# Data Platform Architecture Principles

Julien Le Dem

CTO and co-founder Datakin

@J_

# AGENDA

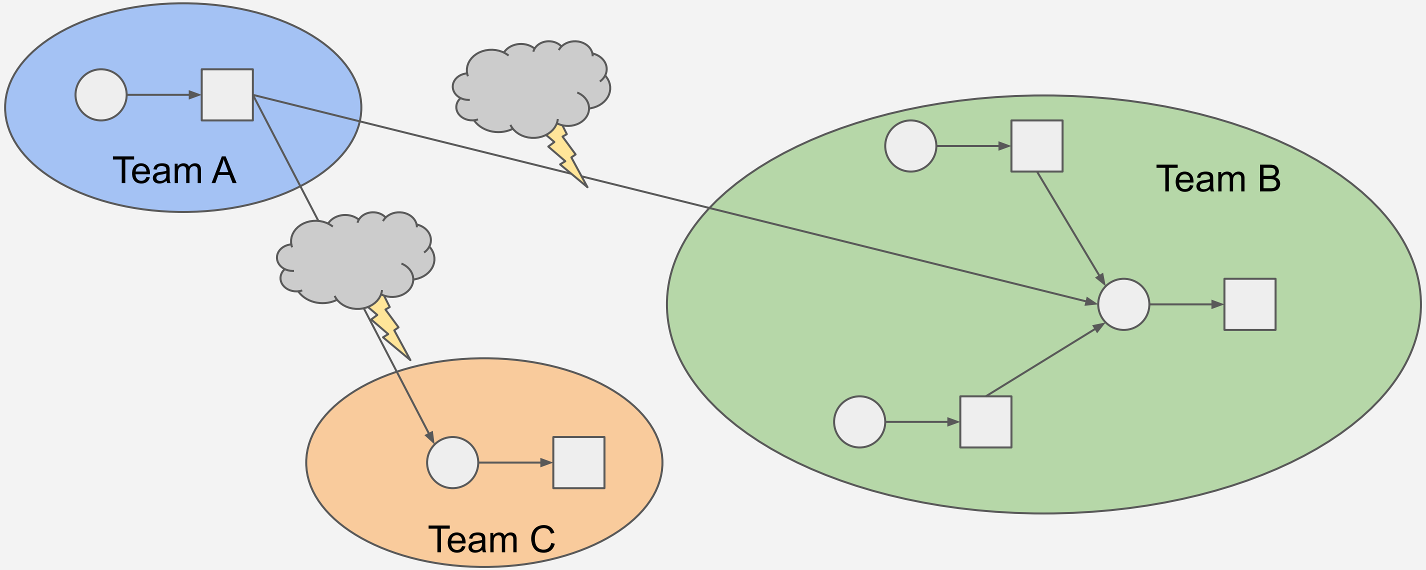
01 A Healthy Data Ecosystem

02 Data Platform Abstractions and Services

03 Observability for data pipelines

# 01 A Healthy Data Ecosystem

# Team interdependencies



# Explicit contracts

- Schemas
- Shared or Private
- SLA: experimental, production ready

# Understanding dependencies

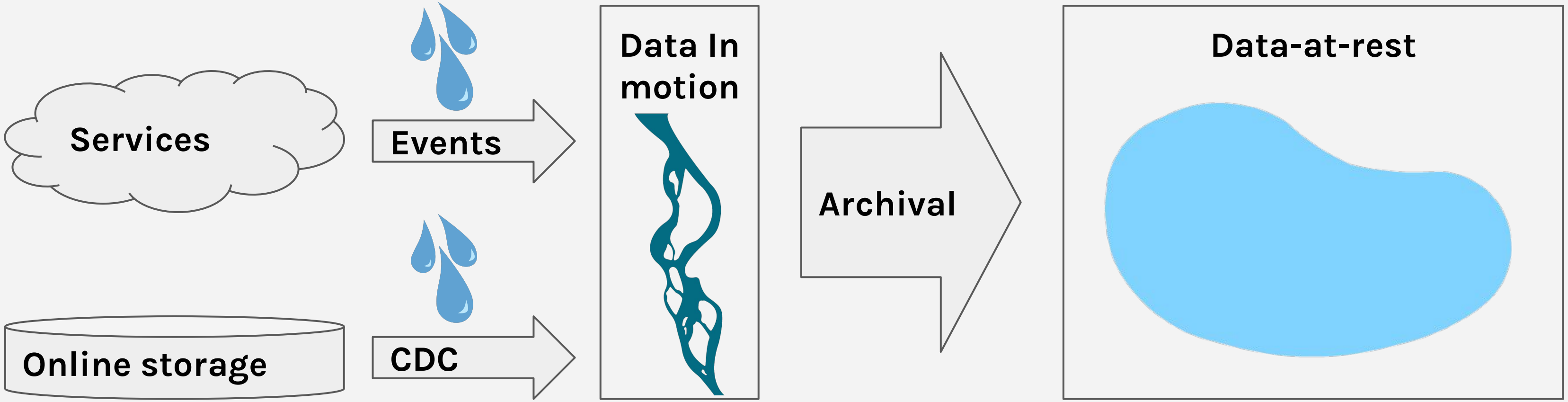
- Who do I depend on?
- Who depends on me?

# Quick iterations

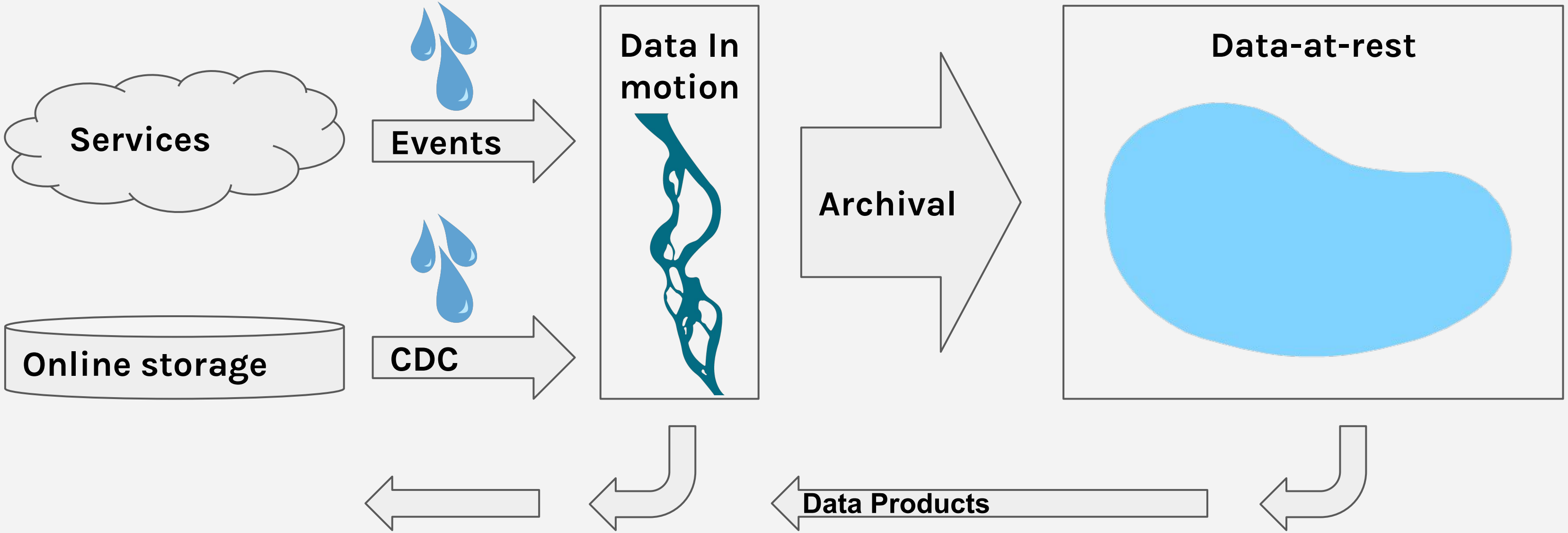
- Fail safe environment: Easy to undo
- Quick troubleshooting
- Quick feedback

# 02 Data Platform Abstractions and Services

# Storage and ingestion



# Storage and ingestion



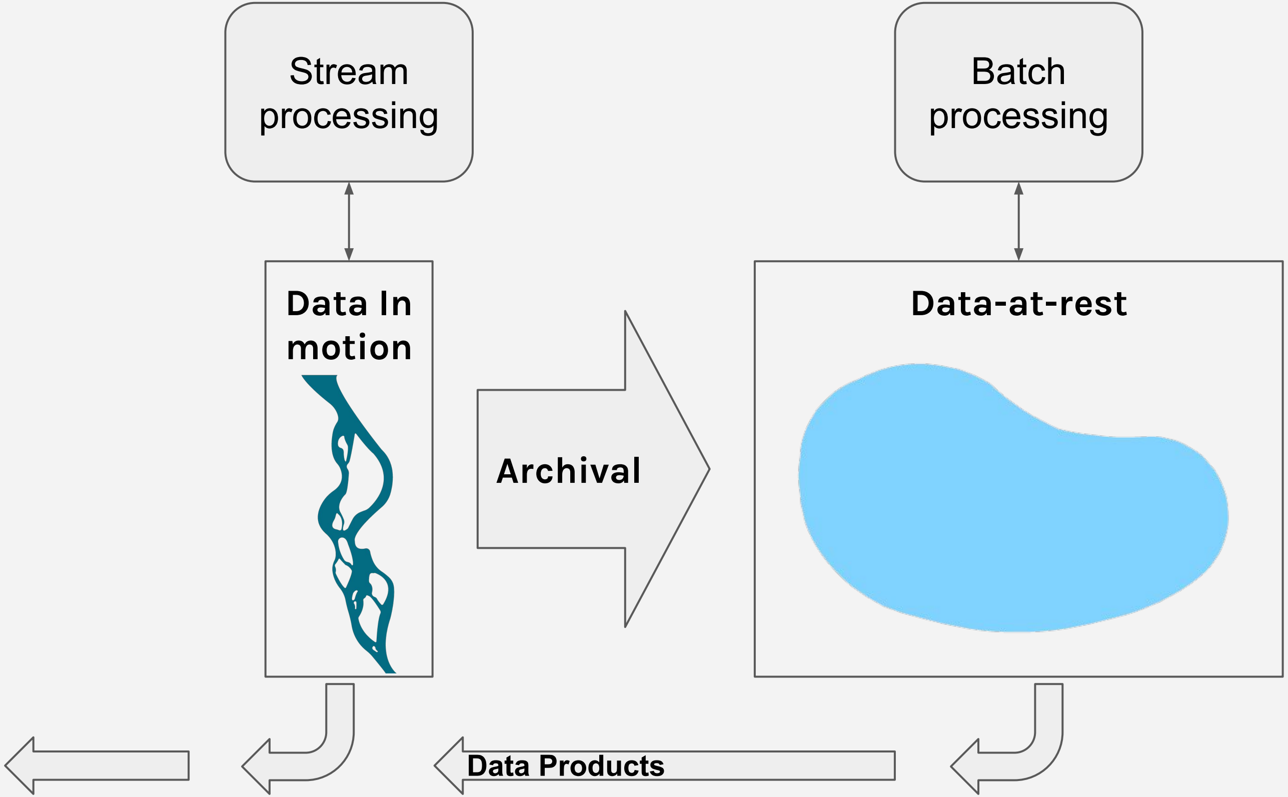
# Data-in-motion

- Schema registry
- Keyed for CDC
- Horizontally scalable
  - Partitioning
- Candidates: Kafka, Pulsar, ...

# Data-at-rest

- Table abstraction:
  - Snapshot Isolation
  - Time travel: can roll back a change
  - Schema evolution
  - Partitioning decoupled from job
- Candidates:
  - Iceberg,
  - Deltalake over cloud blob storage

# Processing



Stream processing

Batch processing

Data In motion

Archival

Data-at-rest

Data Products

# Stream processing

- Anti-pattern:
    - Dependencies outside the streaming bubble:
        - Synchronous service calls
        - Database lookup
    - Ingest that data instead (CDC / Domain events)
        - kafka.KTable, flink.DynamicTable
- Candidates:
    - Flink, Spark Streaming, Kafka Streams

# Batch processing

- Your job as a function: inputs and outputs are parameters.
  - Testable transformation:
  - Multiple instances in parallel
- Atomic runs:
  - output is complete or not visible
- Understand dependencies
  - Jobs depend on their inputs

# Interactive

- Notebooks:
    - Source control for saving state
    - Repeatable environments: docker images
- Warehouse technology:
    - Decoupled storage and compute
    - Interconnection with data storage

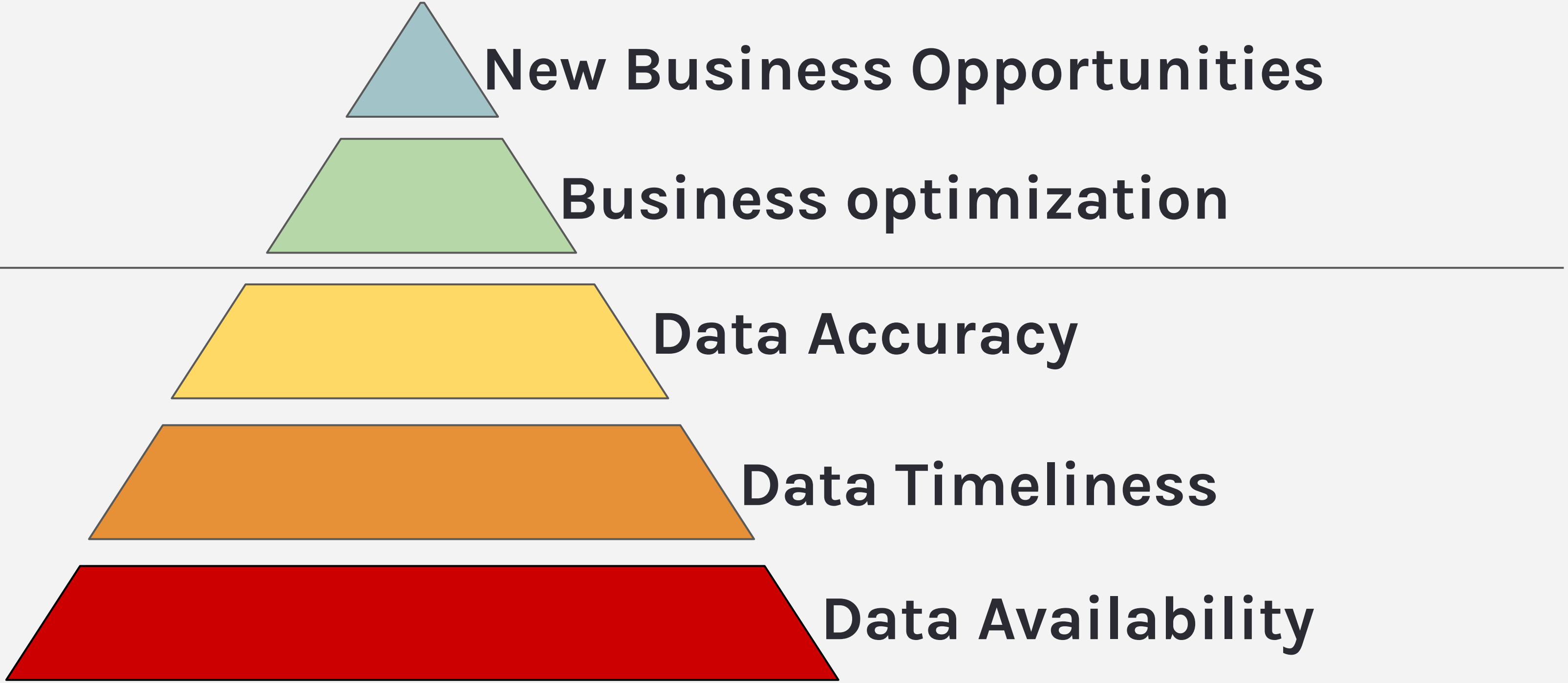
# 03 Observability for data pipelines

# Today: Limited context

DATA

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

# ~~Maslow's~~ Data hierarchy of needs



# Observability for data

- Dependencies: Lineage
- availability, timeliness, accuracy
- Change management
  - Schema
  - Code
  - Size
  - Duration

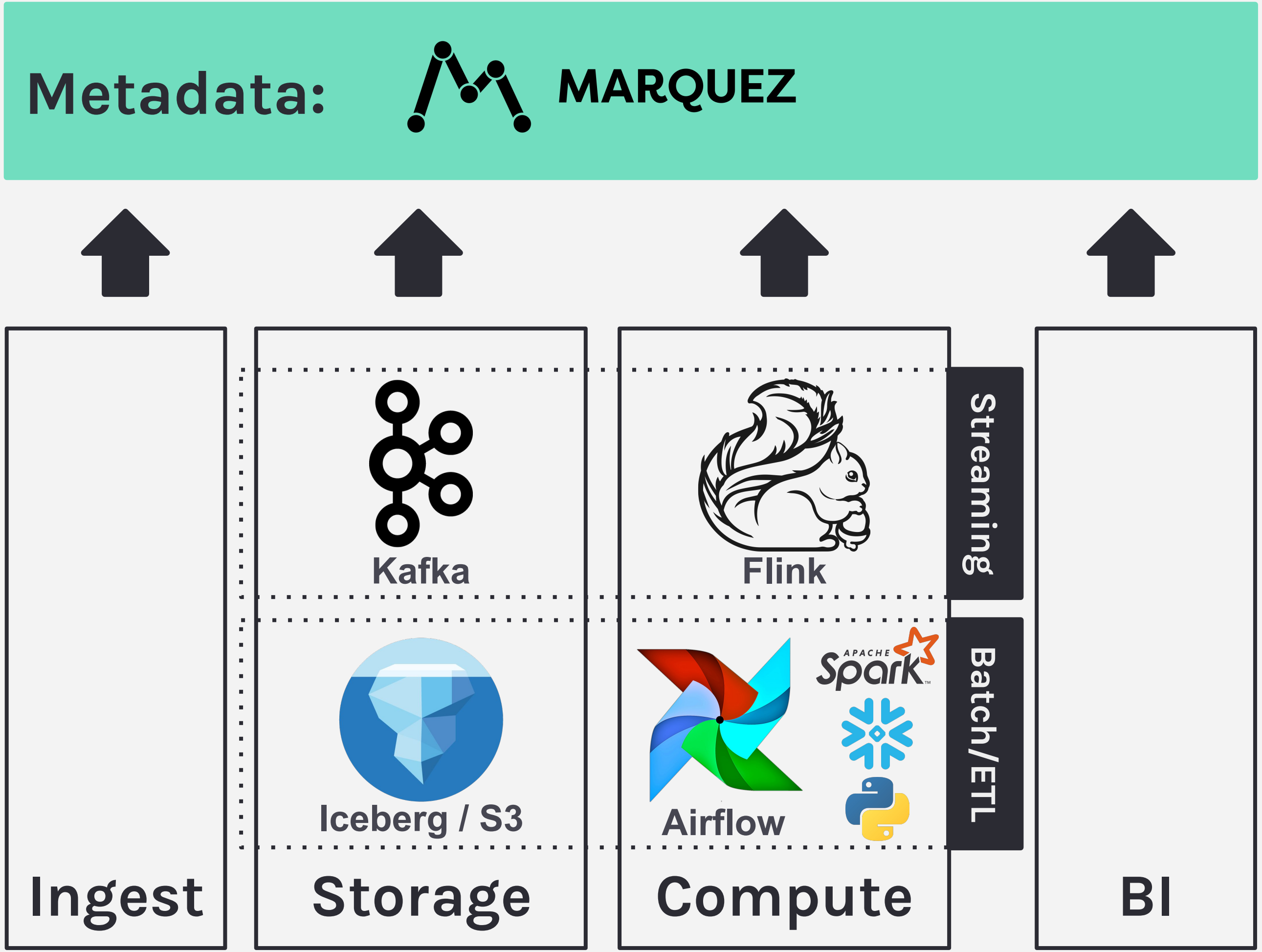
# Observability for data

In the **services** world it's called **traces**

- Dependencies: Lineage
- availability, timeliness, accuracy
- Change management
  - Schema
  - Code
  - Size
  - Duration

- **Data Platform** built around **Observability**

- **Integrations**
  - Ingest
  - Storage
  - Compute
  - BI dashboards



Metadata: MARQUEZ

Streaming

Kafka

Flink

Batch/ETL

Iceberg / S3

Airflow

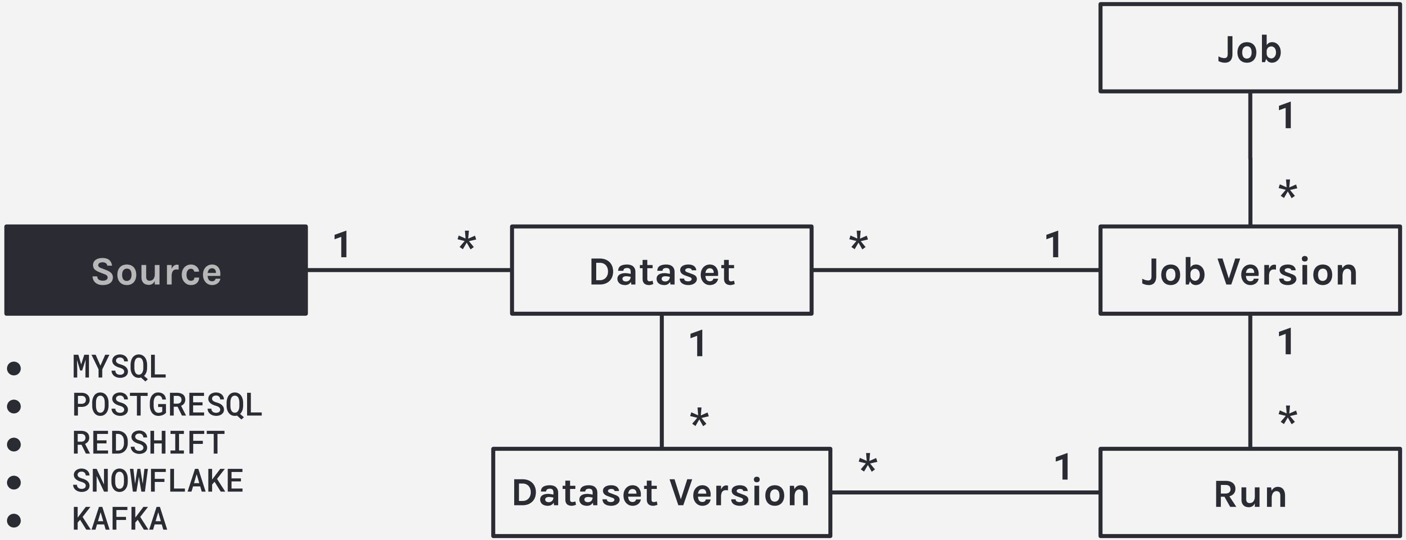
Ingest

Storage

Compute

BI

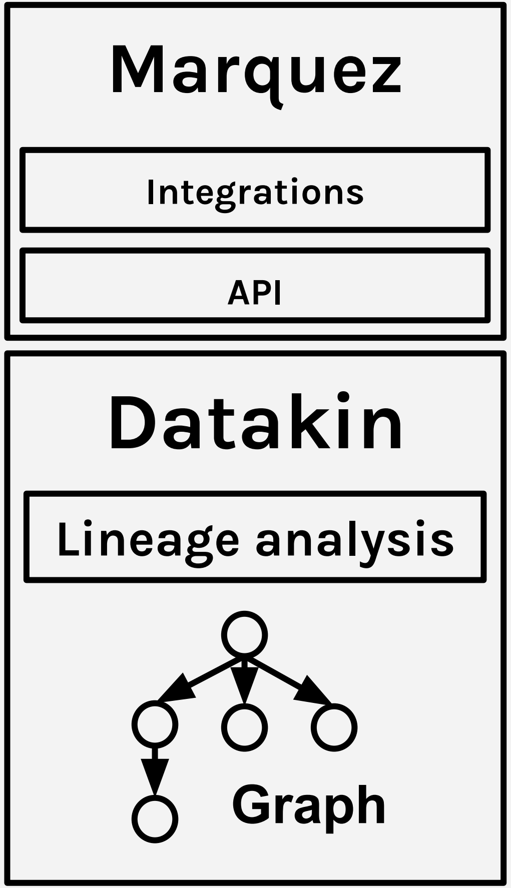
# Marquez: Data model



- BATCH
- STREAM
- SERVICE

- MYSQL
- POSTGRESQL
- REDSHIFT
- SNOWFLAKE
- KAFKA
- S3
- ICEBERG
- DELTALAKE

# Datakin leverages Marquez metadata



Marquez

Integrations

API

Datakin

Lineage analysis

Graph

- **Marquez standardizes metadata collection**
  - Job runs
  - parameters
  - version
  - inputs / outputs
- **Datakin enables**
  - Understanding operational dependencies
  - Impact analysis
  - Troubleshooting: What has changed since the last time it worked?

# Thanks! <o/

# Questions?