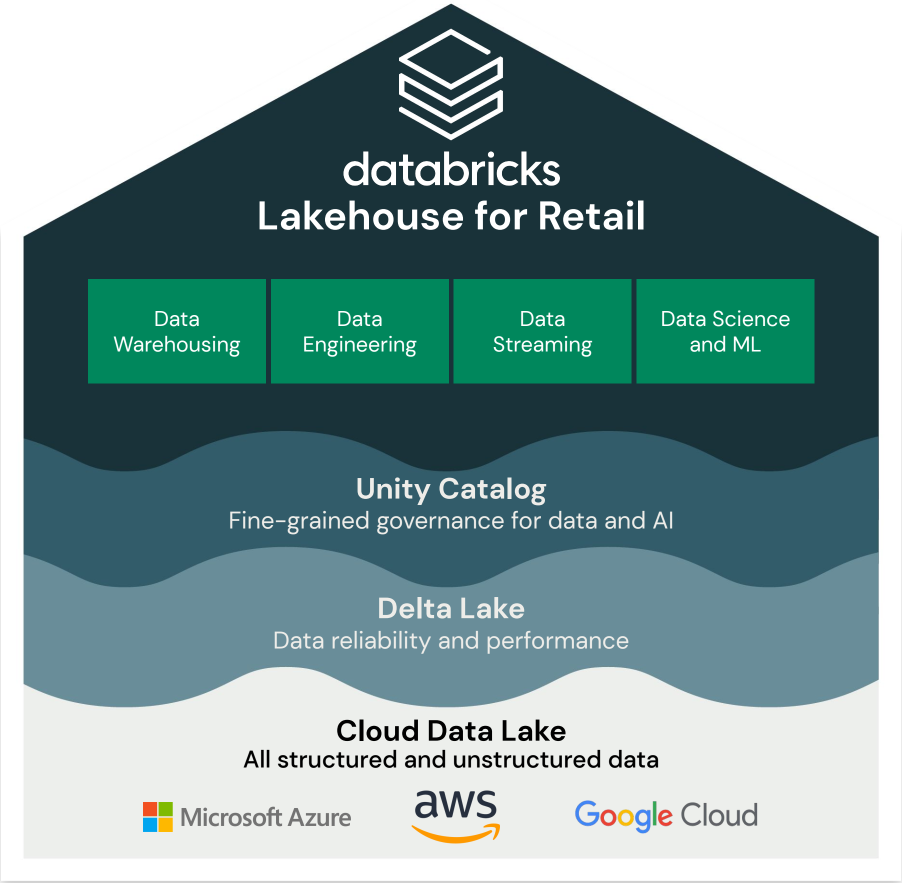
Delta Live Tables



Databricks Lakehouse for Retail

Simple

Unify your data warehousing and AI use cases on a single platform

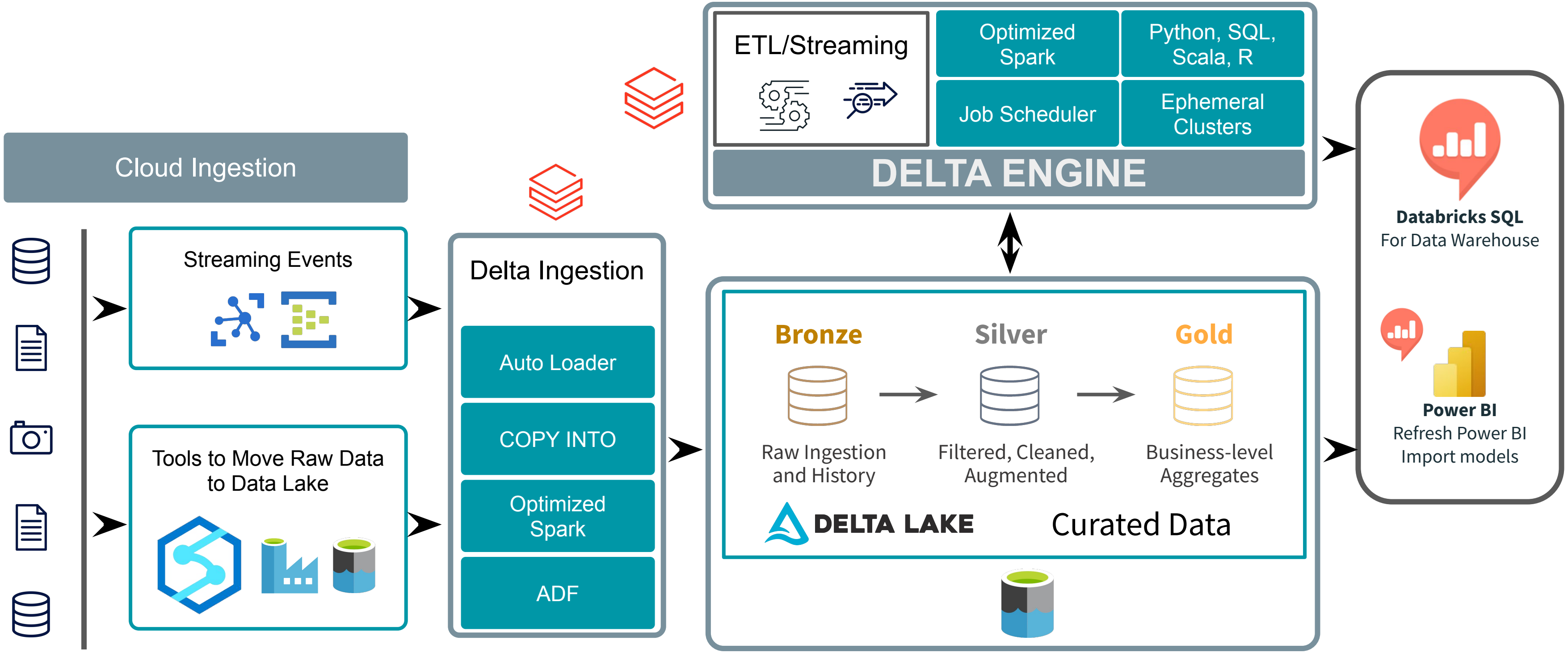
Open

Built on open source and open standards

Multi-cloud

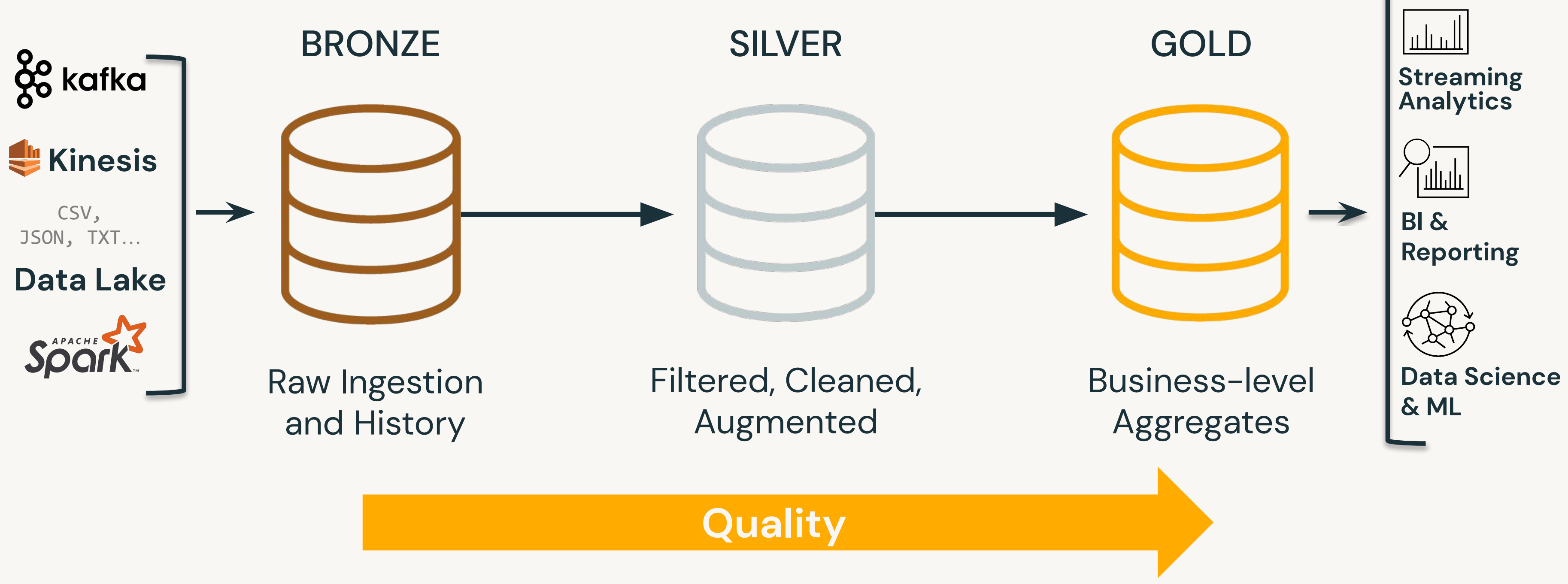
One consistent data platform across clouds

Data Engineering and Real Time Data Applications



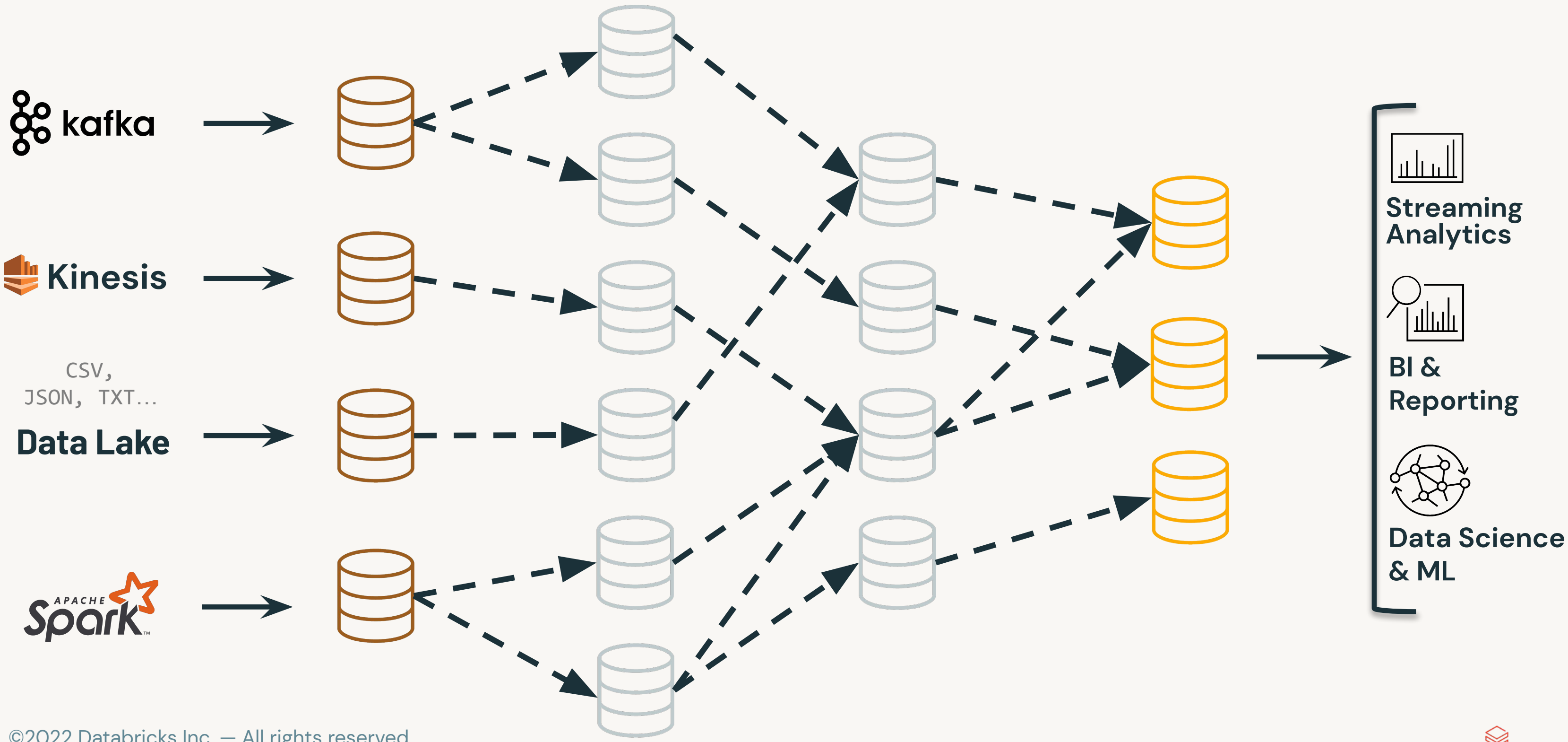
Building the foundation of the Lakehouse

Greatly improve data quality for end users – on your existing data lake



But the reality is not so simple

Maintaining data quality and reliability at scale is complex and brittle



Introducing “Workflows”

From

“Jobs”

“Cron for Databricks Spark”

To

“Workflows” containing

- Jobs
- Delta Live Tables

General purpose orchestration for the entire Lakehouse

…more than Spark!

Delta Live Tables

What is Delta Live Tables?

Modern software engineering for ETL processing

Delta Live Tables (DLT) is the first ETL framework that uses a simple, declarative approach to building reliable data pipelines. DLT automatically manages your infrastructure at scale so data analysts and engineers can spend less time on tooling and focus on getting value from data.

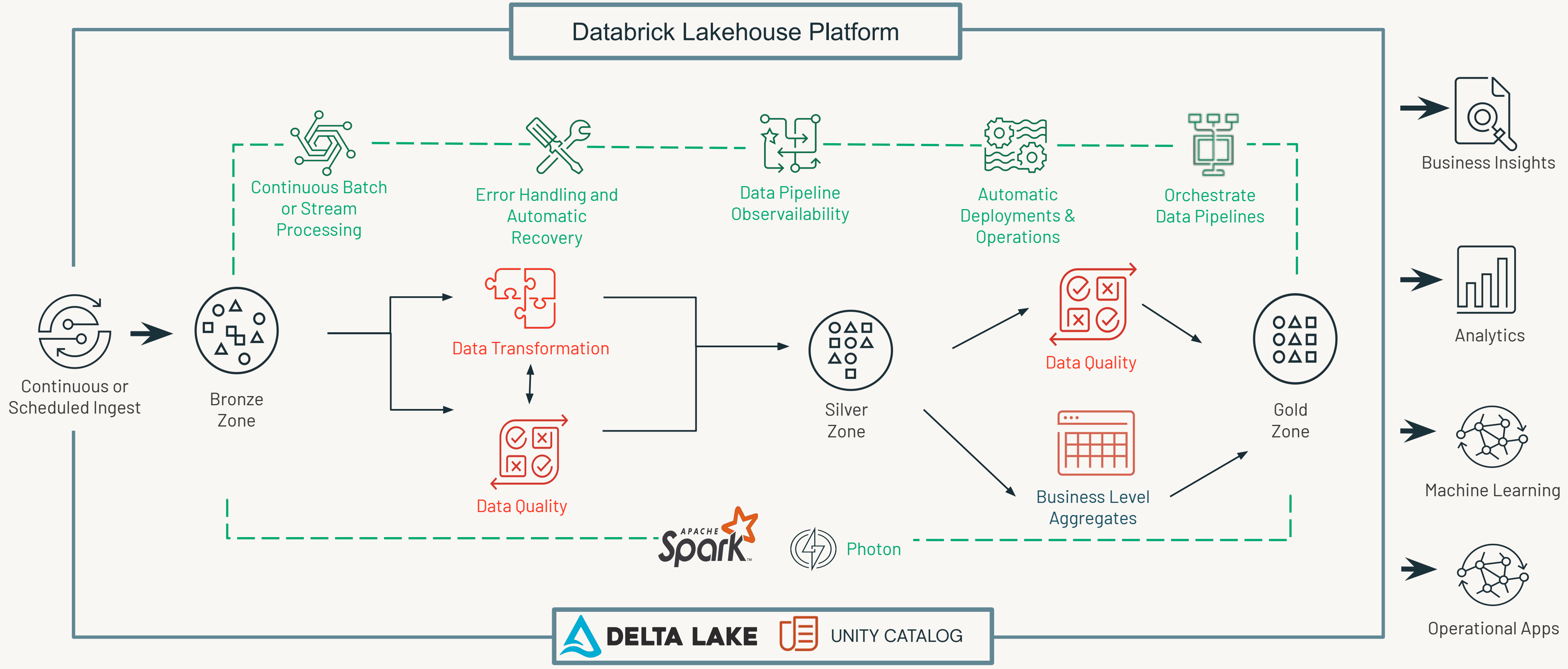
Accelerate ETL Development

Automatically manage your infrastructure

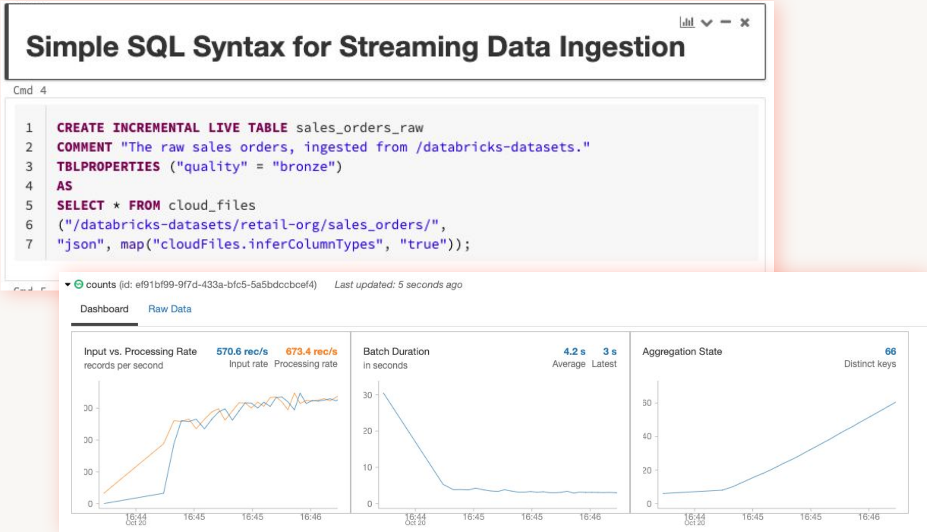
Have confidence in your data

Simplify batch and streaming

Build Production ETL Pipelines with DLT


Databrick Lakehouse Platform
Continuous Batch or Stream Processing
Error Handling and Automatic Recovery
Data Pipeline Observability
Automatic Deployments & Operations
Orchestrate Data Pipelines
Continuous or Scheduled Ingest
Bronze Zone
Data Transformation
Data Quality
Silver Zone
Data Quality
Business Level Aggregates
Gold Zone
APACHE Spark
Photon
DELTA LAKE
UNITY CATALOG
Business Insights
Analytics
Machine Learning
Operational Apps

Continuous or scheduled data ingestion



Simple SQL Syntax for Streaming Data Ingestion

Cmd 4

```
CREATE INCREMENTAL LIVE TABLE sales_orders_raw
COMMENT "The raw sales orders, ingested from /databricks-datasets."
TBLPROPERTIES ("quality" = "bronze")
AS
SELECT * FROM cloud_files
("/databricks-datasets/retail-org/sales_orders/",
"json", map("cloudFiles.inferColumnTypes", "true"));
```

counts (id: ef91bf99-9f7d-433a-bfc5-5a5bdccbcef4) *Last updated: 5 seconds ago*

- Incrementally and efficiently process new data files as they arrive in cloud storage using Auto Loader
- Automatically infer schema of incoming files or superimpose what you know with Schema Hints
- Automatic schema evolution
- Rescue data column – never lose data again

Schema Evolution

✅ JSON

✅ CSV

✅ AVRO

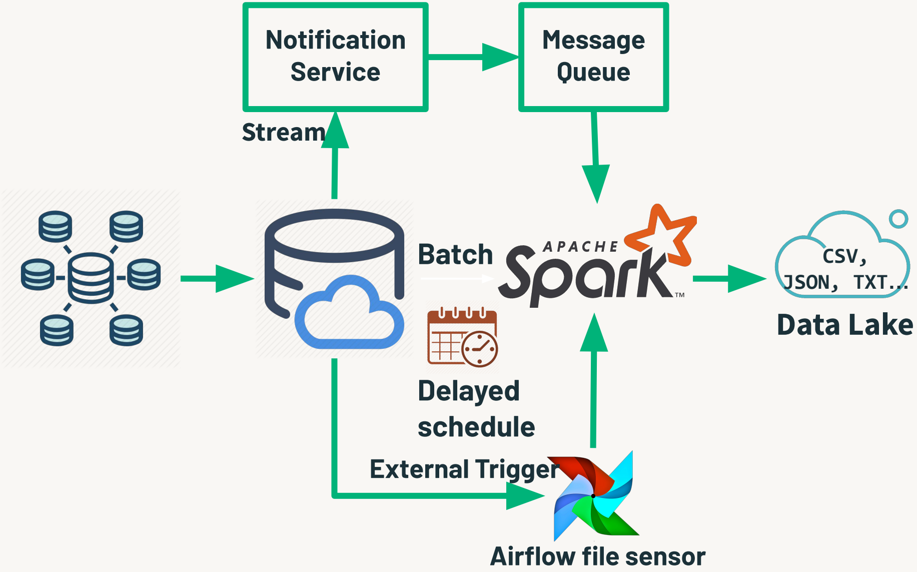
Coming Soon

PARQUET

Databricks Ingest: Auto Loader

Load new data easily and efficiently as it arrives in cloud storage

Before



Notification Service

Message Queue

Stream

Batch

Apache Spark

CSV, JSON, TXT...

Data Lake

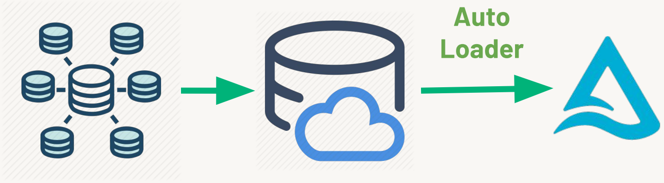
Delayed schedule

External Trigger

Airflow file sensor

Gets too complicated for multiple jobs

After

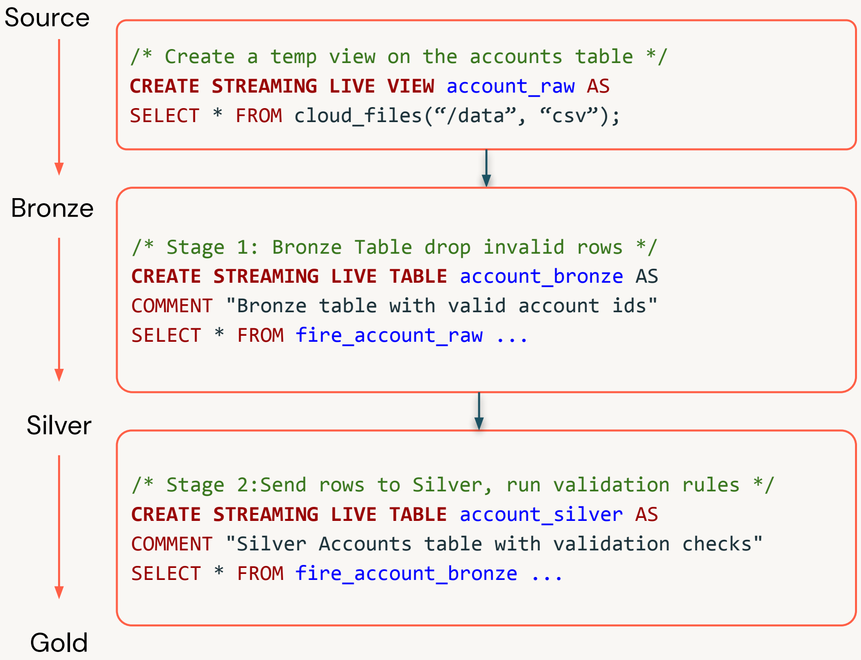


Auto Loader

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

Launch Blog Post

Declarative SQL & Python APIs



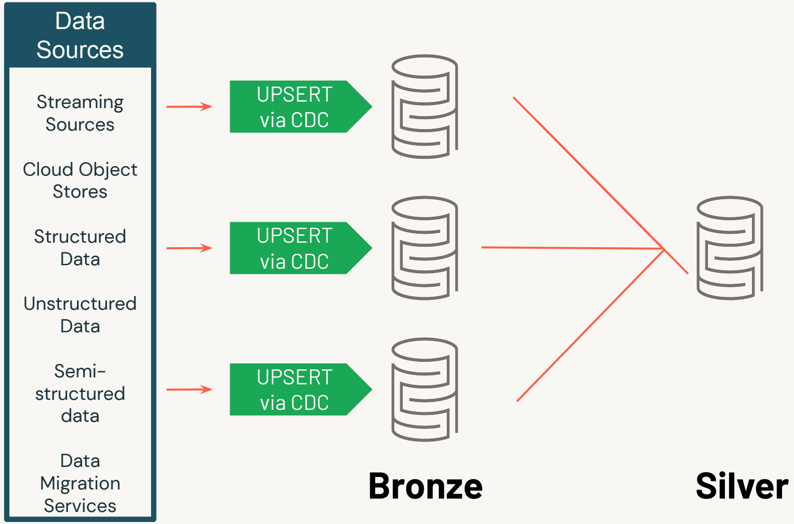
```sql
/* Create a temp view on the accounts table */
CREATE STREAMING LIVE VIEW account_raw AS
SELECT * FROM cloud_files(“/data”, “csv”);
```

```sql
/* Stage 1: Bronze Table drop invalid rows */
CREATE STREAMING LIVE TABLE account_bronze AS
COMMENT "Bronze table with valid account ids"
SELECT * FROM fire_account_raw ...
```

```sql
/* Stage 2:Send rows to Silver, run validation rules */
CREATE STREAMING LIVE TABLE account_silver AS
COMMENT "Silver Accounts table with validation checks"
SELECT * FROM fire_account_bronze ...
```

- Use intent-driven declarative development to abstract away the **“how”** and define **“what”** to solve
- Automatically generate **lineage** based on table dependencies across the data pipeline
- Automatically checks for errors, missing dependencies and syntax errors

Change data capture (CDC)



- Stream change records (inserts, updates, deletes) from any data source supported by DBR, cloud storage, or DBFS
- Simple, declarative “APPLY CHANGES INTO” API for SQL or Python
- Handles out-of-order events
- Schema evolution
- SCD2 support

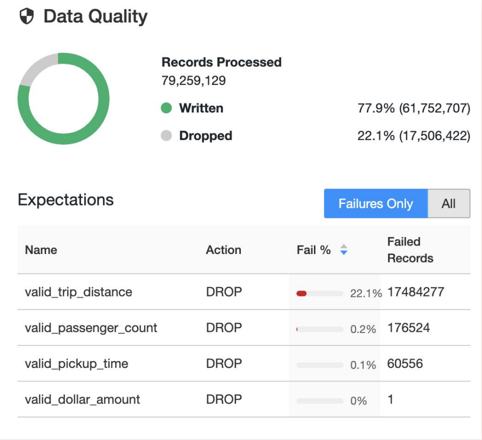
Data quality validation and monitoring

- Define data quality and integrity controls within the pipeline with data expectations

- Address data quality errors with flexible policies: fail, drop, alert, quarantine(future)

- All data pipeline runs and quality metrics are captured, tracked and reported

```
/* Stage 1: Bronze Table drop invalid rows */
CREATE STREAMING LIVE TABLE fire_account_bronze AS
( CONSTRAINT valid_account_open_dt EXPECT (acconut_dt is not null
and (account_close_dt > account_open_dt)) ON VIOLATION DROP ROW
COMMENT "Bronze table with valid account ids"
SELECT * FROM fire_account_raw ...
```



Data Quality

Records Processed
79,259,129

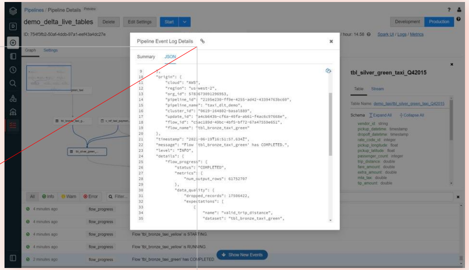
● Written 77.9% (61,752,707)

● Dropped 22.1% (17,506,422)

Expectations

Name	Action	Fail %	Failed Records
valid_trip_distance	DROP	22.1%	17484277
valid_passenger_count	DROP	0.2%	176524
valid_pickup_time	DROP	0.1%	60556
valid_dollar_amount	DROP	0%	1

Data pipeline observability

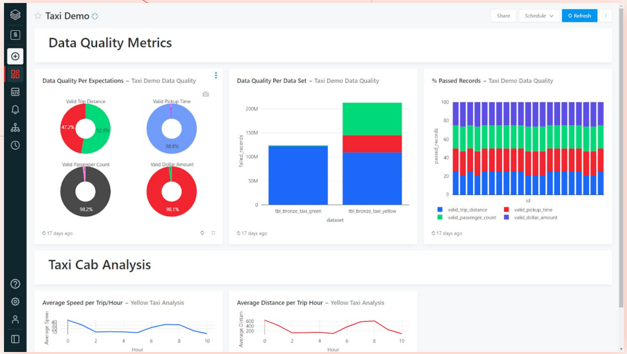


Pipeline Event Log Details

Summary JSON

```
    },
    "origin": {
        "cloud": "AWS",
        "region": "us-west-2",
        "org_id": 5783673891296953,
        "pipeline_id": "2195e230-ff9e-4255-ad42-43394763bc60",
        "pipeline_name": "taxi_dlt_demo",
        "cluster_id": "0619-164802-basal889",
        "update_id": "a4cb643b-cf6a-40fa-ab61-f4ac6c97668e",
        "flow_id": "c5ac189d-49bc-4bf5-bf72-67a47559e651",
        "flow_name": "tbl_bronze_taxi_green"
    },
    "timestamp": "2021-06-19T16:51:57.634Z",
    "message": "Flow 'tbl_bronze_taxi_green' has COMPLETED.",
    "level": "INFO",
    "details": {
        "flow_progress": {
            "status": "COMPLETED",
            "metrics": {
                "num_output_rows": 61752707
            },
            "data_quality": {
                "dropped_records": 17506422,
                "expectations": [
                    {
                        "name": "valid_trip_distance",
                        "dataset": "tbl_bronze_taxi_green",
```

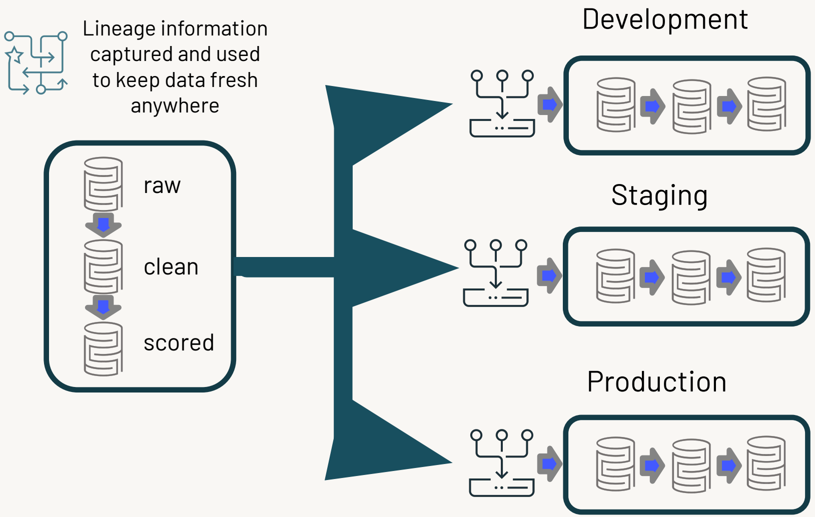
```
flow_progress   Flow 'tbl_bronze_taxi_yellow' is STARTING.
flow_progress   Flow 'tbl_bronze_taxi_yellow' is RUNNING.
flow_progress   Flow 'tbl_bronze_taxi_green' has COMPLETED.
```



- High-quality, high-fidelity lineage diagram that provides visibility into how data flows for impact analysis
- Granular logging for operational, governance, quality and status of the data pipeline at a row level
- Continuously monitor data pipeline jobs to ensure continued operation
- Notifications using Databricks SQL

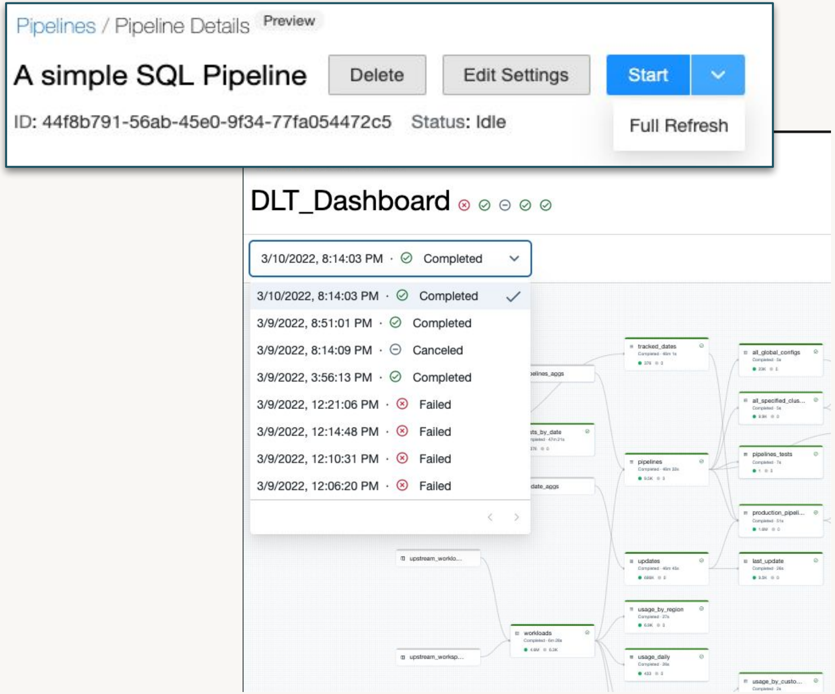
Automated ETL development lifecycle

- Develop in environment(s) separate from production with the ability to easily test it before deploying – entirely in SQL
- Deploy and manage environments using parameterization
- Unit testing and documentation
- Enables metadata-driven ability to programatically scale to 100s of tables/pipelines dynamically



Lineage information captured and used to keep data fresh anywhere
raw
clean
scored
Development
Staging
Production

Automated ETL operations



Pipelines / Pipeline Details Preview

A simple SQL Pipeline

ID: 44f8b791-56ab-45e0-9f34-77fa054472c5 Status: Idle

DLT_Dashboard

3/10/2022, 8:14:03 PM · Completed

- 3/10/2022, 8:14:03 PM · Completed ✓
- 3/9/2022, 8:51:01 PM · Completed
- 3/9/2022, 8:14:09 PM · Canceled
- 3/9/2022, 3:56:13 PM · Completed
- 3/9/2022, 12:21:06 PM · Failed
- 3/9/2022, 12:14:48 PM · Failed
- 3/9/2022, 12:10:31 PM · Failed
- 3/9/2022, 12:06:20 PM · Failed

- Reduce down time with automatic error handling and easy replay
- Eliminate maintenance with automatic optimizations of all Delta Live Tables
- Auto-scaling adds more resources automatically when needed.

DLT Demo