

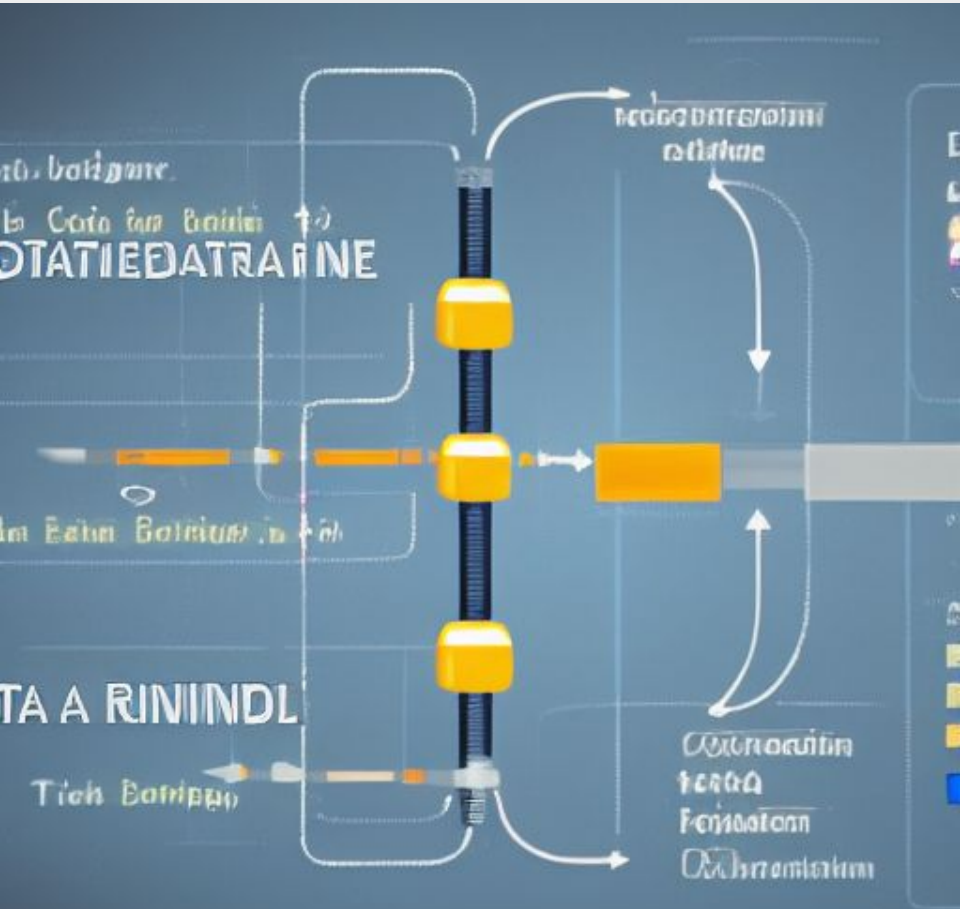
Open  Lineage

An open standard for data lineage

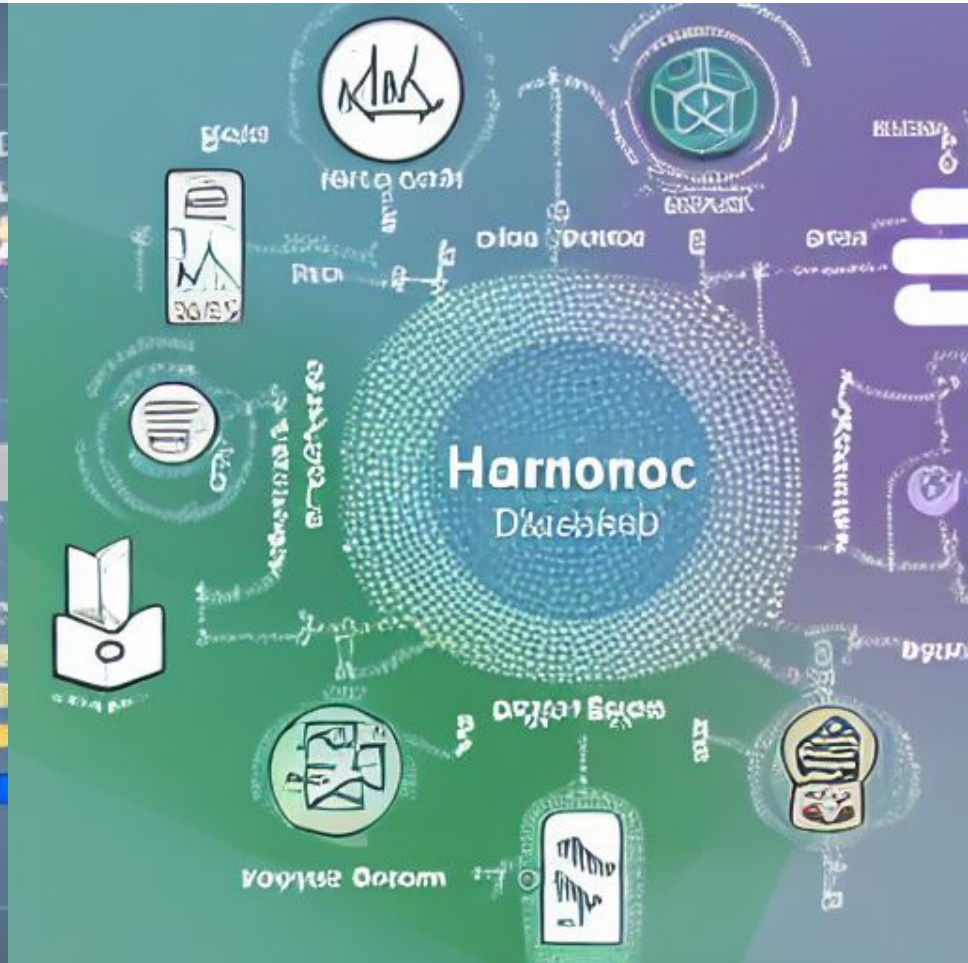
Ross Turk (ross.turk@astronomer.io)

Why now?

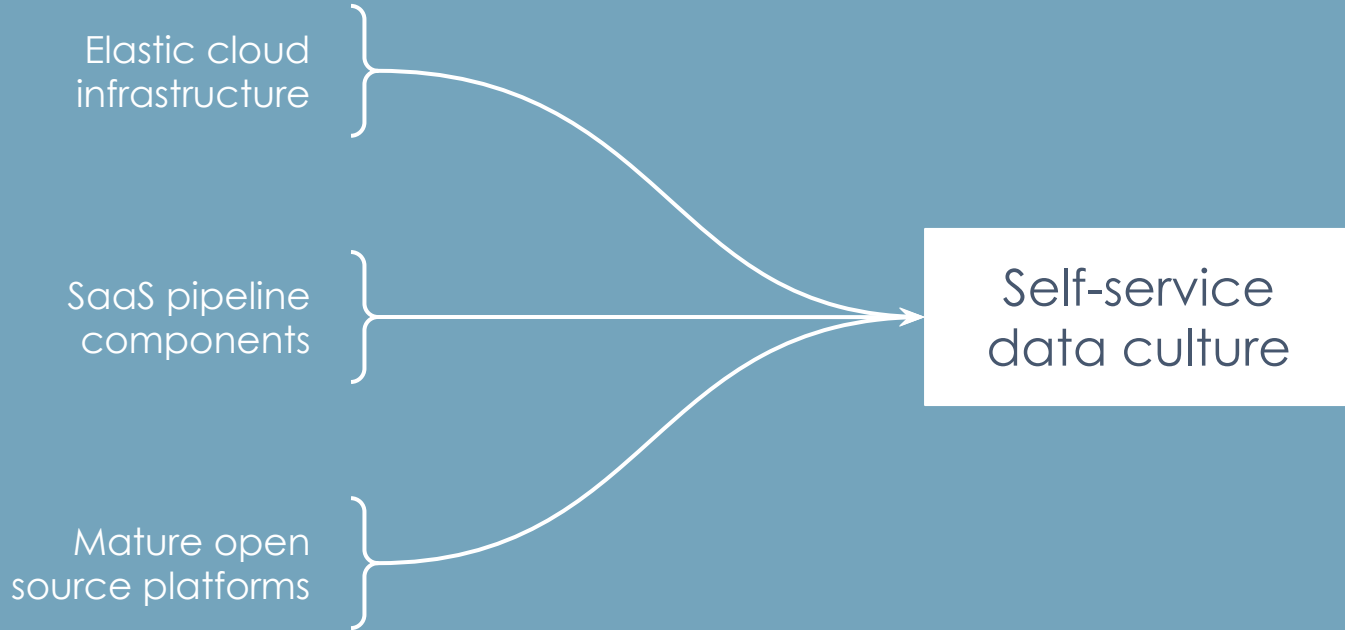
a data pipeline



a harmonious data ecosystem



What else changed?



The defining dilemma

What kind of pipeline
should I build?

How will I go about
building it?

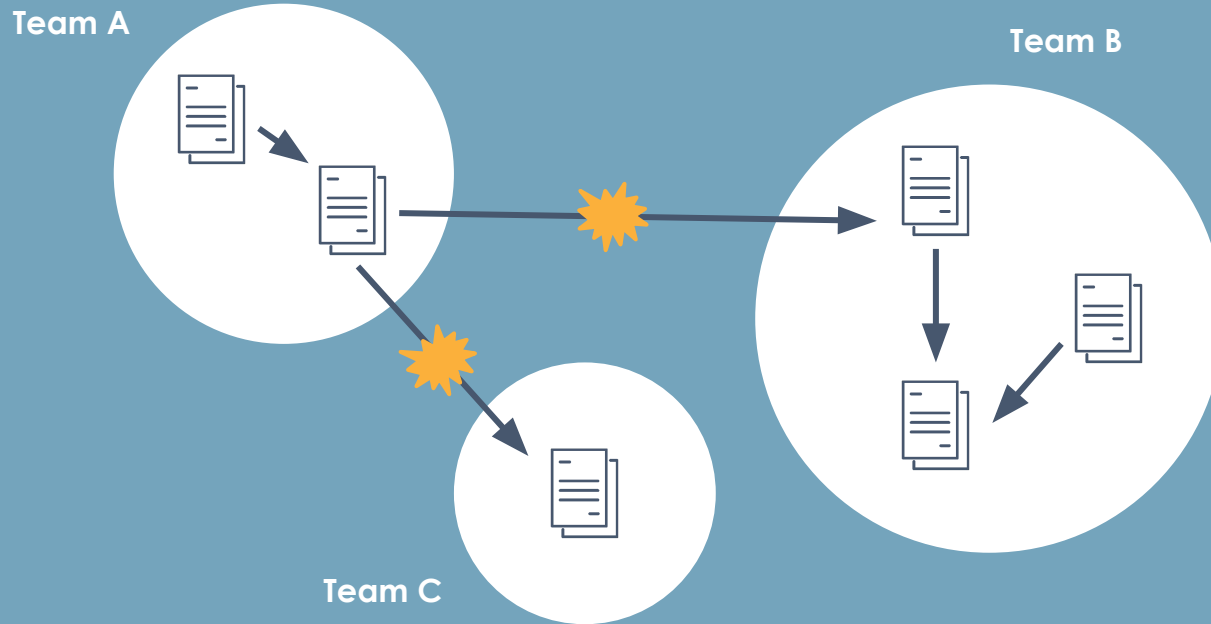


How many pipelines are
currently running?

How can we learn about
all of them?

How can we know what
goes on inside them?

Building a healthy data ecosystem



Ecosystems form around shared understanding



DATA

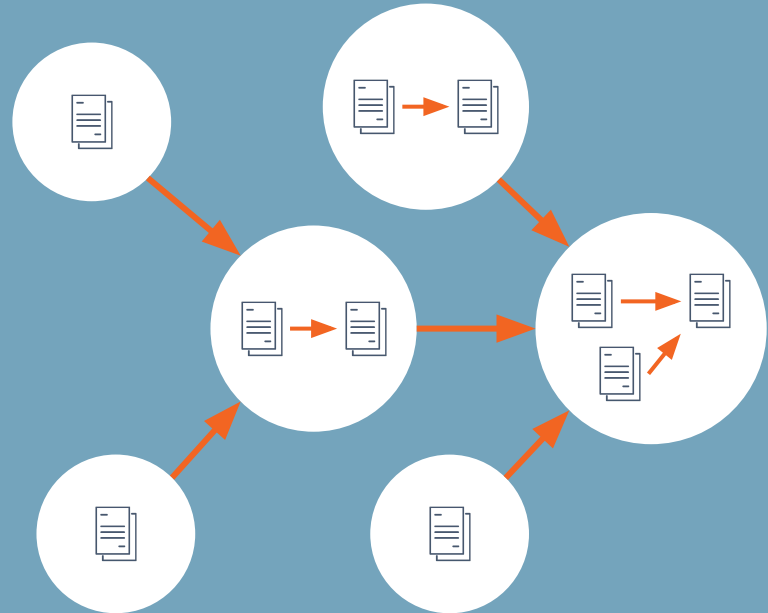
What is the data source?
What is the schema?
Who is the owner?
How often is it updated?
Where does it come from?
Who is using it?
What has changed?

What is data
lineage?

What is 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



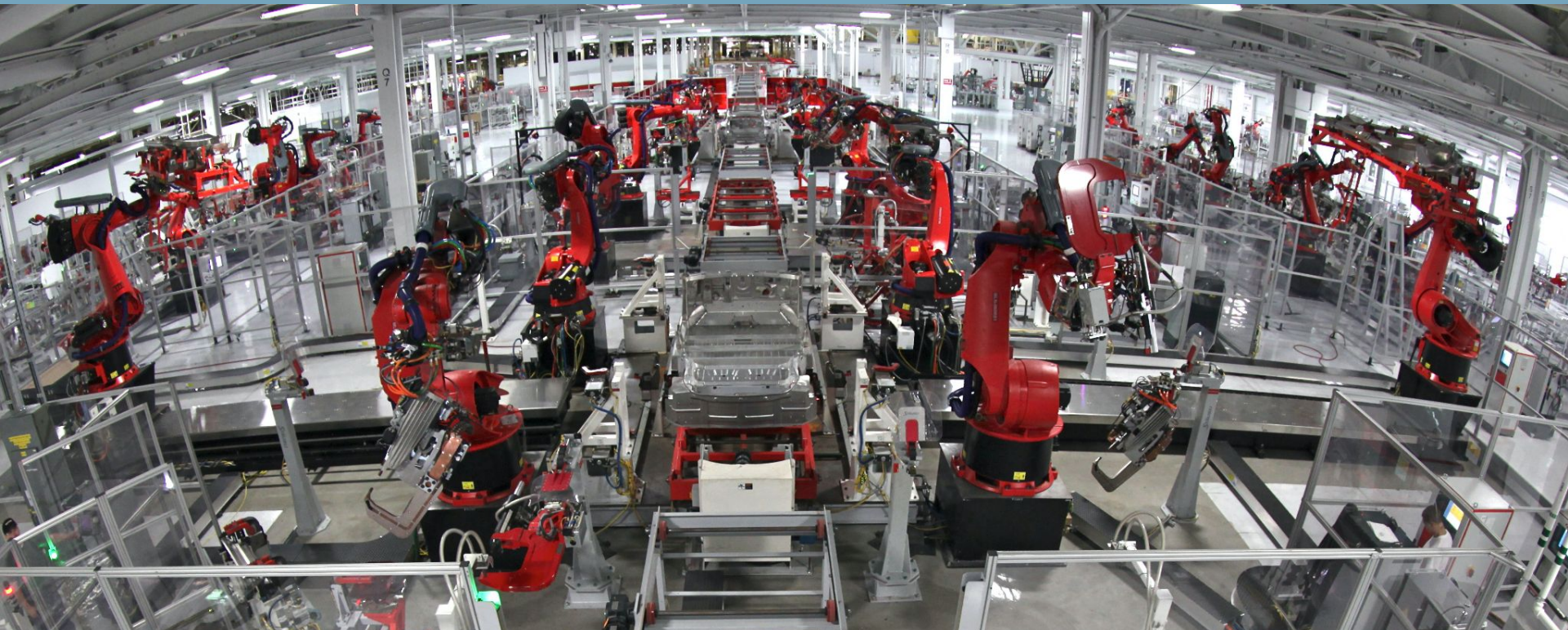
That's it 

Just know everything, right?

Verifying compliance



Optimizing data operations



Establishing context & language



OMG the possibilities are endless

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



Ok, sounds great.
So how?

The best time to collect metadata

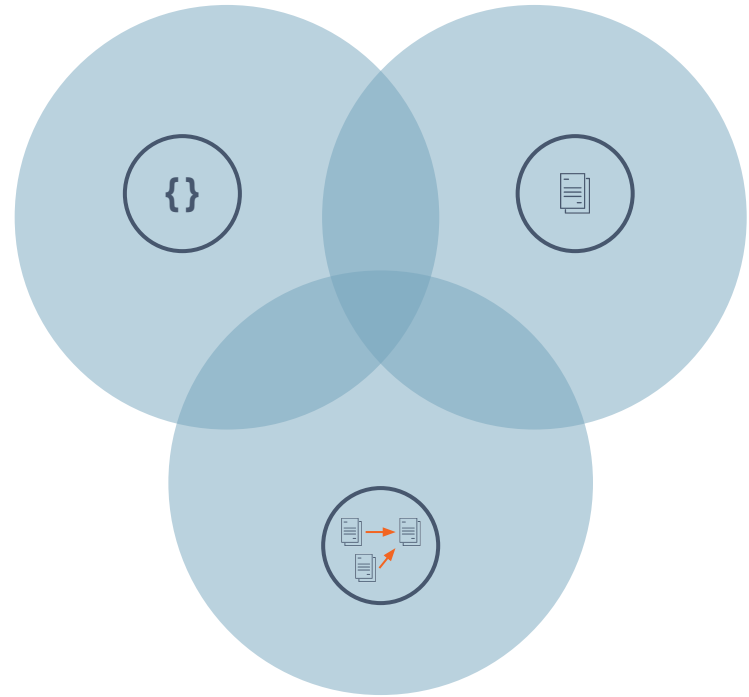


You can try to infer the date and location of an image after the fact...

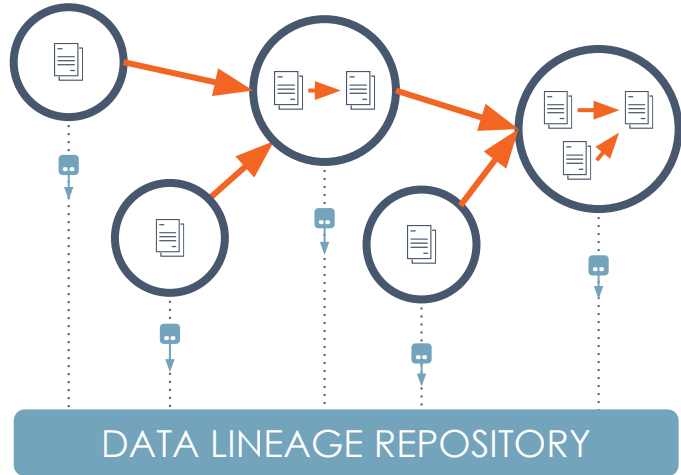


...or you can capture it when the image is originally created!

Comparing approaches

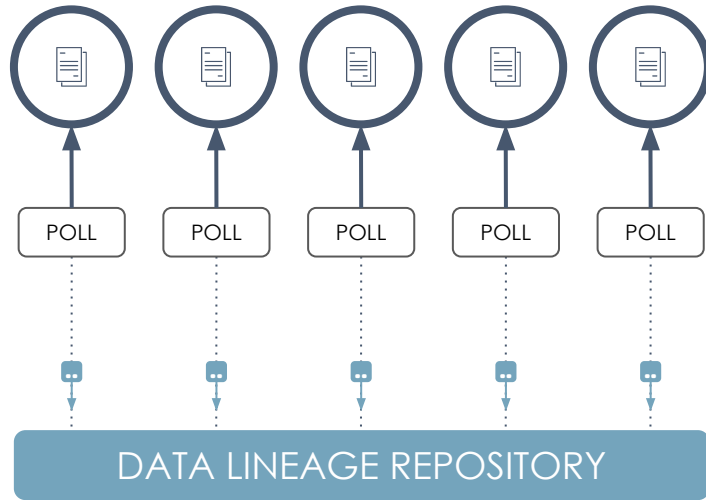


Observe the pipeline



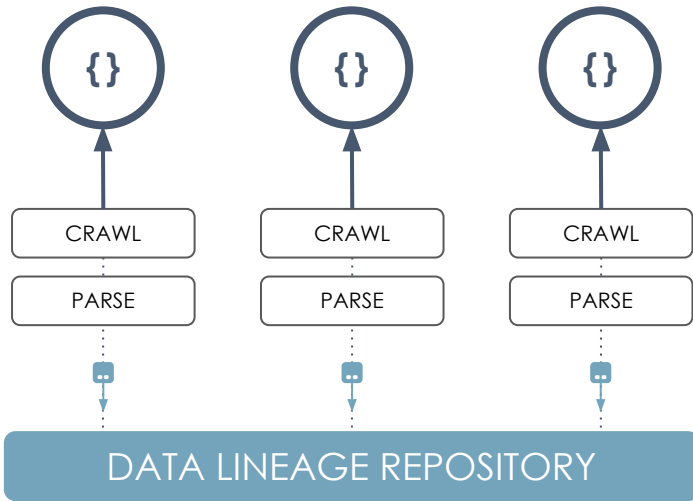
- Integrate with data orchestration systems
- As jobs run, observe the way they affect data
- Report to a lineage metadata repository

Process query / activity logs



- Integrate with data stores and warehouses
- Regularly process query logs to trace lineage
- Report to a lineage metadata repository

Analyze source code



- Integrate with source code repositories
- Look for queries and parse them for lineage
- Report to a lineage metadata repository

It's a patchwork



Non-malicious (yet common) lineage lies

Fully-automated	
Real-time	
End-to-end	360° visibility
Easy	AI/ML enhanced

OpenLineage

Mission

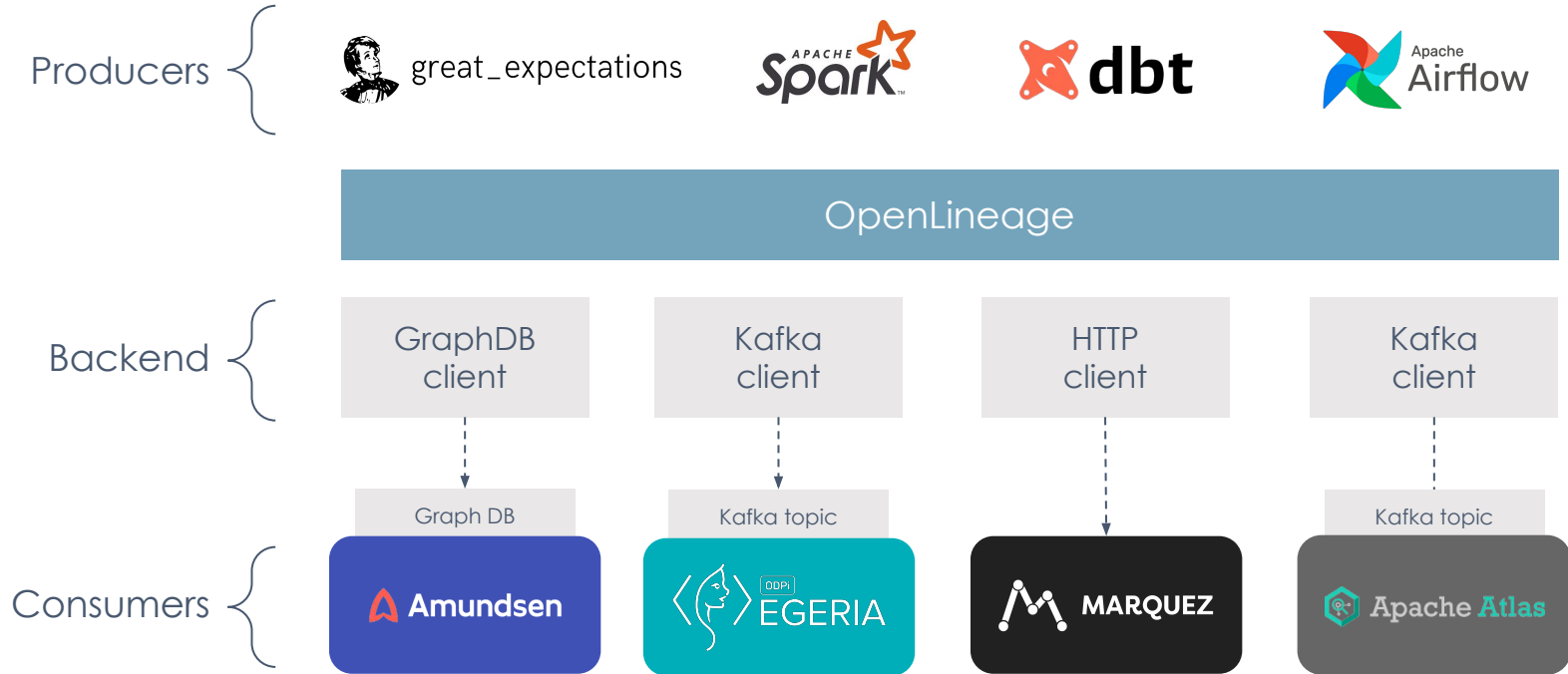
To define an **open standard** for the collection of lineage metadata from pipelines **as they are running**.



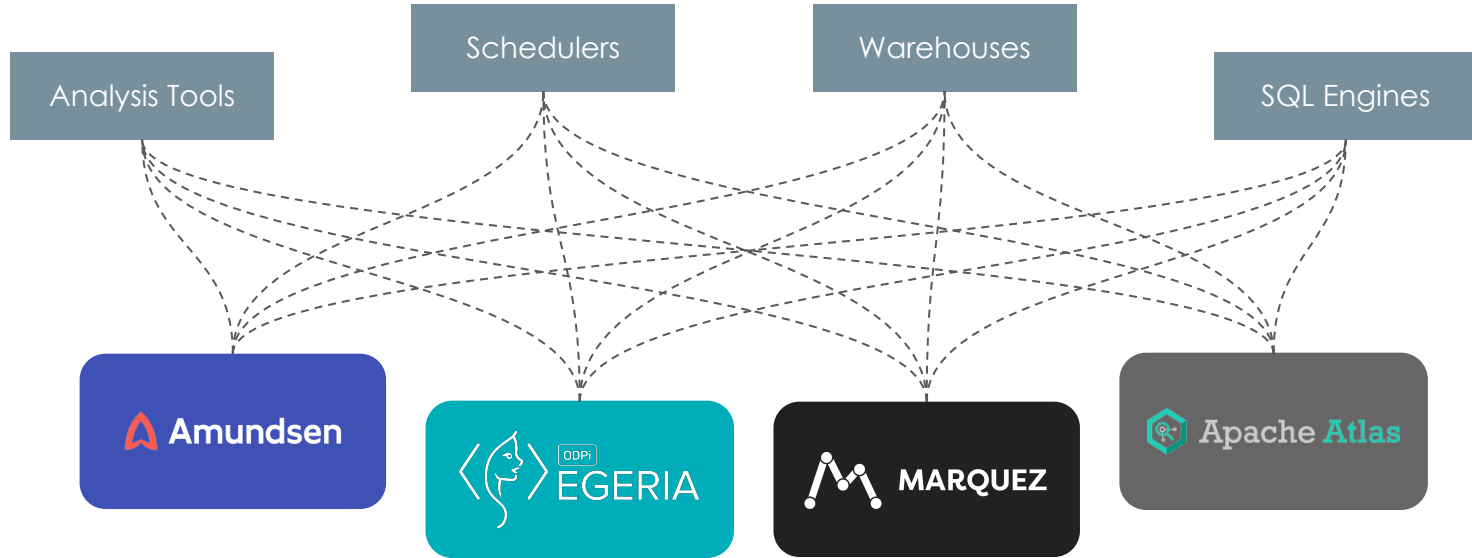
Stone Soup, a fable about community



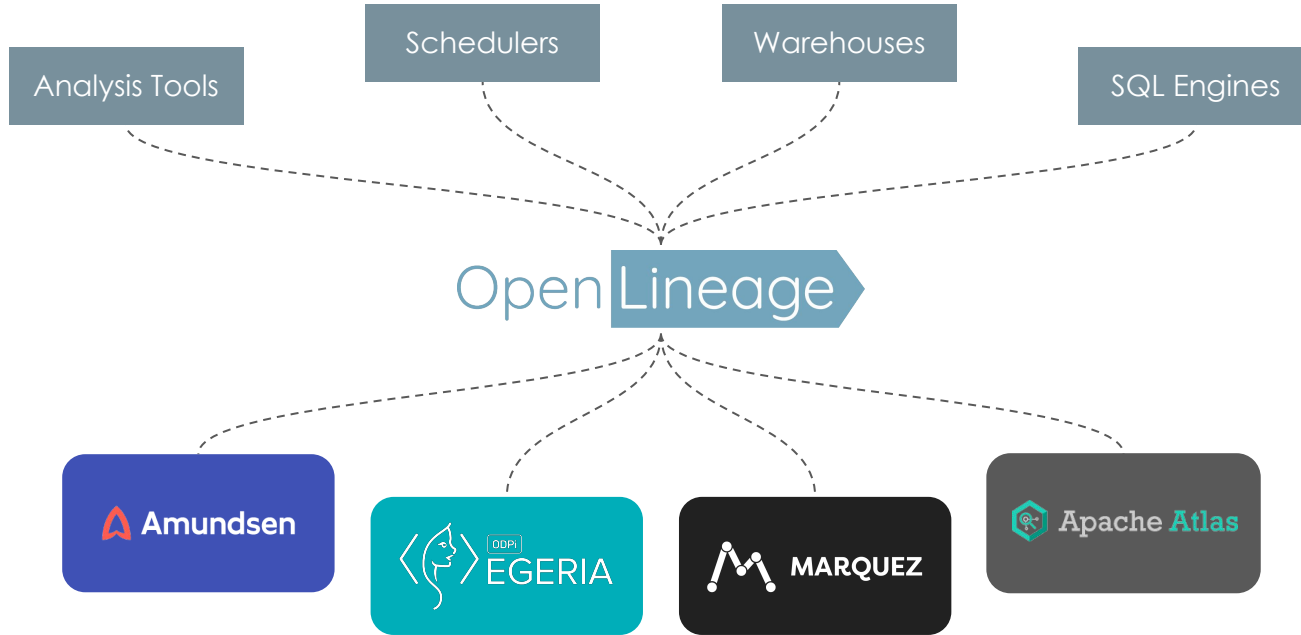
Where OpenLineage potentially fits



Before OpenLineage

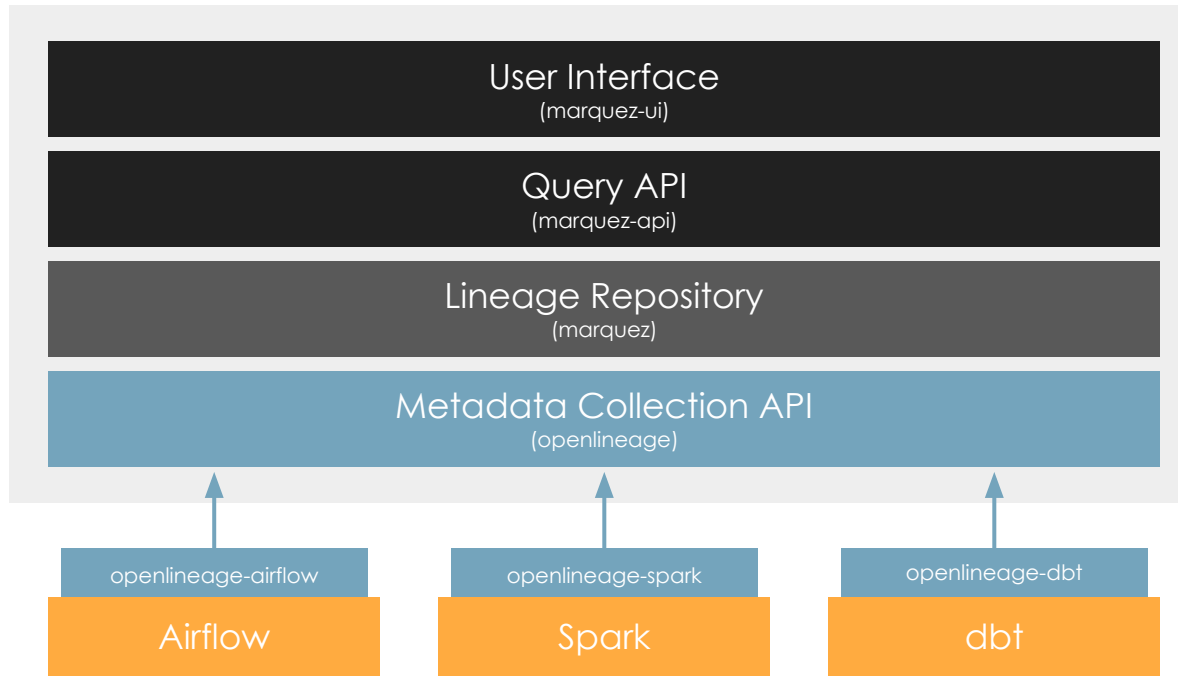


With OpenLineage

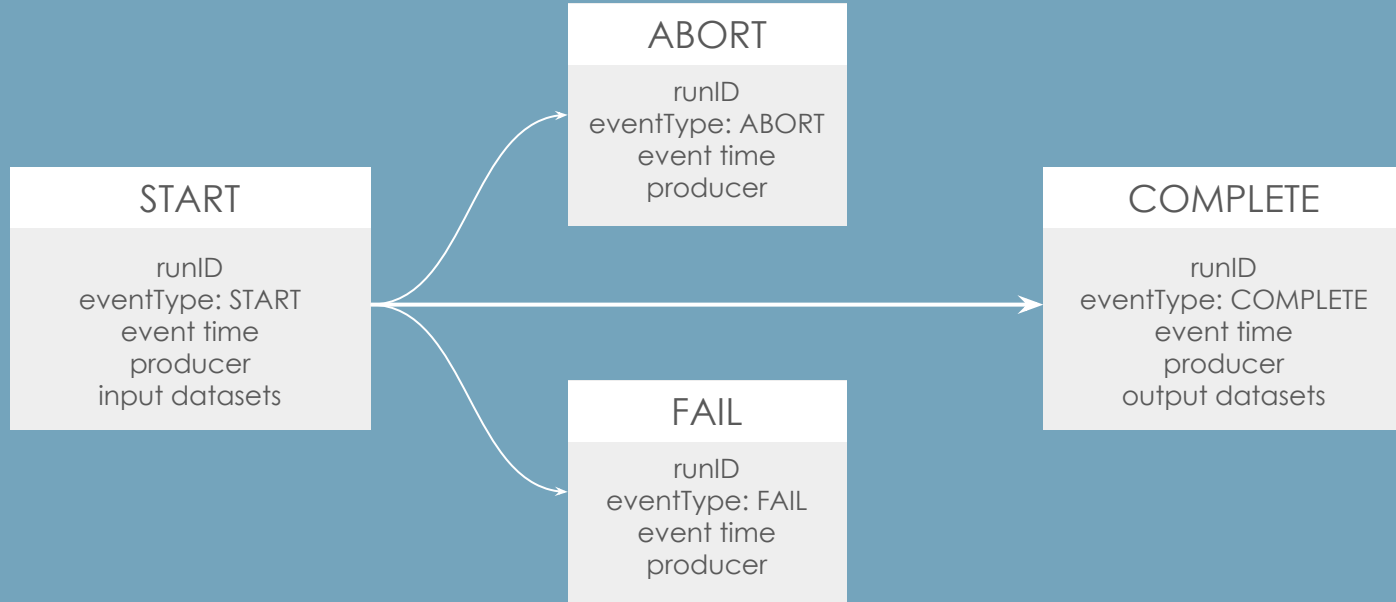


How does
OpenLineage
work?

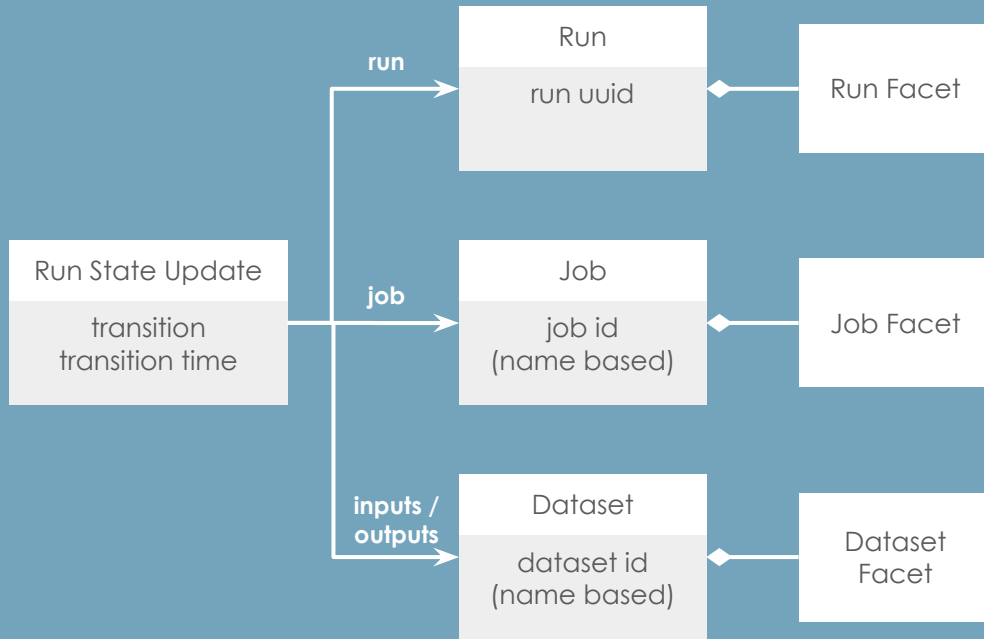
The OpenLineage Stack



Lifecycle of a job run



Data model



Built around core entities:
Datasets, Jobs, and Runs

Defined as a JSON
Schema spec

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

Facet examples

Dataset:

- Stats
- Schema
- Version

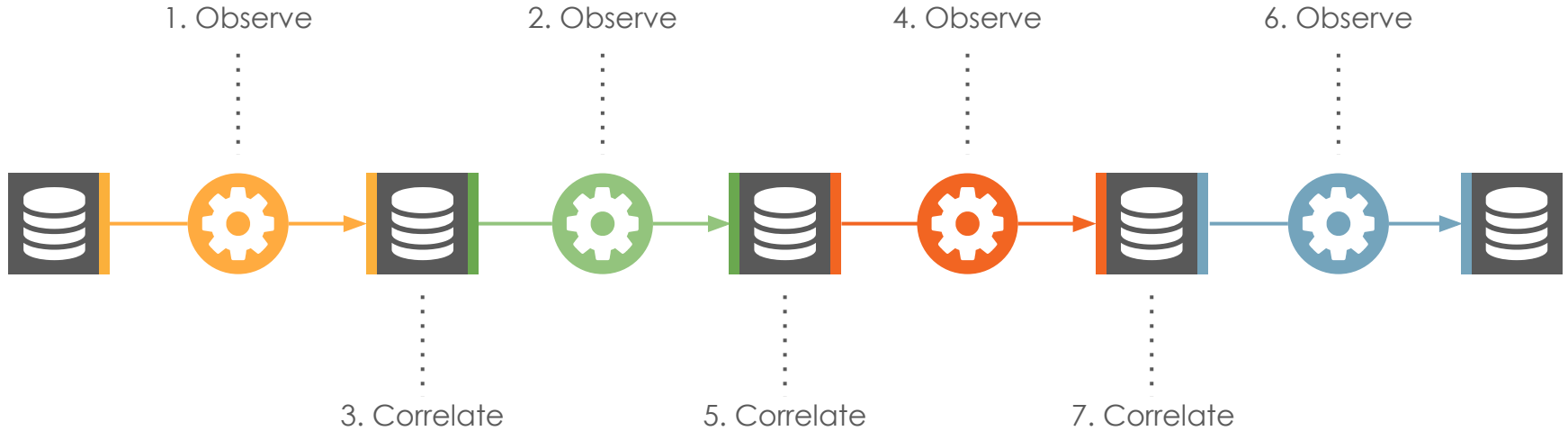
Job:

- Source code
- Dependencies
- Source control
- Query plan

Run:

- Scheduled time
- Batch ID
- Query profile
- Params

Lineage is built on correlations



Dataset names are used to stitch together observations of job runs into a lineage graph.

Naming conventions

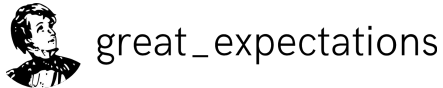
	Formulae	Examples
Datasets	host + database + table bucket + path host + port + path project + dataset + table	postgres://db.foo.com/metrics.salesorders s3://sales-metrics/orders.csv hdfs://stg.foo.com:salesorders.csv bigquery:metrics.sales.orders
Jobs	namespace + name namespace + project + name	staging.load_orders_from_csv prod.orders_etl.count_orders
Runs	Client-provided UUID	1c0386aa-0979-41e3-9861-3a330623effa

The snowball effect



OpenLineage Integrations

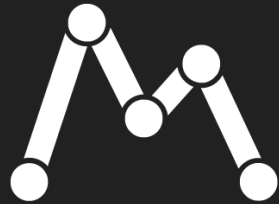
Metadata producers



Metadata consumers



Marquez: open source metadata



MARQUEZ

Checking out the Marquez project

```
rturk@maxwell:~/projects/workshops/airflow/e1-marquez/marquez

~/p/workshops/a/e1-marquez ) main git clone git@github.com:Marquez
Project/marquez.git
Cloning into 'marquez'...
remote: Enumerating objects: 21854, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 21854 (delta 2), reused 6 (delta 0), pack-reused 21844
Receiving objects: 100% (21854/21854), 20.87 MiB | 26.12 MiB/s, done.
Resolving deltas: 100% (12855/12855), done.

~/p/workshops/a/e1-marquez ) main cd marquez ✓

~/p/workshops/a/e1/marquez ) main | ✓
```

Starting up Marquez

```
./docker/up.sh -s
marquez_utils
marquez_db-init
41bd218c1b8bd488b56312780142ed9c099eef49c1f6c8b6a1ca7864470d7b12
marquez-volume-helper
Creating marquez-db ... done
Creating marquez-api ... done
Creating marquez-web ... done
Creating seed-marquez-with-metadata ... done
Attaching to marquez-db, marquez-api, marquez-web, seed-marquez-with-metadata
marquez-db | The files belonging to this database system will be owned by u
ser "postgres".
marquez-db | This user must also own the server process.
marquez-db |
marquez-db | The database cluster will be initialized with locale "en_US.ut
f8".
marquez-api | wait-for-it.sh: waiting 15 seconds for db:5432
marquez-db | The default database encoding has accordingly been set to "UTF
```


About the Marquez start script

```
docker/up.sh --seed
```

Load the database with seed data

After starting Marquez, simulate a series of lineage events for a fictional food delivery service pipeline. Good for exploring the Marquez UI + the OpenLineage data model and API.

```
docker/up.sh --detach
```

Run in detached mode

This will cause everything to run in the background (cool!) but also it won't show logs (aww!)

```
docker/up.sh --build
```

Build from source

Build everything, instead of grabbing the latest images from Docker Hub. For development.

localhost

MARQUEZ

Search Jobs and Datasets

ns workshop API Docs

```
graph LR; A((order_analysis.import_orders)) --> B[(workshop.p...lic.orders)]; B --> C((order_analysis.r_products)); B --> D((order_analysis.ize_months)); C --> E[(workshop.p...p_products)]; D --> F[(workshop.p...ly_summary)];
```

LATEST RUN RUN HISTORY LOCATION X

● **order_analysis.import_orders**

```
DROP TABLE IF EXISTS orders;

CREATE TABLE orders (
  order_id SERIAL PRIMARY KEY,
  order_date TIMESTAMP NOT NULL,
  product CHAR(50) NOT NULL,
  price INT NOT NULL,
  quantity INT NOT NULL
);

INSERT INTO
  orders (
    order_date,
```

Starting a job run

```
rturk@mastro:~/projects/workshops/airflow/e2-lineage-api
~/p/workshops/a/e2-lineage-api main bat -p json/startjob.json ✓
{
  "eventType": "START",
  "eventTime": "2020-12-28T19:52:00.001+10:00",
  "run": {
    "runId": "d46e465b-d358-4d32-83d4-df660ff614dd"
  },
  "job": {
    "namespace": "my-namespace",
    "name": "my-job"
  },
  "inputs": [{
    "namespace": "my-namespace",
    "name": "my-input"
  }],
  "producer": "https://github.com/OpenLineage/OpenLineage/blob/v1-0-0/client"
}

~/p/workshops/a/e2-lineage-api main curl -X POST http://localhost:5000/api/v1/lineage \
-H 'Content-Type: application/json' \
-d @json/startjob.json ✓

~/p/workshops/a/e2-lineage-api main | ✓
```

Completing a job run

```
rturk@mastro:~/projects/workshops/airflow/e2-lineage-api  
~/p/workshops/a/e2-lineage-api main bat -p json/completejob.json ✓  
{  
  "eventType": "COMPLETE",  
  "eventTime": "2020-12-28T20:52:00.001+10:00",  
  "run": {  
    "runId": "d46e465b-d358-4d32-83d4-df660ff614dd"  
  },  
  "job": {  
    "namespace": "my-namespace",  
    "name": "my-job"  
  },  
  "outputs": [{  
    "namespace": "my-namespace",  
    "name": "my-output"  
  }],  
  "producer": "https://github.com/OpenLineage/OpenLineage/blob/v1-0-0/client"  
}  
~/p/workshops/a/e2-lineage-api main curl -X POST http://localhost:5000/api/v1/lineage \ ✓  
-H 'Content-Type: application/json' \  
-d @json/completejob.json  
~/p/workshops/a/e2-lineage-api main | ✓
```

Example:
viewing a
job run

The screenshot shows the Marquez web interface in a browser window. The top navigation bar includes the Marquez logo, a search bar for jobs and datasets, and a namespace dropdown set to 'my-namespace'. Below the navigation, a workflow diagram shows three components: 'my-input' (represented by a database icon), 'my-job' (represented by a gear icon), and 'my-output' (represented by a database icon), connected by arrows. The 'my-job' component is highlighted with a green circle. Below the diagram, there are tabs for 'LATEST RUN' and 'RUN HISTORY', with 'RUN HISTORY' selected. A 'LOCATION' button with a close icon is also present. The main content area displays a table of job runs for 'my-job'.

ID	State	Created At	Started At	Ended At	Duration
d46e465b-d358-4d32-83d4-df660ff614dd	COMPLETED	Dec 28, 2020 04:52am	Dec 28, 2020 04:52am	Dec 28, 2020 05:52am	0m 00s

```
#!/usr/bin/env python3
```

```
from openlineage.client.run import RunEvent, RunState, Run, Job, Dataset
from openlineage.client import OpenLineageClient
from datetime import datetime
from uuid import uuid4
```

```
# Initialize the OpenLineage client
client = OpenLineageClient.from_environment()
```

```
# Specify the producer of this lineage metadata
producer = "https://github.com/OpenLineage/workshops"
```

```
# Create some basic Dataset objects for our fictional pipeline
online_orders = Dataset(namespace="workshop", name="online_orders")
mail_orders = Dataset(namespace="workshop", name="mail_orders")
orders = Dataset(namespace="workshop", name="orders")
```

```
# Create a Run object with a unique ID
run = Run(str(uuid4()))
```

```
# Create a Job object
job = Job(namespace="workshop", name="process_orders")
```

```
# Emit a START run event
```

```
client.emit(
    RunEvent(
        RunState.START,
        datetime.now().isoformat(),
        run, job, producer
    )
)
```

```
#
```

```
# This is where our application would do the actual work :)
```

```
#
```

```
# Emit a COMPLETE run event
```

```
client.emit(
    RunEvent(
        RunState.COMPLETE,
        datetime.now().isoformat(),
        run, job, producer,
        inputs=[online_orders, mail_orders],
        outputs=[orders],
    )
)
```

Using the Python client

Thanks :)