



# Getting started with Delta Live Tables (DLT) pipelines

Reliable data pipelines made easy

---

**Zoé Durand, Sr. Product Manager at Databricks**

**Maarten de Haas, Product Architect at Heineken**

©2024 Databricks Inc. — All rights reserved



# Life as an data engineer...



# Life as a data engineer...

Email from your CEO



# Life as an data engineer...

Email from your CEO



*“Hey, I’m really excited about this new genAI thing. Can we use genAI to help our support agents draft replies to customer queries?”*

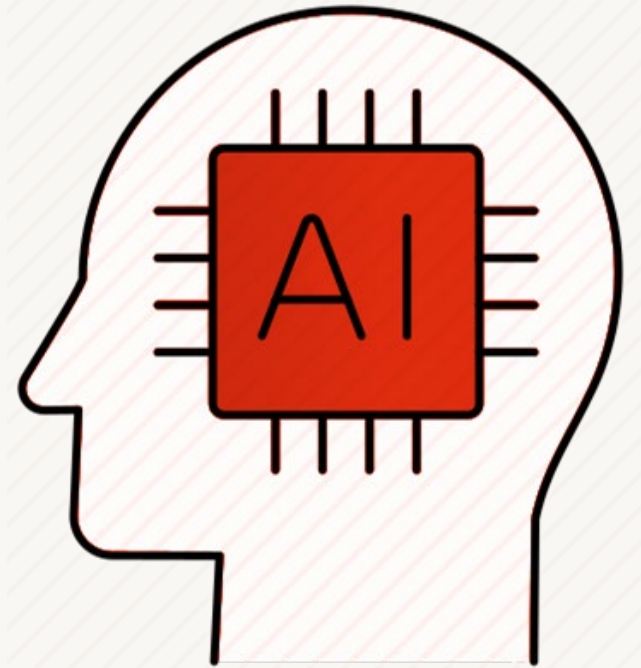
# Life as an data engineer...

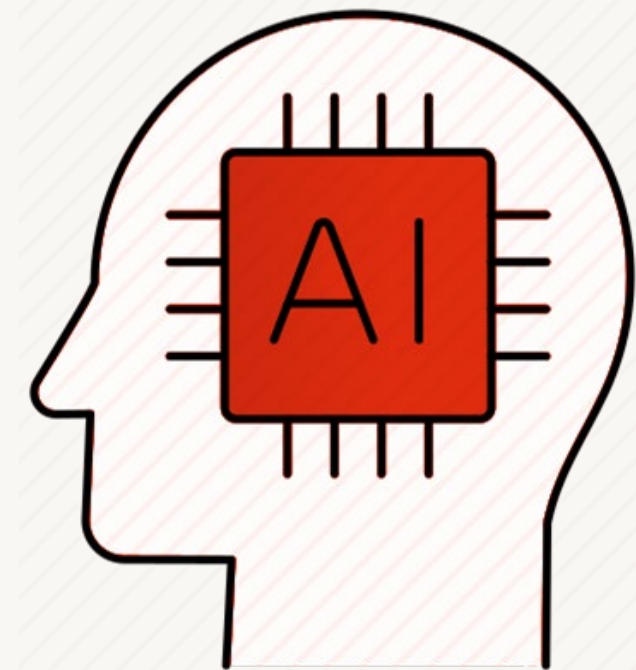
Email from your CEO





Hey, I have an issue, can you help me resolve it? When I look at my orders, it seems that....







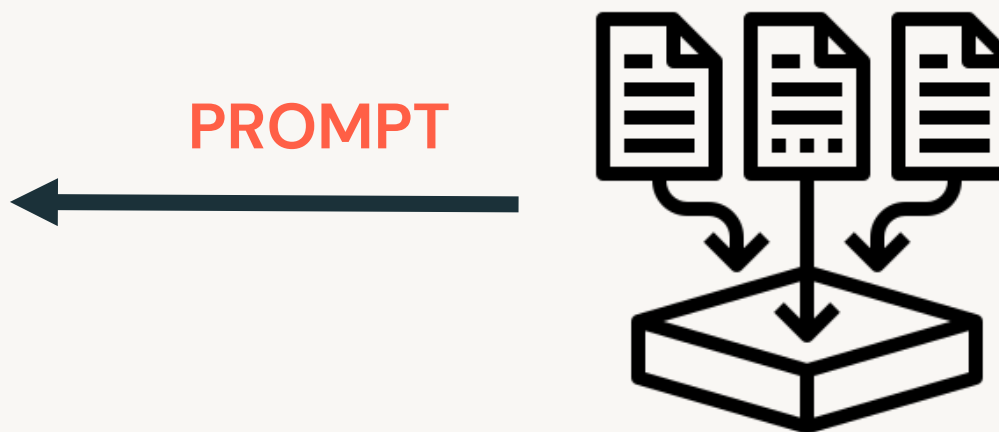


# Two key ingredients: a good model and good data

A good model...



...is only as good as the data it has access to



- Customer interaction logs
- Support tickets
- CRM systems

# Having fresh, clean, good quality data is hard



Version Control



Deployment Infrastructure



Quality Checks



Governance



Data Discovery

Backfill Handling



Dependency Management



```
CREATE TABLE raw_data as SELECT * FROM json.`...`  
CREATE TABLE clean_data as SELECT ... FROM raw_data
```

Machine learning applications

SQL analytics and BI

Daily Partition Computation

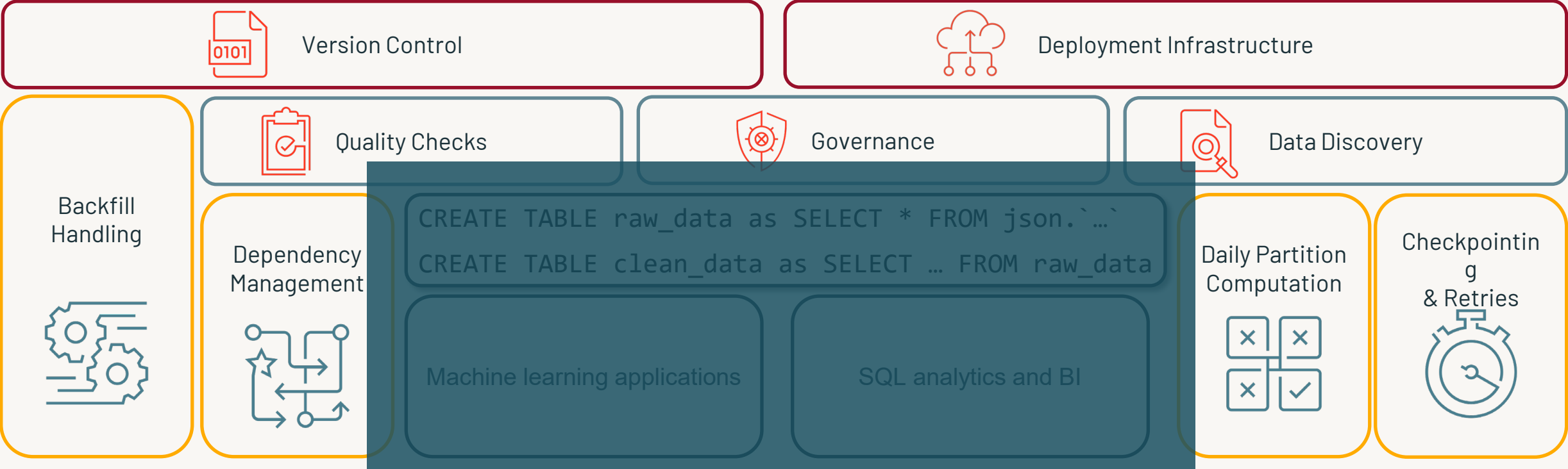


Checkpointing & Retries



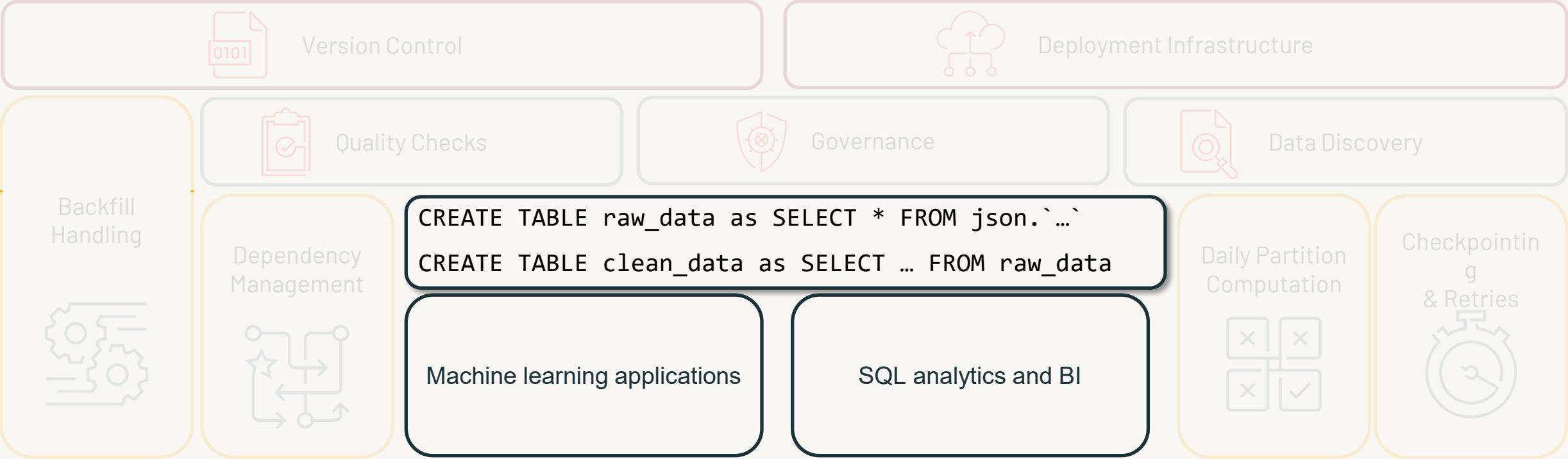
# Operational complexity dominates

Time is spent on **tooling** instead of on **transforming**



# Where you should focus your time

Getting value from data



# Data Pipelines Made Simple with DLT pipelines

DLT takes care of the operational challenges to help you focus only on business logic

- **Simple declarative programming** (SQL and Python)
  - Built-in incremental ingestion and transformations
  - Data quality
  - Easy data backfills
  - Automatic dependency parsing & orchestration
  - No checkpoint management
- **Easy infrastructure management**
  - CI/CD
  - Efficient autoscaling
  - Escalated retries
  - Fast startup times in Serverless
- **Easy monitoring** via the event log

## Delta Live Tables

```
CREATE STREAMING TABLE raw_data
AS SELECT *
FROM cloud_files ("/raw_data",
"json")
```

```
CREATE MATERIALIZED VIEW clean_data
AS SELECT ...
FROM raw_data
```



# DLT core concepts



# Streaming Tables and Materialized Views

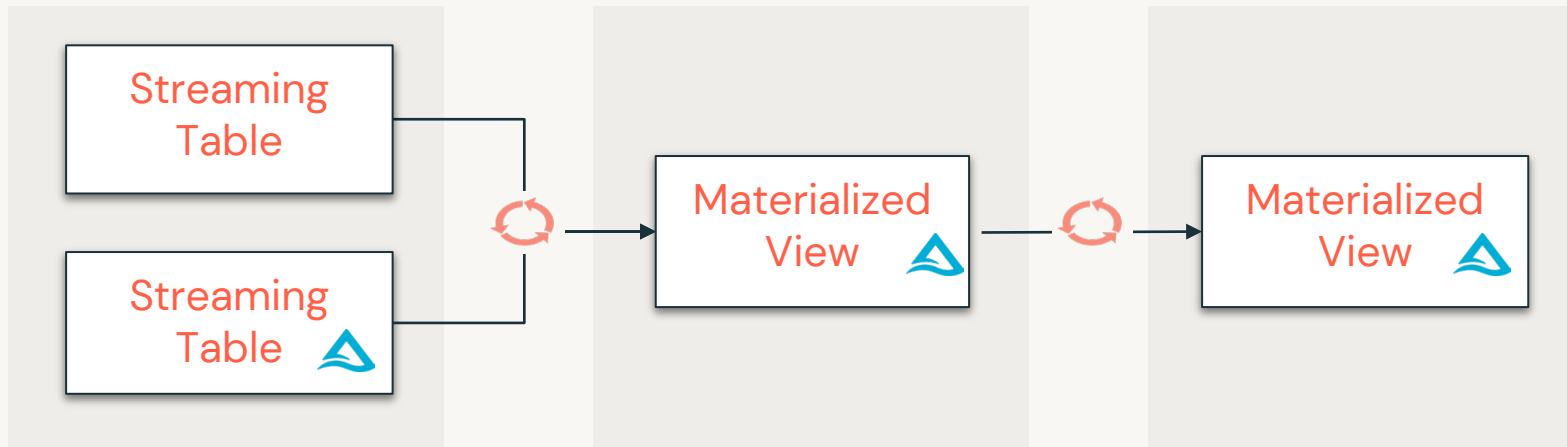
The building blocks of your pipelines

```
CREATE STREAMING TABLE
web_clicks
AS
SELECT *
FROM STREAM
  read_files('s3://mybucket')
```

```
CREATE MATERIALIZED VIEW
customer_orders
AS
SELECT
  customers.name,
  sum(orders.amount),
FROM orders
  LEFT JOIN customers ON
    orders.custkey =
customers.c_custkey
GROUP BY
  name;
```

# DLT Pipelines

Combine several Streaming Tables and Materialized Views with their dependencies



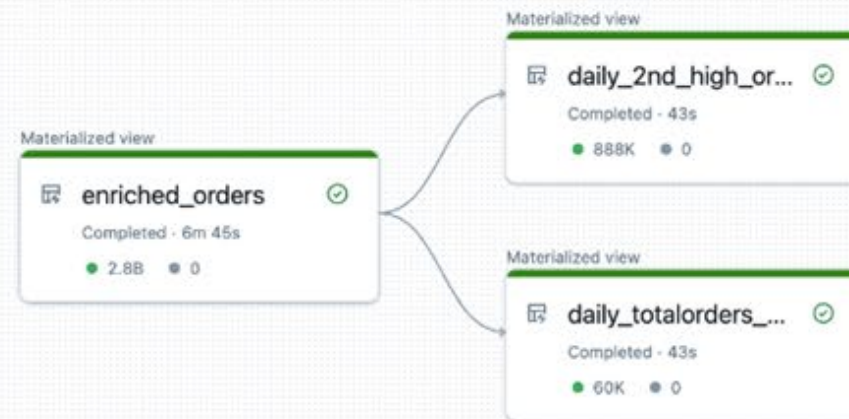


# Let's build a DLT pipeline



# Let's build a pipeline!

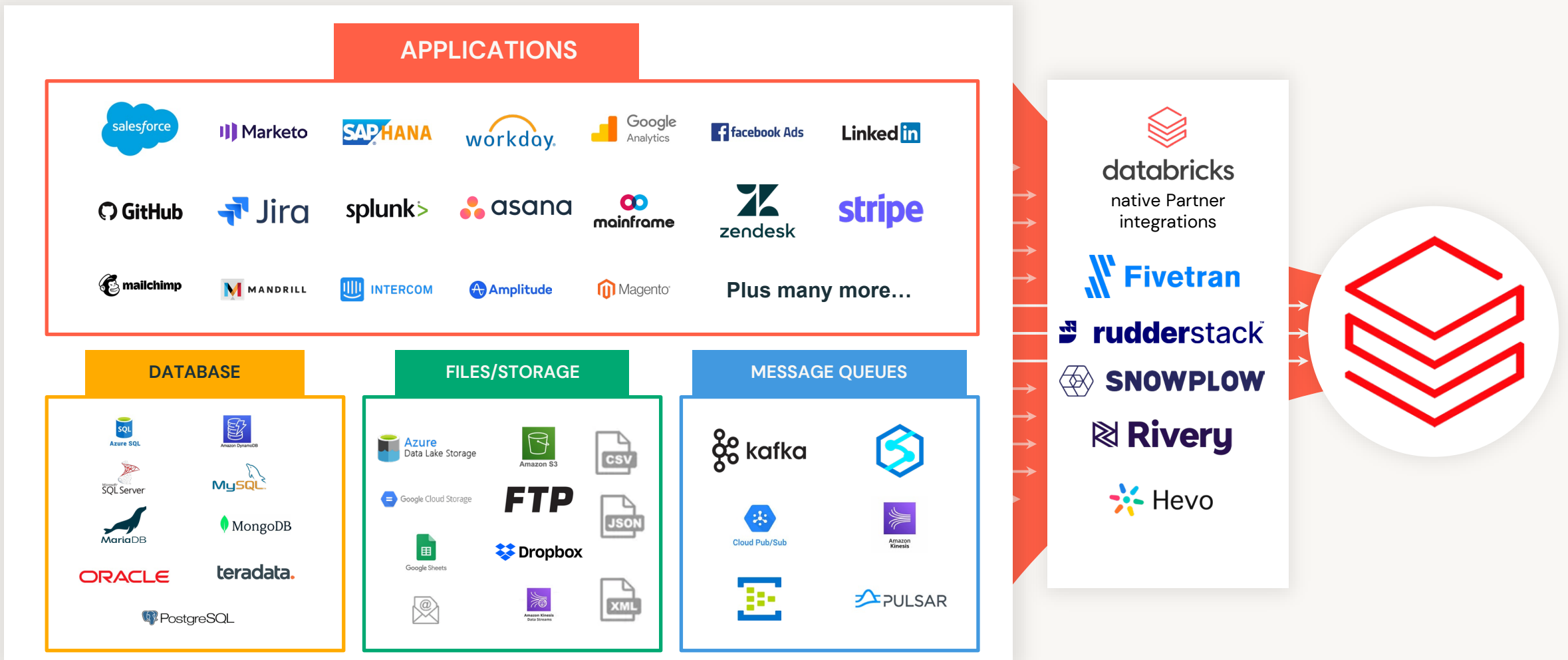
DLT graph DLT event log



1

Ingestion: landing data in the bronze layer

# Ingest data from any source



# Using Streaming Tables for ingestion

Easily ingest files from cloud storage as they are uploaded

```
CREATE STREAMING TABLE raw_data  
AS SELECT *  
FROM cloud_files("/data", "json")
```

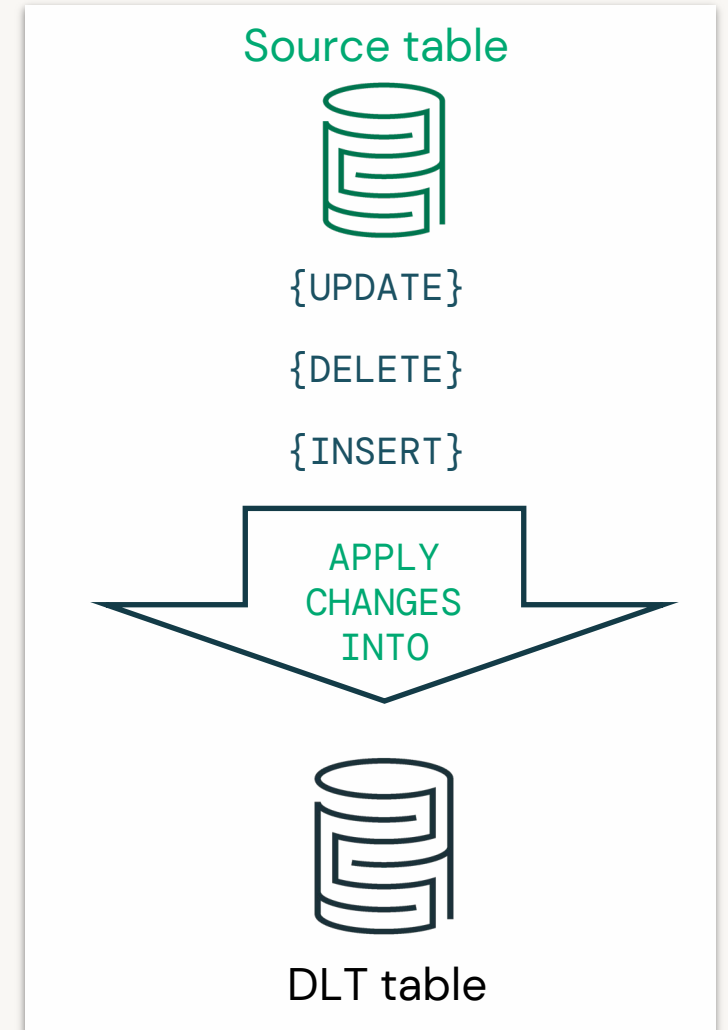
This example creates a table with all the json data stored in "/data":

- `cloud_files` keeps track of which files have been read to avoid duplication and wasted work
- Supports both listing and notifications for arbitrary scale

# Easily ingest **Change Data Capture (CDC)** data with **APPLY CHANGES INTO**

Maintain an up-to-date replica of a table stored elsewhere

```
CREATE STREAMING TABLE cities  
  
APPLY CHANGES INTO LIVE.cities  
FROM STREAM(city_updates)  
KEYS (id)  
SEQUENCE BY ts
```



2

# Transformation: from raw data to usable data

# Using Materialized Views for **transformation**

Easily transform **data** and process changes **incrementally**

```
CREATE MATERIALIZED VIEW silver_data
AS SELECT
date, country, count (distinct products)
FROM LIVE.raw_data
GROUP BY date, country
```

This example creates a table `silver_data` `silver_data` that is :

- Always correct given the source data and the query
- Incrementally refreshed: when new data arrives, **Enzyme** will choose the best technique to incrementally refresh the MV, avoiding time-consuming full recomputes



3

# Data quality

# Delta Live Tables expectations

## Manage and monitor data quality in real-time

### Retain Invalid Records

```
%sql
CREATE OR REPLACE LIVE TABLE orders_valid
(CONSTRAINT valid_timestamp EXPECT (timestamp > '2020-01-01'))
SELECT * FROM LIVE.orders_vw

%python
@dlt.expect("valid timestamp", "timestamp > '2012-01-01'")
```

Records violate expectations are  
Added to target table  
Flagged in invalid in event log  
Pipeline continues

### Drop Invalid Records

```
%sql
CREATE OR REPLACE LIVE TABLE orders_valid
(CONSTRAINT valid_timestamp EXPECT (timestamp > '2020-01-01') ON VIOLATION DROP ROW)
SELECT * FROM LIVE.orders_vw

%python
@dlt.expect_or_drop("valid timestamp", "timestamp > '2012-01-01'")
```

Records violate expectation are  
Dropped from target table  
Flagged invalid in event log  
Pipeline Continues

### Fail on Invalid Records

```
%sql
CREATE OR REPLACE LIVE TABLE orders_valid
(CONSTRAINT valid_timestamp EXPECT (timestamp > '2020-01-01') ON VIOLATION FAIL)
SELECT * FROM LIVE.orders_vw

%python
@dlt.expect_or_fail("valid timestamp", "timestamp > '2012-01-01'")
```

Records violate expectation  
Cause the job to fail

### Data Quality



#### Records Processed

79,259,129

● **Written** 77.9% (61,752,707)  
● **Dropped** 22.1% (17,506,422)

#### Expectations

Failures Only All

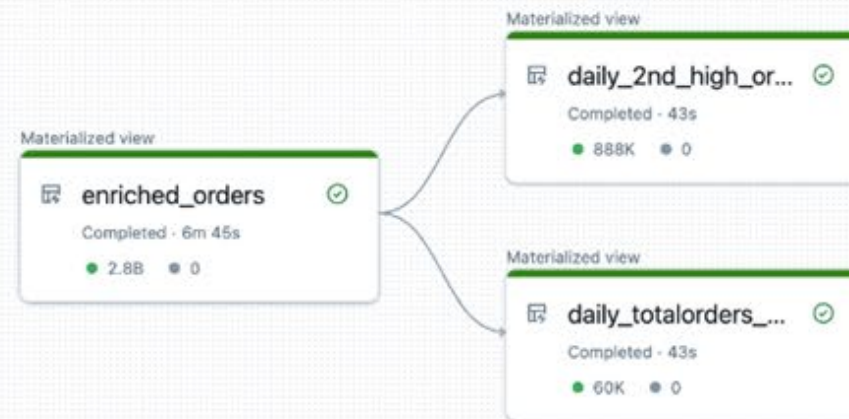
Name	Action	Fail %	Failed Records
valid_trip_distance	DROP	22.1%	17484277
valid_passenger_count	DROP	0.2%	176524
valid_pickup_time	DROP	0.1%	60556
valid_dollar_amount	DROP	0%	1

4

Development experience  
for an entire pipeline

# Let's build a pipeline!

DLT graph DLT event log



- New
- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
  - SQL Editor
  - Queries
  - Dashboards
  - Genie
  - Alerts
  - Query History
  - SQL Warehouses
- Data Engineering
  - Job Runs
  - Pipelines
- Machine Learning
  - Playground
  - Experiments
  - Features
  - Models
  - Serving
- Marketplace
- Partner Connect

# Daily-Orders-Trend Python

File Edit View Run Help Last edit was now Provide feedback

Run all Connected Schedule Share

```
1 Start typing or generate with AI (* + I)...
```

[Shift+Enter] to run and move to next cell  
 [Esc H] to see all keyboard shortcuts

+ New

## Daily-Orders-Trend

Python

File Edit View Run Help [Last edit was now](#) [Provide feedback](#)

▶ Run all

● Connected

Share

Workspace

Recents

Catalog

Workflows

Compute

SQL

SQL Editor

Queries

Dashboards

Genie

Alerts

Query History

SQL Warehouses

Data Engineering

Job Runs

Pipelines

Machine Learning

Playground

Experiments

Features

Models

Serving

Marketplace

Partner Connect

```
1 %sql
2 -- SILVER
3 CREATE MATERIALIZED VIEW Enriched_Orders
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT
8 o.* ,
9 c_name, c_mktsegment,
10 n_name
11 FROM dais_2023_dlt.tpch.orders o
12 INNER JOIN dais_2023_dlt.tpch.customer ON c_custkey = o_custkey
13 INNER JOIN dais_2023_dlt.tpch.nation ON n_nationkey = c_nationkey
```

```
1 %sql
2 -- GOLD
3 CREATE MATERIALIZED VIEW Daily_TotalOrders_By_Nation
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT o_orderdate, n_name, sum(o_totalprice) as totalprice_by_nation, count(o_totalprice) as item_sold_by_nation
8 FROM Live.Enriched_Orders
9 GROUP BY o_orderdate, n_name
```

```
1 %sql
2 -- GOLD
3 CREATE MATERIALIZED VIEW Daily_2nd_high_OrderPrice
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT o_orderdate, o_totalprice as secondHighestOrderPrice
8 FROM /
```



+ New

Workspace

Recents

Catalog

Workflows

Compute

SQL

SQL Editor

Queries

Dashboards

Genie

Alerts

Query History

SQL Warehouses

Data Engineering

Job

ETL Pipeline

Ingestion Pipeline

Machine Learning

AutoML Experiment

Experiment

Model

Serving endpoint

Notebook

Git Folder

Data

Compute

Cluster

SQL Warehouse

App

SQL

Query

Dashboard

Legacy dashboard

Genie space

Alert

Data Engineering

Job

ETL Pipeline

Ingestion Pipeline

Machine Learning

AutoML Experiment

Experiment

Model

Serving endpoint

Python Last edit was 1 minute ago Provide feedback Run all Connected Share

```
%sql
-- SILVER
CREATE MATERIALIZED VIEW Enriched_Orders
PARTITIONED BY (o_orderdate)
TBLPROPERTIES (delta.enableChangeDataFeed = true)
AS
SELECT
o.* ,
c_name, c_mktsegment,
n_name
FROM dais_2023_dlt.tpch.orders o
INNER JOIN dais_2023_dlt.tpch.customer ON c_custkey = o_custkey
INNER JOIN dais_2023_dlt.tpch.nation ON n_nationkey = c_nationkey
```

```
%sql
-- GOLD
CREATE MATERIALIZED VIEW Daily_TotalOrders_By_Nation
PARTITIONED BY (o_orderdate)
TBLPROPERTIES (delta.enableChangeDataFeed = true)
AS
SELECT o_orderdate, n_name, sum(o_totalprice) as totalprice_by_nation, count(o_totalprice) as item_sold_by_nation
FROM Live.Enriched_Orders
GROUP BY o_orderdate, n_name
```

```
1 %sql
2 -- GOLD
3 CREATE MATERIALIZED VIEW Daily_2nd_high_OrderPrice
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
```

+ New

- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- SQL Editor
- Queries
- Dashboards
- Genie
- Alerts
- Query History
- SQL Warehouses
- Data Engineering
- Job Runs
- Pipelines**
- Machine Learning
- Playground
- Experiments
- Features
- Models
- Serving
- Marketplace
- Partner Connect

Workflows > Delta Live Tables >

# Create pipeline [Provide feedback](#)

UI JSON

## General

\* Pipeline name

Serverless

Pipeline mode

Triggered  Continuous

### Summary

- ✓ Serverless enabled
- ✓ Photon enabled

## Source code

Paths to notebooks or files that contain pipeline source code. These paths can be modified after the pipeline is created.

Paths

If you don't add any source code, Databricks will create an empty notebook for the pipeline. You can edit this notebook later.

## Destination

Storage options

Hive Metastore  Unity Catalog [Preview](#)

Catalog

Target schema



+ New

### Daily-Orders-Trend Python

File Edit View Run Help Last edit was 2 minutes ago Provide feedback

Validate Start **Orders Daily Trend** Share

- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- SQL Editor
- Queries
- Dashboards
- Genie
- Alerts
- Query History
- SQL Warehouses
- Data Engineering
- Job Runs
- Pipelines
- Machine Learning
- Playground
- Experiments
- Features
- Models
- Serving
- Marketplace
- Partner Connect

```

1 %sql
2 -- SILVER
3 CREATE MATERIALIZED VIEW Enriched_Orders
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT
8 o.* ,
9 c_name, c_mktsegment,
10 n_name
11 FROM dais_2023_dlt.tpch.orders o
12 INNER JOIN dais_2023_dlt.tpch.customer ON c_custkey = o_custkey
13 INNER JOIN dais_2023_dlt.tpch.nation ON n_nationkey = c_nationkey

```

```

1 %sql
2 -- GOLD
3 CREATE MATERIALIZED VIEW Daily_TotalOrders_By_Nation
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT o_orderdate, n_name, sum(o_totalprice) as totalprice_by_nation, count(o_totalprice) as item_sold_by_nation
8 FROM Live.Enriched_Orders

```

Debug Console **DLT graph** DLT event log

A graph will be generated here once a pipeline update has started. Click "Start" to start an update.

+ New

### Daily-Orders-Trend Python

File Edit View Run Help Last edit was 2 minutes ago Provide feedback

Validate

Start

Orders Daily Trend

Share

- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- SQL Editor
- Queries
- Dashboards
- Genie
- Alerts
- Query History
- SQL Warehouses
- Data Engineering
- Job Runs
- Pipelines
- Machine Learning
- Playground
- Experiments
- Features
- Models
- Serving
- Marketplace
- Partner Connect

```

1 %sql
2 -- SILVER
3 CREATE MATERIALIZED VIEW Enriched_Orders
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT
8 o.* ,
9 c_name, c_mktsegment,
10 n_name
11 FROM dais_2023_dlt.tpch.orders o
12 INNER JOIN dais_2023_dlt.tpch.customer ON c_custkey = o_custkey
13 INNER JOIN dais_2023_dlt.tpch.nation ON n_nationkey = c_nationkey

```

```

1 %sql
2 -- GOLD
3 CREATE MATERIALIZED VIEW Daily_TotalOrders_By_Nation
4 PARTITIONED BY (o_orderdate)
5 TBLPROPERTIES (delta.enableChangeDataFeed = true)
6 AS
7 SELECT o_orderdate, n_name, sum(o_totalprice) as totalprice_by_nation, count(o_totalprice) as item_sold_by_nation
8 FROM Live.Enriched_Orders

```

Debug Console DLT graph DLT event log

A graph will be generated here once a pipeline update has started. Click "Start" to start an update.

- New
- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- SQL Editor
- Queries
- Dashboards
- Genie
- Alerts
- Query History
- SQL Warehouses
- Data Engineering
- Job Runs
- Pipelines
- Machine Learning
- Playground
- Experiments
- Features
- Models
- Serving
- Marketplace
- Partner Connect

### Daily-Orders-Trend

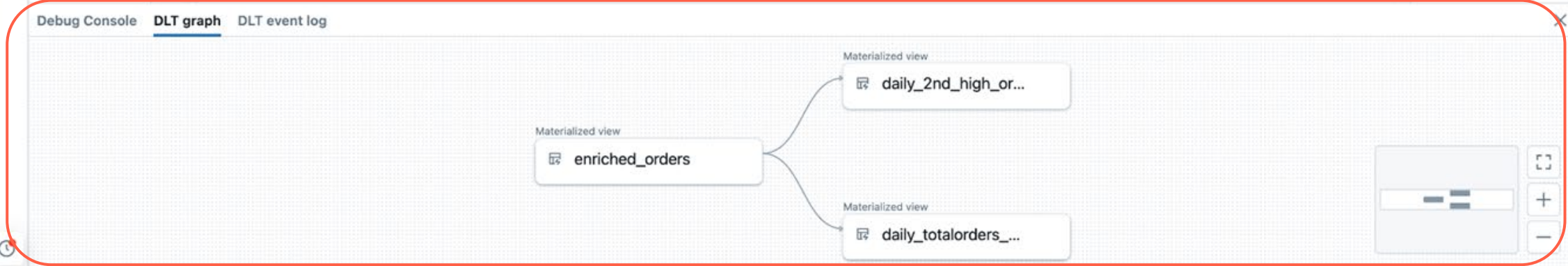
SQL ☆

File Edit View Run Help Last edit was 29 days ago Provide feedback

Validate Start Orders Daily Trend Share

```
1 -- SILVER
2 CREATE MATERIALIZED VIEW Enriched_Orders
3 PARTITIONED BY (o_orderdate)
4 TBLPROPERTIES (delta.enableChangeDataFeed = true)
5 AS
6 SELECT
7 o.* ,
8 c_name, c_mktsegment,
9 n_name
10 FROM dais_2023_dlt.tpch.orders o
11 INNER JOIN dais_2023_dlt.tpch.customer ON c_custkey = o_custkey
12 INNER JOIN dais_2023_dlt.tpch.nation ON n_nationkey = c_nationkey
```

```
1 -- GOLD
2 CREATE MATERIALIZED VIEW Daily_TotalOrders_By_Nation
3 PARTITIONED BY (o_orderdate)
4 TBLPROPERTIES (delta.enableChangeDataFeed = true)
5 AS
6 SELECT o_orderdate, n_name, sum(o_totalprice) as totalprice_by_nation, count(o_totalprice) as item_sold_by_nation
7 FROM Live.Enriched_Orders
8 GROUP BY o_orderdate, n_name
9
```





- New
- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- SQL Editor
- Queries
- Dashboards
- Alerts
- Query History
- SQL Warehouses
- Genie Spaces
- Data Engineering
- Job Runs
- Pipelines
- Machine Learning
- Playground
- Experiments
- Features
- Models
- Serving
- Marketplace
- Partner Connect

# Daily-Orders-Trend

File Edit View Run Help Last edit was 27 days ago Provide feedback

Validate Start Orders Daily Trend Share

```

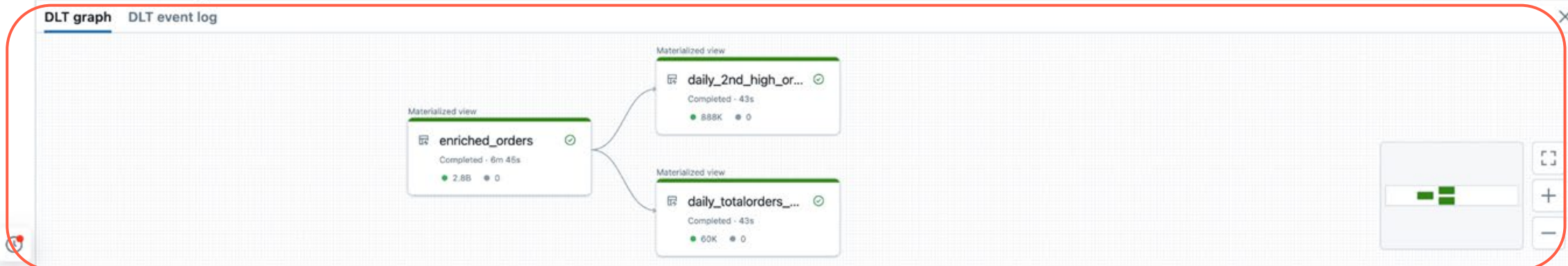
1  -- SILVER
2  CREATE MATERIALIZED VIEW Enriched_Orders
3  PARTITIONED BY (o_orderdate)
4  TBLPROPERTIES (delta.enableChangeDataFeed = true)
5  AS
6  SELECT
7  o.* ,
8  c_name, c_mktsegment,
9  n_name
10 FROM dais_2023_dlt.tpch.orders o
11 INNER JOIN dais_2023_dlt.tpch.customer ON c_custkey = o_custkey
12 INNER JOIN dais_2023_dlt.tpch.nation ON n_nationkey = c_nationkey

```

```

1  -- GOLD
2  CREATE MATERIALIZED VIEW Daily_TotalOrders_By_Nation
3  PARTITIONED BY (o_orderdate)
4  TBLPROPERTIES (delta.enableChangeDataFeed = true)
5  AS
6  SELECT o_orderdate, n_name, sum(o_totalprice) as totalprice_by_nation, count(o_totalprice) as item_sold_by_nation
7  FROM Live.Enriched_Orders
8  GROUP BY o_orderdate, n_name

```

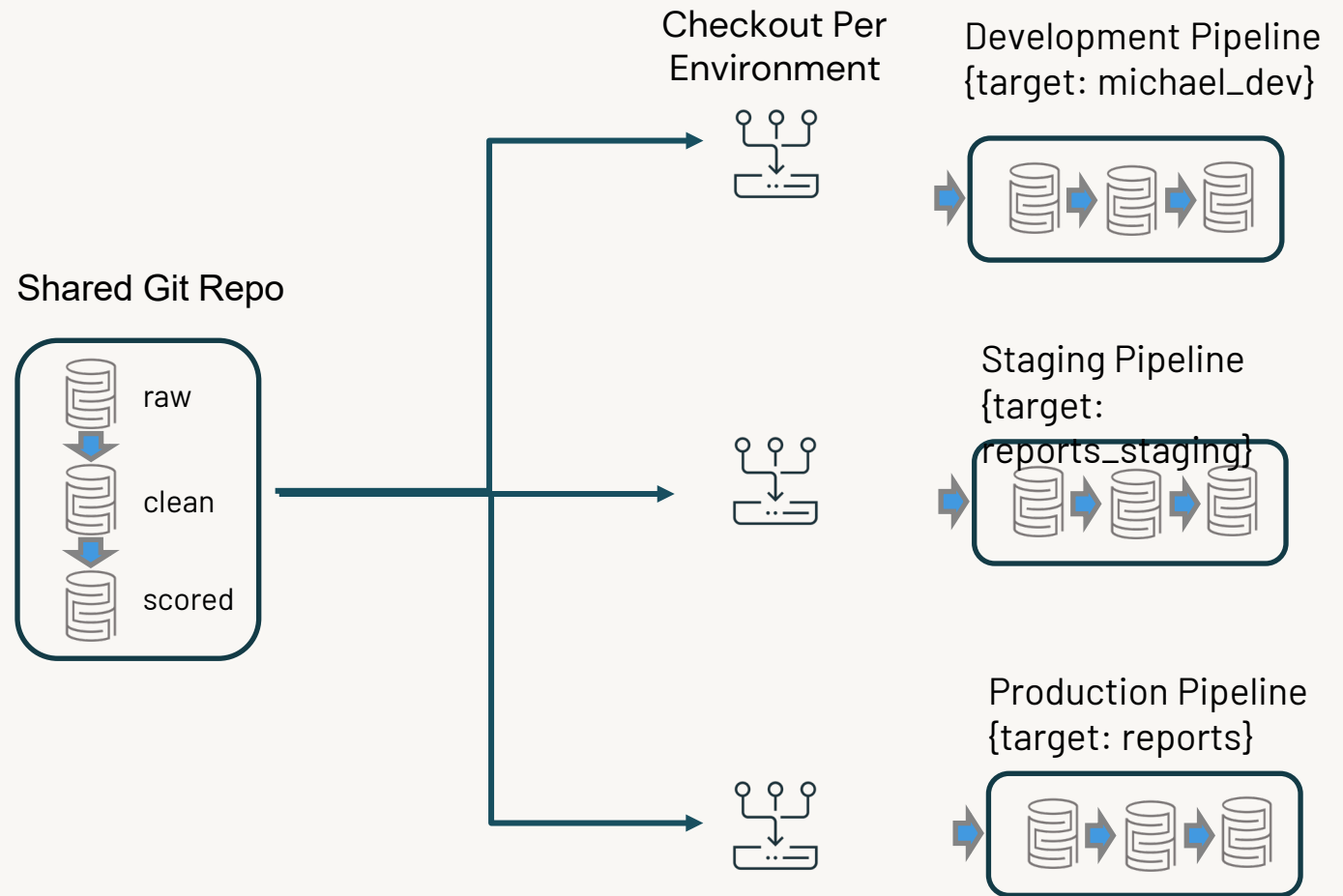


5

# Deploy your pipeline in production

# Version control and deploy your DLT code using Databricks Asset Bundles (DABs)

- Databricks Asset Bundles (DABs) allow you to version control your **source code** and your **pipeline configurations** (e.g. permissions, schedules...)
- Automate the creation of pipelines for each environment

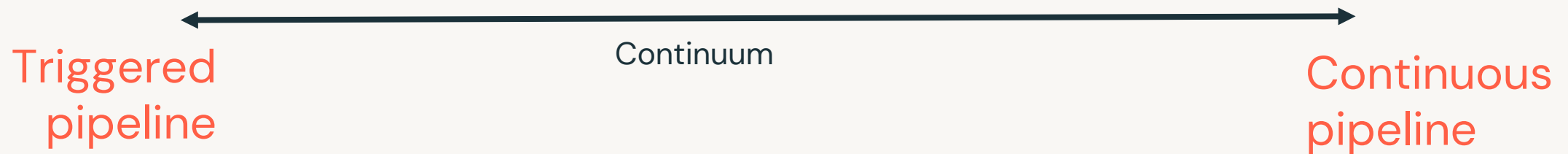


# Optimize your costs and data freshness

DLT pipelines let you switch between triggered and continuous scheduling

- Lower cost
- Data less fresh

- Higher cost
- Freshest data



**New schedule**

Job name  
Orders Daily Trend 2024-06-07 08:31:31

Schedule  
Every 1 Day

More options

**Pipeline settings** [Provide feedback](#)

**General**

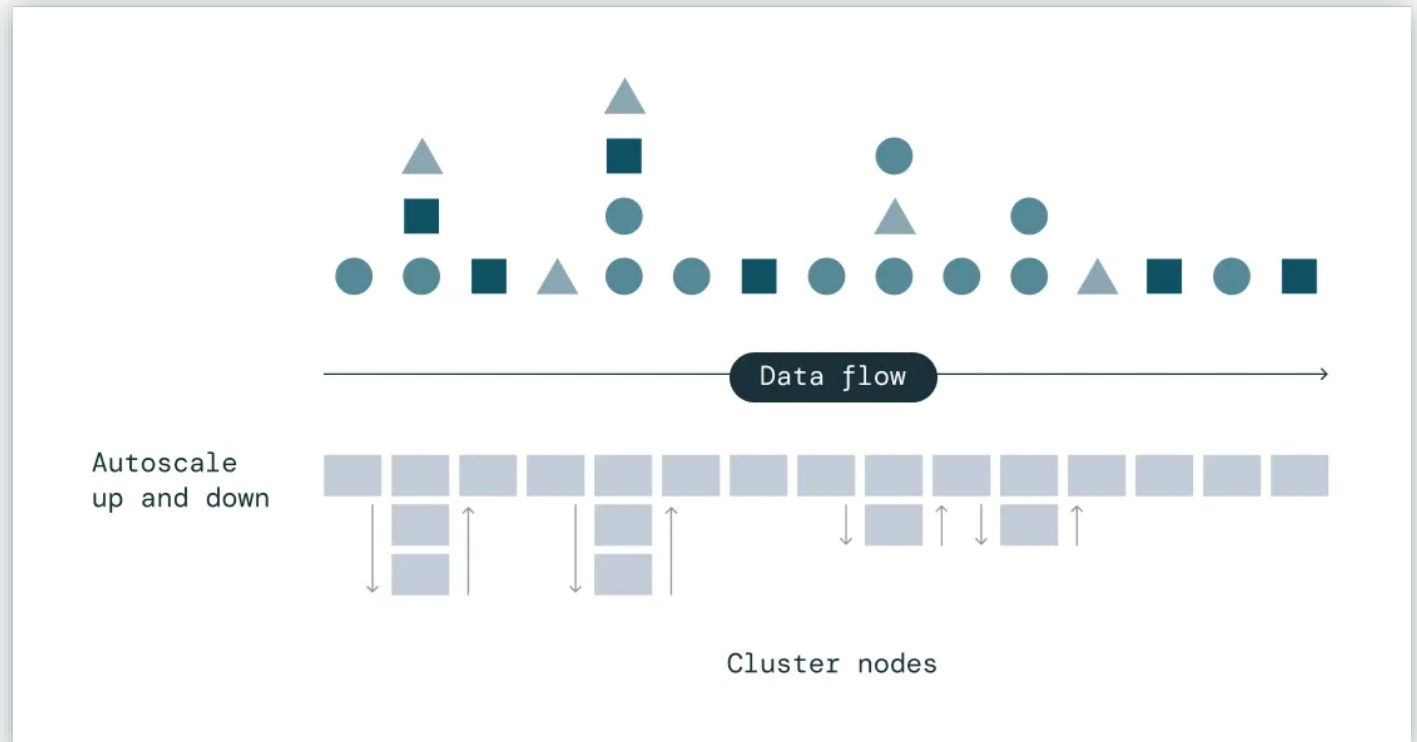
\* Pipeline name  
Orders Daily Trend

Serverless ⓘ

Pipeline mode ⓘ  
 Triggered  Continuous

# DLT pipelines **autoscale** compute resources

Features like **Enhanced Autoscaling** optimize cluster utilization by only scaling up to the necessary number of nodes, and gracefully shut down nodes when utilization is low to avoid unnecessary spend



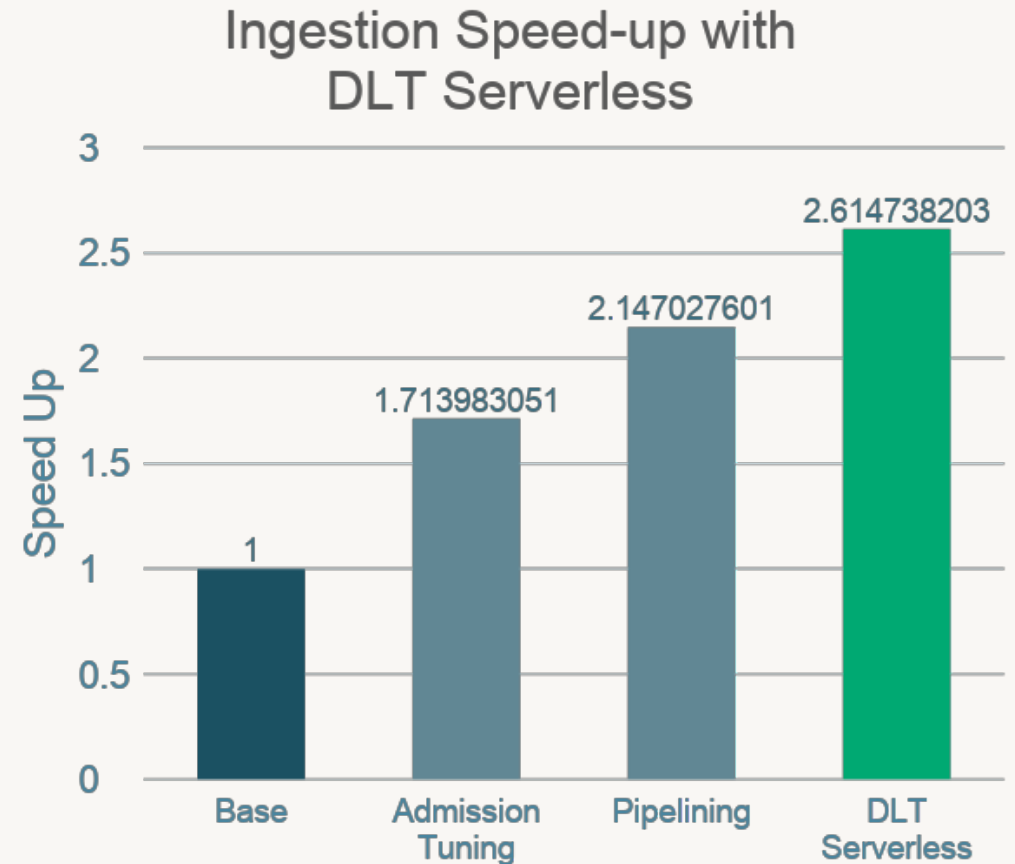


# Use **Serverless** compute for more optimizations

DLT Serverless *further optimizes TCO and latency*

DLT Serverless dynamically optimizes compute and scheduling

- **Pipelined execution** of multiple microbatches
- **Dynamically tuning** of batches sizes based on the amount of compute available



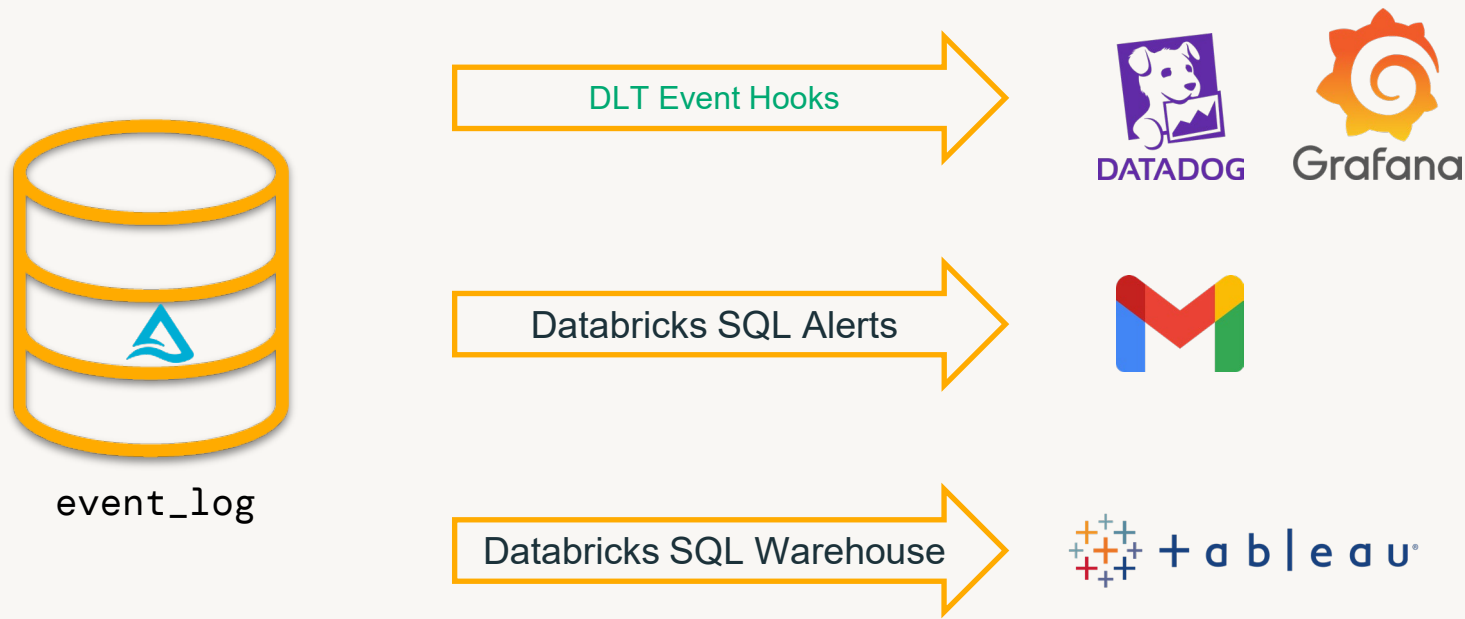


# Monitoring and alerting

# Monitor your pipelines using the DLT event log

Use the information in the event log with your **existing operational tools**.

The event log is just a delta table created for each pipeline.



**+ New** Workflows > Delta Live Tables > **Orders Daily Trend** [Provide feedback](#) Development Production Settings Schedule (10+) Start

6/3/2024, 5:25:33 PM · Completed Select tables for refresh

All Info Warning Error

23 hours ago	flow_progress	Flow 'enriched_orders' is PLANNING.
23 hours ago	planning_information	Flow 'enriched_orders' has been planned in DLT to be executed as COMPLETE_RECOMPUTE.
23 hours ago	flow_progress	Flow 'enriched_orders' is STARTING.
23 hours ago	flow_progress	Flow 'enriched_orders' is RUNNING.
23 hours ago	autoscale	Scaling up to 5 executors from current cluster size of 3
23 hours ago	autoscale	Achieved cluster size 5 for cluster 0603-235947-n7mfs30p with status SUCCEEDED.
23 hours ago	flow_progress	Flow 'enriched_orders' has COMPLETED.
23 hours ago	flow_progress	Flow 'daily_2nd_high_orderprice' is PLANNING.
23 hours ago	planning_information	Flow 'daily_2nd_high_orderprice' has been planned in DLT to be executed as COMPLETE_RECOMPUTE.
23 hours ago	flow_progress	Flow 'daily_2nd_high_orderprice' is STARTING.
23 hours ago	flow_progress	Flow 'daily_totalorders_by_nation' is PLANNING.
23 hours ago	flow_progress	Flow 'daily_2nd_high_orderprice' is RUNNING.
23 hours ago	planning_information	Flow 'daily_totalorders_by_nation' has been planned in DLT to be executed as COMPLETE_RECOMPUTE.
23 hours ago	flow_progress	Flow 'daily_totalorders_by_nation' is STARTING.
23 hours ago	flow_progress	Flow 'daily_totalorders_by_nation' is RUNNING.
23 hours ago	flow_progress	Flow 'daily_2nd_high_orderprice' has COMPLETED.
23 hours ago	flow_progress	Flow 'daily_totalorders_by_nation' has COMPLETED.
23 hours ago	memory_utilization	Collected memory utilization on the cluster during termination

- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- SQL Editor
- Queries
- Dashboards
- Alerts
- Query History
- SQL Warehouses
- Genie Spaces
- Data Engineering
- Job Runs
- Pipelines**
- Machine Learning
- Playground
- Experiments
- Features
- Models
- Serving
- Marketplace
- Partner Connect

7

Implement governance  
on all your data assets

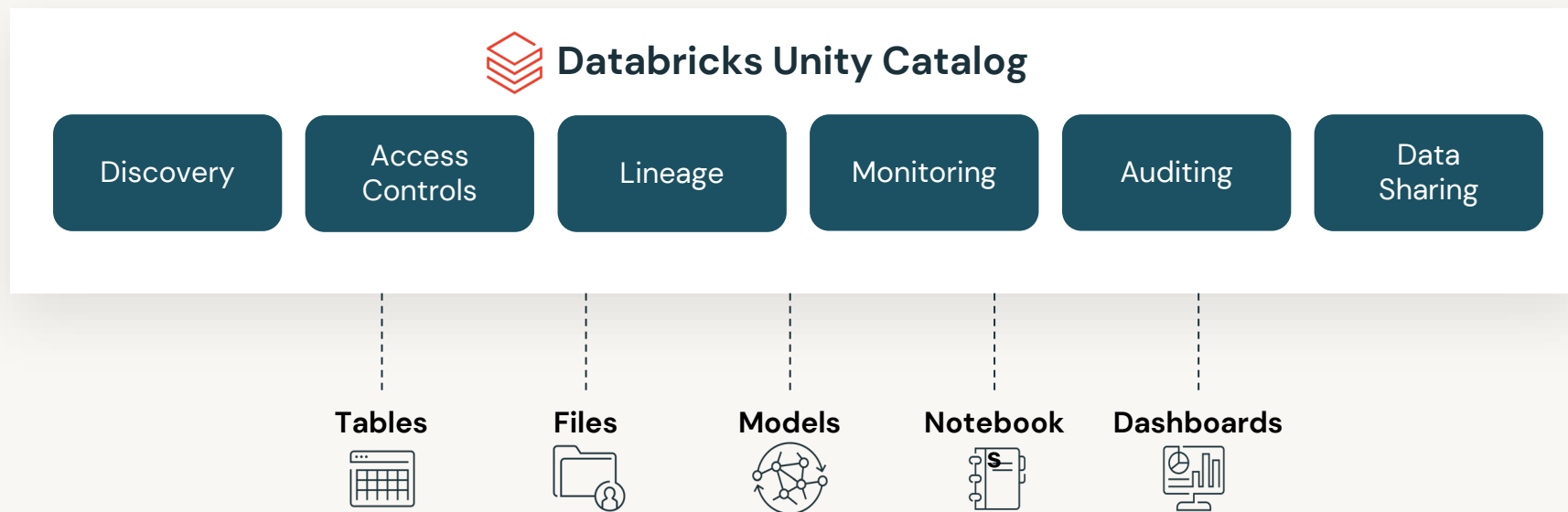
# Databricks Unity Catalog

## Unified governance for data and AI

Unified visibility into data and AI

Single permission model

Open data sharing



# DLT and Unity Catalog helps you reach your governance goals



# Centralized Access Controls

Centrally grant and manage access permissions across workloads

## Using ANSI SQL DCL

```
GRANT <privilege> ON <securable_type>  
<securable_name> TO `<principal>`
```

```
GRANT SELECT ON iot.events TO engineers
```

Choose  
permission level

Streaming Table or  
Materialized View

Sync groups from  
your identity  
provider

## Using UI

The screenshot shows the Data Explorer interface with a modal dialog titled "Grant on main.default.department". The dialog contains the following information:

- A warning: "Users also require `USE CATALOG` and `USE SCHEMA` on the parent catalog and schema to perform actions in this table. [Learn more](#)"
- A section for "Users and groups" with a search box containing "analysts".
- A "Privileges" section with three options:
  - SELECT** gives read access to an object
  - MODIFY** gives ability to add, delete, and modify data to or from an object
  - ALL PRIVILEGES** gives all privileges
- Buttons for "Cancel" and "Grant".



Coming soon to STs and MVs!

# Row Level Security and Column Level Masking

Provide differential fine grained access to datasets

## Only show specific rows

```
CREATE FUNCTION <name> ( <parameter_name >  
<parameter_type> .. )  
RETURN {filter clause whose output must be a boolean}
```

```
CREATE FUNCTION us_filter(region STRING)  
RETURN IF(IS_MEMBER('admin'), true, region="US");
```

```
ALTER TABLE sales SET ROW FILTER us_filter ON region;
```

Test for group membership

Assign reusable filter to table

Specify filter predicates

## Mask or redact sensitive columns

```
CREATE FUNCTION <name> (<parameter_name>,  
<parameter_type>, [, <column>...])  
RETURN {expression with the same type as the first parameter}
```





```
CREATE FUNCTION ssn_mask(ssn STRING)  
RETURN IF(IS_MEMBER('admin'), ssn, "****");
```

```
ALTER TABLE users ALTER COLUMN table_ssn SET MASK  
ssn_mask;
```

Test for group membership

Assign reusable mask to column

Specify mask or function to mask



# Scaling the Heineken data platform with DLT

Maarten de Haas, Product Architect at Heineken