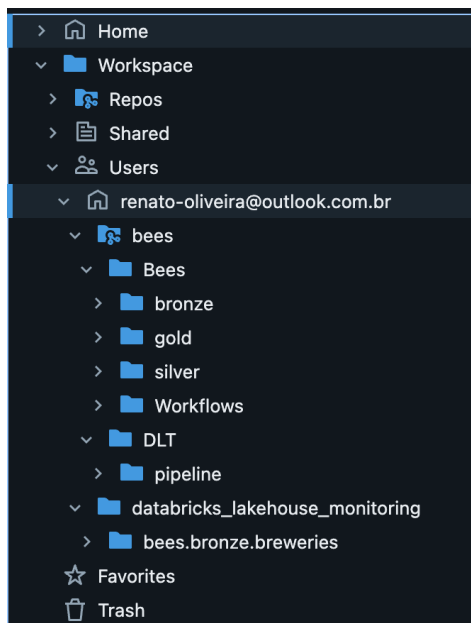


# BEES Data Engineering Case – Visual Workflow & Execution Overview

This document provides a detailed visual and descriptive overview of the workflows, executions, and monitoring strategies applied in the BEES Data Engineering case developed in Databricks.

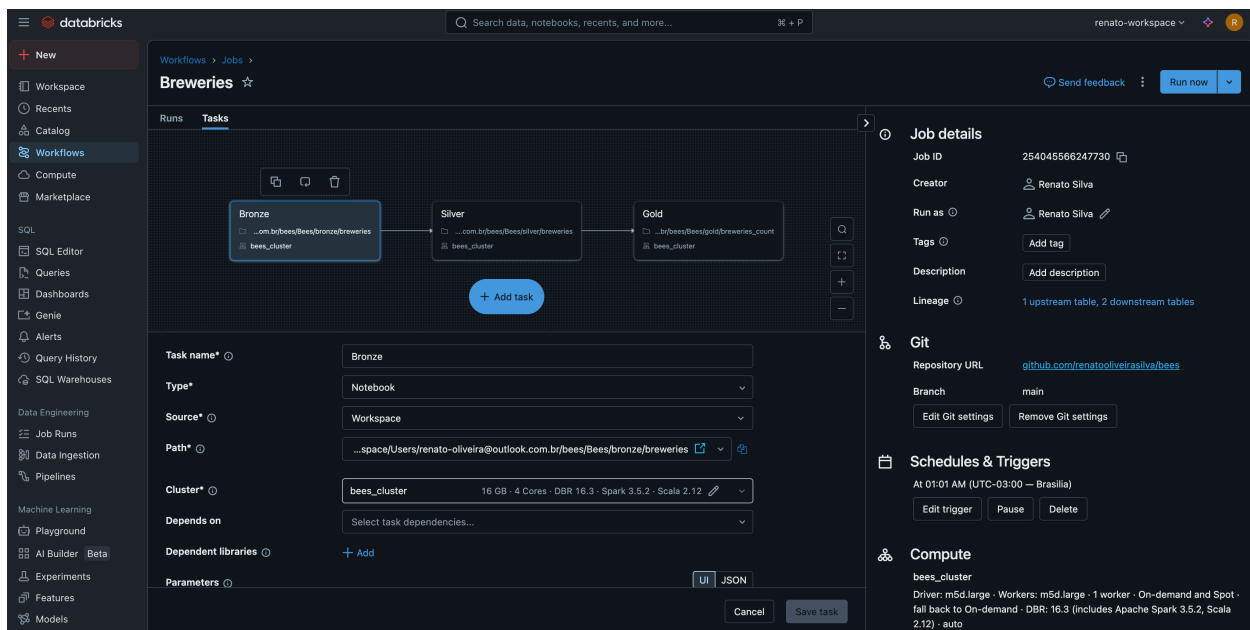
## Project Structure Visualization

Below is the visual representation of the structured folder architecture within the Databricks workspace. This organization follows data lakehouse principles, separating ingestion (raw - bronze), transformation (cleaned - silver), and final analytics (aggregated - gold) stages.



## Databricks Workflow Job

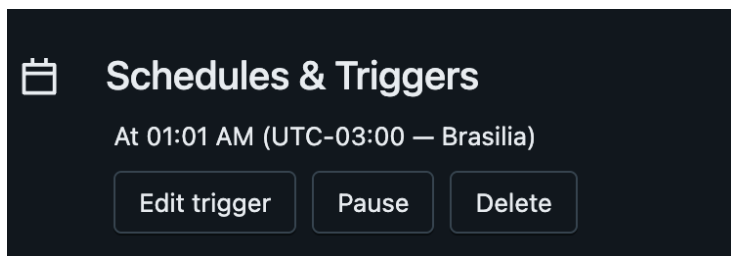
The following image shows the Databricks Workflow Job created to orchestrate the pipeline. This job coordinates notebook execution, defines task dependencies, and ensures proper execution order.



The JSON configuration used to create this job is available in the project's GitHub repository.


## Scheduled Job Simulation

A simulation of the scheduled execution was set up to mimic a production-like environment. This helps demonstrate how automated, recurring executions would work over time.



## Cluster Configuration

A job cluster configuration was used, as recommended by Databricks. This setup ensures compute resources are only active during pipeline execution, optimizing cost efficiency.

 **Compute**

bees\_cluster


Driver: m5d.large · Workers: m5d.large · 1 worker · On-demand and Spot · fall back to On-demand · DBR: 16.3 (includes Apache Spark 3.5.2, Scala 2.12) · auto


Configure

Swap

## Failure Monitoring & Alerts

To improve observability and operational awareness, email alerts were configured for failed runs. The job was also configured with a 1-hour timeout and an early warning alert set at the 20-minute mark.

 **Job notifications** ⓘ

 renato-oliveira@outlook.com.br


On failure

Edit notifications

Duration and streaming backlog thresholds ⓘ


Duration warning:

20m 0s



Duration timeout:

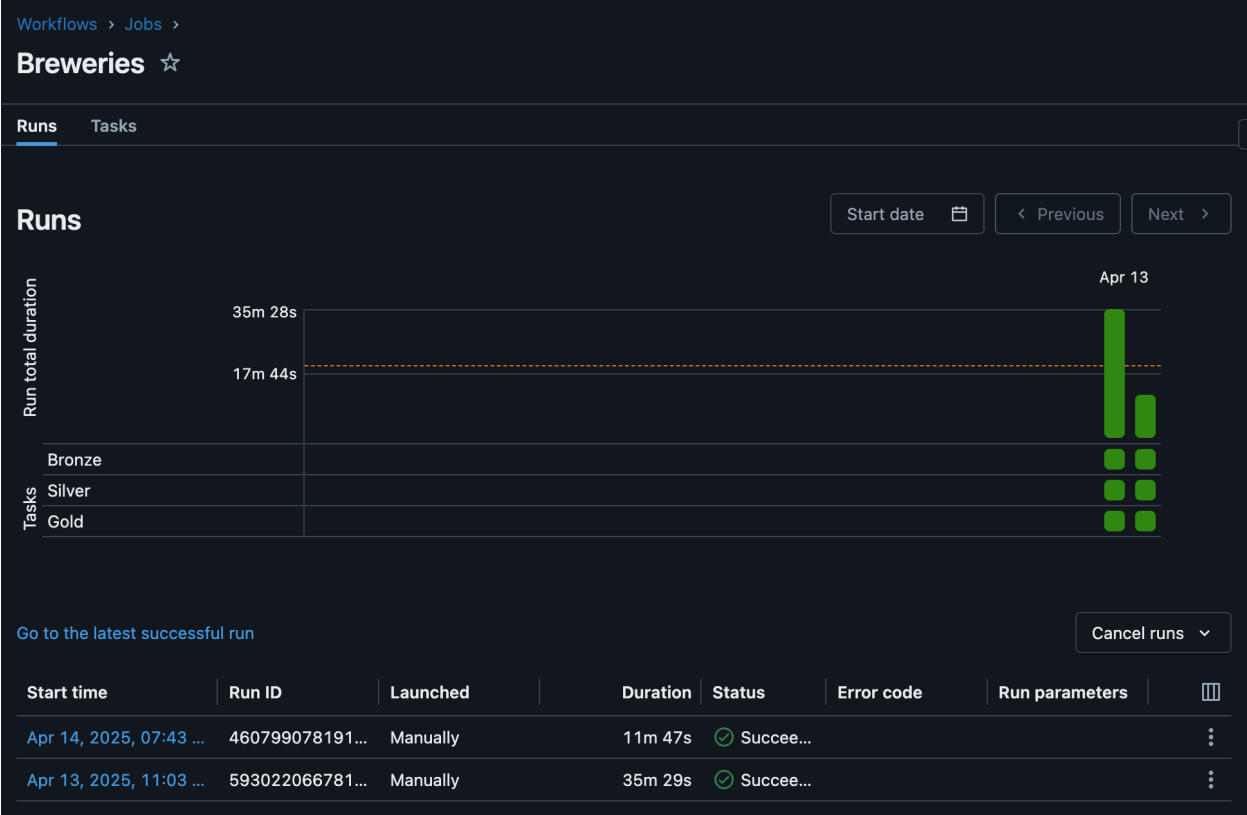
1h 0m 0s



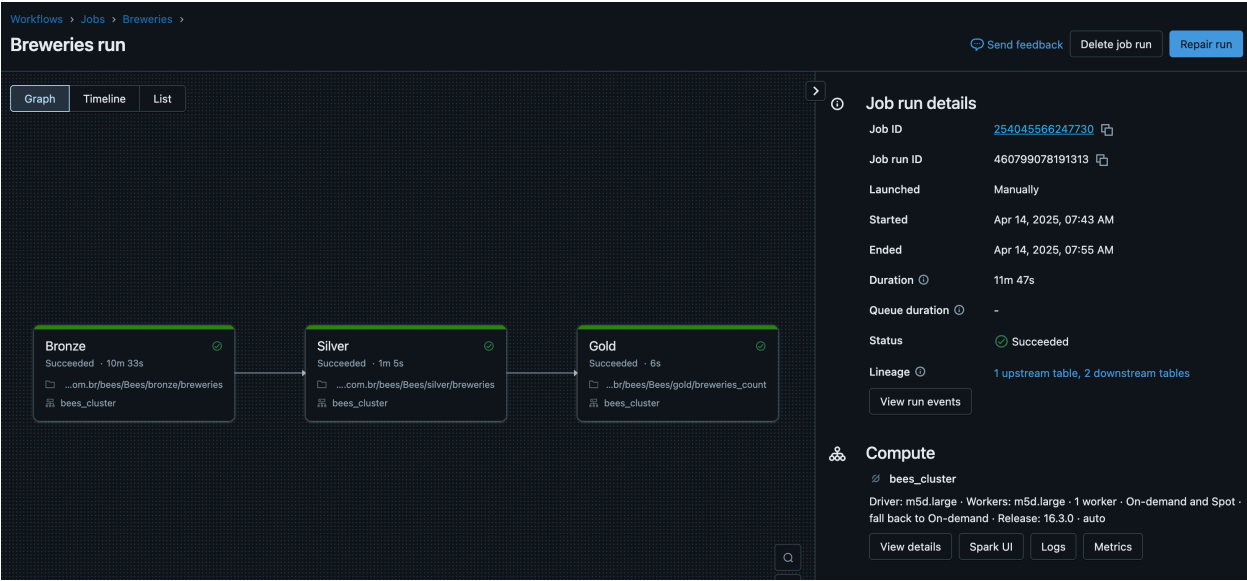
Edit metric thresholds

## Execution Example & Lineage

Below are visual examples of two pipeline runs, showcasing task status and execution details. Following that, a data lineage graph highlights how data progresses from the ingestion layer to the final curated tables.







- Workflow lineage






- Table lineage


Catalog Explorer > bees > bronze >



**breweries**  

Open in a dashboard  Create 

Overview Sample Data Details Permissions History **Lineage** Insights Quality

Filter lineage  All connections    See lineage graph

Last 3 months 

Tables	Table name	Last activity	Lineage direction
	bees.silver.breweries	Apr 14, 2025, 7:54 AM GMT-3	Downstream
	bees.bronze.breweries_profile_metrics	Apr 13, 2025, 11:48 PM GM...	Downstream

Notebooks

Workflows

Pipelines

Dashboards

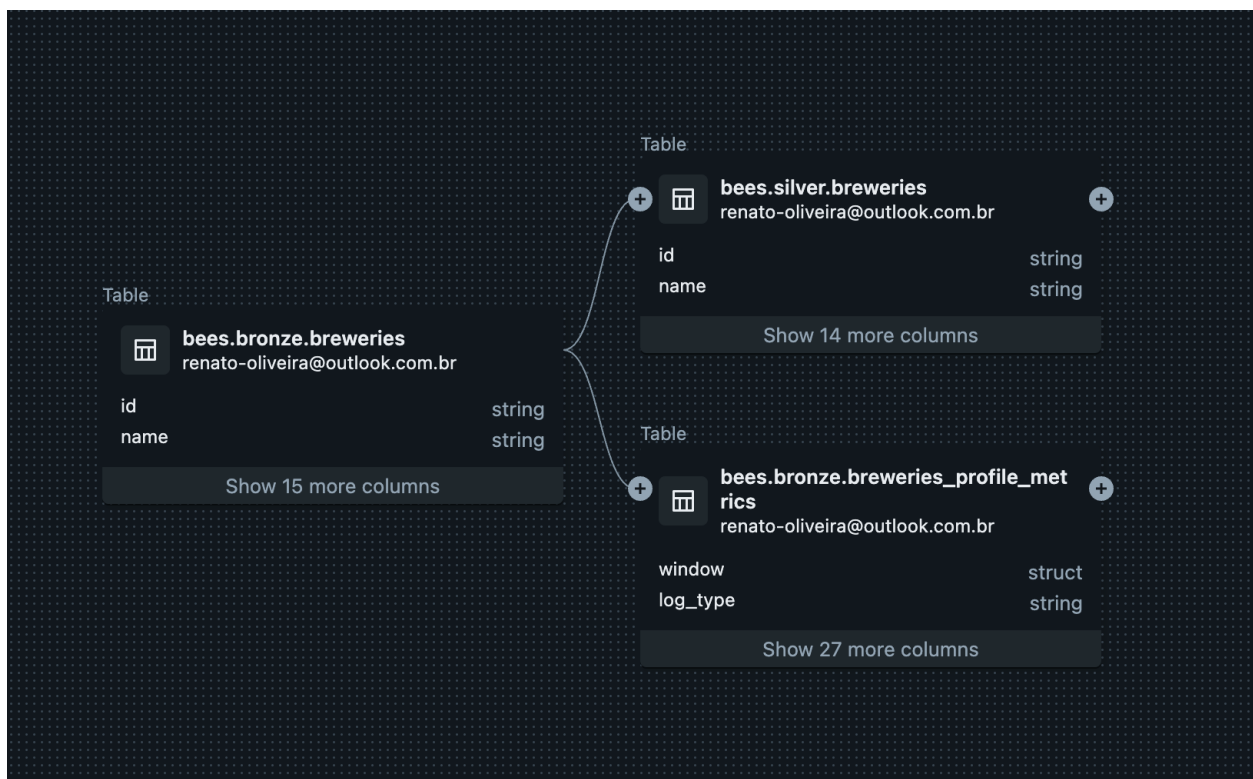
Paths

Queries

Models

Endpoints

- Table lineage graph



## ✓ Data Quality Layer

An example of the data quality monitoring layer is shown below. This dashboard provides real-time validation metrics such as null checks, unique constraints, and domain-specific validations.

Catalog Explorer > bees > bronze >

breweries

☆

Open in a dashboard

Create

Overview

Sample Data

Details

Permissions

History

Lineage

Insights

Quality

View dashboard

View metrics in the automatically generated DB SQL dashboard.

[Learn more about the monitoring dashboard.](#)

Edit monitor configuration

View refresh history

Refresh metrics

Send feedback

Metric freshness

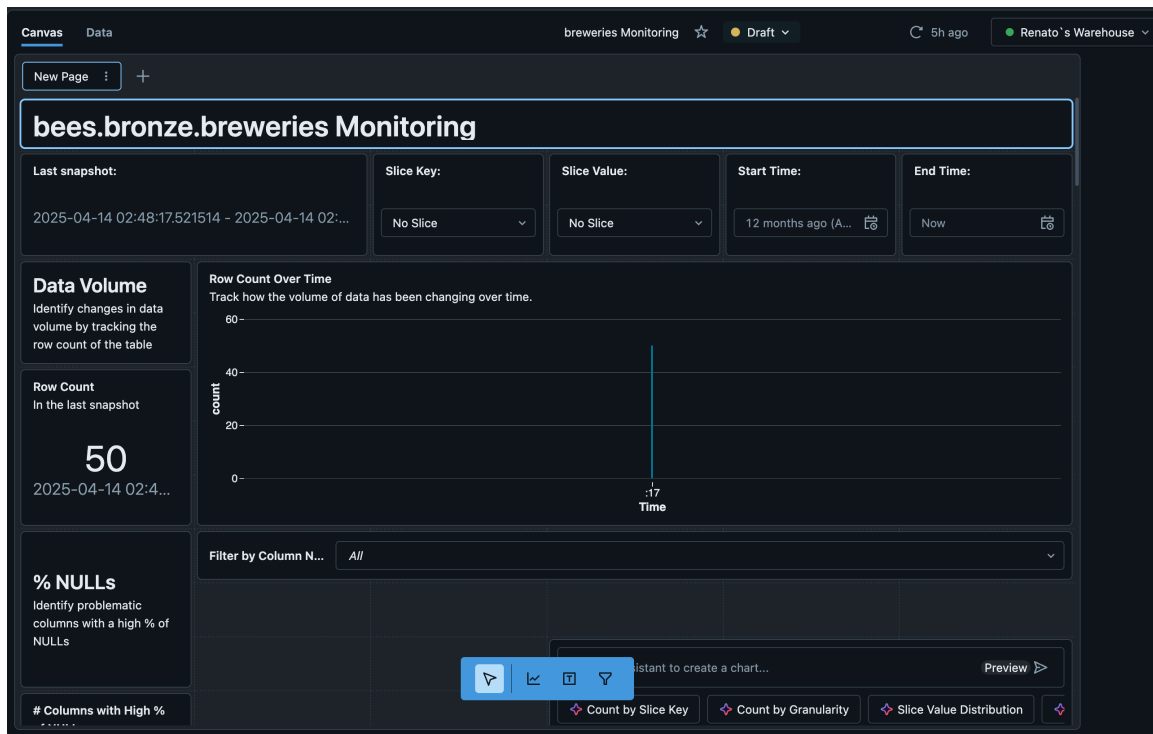
Monitor refresh schedule	No schedule set
Last metrics refresh	13 hours ago
Last table update	5 hours ago

Metric tables

Profile	<a href="#">bees.bronze.breweries_profile_metrics</a>
Drift	<a href="#">bees.bronze.breweries_drift_metrics</a>

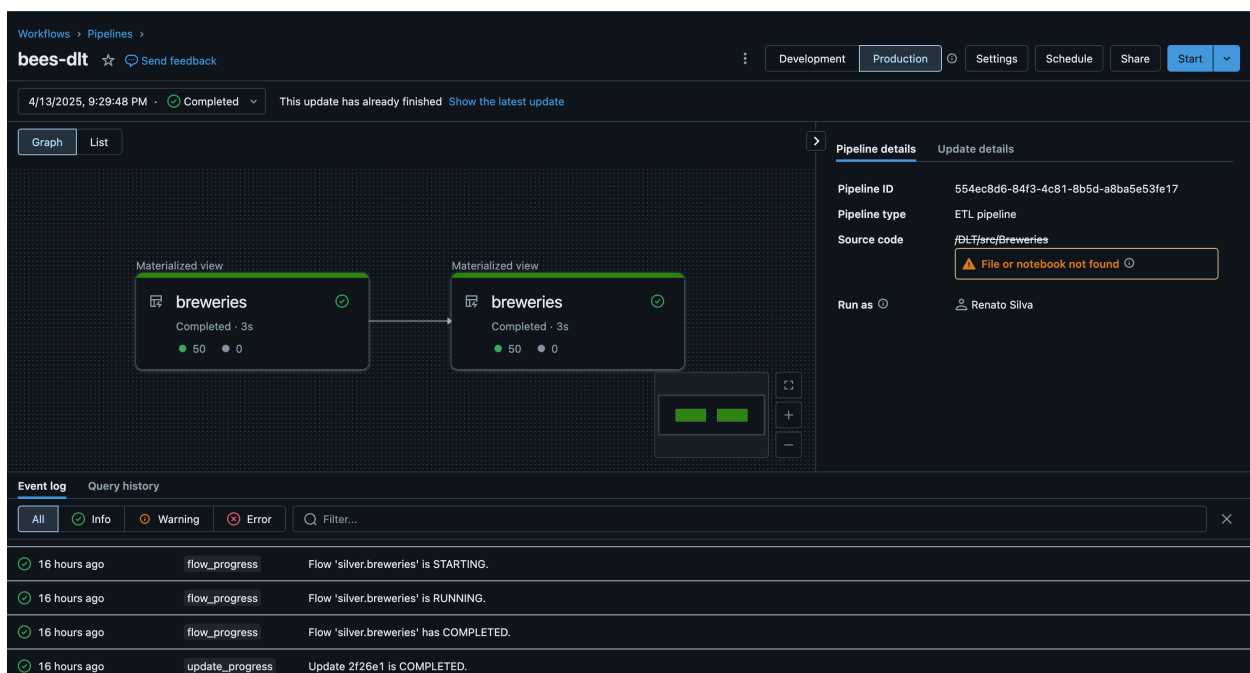
Configuration

Baseline	--
Slicing expressions	--
Custom metrics	--
Monitor version	0 <a href="#">Learn more</a>



## Delta Live Tables (DLT) Execution

Additionally, a Delta Live Tables (DLT) pipeline was configured and successfully executed. This highlights the project's capability to support declarative pipelines using Databricks' managed features.



## Final Notes

This visual documentation serves to complement the GitHub README by providing concrete examples and operational insights from within the Databricks environment.