

Lab 9: Databricks – A Quick Start Tutorial

Introduction - Databricks Setup & Installation

Databricks is a company founded by the original creators of Apache Spark. Databricks grew out of the AMPLab project at University of California, Berkeley that was involved in making Apache Spark, an open-source distributed computing framework built atop Scala. However, despite being built atop Scala, users can use other programming language to work with Databricks.

Databricks is a cloud-enabled platform, therefore many users can collaborate on a project. Databricks is an open and unified platform to collaboratively run all types of analytics workloads, from data preparation to exploratory analysis and predictive analytics, at scale.

Setup

You can either setup a trial or a community edition account. However, for the purpose of this workshop let us go with community edition.

Go to the following link in your browser: <https://databricks.com/try-databricks>

Put in all your info to sign up for a databricks account:

Try Databricks

An open and unified data analytics platform for data engineering, machine learning, and analyt

From the original creators of Apache Spark™, Delta Lake, MLflow, and Koalas

Tell us a little about yourself to get started.

* First Name:	* Last Name:
<input type="text" value="Ngoc"/>	<input type="text" value="Dinh"/>
* Company Name	* Work Email
<input type="text" value="Fordham University"/>	<input type="text" value=""/>
* How would you describe your role?	* What is your intended use case?
<input type="text" value="Student"/>	<input type="text" value="Research - Academia"/>

Click on sign up. It should take you to the following page. Click on “Get Started” for Community edition.

Try Databricks

An open and unified data analytics platform for data engineering, machine learning, and analytics

From the original creators of Apache Spark™, Delta Lake, MLflow, and Koalas

Select a platform

DATABRICKS PLATFORM - FREE TRIAL

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

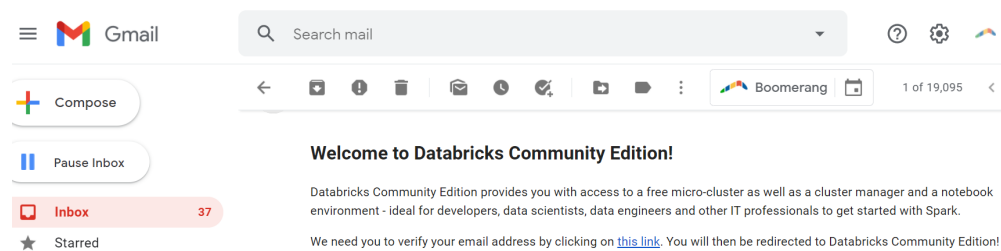
COMMUNITY EDITION

For students and educational institutions

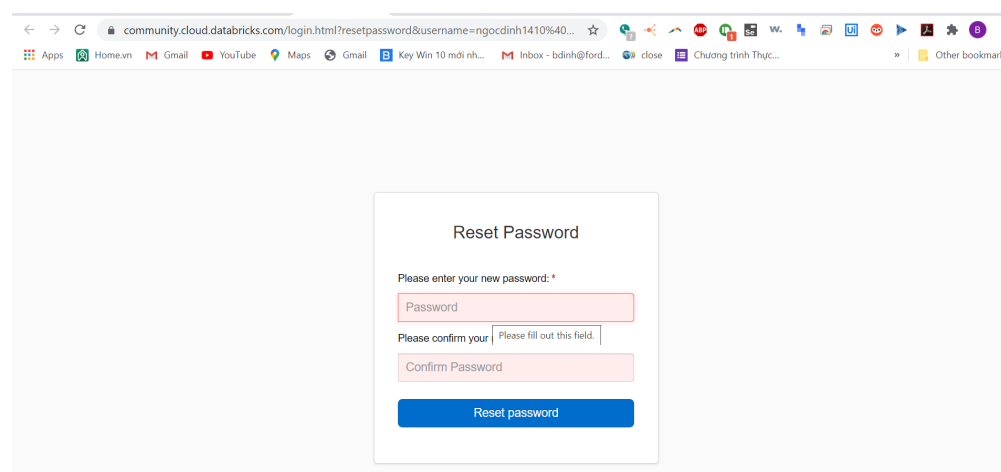
- Single cluster limited to 6GB and no worker nodes
- Basic notebooks without collaboration
- Limited to 3 max users
- Public environment to share your work

GET STARTED

Check your email and verify the email address by clicking on the link provided.



Reset your password:



Part 1 - A Quick Start Tutorial

Requirements

You are logged into a Databricks workspace. See [Sign up](#) for a free Databricks trial.

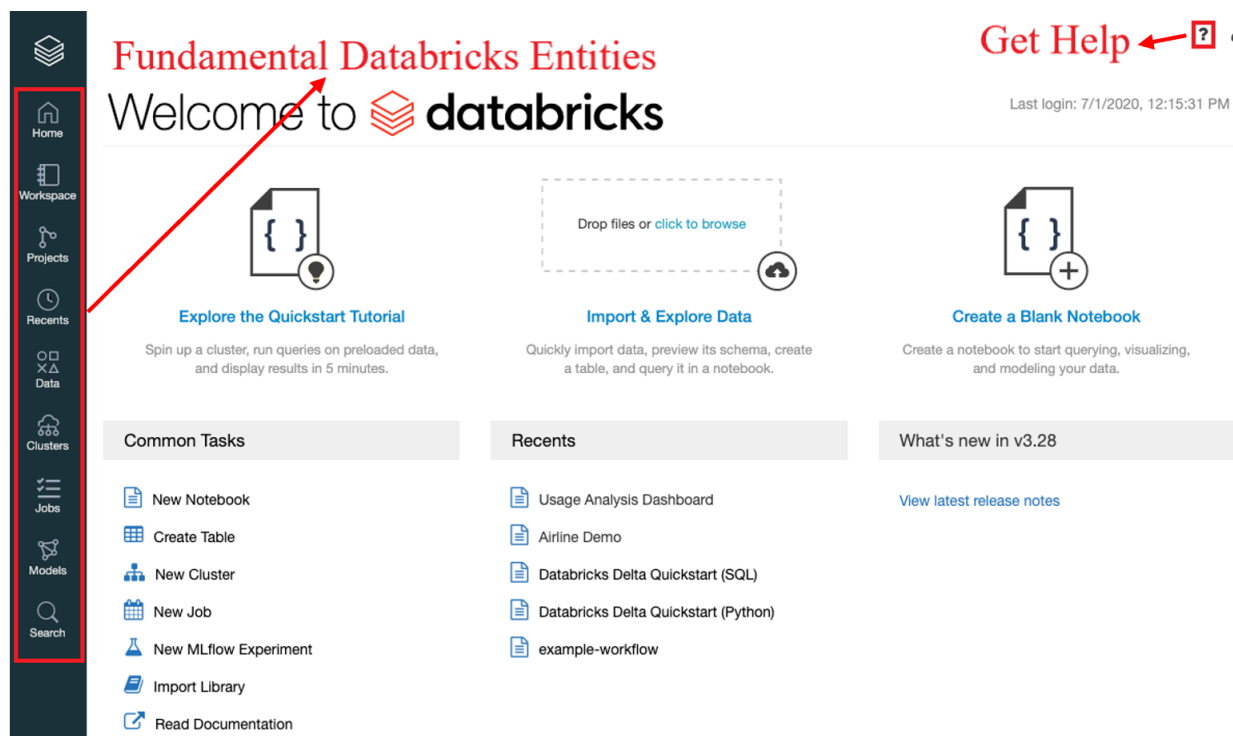
Get started with Databricks

Step 1: Orient yourself to the Databricks UI

Landing page

From the sidebar at the left and the Common Tasks list on the landing page, you access fundamental Databricks entities: Workspace, clusters, tables, notebooks, jobs, and libraries. The Workspace is the special root folder that stores your Databricks assets, such as notebooks and libraries, and the data that you import.

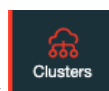
To get help, click the question icon Question Icon at the top right-hand corner.



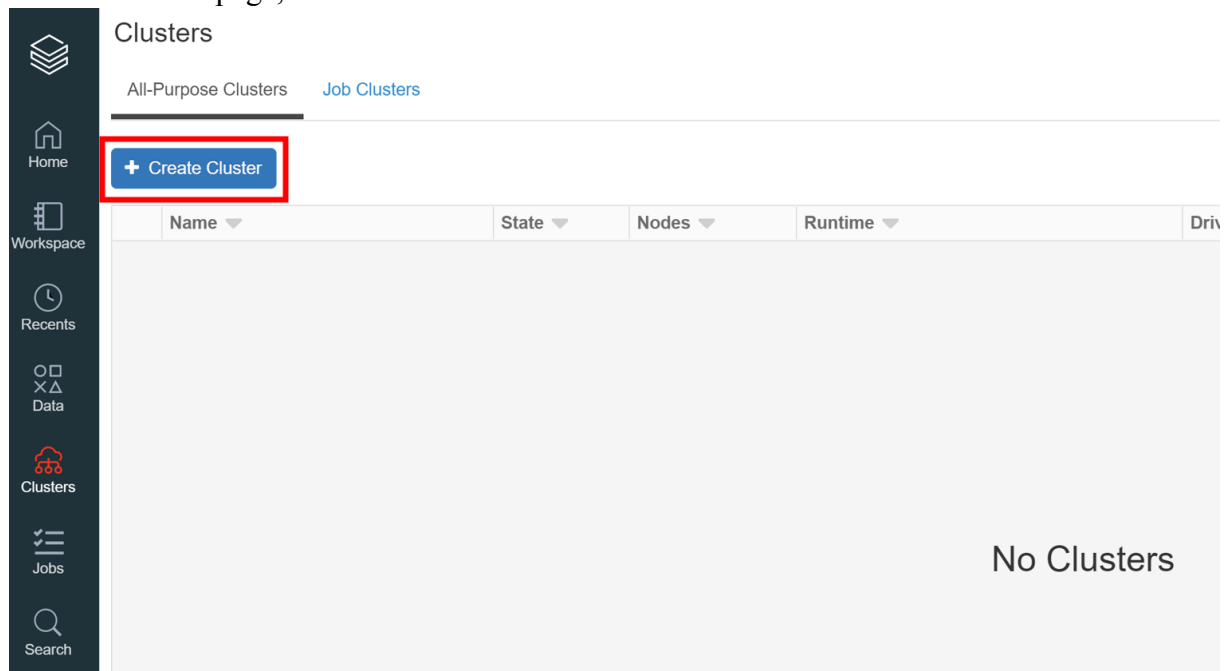
Step 2: Create a cluster

A cluster is a collection of Databricks computation resources. To create a cluster:

1. In the sidebar, click the **Clusters** button

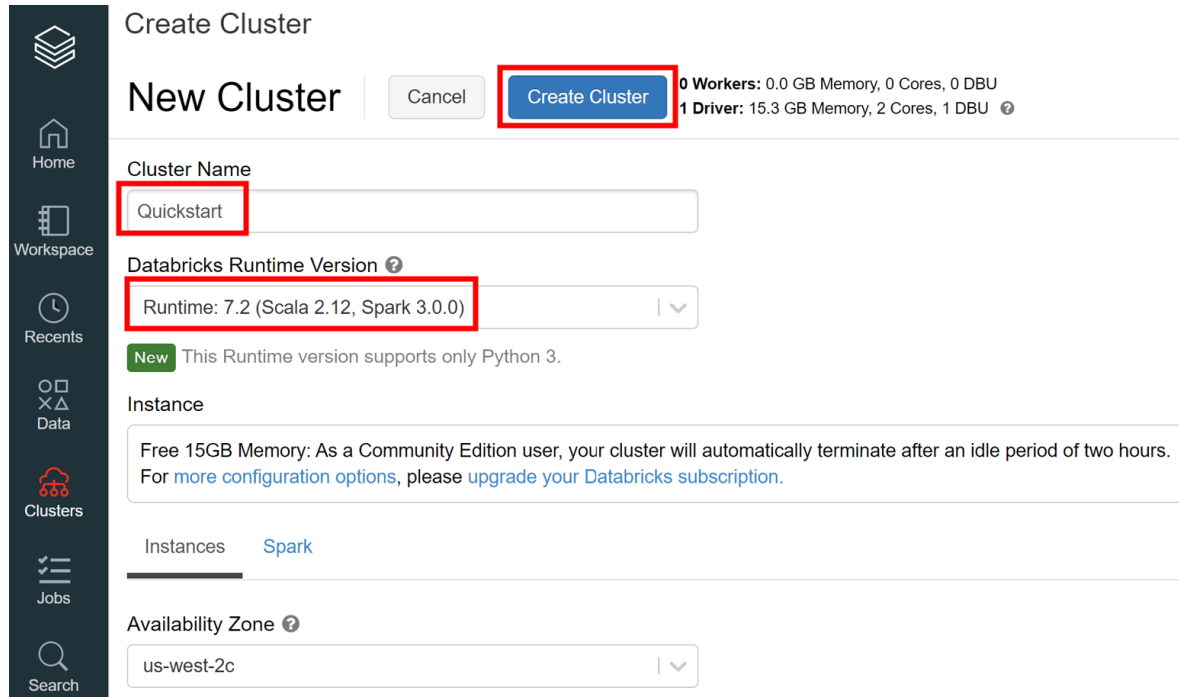


2. On the Clusters page, click **Create Cluster**.



3. On the Create Cluster page, specify the cluster name **Quickstart** and select **7.2 (Scala 2.12, Spark 3.0.0)** in the Databricks Runtime Version drop-down.

4. Click **Create Cluster**.

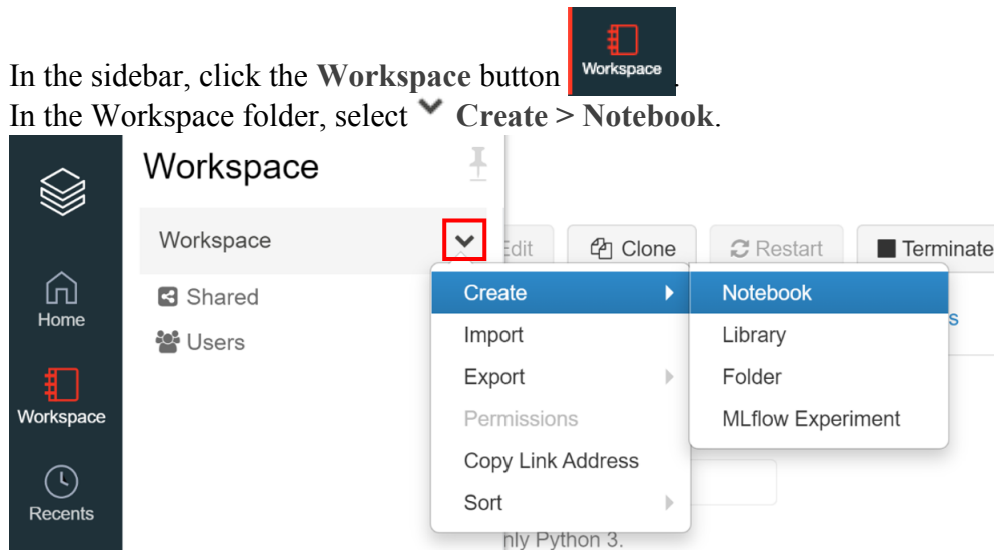


Please notice that for free account, the maximum number of active clusters is 1. You need to terminate an existing cluster or upgrade to a larger plan if you want more than one active cluster.

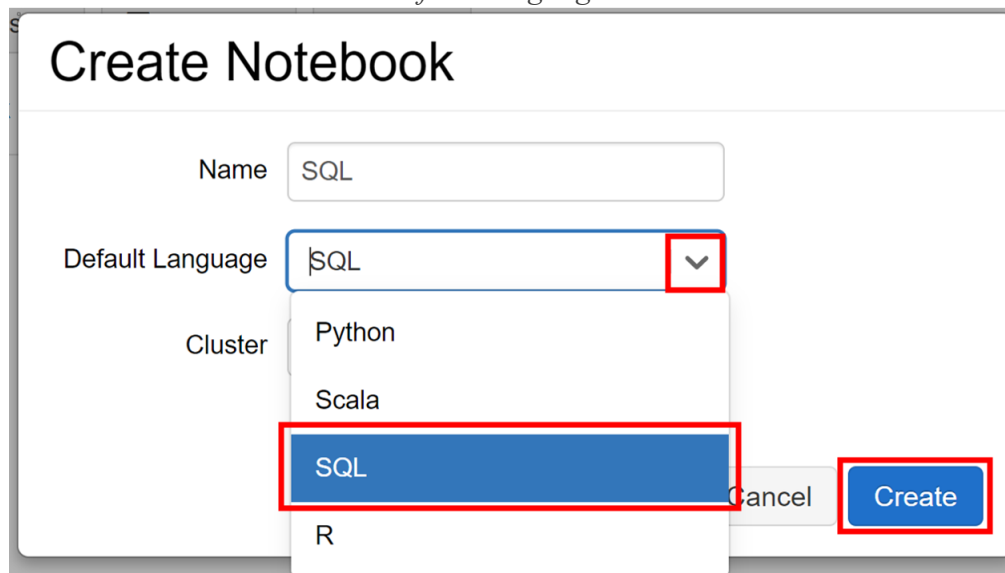
Step 3: Create a notebook

A notebook is a collection of cells that run computations on an Apache Spark cluster. There can be multiple workspaces in one cluster. To create a notebook in the Workspace:

1. In the sidebar, click the **Workspace** button
2. In the Workspace folder, select **Create > Notebook**.



3. On the Create Notebook dialog, enter a name and select **SQL** in the Language drop-down. This selection determines the *default language* of the notebook.



4. Click **Create**. The notebook opens with an empty cell at the top.

Step 4: Create a table

Create a table using data from a sample CSV data file available in [Databricks datasets](#), a collection of datasets mounted to [Databricks File System \(DBFS\)](#), a distributed file system installed on Databricks clusters. You have two options for creating the table.

Option 1: Create a Spark table from the CSV data

Use this option if you want to get going quickly, and you only need standard levels of performance. Copy and paste this code snippet into a notebook cell:

```
DROP TABLE IF EXISTS diamonds;
```

```
CREATE TABLE diamonds USING CSV OPTIONS (path "/databricks-  
datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header "true")
```

Run cells by pressing **SHIFT + ENTER**. The notebook automatically attaches to the cluster you created in Step 2 and runs the command in the cell. Or click the **Run Cell** button here:



Option 2: Write the CSV data to Delta Lake format and create a Delta table

Delta Lake offers a powerful transactional storage layer that enables fast reads and other benefits. Delta Lake format consists of Parquet files plus a transaction log. Use this option to get the best performance on future operations on the table.

1. Read the CSV data into a DataFrame and write out in Delta Lake format. This command uses a Python **language magic command**, which allows you to interleave commands in languages other than the notebook default language (SQL). Copy and paste this code snippet into a notebook cell:

```
%python
```

```
diamonds = spark.read.csv("/databricks-datasets/Rdatasets/data-  
001/csv/ggplot2/diamonds.csv", header="true", inferSchema="true")
```

```
diamonds.write.format("delta").save("/mnt/delta/diamonds")
```

2. Create a Delta table at the stored location. Copy and paste this code snippet into a notebook cell:


```
DROP TABLE IF EXISTS diamonds;
```

```
CREATE TABLE diamonds USING DELTA LOCATION '/mnt/delta/diamonds/'
```

Run cells by pressing **SHIFT + ENTER** or click the **Run Cell** button. The notebook automatically attaches to the cluster you created in Step 2 and runs the command in the cell.

Step 5: Query the table

Run a SQL statement to query the table for the average diamond price by color.

1. To add a cell to the notebook, mouse over the cell bottom and click the  icon.

OK

Command took 8.48 seconds

Shift+Enter to run

[shortcuts](#)

Insert a new cell

- Copy this snippet and paste it in the cell.

```
SELECT color, avg(price) AS price FROM diamonds GROUP BY color ORDER BY
COLOR
```

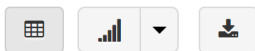
- Press **SHIFT + ENTER** or click the **Run Cell** button. The notebook displays a table of diamond color and average price.

```
1 SELECT color, avg(price) AS price FROM diamonds GROUP BY color ORDER BY COLOR
```

▶ (1) Spark Jobs

	color	price
1	D	3169.9540959409596
2	E	3076.7524752475247
3	F	3724.886396981765
4	G	3999.135671271697
5	H	4486.669195568401
6	I	5091.874953891553
7	J	5323.81801994302

Showing all 7 rows.



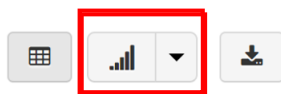
Step 6: Display the data



Display a chart of the average diamond price by color.

- Click the Bar chart icon .

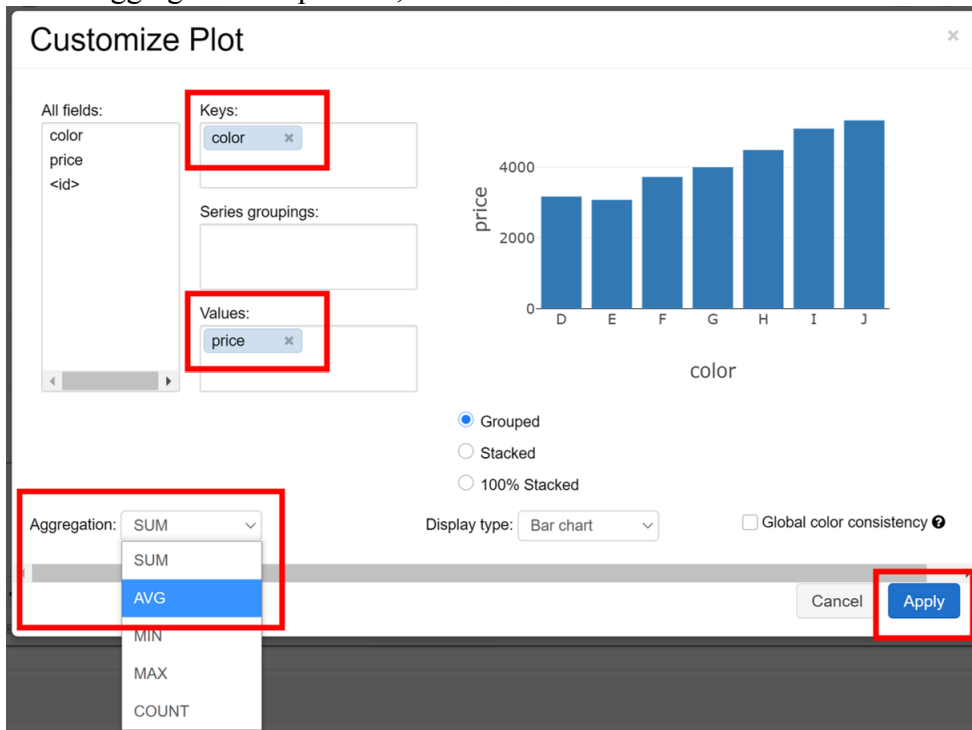
6	I	5091.874953891553
7	J	5323.81801994302

Showing all 7 rows.

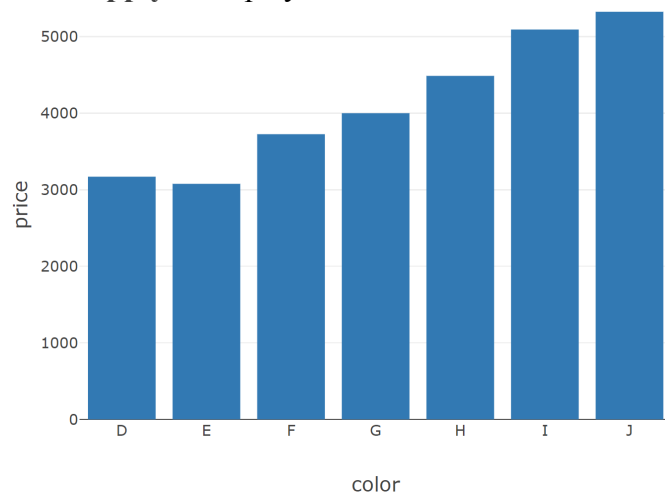


- Click **Plot Options**   **Plot Options...** .
- Drag **color** into the Keys box.
- Drag **price** into the Values box.

5. In the Aggregation drop-down, select **AVG**.



6. Click **Apply** to display the bar chart.



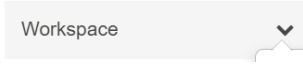
Import Notebooks

Databricks is a diverse community, and there are many free or available notebooks you might see on the internet.

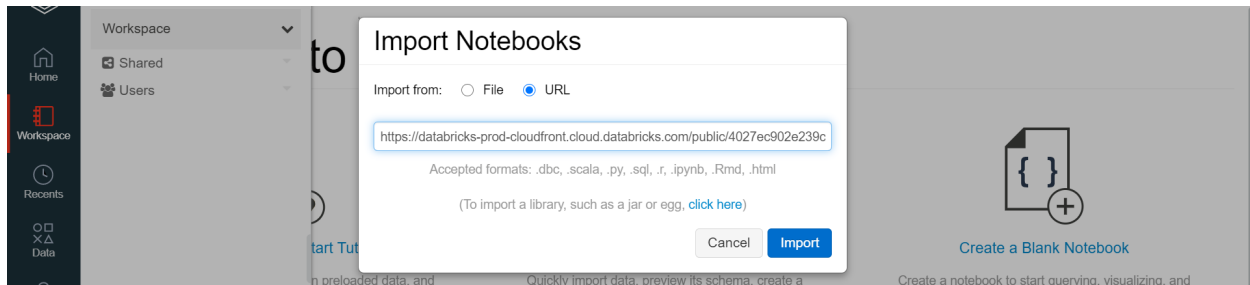
One resource is the databricks user guide: <https://docs.databricks.com/user-guide/index.html>

If you find a databricks notebook on the internet, you can import it into your workspace. For example, we will try to import this one: <https://databricks-prod->

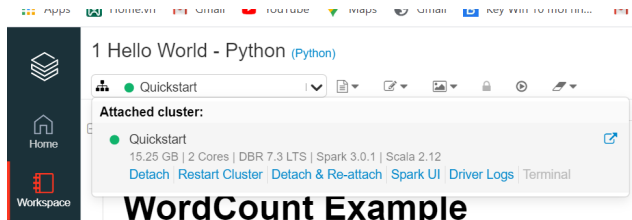
cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bfcf/4954928053318020/3029198689315822/167703932442645/latest.html

Go to your databricks page, Click workspace and then click the  down arrow next to workspace, and select **Import**.

Tick the **URL** box and paste the link above. Select **Import**



Every time you want to run a notebook, make sure it is attached to the quickstart cluster. The quickstart cluster works with Python 3. If you need a specific cluster, you can create another one with different specification.




WordCount Example

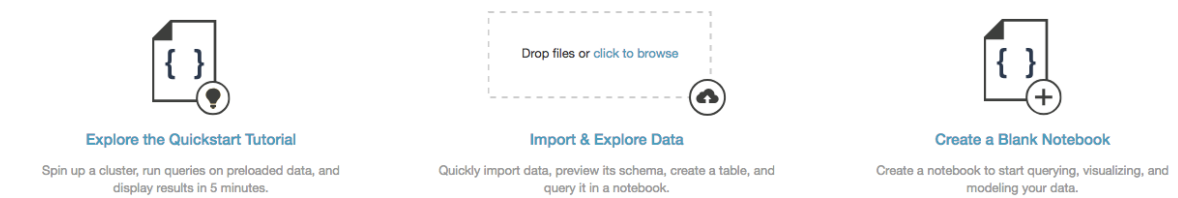
Feel free to explore this notebook however you want.

Import a dataset

Databricks does have a lot of sample datasets to work with, but sometimes you might have to put in your own dataset.



Click on  to return to the main page. The simplest way is to click on import & explore data on the page.



Files imported to DBFS using these methods are stored in FileStore.

For production environments, we recommend that you explicitly upload files into DBFS using the DBFS CLI, DBFS API, Databricks file system utilities (dbutils.fs). For more complicated file upload, refer to: <https://docs.databricks.com/data/data.html>

When you have a file uploaded, you can either access it using Spark API or local API such as pandas. If you upload a file to databricks, you can access the file using sample code as follows:

Sample code – Python spark

```
sparkDF = spark.read.csv('/FileStore/tables/state_income-9f7c5.csv', header="true",
inferSchema="true")
```

Sample Code – R Spark

```
sparkDF <- read.df(source = "csv", path = "/FileStore/tables/state_income-9f7c5.csv",
header="true", inferSchema = "true")
```

Sample Code – Scala Spark

```
val sparkDF = spark.read.format("csv")
.option("header", "true")
.option("inferSchema", "true")
.load("/FileStore/tables/state_income-9f7c5.csv")
```

Sample code – local APIs python:

```
pandas_df = pd.read_csv("/dbfs/FileStore/tables/state_income-9f7c5.csv", header='infer')
```

Sample Code – local APIs R:

```
df = read.csv("/dbfs/FileStore/tables/state_income-9f7c5.csv", header = TRUE)
```