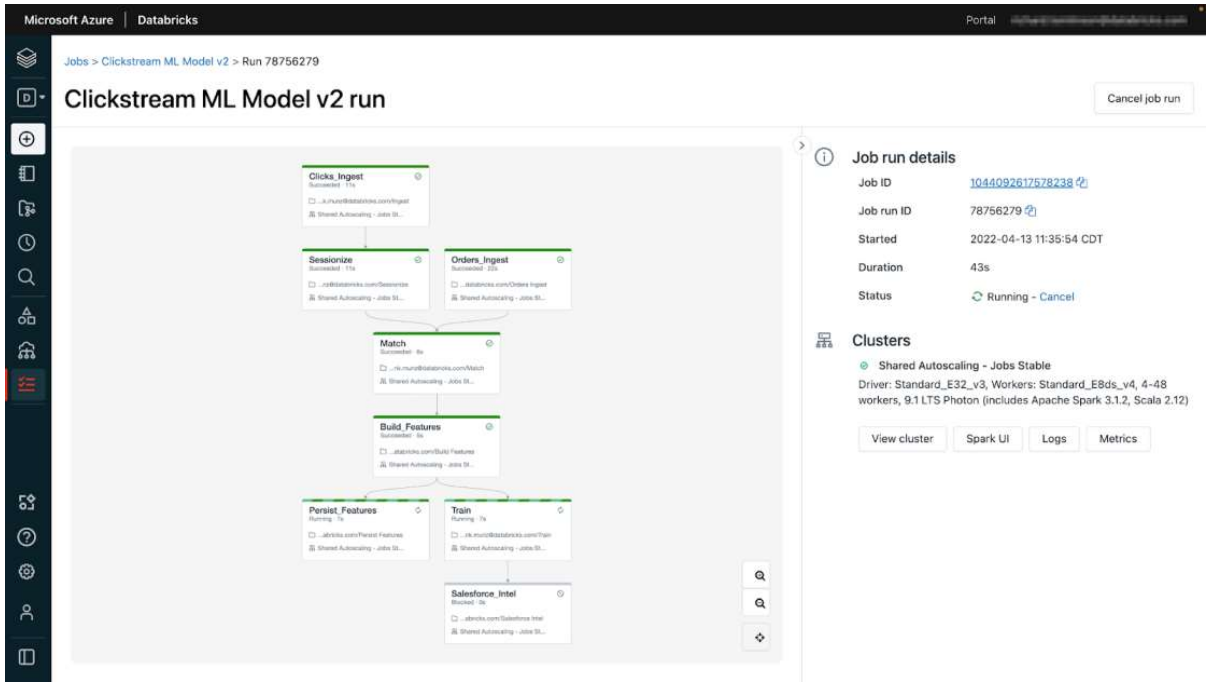
databricks 04 - Orchestrating with Workflows

(https://databricks.com)

Orchestrating our Churn pipeline with Databricks Workflows

With Databricks Lakehouse, no need for external orchestrator. We can use Workflows (/#job/list) (available on the left menu) to orchestrate our Churn pipeline within a few click.

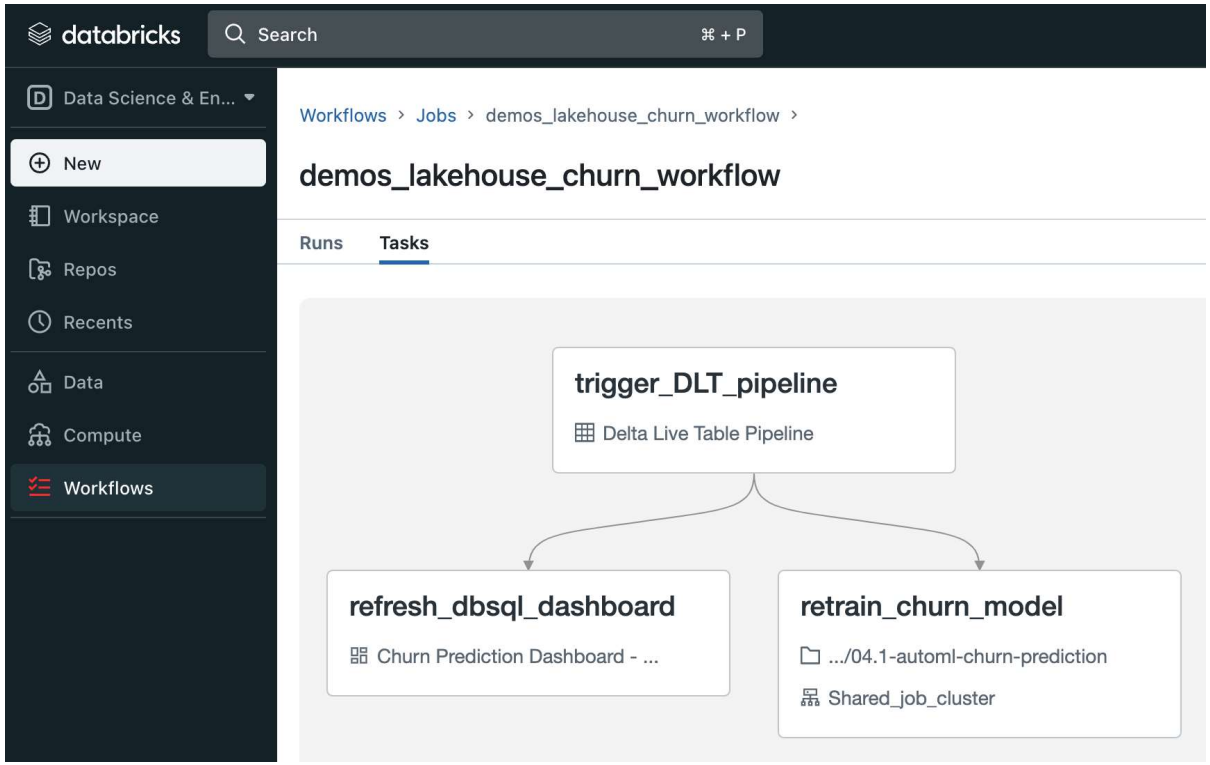


Orchestrate anything anywhere

With workflow, you can run diverse workloads for the full data and AI lifecycle on any cloud. Orchestrate Delta Live Tables and Jobs for SQL, Spark, notebooks, dbt, ML models and more.

Simple - Fully managed

Creating your workflow



A Databricks Workflow is composed of Tasks.

Each task can trigger a specific job:

- Delta Live Tables
- SQL query / dashboard
- Model retraining / inference
- Notebooks
- dbt
- ...

In this example, can see our 3 tasks:

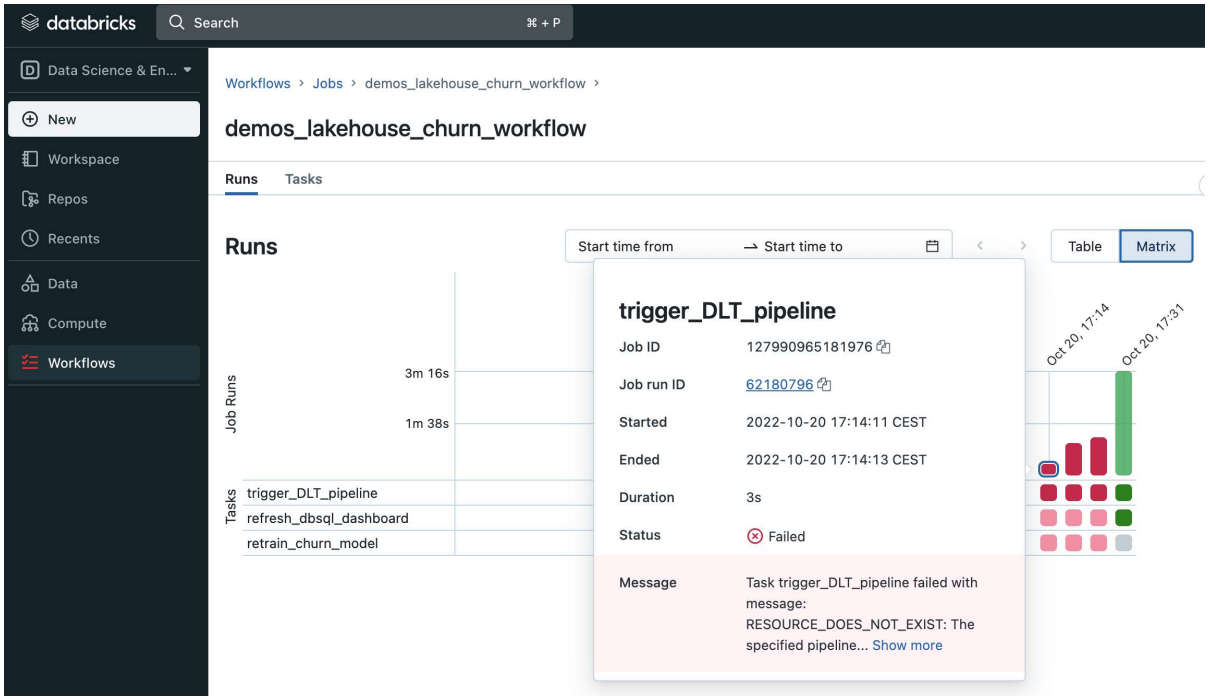
- Start the DLT pipeline to ingest new data and refresh our tables

Monitoring your runs

Once your workflow is created, we can access historical runs and receive alerts if something goes wrong!

In the screenshot we can see that our workflow had multiple errors, with different runtime, and ultimately got fixed.

Workflow monitoring includes errors, abnormal job duration and more advanced control!



Lab exercise - Create a Workflow

From the Workflows page **Create a New Job** with the following tasks

- **1. Ingest_and_process_new_data** Use the notebook 01 - Data Engineering with Delta ($./01%20-%20Data%20Engineering%20with%20Delta) as the task source
- **2. Create_Predictions** Use the notebook 02.1 - Machine Learning - Inference ($./02.1%20-%20Machine%20Learning%20-%20Inference) as the task source
  Important: for this task create a new job cluster that runs on an ML-enabled runtime!

- **3. Refresh_Dashboard** Specify **SQL** for the task type and **Dashboard** as the SQL task. Select the dashboard you created in the previous step as well as an existing SQL Warehouse

Congratulations!

You have reached the end of this lab and learned how to **create business value** in record time thanks to the **Databricks Lakehouse.**