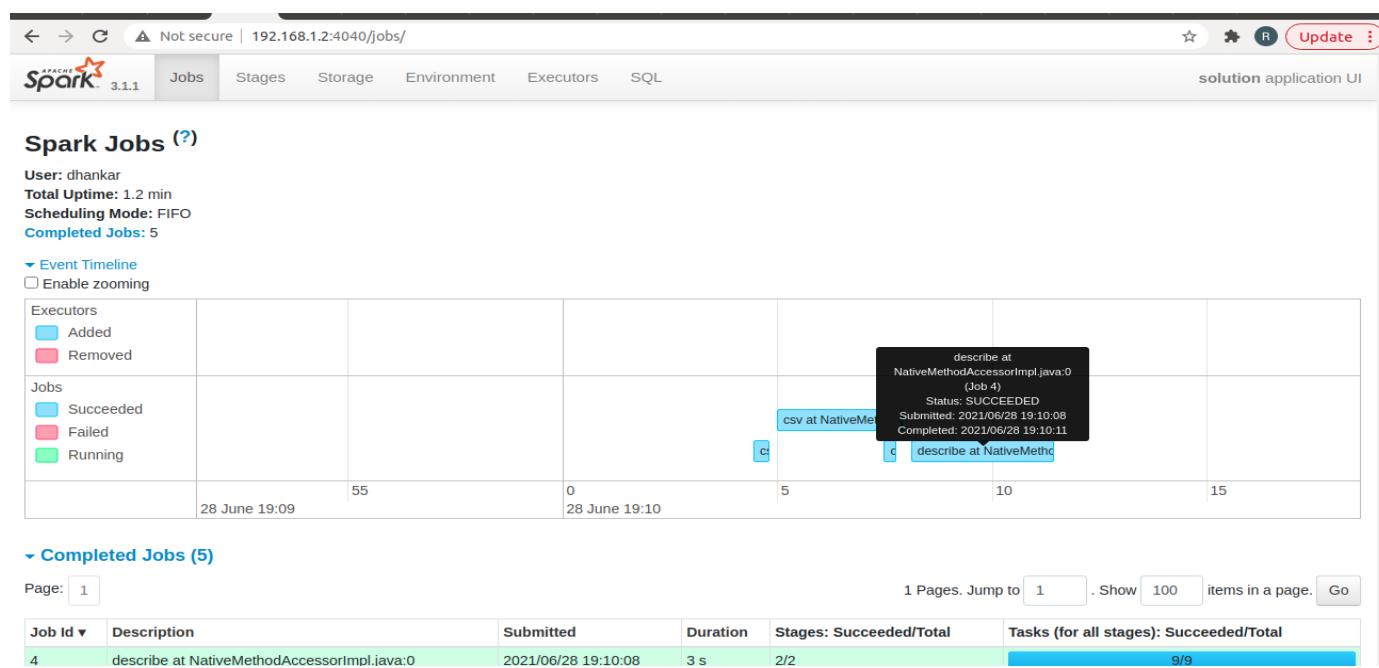
from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import VectorAssembler, StringIndexer, MinMaxScaler
```

In [4]:

```
spark_session1 = SparkSession.builder.appName('solution').getOrCreate()
print(type(spark_session1)) #<class 'pyspark.sql.session.SparkSession'>
```

localhost:8889/notebooks/GitUp_PySpark_june21/GitUp/pySpark_jun21/pySpark_local_flights_data.ipynb#

The Spark Jobs running can be seen at the Spark UI - default URL = <http://192.168.1.2:4040/jobs/>



Added						
Removed						
Jobs						
Succeeded						
Failed						
Running						
	55	0	5	10	15	
	28 June 19:09	28 June 19:10				

Completed Jobs (5)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4	describe at NativeMethodAccessorImpl.java:0 describe at NativeMethodAccessorImpl.java:0	2021/06/28 19:10:08	3 s	2/2	9/9
3	head at <ipython-input-5-13656e7e9637>:11 head at <ipython-input-5-13656e7e9637>:11	2021/06/28 19:10:07	27 ms	1/1	1/1
2	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2021/06/28 19:10:07	0.3 s	2/2	9/9
1	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/06/28 19:10:04	2 s	1/1	8/8
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/06/28 19:10:04	0.4 s	1/1	1/1

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Azure DELTA Lake

As seen below - these are the screen captures for the Azure delta lake datasets created from the default (databricks in-built datasets)

This has been done on the databricks Community Edition

The jupyter notebooks have been updated

```

2021-06-24 - DBFS Example (1) (Python)
Overview
1 events.write.format("delta").mode("overwrite").partitionBy("date").save("/delta/rohit_events_data_set/")
(4) Spark Jobs
  Job 7 View (Stages: 1/1)
    Stage 7: 8/8
      Job 9 View (Stages: 1/1)
      Job 10 View (Stages: 1/1, 1 skipped)
      Job 11 View (Stages: 1/1, 2 skipped)

Command took 27.06 seconds -- by rohit.dhankar@strategic-leadership-lcc-india.com at 8/7/2021, 1:32:39 pm on mlflow_clus_1 (clone)

Cmd 8
1 %fs ls /delta/rohit_events_data_set/
path          name   size
1 dbfs:/delta/rohit_events_data_set/_delta_log/ _delta_log/ 0
2 dbfs:/delta/rohit_events_data_set/date=2016-07-26/ date=2016-07-26/ 0
3 dbfs:/delta/rohit_events_data_set/date=2016-07-27/ date=2016-07-27/ 0
4 dbfs:/delta/rohit_events_data_set/date=2016-07-28/ date=2016-07-28/ 0

Showing all 4 rows.

Command took 1.26 seconds -- by rohit.dhankar@strategic-leadership-lcc-india.com at 8/7/2021, 1:36:35 pm on mlflow_clus_1 (clone)

```

2021-06-24 - DBFS Example (1) (Python)

mlflow_clus_1 (clone)

Overview

5	Open	2016-07-28	2016-07-28
6	Open	2016-07-28	2016-07-28
...

Truncated results, showing first 1000 rows.

Command took 8.96 seconds -- by rohit.dhankar@strategic-leadership-llc-india.com at 8/7/2021, 1:28:54 pm on mlflow_clus_1 (clone)

Cmd 7

```
1 events.write.format("delta").mode("overwrite").partitionBy("date").save("/delta/rohit_events_data_set/")
```

(4) Spark Jobs

- Job 7 View (Stages: 1/1)
Stage 7: 8/8
- Job 9 View (Stages: 1/1)
- Job 10 View (Stages: 1/1, 1 skipped)
- Job 11 View (Stages: 1/1, 2 skipped)

Command took 27.06 seconds -- by rohit.dhankar@strategic-leadership-llc-india.com at 8/7/2021, 1:32:39 pm on mlflow_clus_1 (clone)

Cmd 8

```
1
```

Cmd 9

```
1
```

Azure DELTA Engine

[Delta Lake and Delta Engine guide - Azure Databricks - Workspace](#)

Hosting RStudio Server Pro or RServer Open Source- On DataBricks

[RStudio on Azure Databricks - Azure Databricks - Workspace](#)

RServer Open Source- On DataBricks

We create a new Cluster - name it - **r_test_1** , in this cluster we choose the Runtime as a ML Runtime and the RStudio Open Source- On DataBricks , is pre installed as seen below .

The screenshot shows the Microsoft Azure Databricks Clusters page. A cluster named "r_test_1" is selected. The "Apps" tab is active, showing the RStudio Server configuration. A red box highlights the "Open RStudio" link and the login credentials provided: Username: infobot2016@gmail.com and Password: [REDACTED].

Microsoft Azure | Databricks

Clusters /

Free trial ends in 14 days. [Upgrade to Premium](#) in Azure Portal ?

Portal infobot2016@gmail.com

r_test_1 Edit Permissions Clone Restart Terminate Delete

Configuration Notebooks Libraries Event Log Spark UI Driver Logs Metrics Apps Spark Cluster UI - Master

Web Terminal
Web terminal provides a Bash terminal running in the driver node. See the [documentation](#) for more details.
[Launch Web Terminal](#)

RStudio Server
RStudio is a registered trademark of RStudio Inc.

To use RStudio Server, you need to install the RStudio Server binary package on the Spark driver. See the [documentation](#) for instructions.

[Open RStudio](#)

For RStudio Server Free, you must log in using this username and password:
Username: infobot2016@gmail.com
Password: [REDACTED] show

Microsoft Azure | Databricks

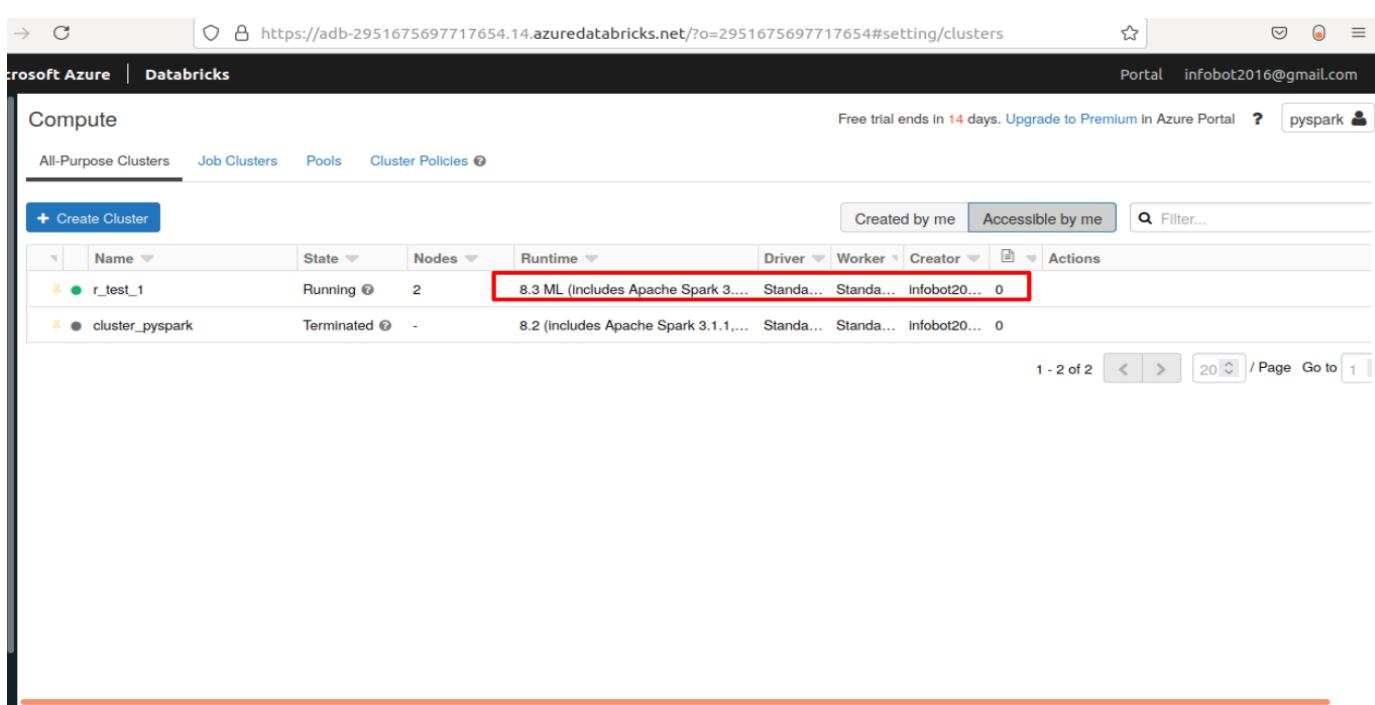
Compute

All-Purpose Clusters Job Clusters Pools Cluster Policies

Created by me Accessible by me Filter... Actions

Name	State	Nodes	Runtime	Driver	Worker	Creator	Actions
r_test_1	Running	2	8.3 ML (includes Apache Spark 3....)	Standar...	Standar...	infobot20...	0
cluster_pyspark	Terminated	-	8.2 (includes Apache Spark 3.1.1,...)	Standar...	Standar...	infobot20...	0

1 - 2 of 2 / Page Go to 1



https://adb-2951675697717654.14.azuredatabricks.net/driver-proxy/o/2951675697717654/0629-11294

R File Edit Code View Plots Session Build Debug Profile Tools Help infobot2016 Project: (None)

Console Terminal Jobs

```
R version 4.0.4 (2021-02-15) -- "Lost Library Book"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

Environment History Connections

Global Environment

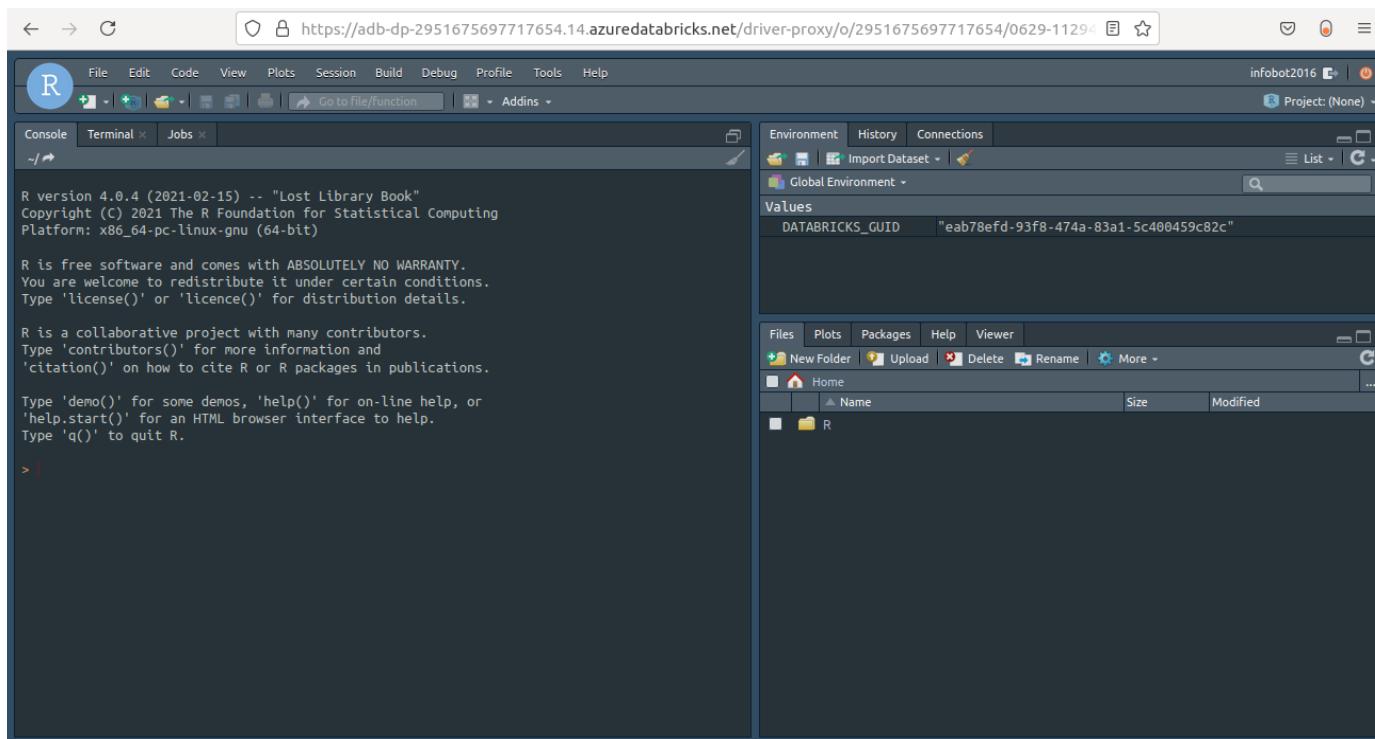
Values

DATABRICKS_GUID	"eab78efd-93f8-474a-83a1-5c400459c82c"
-----------------	--

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Name	Size	Modified
Home		
R		



The screenshot shows the RStudio interface running on an Azure Databricks workspace. The left pane displays the R console output, which includes session setup, package loading (sparklyr), and connection to a Databricks cluster. The right pane shows the Environment tab with connections to Spark, Log, and SQL, and the Packages tab listing installed packages: sparklyr (version 1.5.2), SparkR (version 3.1.2), and Matrix (version 1.3-2). A search bar at the top right is set to 'Spark'.

```

The following objects are masked from 'package:stats':
  cov, filter, lag, na.omit, predict, sd, var, window

The following objects are masked from 'package:base':
  as.data.frame, colnames, colnames<-, drop, endsWith, intersect,
  rank,
  rbind, sample, startsWith, subset, summary, transform, union

> sparkR.session()
Spark package found in SPARK_HOME: /databricks/spark
Java ref type org.apache.spark.sql.SparkSession id 1
> sparkR:::sparkR.session()
Java ref type org.apache.spark.sql.SparkSession id 1
> library(sparklyr)

Attaching package: 'sparklyr'

The following objects are masked from 'package:SparkR':
  collect, distinct

> sc <- spark_connect(method = "databricks")
Warning messages:
1: In file.create(to[okay]) :
  cannot create file '/usr/local/lib/R/site-library/sparklyr/java
//sparklyr-2.2.2-11.jar', reason 'Permission denied'
2: In file.create(to[okay]) :
  cannot create file '/usr/local/lib/R/site-library/sparklyr/java
//sparklyr-2.1-2.11.jar', reason 'Permission denied'
>

```

Name	Description	Version
<input checked="" type="checkbox"/> sparklyr	R Interface to Apache Spark	1.5.2
<input checked="" type="checkbox"/> SparkR	R Front End for 'Apache Spark'	3.1.2
<input type="checkbox"/> Matrix	Sparse and Dense Matrix Classes and Methods	1.3-2

[RStudio on Azure Databricks - Azure Databricks - Workspace](#)

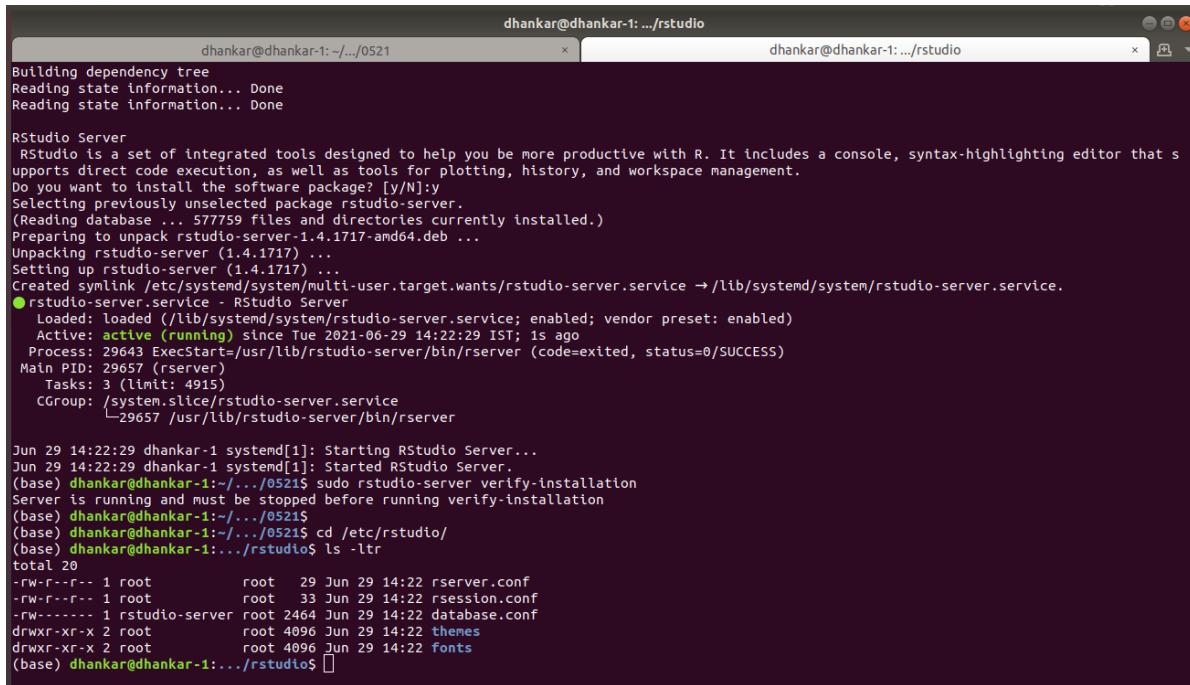
In this manner the RStudio is installed in the DRIVER Node of the Cluster on DataBricks

```

> getwd()
[1] "/home/infobot2016@gmail.com"
>
> clearCache()
> setwd("/dbfs/")
> getwd()
[1] "/dbfs"
> savehistory("/dbfs/r_session_history/session_history.txt")

```

Hosting RStudio Server Pro (RStudio Workbench) - On Local Ubuntu and Connecting to DataBricks - Remote Connect and ODBC



```
dhankar@dhankar-1:~/.../rstudio
Building dependency tree
Reading state information... Done
Reading state information... Done

RStudio Server
RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, and workspace management.
Do you want to install the software package? [y/N]:y
Selecting previously unselected package rstudio-server.
(Reading database ... 577759 files and directories currently installed.)
Preparing to unpack rstudio-server-1.4.1717-amd64.deb ...
Unpacking rstudio-server (1.4.1717) ...
Setting up rstudio-server (1.4.1717) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rstudio-server.service → /lib/systemd/system/rstudio-server.service.
● rstudio-server.service - RStudio Server
   Loaded: loaded (/lib/systemd/system/rstudio-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2021-06-29 14:22:29 IST; 1s ago
       Process: 29643 ExecStart=/usr/lib/rstudio-server/bin/rserver (code=exited, status=0/SUCCESS)
      Main PID: 29657 (rserver)
         Tasks: 3 (limit: 4915)
        CGroup: /system.slice/rstudio-server.service
               └─29657 /usr/lib/rstudio-server/bin/rserver

Jun 29 14:22:29 dhankar-1 systemd[1]: Starting RStudio Server...
Jun 29 14:22:29 dhankar-1 systemd[1]: Started RStudio Server.
(base) dhankar@dhankar-1:~/.../0521$ sudo rstudio-server verify-installation
Server is running and must be stopped before running verify-installation
(base) dhankar@dhankar-1:~/.../0521$ 
(base) dhankar@dhankar-1:~/.../0521$ cd /etc/rstudio/
(base) dhankar@dhankar-1:.../rstudio$ ls -ltr
total 20
-rw-r--r-- 1 root          root  29 Jun 29 14:22 rserver.conf
-rw-r--r-- 1 root          root  33 Jun 29 14:22 rsession.conf
-rw----- 1 rstudio-server root 2464 Jun 29 14:22 database.conf
drwxr-xr-x 2 root          root 4096 Jun 29 14:22 themes
drwxr-xr-x 2 root          root 4096 Jun 29 14:22 fonts
(base) dhankar@dhankar-1:.../rstudio$ 
```

Databricks AutoML

Databricks AutoML helps you automatically apply machine learning to a dataset.

Databricks AutoML - Source (

<https://docs.microsoft.com/en-us/azure/databricks/applications/machine-learning/automl>)

RStats → Final Model saved as .rds files .

(Sample R Code and process discussed)

```
java ref type org.apache.spark.sql.SparkSession id
> library(sparklyr)

Attaching package: 'sparklyr'

The following objects are masked from 'package:Spark
  collect, distinct

> sc <- spark_connect(method = "databricks")
Warning messages:
1: In file.create(to[okay]) :
  cannot create file '/usr/local/lib/R/site-library
//sparklyr-2.2.2.11.jar', reason 'Permission denied'
2: In file.create(to[okay]) :
  cannot create file '/usr/local/lib/R/site-library
//sparklyr-2.1.2.11.jar', reason 'Permission denied'
Error in spark_web_spark_gateway_connection(scon) :
  spark_web is not available while connecting through
gateway
In addition: Warning message:
In value[[3L]](cond) : restarting interrupted promise!
Error in spark_log(spark_gateway_connection(sc, n =
  spark_log is not available while connecting through
gateway
> getwd()
[1] "/home/infobot2016@gmail.com"
>
> clearCache()
> setwd("/dbfs/")
> getwd()
[1] "/dbfs"
>
```

df

Name	Type	Value
df	S4 [272 x 2] (SparkR::SparkDataFrame)	S4 object of class SparkD
env	environment [1]	<environment: 0x5b22e
isCached	logical [1]	FALSE
sdf	environment [2] (S3:jobj)	<environment: 0x5b22e
appid	integer [1]	1624967448
id	character [1]	'7'

df

```
/dbfs/
> clearCache()
> setwd("/dbfs/")
> getwd()
[1] "/dbfs"
> savehistory("/dbfs")
> # Create the SparkR
> df <- as.DataFrame(
> View(df)
> SparkR::sparkR.session()
> Java.ref.type.org.apache.sparklyr.MethodAddCoerce
> save.image("/dbfs/r")
> library(sparklyr)
> sc <- spark_connect(method )
```

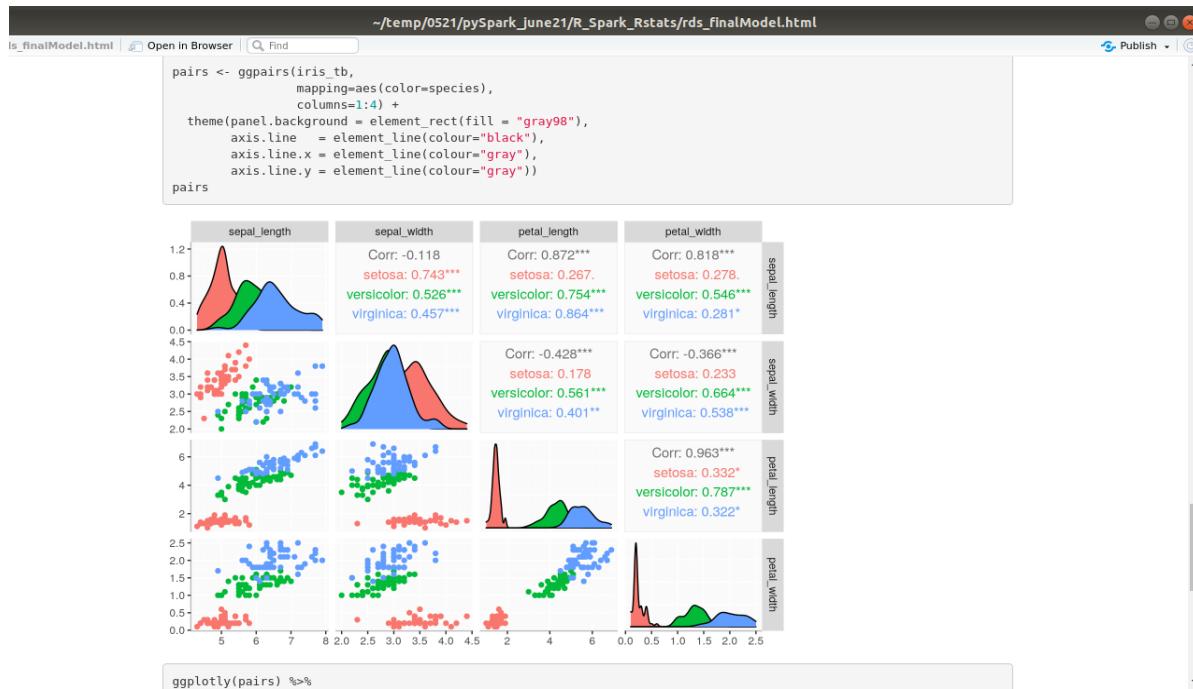
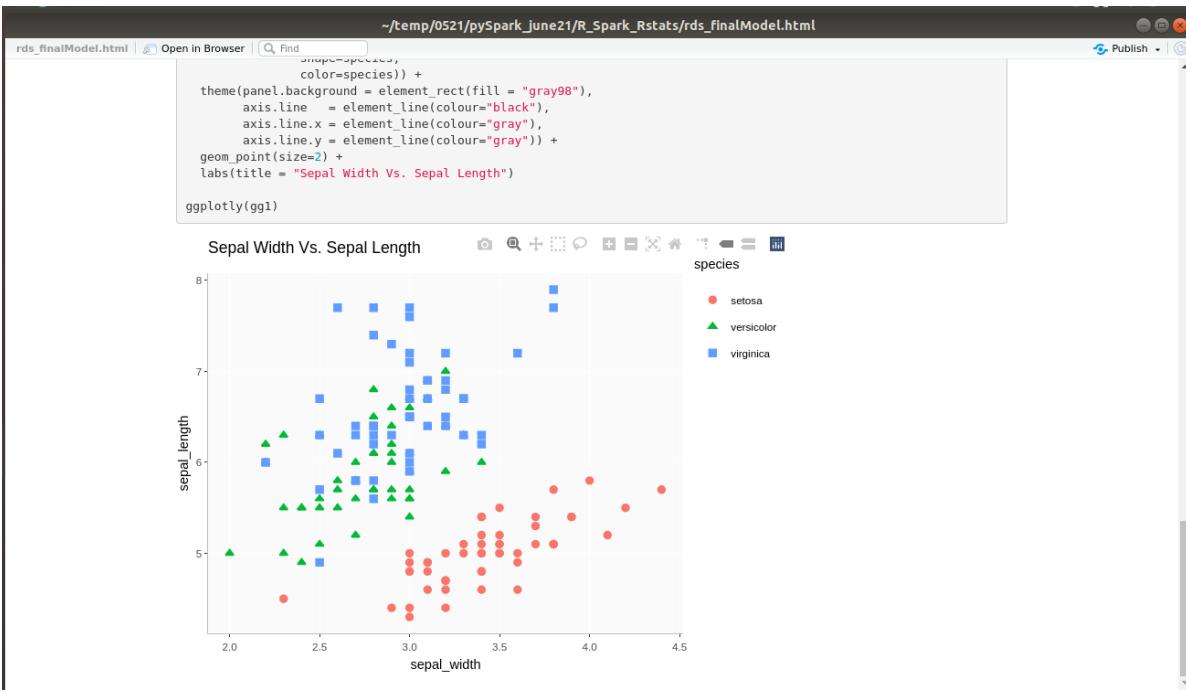
method =
method.skeleton(generic, signature, file, external = FALSE, where
= topenv(parent.frame()))
This function writes a source file containing a call to setMethod to
define a method for the generic function and signature supplied. By
default the method definition is in line in the call, but can be made an
external (previously assigned) function.
Press F1 for additional help

method = c("shell", "livy", "databricks", "test", "qubole"),
app.name = "sparklyr",
version = NULL

Manage Spark Connections

Description

your connections to Spark.



The screenshot shows a web browser window displaying the documentation for the `gknn` package on CRAN. The URL is <https://cran.r-project.org/web/packages/e1071/e1071.pdf>. The page title is "gknn Generalized k-Nearest Neighbors Classification or Regression". The left sidebar lists various R packages, and the main content area contains the **Description** and **Usage** sections of the `gknn` package.

```
gknn Generalized k-Nearest Neighbors Classification or Regression

Description
gknn is an implementation of the k-nearest neighbours algorithm making use of general distance measures. A formula interface is provided.

Usage
## S3 method for class 'formula'
gknn(formula, data = NULL, ..., subset, na.action = na.pass, scale = TRUE)
## Default S3 method:
gknn(x, y, k = 1, method = NULL,
      scale = TRUE, use_all = TRUE,
      FUN = mean, ...)
## S3 method for class 'gknn'
predict(object, newdata,
        type = c("class", "votes", "prob"),
        ...,
        na.action = na.pass)
```

Further Reading -- `gknn` -- Generalized k-Nearest Neighbors Classification or Regression - <https://cran.r-project.org/web/packages/e1071/e1071.pdf>

The screenshot shows a web browser window displaying R code for training a k-Nearest Neighbors (k-NN) model. The code is contained within a single large code block. It starts by fitting a model to a dataset, then prints the results, and finally displays a confusion matrix.

```
# USING CARET PACKAGE TO ESTIMATE OPTIMAL K

fit <- train(species ~.,
             data = training_set,
             method = "knn")

# Here we output the results from fit!
fit

## k-Nearest Neighbors
##
## 120 samples
##  4 predictor
##  3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 120, 120, 120, 120, 120, ...
## Resampling results across tuning parameters:
##
##   k Accuracy Kappa
##   5  0.9561614  0.9337935
##   7  0.9629487  0.9441134
##   9  0.9666055  0.9496128
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```


An example of **Getting HELP** on various Spark Commands within the **iPython Shell** -

```
In [5]: spark.read?
Type:      property
String form: <property object at 0x7f6d4e047bd0>
Docstring:
>Returns a :class:`DataFrameReader` that can be used to read data
in as a :class:`DataFrame`.

.. versionadded:: 2.0.0

Returns
-----
:class:`DataFrameReader`


In [6]: spark.read??
Type:      property
String form: <property object at 0x7f6d4e047bd0>
Source:
# spark.read.fget
@property
def read(self):
    """
    Returns a :class:`DataFrameReader` that can be used to read data
    in as a :class:`DataFrame`.

    .. versionadded:: 2.0.0

    Returns
    -----
    :class:`DataFrameReader`
    """
    return DataFrameReader(self._wrapped)

In [7]:
```

ORC Data Format --- is similar to Parquet (parquet is default) , ORC (Optimized Row Columnar) read further here - <https://spark.apache.org/docs/latest/sql-data-sources-orc.html>

```
file_location = "/FileStore/tables/listingsAndReviews.json"
imdb_json = spark.read.option("multiline","true").json(file_location)
file_location_1 = "/FileStore/tables/weather_data.json"
weather_json = spark.read.option("multiline","true").json(file_location_1)
weather_json_text = spark.read.option("multiline","true").text(file_location_1)
```

The screenshot shows the Databricks Cluster UI for the cluster `ml_83_sp_311`. The top navigation bar includes links for Clusters, Notebooks, Libraries, Event Log, Spark UI (which is selected), Driver Logs, Metrics, Apps, and Spark Cluster UI - Master. The main content area displays the cluster's configuration, including its hostname (`ec2-54-188-20-235.us-west-2.compute.amazonaws.com`) and Spark version (`8.3.x-cpu-ml-scala2.12`). Below this, there are tabs for Jobs, Stages, Storage (which is selected), Environment, Executors, SQL, JDBC/ODBC Server, and Structured Streaming. The Storage section contains a table for Parquet IO Cache:

Data Read from External Filesystem	Data Read from IO Cache	Data Written to IO Cache	Estimated Size of Repeatedly Read Data	Cache Metadata Manager Peak Disk Usage
0.0 B	0.0 B	0.0 B	0.0 B (0%) - 0.0 B (0%)	0.0 B

Further Readings :-

Directed Acyclic Graphs - DAG's --

Viewing the DAG within the Spark UI --

Resource Profiling - How much of available resources being used by a SPARK APP -

DataBricks CLI - (<https://docs.microsoft.com/en-us/azure/databricks/dev-tools/cli/>,)

Cluster Mode --- Standard , **High Concurrency Clusters** and **Single Node** Clusters
(<https://docs.databricks.com/clusters/configure.html#cluster-mode>)

Authentication --- Authenticate using **Azure Active Directory tokens**

(<https://docs.microsoft.com/en-us/azure/databricks/dev-tools/api/latest/aad/>)

Connect **SQL Workbench** to Azure DataBricks Spark Custer -

(<https://docs.microsoft.com/en-us/azure/databricks/integrations/bi/workbench>)

Simba Connect ODBC / JDBC - DataBricks SPARK -

(https://docs.datafabric.hpe.com/62/attachments/JDBC_ODBC_drivers/SparkODBCInstallandConfigurationGuide_2.6.1.pdf)