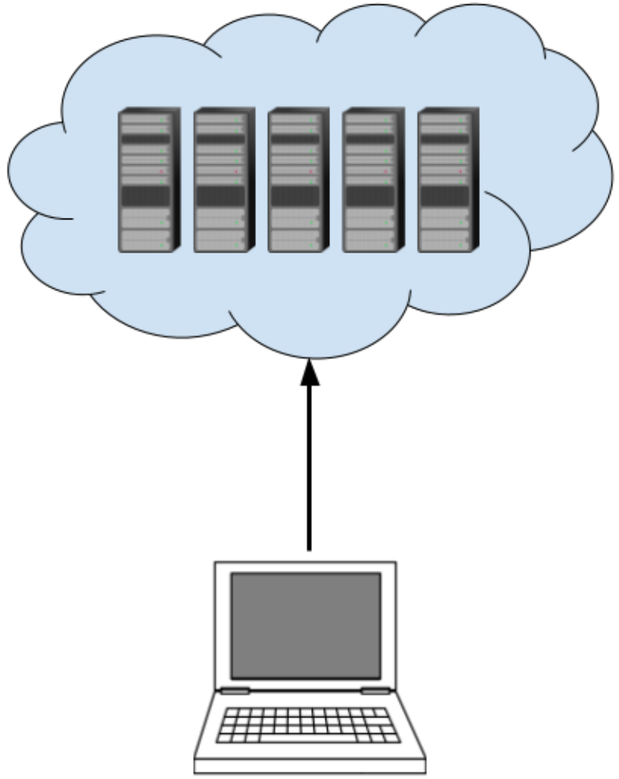
Getting started with Databricks

INTRODUCTION TO DATABRICKS

Kevin Barlow
Data Practitioner

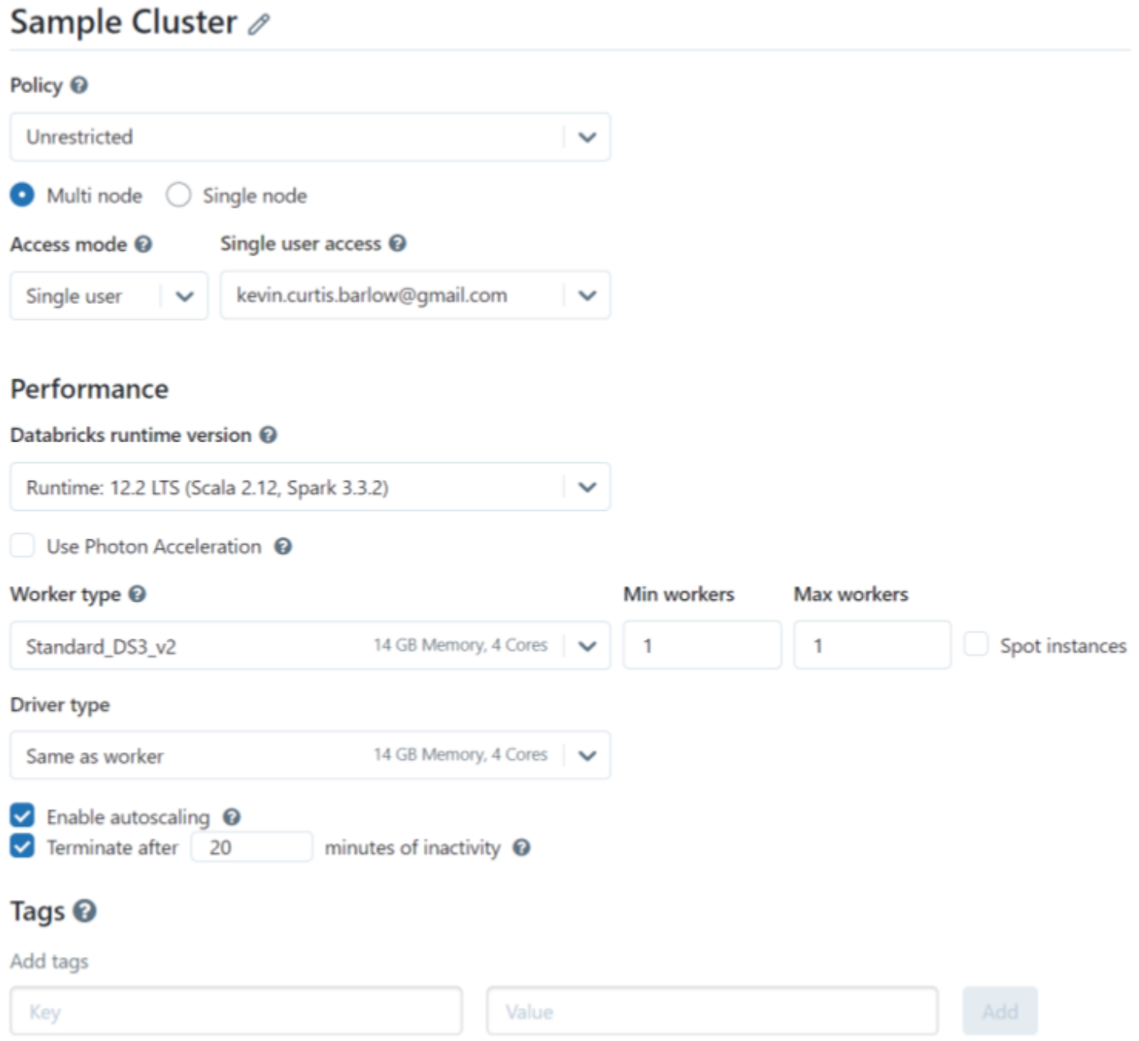
Compute cluster refresh



Create your first cluster

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

Configuration options:

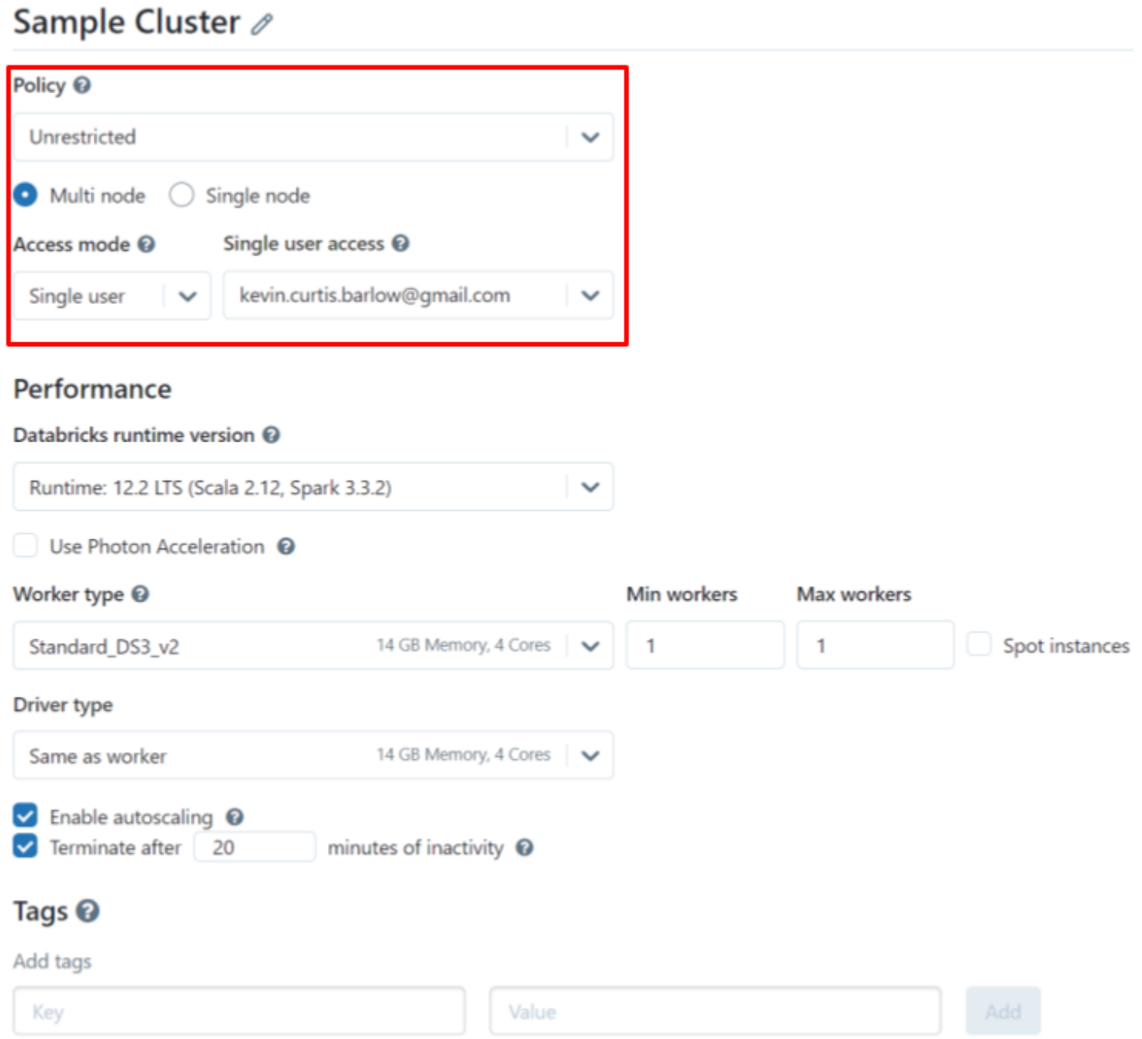


Create your first cluster

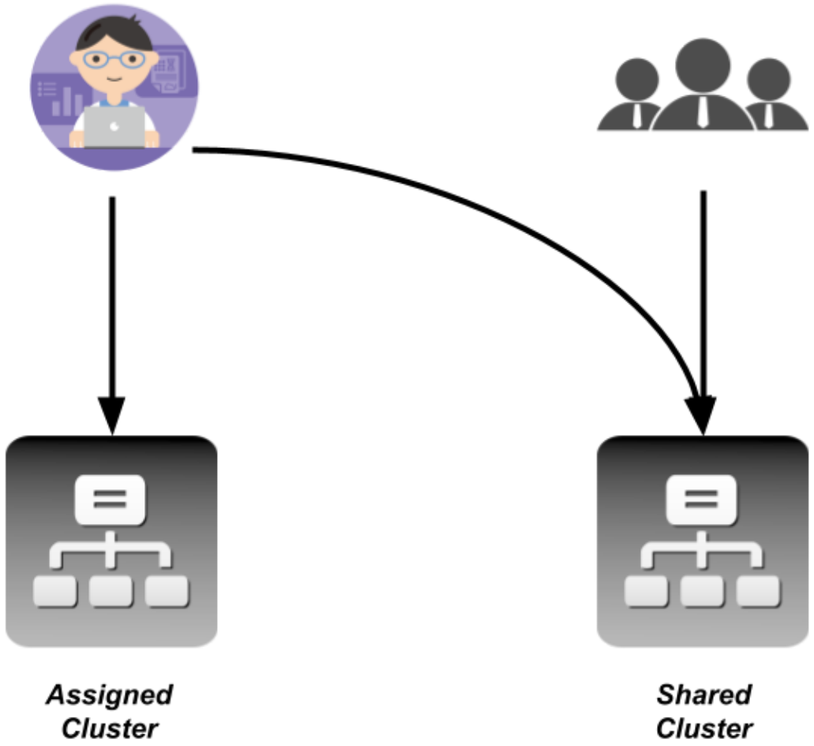
The first step is to create a cluster for your data processing!

Configuration options:

- Cluster policies and access



Cluster Access

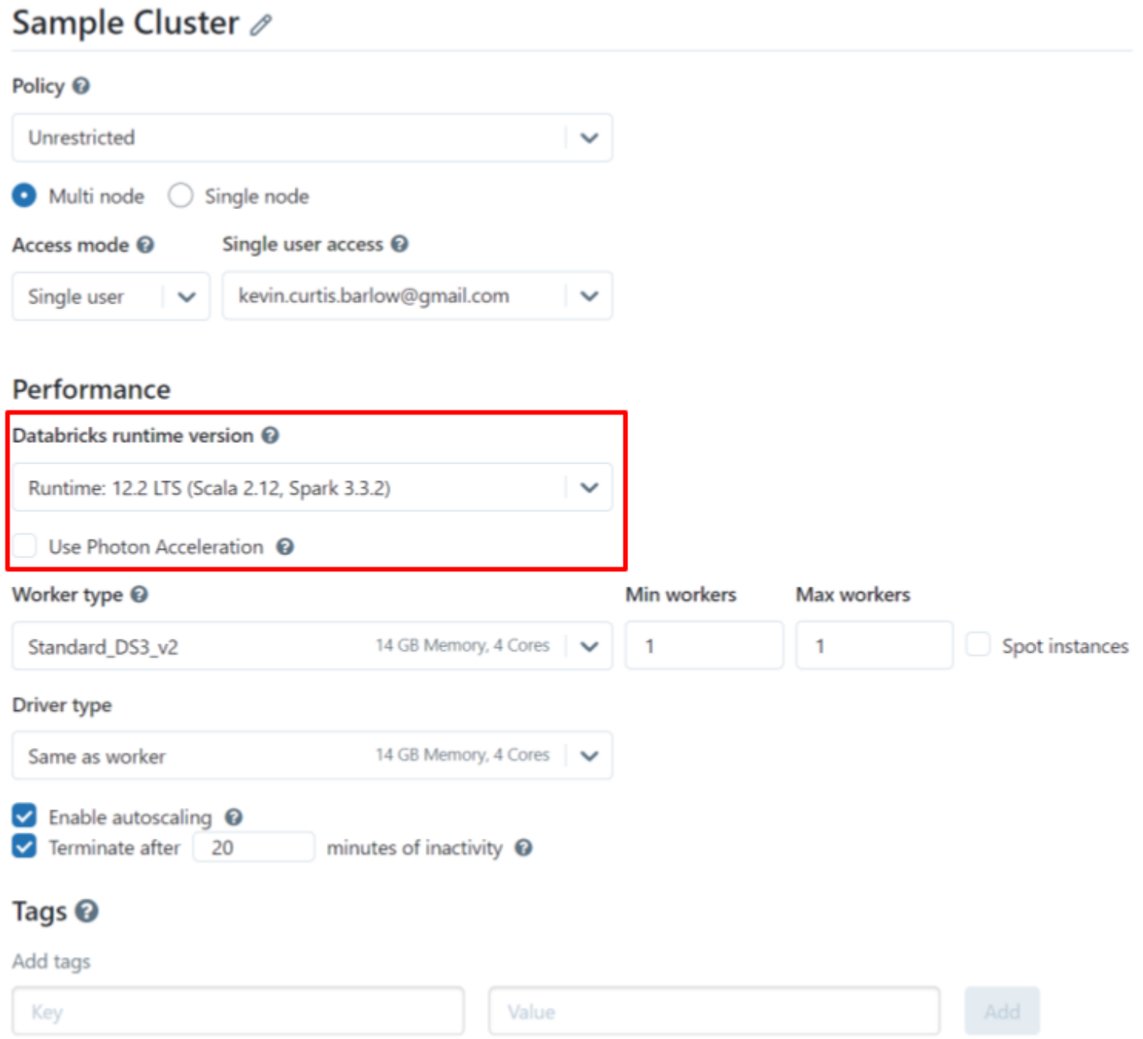


Create your first cluster

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

Configuration options:

- Cluster policies and access
- Databricks Runtime
- Photon Acceleration

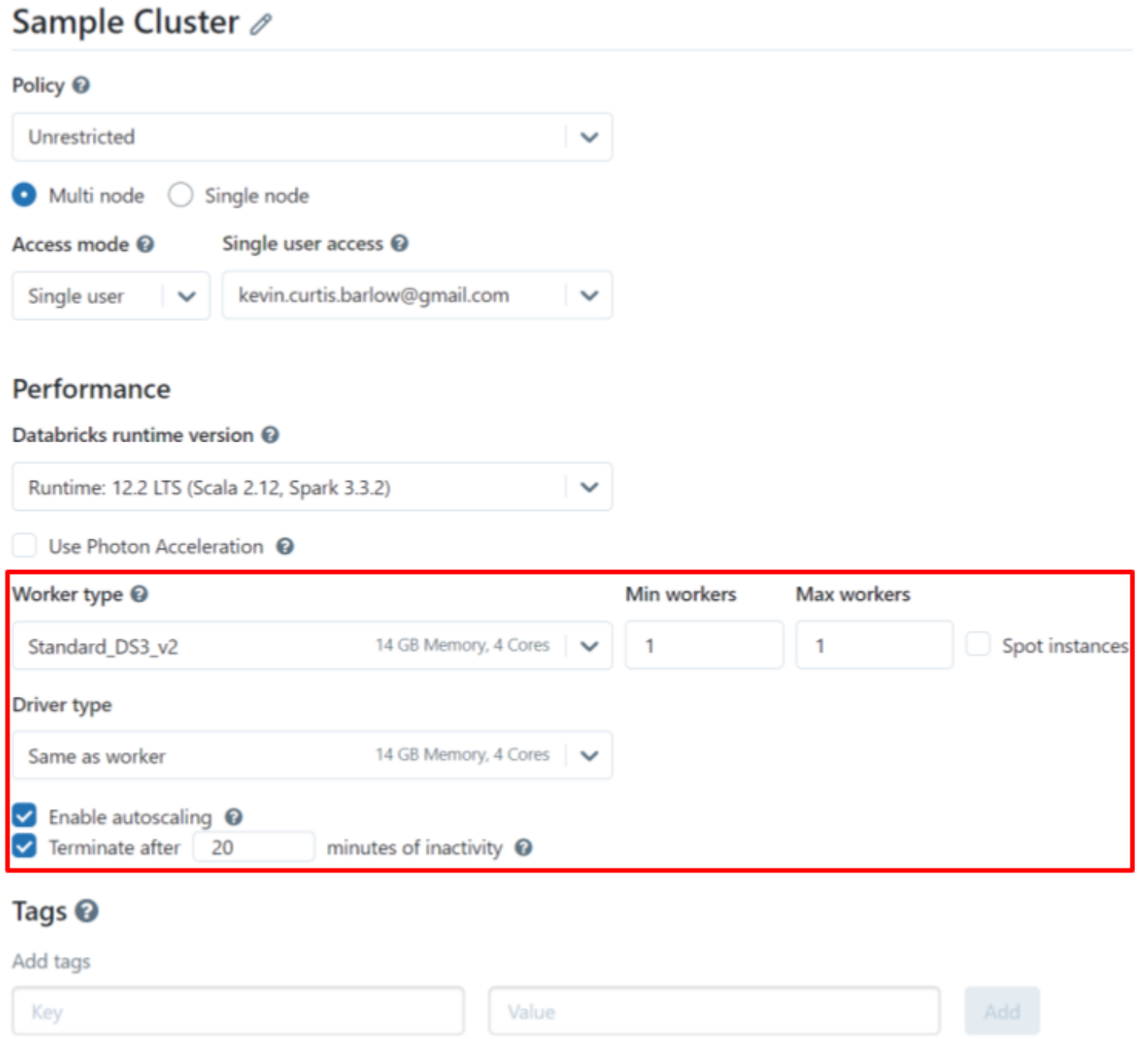


Create your first cluster

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



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

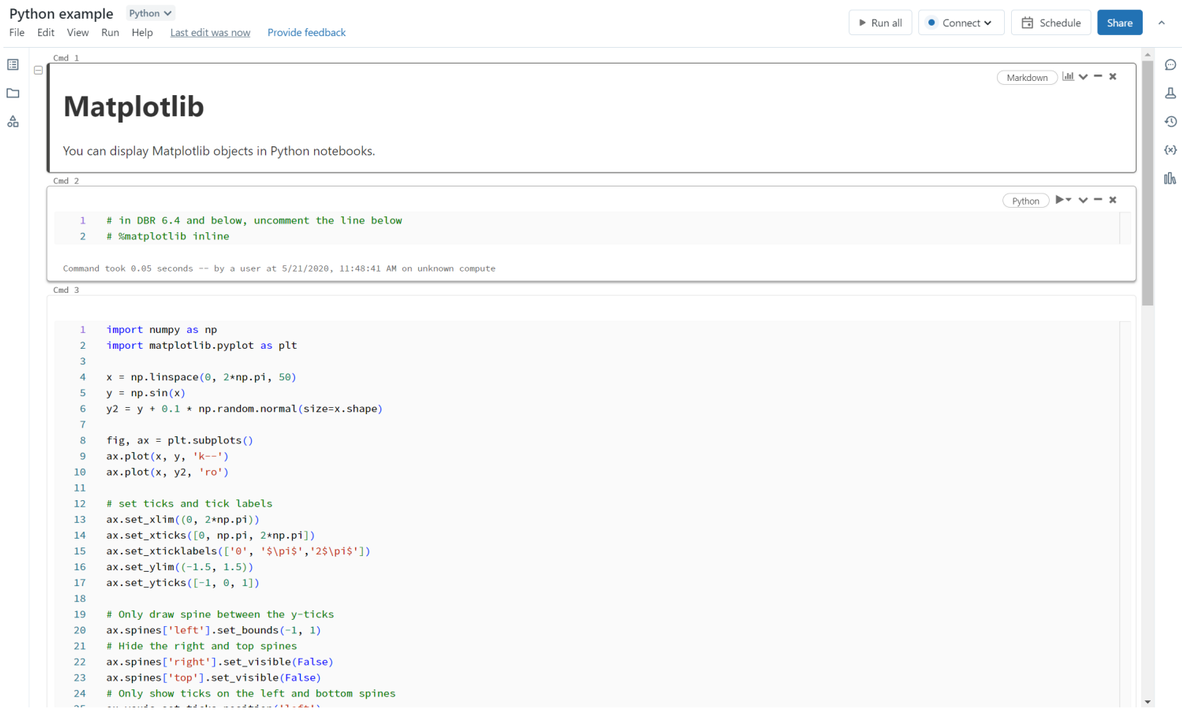
You can also upload new data by clicking the "plus" icon!

[1] Photo by Jakub 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



Let's practice!

INTRODUCTION TO DATABRICKS

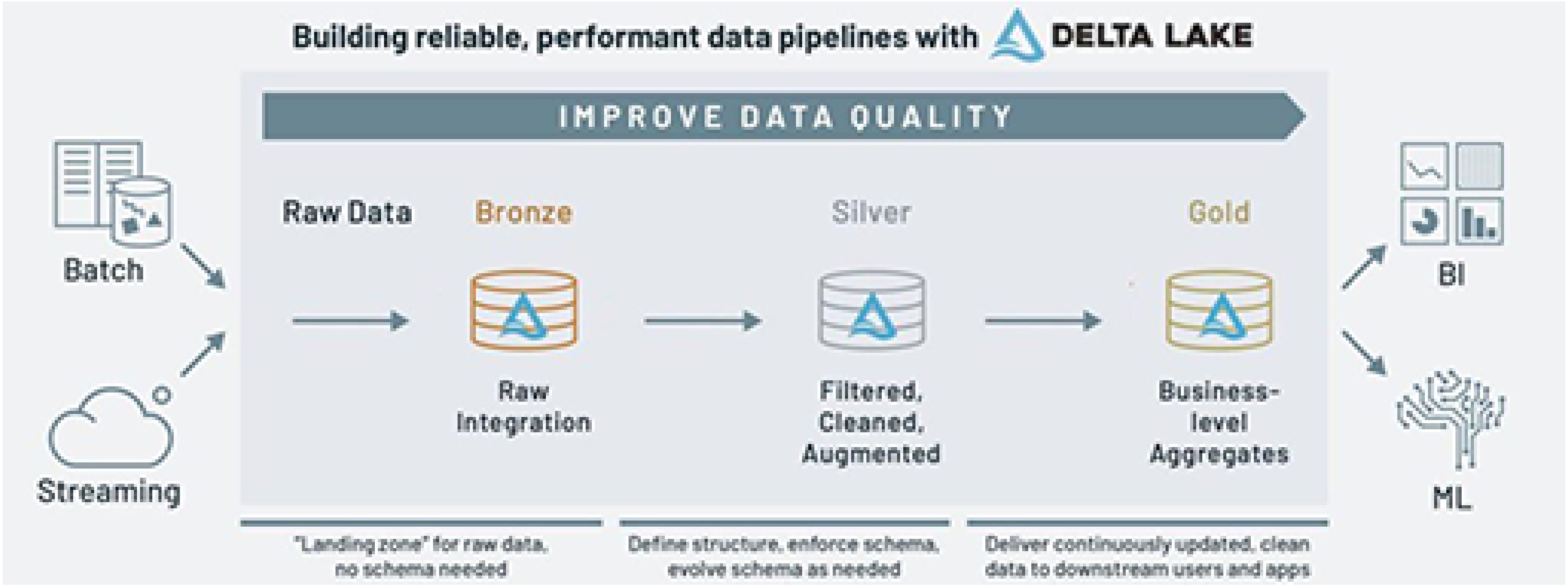
Data Engineering foundations in Databricks

INTRODUCTION TO DATABRICKS

Kevin Barlow
Data Practitioner

datacamp

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.

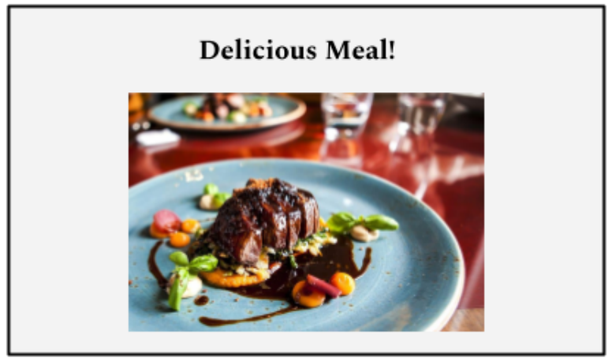
- Feels like a table when reading
- Access to underlying files (Parquet and JSON)

```
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
• Ribeye Steak • Carrots • Red Wine •	1. Preheat oven 2. Mise en place 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
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

datacamp

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

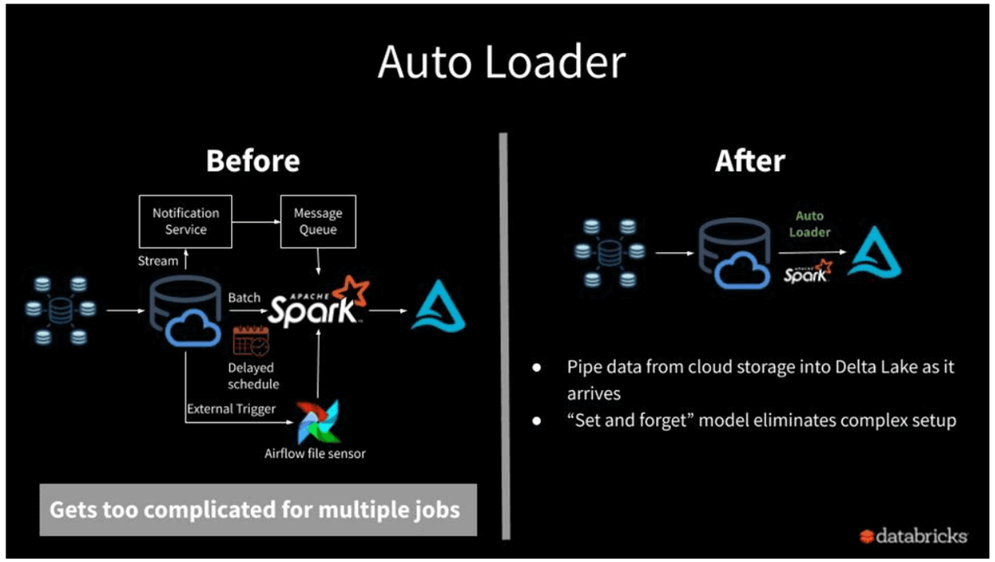
```
df
  .groupBy(col('region'))
  .agg(sum(col('sales')))
```

Auto Loader

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

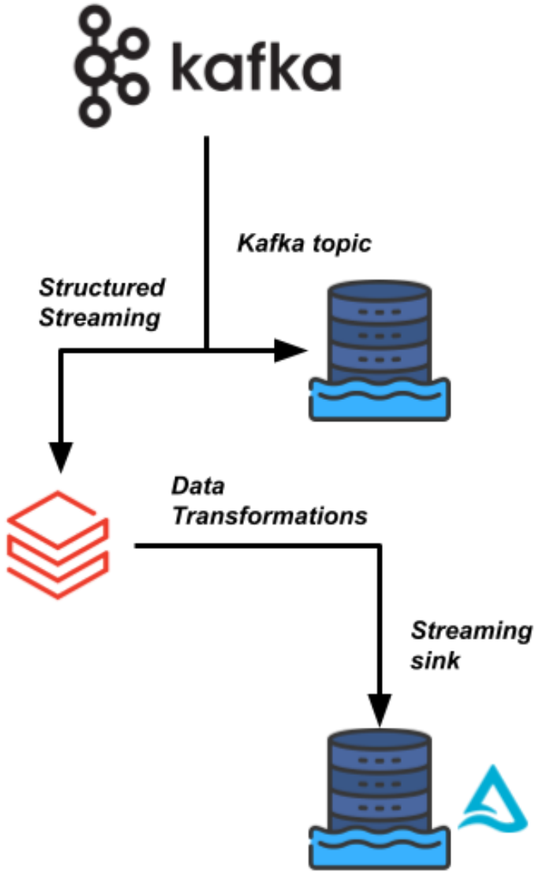
- Incremental processing
- Efficient processing
- Automatic

```
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!

INTRODUCTION TO DATABRICKS

Orchestration in Databricks

INTRODUCTION TO DATABRICKS

Kevin Barlow
Data Analytics Practitioner

datacamp

What is data orchestration?

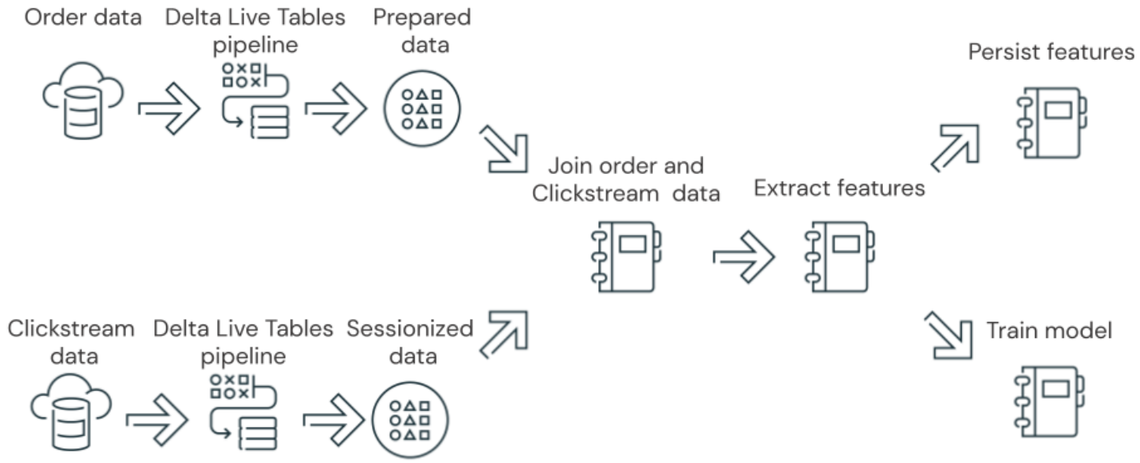
- Data orchestration is a form of automation!

- 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/workflows

What can we orchestrate?

Data engineers/data scientists

Databricks Notebook

Delta Live Tables

JAR

APACHE Spark™

dbt™

Data analysts

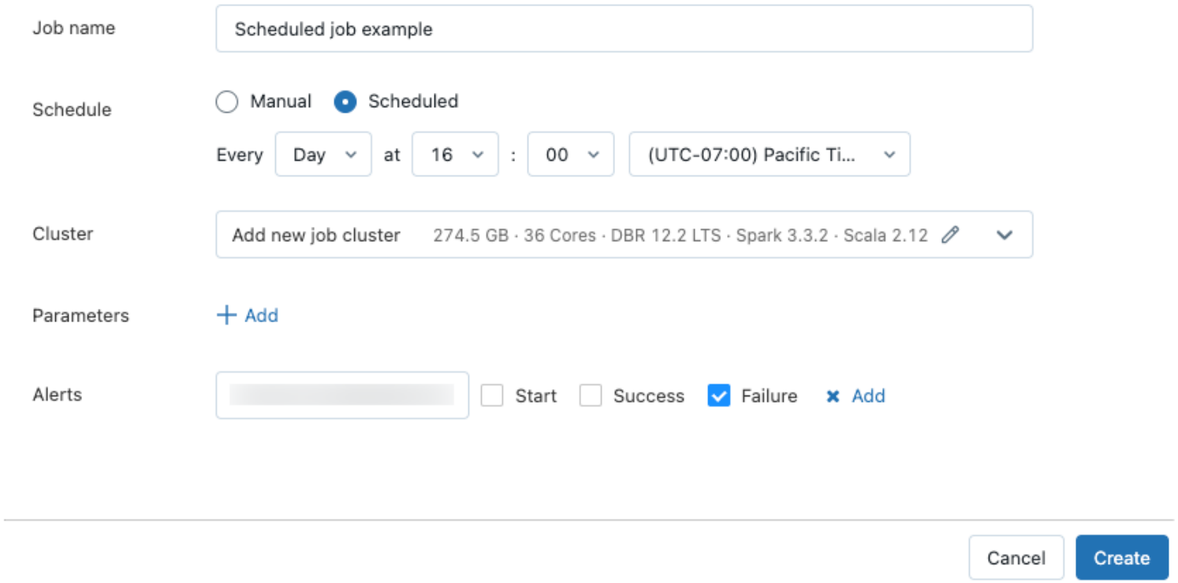
SQL

Databricks Jobs

Workflows UI

Users can create jobs directly from the Databricks UI:

- Directly from a notebook
- In the Workflows section



[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	Data
`CREATE LIVE TABLE raw_data as SELECT * FROM cloud_files("/data","json")`	raw_data
`CREATE LIVE TABLE clean_data as SELECT ... FROM LIVE. raw_data`	clean_data

`@dlt.table`
`def scord_records():`
`return read("clean_data").map(lambda d: model_score(d))`
	scored_records

Delta Live Tables

Delta Live Tables understands and coordinates data flow between your queries

Delta Live Tables

Delta Live Tables understands and coordinates data flow between your queries

Delta Live Tables Pipeline	Data
`CREATE LIVE TABLE raw_data as SELECT * FROM cloud_files("/data","json")`	raw_data
`CREATE LIVE TABLE clean_data as SELECT ... FROM LIVE. raw_data`	clean_data

`@dlt.table`
`def scord_records():`
`return read("clean_data").map(lambda d: model_score(d))`
	scored_records

Let's practice!

INTRODUCTION TO DATABRICKS

End-to-end data pipeline example in Databricks

INTRODUCTION TO DATABRICKS

Kevin Barlow
Data Practitioner

datacamp

Let's practice!

INTRODUCTION TO DATABRICKS