Data Lineage with Apache Airflow using OpenLineage

Julien Le Dem and Willy Lulciuc, Datakin | March 2022

datakin

The principal sponsors of
OpenLineage & Marquez

ASTRONOMER

The company behind
Apache Airflow

ASTRONOMER

Agenda

The need for lineage metadata

OpenLineage and Marquez

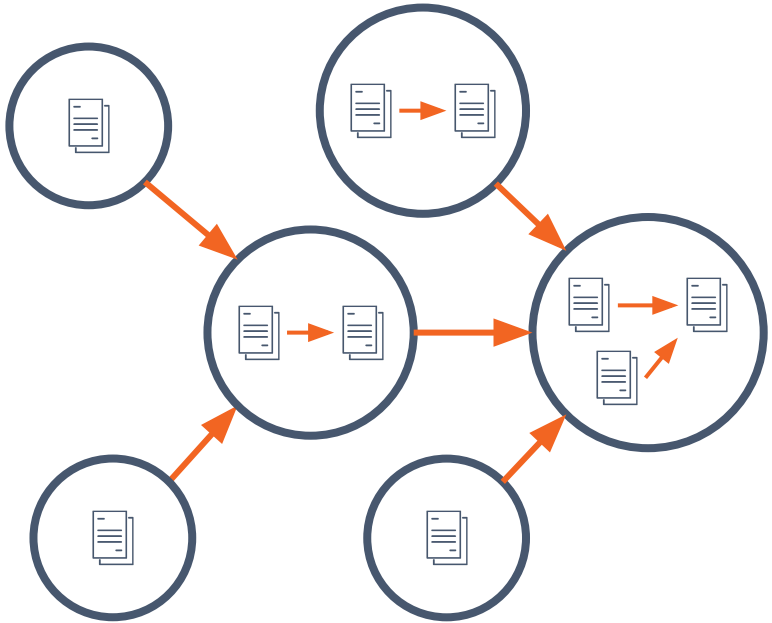
- OpenLineage, an open standard for lineage collection
- Marquez, its reference implementation

Airflow observability with OpenLineage

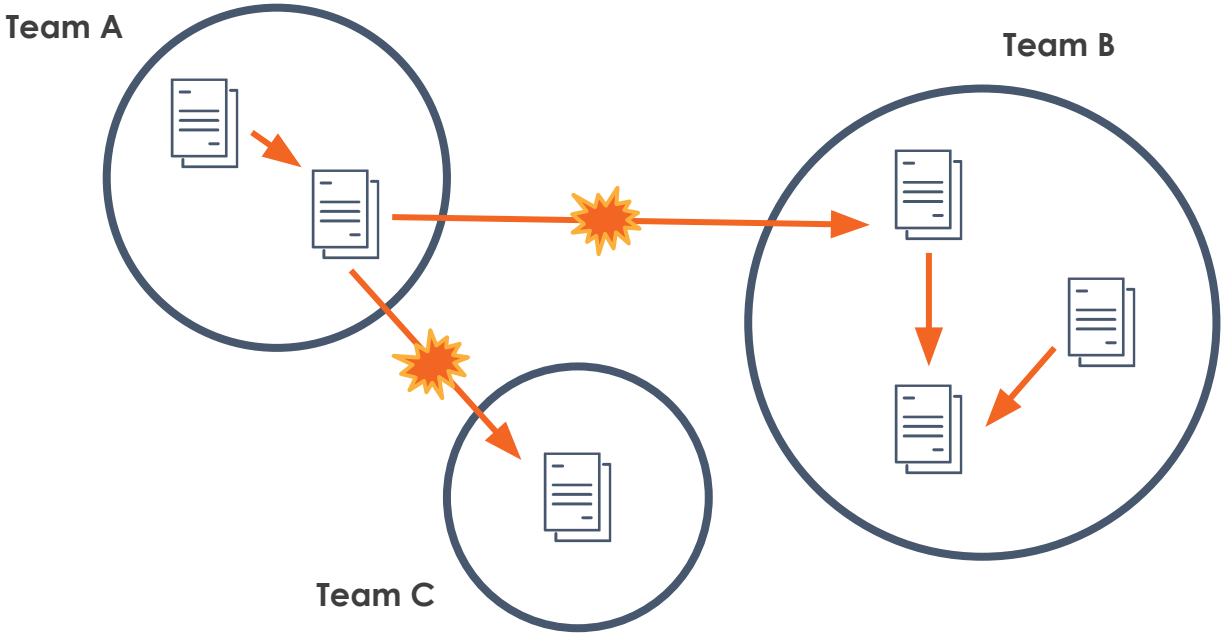
The key = data lineage

Data lineage contains what we need to know to solve our most complicated problems.

- Producers & consumers of each dataset
- Inputs and outputs of each job



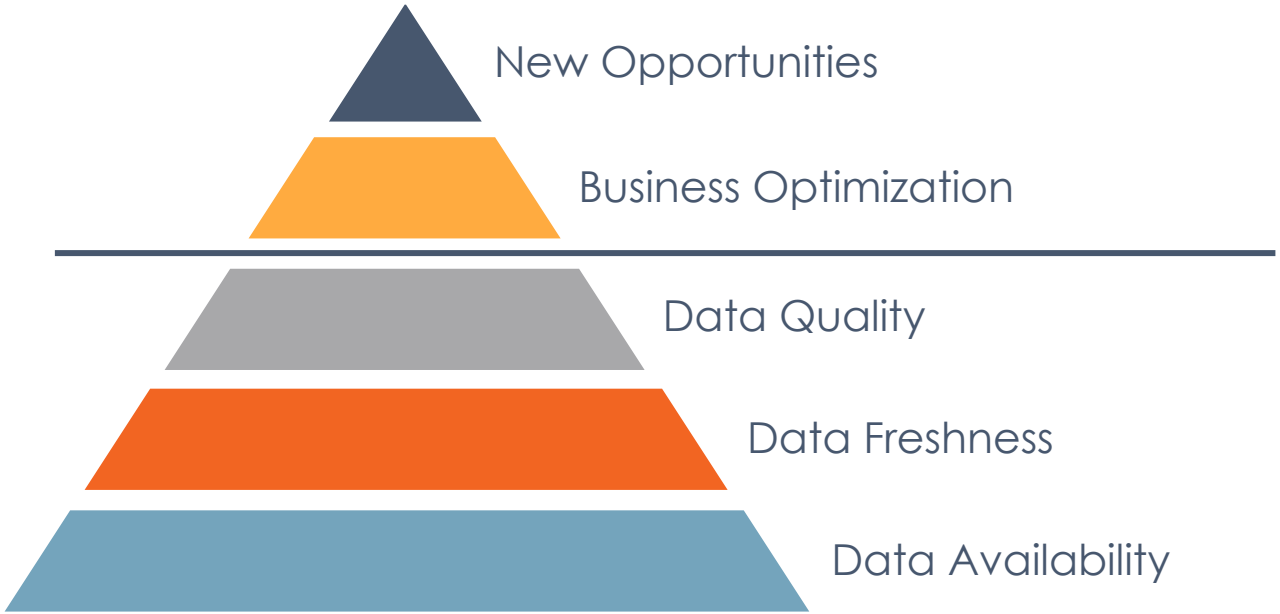
Building a healthy data ecosystem



Limited metadata = limited context

- What is the data source?
- What is the schema?
- Who is the owner?
- How often is it updated?
- Where does it come from?
- Who is using it?
- What has changed?

~~Maslow's~~ Data hierarchy of needs



OpenLineage

Mission

To define an **open standard** for the collection of lineage metadata from pipelines **as they are running**.

OpenLineage contributors

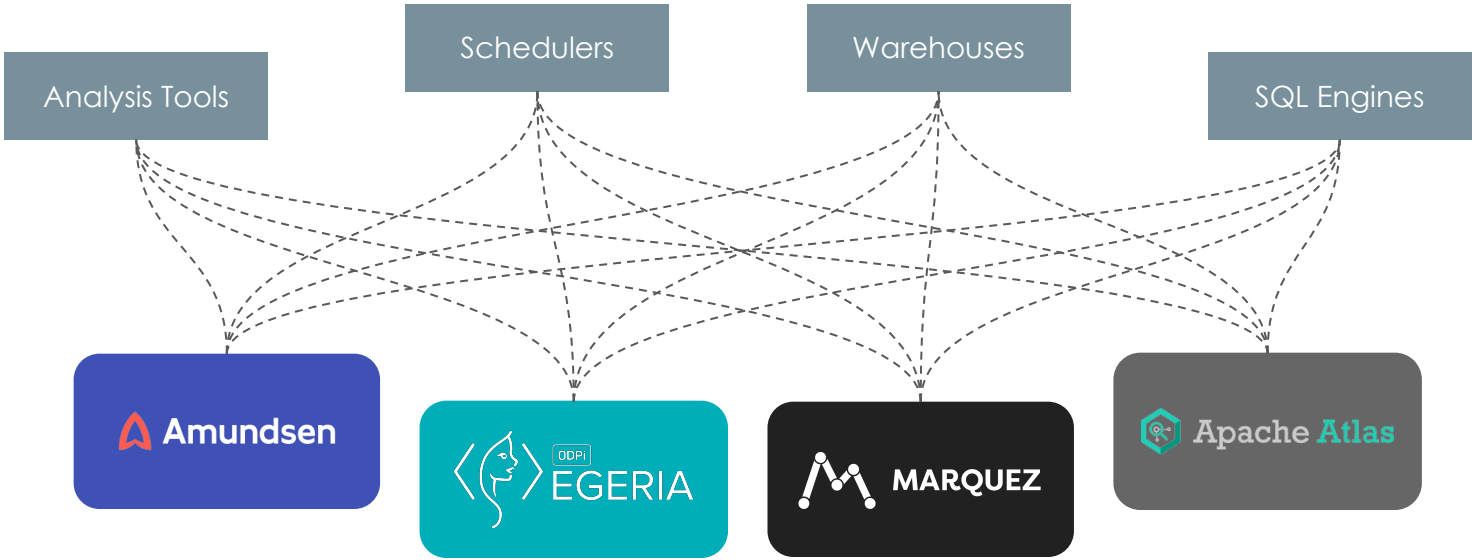
The snowball effect

The best time to collect metadata

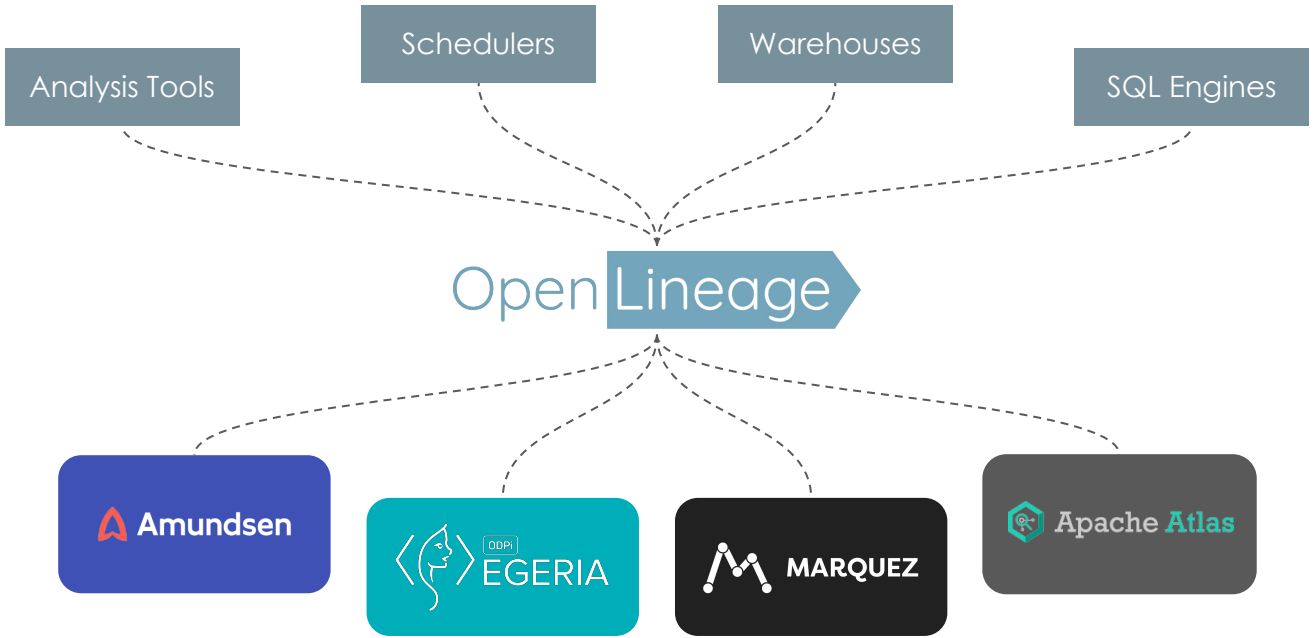
You can try to infer the date and location of an image after the fact…

…or you can capture it when the image is originally created!

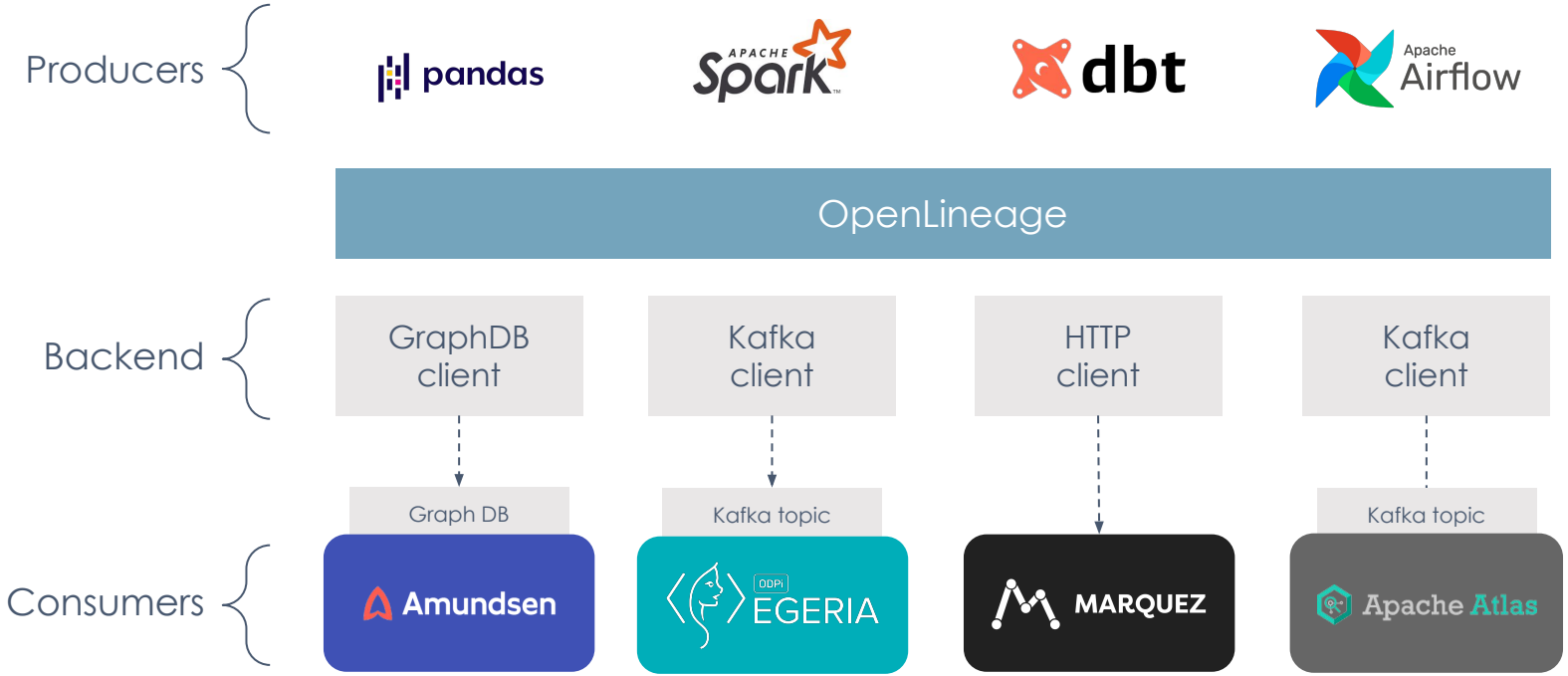
Before OpenLineage



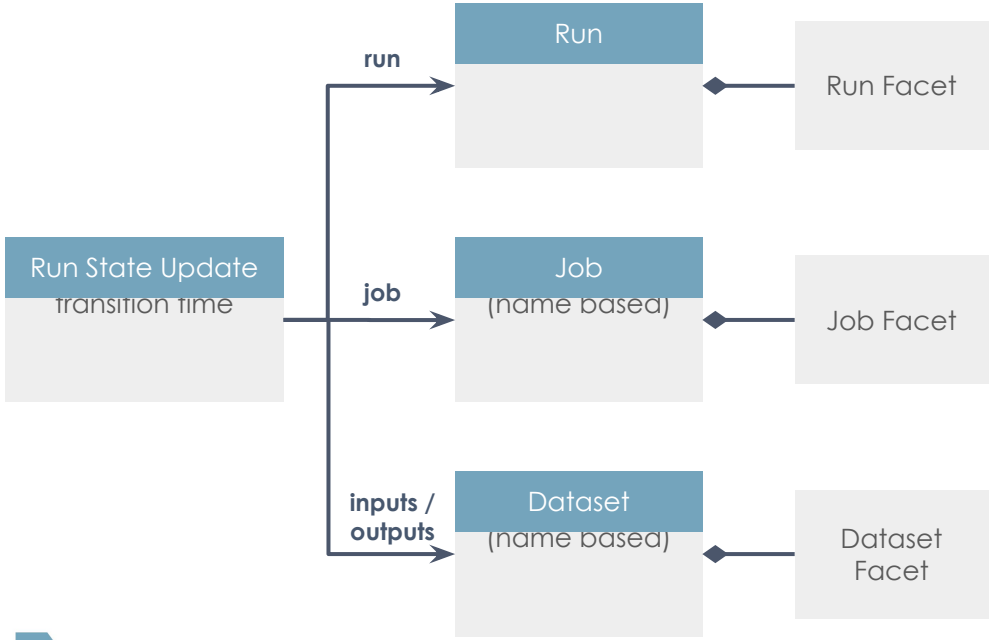
With OpenLineage



Where OpenLineage potentially fits



Data model



Built around core entities: Datasets, Jobs, and Runs

Defined as a JSONSchema spec

Consistent naming for:
Jobs (*scheduler.job.task*)
Datasets (*instance.schema.table)*

How OpenLineage events work

Lineage is reported as a series of asynchronous run events.

Each event passes a unique client-generated run ID to:

- identify the run
- correlate events

Typical event series:

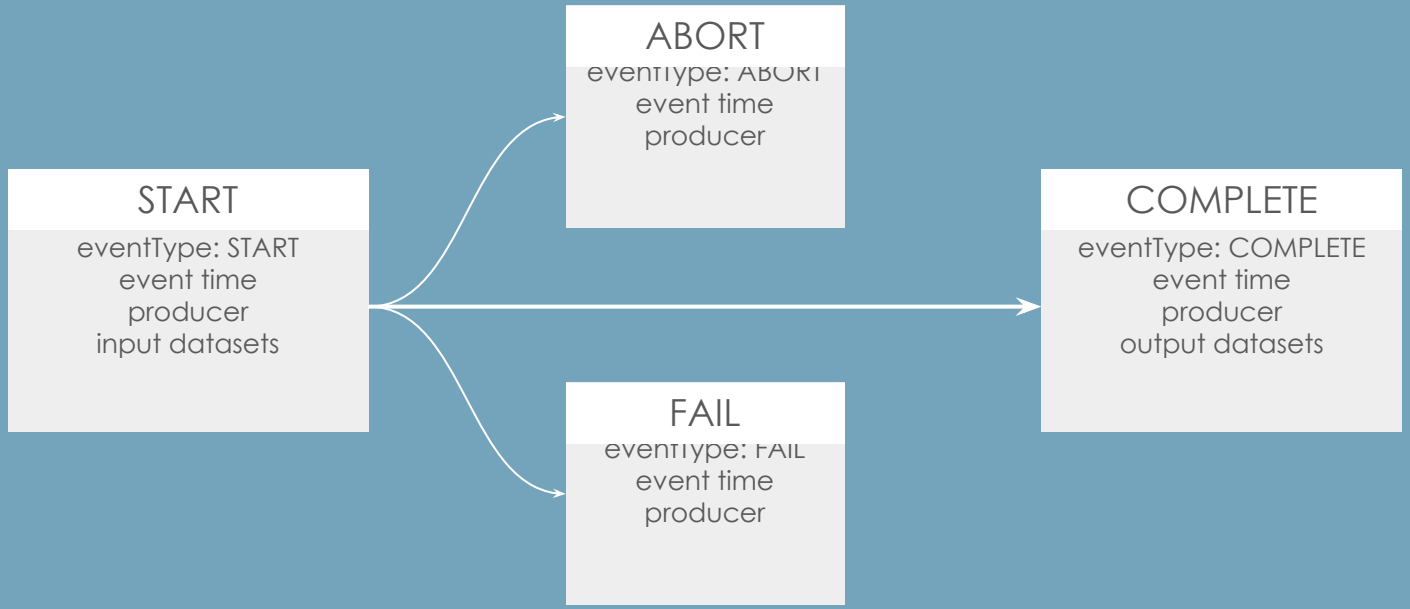
Send start event

- source code version
- run parameters

Send complete event

- input dataset
- output dataset version
- output schema

Lifecycle of a job run



Extending the model with Facets

Facets are atomic pieces of metadata attached to core entities.

Self-documenting

Facets can be given unique, memorable names

Familiar

Facets are defined using JSON schema objects

Flexible

Facets can be attached to any core entity: Job, Dataset & Run

Scalable

Prefixes on names are used to establish discrete namespaces

Facet examples

Dataset:

- Stats
- Schema
- Version

Job:

- Source code
- Dependencies
- Source control
- Query plan

Run:

- Scheduled time
- Batch ID
- Query profile
- Params

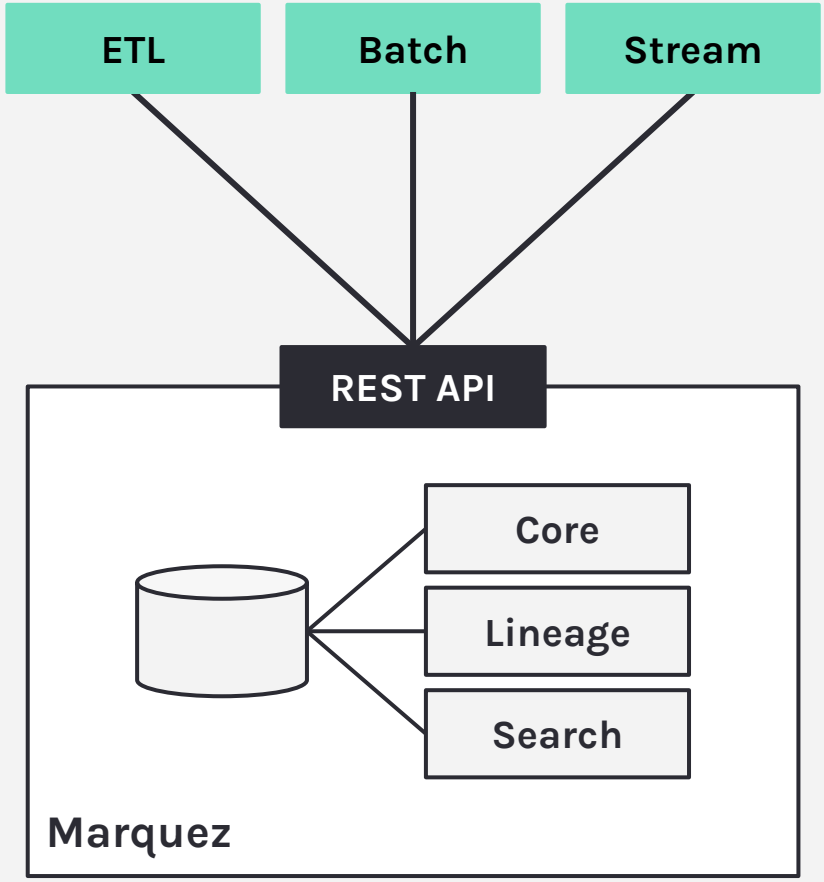
OMG the possibilities are endless

Dependency tracing
Root cause identification
Issue prioritization
Impact mapping
Precision backfills
Anomaly detection
Change management
Historical analysis
Compliance

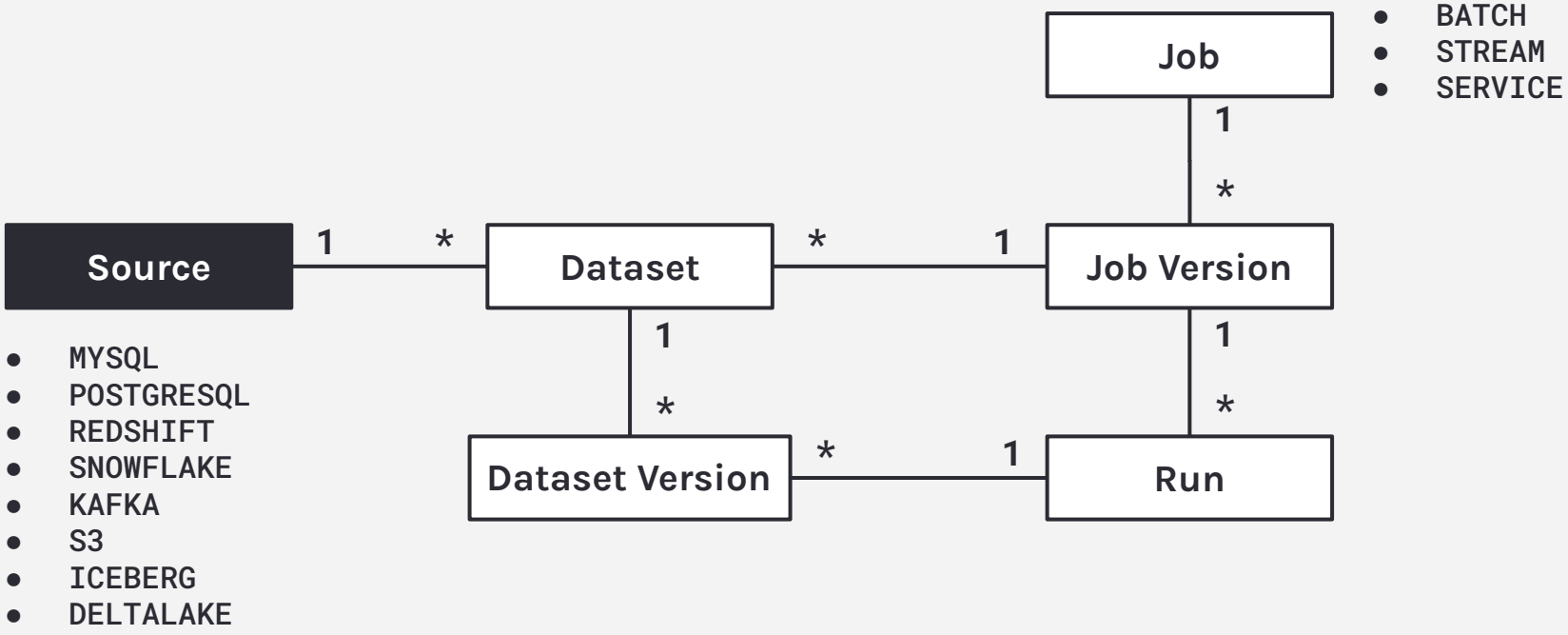
MARQUEZ

Metadata Service

- **Centralized metadata management**
  - Sources
  - Datasets
  - Jobs
- **Features**
  - Data governance
  - Data lineage
  - Data discovery + exploration



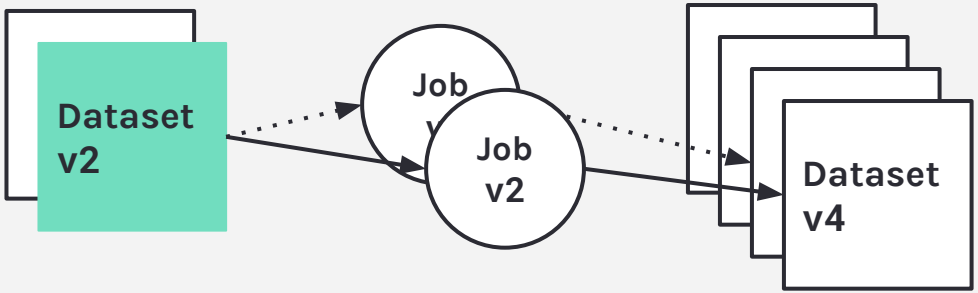
Marquez: Data model



MARQUEZ

Design benefits

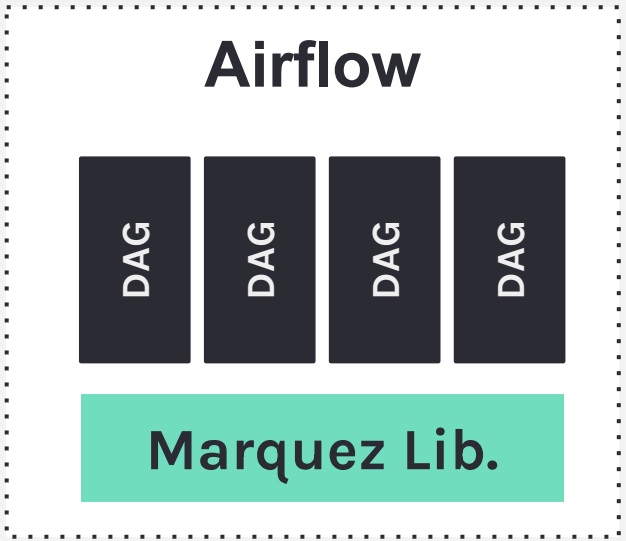
- Debugging
  - What **job version(s)** produced and consumed **dataset version X**?



- Backfilling
  - Full / incremental processing

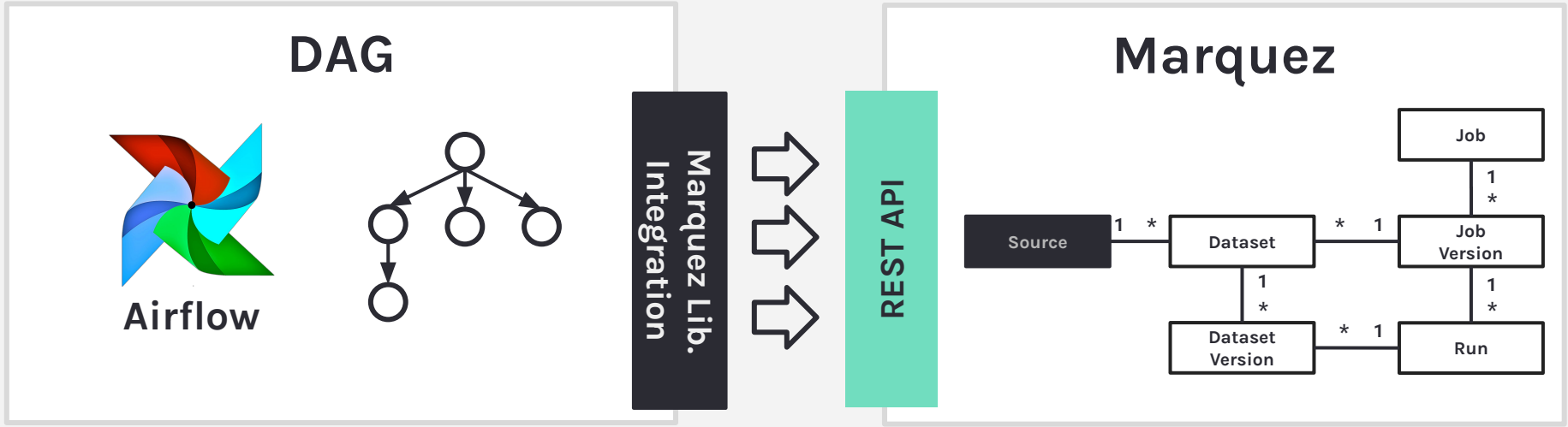
Airflow observability with OpenLineage

Airflow support for Marquez



- **Metadata**
  - Task lifecycle
  - Task parameters
  - Task runs linked to **versioned** code
  - Task inputs / outputs
- **Lineage**
  - Track inter-DAG dependencies
- **Built-in**
  - SQL parser
  - Link to code builder (**GitHub**)
  - Metadata extractors

Capturing task-level metadata in a nutshell



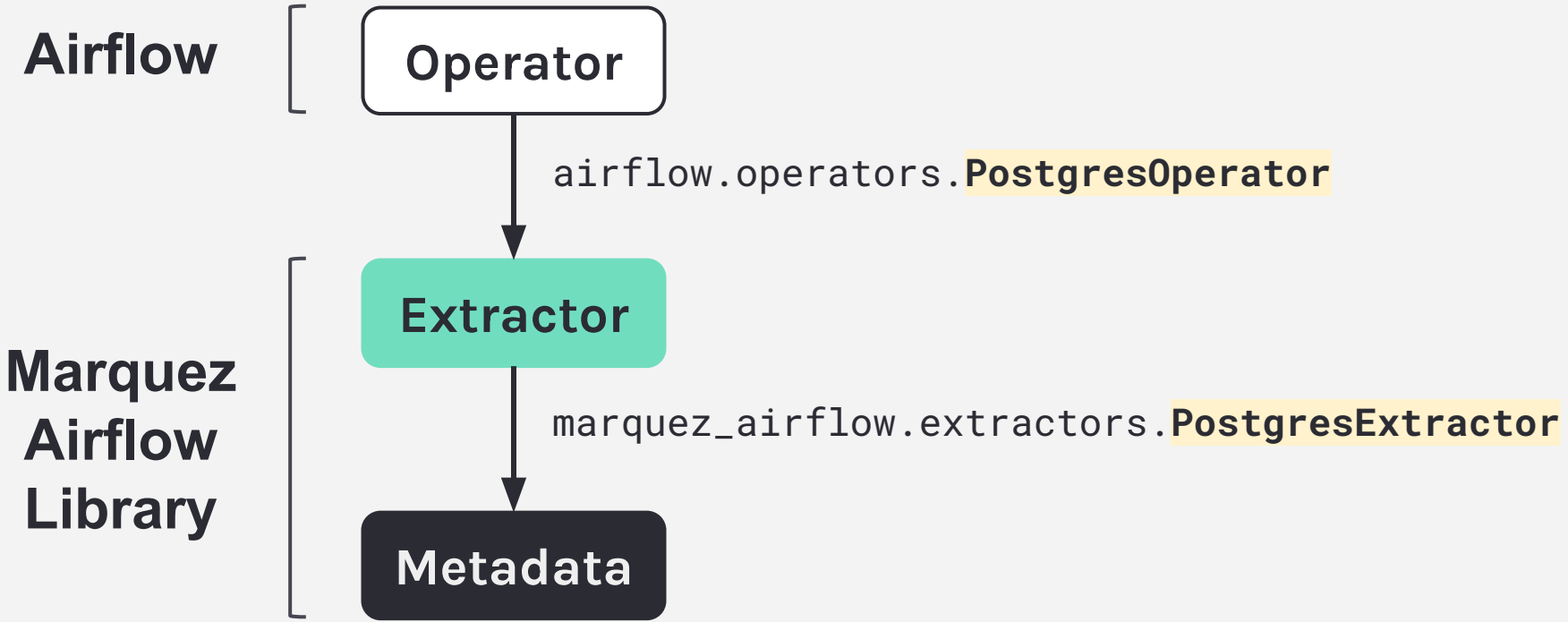
Marquez Airflow Lib.

- Open source! 🥇
- Enables **global** task-level metadata collection
- Extends Airflow's **DAG** class

room_bookings_7_days_dag.py

```
from marquez_airflow import DAG
from airflow.operators.postgres_operator import PostgresOperator

...
```

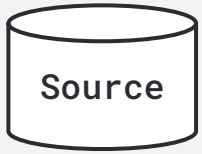

Airflow
Operator
airflow.operators.PostgresOperator
Marquez
Airflow
Library
Extractor
marquez_airflow.extractors.PostgresExtractor
Metadata

Operator Metadata

new_room_booking_dag.py

```
t1=PostgresOperator(
 task_id='new_room_booking',
 postgres_conn_id='analyticsdb',
 sql='''
   INSERT INTO room_bookings VALUES(%s, %s, %s)
 ''',
 parameters=... # room booking
)
```

01

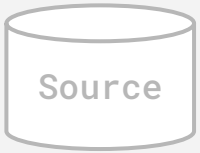


Operator Metadata

new_room_booking_dag.py

```
t1=PostgresOperator(
 task_id='new_room_booking',
 postgres_conn_id='analyticsdb',
 sql='''
   INSERT INTO room_bookings VALUES(%s, %s, %s)
 ''',
 parameters=... # room booking
)
```

01



02

Operator Metadata

new_room_booking_dag.py

```
t1=PostgresOperator(
 task_id='new_room_booking',
 postgres_conn_id='analyticsdb',
 sql='''
   INSERT INTO room_bookings VALUES(%s, %s, %s)
 ''',
 parameters=... # room booking
)
```

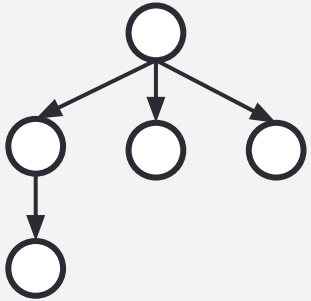
01 Source

02 Dataset

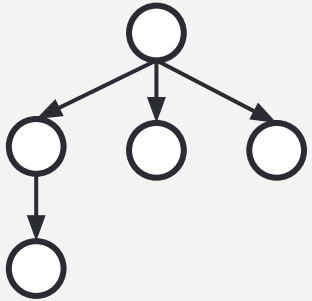
03 Job

Managing inter-DAG dependencies

`new_room_bookings_dag.py`

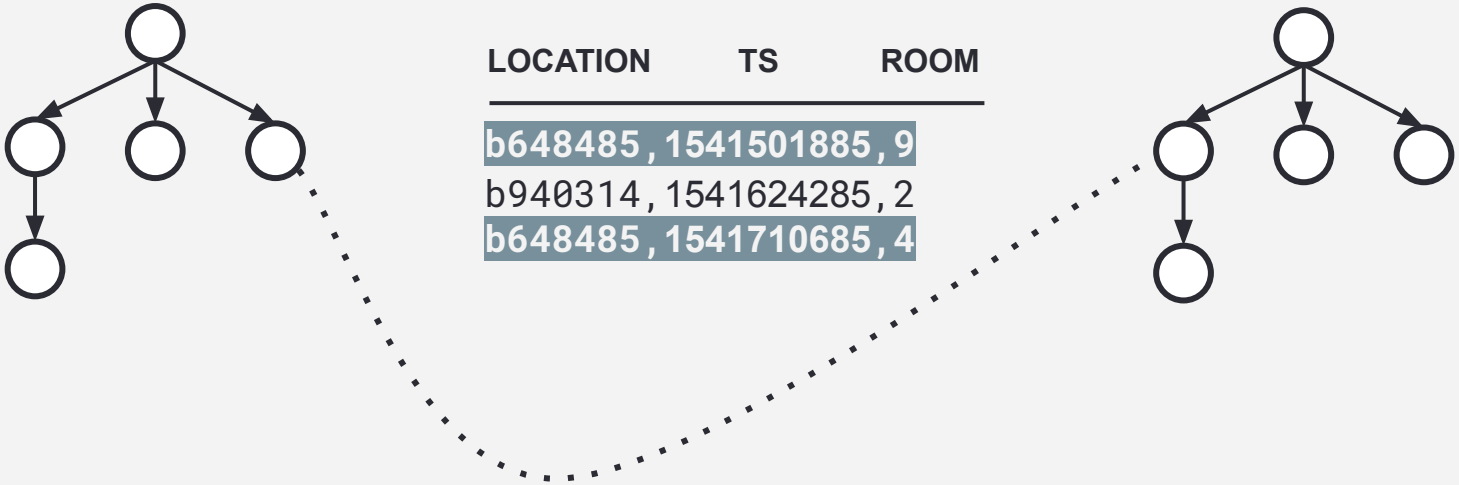


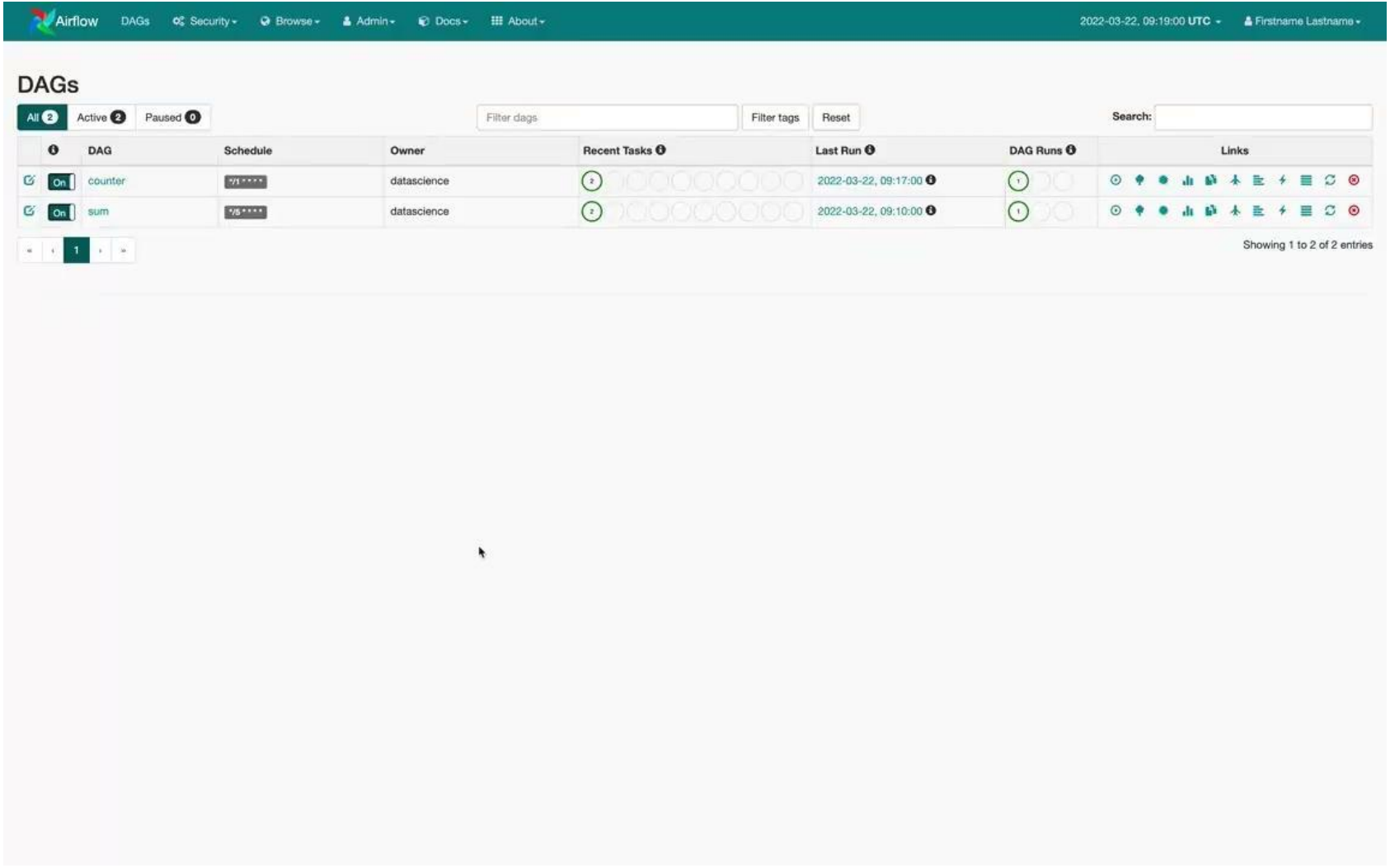
`top_room_bookings_dag.py`



Managing inter-DAG dependencies

`new_room_bookings_dag.py` `public.room_bookings` `top_room_bookings_dag.py`




Airflow
DAGs
Security
Browse
Admin
Docs
About
2022-03-22, 09:19:00 UTC
Firstname Lastname
DAGs
All 2
Active 2
Paused 0
Filter dags
Filter tags
Reset
Search:
DAG
Schedule
Owner
Recent Tasks
Last Run
DAG Runs
Links
On
counter
*/1 * * * *
datascience
2022-03-22, 09:17:00
On
sum
*/5 * * * *
datascience
2022-03-22, 09:10:00
Showing 1 to 2 of 2 entries

Join the conversation

OpenLineage

github.com/openlineage

openlineage.slack.com

@openlineage

groups.google.com/g/openlineage

MARQUEZ

github.com/marquezproject

marquezproject.slack.com

@marquezproject

Thanks :)