

# Workflows Lab & Advanced Scheduling Options

---

Exploring workflow setups and scheduling features



# Agenda Overview

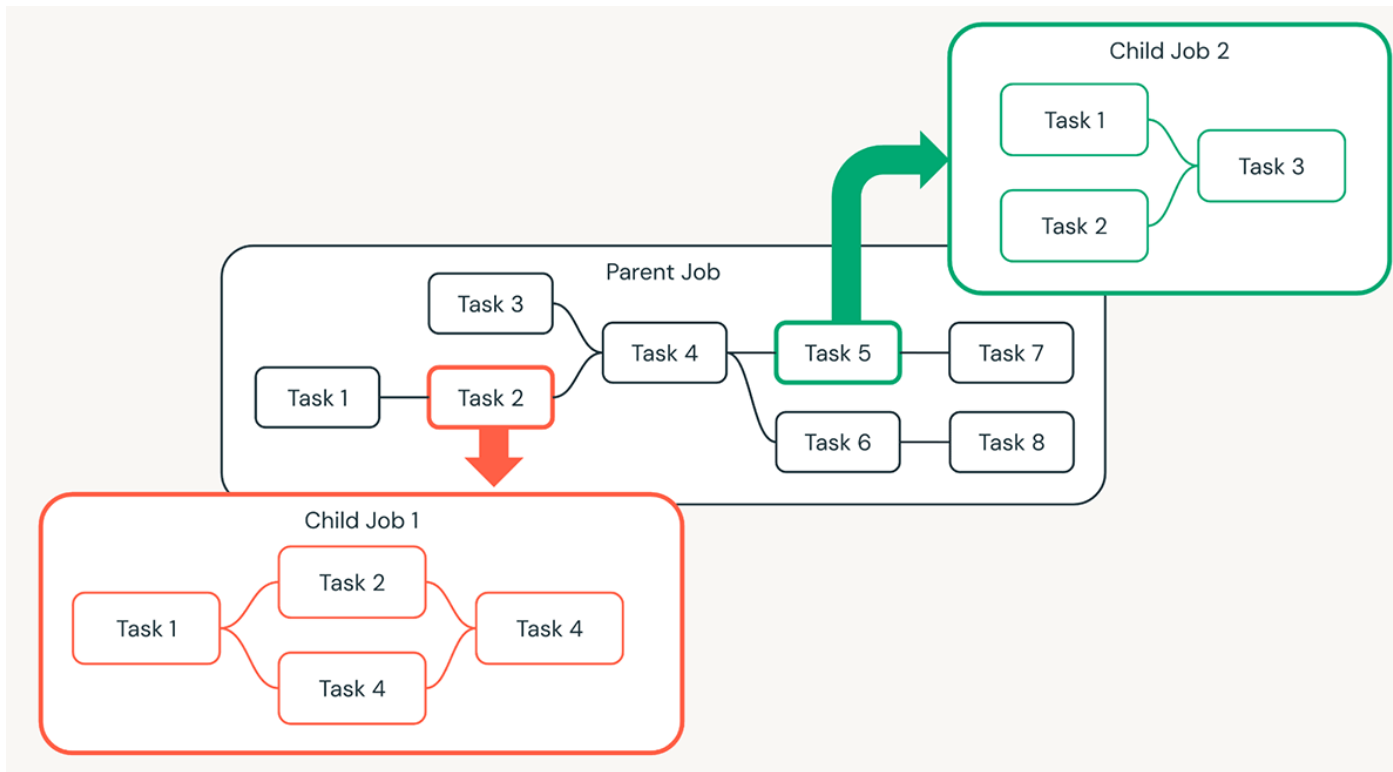
- End-to-End Workflow Setup
- Demo: Building an End-to-End Workflow
- Sample Code for Notebooks
- Advanced Scheduling Features
- Cron Expression Examples
- Hands-On Lab and Challenge



# End-to-End Workflow Setup

---

# Key Points of Workflow Setup



## Chaining Tasks

Chaining multiple tasks in a job enhances efficiency by organizing and managing various processes together.

## Output Passing

Passing outputs between tasks is essential for creating seamless data flows in workflows, ensuring continuity.

## Setting Dependencies

Establishing dependencies is crucial for reliable data pipelines, ensuring tasks execute in the correct order.

# Workflow DAG

## Understanding DAG

A Directed Acyclic Graph is used to represent workflows with multiple tasks that have dependencies.

## Task Dependencies

In a workflow DAG, tasks are interconnected, and their execution order is determined by their dependencies.

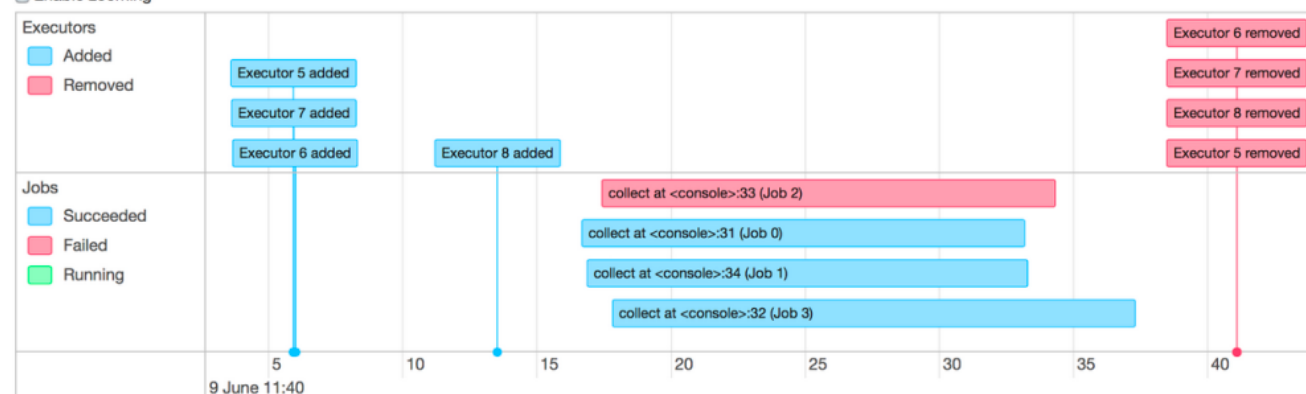
## Applications of DAG

DAGs are widely used in computer science, project management, and scheduling to optimize workflows.

### Spark Jobs (?)

Total Uptime: 2.2 min  
Scheduling Mode: FIFO  
Completed Jobs: 3  
Failed Jobs: 1

▼ Event Timeline  
✓ Enable zooming



# Demo: Building an End-to-End Workflow

---

# Sample Code for Notebooks

---

```
data_prep_notebook  
dbutils.widgets.text("input_data", "LabData2025",  
"Input Data Name")  
input_data = dbutils.widgets.get("input_data")  
import pandas as pd  
  
Simulate ETL  
df = pd.DataFrame({'id': range(10), 'value': [x*2 for x  
in range(10)]})  
df_clean = df[df['id'] % 2 == 0]  
dbutils.notebook.exit(df_clean.to_json())
```



# Notebook 1: Data Prep

## Data Input Widget

The code snippet initializes a widget for inputting the data name, crucial for data processing workflows.

## ETL Simulation

The sample code simulates an Extract, Transform, Load (ETL) process using a simple DataFrame.

## Data Cleaning

The DataFrame is filtered to clean the data, which is essential for accurate analysis and insights.

```
# Notebook: data_prep_notebook
dbutils.widgets.text("input_data",
"LabData2025", "Input Data Name")
input_data =
dbutils.widgets.get("input_data")
import pandas as pd
# Simulate ETL
df = pd.DataFrame({'id': range(10), 'value':
[x*2 for x in range(10)]})
df_clean = df[df['id'] % 2 == 0]
dbutils.notebook.exit(df_clean.to_json())
```



# Notebook 2: Data Analysis

## Sample Code Overview

This slide presents sample code for data analysis using Python. It illustrates how to read JSON data and calculate sums.

## Data Input Widget

The code utilizes a widget to input JSON data, enhancing interactivity for users conducting data analysis.

## DataFrame and Analysis

Using Pandas, the code converts JSON into a DataFrame and performs analysis, such as summing specific values.

```
data_analysis_notebook
```

```
dbutils.widgets.text("input_json", "",  
"Input Data JSON")  
import pandas as pd  
input_json =  
dbutils.widgets.get("input_json")  
df = pd.read_json(input_json)  
print("Sum of values:",  
df['value'].sum())  
display(df)
```



# Advanced Scheduling Features

---

# Key Features of Advanced Scheduling

The screenshot displays a workflow management interface for a job named 'revenue\_job'. The interface is divided into two main sections: a task graph on the left and a 'Job details' panel on the right.

**Task Graph:** The graph shows a sequence of tasks: 'ingest\_revenue' (labeled 1), 'check\_nulls' (labeled 2), 'aggregate\_customer\_data' (labeled 3), and 'run\_data\_quality\_validation' (labeled 4). The 'ingest\_revenue' task is highlighted with a blue border. A '+ Add task' button is visible below the graph.

**Job details panel:** This panel contains various configuration options for the job, including 'Job ID', 'Creator', 'Run as', 'Tags', 'Description', 'Lineage', and 'Git'. A red box labeled '5' highlights the 'Schedules & Triggers' section, which shows the job is scheduled to run 'Every day, next run at Jan 04, 2025, 11:29 AM'. Below this, there are buttons for 'Edit trigger', 'Pause', and 'Delete'.

**Form Fields:** Below the task graph, there are form fields for 'Task name\*' (containing 'ingest\_revenue'), 'Type\*' (set to 'Delta Live Tables pipeline'), and 'Pipeline\*' (set to 'ingest\_revenue'). There is also a checkbox for 'Trigger a full refresh on the Delta Live Tables pipeline' and a 'Depends on' dropdown menu.

## Multiple Schedules

Advanced scheduling allows for setting multiple schedules for the same job, enhancing flexibility and efficiency.

## Complex Cron Expressions

Users can utilize complex cron expressions to run jobs on specific days and times, providing precise control.

## Pause/Resume Controls

Advanced scheduling features include pause/resume options and concurrency controls to manage job execution effectively.

# Jobs UI Screenshot

## Advanced Scheduling Features

The Jobs UI includes advanced scheduling options that allow users to set multiple schedules for tasks.

## Cron Entry Box

The cron entry box enables users to define specific time intervals for job execution, providing flexibility.

### Schedule



#### Trigger Status

☒ Active

☐ Paused

#### Trigger type

Scheduled

#### Schedule type

Simple

Advanced

#### Schedule ⓘ

Every

Day

at

15

:

26

(UTC+00:00) UTC

☐ Show cron syntax

Cancel

Save

### Schedule



#### Trigger Status

☒ Active

☐ Paused

#### Trigger type

Scheduled

#### Schedule type

Simple

Advanced

#### Schedule ⓘ

7 26 15 \* \* ?

(UTC+00:00) UTC

☒ Show cron syntax

Cancel

Save



# Cron Expression Examples



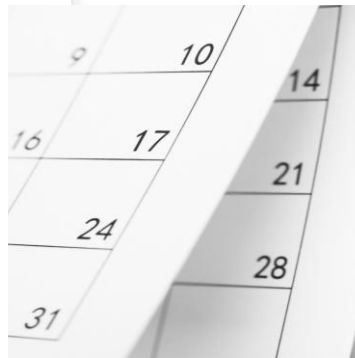
# Common Cron Expressions

---



## Weekday Scheduling

Cron expression for executing tasks every weekday at 8 AM is represented by `0 8 * * 1-5`.



## Monthly Scheduling

To schedule a task on the first day of every month, use the expression `0 0 1 * *`.



## Frequent Scheduling

The expression `*/30 * * * *` allows tasks to run every 30 minutes, ensuring frequent execution.

# Hands-On Lab and Challenge

---

# Lab Objectives and Steps

## **Workflow Creation**

Build a comprehensive workflow consisting of at least two tasks, ensuring proper dependencies are configured.

## **Data Passing Between Tasks**

Utilize notebook exit and parameter passing to effectively transfer data between the tasks in the workflow.

## **Scheduling Workflows**

Incorporate at least two different schedules into the workflow, using an advanced cron expression for one of them.

---