



Deploy Workloads with Databricks Workflows

Module 05



Module Agenda

Deploy Workloads with Databricks Workflows

[Introduction to Workflows](#)

[Building and Monitoring Workflow Jobs](#)

DE 5.1 – Scheduling Tasks with the Jobs UI

DE 5.2L – Jobs Lab

Introduction to Workflows

Course Objectives

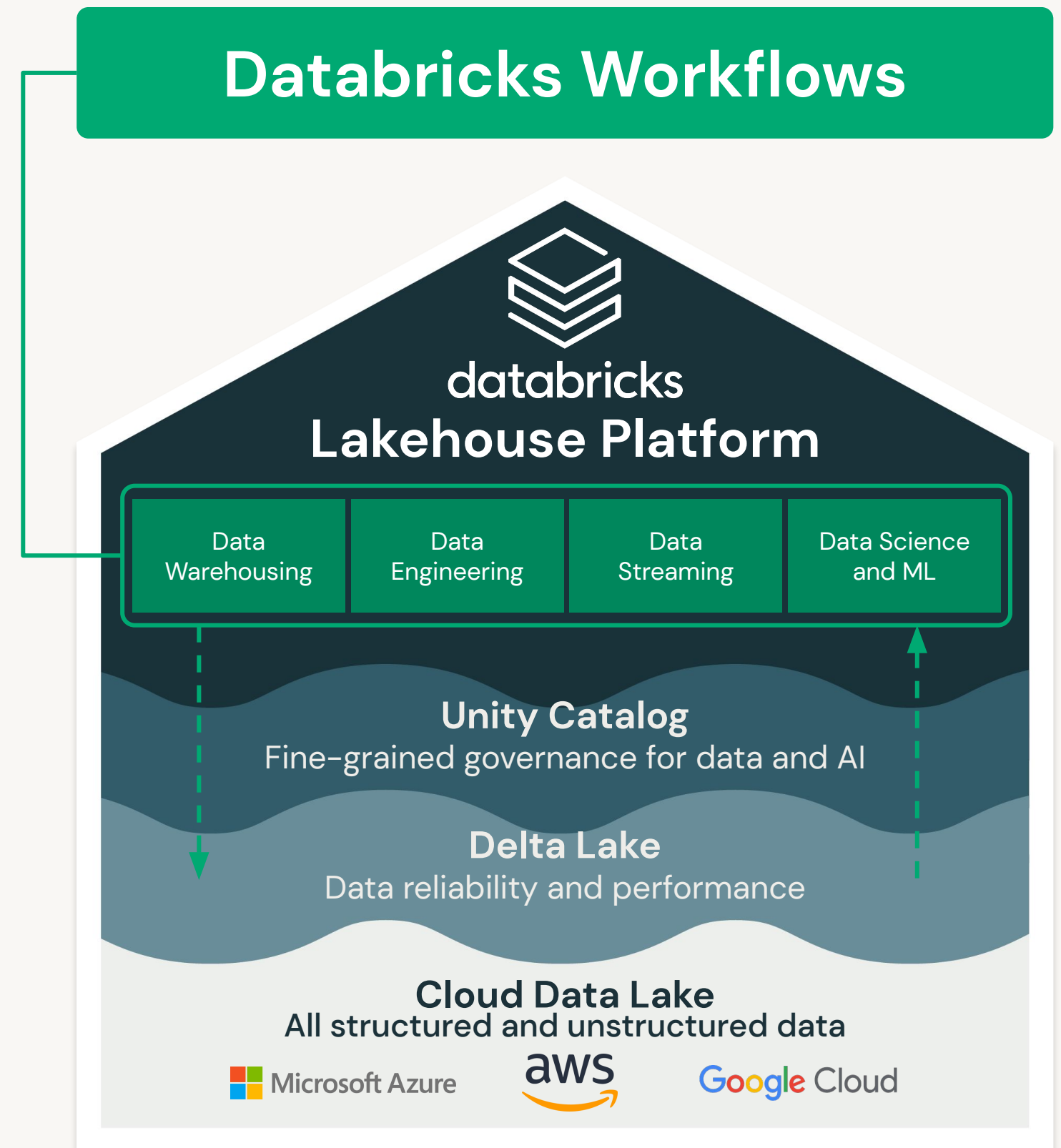
- 1 Describe the main features and use cases of Databricks Workflows
- 2 Create a task orchestration workflow composed of various task types
- 3 Utilize monitoring and debugging features of Databricks Workflows
- 4 Describe workflow best practices



Databricks Workflows

Workflows is a **fully-managed cloud-based general-purpose task orchestration service** for the entire Lakehouse.

Workflows is a service for data engineers, data scientists and analysts to build reliable data, analytics and AI workflows on any cloud.

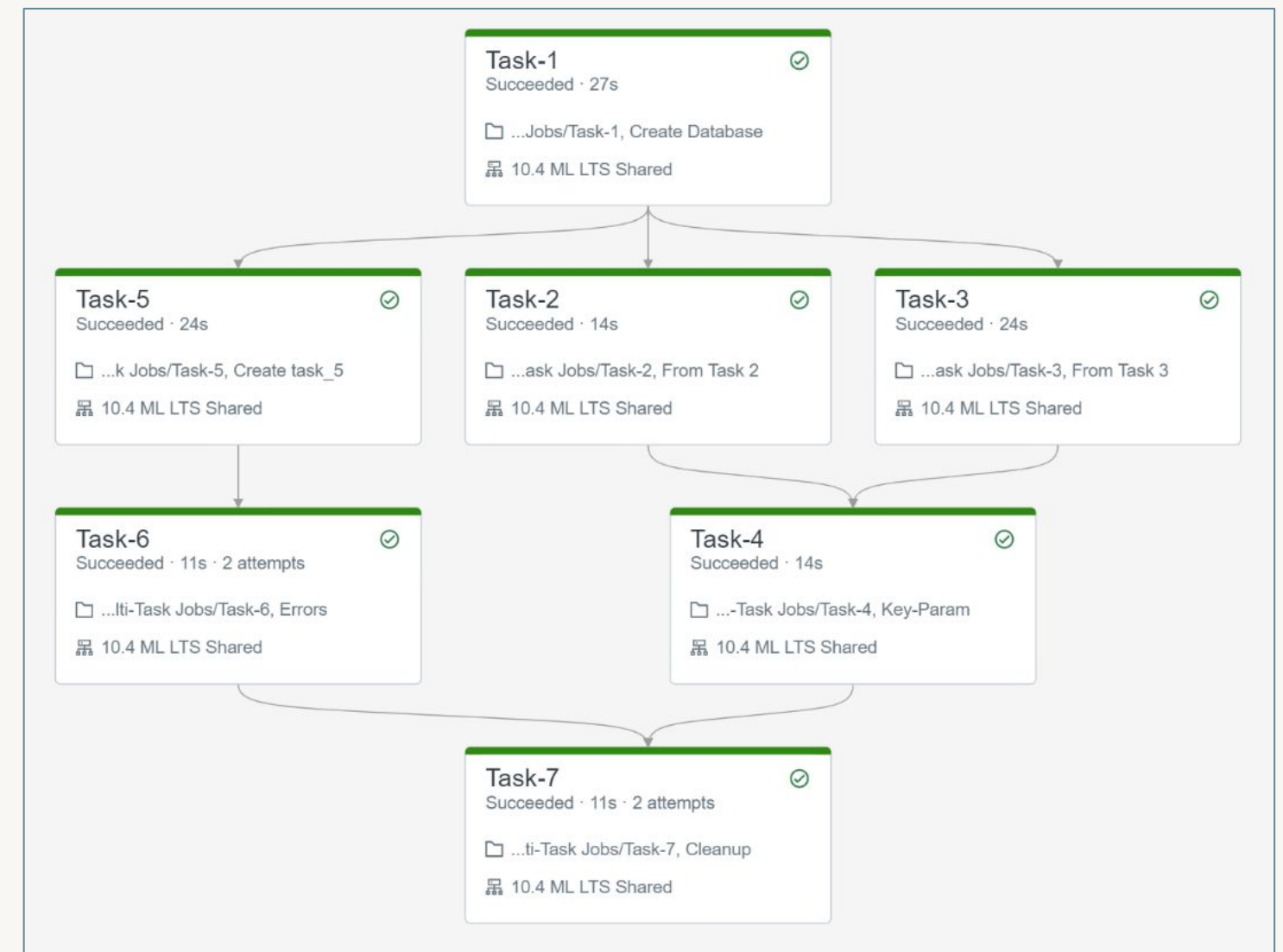


Databricks Workflows

Databricks has two main task orchestration services:

- **Workflow Jobs (Workflows)**
 - Workflows for every job
- **Delta Live Tables (DLT)**
 - Automated data pipelines for Delta Lake

Note: DLT pipeline can be a task in a workflow



DLT versus Workflow Jobs

Considerations

	Delta Live Tables	Workflow Jobs
Source	Notebooks only	JARs, notebooks, DLT, application written in Scala, Java, Python
Dependencies	Automatically determined	Manually set
Cluster	Self-provisioned	Self-provisioned or existing
Timeouts and Retries	Not supported	Supported
Import Libraries	Not supported	Supported

DLT versus Jobs

Use Cases

Orchestration of Dependent Jobs

Jobs running on schedule, containing dependent tasks/steps

Jobs Workflows

Machine Learning Tasks

Run MLflow notebook task in a job

Jobs Workflows

Arbitrary Code, External API Calls, Custom Tasks

Run tasks in a job which can contain Jar file, Spark Submit, Python Script, SQL task, dbt

Jobs Workflows

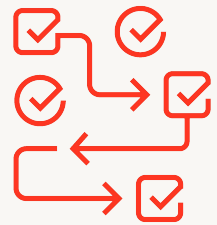
Data Ingestion and Transformation

ETL jobs, Support for batch and streaming, Built in data quality constraints, monitoring & logging

Delta Live Tables

Workflows Features

Part 1 of 2



Orchestrate Anything Anywhere

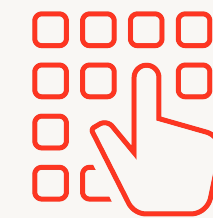
Run diverse workloads for the full data and AI lifecycle, on any cloud. Orchestrate;

- Notebooks
- Delta Live Tables
- Jobs for SQL
- ML models, and more



Fully Managed

Remove operational overhead with a fully managed orchestration service enabling you to focus on your workflows not on managing your infrastructure

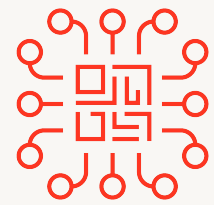


Simple Workflow Authoring

An easy point-and-click authoring experience for all your data teams not just those with specialized skills

Workflows Features

Part 2 of 2



Deep Platform Integration

Designed and built into your lakehouse platform giving you deep monitoring capabilities and centralized observability across all your workflows



Proven Reliability

Have full confidence in your workflows leveraging our proven experience running tens of millions of production workloads daily across AWS, Azure, and GCP

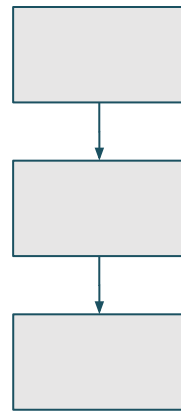
How to Leverage Workflows

- Allows you to build simple ETL/ML task orchestration
- Reduces infrastructure overhead
- Easily integrate with external tools
- Enables non-engineers to build their own workflows using simple UI
- Cloud-provider independent
- Enables re-using clusters to reduce cost and startup time



Common Workflow Patterns

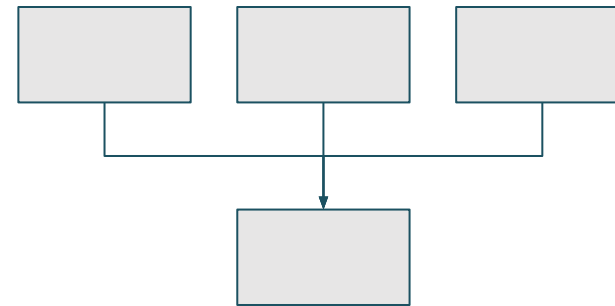
Sequence



Sequence

- Data transformation/processing/cleaning
- Bronze/silver/gold tables

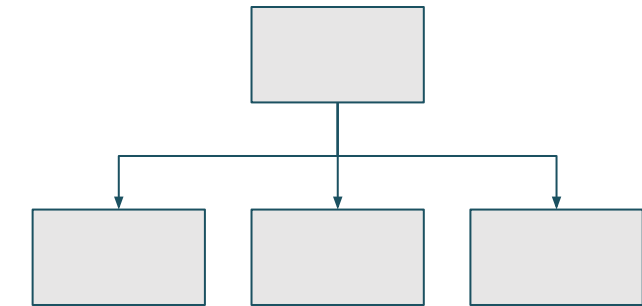
Funnel



Funnel

- Multiple data sources
- Data collection

Fan-out

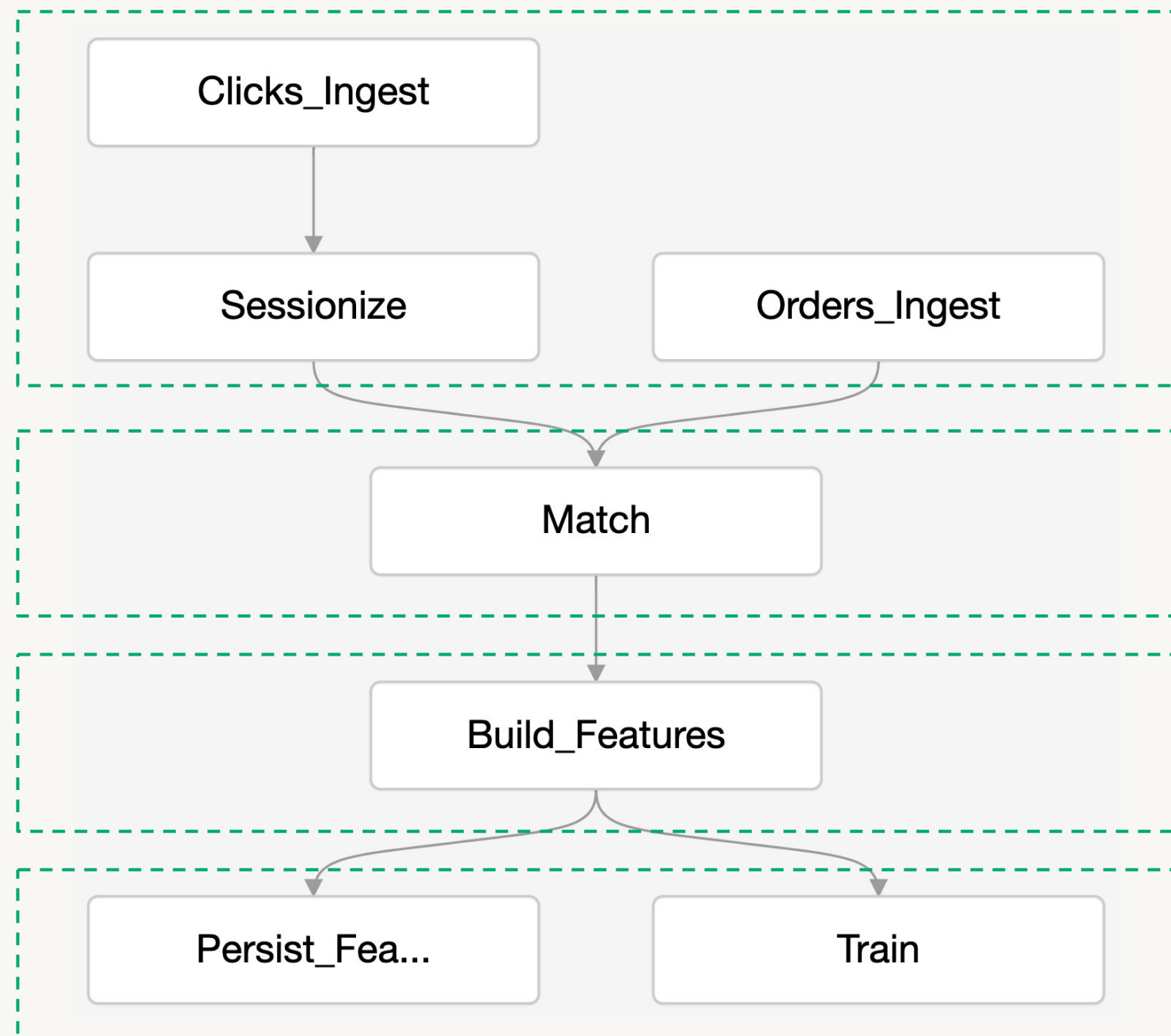


Fan-out, star pattern

- Single data source
- Data ingestion and distribution



Example Workflow



Data ingestion funnel

E.g. Auto Loader, DLT

Data filtering, quality assurance, transformation

E.g. DLT, SQL, Python

ML feature extraction

E.g. MLflow

Persisting features and training prediction model

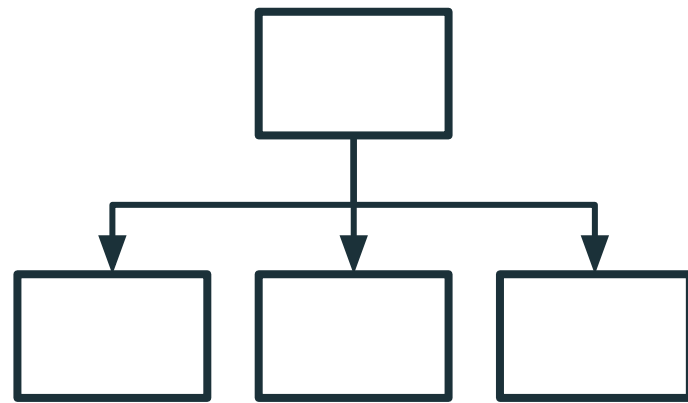




Building and Monitoring Workflow Jobs

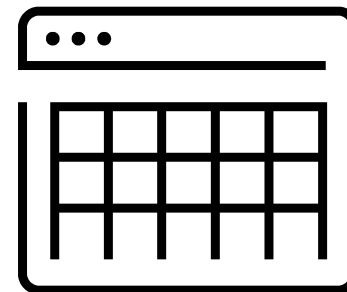
Workflows Job Components

TASKS



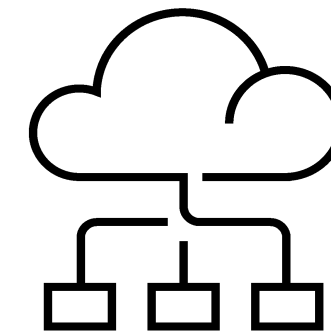
What?

SCHEDULE



When?

CLUSTER



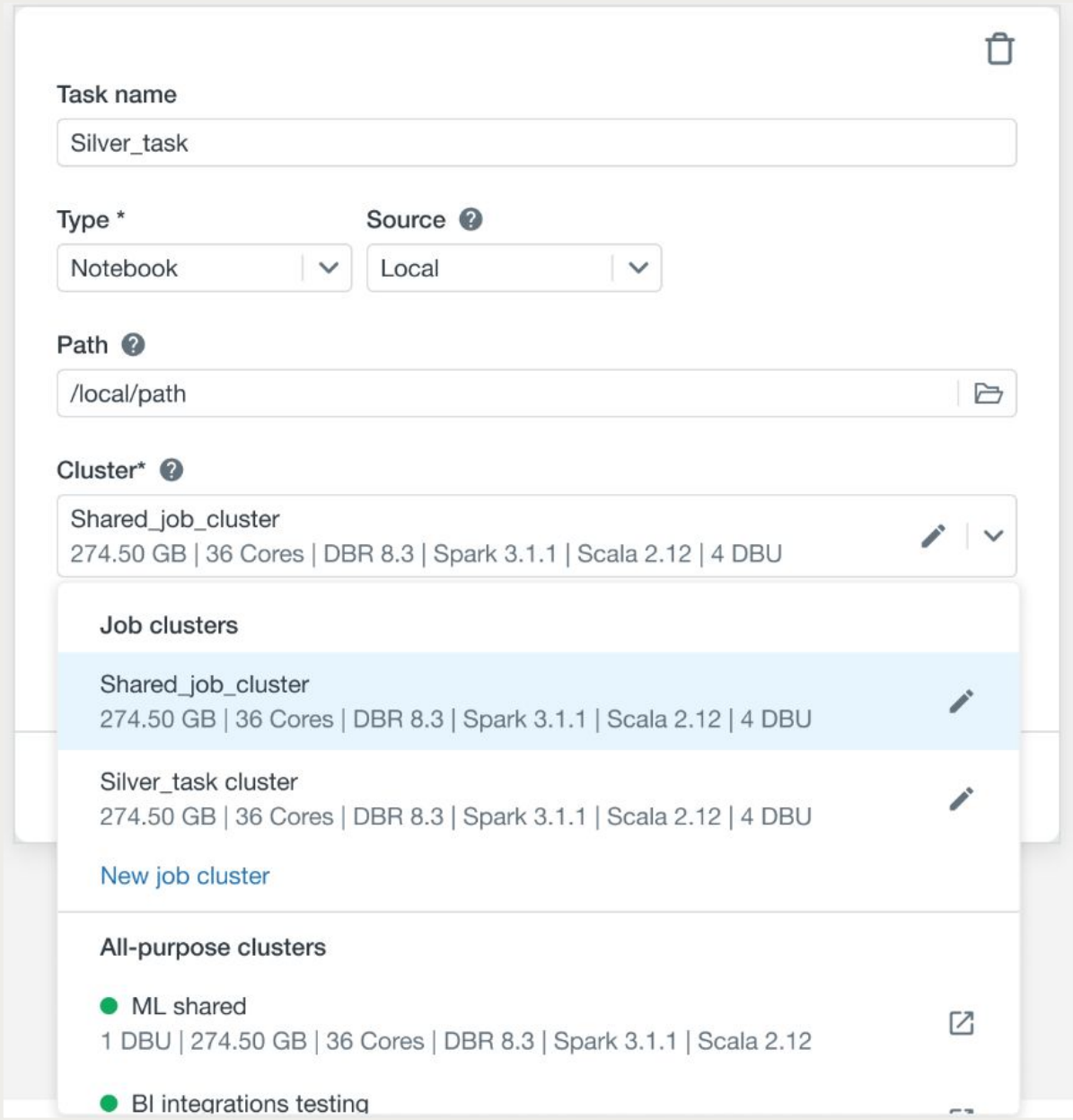
How?

Creating a Workflow

Task Definition

While creating a task;

- Define the task type
- Choose the cluster type
 - Job clusters and All-purpose clusters can be used.
 - A cluster can be used by multiple tasks. This reduces cost and startup time.
- If you want to create a new cluster, you must have required permissions.
- Define task dependency if task depends on another task



The screenshot displays the 'Task Definition' configuration interface in Databricks. The task is named 'Silver_task'. The 'Type' is set to 'Notebook' and the 'Source' is 'Local'. The 'Path' is '/local/path'. The 'Cluster' is 'Shared_job_cluster', which has 274.50 GB of memory, 36 Cores, DBR 8.3, Spark 3.1.1, Scala 2.12, and 4 DBU. A dropdown menu is open, showing a list of available clusters under 'Job clusters' and 'All-purpose clusters'. The 'Shared_job_cluster' is selected in the dropdown. The 'All-purpose clusters' section includes 'ML shared' (1 DBU, 274.50 GB, 36 Cores, DBR 8.3, Spark 3.1.1, Scala 2.12) and 'BI integrations testing'.

Cluster Name	Memory	Cores	DBR	Spark	Scala	DBU
Shared_job_cluster	274.50 GB	36 Cores	DBR 8.3	Spark 3.1.1	Scala 2.12	4 DBU
Silver_task cluster	274.50 GB	36 Cores	DBR 8.3	Spark 3.1.1	Scala 2.12	4 DBU
ML shared	274.50 GB	36 Cores	DBR 8.3	Spark 3.1.1	Scala 2.12	1 DBU
BI integrations testing						



Monitoring and Debugging

Scheduling and Alerts

You can run your jobs **immediately** or **periodically** through an easy-to-use scheduling system.

You can specify alerts to be notified when runs of a job **begin, complete or fail**. Notifications can be sent via email, Slack or AWS SNS.

The screenshot shows a configuration interface for a job. It includes a 'Schedule Type' section with radio buttons for 'Manual (Paused)' and 'Scheduled' (selected). Below is a 'Schedule' section with a frequency of 'Every Day' at '15:50' in '(UTC+02:00)' time zone, and a 'Show cron syntax' checkbox. The 'Alerts' section lists two email addresses: 'admin@anycompany.com' (with checkboxes for Start, Success, and Failure all checked) and 'notification@databricks.com' (with checkboxes for Start and Success unchecked, and Failure checked). A 'Do not send alerts for skipped runs' checkbox is also present.






Monitoring and Debugging

Access Control

Workflows integrates with existing resources access controls, enabling you to easily manage access across different teams.

Permission Settings for: **Task-1** ✕

NAME	PERMISSION
 [Redacted]	Is Owner ▾ ✕
 admins	Can Manage inherited
 users	Can View ▾ ✕

Select User, Group or Service Principal... ▾ Is Owner ▾ + Add

Cancel Save

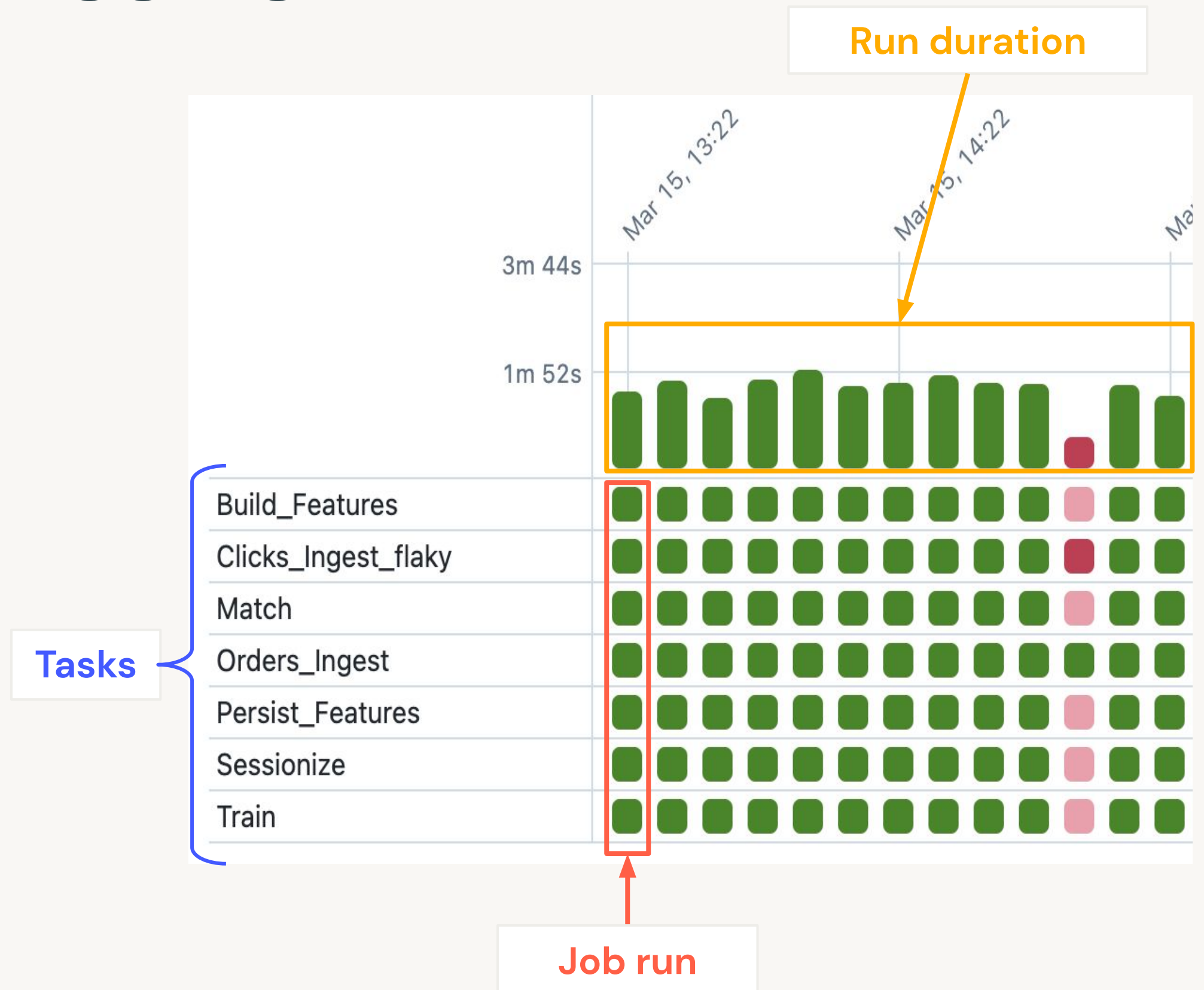
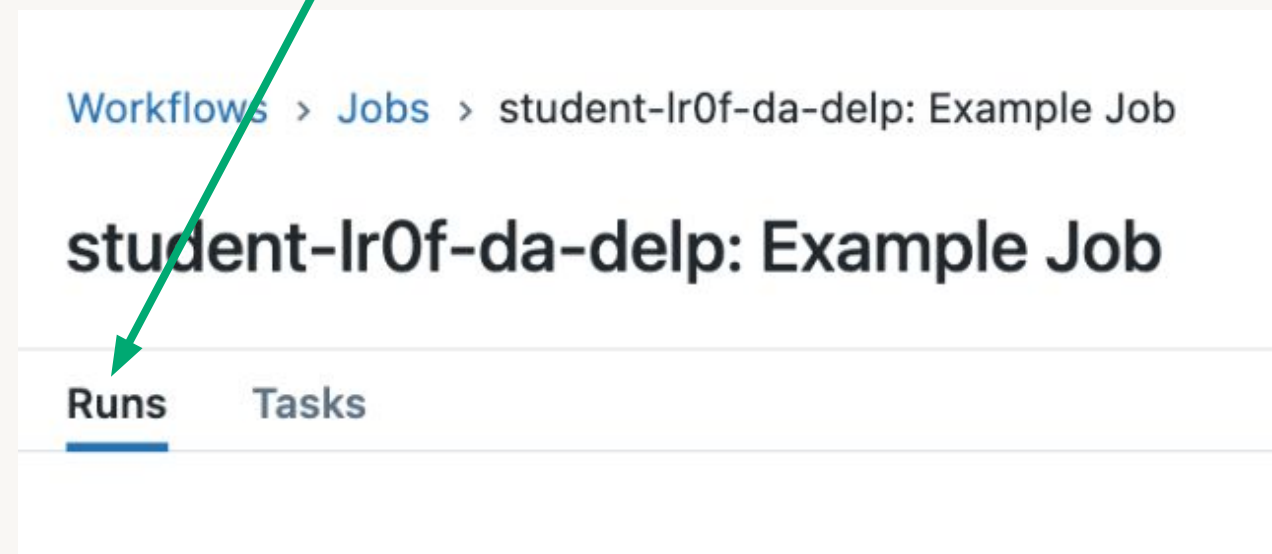


Monitoring and Debugging

Job Run History

Workflows keeps track of job runs and save information about the success or failure of each task in the job run.

Navigate to the **Runs** tab to view completed or active runs for a job.



Monitoring and Debugging

Repair a Failed Job Run

Repair feature allows you to re-run only the failed task and sub-tasks, which reduces the time and resources required to recover from unsuccessful job runs.

The screenshot displays a Databricks job run monitoring interface. On the left, a job run is shown with a 'Filter' button at the top. The job consists of three tasks: 'Bronze' (Succeeded · 5m), 'Silver' (Failed · 5m), and two sub-tasks 'Gold_1' and 'Gold_2' (Skipped). The 'Silver' task is highlighted with a red border and a red 'X' icon, indicating it failed. Below it, 'Gold_1' and 'Gold_2' are also highlighted with red borders and blue checkmarks, indicating they were skipped. Each task card shows details like 'Shared autoscaling' and '/path/to/notebook', and an 'Edit' button. On the right, a 'Summary of changes' panel is visible, containing a 'Summary' field with the text 'Changed cluster on silver', a 'Description' field with the text 'The shared autoscaling cluster was acting up on Silver for some reason. I'm moving it over to the BI integrations testing cluster as I've found it more reliable', and a 'Rerun' button.



Navigating the Jobs UI

Use breadcrumbs to navigate back to your **job** from a specific run page

The screenshot displays the Databricks Jobs UI. At the top, a breadcrumb trail reads: [Workflows](#) > [Jobs](#) > **student-lr0f-da-delp: Example Job** > [Run 92643](#). The breadcrumb **student-lr0f-da-delp: Example Job** is highlighted with a blue box. Below the breadcrumbs, the page title is **student-lr0f-da-delp: Example Job run**.

The main content area shows a job run visualization with two steps: **Reset** (Succeeded · 18s) and **DLT** (Succeeded · 4m 1s). The **DLT** step is expanded to show it is a **Delta Live Table Pipeline**. An arrow points from the **Reset** step to the **DLT** step.

On the right side, the **Job run details** panel is visible, containing the following information:

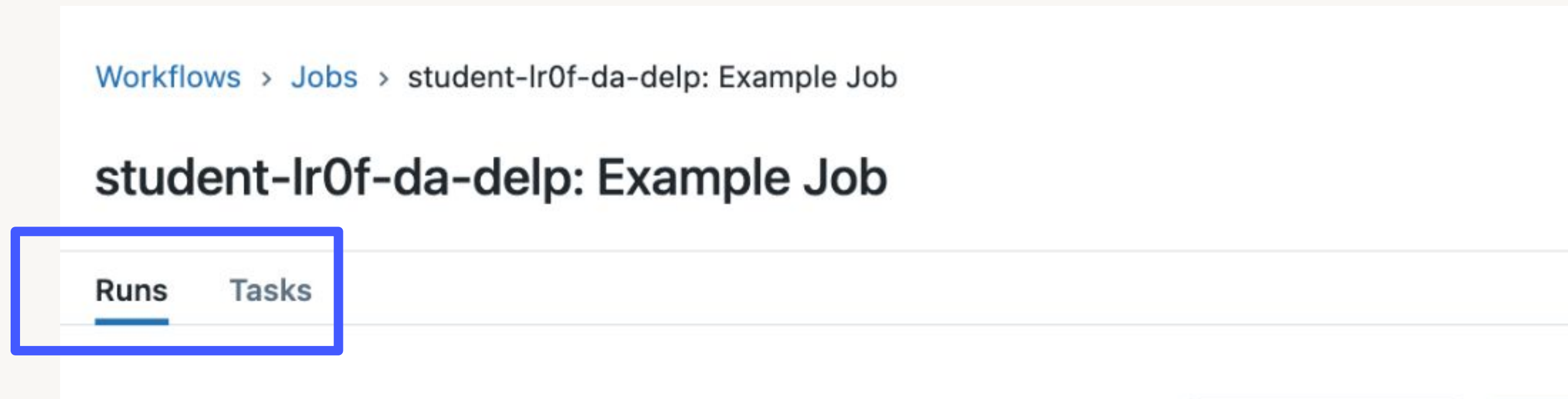
- Job ID**: [354156443889724](#) (copy icon)
- Job run ID**: 92643 (copy icon)
- Started**: 2023-01-06 00:49:55 EST
- Ended**: (partially visible)

Navigating the Jobs UI

Runs vs Tasks tabs on the job page

Use **Runs** tab to view completed or active runs for the job

Use **Tasks** tab to modify or add tasks to the job

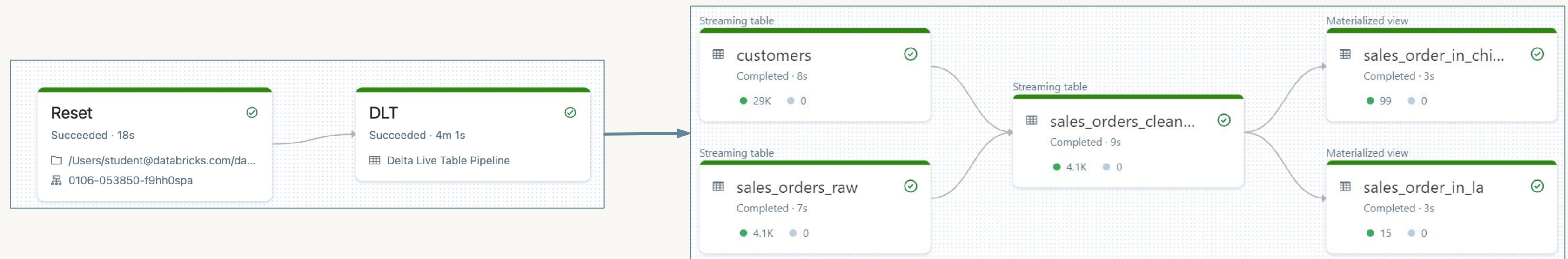


DE 5.1.1: Task Orchestration

Demo: Task Orchestration

DE 5.1.1 – Task Orchestration

- Schedule a notebook task in a Databricks Workflow Job
- Describe job scheduling options and differences between cluster types
- Review Job Runs to track progress and see results
- Schedule a DLT pipeline task in a Databricks Workflow Job
- Configure dependency between tasks via Databricks Workflows UI



DE 5.2.1.L: Task Orchestration Lab

Lab: Task Orchestration

DE 5.2.1.L – Task Orchestration

