



The Databricks Platform Introduction

All your data, analytics and AI on one platform

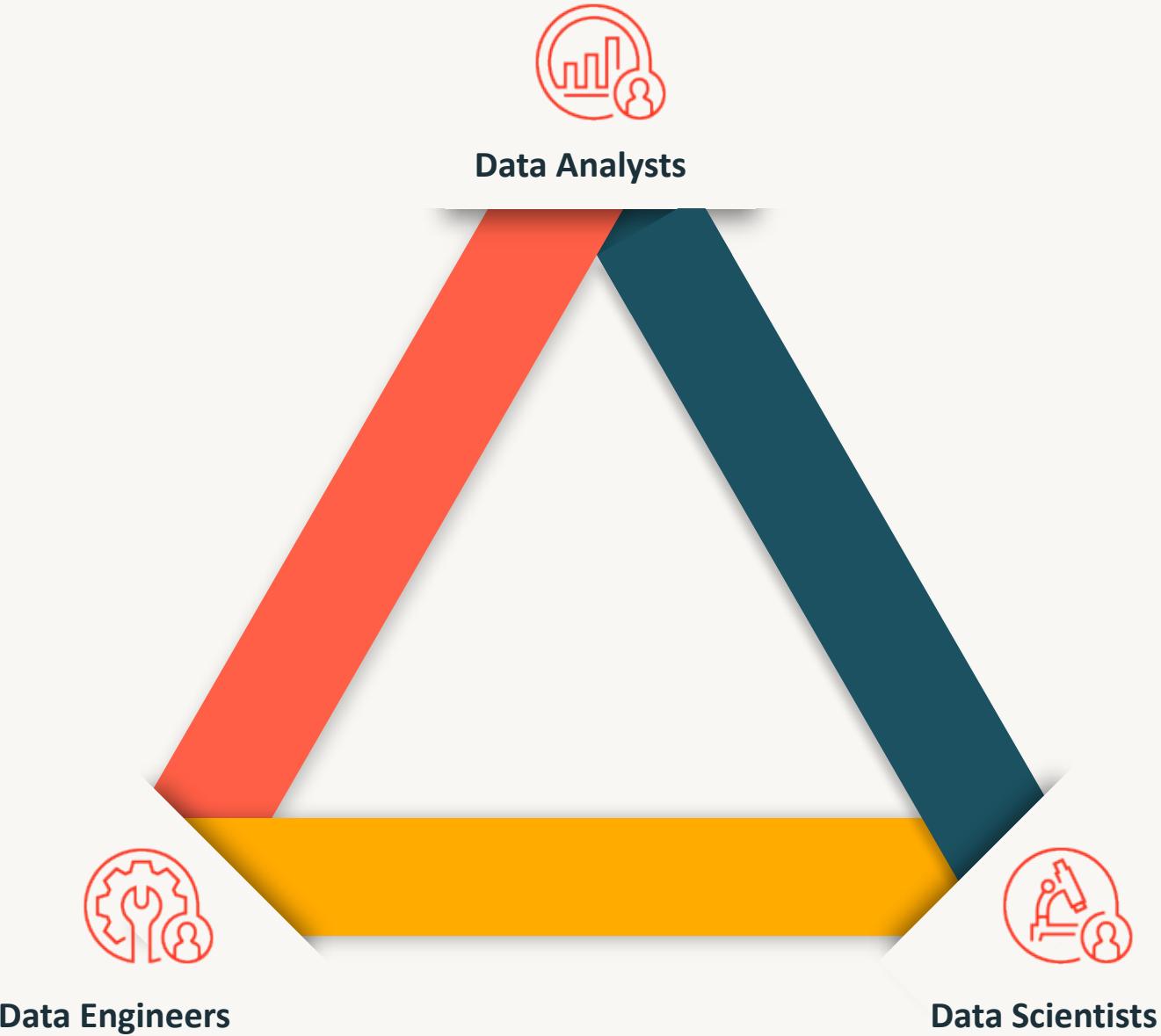


What is DataBricks ?

DataBricks is a unified & open Data
and Analytics Platform

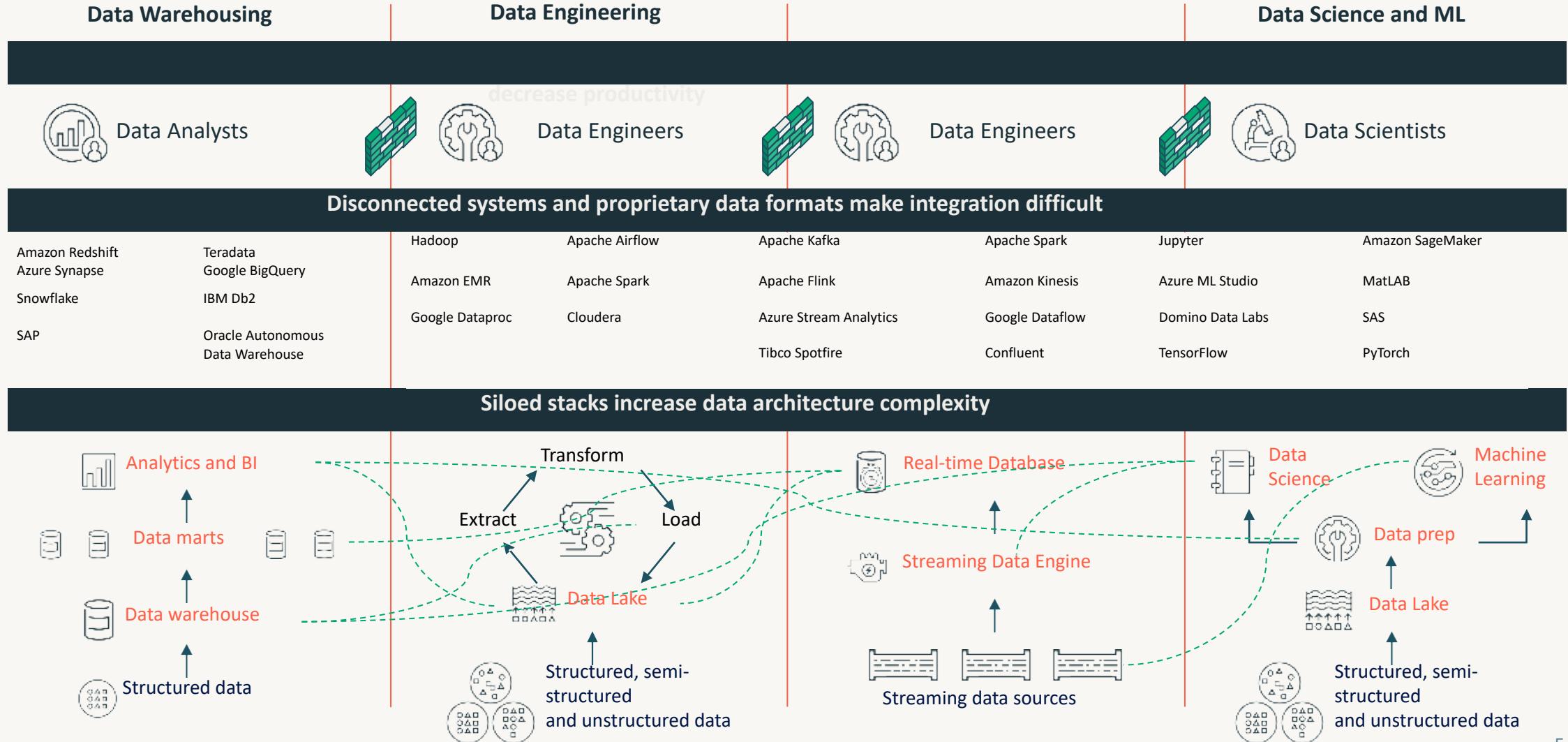


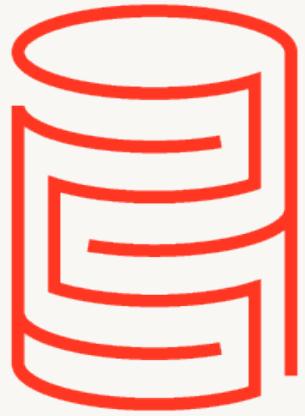
Modern Data Teams



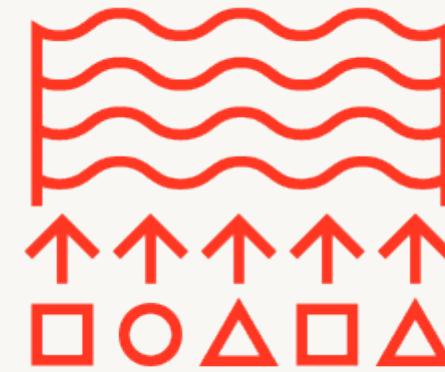
How the data management looks
like today ?

Data management complexity





Data Warehouse



Data Lake

Warehouses and lakes create complexity

Two separate copies of the data

Warehouses

Proprietary

Lakes

Open

Incompatible interfaces

Warehouses

SQL

Lakes

Python

Incompatible security and governance models

Warehouses

Tables

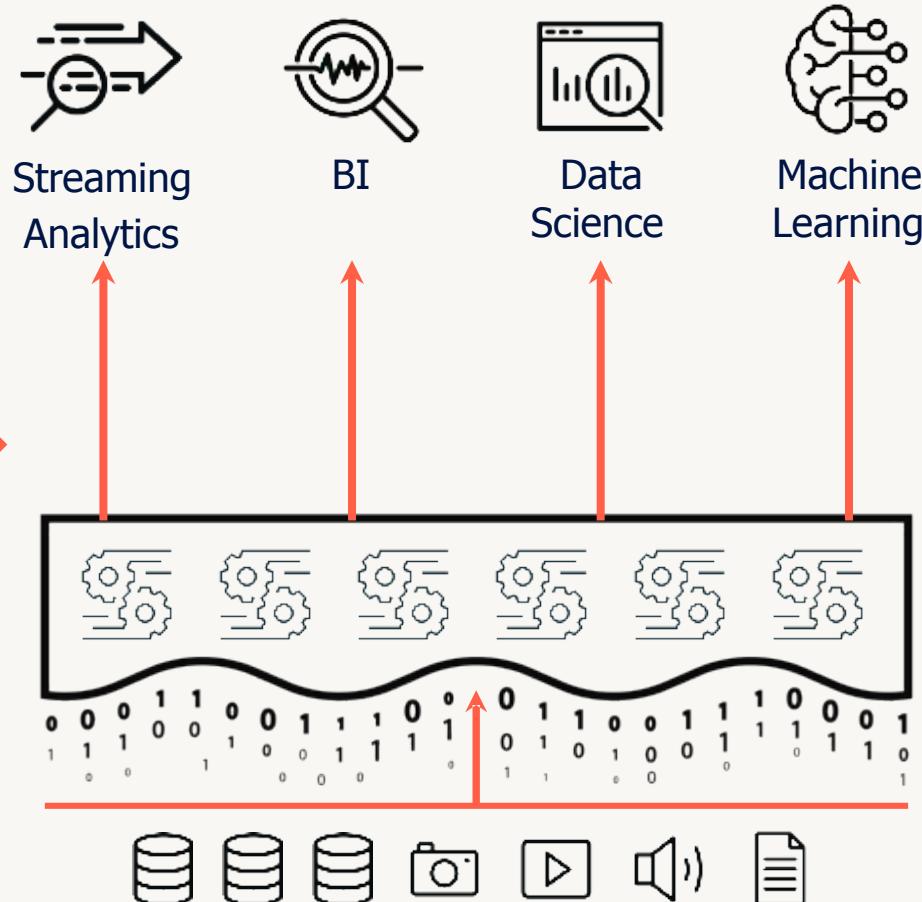
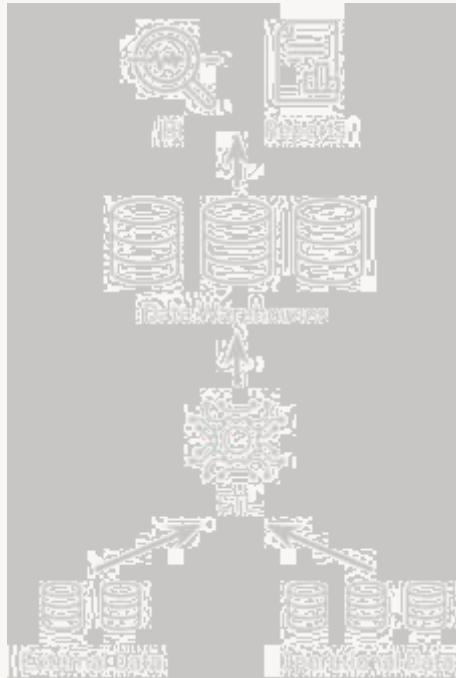
Lakes

Files

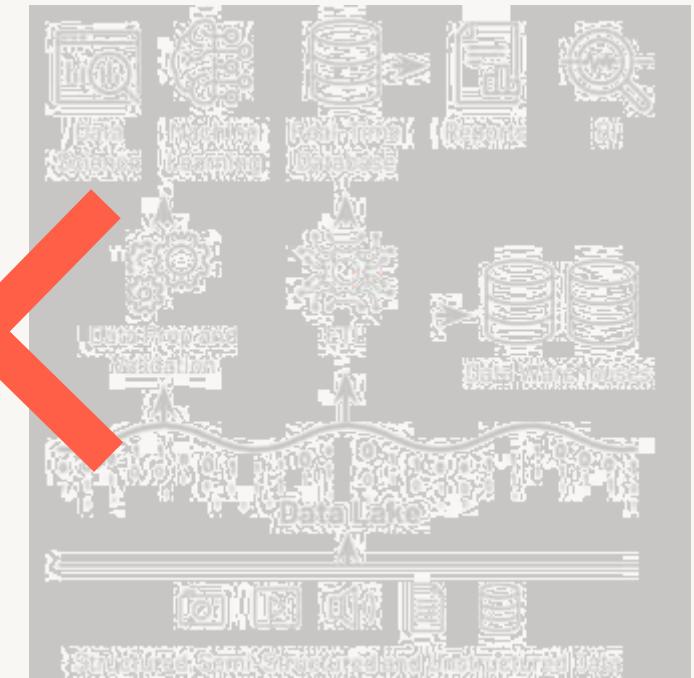
Data Lakehouse

One platform to unify all of
your data, analytics, and AI workloads

Data Warehouse

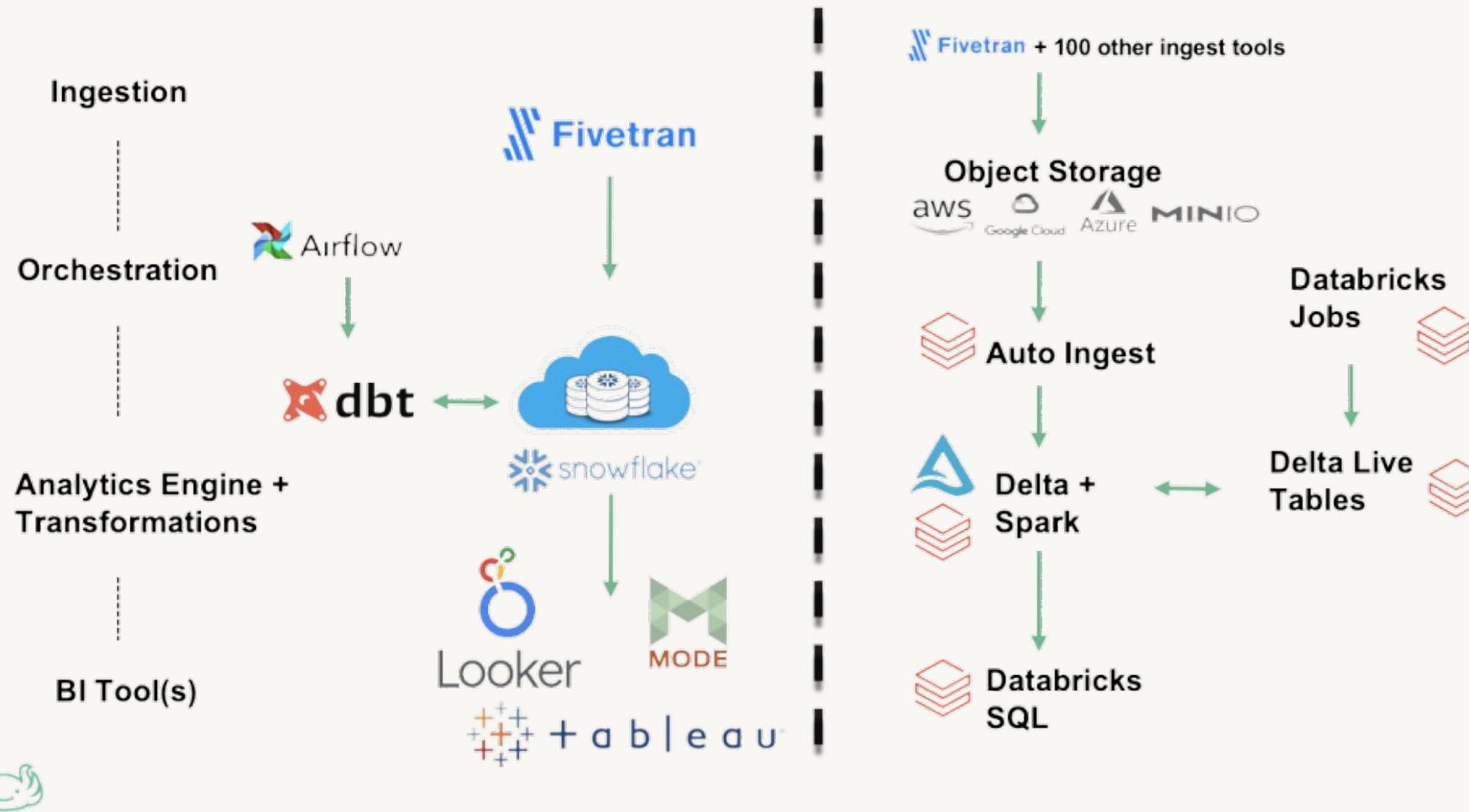


Data Lake

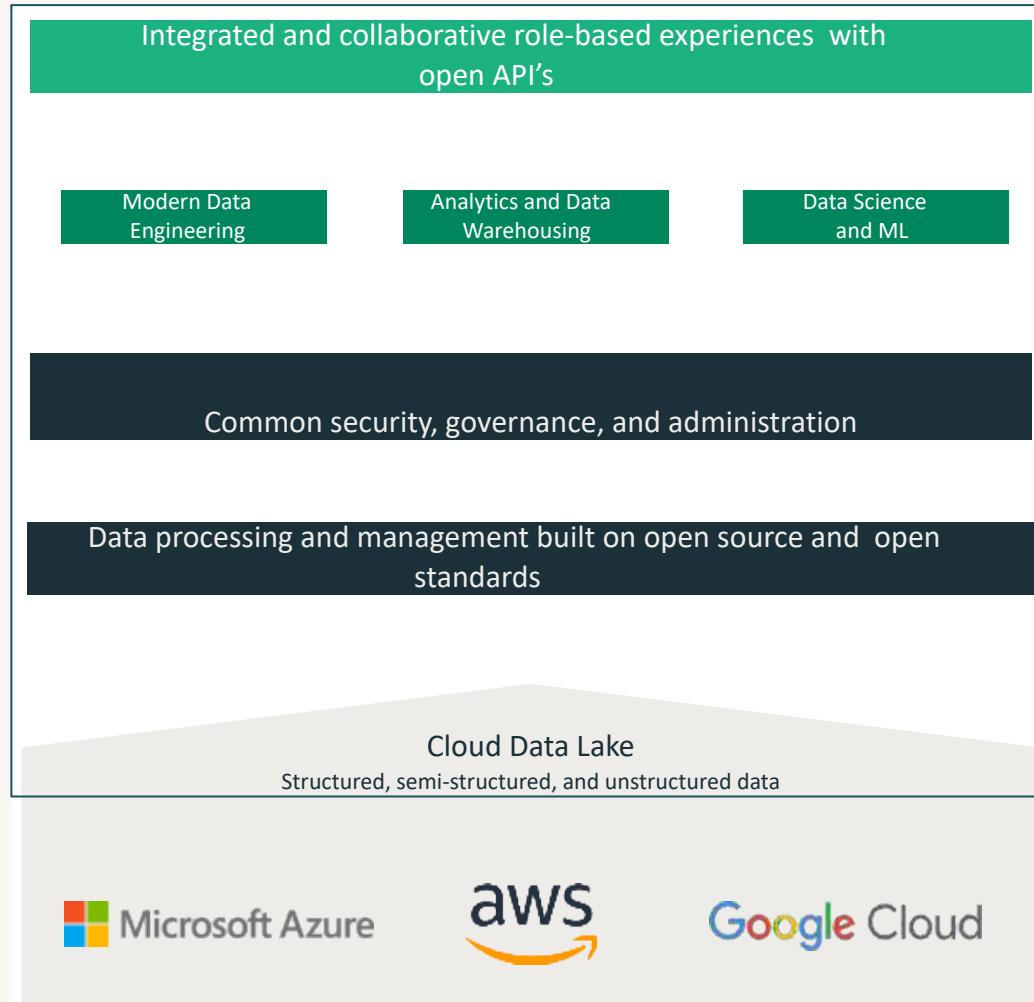


Why choose Databricks ?

Modern Data Stack vs Databricks Ecosystem



The data lakehouse offers a better path



Lake-first approach that builds upon where the freshest, most complete data resides

AI/ML from the ground up
High reliability and performance **Single approach** to managing data

Support for all use cases on a single platform:

- Data engineering
- Data warehousing
- Real time streaming
- Data science and ML

Built on **open source** and open standards

Multi-cloud, work with your cloud of choice

The Data Lakehouse Foundation

Data Lake



An open approach to bringing data management and governance to data lakes

Better reliability with transactions

48x faster data processing with indexing

Data governance at scale with fine-grained access control lists

Data Warehouse



What is Delta Lake?

- A open source project that enables building a Lakehouse architecture on top of data lakes.
- An storage layer that brings scalable, ACID transactions to Apache Spark and other big-data engines.
- Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing on top of existing data lakes, such as S3, ADLS, GCS, and HDFS.

Key Features:

- ACID Transactions
- Scalable Metadata Handling
- Time Travel (data versioning)
- Open Format
- Delta Lake change data feed
- Unified Batch and Streaming Source and Sink
- Schema Enforcement
- Schema Evolution
- Audit History
- Updates and Delete
- 100% Compatible with Apache Spark API
- Data Clean-up

<https://databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html>

Delta Lake solves challenges with data lakes

**RELIABILITY &
QUALITY**



ACID transactions

**PERFORMANCE &
LATENCY**



Advanced indexing & caching

GOVERNANCE



Governance with Data Catalogs

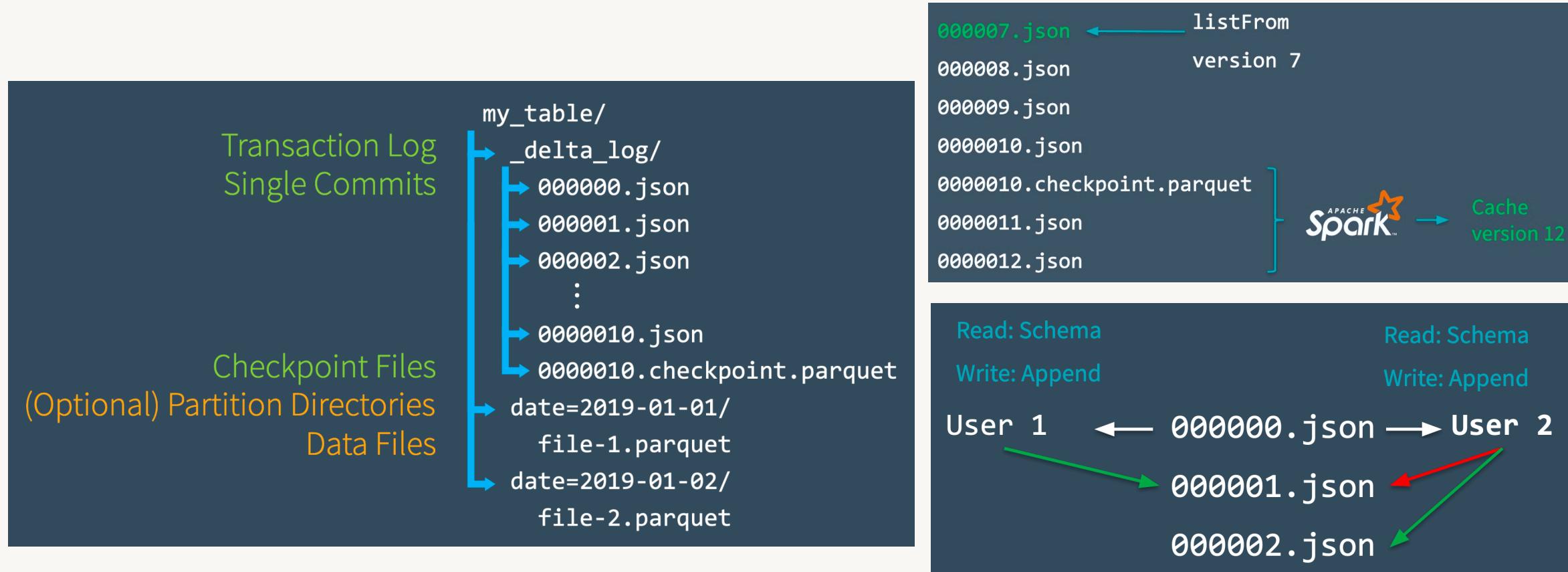
Delta Lake key feature - ACID transaction

- Add File: It adds the data file
- Remove File: It removes the data file
- Update Metadata: It updates the table metadata.
- Set Transaction: It records that a structure streaming job created a micro-batch with ID
- Change Protocol: Makes more secure by transferring Delta Lakes to the latest securing protocol.
- Commit Info: It contains the information about the Commits.



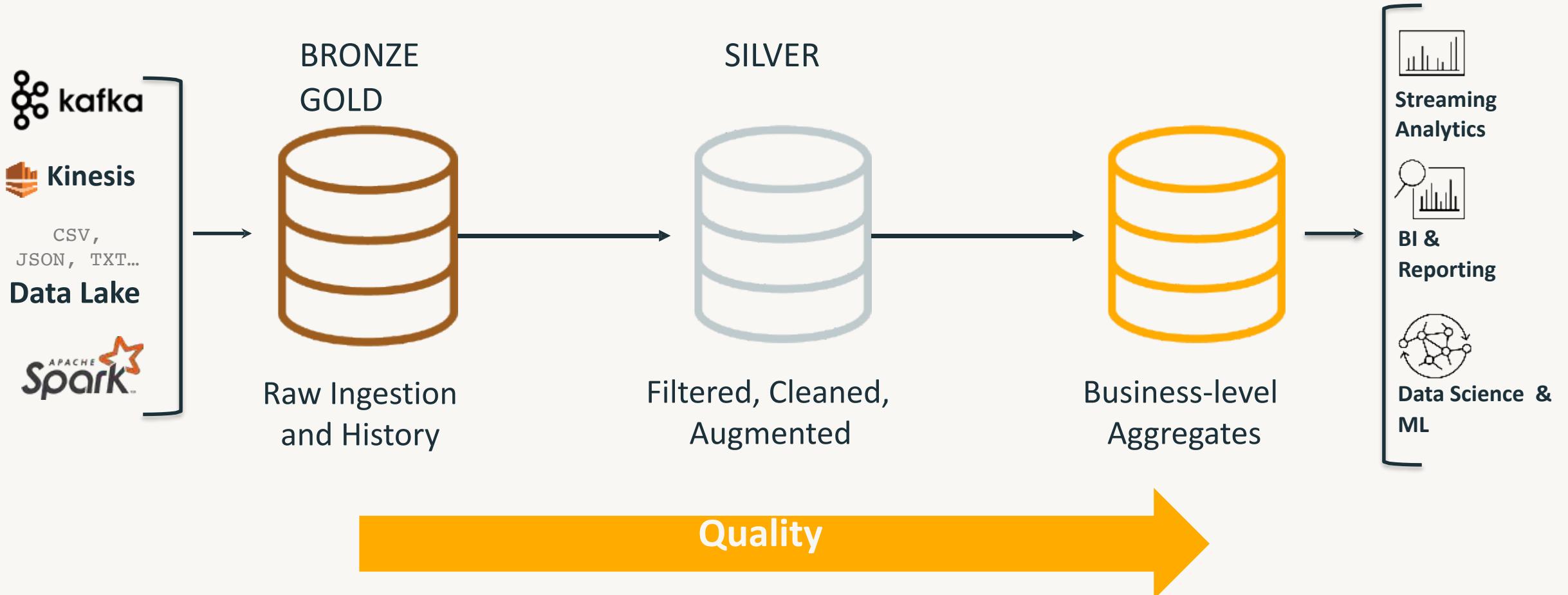
State Recomputing With Checkpoint Files

- Delta Lake automatically generates checkpoint files every 10 commits
- Delta Lake saves a checkpoint file in Parquet format in the same `_delta_log` subdirectory.



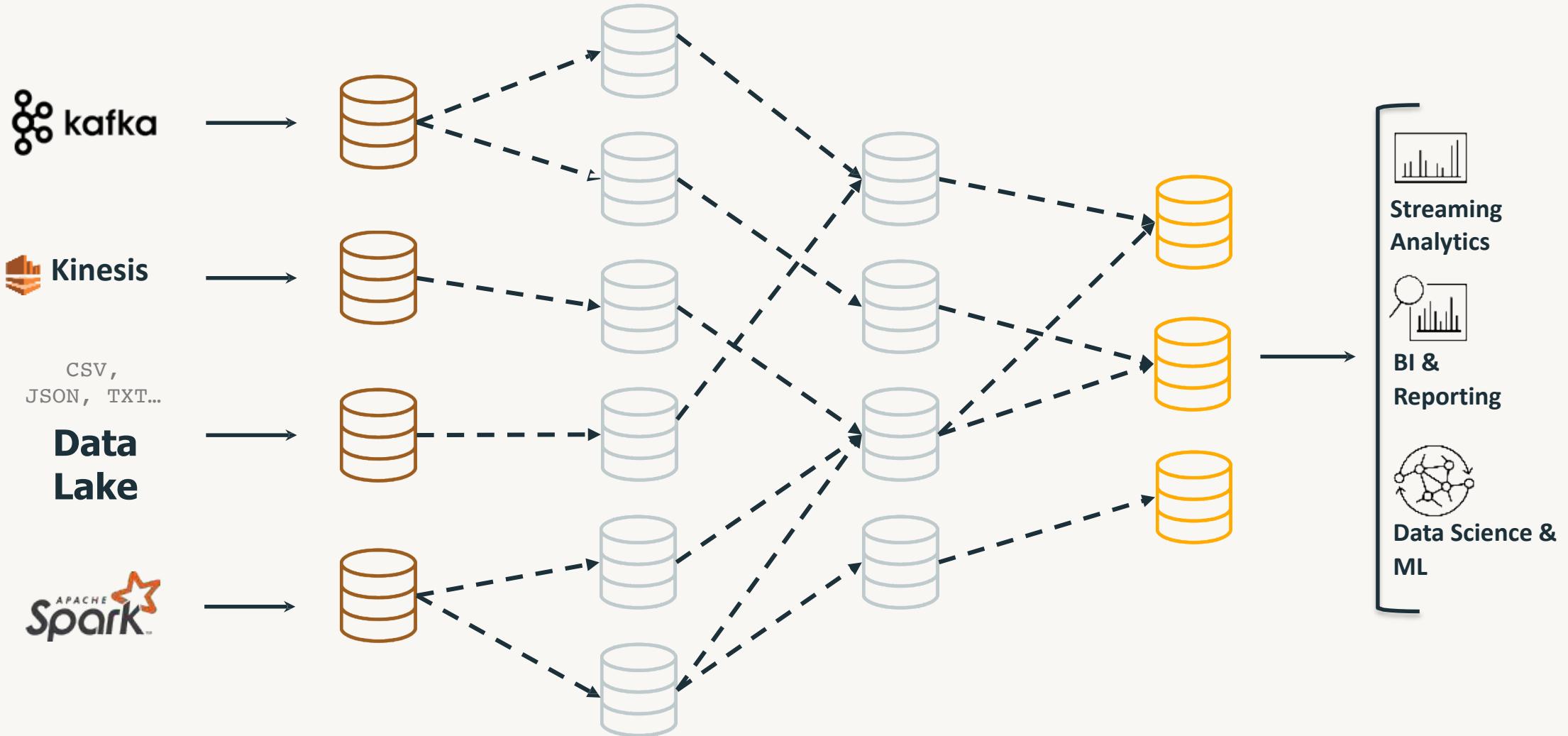
Building the foundation of a Lakehouse

Greatly improve the quality of your data for end users

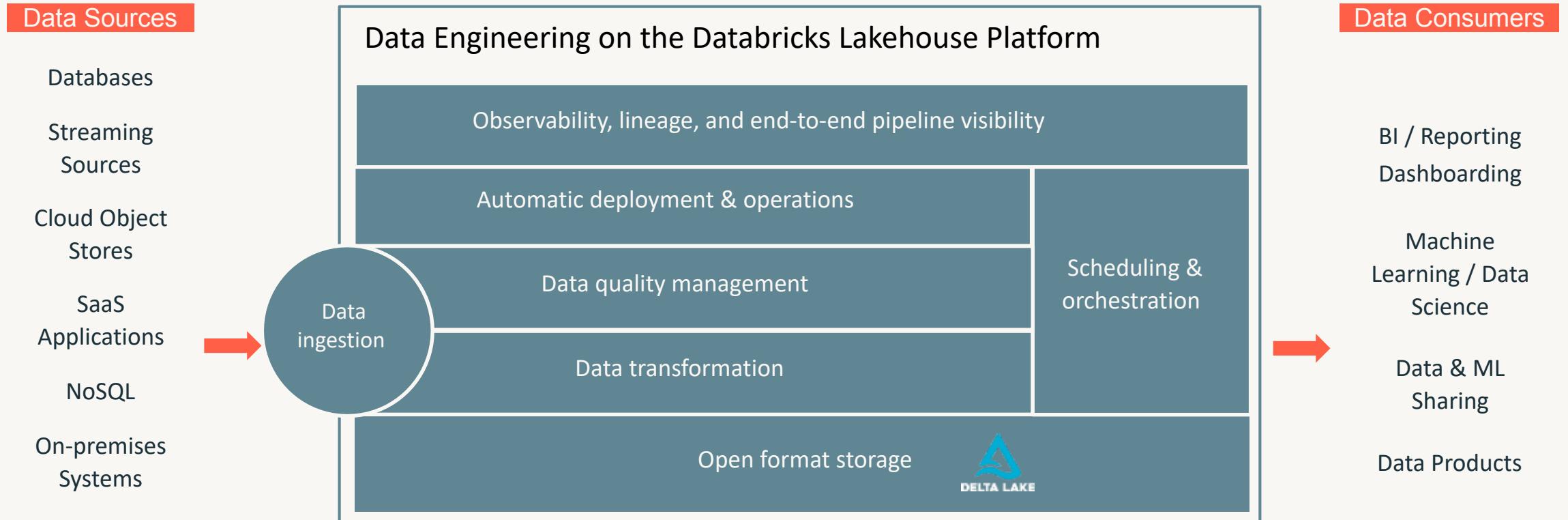


But the reality is not so simple

Maintaining data quality and reliability at scale is complex and brittle



Modern data engineering on the lakehouse



Data Science & Engineering Workspace

Databricks Workspaces: Clusters

It is a set of computation resources where a developer can run Data Analytics, Data Science, or Data Engineering workloads.

The workloads can be executed in the form of a set of commands written in a notebook

The screenshot shows the Databricks Clusters interface. On the left is a dark sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area is titled "Clusters / Test". It has tabs for Configuration (selected), Notebooks, Libraries, Event Log, Spark UI, Driver Logs, Metrics, Apps, and Spark Cluster UI - Master. Below these tabs are sections for Databricks Runtime Version (7.5 ML), Driver Type (Community Optimized, 15.3 GB Memory, 2 Cores, 1 DBU), and Instance (Free 15GB Memory note). At the bottom are tabs for Instances (selected), Spark, JDBC/ODBC, and Permissions. The Availability Zone is set to us-west-2c.

Databricks Workspaces: Notebooks

It is a Web Interface where a developer can write and execute codes. Notebook contains a sequence of runnable cells that helps a developer to work with files, manipulate tables, create visualizations, and add narrative texts

The screenshot shows the Databricks Notebook interface. The left sidebar has a dark theme with icons for Home, Workspace (which is selected), Recents, Data, Clusters, Jobs, and Search. The main area is titled "Sampling (Python)". It shows a "Detached" cluster named "Test_Cluster" (15.25 GB | 2 Cores | DBR 7.4 | Spark 3.0.1 | Scala 2.12). The notebook content includes:

- Sampling strategies**: A section with text about sampling from complex distributions.
- Reference**: A link to "An Introduction to Statistical Computing: A Simulation-based Approach, Jochen Voss".
- Cmd 2**: A text cell containing:

We want to look at ways to sample from a distribution. In most practical scenarios, the distributions are complex enough that it is difficult enough to sample from appropriately but they can often be evaluated at a certain point. The simple strategy of sampling uniformly fails for a couple of reasons.

 1. This quickly becomes inefficient as the number of dimensions grow.
 2. In high-dimensional spaces, there are vast regions of nothingness and most of the probability density is concentrated in a small region.

Ideally, we want to sample from regions in space where the function $f(x)$ and the probability of that value $f(x)$ given by $p(x)$ is high so that the contribution of this term $f(x)p(x)$ is high.

We will look at various ways to perform efficient sampling here. This is useful not only to understand how distributions are sampled in practice in a package such as SciPy but it also helps you to write your own custom distribution should that need arise.
- Cmd 3**: A code cell containing:

```
1 %%md
2 ### Sampling from Discrete Distributions
3
4 ##### Uniform Continuous to Uniform Discrete
5
6 In order to sample data from a particular distribution we can start with a uniform continuous distribution between 0 and 1 denoted as U[0,1]. The probability of picking a value less than 'a' that is between 0 and 1 is given by
```

The top navigation bar includes "View: Standard", "Permissions", "Run All", "Clear", "Publish", "Comments", "Experiment", and "Revision history". The revision history on the right shows the following log:

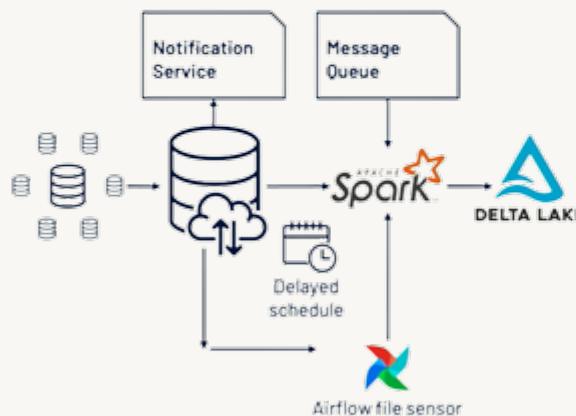
- Dec 21 2020, 12:28 PM EST (Srijith Rajamohan) - All changes saved. [Save now](#)
- Dec 21 2020, 12:18 PM EST (Srijith Rajamohan)
- Dec 21 2020, 12:02 PM EST (Srijith Rajamohan)
- Dec 21 2020, 11:52 AM EST (Srijith Rajamohan)

Databricks Workspaces: AutoLoader

Auto Loader incrementally and efficiently processes new data files as they arrive in cloud storage. Auto Loader can load data files from Google Cloud Storage (GCS, gs://) in addition to Databricks File System (DBFS, dbfs:/)

Databricks Ingest: Auto Loader

Before



After



- Pipe data from cloud storage into Delta Lake as it arrives
- "Set and forget" model eliminates complex setup

```
val checkpoint_path = "/tmp/delta/population_data/_checkpoints"
val write_path = "/tmp/delta/population_data"

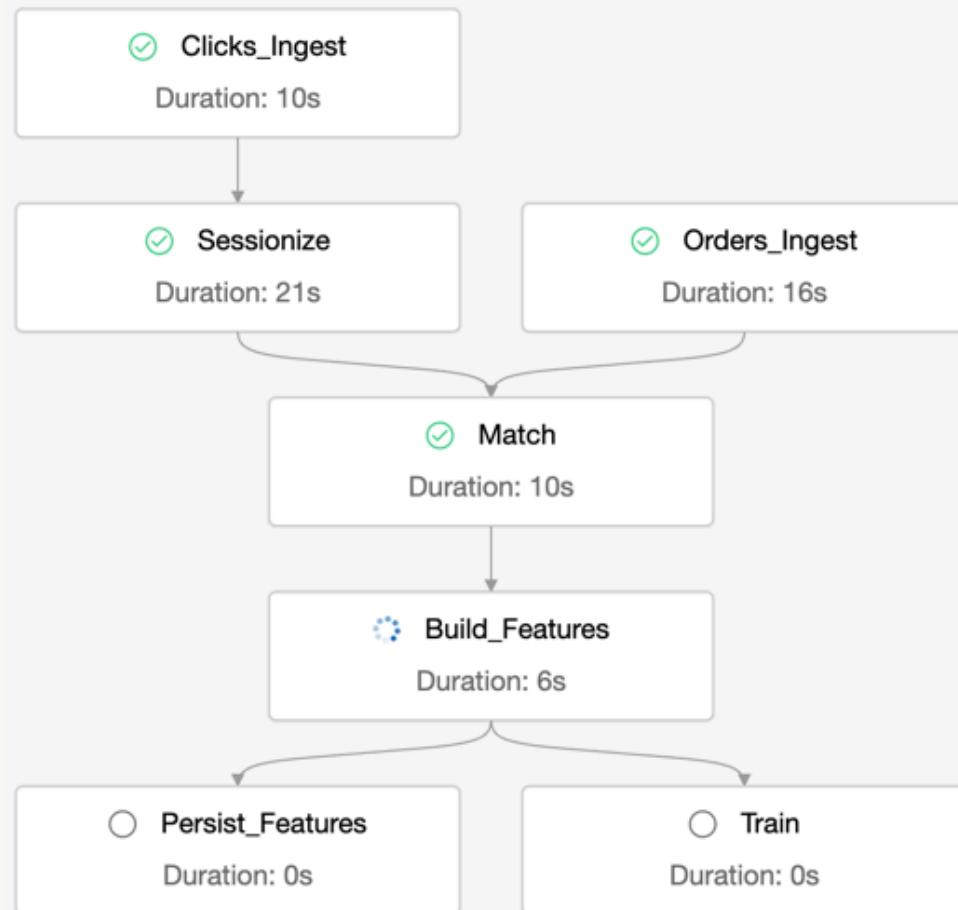
// Set up the stream to begin reading incoming files from the
// upload_path location.
val df = spark.readStream.format("cloudFiles")
  .option("cloudFiles.format", "csv")
  .option("header", "true")
  .schema("city string, year int, population long")
  .load(upload_path)

// Start the stream.
// Use the checkpoint_path location to keep a record of all files that
// have already been uploaded to the upload_path location.
// For those that have been uploaded since the last check,
// write the newly-uploaded files' data to the write_path location.
df.writeStream.format("delta")
  .option("checkpointLocation", checkpoint_path)
  .start(write_path)
```

** Supports: JSON, CSV, PARQUET, AVRO, ORC, TEXT, and BINARYFILE file formats. **

Databricks Workspaces: Jobs

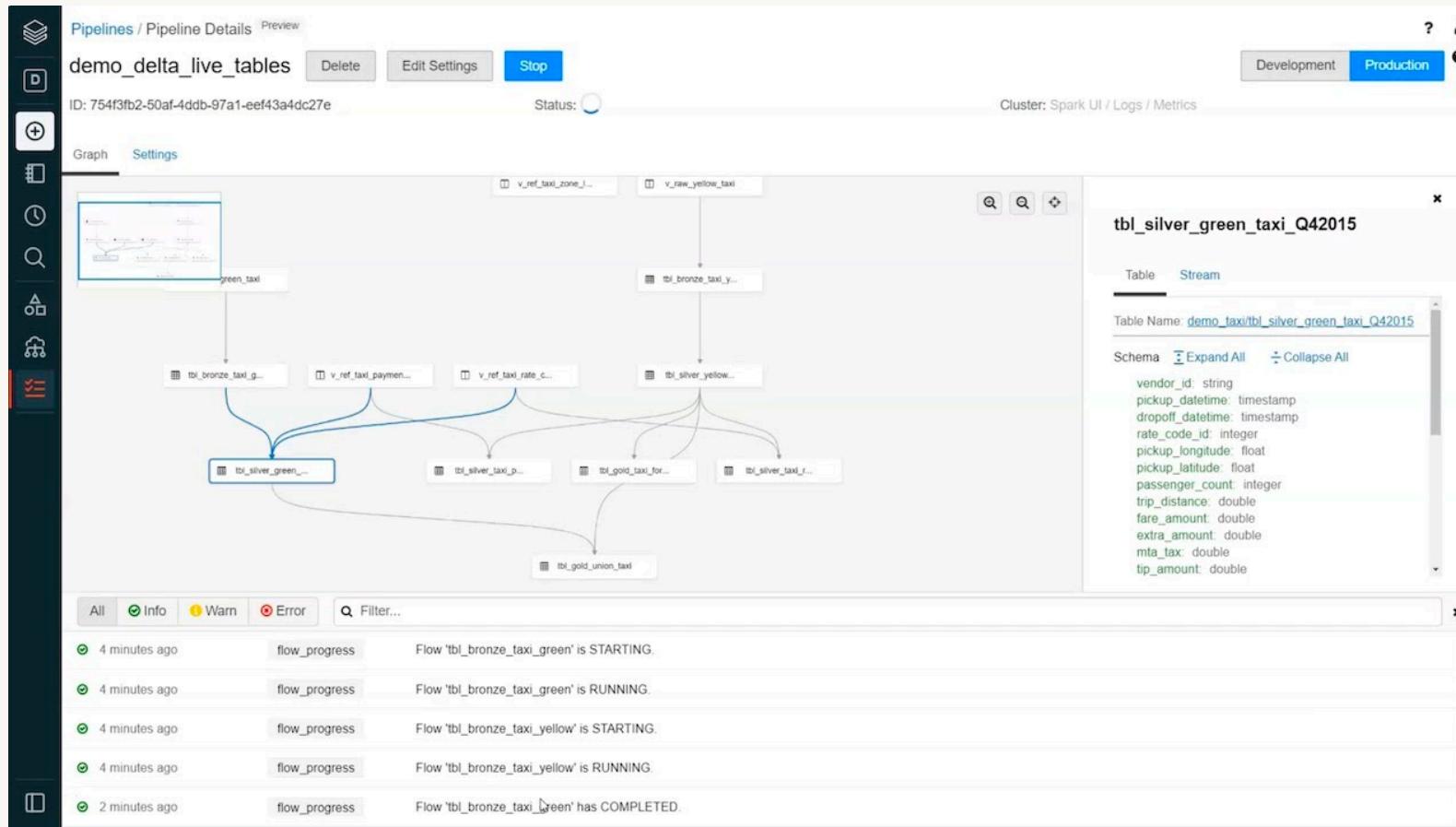
Jobs allow a user to run notebooks on a scheduled basis. It is a method of executing or automating specific tasks like ETL, Model Building, and more.



The pipeline of the ML workflow can be organized into jobs so that it sequentially runs the series of steps one after another

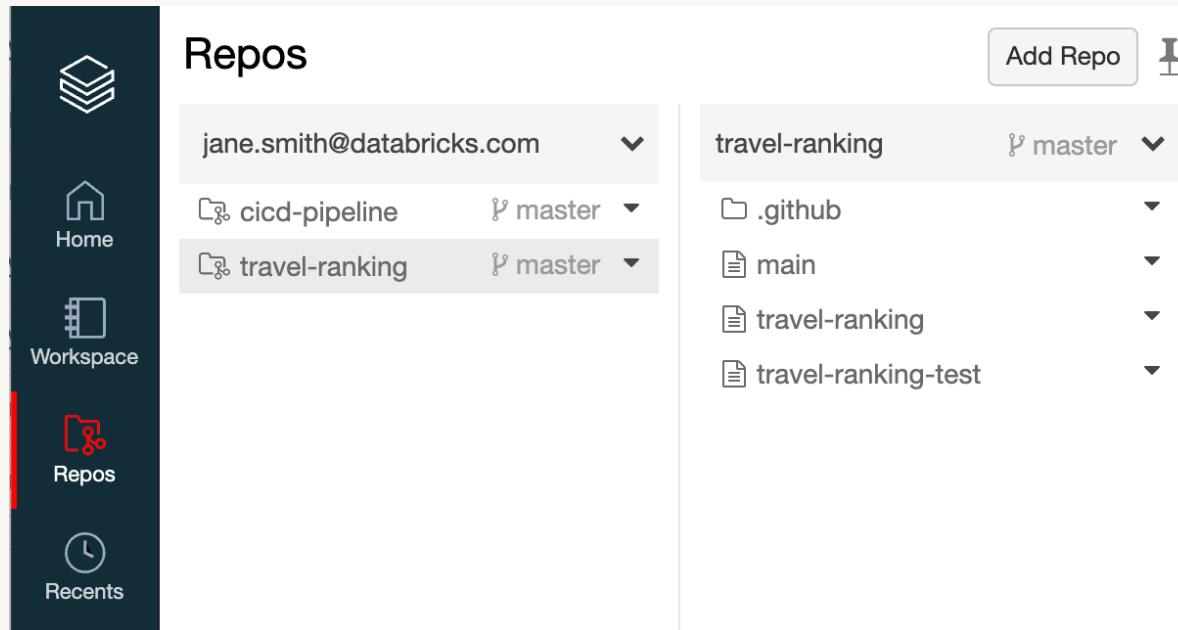
Databricks Workspaces:Delta Live Tables

Delta Live Tables is a framework designed to enable declaratively define, deploy, test & upgrade data pipelines and eliminate operational burdens associated with the management of such pipelines.



Databricks Workspaces: Repos

To empower the process of ML application development, repo's provide repository-level integration with Git-based hosting providers such as GitHub, GitLab, bitBucket, and Azure DevOps



The screenshot shows the Databricks workspace interface with the 'Repos' tab selected in the sidebar. The main area displays a list of repositories under the heading 'Repos'. A dropdown menu shows the email 'jane.smith@databricks.com'. The repository list includes:

Repository	Branch	Last Commit
cicd-pipeline	master	...
travel-ranking	master	...

An 'Add Repo' button is located at the top right. The right panel shows the detailed structure of the 'travel-ranking' repository, which contains branches: .github, main, travel-ranking, and travel-ranking-test.

Developers can write code in a Notebook and Sync it with the hosting provider, allowing developers to clone, manage branches, push changes, pull changes, etc.

Databricks Workspaces: Models

It refers to a Developer's ML Workflow Model registered in the MLflow Model Registry, a centralized model store that manages the entire life cycle of MLflow models.

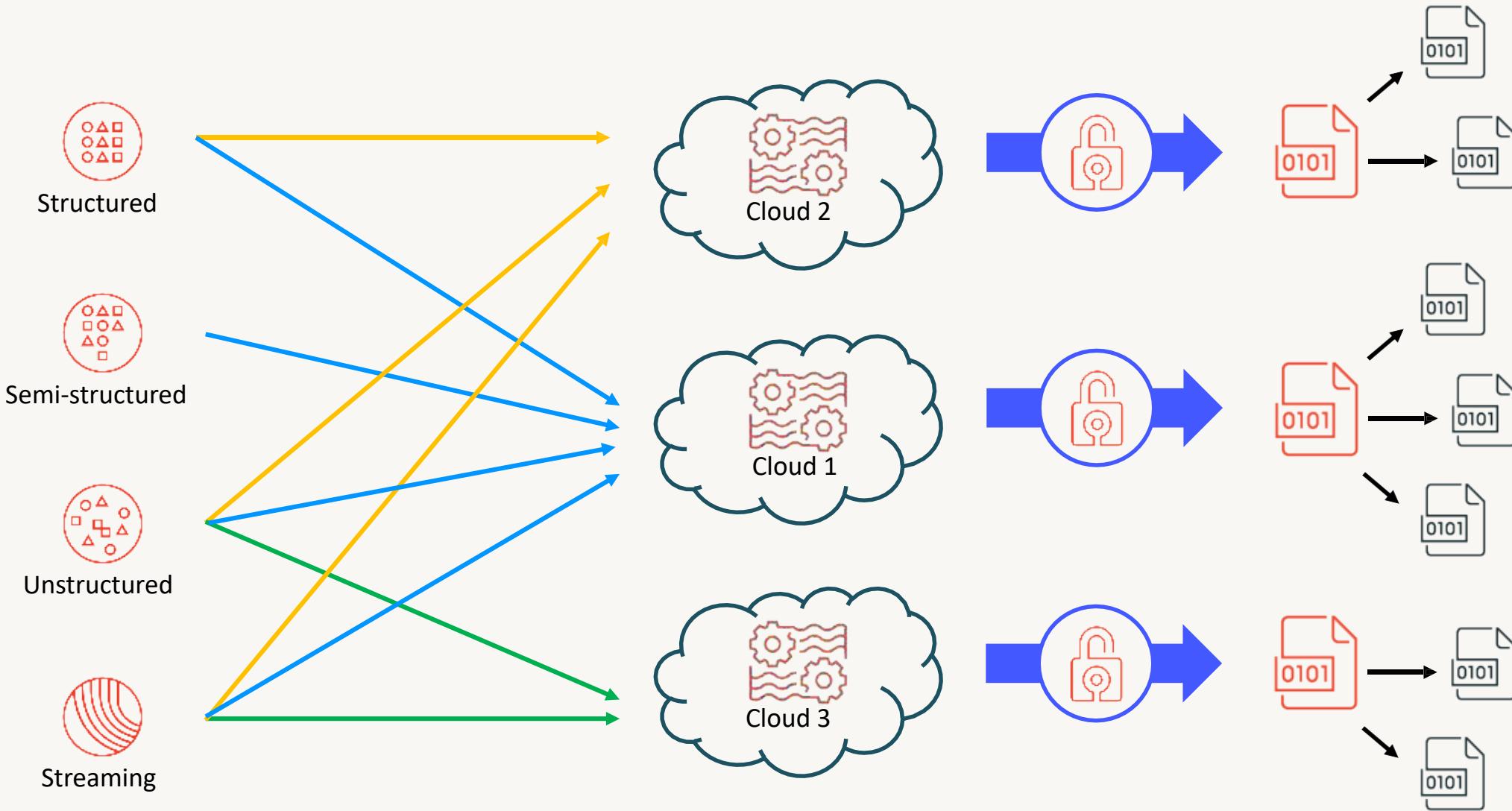
The screenshot shows the Databricks Registered Models interface for the 'Airline_Delay_SparkML' model. The left sidebar includes icons for Home, Workspace, Recents, Data, Clusters, Jobs, Models, and Search. The main area displays the model's registration details: Created Time: 2019-10-10 15:20:29 and Last Modified: 2019-10-14 12:17:04. A 'Description' section states: 'Predicts airline delays (in minutes) using the best Spark RF model from the AutoML Toolkit.' Below this is a 'Versions' section with tabs for 'All' (selected) and 'Active(1)'. The table lists five versions:

Version	Registered at	Created by	Stage	Pending Requests
Version 1	2019-10-10 15:20:30	clemens@demo.com	Archived	-
Version 2	2019-10-10 21:47:29	clemens@demo.com	Archived	-
Version 3	2019-10-10 23:39:43	clemens@demo.com	Production	-
Version 4	2019-10-11 09:55:29	clemens@demo.com	None	-
Version 5	2019-10-11 12:44:44	matei@demo.com	Staging	1

MLflow Model Registry provides all the information about modern lineage, model versioning, present condition, workflow, and stage transition (whether promoted to production or archived).

Governance requirements for
data are quickly evolving

Governance is hard to enforce on data lakes



The problem is getting bigger

Enterprises need a way to share and govern a wide variety of data products



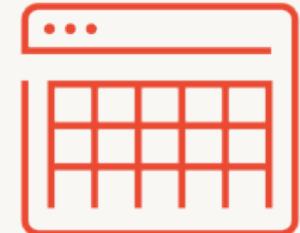
Files



Dashboards



Models



Tables

Unity Catalog for Lakehouse Governance

The screenshot shows the Unity Catalog interface for a table named 'Firehose'. The left sidebar contains navigation icons and a search bar. The main panel has the following sections:

- Description:** This table contains raw events from across the LOC platform. Use this table to find all reported game-play events.
- Recommendations:** Creating an index on ProductID may improve performance. Learn more. Firehose has not been vaccinated in 90 days. Learn more.
- Owners:** SP, AA, PA, JS
- Frequent users:** BU, BT, SB, AD
- ABAC Policies:** PII, Cost, Inventory
- Top Queries:** Champions stats 1H 2021, Gameplay analysis Q2 2021, Gameplay hours LOC semi-finals, Purchase prediction model 2021
- Statistics:** Format: Delta, Updated 5 hours ago, Created 1 year ago, Size: 500 GB

Schema:

Column	Type
ProductID	integer
Status	string
PricingTier	float
DistributionTier	string
AccountInfo	string

Lineage: A diagram showing data flow from 'inbndcall' to 'Firehose', which then feeds into 'disp_rec' and 'inventory'.

Privileges:

User / Group	Permissions	Modified
bi_team	Select, Modify, Manage	2021-05-20 15:56
dev	Select	2021-05-12 10:25

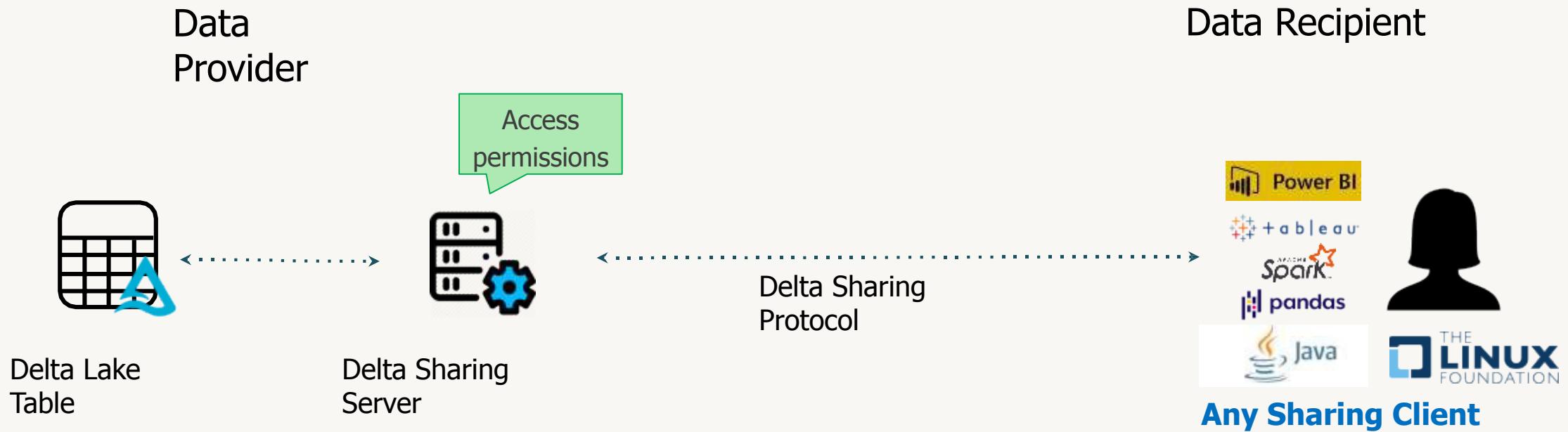
Data Sample:

ProductID	Status	PricingTier	DistributionTier	AccountInfo
135018	active	Enterprise	P1	15

- **Centrally catalog, Search, and discover** data and AI assets
- Simplify governance with a **unified Cross- cloud governance model**
- **Easily integrate** with your existing Enterprise Data Catalogs
- **Securely share live data** across platforms with delta sharing



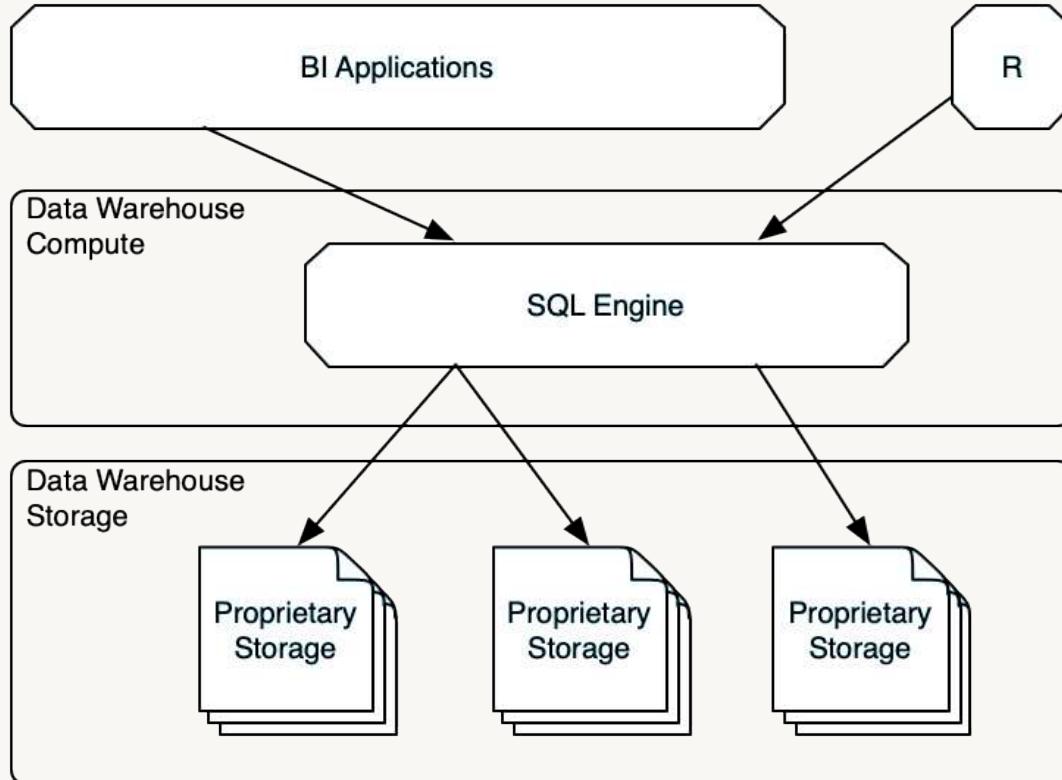
Delta Sharing on Databricks



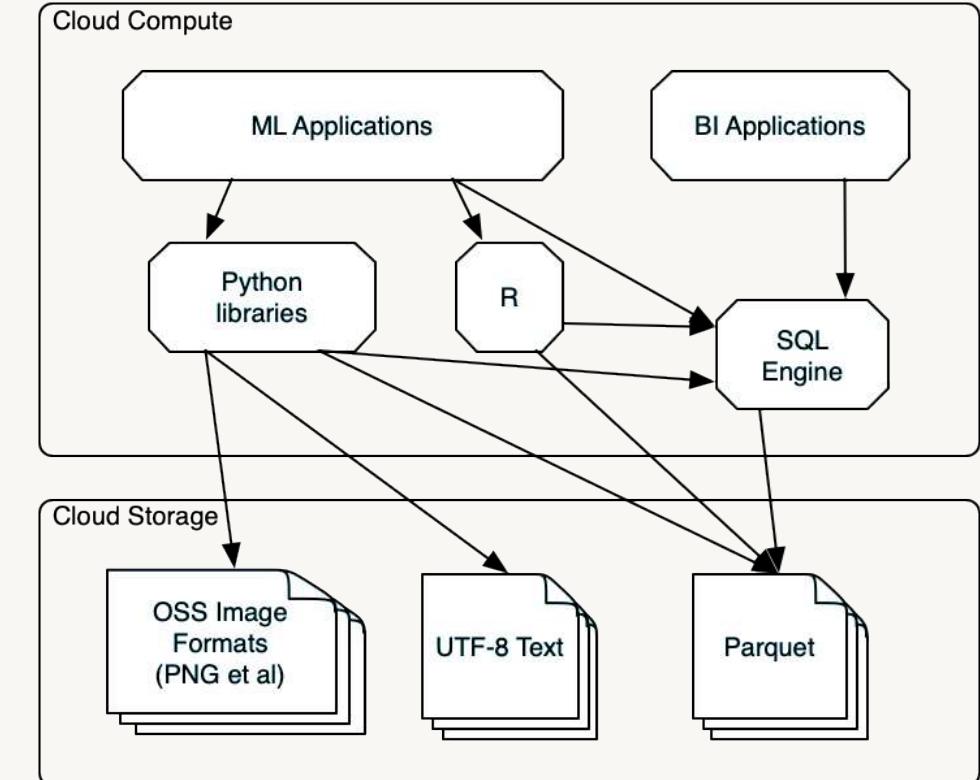
Machine Learning Workspace

ML Architecture: Data Warehouse VS Data Lakehouse

Data Warehouse

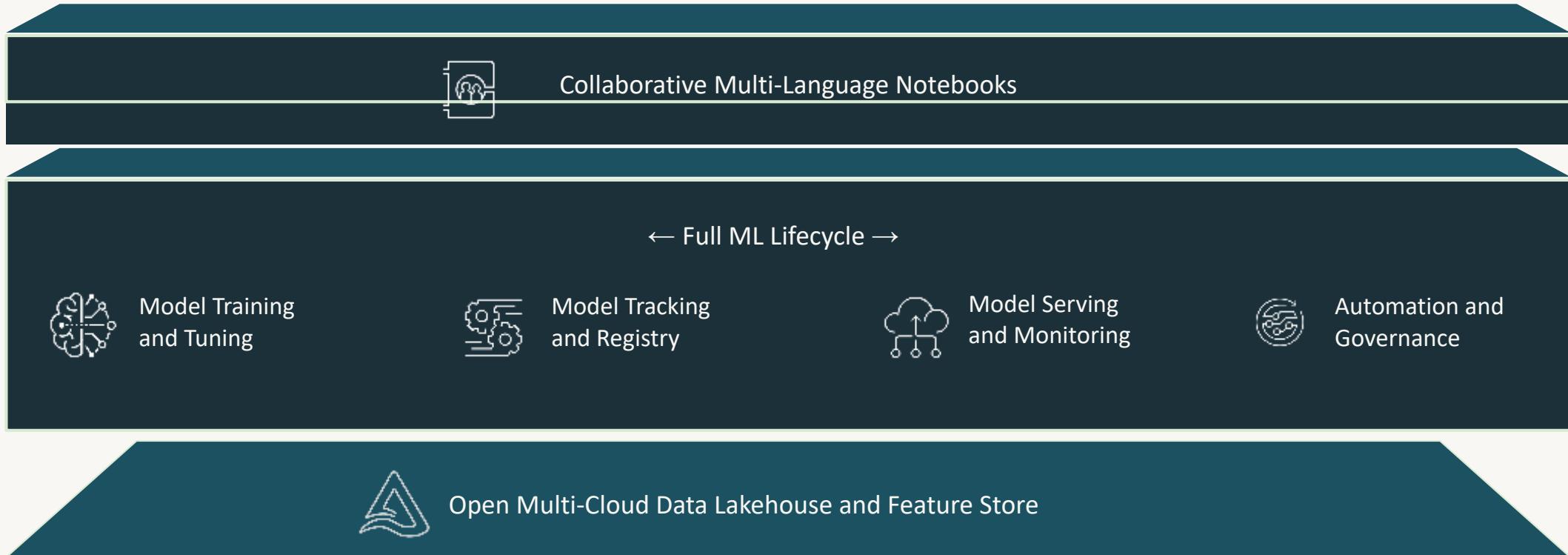


Data Lakehouse



Data Science and Machine Learning

A data-native and collaborative solution for the full ML lifecycle



What Does ML Need from a Lakehouse?

Access to Unstructured Data

- Images, text, audio, custom formats
- Libraries understand files, not tables
- Must *scale* to petabytes

Open Source Libraries

- OSS dominates ML tooling (Tensorflow, scikit-learn, xgboost, R, etc)
- Must be able to apply these in Python, R

Specialized Hardware, Distributed Compute

- Scalability of algorithms
- GPUs, for deep learning
- Cloud elasticity to manage that cost!

Model Lifecycle Management

- Outputs are model artifacts
- Artifact lineage
- Productionization of model

Three Data Users



Business Intelligence

- SQL and BI tools
- Prepare and run reports
- Summarize data
- Visualize data
- (Sometimes) Big Data
- Data Warehouse data store



Data Science

- R, SAS, some Python
- Statistical analysis
- Explain data
- Visualize data
- Often small data sets
- Database, data warehouse data store; local files



Machine Learning

- Python
- Deep learning and specialized GPU hardware
- Create predictive models
- Deploy models to prod
- Often big data sets
- Unstructured data in files

How Is ML Different?

- Operates on **unstructured** data like text and images
- Can require learning from **massive data** sets, not just analysis of a sample
- Uses **open source** tooling to manipulate data as “DataFrames” rather than with SQL
- Outputs are **models** rather than data or reports
- Sometimes needs **special hardware**



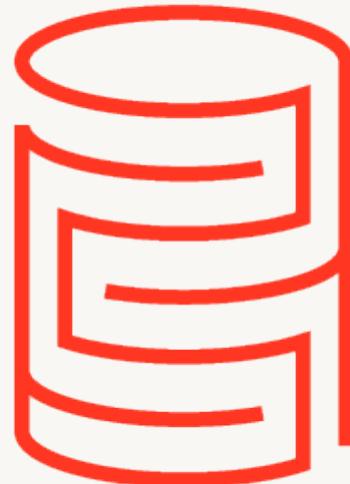
MLOps and the Lakehouse

- Applying open tools in-place to data in the lakehouse is a win for *training*
- Applying them for *operating* models is important too!
- "Models are data too"
- Need to apply models to *data*
- **MLFlow** for MLOps on the lakehouse
 - Track and manage model data, lineage, inputs
 - Deploy models as lakehouse "services"



Feature Stores for Model Inputs

- Tables are OK for managing model input
 - Input often structured
 - Well understood, easy to access
- ... but not quite enough
 - Upstream lineage: how were features computed?
 - Downstream lineage: where is the feature used?
 - Model caller has to read, feed inputs
 - How to do (also) access in real time?



SQL Analytics Workspace

Query data lake data using familiar **ANSI SQL**, and find and share new insights faster with the built-in SQL query editor, alerts, visualizations, and interactive dashboards.

Databricks Workspaces: Queries

Provides a simplified control (which is SQL only) to query the data

The screenshot shows the Databricks Workspaces interface with a dark sidebar on the left containing navigation icons for Dashboards, Queries, Endpoints, History, Alerts, and Help.

The main area is titled "New Query". It displays a dropdown menu for "TestSQLEndpoint" and a table selection dropdown for "default". Below these are the table details for "nyc_taxi".

The query editor contains the following SQL code:

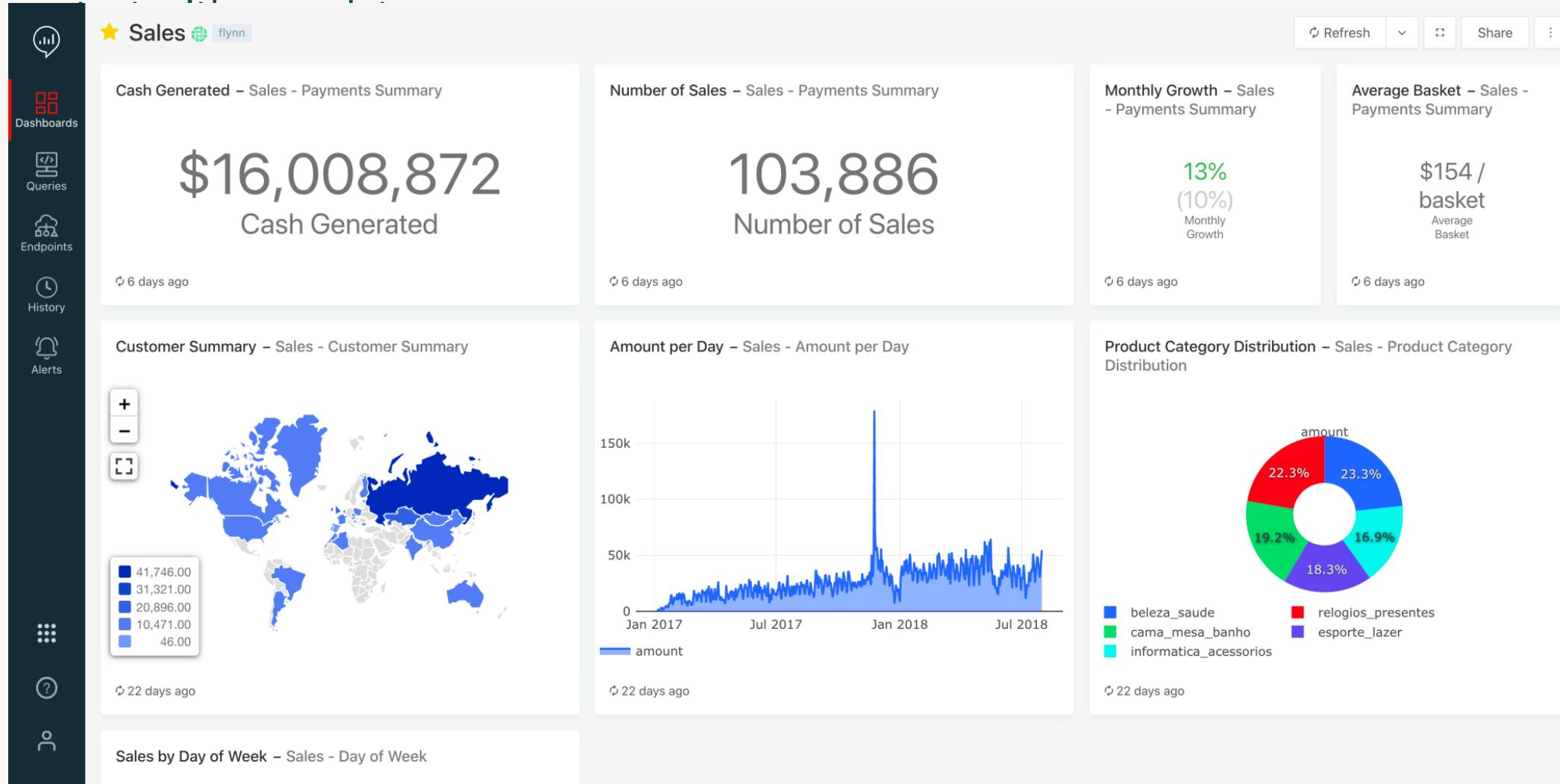
```
1 select
2   to_date(pickupDateTime), count(*)
3   from default.nyc_taxi
4   group by pi|
```

A tooltip is displayed over the "pi" column name, listing several keywords: pickupDateTime, puLocationId, dropOffDateTime, primary, partition, and explain.

At the bottom of the editor, there are three small icons and a checkbox labeled "LIMIT 1000".

Databricks Workspaces: Dashboards

A Databricks SQL dashboard lets you combine visualizations and text boxes that provide



Databricks Workspaces: Alerts

Alerts notify you when a field returned by a scheduled query meets a threshold. Alerts complement scheduled queries, but their criteria are checked after every execution.

New Alert

Start by selecting the query that you would like to monitor using the search bar. Keep in mind that Alerts do not work with queries that use parameters.

Query:

Setup Instructions [?](#)

Create Alert

Start by selecting the query that you would like to monitor using the search bar. Keep in mind that Alerts do not work with queries that use parameters.

Query: (This query has no refresh schedule. [Why It's recommended](#))

Value column Condition Threshold
Trigger when: value > 1

When triggered, send notification: Just once

Template: Use default template

Create Alert

Alerts

Name	Created By	State	Created At
Count scheduled queries that return no data #ops: count(0) greater than 0		TRIGGERED since 2 years ago	2 years ago
Adhoc Queues Query Counts: count(0) greater than 80		OK since 3 months ago	8 months ago
Seen Record Count: count greater than 1000		OK since 8 months ago	3 years ago
Previous Day Beacon Reports: count less than 3400		OK since a year ago	a year ago
Users created in the past hour > 10		OK since 4 days ago	11 days ago
Accounts created in the past hour > 10		OK since 11 days ago	11 days ago
No queries created		OK since 3 years ago	3 years ago
Query Results growing again		TRIGGERED since 8 months ago	a year ago
Is Brian back?: count greater than 0		OK since a month ago	a month ago
Generic Query : health_level greater than 30		OK since 4 days ago	4 days ago

Generic Alert

STATUS: TRIGGERED
Last triggered 7 minutes ago

Query: [Generic Query 1](#) (Scheduled to refresh every minute)

Value column Condition Threshold
Trigger when: value = 1808

Top row value is 1808

Destinations [?](#) + Add Toggle

<input type="checkbox"/> @ [REDACTED]	X
<input type="checkbox"/> #ops	X
<input type="checkbox"/> #Platform	X
<input type="checkbox"/> Test Webhook	X

Notifications: Notifications are sent just once, until back to normal. Set to default notification template.

Databricks Workspaces: Query History

The query history shows SQL queries performed using SQL endpoints.

The screenshot illustrates the Databricks Workspaces interface, specifically the Query History feature. On the left, the sidebar includes options like Create, SQL Editor, Queries, Dashboards, Alerts, Data, SQL Endpoints, and the highlighted Query History. The main area displays a table of query logs. A specific query entry for "TPC-H Q1" is selected and expanded into a detailed view. This view shows the query ID (598b82bf-510f-438e-8da3-37fa6bbf9a2c), the status (Finished), the endpoint (Data Engineering endpoint), and the query text:

```
-- TPC-H Q1
select
    l_returnflag,
    l_linenumber,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_qty,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    samples.tpch.lineitem
where
```

Below the query text, a "Duration" section provides a breakdown of execution time:

Duration Type	Time	Percentage
Total	25.45 s	
Optimizing query & pruning files	8.86 s	35%
Execution	16.34 s	64%
Result fetching	257 ms	1%

The "Overview" section indicates the SQL Endpoint used was the Data Engineering endpoint. The "Execution summary" section provides a detailed log of the query's execution metrics:

Metric	Value
Start - End time	12:01:59.799 - 12:02:25.596
Task total time	1.44 m
Rows returned	4 (29,999,795 read)
Files read	6
Partitions read	0
Bytes read	252.37 MB
Bytes spilled to disk	0 bytes

A blue button at the bottom right of the expanded view says "View full execution details".

Thank you

