

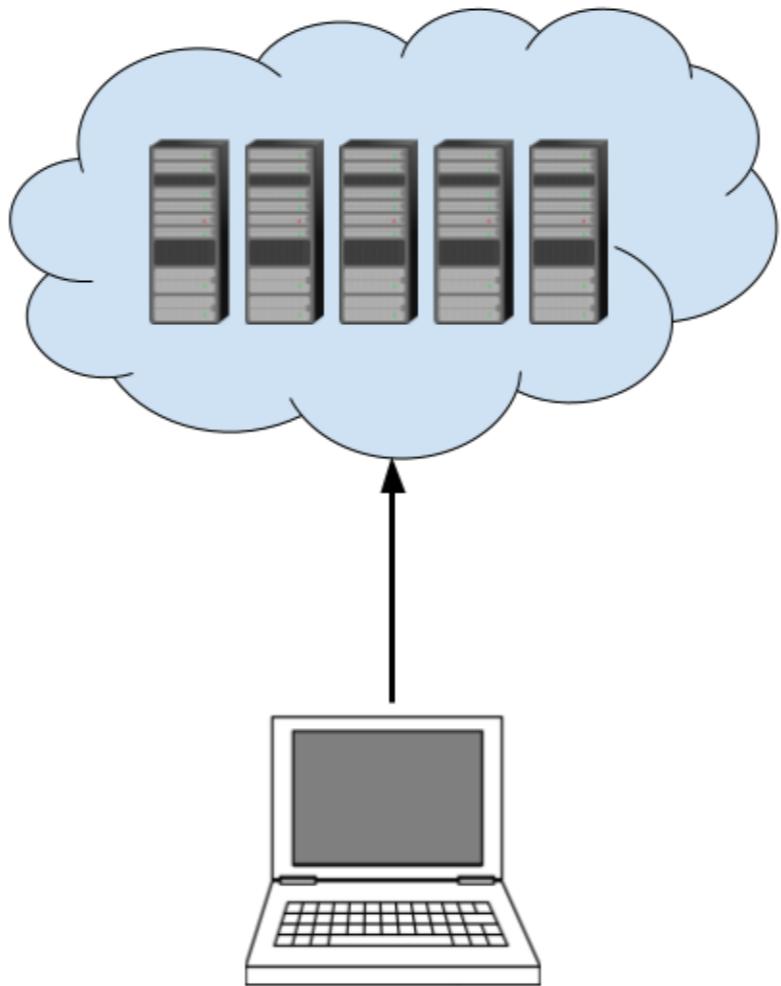
# Getting started with Databricks

INTRODUCTION TO DATABRICKS



**Kevin Barlow**  
Data Practitioner

# Compute cluster refresh



# Create your first cluster

The first step is to create a cluster for your data processing!

## ***Configuration options:***

### Sample Cluster

#### Policy

Unrestricted

Multi node  Single node

#### Access mode

Single user access 

Single user

kevin.curtis.barlow@gmail.com

### Performance

#### Databricks runtime version

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)

Use Photon Acceleration 

#### Worker type

Standard\_DS3\_v2

14 GB Memory, 4 Cores

Min workers

Max workers

1

1

Spot instances

#### Driver type

Same as worker

14 GB Memory, 4 Cores

Enable autoscaling 

Terminate after  minutes of inactivity 

### Tags

Add tags

Key

Value

Add

# Create your first cluster

The first step is to create a cluster for your data processing!

## ***Configuration options:***

- Cluster policies and access

Sample Cluster [Edit](#)

**Policy** [?](#)

Unrestricted

Multi node  Single node

Access mode [?](#) Single user access [?](#)

Single user kevin.curtis.barlow@gmail.com

**Performance**

Databricks runtime version [?](#)

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)

Use Photon Acceleration [?](#)

**Worker type** [?](#)

	Min workers	Max workers	Spot instances
Standard_DS3_v2	14 GB Memory, 4 Cores	1	1

**Driver type**

Same as worker	14 GB Memory, 4 Cores
----------------	-----------------------

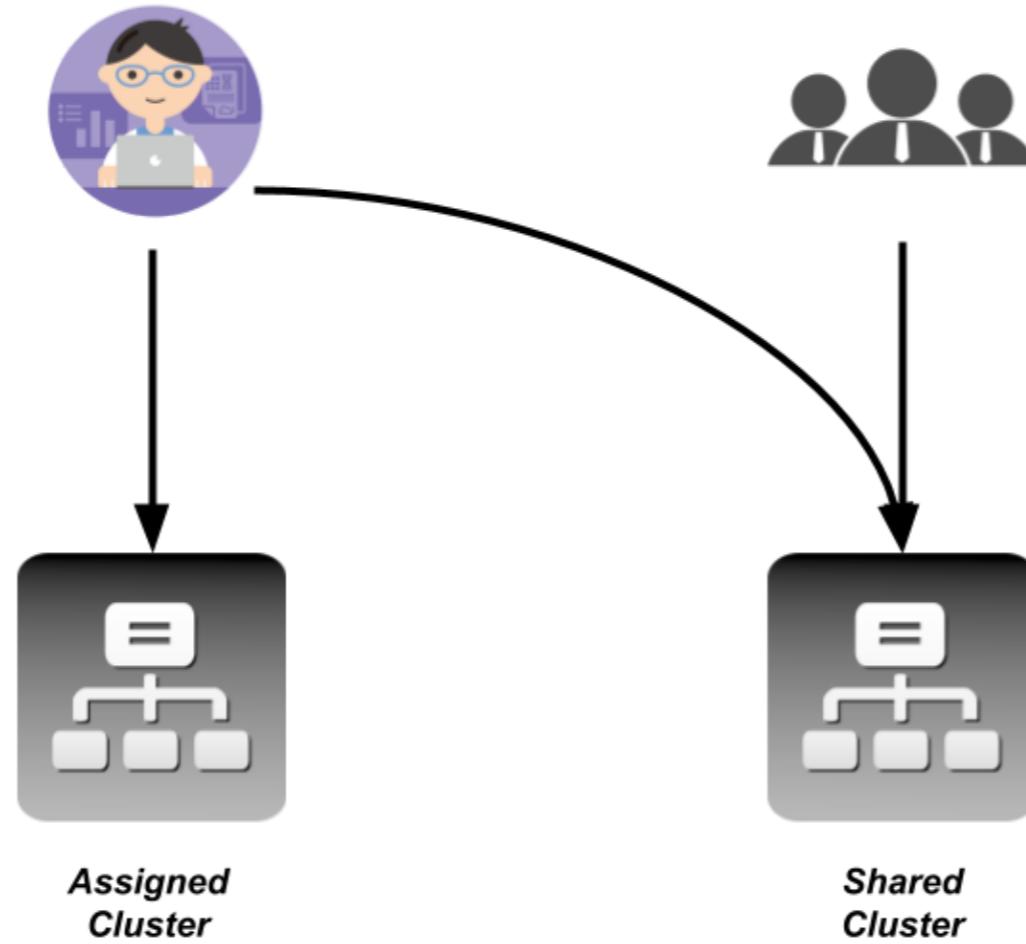
Enable autoscaling [?](#)  
 Terminate after  minutes of inactivity [?](#)

**Tags** [?](#)

Add tags

Key	Value	Add
-----	-------	-----

# Cluster Access



# Create your first cluster

The first step is to create a cluster for your data processing!

## ***Configuration options:***

- Cluster policies and access
- Databricks Runtime
- Photon Acceleration

Sample Cluster 

Policy  Unrestricted 

Multi node  Single node

Access mode  Single user access 

Single user  kevin.curtis.barlow@gmail.com 

Performance

Databricks runtime version 

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2) 

Use Photon Acceleration 

Worker type 

Standard\_DS3\_v2  14 GB Memory, 4 Cores  1  Spot instances

Min workers Max workers

Driver type

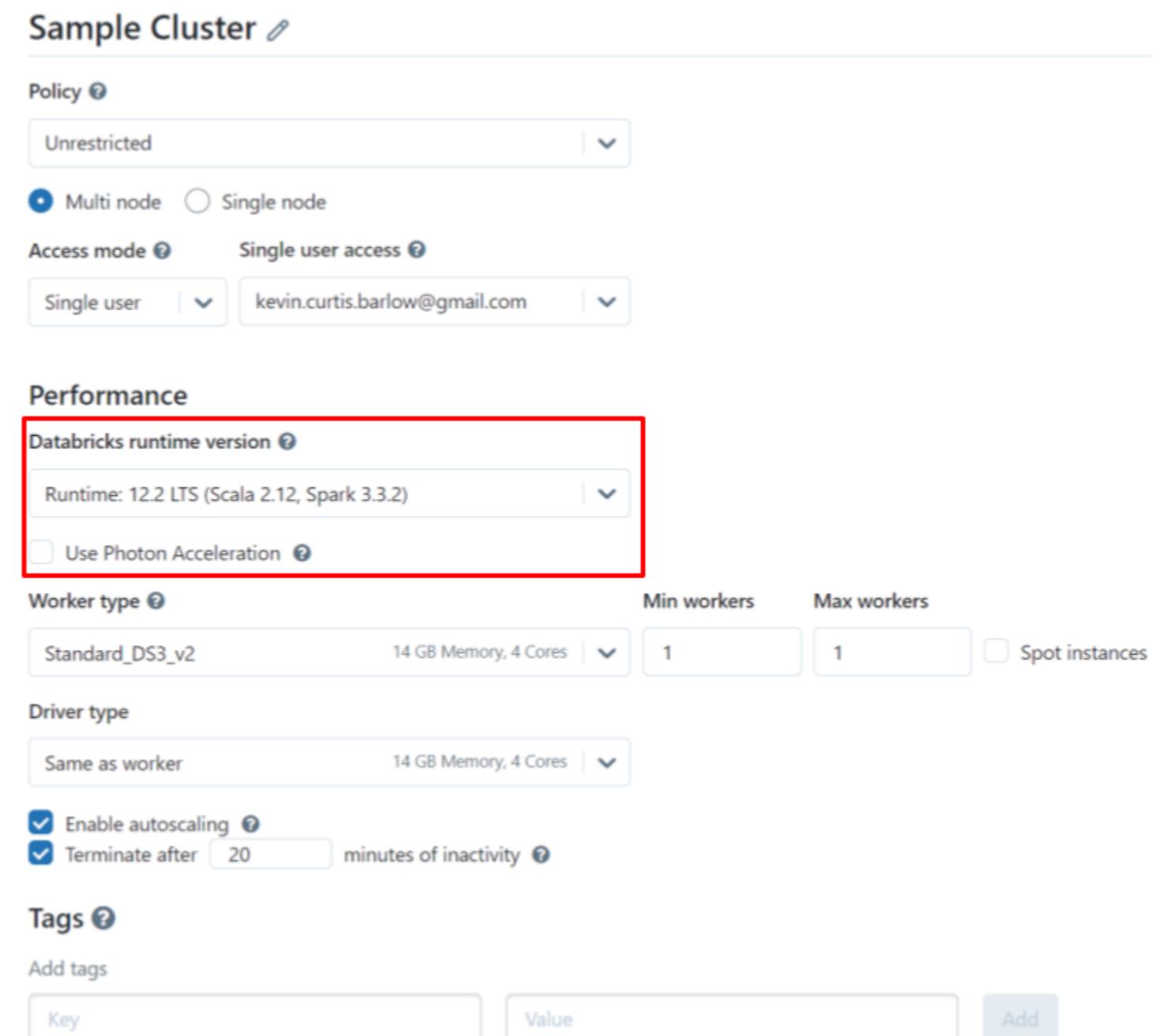
Same as worker  14 GB Memory, 4 Cores 

Enable autoscaling   
 Terminate after  minutes of inactivity 

Tags 

Add tags

Key  Value  Add



# Create your first cluster

The first step is to create a cluster for your data processing!

## ***Configuration options:***

- Cluster policies and access
- Databricks Runtime
- Photon Acceleration
- Node instance types and number
- Auto-scaling / Auto-termination

Sample Cluster 

Policy  Unrestricted 

Multi node  Single node

Access mode  Single user access 

Single user  kevin.curtis.barlow@gmail.com 

Performance

Databricks runtime version  Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2) 

Use Photon Acceleration 

Worker type 	Min workers	Max workers	<input type="checkbox"/> Spot instances
Standard_DS3_v2  14 GB Memory, 4 Cores 	1	1	<input type="checkbox"/>

Driver type

Same as worker  14 GB Memory, 4 Cores 

Enable autoscaling   
 Terminate after  minutes of inactivity 

Tags 

Add tags

Key	Value	Add
-----	-------	-----

# Data Explorer

Get familiar with the Data Explorer! In this UI, you can:

1. Browse available catalogs/schemas/tables
2. Look at sample data and summary statistics
3. View data lineage and history

You can also upload new data by clicking the "plus" icon!

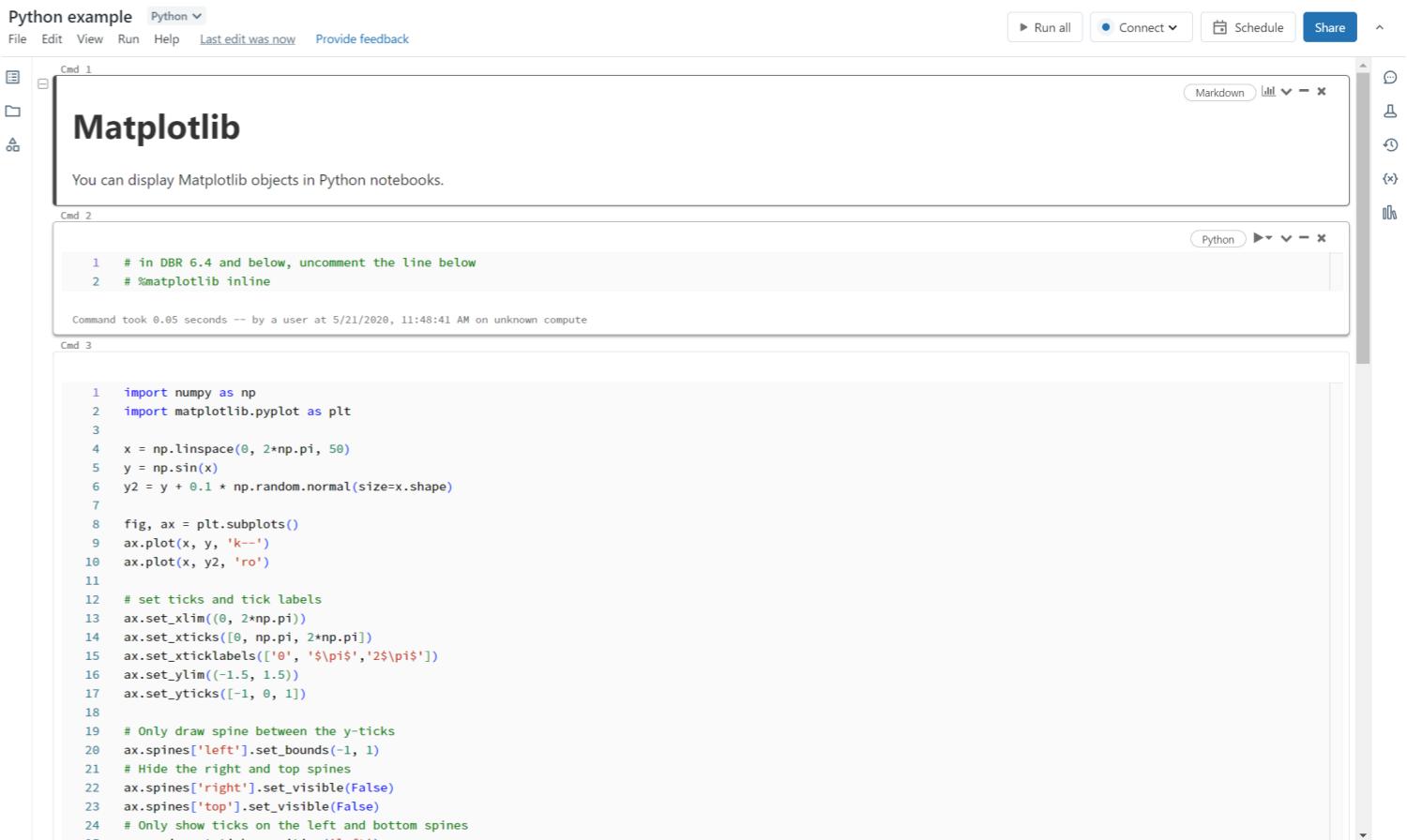


<sup>1</sup> Photo by Jakub Zerdzicki: <https://www.pexels.com/photo/magnifier-loupe-17284804/>

# Create a notebook

## Databricks notebooks:

- Standard interface for Databricks
- Improvements on open-source Jupyter
- Support for many languages
  - Python, R, Scala, SQL
  - Magic commands (%sql)
- Built-in visualizations
- Real-time commenting and collaboration



The screenshot shows a Databricks notebook titled "Python example" in Python mode. The interface includes a top navigation bar with "Run all", "Connect", "Schedule", and "Share" buttons. Below the title, there's a section titled "Matplotlib" with the subtext "You can display Matplotlib objects in Python notebooks." In the code editor, three cells are visible:

- Cell 1: A header cell containing the text "Matplotlib".
- Cell 2: A code cell containing the command "# in DBR 6.4 and below, uncomment the line below" followed by "# %matplotlib inline". It also includes a timestamp: "Command took 0.05 seconds -- by a user at 5/21/2020, 11:48:41 AM on unknown compute".
- Cell 3: A code cell containing Python code for generating a plot. The code imports numpy and matplotlib.pyplot, generates data points, creates a plot with two lines, and sets specific tick and spine properties.

```
1 # in DBR 6.4 and below, uncomment the line below
2 # %matplotlib inline

Command took 0.05 seconds -- by a user at 5/21/2020, 11:48:41 AM on unknown compute

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0, 2*np.pi, 50)
5 y = np.sin(x)
6 y2 = y + 0.1 * np.random.normal(size=x.shape)
7
8 fig, ax = plt.subplots()
9 ax.plot(x, y, 'k--')
10 ax.plot(x, y2, 'ro')
11
12 # set ticks and tick labels
13 ax.set_xlim([0, 2*np.pi])
14 ax.set_xticks([0, np.pi, 2*np.pi])
15 ax.set_xticklabels(['0', '$\\pi$', '2$\\pi$'])
16 ax.set_ylim([-1.5, 1.5])
17 ax.set_yticks([-1, 0, 1])
18
19 # Only draw spine between the y-ticks
20 ax.spines['left'].set_bounds(-1, 1)
21 # Hide the right and top spines
22 ax.spines['right'].set_visible(False)
23 ax.spines['top'].set_visible(False)
24 # Only show ticks on the left and bottom spines
```

# **Let's practice!**

## **INTRODUCTION TO DATABRICKS**

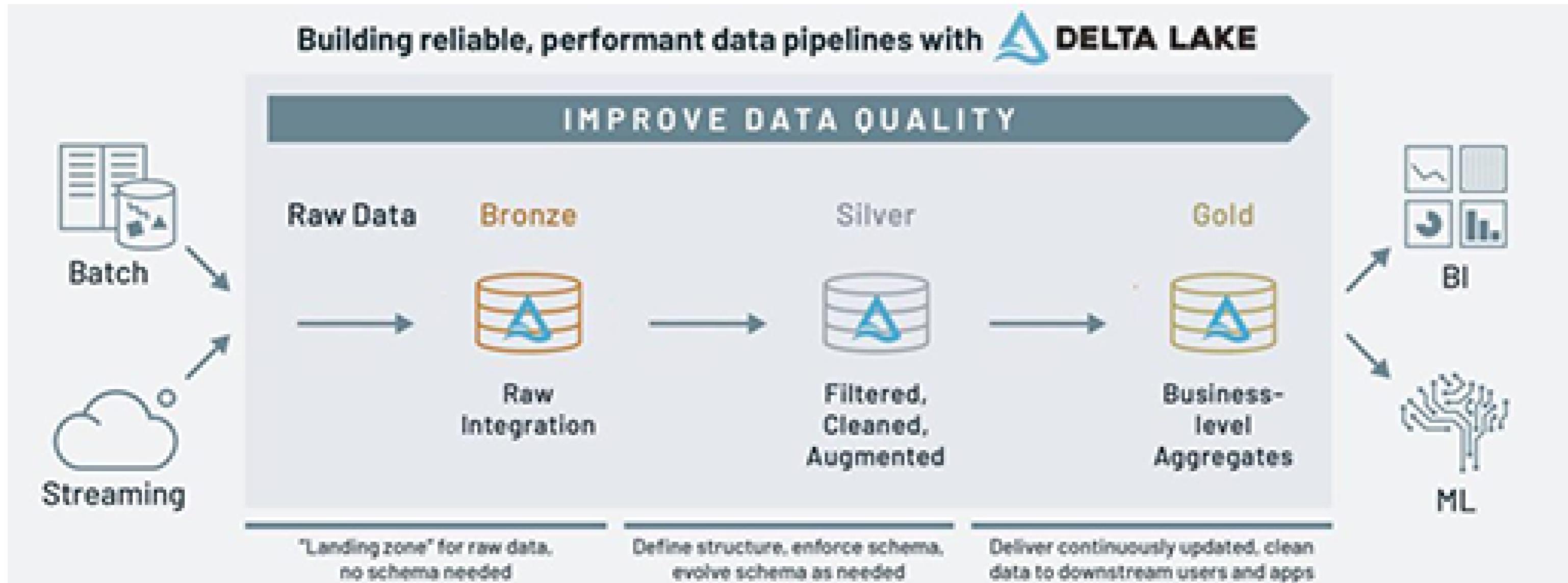
# Data Engineering foundations in Databricks

INTRODUCTION TO DATABRICKS



**Kevin Barlow**  
Data Practitioner

# Medallion architecture

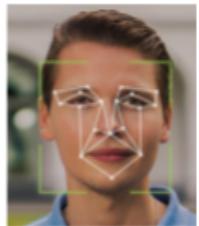


# Reading data

Spark is a highly flexible framework and can read from various data sources/types.

## *Common data sources and types:*

- Delta tables
- File formats (CSV, JSON, Parquet, XML)
- Databases (MySQL, Postgres, EDW)
- Streaming data
- Images / Videos



# Reading data

Spark is a highly flexible framework and can read from various data sources/types.

## ***Common data sources and types:***

- Delta tables
- File formats (CSV, JSON, Parquet, XML)
- Databases (MySQL, Postgres, EDW)
- Streaming data
- Images / Videos

```
#Delta table  
spark.read.table()  
  
#CSV files  
spark.read.format('csv').load('*.*csv')  
  
#Postgres table  
spark.read.format("jdbc")  
    .option("driver", driver)  
    .option("url", url)  
    .option("dbtable", table)  
    .option("user", user)  
    .option("password", password)  
    .load()
```

# Structure of a Delta table

A Delta table provides table-like qualities to an open file format.

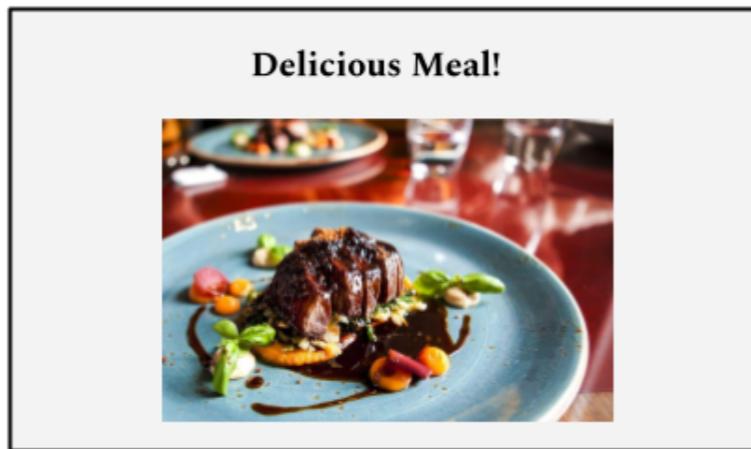
- Feels like a table when reading
- Access to underlying files (Parquet and JSON)

```
1 %fs ls /databricks-datasets/nyctaxi-with-zipcodes/subsampled
```

Table ▾ +

	path	name
1	dbfs:/databricks-datasets/nyctaxi-with-zipcodes/subsampled/_delta_log/	_delta_log/
2	dbfs:/databricks-datasets/nyctaxi-with-zipcodes/subsampled/nyc-zips-dataset-readme.txt	nyc-zips-dataset-readme.txt
3	dbfs:/databricks-datasets/nyctaxi-with-zipcodes/subsampled/part-00000-80b68cae-ce6a-41cf-87cd-2573d91b4c07-c000.snappy.parquet	part-00000-80b68cae-ce6a-41cf-87cd-2573d91b4c07-c000.snappy.parquet
4	dbfs:/databricks-datasets/nyctaxi-with-zipcodes/subsampled/part-00001-c883942d-366f-478a-be3b-f13fd4bee0ab-c000.snappy.parquet	part-00001-c883942d-366f-478a-be3b-f13fd4bee0ab-c000.snappy.parquet
5	dbfs:/databricks-datasets/nyctaxi-with-zipcodes/subsampled/part-00002-bbf9fd81-4b3a-46f3-943e-841b48ae743e-c000.snappy.parquet	part-00002-bbf9fd81-4b3a-46f3-943e-841b48ae743e-c000.snappy.parquet
6	dbfs:/databricks-datasets/nyctaxi-with-zipcodes/subsampled/part-00003-3d80435e-15f8-4154-92c7-515307e41c1b-c000.snappy.parquet	part-00003-3d80435e-15f8-4154-92c7-515307e41c1b-c000.snappy.parquet

# Explaining the Delta Lake structure



Ingredients	Steps
<ul style="list-style-type: none"><li>• Ribeye Steak</li><li>• Carrots</li><li>• Red Wine</li><li>• .....</li></ul>	<ol style="list-style-type: none"><li>1. Preheat oven</li><li>2. Mise en place</li><li>3. Sear meat</li><li>4. .....</li></ol>

# DataFrames

*DataFrames* are two-dimensional representations of data.

- Look and feel similar to tables
- Similar concept for many different data tools
  - Spark (default), pandas, dplyr, SQL queries
- Underlying construct for most data processes

<b>id</b>	<b>customerName</b>	<b>bookTitle</b>
1	John Data	Guide to Spark
2	Sally Bricks	SQL for Data Engineering
3	Adam Delta	Keeping Data Clean

```
df = (spark.read  
      .format("csv")  
      .option("header", "true")  
      .option("inferSchema", "true")  
      .load("/data.csv"))
```

# Writing data

## *Kinds of tables in Databricks*

### 1. Managed tables

- Default type
- Stored with Unity Catalog
- Databricks managed

### 2. External tables

- Stored in another location
- Set LOCATION
- Customer managed

```
df.write.saveAsTable(table_name)
```

```
CREATE TABLE table_name  
USING delta  
AS ...
```

```
df.write  
.location('').saveAsTable(table_name)
```

```
CREATE TABLE table_name  
USING delta  
LOCATION "<path>"  
AS ...
```

# **Let's practice!**

## INTRODUCTION TO DATABRICKS

# Data transformations in Databricks

INTRODUCTION TO DATABRICKS



**Kevin Barlow**  
Data Practitioner

# SQL for data engineering

## SQL

- Familiar for Database Administrators (DBAs)
- Great for standard manipulations
- Execute pre-defined UDFs

```
-- Creating a new table in SQL
```

```
CREATE TABLE table_name  
USING delta  
AS (  
    SELECT *  
    FROM source_table  
    WHERE date >= '2023-01-01'  
)
```

# Other languages for data engineering

## *Python, R, Scala*

- Familiar for software engineers
- Standard and complex transformations
- Use and define custom functions

```
#Creating a new table in Pyspark  
  
spark  
    .read  
        .table('source_table')  
        .filter(col('date') >= '2023-01-01')  
    .write  
        .saveAsTable('table_name')
```

# Common transformations

## *Schema manipulation*

- Add and remove columns
- Redefine columns

## #Pyspark

```
df  
  .withColumn(col('newCol'), ...)  
  .drop(col('oldCol'))
```

## *Filtering*

- Reduce DataFrame to subset of data
- Pass multiple criteria

## #Pyspark

```
df  
  .filter(col('date') >= target_date)  
  .filter(col('id') IS NOT NULL)
```

# Common transformations (continued)

## *Nested data*

- Arrays or Struct data
- Expand or contract

```
df
```

```
.explode(col('arrayCol')) #wide to long  
.flatten(col('items')) #long to wide
```

## *Aggregation*

- Group data based on columns
- Calculate data summarizations

```
df
```

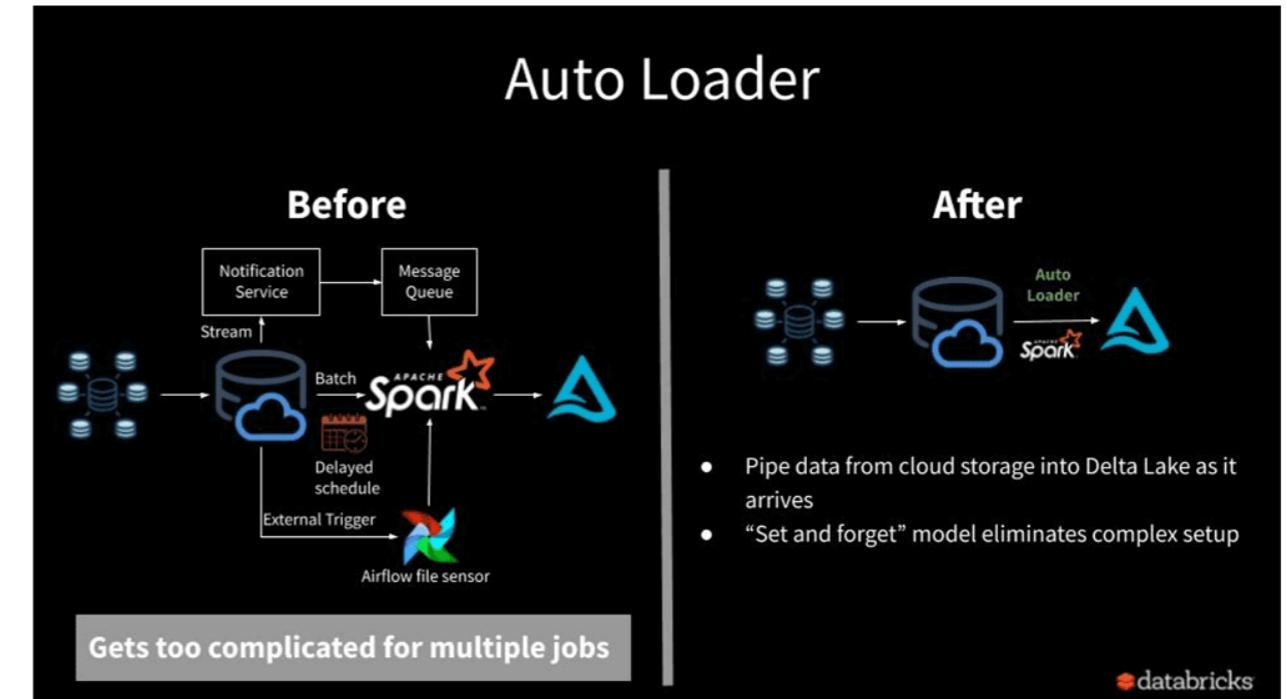
```
.groupBy(col('region'))  
.agg(sum(col('sales')))
```

# Auto Loader

Auto Loader processes new data files as they land in a data lake.

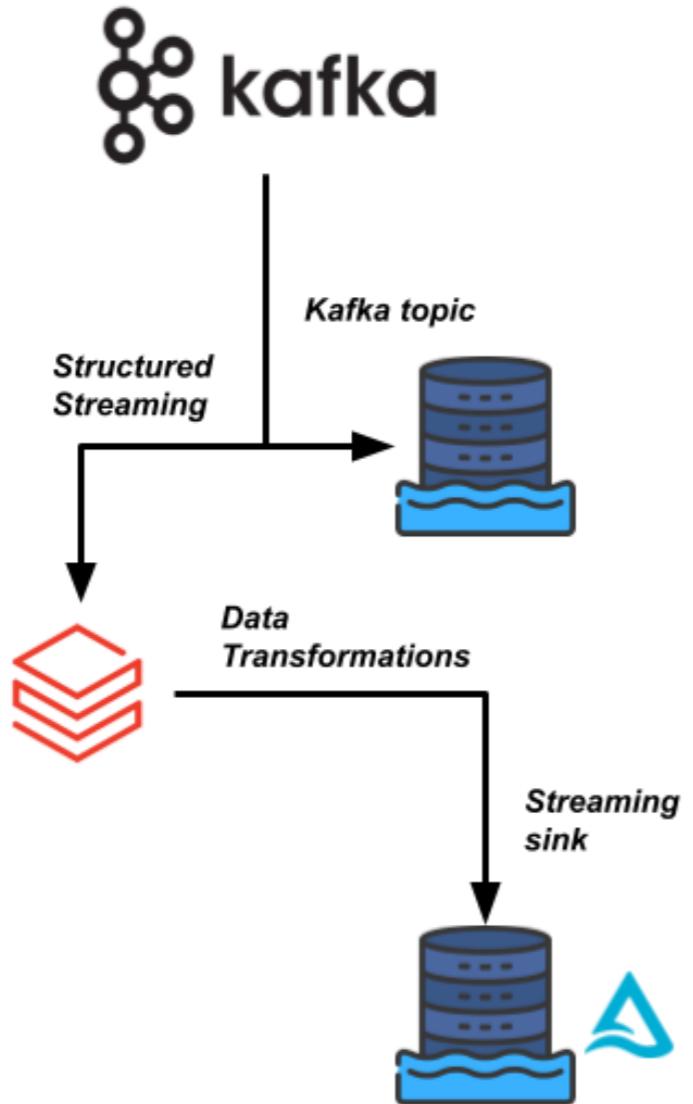
- Incremental processing
- Efficient processing
- Automatic

```
spark.readStream  
  .format("cloudFiles")  
  .option("cloudFiles.format", "json")  
  .load(file_path)
```



<sup>1</sup> <https://www.databricks.com/blog/2020/02/24/introducing-databricks-ingest-easy-data-ingestion-into-delta-lake.html>

# Structured Streaming



```
spark.readStream  
  .format("kafka")  
  .option("subscribe", "<topic>")  
  .load()  
  .join(table_df,  
        on=<id>, how="left")  
  .writeStream  
  .format("kafka")  
  .option("topic", "<topic>")  
  .start()
```

# **Let's practice!**

## INTRODUCTION TO DATABRICKS

# Orchestration in Databricks

INTRODUCTION TO DATABRICKS



**Kevin Barlow**

Data Analytics Practitioner

# What is data orchestration?

- Data orchestration is a form of automation!

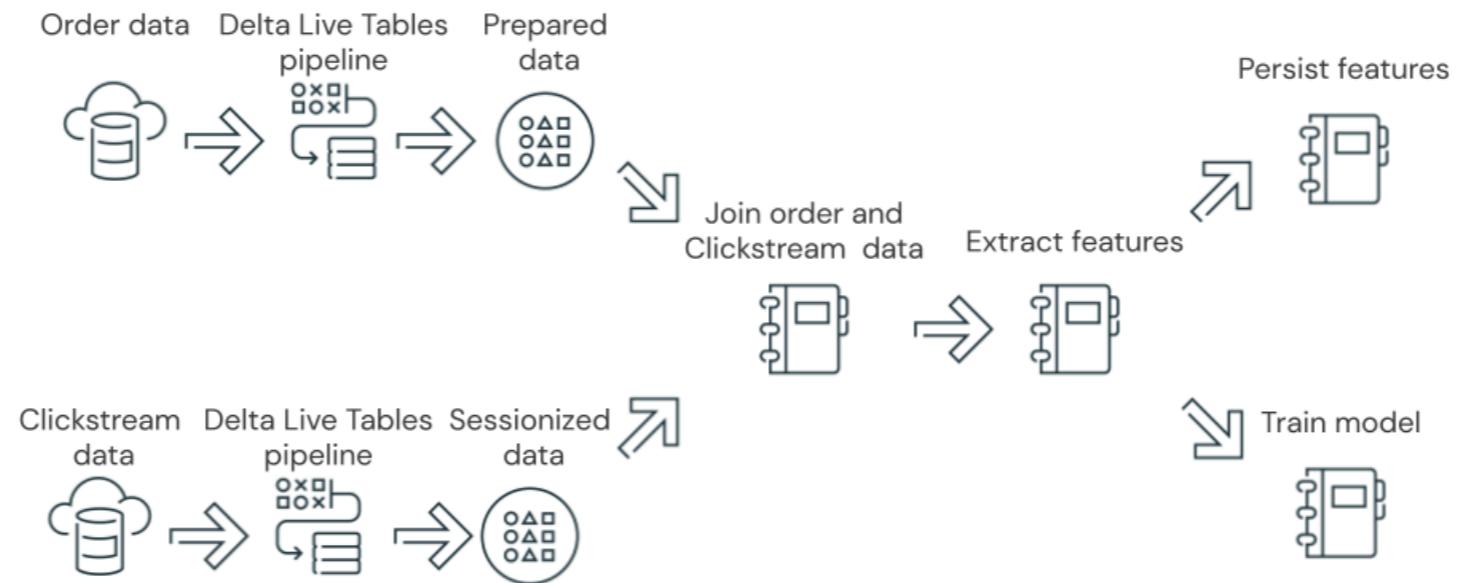


- Enables data engineers to automate the end-to-end data life cycle

# Databricks Workflows

**Databricks Workflows** is a collection of built-in capabilities to orchestrate all your data processes, at no additional cost!

## Example Databricks Workflow



<sup>1</sup> <https://docs.databricks.com/workflows>

# What can we orchestrate?

Data engineers/data scientists

Data analysts



Databricks Notebook



Delta Live Tables



# Databricks Jobs

## Workflows UI

Users can create jobs directly from the Databricks UI:

- Directly from a notebook
- In the Workflows section

The screenshot shows the Databricks Workflows UI for creating a new job. The form includes the following fields:

- Job name:** Scheduled job example
- Schedule:** Manual (radio button) is selected, but Scheduled is also available. The schedule is set to "Every Day at 16 : 00 (UTC-07:00) Pacific Ti...".
- Cluster:** Add new job cluster (button) - 274.5 GB · 36 Cores · DBR 12.2 LTS · Spark 3.3.2 · Scala 2.12
- Parameters:** + Add (button)
- Alerts:** A dropdown menu for alerts, with "Failure" checked and "Start", "Success", and "Add" also listed.

At the bottom right are two buttons: "Cancel" and "Create".

<sup>1</sup> <https://docs.databricks.com/workflows/jobs>

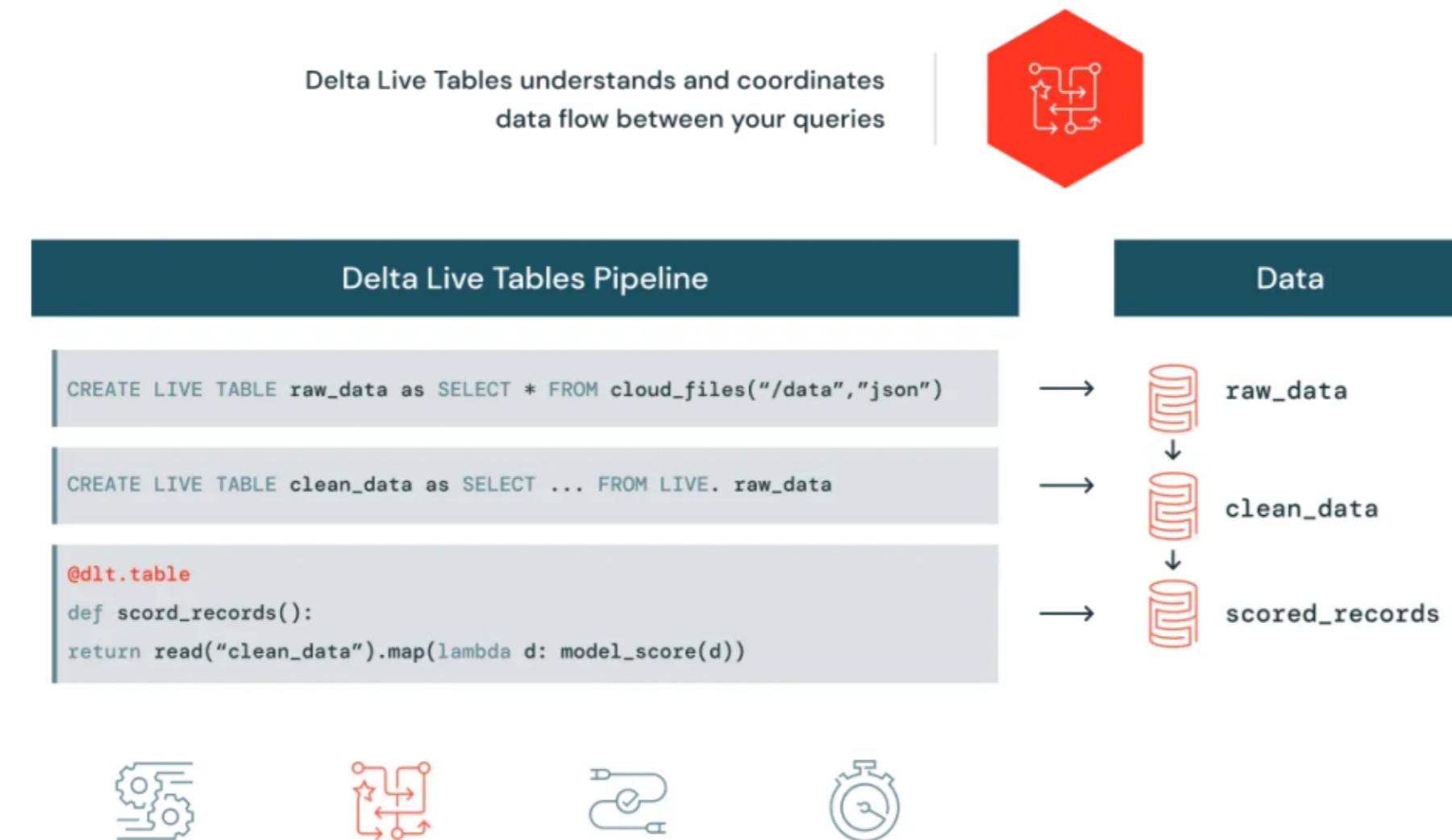
# Databricks Jobs

## Programmatic

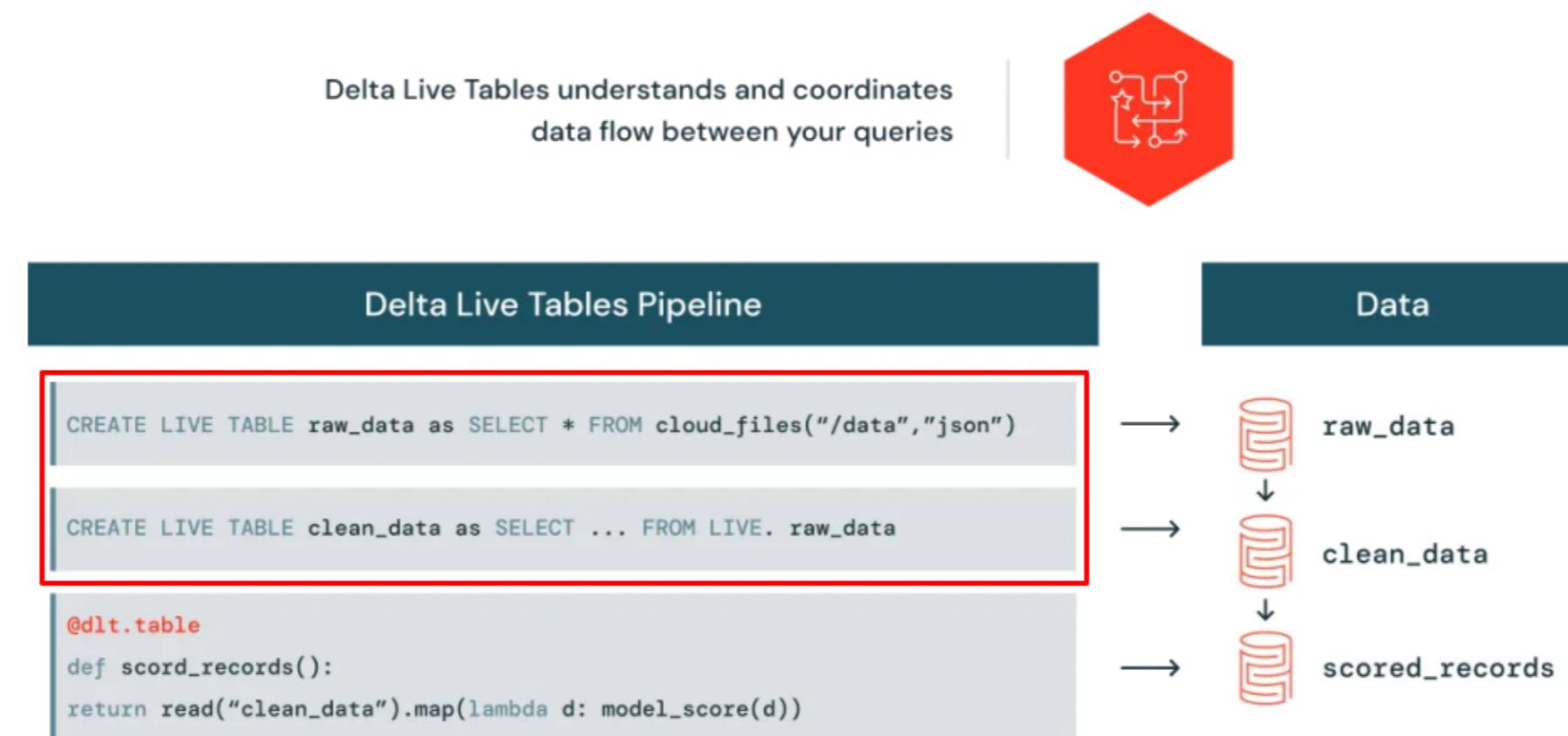
Users can also programmatically create jobs using the Jobs CLI or Jobs API with the Databricks platform.

```
{  
  "name": "A multitask job",  
  "tags": {},  
  "tasks": [],  
  "job_clusters": [],  
  "format": "MULTI_TASK",  
}
```

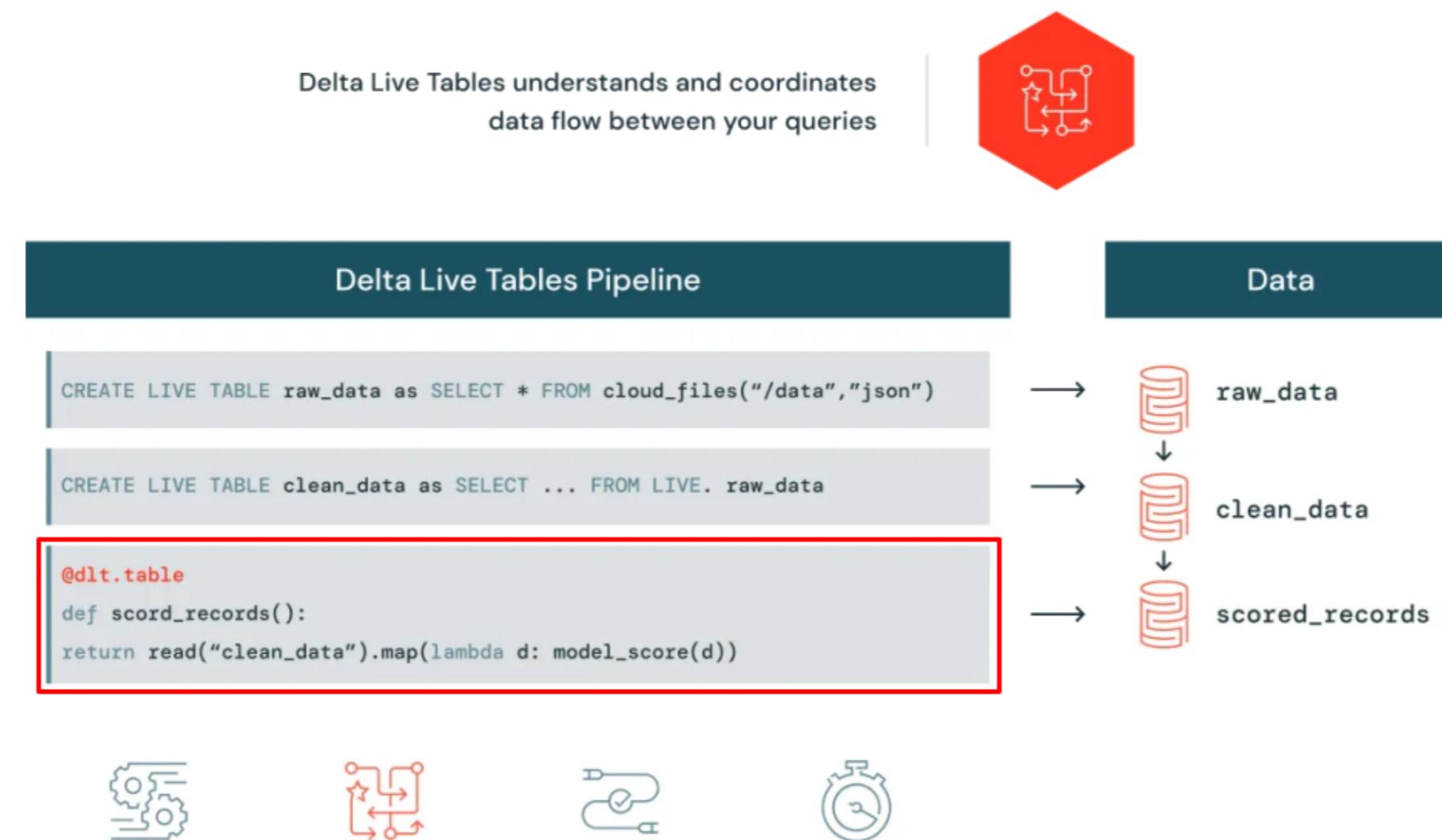
# Delta Live Tables



# Delta Live Tables



# Delta Live Tables



# **Let's practice!**

## **INTRODUCTION TO DATABRICKS**

# End-to-end data pipeline example in Databricks

INTRODUCTION TO DATABRICKS



**Kevin Barlow**  
Data Practitioner

# **Let's practice!**

## INTRODUCTION TO DATABRICKS