

Getting started with Databricks

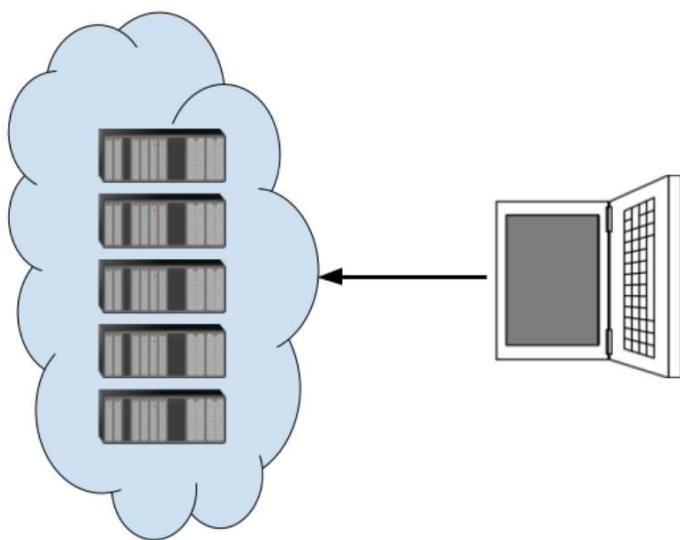
DATABRICKS CONCEPTS



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 1

Max workers 1

Spot instances

Driver type

Same as worker

14 GB Memory, 4 Cores

Enable autoscaling

Terminate after 20 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

Policy

Unrestricted

Multi node Single node

Access mode

Single user kevin.curtis.barlow@gmail.com

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)

Use Photon Acceleration

Worker type

Standard_DS3_v2 14 GB Memory, 4 Cores

Driver type

Same as worker 14 GB Memory, 4 Cores

Enable autoscaling
 Terminate after 20 minutes of inactivity

Tags

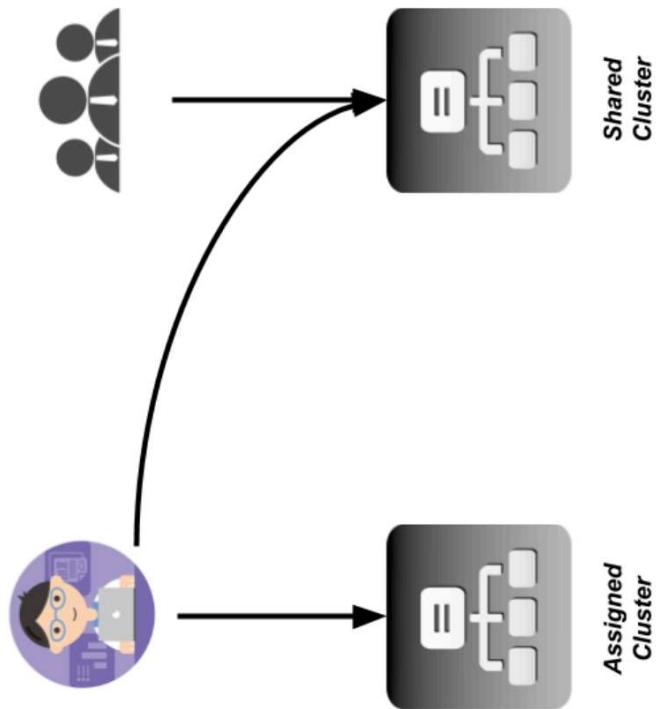
Add tags

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

The screenshot shows the 'Sample Cluster' configuration page. It includes sections for Policy (Unrestricted), Access mode (Multi node selected), and Driver type (Standard_DS3_v2). The 'Performance' section is highlighted with a red box, containing options for Databricks runtime version (12.2 LTS (Scala 2.12, Spark 3.3.2) selected), Worker type (Standard_DS3_v2 selected), and Driver type (Same as worker). The 'Tags' section shows two tags: 'Enable autoscaling' (checked) and 'Terminate after 20 minutes of inactivity'. A 'Tags' button is also present.

Sample Cluster

Policy

Unrestricted

Access mode

Multi node Single node

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

Driver type

Same as worker 14 GB Memory, 4 Cores

Enable autoscaling

Terminate after 20 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 Standard_DS3_v2 14 GB Memory, 4 Cores Min workers 1 Max workers 1 Spot instances

Driver type

Same as worker 14 GB Memory, 4 Cores

Enable autoscaling Terminate after 20 minutes of inactivity

Tags Add tags Key Value Add

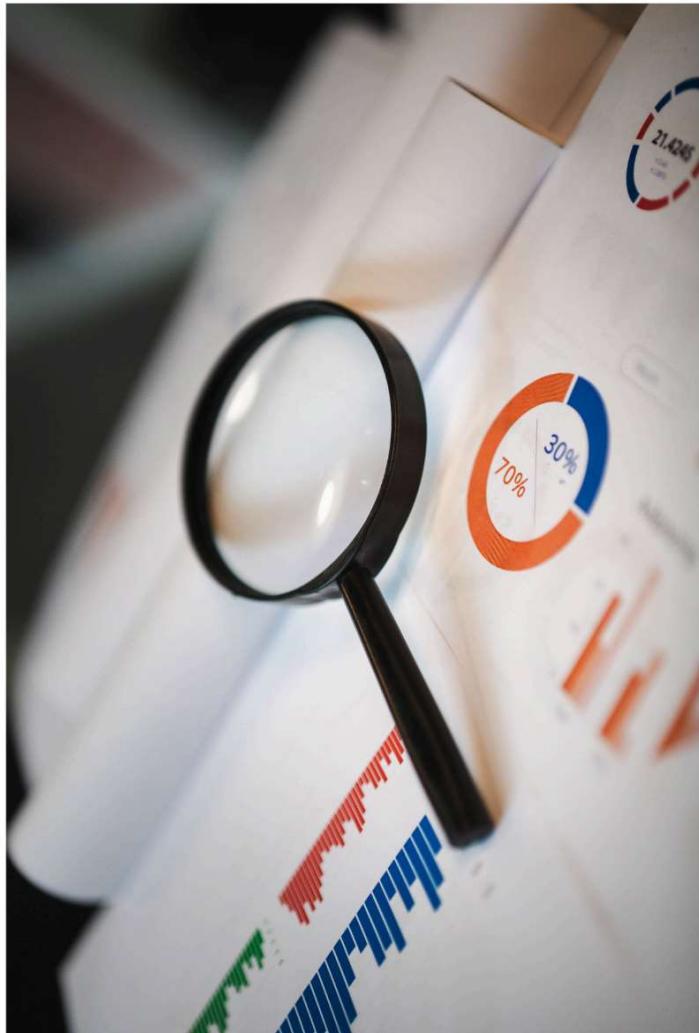


Data Explorer

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!



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

Create a notebook

Databricks notebooks:

The screenshot shows a Databricks notebook interface with three code cells and one visualization cell.

Cell 1: A title cell containing the text "Matplotlib".

Cell 2: Python code demonstrating Matplotlib objects in Python notebooks.

```

1 # In DMR 6.4 and below, uncomment the line below
2 # %matplotlib inline

```

Cell 3: Python code showing a sine wave plot.

```

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 = x + 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, 'r-')
11
12 # set ticks and tick labels
13 ax.set_xticks([0, 2*np.pi])
14 ax.set_xticklabels(['0', '2\pi'])
15 ax.set_yticks([-1.5, 1.5])
16 ax.set_ylabel('y')
17 ax.set_yticks([-1, 0, 1])
18
19 # only raw 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
25 # only show ticks on the left and bottom spines
26 ...

```

Output: A plot titled "Matplotlib" showing a blue sine wave and a red noisy line segment.

- 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

Let's practice!

DATABRICKS CONCEPTS

Data Engineering foundations in Databricks

DATABRICKS CONCEPTS



Kevin Barlow
Data Practitioner



Medallion architecture

Building reliable, performant data pipelines with  DELTA LAKE



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)
 - option("driver", driver)
 - option("url", url)
 - option("dbtable", table)
 - option("user", user)
 - option("password", password)
- 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.

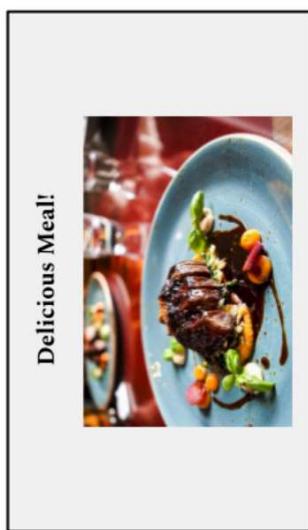
- Feels like a table when reading
- Access to underlying files (Parquet and JSON)

```
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/nyctaxi-with-zipcodes-subsampled/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
Ribeye Steak	1. Preheat oven
Carrots	2. Mise en place
Red Wine	3. Sear meat
.....	4.

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

id **customerName** **bookTitle**

id	customerName	bookTitle
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!

DATABRICKS CONCEPTS

Data transformations in Databricks

DATABRICKS CONCEPTS



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')))
```

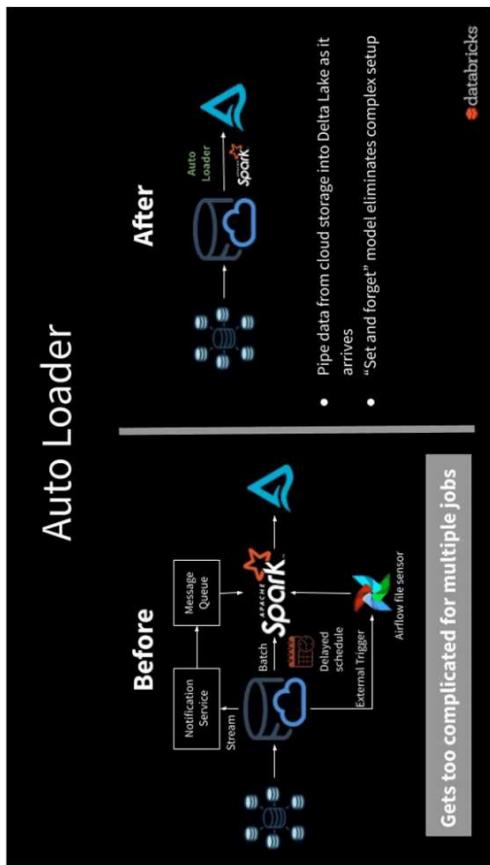
Auto Loader

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

- Incremental processing
- Efficient processing
- Automatic

```
spark.readStream
```

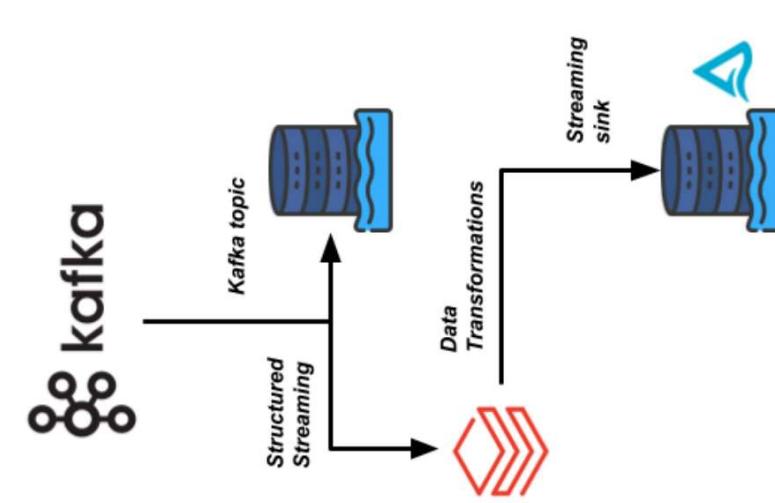
```
.format("cloudFiles")
.option("cloudFiles.format", "json")
.load(file_path)
```



¹ <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!

DATABRICKS CONCEPTS

Orchestration in Databricks

DATABRICKS CONCEPTS

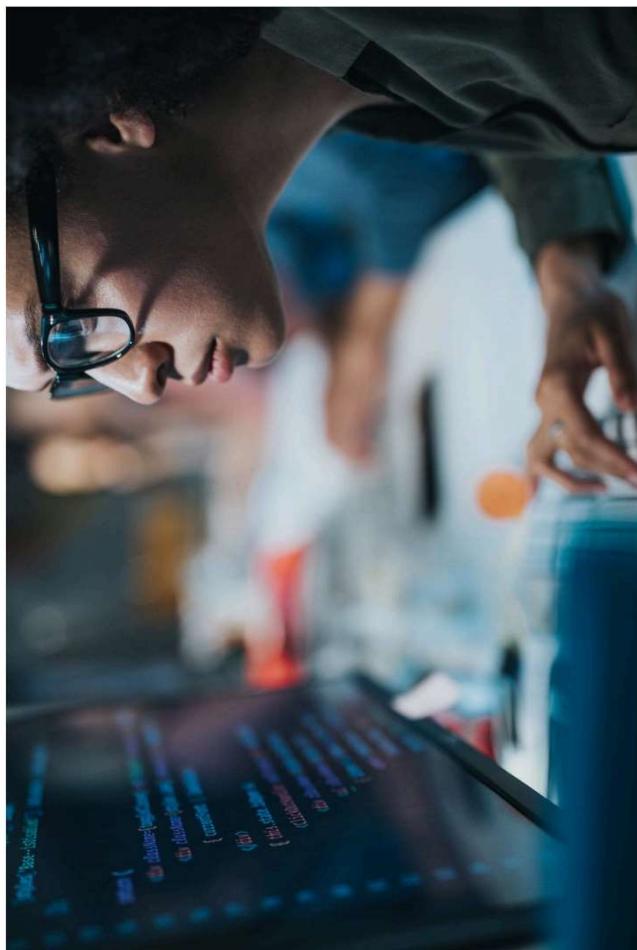


Kevin Barlow
Data Analytics Practitioner



What is data orchestration?

- Data orchestration is a form of automation!

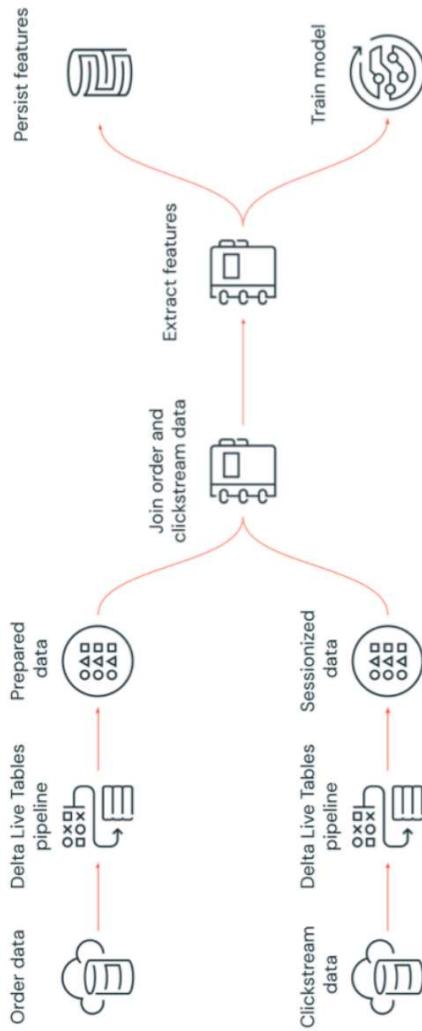


- Enables data engineers to automate the end-to-end data life cycle

Databricks Workflows

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

Example Databricks Workflow



¹ <https://docs.databricks.com/pt/courses/databricks-concepts/data-engineering?ex=2>

What can we orchestrate?

Data engineers/data scientists

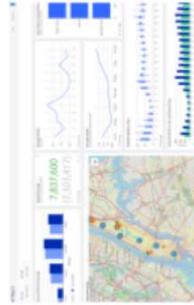
Data analysts



Databricks Notebook



Delta Live Tables



DATABRICKS CONCEPTS

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 Jobs creation interface. It includes fields for:

- Job name: docs example
- Schedule: Manual (radio button selected), Scheduled (radio button unselected). Sub-fields: Every Day at 15 : 07 (UTC-08:00) Pacific Ti... (dropdown menu)
- Compute: New serverless job compute Serverless (dropdown menu)
- Parameters: + Add (button)
- Alerts: Email address (input field), Start (checkbox), Success (checkbox), Failure (checkbox) (selected), X (button)

At the bottom right are 'Cancel' and 'Create' buttons.

¹ <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 understands and coordinates
data flow between your queries

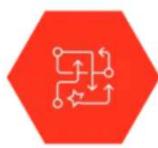


Delta Live Tables Pipeline

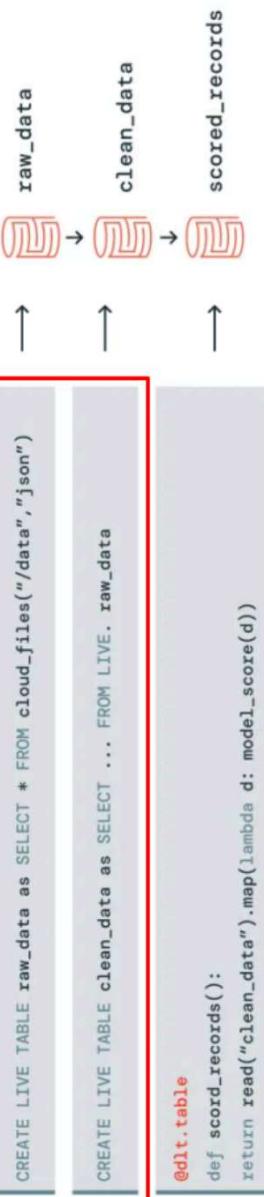


Delta Live Tables

Delta Live Tables understands and coordinates
data flow between your queries



Delta Live Tables Pipeline

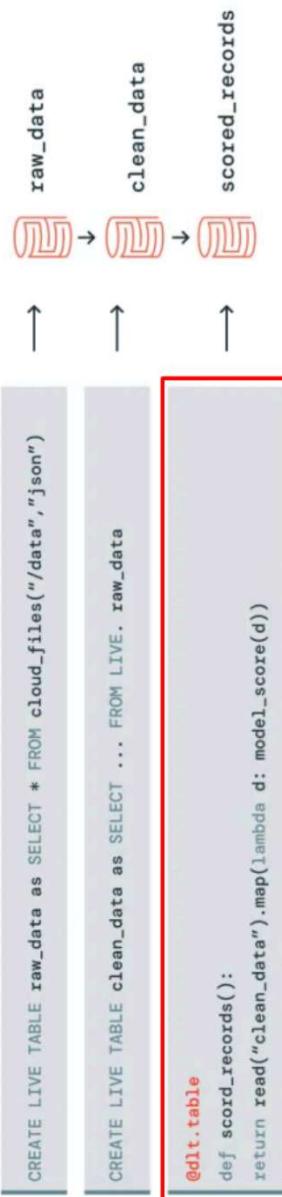


Delta Live Tables

Delta Live Tables understands and coordinates
data flow between your queries



Delta Live Tables Pipeline



Let's practice!

DATABRICKS CONCEPTS

End-to-end data pipeline example in Databricks

DATABRICKS CONCEPTS



Kevin Barlow
Data Practitioner



Let's practice!

DATABRICKS CONCEPTS

