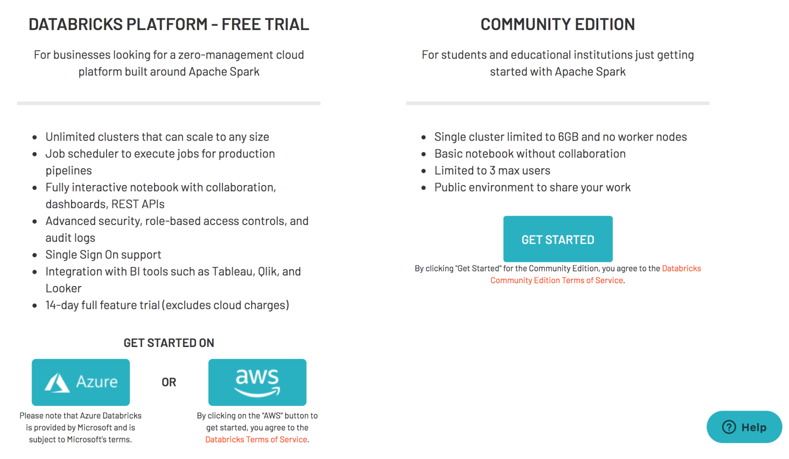
01: Databricks getting started – Spark, Shell, SQL

Posted on April 10, 2020

Step 1: Signup to Databricks community edition – https://databricks.com/try-databricks. Fill in the details and you can leave your mobile number blank. Select "**COMMUNITY EDITION**" ==> "**GET STARTED**".



DATABRICKS PLATFORM - FREE TRIAL

For businesses looking for a zero-management cloud platform built around Apache Spark

- Unlimited clusters that can scale to any size
- Job scheduler to execute jobs for production pipelines
- Fully interactive notebook with collaboration, dashboards, REST APIs
- 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)

GET STARTED ON

Azure OR aws

Please note that Azure Databricks is provided by Microsoft and is subject to Microsoft's terms.

By clicking on the "AWS" button to get started, you agree to the Databricks Terms of Service.

COMMUNITY EDITION

For students and educational institutions just getting started with Apache Spark

- Single cluster limited to 6GB and no worker nodes
- Basic notebook without collaboration
- Limited to 3 max users
- Public environment to share your work

GET STARTED

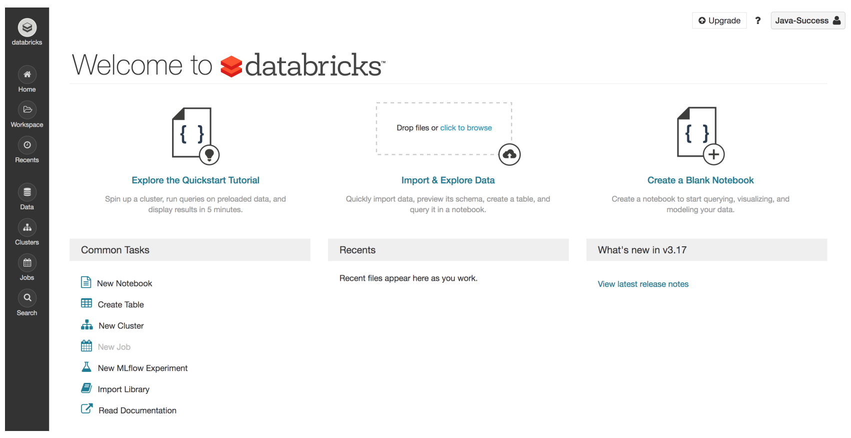
By clicking "Get Started" for the Community Edition, you agree to the Databricks Community Edition Terms of Service.

Databricks getting started

If you have a Cloud account then you can use it.

Step 2: Check your email and click the "link" in the email & reset your password.

Step 3: Login to Databricks notebook: https://community.cloud.databricks.com/login.html.



Welcome to databricks

Drop files or click to browse

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

- New Notebook
- Create Table
- New Cluster
- New Job
- New MLflow Experiment
- Import Library
- Read Documentation

Recents

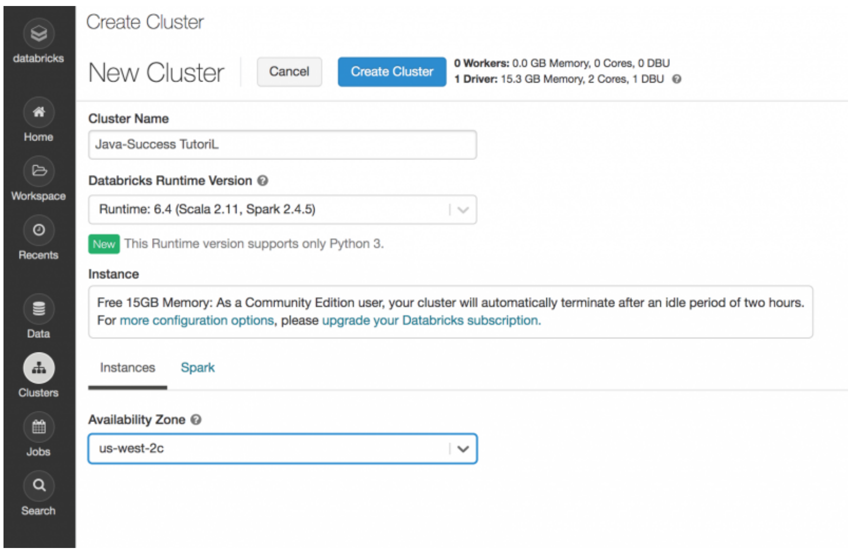
Recent files appear here as you work.

What's new in v3.17

View latest release notes

Databricks getting started dashboard

Step 4: Create a **CLUSTER** and it will take a few minutes to come up. This cluster will go down after 2 hours.


Create Cluster

New Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU

Cluster Name: Java-Success TutoriL

Databricks Runtime Version: Runtime: 6.4 (Scala 2.11, Spark 2.4.5)

New This Runtime version supports only Python 3.

Instance

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.

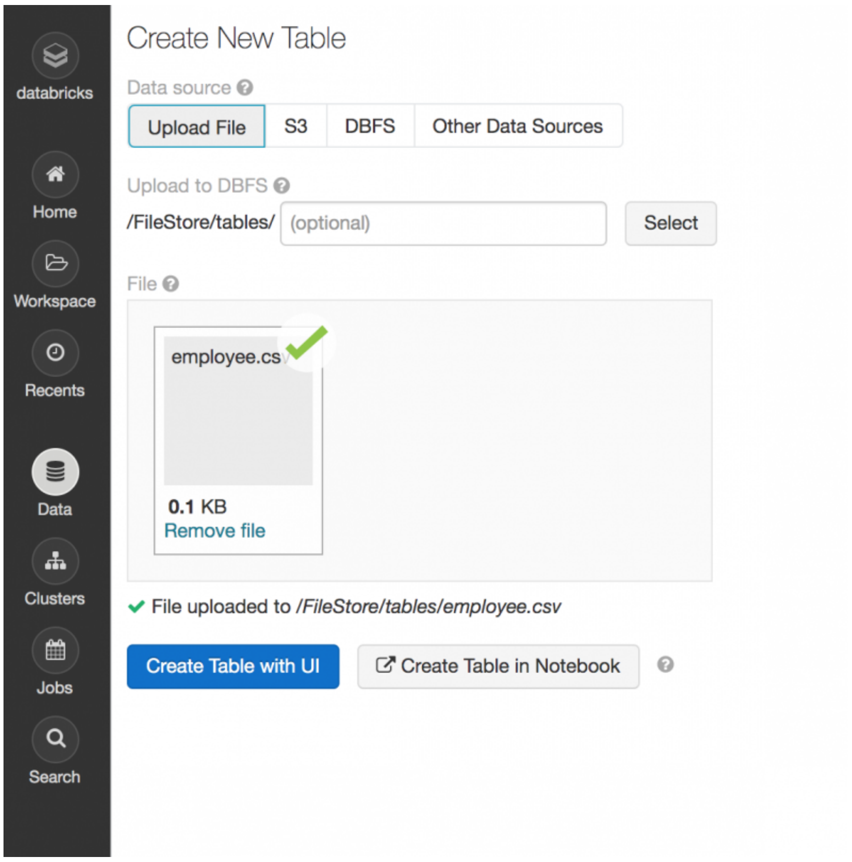
Availability Zone: us-west-2c

Databricks create a cluster

Step 5: Select "**DATA**", and upload a file named "employee.csv".

```
emp_id,emp_name,emp_city,emp_salary
1, John, Sydney, 35000.00
2, Peter, Melbourne, 45000.00
3, Sam, Sydney,55000.00

```


Create New Table

Data source: Upload File

Upload to DBFS: /FileStore/tables/ (optional)

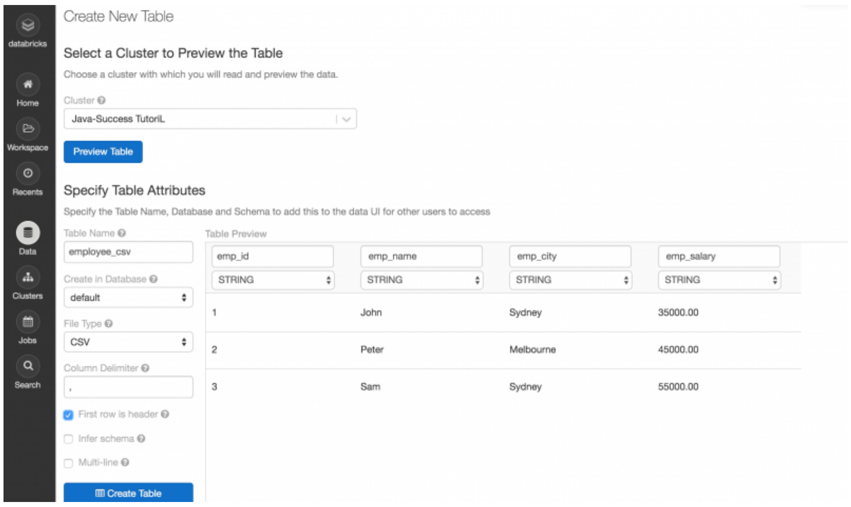
File: employee.csv 0.1 KB

✓ File uploaded to */FileStore/tables/employee.csv*

Databricks upload a .csv

Step 6: "**Create Table With UI**" as shown below:

Note: Please check the "**First row is header**" check box on the LHS so that column names appear from the file.


Create New Table

Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.

Cluster: Java-Success TutoriL

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name: employee_csv

Create in Database: default

File Type: CSV

Column Delimiter: ,

[✓] First row is header
[ ] Infer schema
[ ] Multi-line

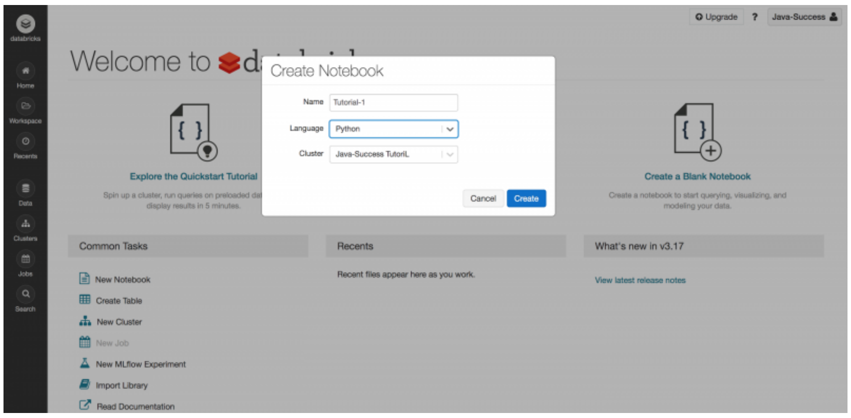
Table Preview

emp_id	emp_name	emp_city	emp_salary
STRING	STRING	STRING	STRING
1	John	Sydney	35000.00
2	Peter	Melbourne	45000.00
3	Sam	Sydney	55000.00

Databricks create a table with .csv

Click on "**Create Table**".

Step 7: Click on the "**databricks**" icon on the LHS menu, and then "**Create a Blank Notebook**".


Create Notebook

Name: Tutorial-1

Language: Python

Cluster: Java-Success TutoriL

Spin up a cluster, run queries on preloaded da… display results in 5 minutes.

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

Recent files appear here as you work.

Databricks blank notebook

Spark in Python (i.e.PySpark)

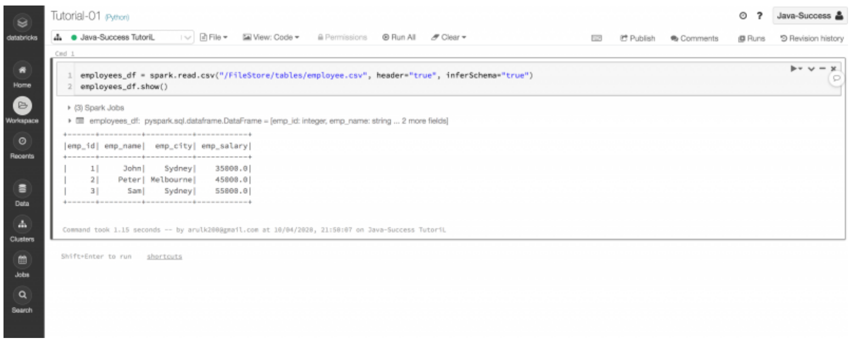
Since we created the notebook as "**python**", we don't have to do "**%python**" as it is the default language. If you want to use "scala" then add "**%scala**" as the first line in a **cell**.

Step 8: Run the below PySpark code to display uploaded "**/FileStore/tables/employee.csv**" as a **Dataframe**.

```
employees_df = spark.read.csv("/FileStore/tables/e
employees_df.show()

```

Click on "**Run**" in the top "RHS" menu.



```
employees_df = spark.read.csv("/FileStore/tables/employee.csv", header="true", inferSchema="true")
employees_df.show()
```

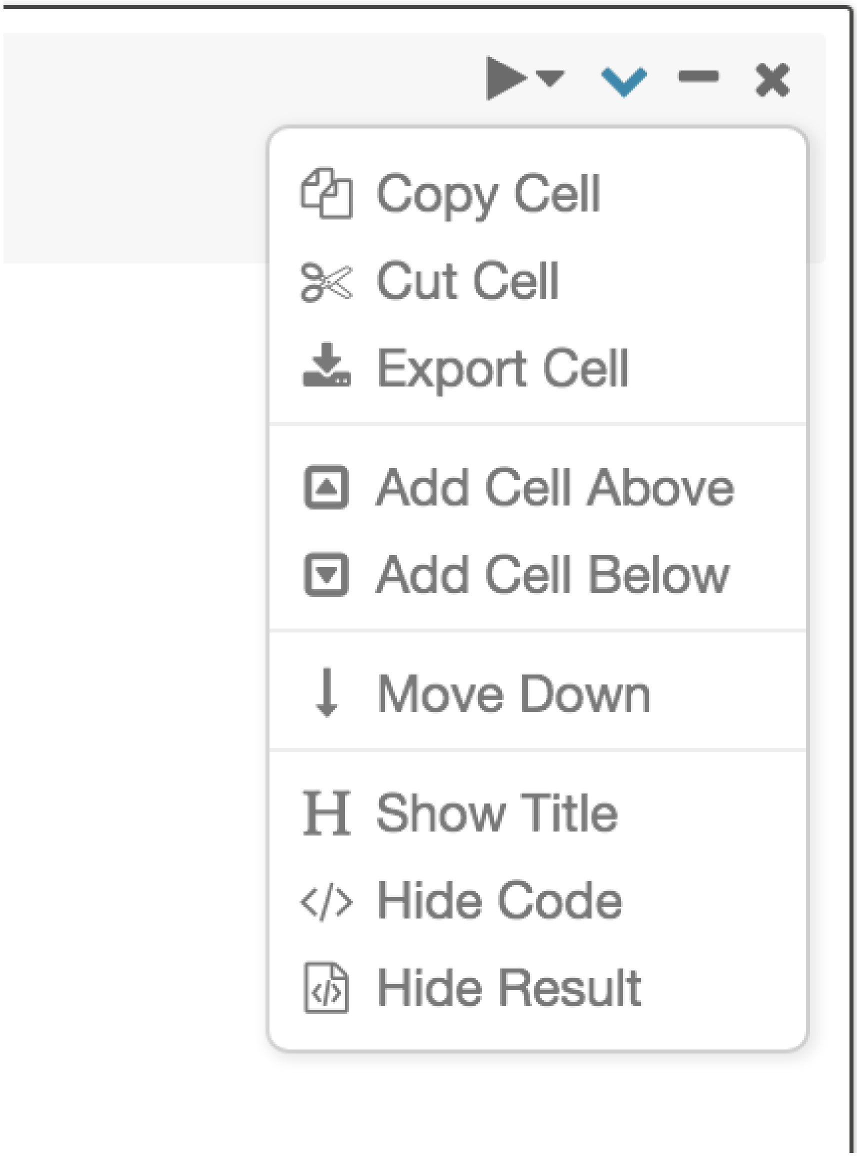
(3) Spark Jobs
employees_df: pyspark.sql.dataframe.DataFrame = [emp_id: integer, emp_name: string ... 2 more fields]

```
+------+---------+---------+----------+
|emp_id| emp_name| emp_city|emp_salary|
+------+---------+---------+----------+
|     1|     John|   Sydney|   35000.0|
|     2|    Peter|Melbourne|   45000.0|
|     3|      Sam|   Sydney|   55000.0|
+------+---------+---------+----------+

Command took 1.15 seconds -- by arulk200@gmail.com at 10/04/2020, 21:50:07 on Java-Success TutoriL
```

Shift+Enter to run   shortcuts

Databricks Notebook simple Dataframe



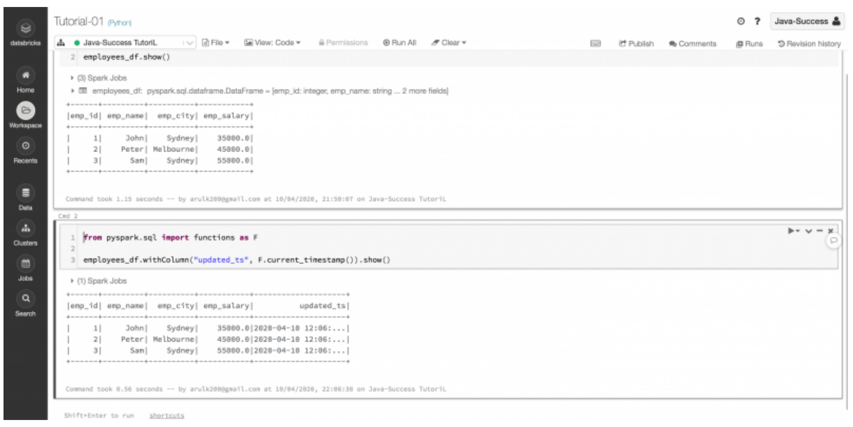
Use “Down Arrow” on in the RHS of a cell to create a new cell.

Step 9: Add a new column to the Dataframe in a separate cell and then **Run**.

```
from pyspark.sql import functions as F

employees_df.withColumn("updated_ts", F.current_ti

```



```
employees_df.show()
```

```
(3) Spark Jobs
employees_df:  pyspark.sql.dataframe.DataFrame = [emp_id: integer, emp_name: string ... 2 more fields]
+------+--------+---------+----------+
|emp_id|emp_name| emp_city|emp_salary|
+------+--------+---------+----------+
|     1|    John|   Sydney|   35000.0|
|     2|   Peter|Melbourne|   45000.0|
|     3|     Sam|   Sydney|   55000.0|
+------+--------+---------+----------+

Command took 1.15 seconds -- by arulk200@gmail.com at 10/04/2020, 21:58:07 on Java-Success TutoriL
```

Cmd 2

```
from pyspark.sql import functions as F

employees_df.withColumn("updated_ts", F.current_timestamp()).show()
```

```
(1) Spark Jobs
+------+--------+---------+----------+--------------------+
|emp_id|emp_name| emp_city|emp_salary|          updated_ts|
+------+--------+---------+----------+--------------------+
|     1|    John|   Sydney|   35000.0|2020-04-10 12:06:...|
|     2|   Peter|Melbourne|   45000.0|2020-04-10 12:06:...|
|     3|     Sam|   Sydney|   55000.0|2020-04-10 12:06:...|
+------+--------+---------+----------+--------------------+

Command took 0.56 seconds -- by arulk200@gmail.com at 10/04/2020, 22:06:38 on Java-Success TutoriL
```

Shift+Enter to run   shortcuts

Databricks notebook add new column to dataframe

Run a shell command

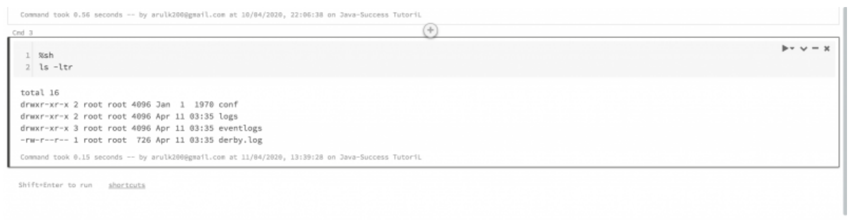
You can run a shell command with “%sh” as shown below:

```
%sh

ls -ltr

```

You can click on the “**+**” in the middle to add new cells.



```
Command took 0.56 seconds -- by arulk200@gmail.com at 10/04/2020, 22:06:38 on Java-Success TutoriL
```

Cmd 3

```
%sh
ls -ltr
```

```
total 16
drwxr-xr-x 2 root root 4096 Jan  1  1970 conf
drwxr-xr-x 2 root root 4096 Apr 11 03:35 logs
drwxr-xr-x 3 root root 4096 Apr 11 03:35 eventlogs
-rw-r--r-- 1 root root  726 Apr 11 03:35 derby.log
Command took 0.15 seconds -- by arulk200@gmail.com at 11/04/2020, 13:39:28 on Java-Success TutoriL
```

Shift+Enter to run   shortcuts

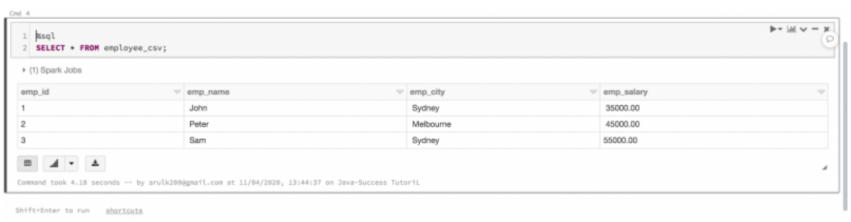
Databricks shell command

Run a SQL command

```
%sql

SELECT * FROM employee_csv;

```



Cmd 4

```
%sql
SELECT * FROM employee_csv;
```

▸ (1) Spark Jobs

emp_id	emp_name	emp_city	emp_salary
1	John	Sydney	35000.00
2	Peter	Melbourne	45000.00
3	Sam	Sydney	55000.00

```
Command took 4.18 seconds -- by arulk200@gmail.com at 11/04/2020, 13:44:37 on Java-Success TutoriL
```

Shift+Enter to run shortcuts

Databricks SQL command

dbutils

```
%python

dbutils.fs.rm("/tmp/my-delta-table", recurse=True)

```

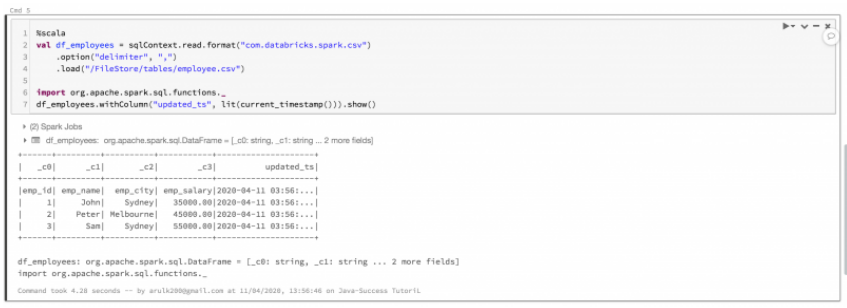
Spark in Scala

```
%scala

val df_employees = sqlContext.read.format("com.data
     .option("delimiter", ",")
     .load("/FileStore/tables/employee.csv")

import org.apache.spark.sql.functions._
df_employees.withColumn("updated_ts", lit(current_

```



Cmd 5

```
%scala
val df_employees = sqlContext.read.format("com.databricks.spark.csv")
    .option("delimiter", ",")
    .load("/FileStore/tables/employee.csv")

import org.apache.spark.sql.functions._
df_employees.withColumn("updated_ts", lit(current_timestamp())).show()
```

▸ (2) Spark Jobs

▸ df_employees: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 2 more fields]

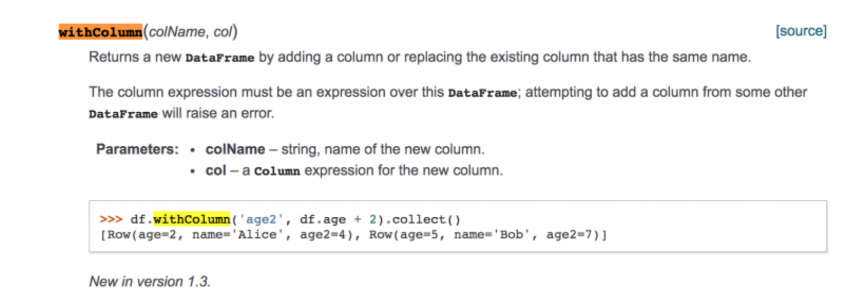
```
+------+---------+---------+----------+--------------------+
|   _c0|      _c1|      _c2|       _c3|          updated_ts|
+------+---------+---------+----------+--------------------+
|emp_id| emp_name| emp_city|emp_salary|2020-04-11 03:56:...|
|     1|     John|   Sydney|  35000.00|2020-04-11 03:56:...|
|     2|    Peter|Melbourne|  45000.00|2020-04-11 03:56:...|
|     3|      Sam|   Sydney|  55000.00|2020-04-11 03:56:...|
+------+---------+---------+----------+--------------------+

df_employees: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 2 more fields]
import org.apache.spark.sql.functions._
Command took 4.28 seconds -- by arulk200@gmail.com at 11/04/2020, 13:56:46 on Java-Success TutoriL
```

Databricks Spark in Scala

Important: PySpark API

Have the PySpark API PySpark modules handy to code. You can click on “Dataframe” to see what functions are available. For example, **withColumn** function in the Dataframe module



`withColumn`(*colName, col*) [source]

Returns a new `DataFrame` by adding a column or replacing the existing column that has the same name.

The column expression must be an expression over this `DataFrame`; attempting to add a column from some other `DataFrame` will raise an error.

Parameters:
- **colName** – string, name of the new column.
- **col** – a `Column` expression for the new column.

```
>>> df.withColumn('age2', df.age + 2).collect()
[Row(age=2, name='Alice', age2=4), Row(age=5, name='Bob', age2=7)]
```

New in version 1.3.

Databricks PySpark Modules & API

Where are my notebooks saved?

Your notebooks will be saved under "**workspace**" ==> "**users**" ==> "**[your username]**"

What if you want to practice in Scala?

You an try the examples from Tutorials – Spark Scala on Zeppelin with some minor changes.

‹ AWS Web Application Security Q&As

02: Databricks – Spark schemas, casting & PySpark API ›

Disclaimer