DEVI SREE PENDYALA

Final Project Report: Road Traffic Data Analysis Using Apache Airflow

Building ETL Data Pipelines with Bash Operator using Apache Airflow

Project Overview

As a **Data Engineer** at a data analytics consulting company, I was assigned a project aimed at reducing congestion on **national highways** by analyzing traffic data from various **toll plazas**. Each highway was operated by a different toll operator, utilizing **different IT setups and file formats**. My task was to collect and consolidate this data into a **single structured file** for further analysis.

Project Objectives

To streamline the data pipeline, I developed an **Apache Airflow DAG** that performed the following operations:

✅ **Extracting Data** from multiple sources:

- **CSV files** containing vehicle entry logs
- **TSV files** with toll plaza identifiers
- **Fixed-width files** with payment details

✅ **Transforming the Data** by standardizing field formats and structuring the dataset for analysis

✅ **Loading the Transformed Data** into a **staging area** for further processing

Implementation Approach

To achieve the project objectives, I designed and implemented an **Apache Airflow DAG** that automated the ETL process. The DAG:

- Unzipped and processed raw data from different file formats
- Extracted specific fields from each file type using **Bash commands**
- Merged data using **the paste command** to create a consolidated dataset
- Transformed certain fields, such as converting vehicle types to uppercase for consistency
- Ensured efficient scheduling and execution using **Apache Airflow**

Technologies Used

🚀 **CloudIDE (Theia-based environment)** for development
- 🔹 **Apache Airflow** for workflow orchestration
- 🔹 **Bash & Shell Scripting** for data extraction and transformation
- 🔹 **CSV, TSV, and Fixed-Width File Handling**

Task-01: Project Setup: Lab Environment Configuration

To successfully implement the **ETL pipeline** for analyzing toll plaza traffic data, the following steps were performed to set up the lab environment:

1. **Start Apache Airflow**
    - Launched Apache Airflow to orchestrate the ETL workflow.
2. **Created a Directory Structure for the Staging Area**
    - A directory was created to store extracted and transformed data before final processing. The following command was used to create the required directory structure:

sudo mkdir -p /home/project/airflow/dags/finalassignment/staging

3. **Set Appropriate Permissions for the Directories**
    - To ensure seamless data processing, permissions were assigned to the directories using the following command:

sudo chmod -R 777 /home/project/airflow/dags/finalassignment

4. **Downloaded the Dataset Using the curl Command**
    - The required dataset was fetched from the source and stored in the appropriate directory using the command:

sudo curl https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/tolldata.tgz -o /home/project/airflow/dags/finalassignment/tolldata.tgz

This setup ensured that the **working environment was correctly structured, secured, and populated with the necessary data** to facilitate the **ETL workflow execution** in Apache Airflow.

Task-02 Creating imports, DAG argument, definition, tasks using BashOperator:

Code:

```
# import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow.models import DAG
# Operators; you need this to write tasks!
from airflow.operators.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago

# defining DAG arguments

default_args = {
    'owner': 'your_name_here',
    'start_date': days_ago(0),
    'email': ['your_email_here'],
    'email_on_failure': True,
    'email_on_retry': True,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

# defining the DAG

dag = DAG(
    'ETL_toll_data',
```

```
    default_args=default_args,
    description='Apache Airflow Final Assignment',
    schedule_interval=timedelta(days=1),
)

# defining the tasks

# Task 1: Unzip data
unzip_data = BashOperator(
    task_id='unzip_data',
    bash_command='tar -xvf tolldata.tgz -C /home/project/airflow/dags/',
    dag=dag,
)

# Task 2: Extract data from CSV
extract_data_from_csv = BashOperator(
    task_id='extract_data_from_csv',
    bash_command='cut -d"," -f1,2,3,4 vehicle-data.csv >
/home/project/airflow/dags/csv_data.csv',
    dag=dag,
)

# Task 3: Extract data from TSV
extract_data_from_tsv = BashOperator(
    task_id='extract_data_from_tsv',
    bash_command='cut -d"\t" -f5,6,7 tollplaza-data.tsv >
/home/project/airflow/dags/tsv_data.csv',
    dag=dag,
```

```
)

# Task 4: Extract data from Fixed Width file
extract_data_from_fixed_width = BashOperator(
    task_id='extract_data_from_fixed_width',
    bash_command='cut -c59-62,63-67 payment-data.txt > /home/project/airflow/dags/fixed_width_data.csv',
    dag=dag,
)

# Task 5: Consolidate data
consolidate_data = BashOperator(
    task_id='consolidate_data',
    bash_command='paste /home/project/airflow/dags/csv_data.csv /home/project/airflow/dags/tsv_data.csv /home/project/airflow/dags/fixed_width_data.csv > /home/project/airflow/dags/extracted_data.csv',
    dag=dag,
)

# Task 6: Transform data
transform_data = BashOperator(
    task_id='transform_data',
    bash_command='tr "[a-z]" "[A-Z]" < /home/project/airflow/dags/extracted_data.csv > /home/project/airflow/dags/transformed_data.csv',
    dag=dag,
)

# Task Pipeline
```

unzip_data >> extract_data_from_csv >> extract_data_from_tsv >>
extract_data_from_fixed_width >> consolidate_data >> transform_data

Explanation for DAG:

This Apache Airflow DAG automates an ETL (Extract, Transform, Load) pipeline for processing toll data. It consists of multiple tasks, each executed using BashOperator, to handle data extraction from different file formats, consolidate the extracted data, and apply transformations. Below is a detailed explanation of each section of the code:

1. Importing Required Libraries

The script imports essential libraries to define and manage the DAG:

timedelta: Used to specify scheduling intervals.

DAG: Core component to define workflow structure.

BashOperator: Executes shell commands as tasks.

days_ago: Simplifies setting the DAG start date.

2. Defining DAG Arguments

The default_args dictionary configures key execution parameters:

Owner: Specifies the DAG owner.

Start Date: Defines when the DAG starts execution (days_ago(0) makes it run from today).

Email Notifications: Sends alerts on failure or retry.

Retries: Specifies one retry attempt with a 5-minute delay in case of task failure.

3. Defining the DAG

The DAG object named 'ETL_toll_data' is instantiated with:

The predefined default arguments.

A description explaining the workflow.

A schedule interval of one day (timedelta(days=1)), meaning the DAG will run daily.

4. Defining Tasks using BashOperator

Each task executes a shell command to process toll data:

Task 1: Unzipping Data

Extracts the compressed tolldata.tgz file into the DAG's directory.

Task 2: Extract Data from CSV

Extracts columns 1 to 4 (e.g., vehicle-data.csv) using cut and saves them to csv_data.csv.

Task 3: Extract Data from TSV

Extracts columns 5 to 7 from tollplaza-data.tsv, storing results in tsv_data.csv.

Task 4: Extract Data from Fixed-Width File

Extracts specific character positions (columns 59-62, 63-67) from payment-data.txt, saving them in fixed_width_data.csv.

Task 5: Consolidating Extracted Data

Merges CSV, TSV, and Fixed Width extracted data into a single file extracted_data.csv using paste.

Task 6: Transforming Data

Converts all lowercase letters to uppercase in the final extracted_data.csv, outputting transformed_data.csv.

5. Defining Task Execution Order (Dependencies)

The >> operator ensures a sequential workflow:

unzip_data >> extract_data_from_csv >> extract_data_from_tsv >> extract_data_from_fixed_width >> consolidate_data >> transform_data

First, the compressed data is extracted.

Then, data extraction from CSV, TSV, and Fixed Width files happens in sequence.

Extracted data is merged into one file.

Finally, text transformation (uppercase conversion) is applied.

This Airflow DAG streamlines ETL automation for toll data processing by:

✓ Extracting structured data from multiple formats.

✓ Consolidating information into a unified dataset.

✓ Transforming text data for consistency.

✓ Automating execution with retry mechanisms for reliability.

This efficient, well-structured pipeline ensures data is processed, cleaned, and prepared for further analysis or storage.

Task -03: Submitting the ETL_toll_data DAG

To successfully deploy and execute the **ETL_toll_data** DAG in **Apache Airflow**, I followed the steps below:

Step 1: Setting the AIRFLOW_HOME Variable

To ensure that Apache Airflow runs in the correct directory, I configured the AIRFLOW_HOME environment variable using the following commands:

```
export AIRFLOW_HOME=/home/project/airflow

echo $AIRFLOW_HOME
```

This step confirmed that the Airflow home directory was correctly set.

Step 2: Copying the DAG File to the Airflow DAGs Directory

Once the **ETL_toll_data.py** DAG script was developed, I submitted it to Airflow by copying it into the DAGs folder:

```
cp ETL_toll_data.py $AIRFLOW_HOME/dags
```

Step 3: Verifying the DAG Submission

To check whether the DAG was successfully submitted and registered in Airflow, I listed all existing DAGs using:

```
airflow dags list
```

This displayed all DAGs currently available in the system.

Step 4: Ensuring the DAG Is Listed in the Output

To confirm that **ETL_toll_data** was correctly recognized by Airflow, I ran the following command:

```
airflow dags list | grep "ETL_toll_data"
```

If the DAG was successfully submitted, it appeared in the output list.

Step 5: Listing All Tasks in the DAG

To verify that all tasks were correctly added to the DAG, I used:

```
airflow tasks list ETL_toll_data
```

- This displayed all the tasks defined in the DAG, including:
  - ✅ unzip_data
  - ✅ extract_data_from_csv
  - ✅ extract_data_from_tsv
  - ✅ extract_data_from_fixed_width
  - ✅ consolidate_data
  - ✅ transform_data
  - ✅ task_pipeline

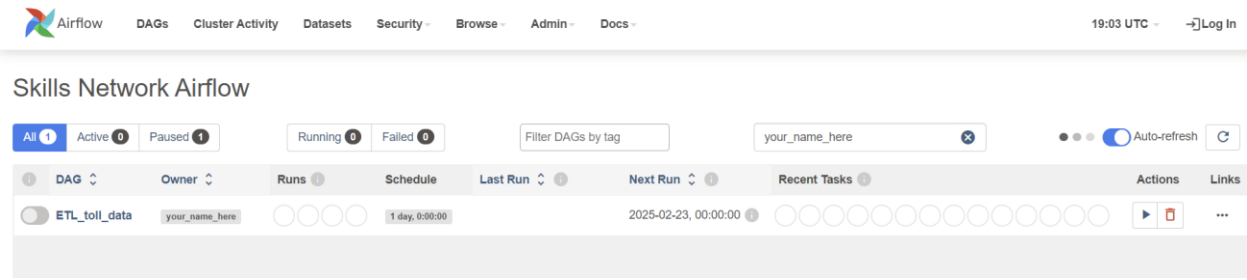
By completing these steps, I successfully deployed and validated the **ETL_toll_data** DAG, ensuring that the workflow was properly configured and ready for execution.

Task-04: Project Validation: Getting the DAG Operational

To ensure that the **ETL_toll_data** DAG was successfully deployed and operational, I performed a series of validation steps, as outlined below. Screenshots have been attached as evidence for each step.

Step 1: DAG Submission Verification

I validated that the **ETL_toll_data** DAG was successfully submitted and recognized by Airflow.



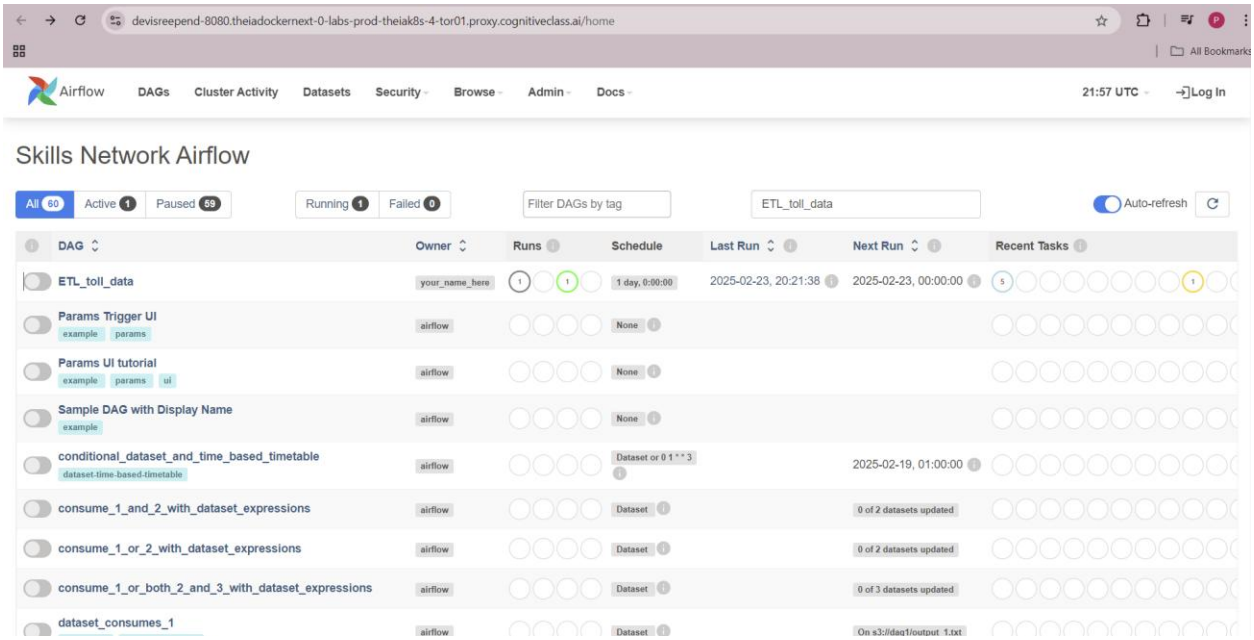
Step 2: DAG Monitoring and Execution Details

During the execution of the **ETL_toll_data** DAG, I utilized the **Airflow Web UI** to monitor its status and verify its operational details. The following key information was observed:

- **Owner of the DAG:** Identifies the user responsible for the DAG.
- **Number of DAG Runs:** Displays how many times the DAG has been executed.
- **DAG Schedule:** Indicates the scheduled frequency for DAG execution.
- **Last Run Time:** Shows the timestamp of the most recent execution.

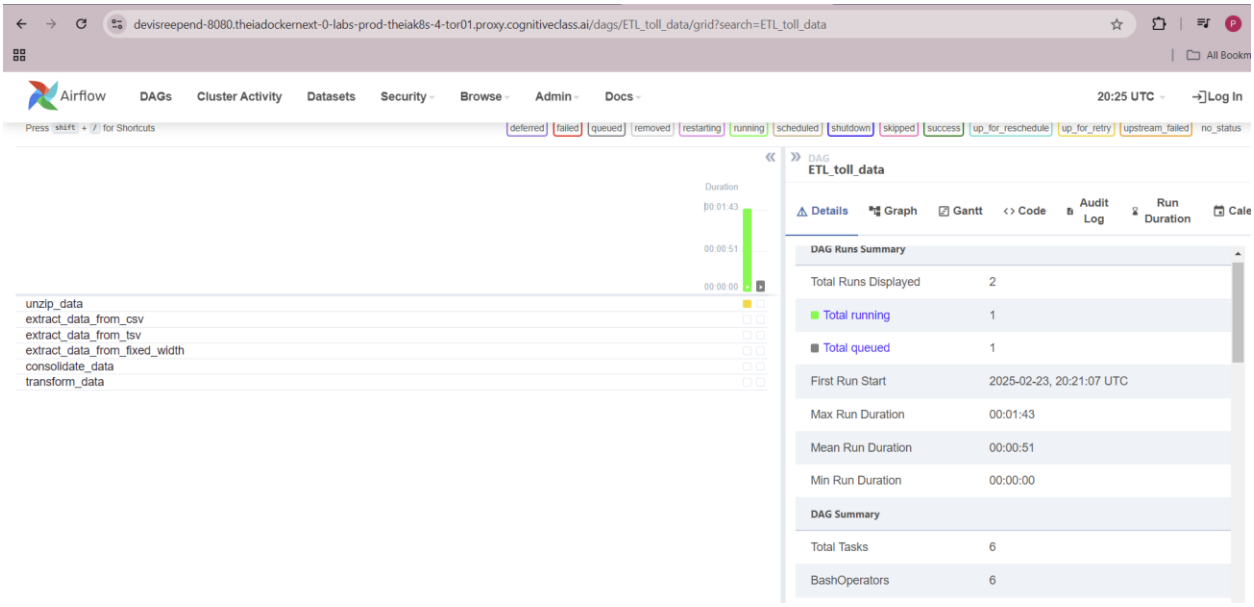
- **Recent Task Status:** Provides insights into the execution status of individual tasks within the DAG, helping to identify successful runs or any failures.

This information allowed me to validate the **proper execution, scheduling, and task completion** of the ETL pipeline, ensuring that the workflow was functioning as expected.

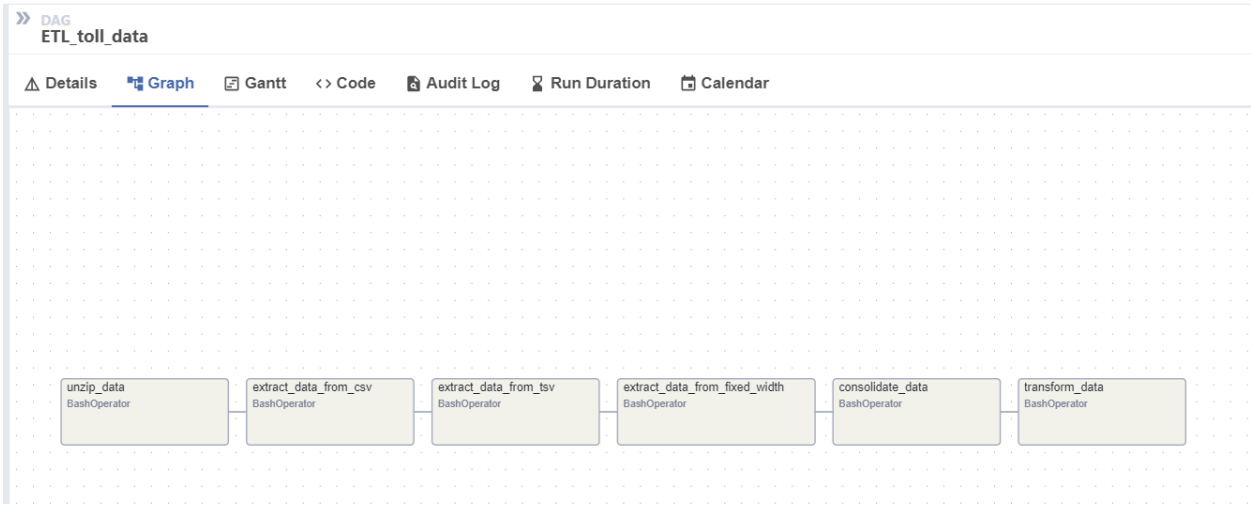


Detailed view of DAG:

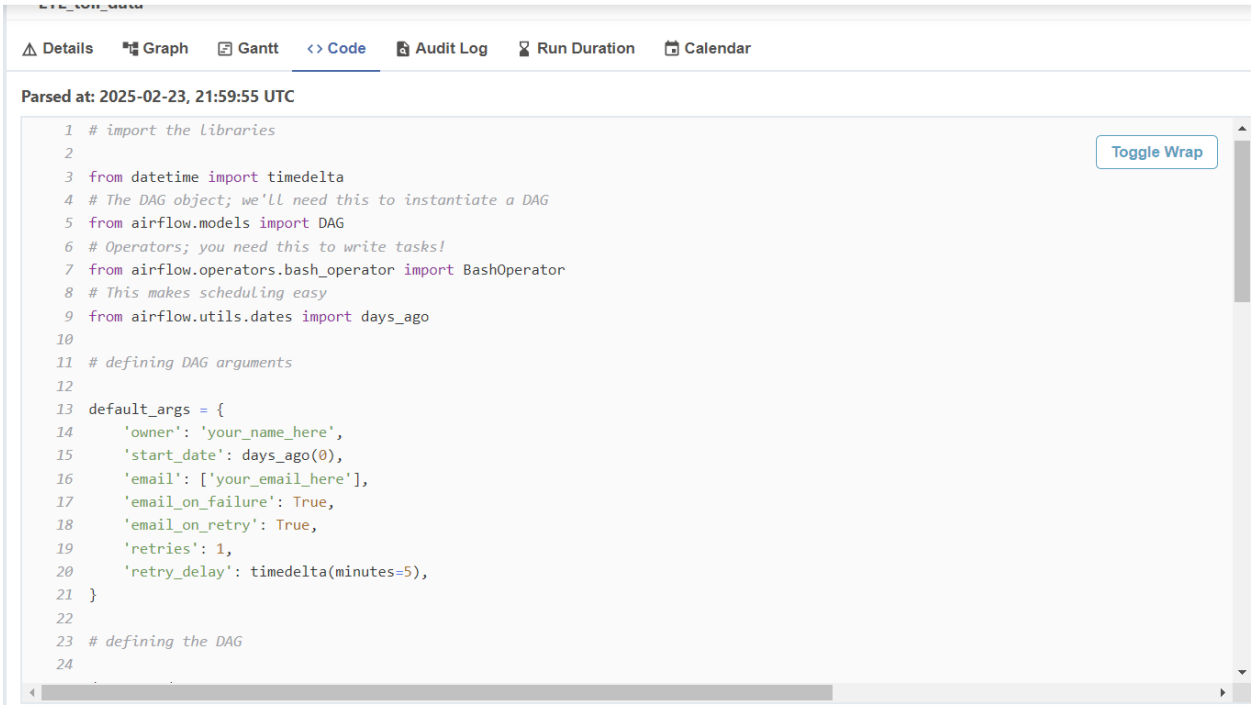
Grid view of DAG Tasks:



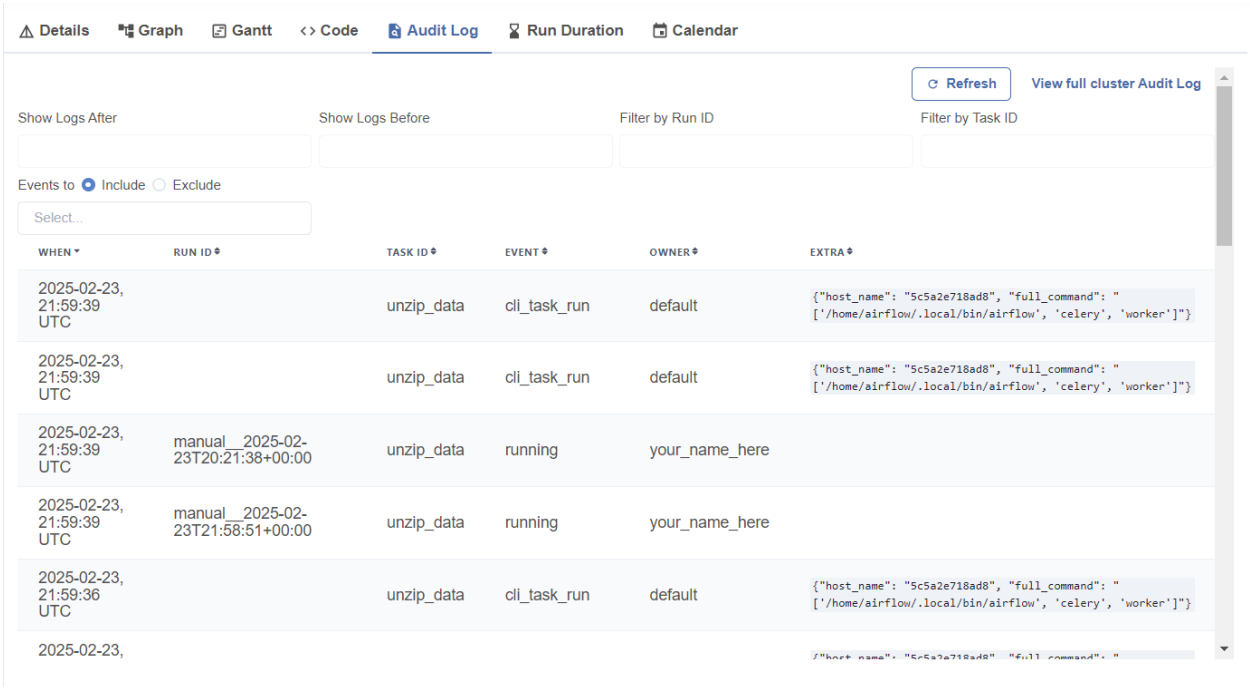
Graph view of DAG:



Code view:



Audit Log:


Details
Graph
Gantt
Code
Audit Log
Run Duration
Calendar
Refresh
View full cluster Audit Log
Show Logs After
Show Logs Before
Filter by Run ID
Filter by Task ID
Events to Include Exclude
Select...
WHEN
RUN ID
TASK ID
EVENT
OWNER
EXTRA
2025-02-23, 21:59:39 UTC
unzip_data
cli_task_run
default
{"host_name": "5c5a2e718ad8", "full_command": "['/home/airflow/.local/bin/airflow', 'celery', 'worker']"}
2025-02-23, 21:59:39 UTC
unzip_data
cli_task_run
default
{"host_name": "5c5a2e718ad8", "full_command": "['/home/airflow/.local/bin/airflow', 'celery', 'worker']"}
2025-02-23, 21:59:39 UTC
manual__2025-02-23T20:21:38+00:00
unzip_data
running
your_name_here
2025-02-23, 21:59:39 UTC
manual__2025-02-23T21:58:51+00:00
unzip_data
running
your_name_here
2025-02-23, 21:59:36 UTC
unzip_data
cli_task_run
default
{"host_name": "5c5a2e718ad8", "full_command": "['/home/airflow/.local/bin/airflow', 'celery', 'worker']"}
2025-02-23,