Mastering Advanced Dataset Scheduling in Apache Airflow

Ankit Chaurasia

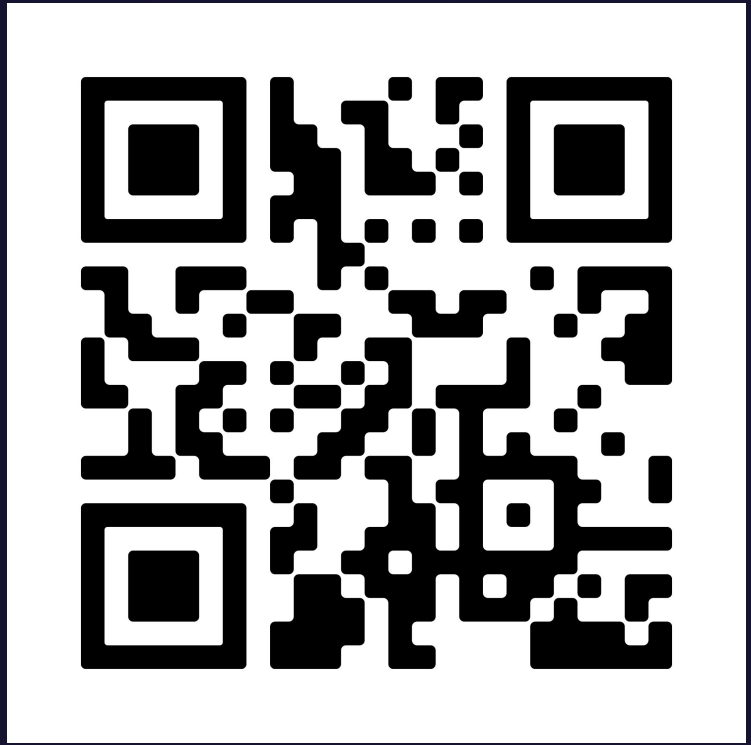
ASTRONOMER

Hi, I'm

Ankit Chaurasia 👨‍💻

- Senior Software Engineer at ASTRONOMER
- Using **Airflow** since April 2021
- **Airflow** Contributor
- https://ankitchaurasia.info/
- ankit.chaurasia@astronomer.io

Scan the QR code to view the slide.



Do you want to:

Schedule DAGs with complex dataset conditions?

Combine dataset updates and time-based triggers for flexible scheduling?

Explore new API endpoints for managing datasets?

Use DatasetAliases for dynamic scheduling?

Real World Application

Batch Processing in Real-Time

Machine Learning Model Training

Data Synchronization


I am about to show you how it's done.

Agenda

1. Introduction to Data-aware Scheduling
2. Advanced Scheduling Techniques
3. API Endpoints for Dataset Management
4. Scheduling DAGs with DatasetAlias
5. Best Practices & Tips

Introduction to Data-aware Scheduling

What is a Dataset?

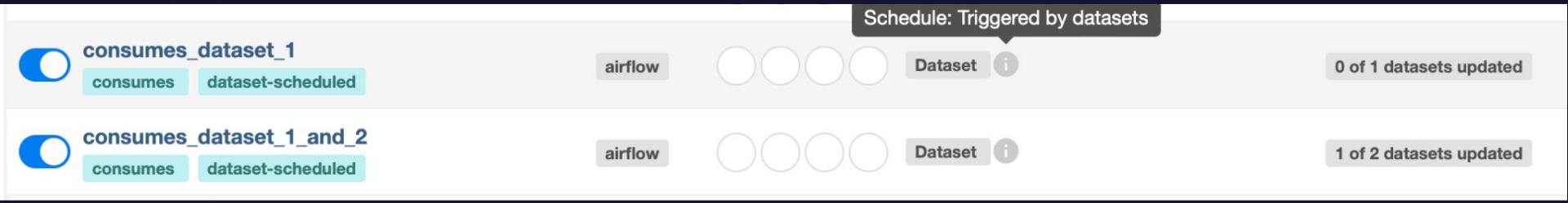
- A dataset is a logical representation of data that DAGs can produce or consume.

```
from airflow.datasets import Dataset

example_dataset = Dataset("s3://dataset-bucket/example.csv")
```

Introduction to Data-aware Scheduling

What is Data-Aware Scheduling?

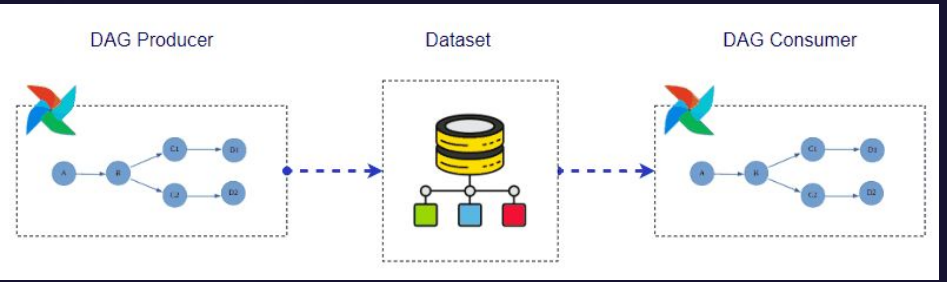
- Scheduling DAGs based on your data changing, using Datasets.
- In Airflow 2.4, you can now schedule DAGs to run based on dataset updates in addition to time-based triggers.



Introduction to Data-aware Scheduling

How It Works:

- Producer DAGs
- Consumer DAGs




```python
from airflow.datasets import Dataset


example_dataset = Dataset("s3://dataset-bucket/example.csv")

# Producer DAG
with DAG(dag_id="producer", ...):
    MyOperator(
        task_id="producer_task",
        outlets=[example_dataset],
        ...,
    )

# Consumer DAG
with DAG(dag_id="consumer", schedule=[example_dataset], ...):
```

Conditional dataset expressions

From Airflow 2.9 and later

And (&)

```
with DAG(
    # Consume dataset 1 and 2 with
    # dataset expressions
    schedule=(dataset1 & dataset12),
    ...,
):
    ...
```

OR (|)

```
with DAG(
    # Consume dataset 1 or 2 with
    # dataset expressions
    schedule=(dataset1 | dataset12),
    ...,
):
    ...
```

Create more complex dataset expressions

```python
with DAG(
    dag_id="complex_dataset_scheduling",
    ...
    schedule=((dataset1 | dataset2) & (dataset3 | dataset4)),
    # Use () instead of [] to be able to use conditional dataset scheduling!
) as dag:
    ...
```

Datasets needed to trigger the next run for
complex_dataset_scheduling

Conditions

```
{
  "all": [
    {
      "any": [
        "s3://dag1/output_1.txt",
        "s3://dag2/output_2.txt"
      ]
    },
    {
      "any": [
        "s3://dag3/output_3.txt",
        "s3://dag4/output_4.txt"
      ]
    }
  ]
}
```

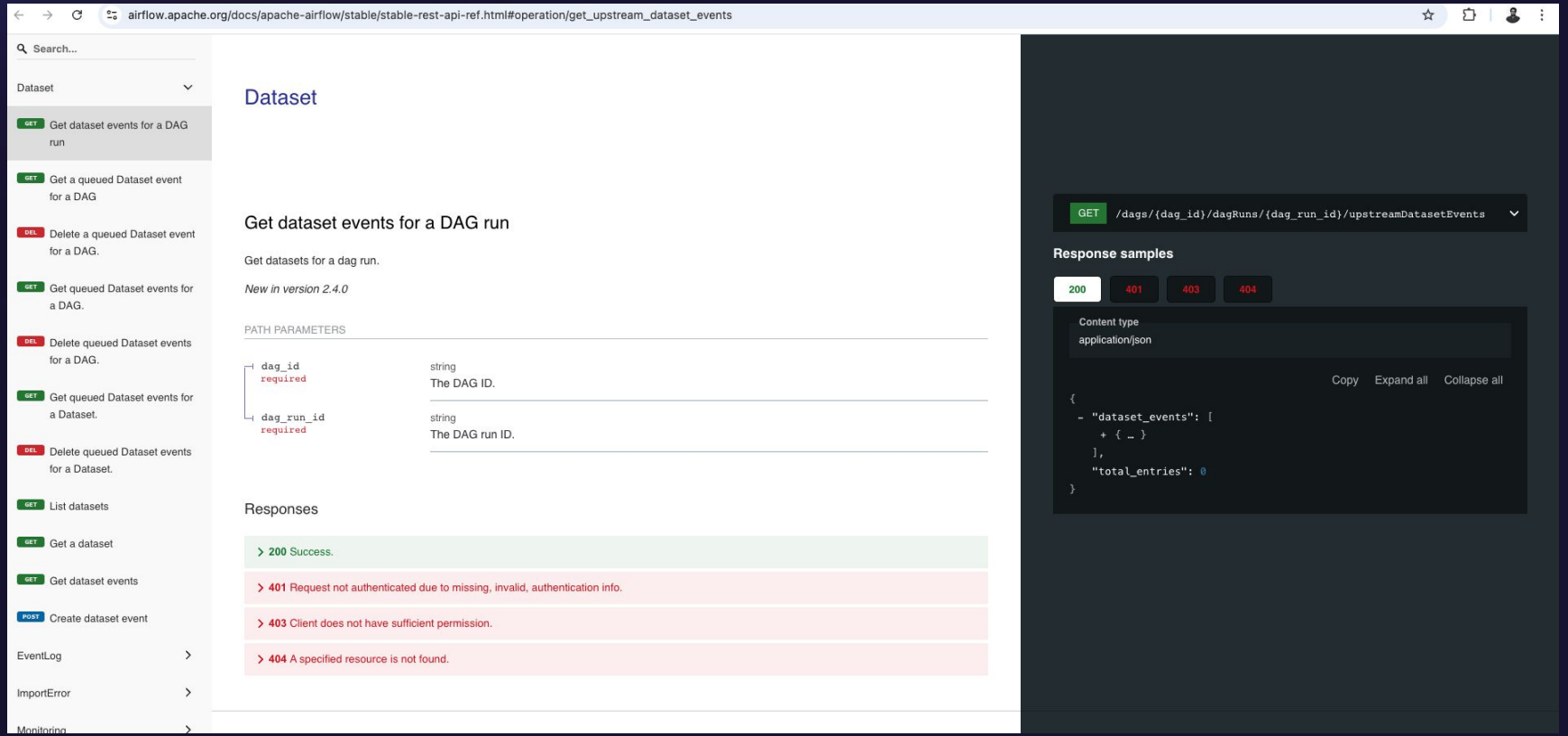
0 of 4 datasets updated

Dataset URI	Latest Update since last Dag Run
s3://dag1/output_1.txt	2024-08-21, 00:05:50
s3://dag2/output_2.txt	
s3://dag3/output_3.txt	
s3://dag4/output_4.txt	

Combined Dataset and Time-Based Scheduling

```python
from airflow.timetables.datasets import DatasetOrTimeSchedule
from airflow.timetables.trigger import CronTriggerTimetable

with DAG(
    dag_id="conditional_dataset_and_time_based_timetable",
    ...
    schedule=DatasetOrTimeSchedule(
        timetable=CronTriggerTimetable("0 0 * * *", timezone="UTC"),
        datasets=(dataset3 & dataset4)
    ),
) as dag:
    ...
```

API Endpoints for Dataset Management



API Documentation: [Dataset API Endpoints]

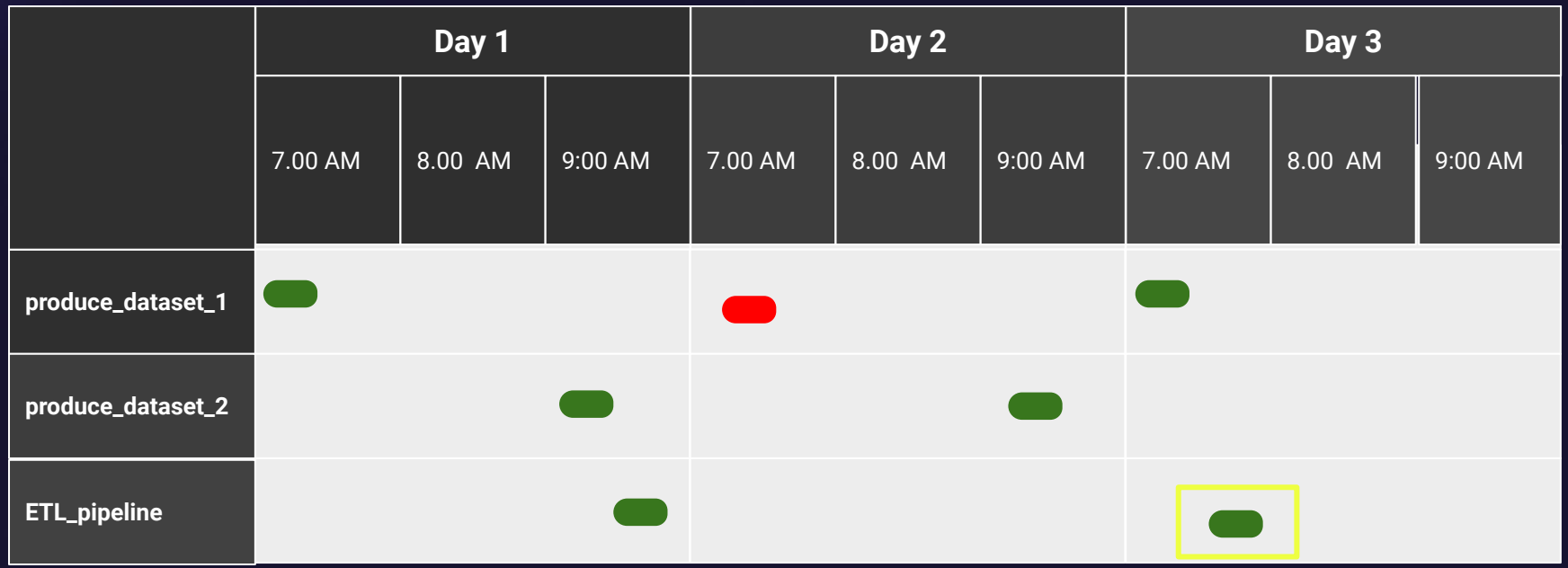
ASTRONOMER

Real-world example of Handling Out-of-Sync Triggers in Dependent DAGs

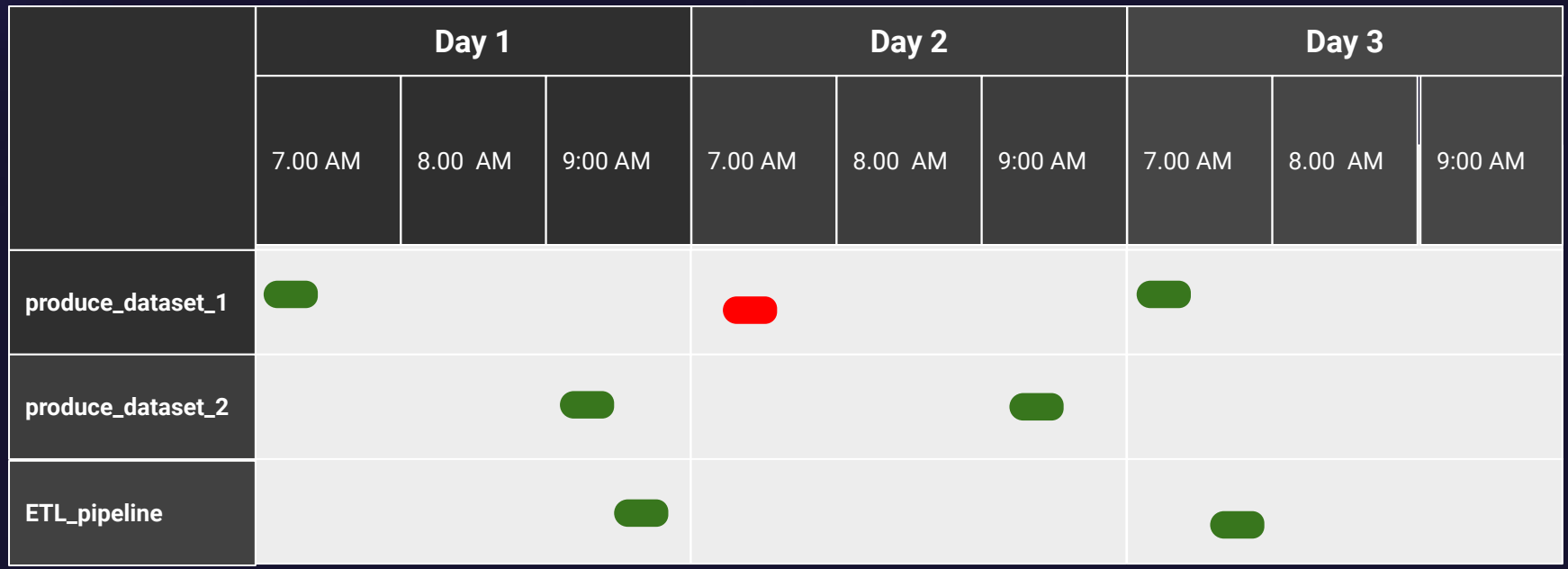
Scenario:

- Two DAGs (produce_dataset_1 and produce_dataset_2) generate DATASET1 and DATASET2 on a daily basis.
- ETL_pipeline depends on both datasets being updated on the same day before it triggers.

Real-world example of Handling Out-of-Sync Triggers in Dependent DAGs



Handling Out-of-Sync Triggers in Dependent DAGs



Real-world example of Handling Out-of-Sync Triggers in Dependent DAGs

Solution:

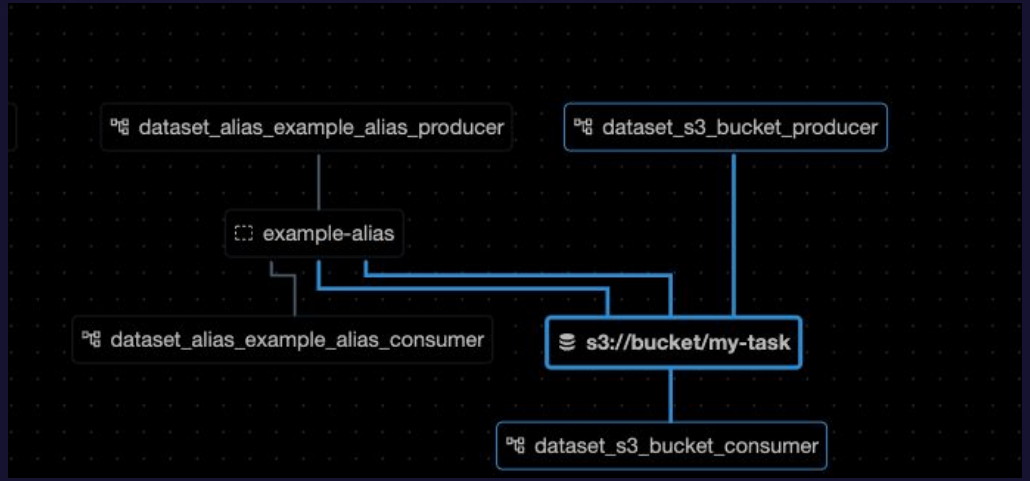
queuedEvent API endpoints are introduced to manually adjust trigger timestamps after out-of-sync events.

- Use API Endpoint: *DELETE /dags/{dag_id}/datasets/queuedEvent/{uri} to delete* produce_dataset_2 DAG for DATASET2 for Day 2.

Scheduling DAGs with DatasetAlias

What is a DatasetAlias?

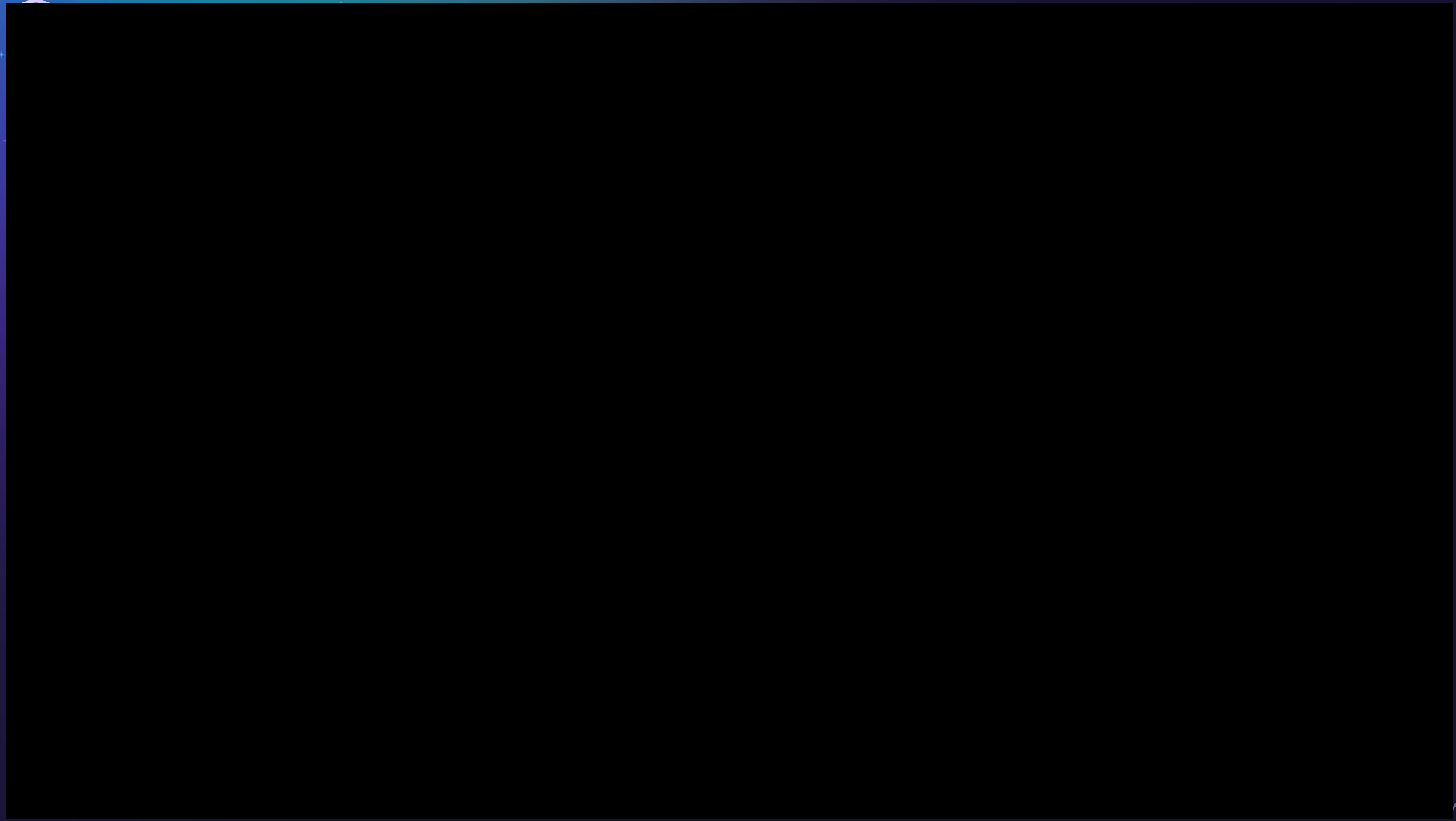
- DatasetAlias (Airflow 2.10+) is an object that can be associated to one or more datasets and used to create schedules based on datasets created at runtime`



Scheduling DAGs with DatasetAlias

Dataset aliases enable dynamic DAG triggering by resolving aliases to actual datasets during runtime, allowing flexible scheduling.

```python
from airflow.datasets import Dataset, DatasetAlias
from airflow.decorators import dag, task
from pendulum import datetime

# Producer DAG
@dag(dag_id="dataset-alias-producer", start_date=datetime(2024, 8, 1), catchup=False)
def produce_datasets_with_alias():
    @task(outlets=[DatasetAlias("example-alias")])
    def produce_dataset(*, outlet_events):
        outlet_events["example-alias"].add(Dataset("s3://bucket/my-task"))

    produce_dataset()

produce_datasets_with_alias()

# Consumer DAG
@dag(
    dag_id="dataset-alias-consumer",
    start_date=datetime(2024, 8, 1),
     schedule=DatasetAlias("example-alias")
     )
def consume_datasets_with_alias():
    @task
    def process_data():
        print("Processing data from dynamically created dataset.")

    process_data()

consume_datasets_with_alias()
```




INITECH

Best Practices and Tips

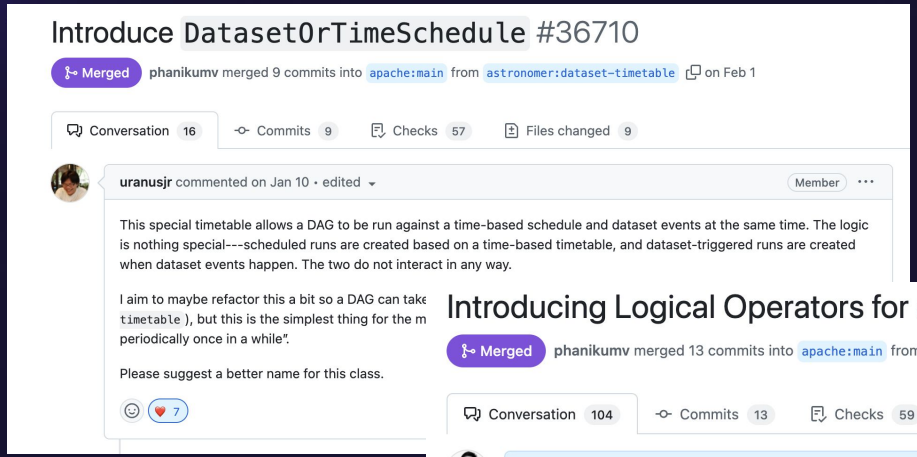
Design DAGs with Clear Dataset Dependencies

Monitor and Reset Dataset States Using API Endpoints

Handle Out-of-Sync Scenarios Proactively

Use DatasetorTime Schedule for flexibility

Interested to know how this was implemented?



Introduce `DatasetOrTimeSchedule` #36710

Merged **phanikumv** merged 9 commits into `apache:main` from `astronomer:dataset-timetable` on Feb 1

Conversation 16 · Commits 9 · Checks 57 · Files changed 9

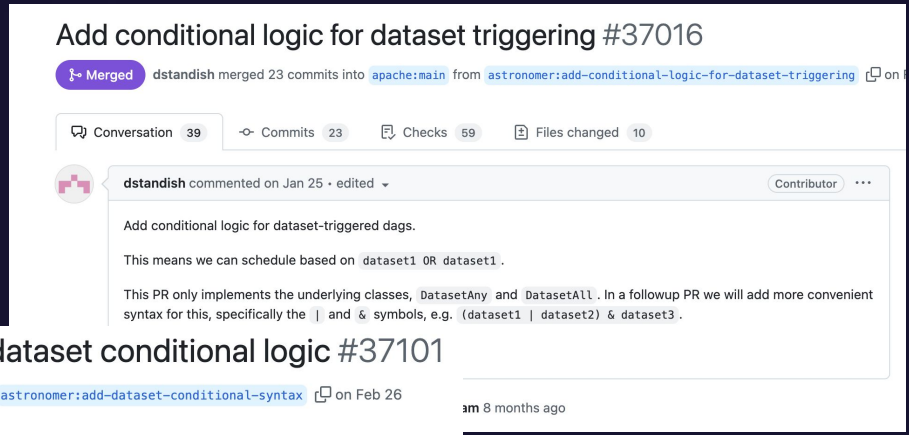
uranusjr commented on Jan 10 • edited — Member

This special timetable allows a DAG to be run against a time-based schedule and dataset events at the same time. The logic is nothing special---scheduled runs are created based on a time-based timetable, and dataset-triggered runs are created when dataset events happen. The two do not interact in any way.

I aim to maybe refactor this a bit so a DAG can take `timetable`), but this is the simplest thing for the m periodically once in a while".

Please suggest a better name for this class.

7



Add conditional logic for dataset triggering #37016

Merged **dstandish** merged 23 commits into `apache:main` from `astronomer:add-conditional-logic-for-dataset-triggering` on

Conversation 39 · Commits 23 · Checks 59 · Files changed 10

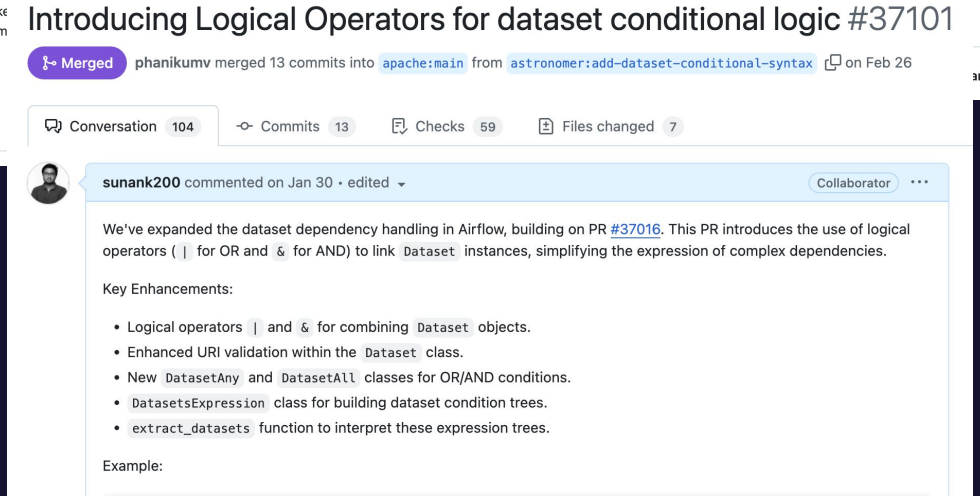
dstandish commented on Jan 25 • edited — Contributor

Add conditional logic for dataset-triggered dags.

This means we can schedule based on `dataset1 OR dataset1`.

This PR only implements the underlying classes, `DatasetAny` and `DatasetAll`. In a followup PR we will add more convenient syntax for this, specifically the `|` and `&` symbols, e.g. `(dataset1 | dataset2) & dataset3`.

am 8 months ago



Introducing Logical Operators for dataset conditional logic #37101

Merged **phanikumv** merged 13 commits into `apache:main` from `astronomer:add-dataset-conditional-syntax` on Feb 26

Conversation 104 · Commits 13 · Checks 59 · Files changed 7

sunank200 commented on Jan 30 • edited — Collaborator

We've expanded the dataset dependency handling in Airflow, building on PR #37016. This PR introduces the use of logical operators (`|` for OR and `&` for AND) to link `Dataset` instances, simplifying the expression of complex dependencies.

Key Enhancements:

- Logical operators `|` and `&` for combining `Dataset` objects.
- Enhanced URI validation within the `Dataset` class.
- New `DatasetAny` and `DatasetAll` classes for OR/AND conditions.
- `DatasetsExpression` class for building dataset condition trees.
- `extract_datasets` function to interpret these expression trees.

Example:

Thank you!
Any questions?