

Data Observability By Openlineage



The Smart Pizza & AI Way

Shuhsi Lin

20251015

Hello-World Conference

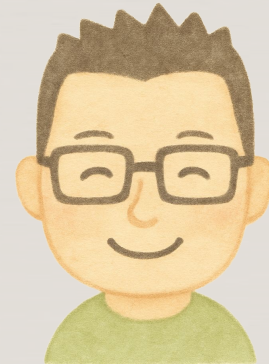


About Me

Working in **Smart Manufacturing & AI**
With **data** and **people**

Focus Areas

- Agile/Engineering Culture/ Developer Experience
- Team Coaching
- Data Engineering

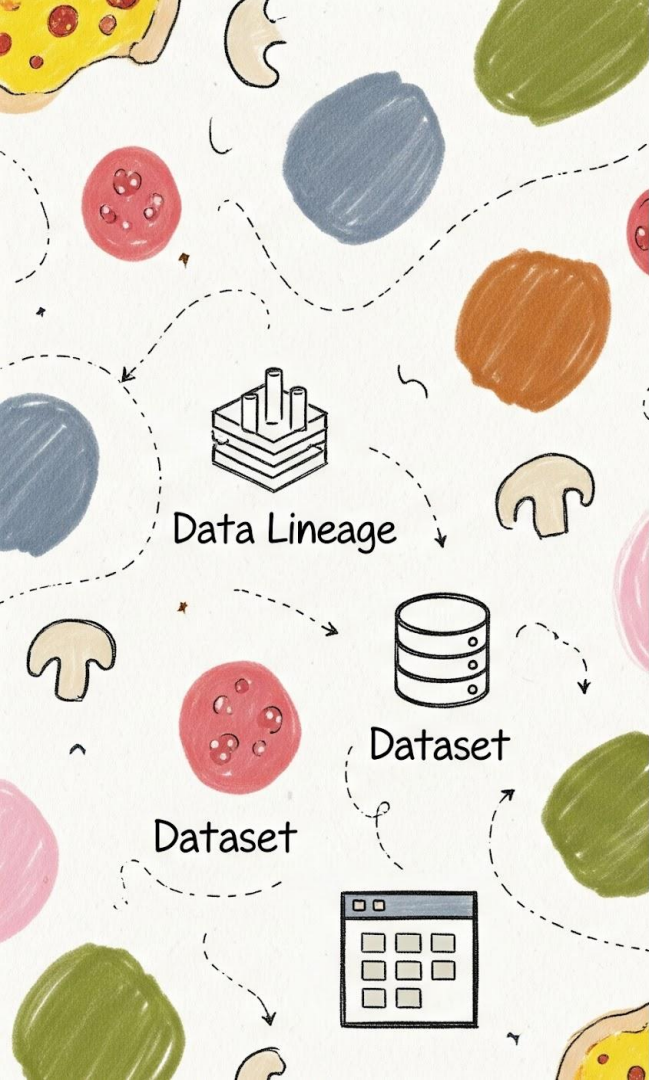


Shuhsi (Suci)



sciwork member

Find me on 



Outline

Start with Basic

Stories in Smart Pizza & AI

Common Data Engineering Challenges

From Monitoring to Observability

Observability and Data Observability

Lineage

Openlineage

The Evolution of Data Observability

Takeaway



[Slide link](#)

Scenario: Smart Pizza & AI

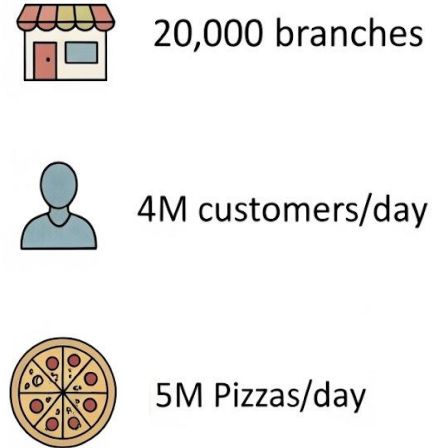
Orders



Core Tables



Scale Assumptions



Smart Pizza Dashboard Anomaly

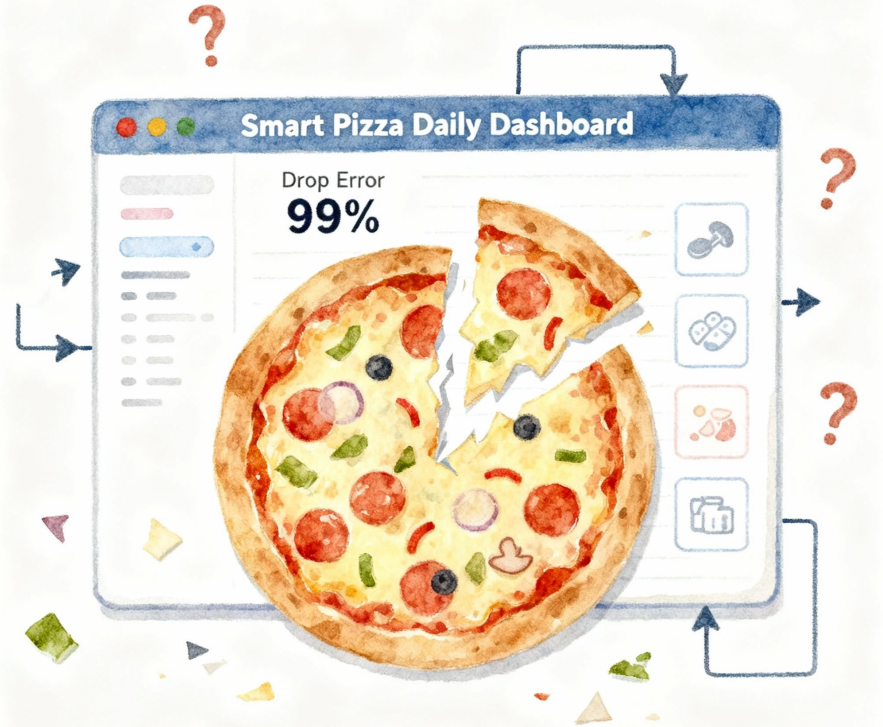
- Daily Sales Dashboard suddenly showed a 99% drop in sales.
- Analytics model had failed
- **Three hours** of backtracking, discovered that the POS (point-of-sale) data import had broken—leaving sales data out of the dashboard entirely.
- **Took much longer** to isolate the root cause.
- **No clear lineage** showing data flow and dependencies



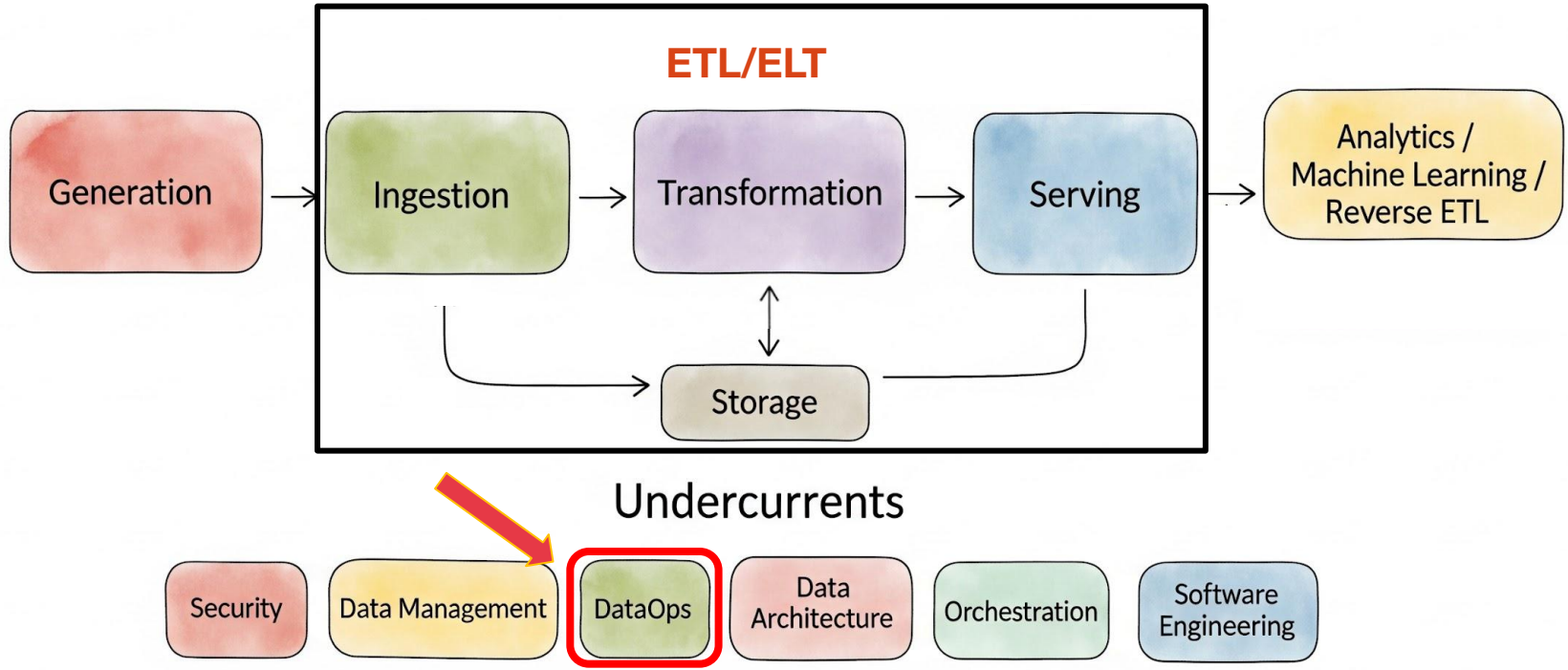
- What is the data source? (Enables traceability)
- What is the schema? (Defines structure)
- Who owns the data? (Clarifies accountability)
- How often is it updated? (Shows freshness)
- Where does it originate? (Ensures transparency)
- Who uses the data? (Tracks impact)
- What has changed? (Supports auditing and trust)



The need for Metadata



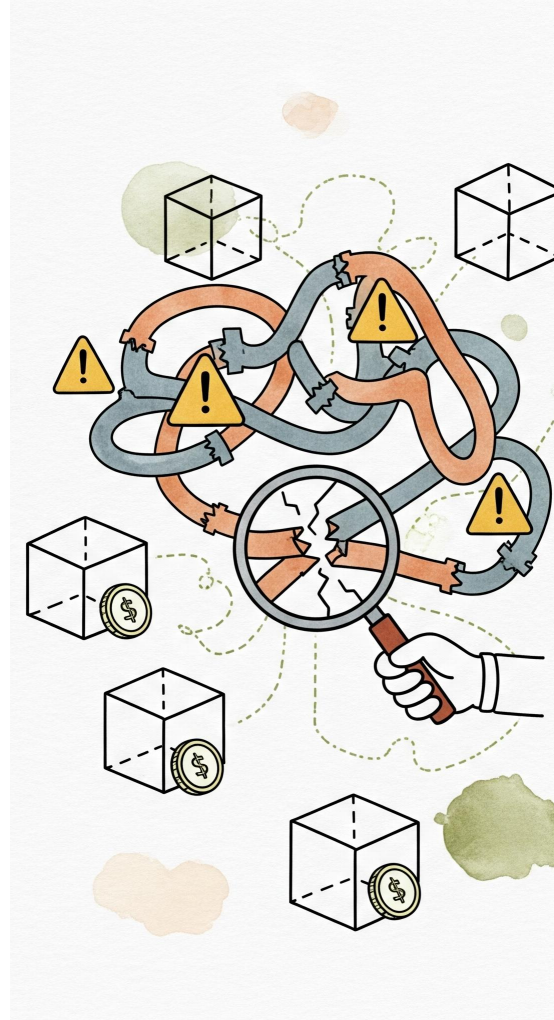
Data Engineering Challenges



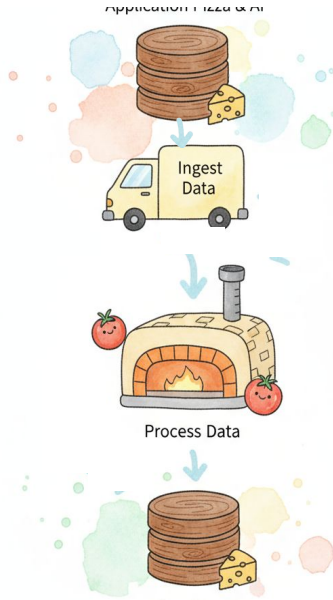
Data Engineering Challenges

How to manage pipelines effectively

- Complex data pipelines
- Inability to centrally view
 - Limited data asset discoverability
 - Error detection and root cause analysis
 - Scattered monitoring and time-consuming troubleshooting
 - Optimization and monitoring workloads at scale
 - Inefficient pipelines that negatively impact data quality and costs

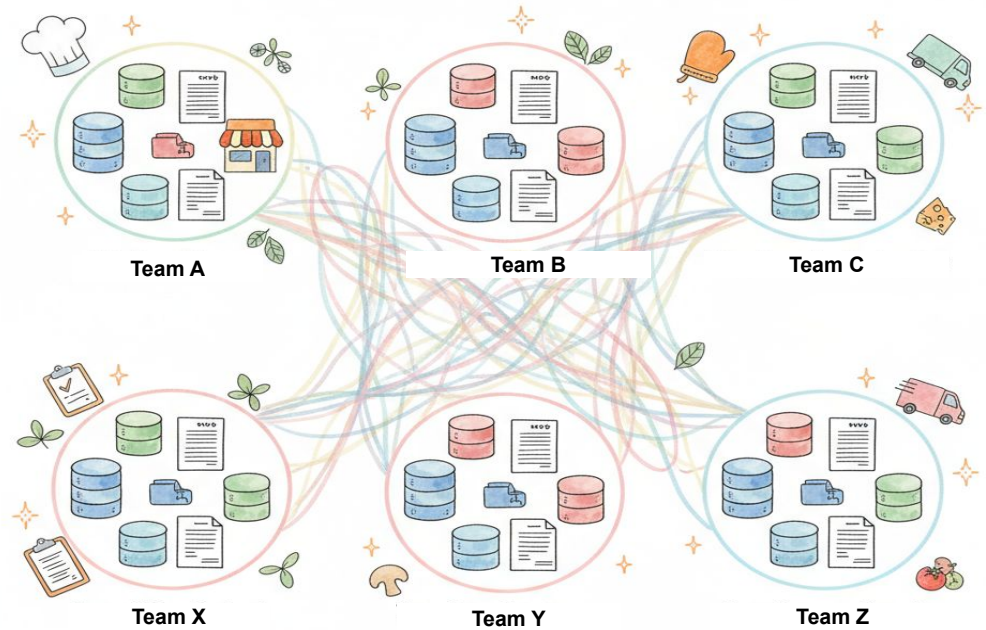


Simplistic Data Flow



- Data movement as flow
- Moving data content from A to B

Many Flow-like Data in a Real World



From Monitoring to Observability

Why Traditional Monitoring is Not Enough?



Monitoring

WHERE the issue is

- Measure and report specific metrics in a system.
- Reactive – collect data to identify abnormal systems.
- **WHEN** and **WHAT** did the system error occur?
- Smart Pizza & AI Example:
 - Checking only if the pizza oven is on and the thermometer is working.
 - Alerts when 'Order Sync Task failed' or 'Order DB CPU > 90%'.

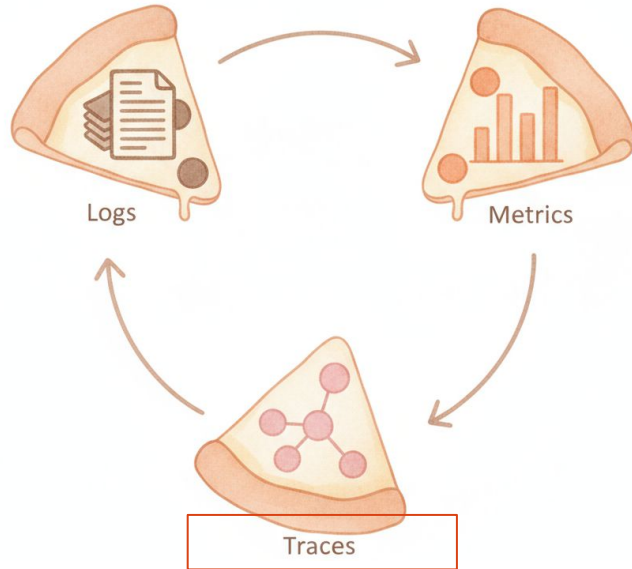


Observability

WHY it happened

- Collect metrics, events, logs, and traces across distributed systems.
- Proactive – investigate root causes of abnormal systems.
- **WHY** and **HOW** did the system error occur?
- Smart Pizza & AI Example:
 - A smart oven that not only tracks temperature but also analyzes pizza color, dough rise, and past baking data to diagnose why it burned.
 - The accuracy dropped due to a schema change in an upstream API that nullified the `pizza_type` field.

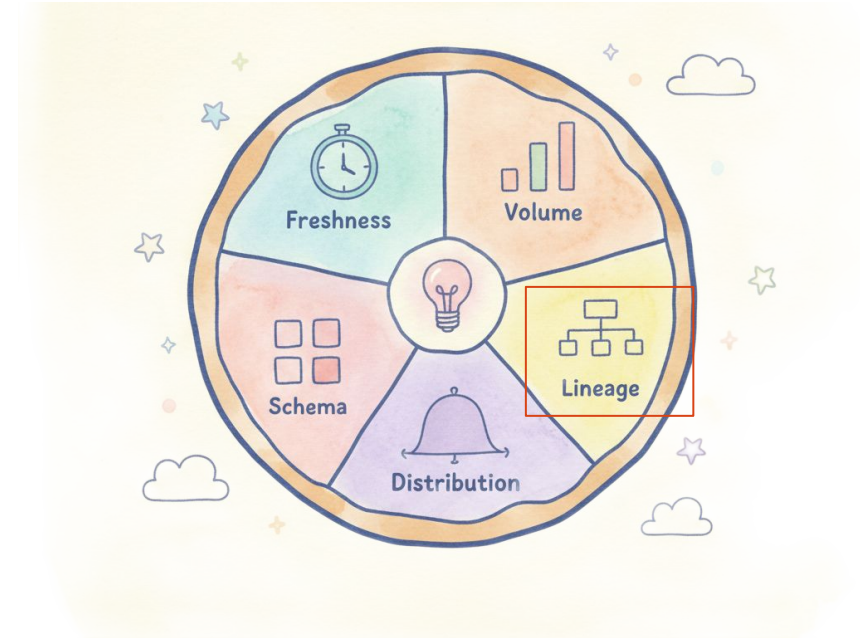
3 Pillars of Observability



Tracing focuses on:

- Performance metrics
- Error tracking
- System behavior
- Service interactions

5 Pillars of Data Observability



Lineage focuses on:

- Data transformation tracking
- Compliance and governance
- Data quality

Three Key Focus Areas of Data Observability

Infrastructure



Pipeline



Data



• Focus

- Hardware & services running pipelines

• Metrics

- CPU, memory, disk, network

• What we want to know

- Is the ML cluster (training AI models) overloaded?



• Focus

- Data transfer & processing flow

• Metrics

- Task duration, success/failure rate, retries

• What we want to know

- Did the daily ETL for orders finish on time



• Focus

- Data content, structure & quality

• Metrics

- Freshness, volume, distribution, schema, lineage

• What we want to know

- Are order fields valid? Any anomaly in new order volume?



Prometheus OpenLineage



OpenMetadata

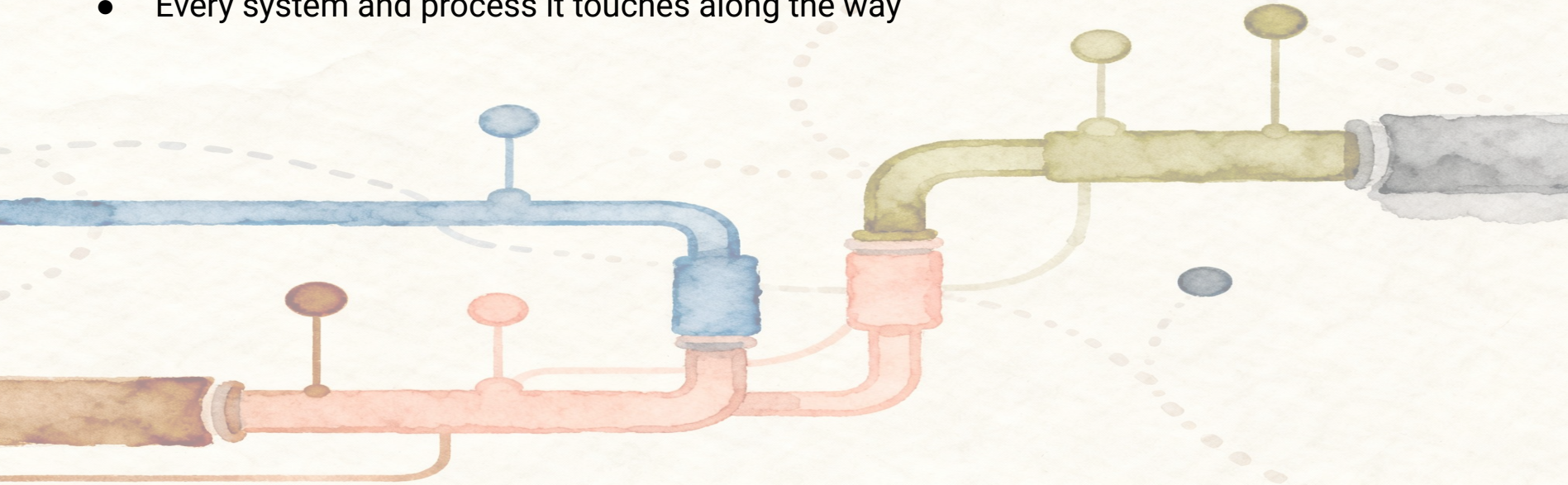


great expectations

What is Data Lineage

The process of **mapping and tracking how data flows** throughout its lifecycle in an organization

- Where a piece of data came from (origin/source)
- How it has changed (transformations, calculations, integrations)
- Where it is used or stored (destination or endpoint)
- Every system and process it touches along the way





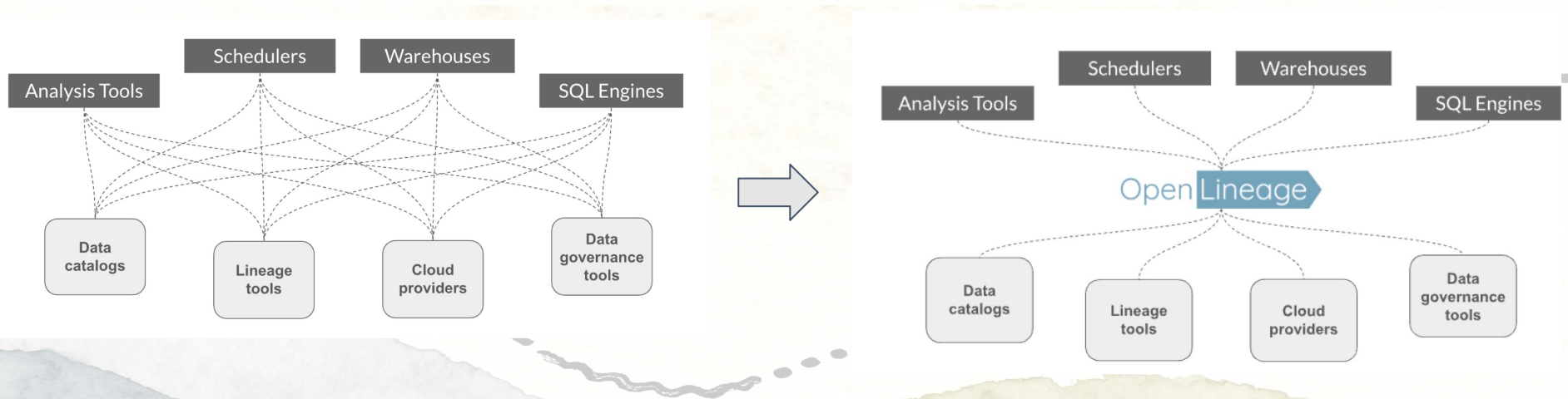
What is OpenLineage



OpenLineage



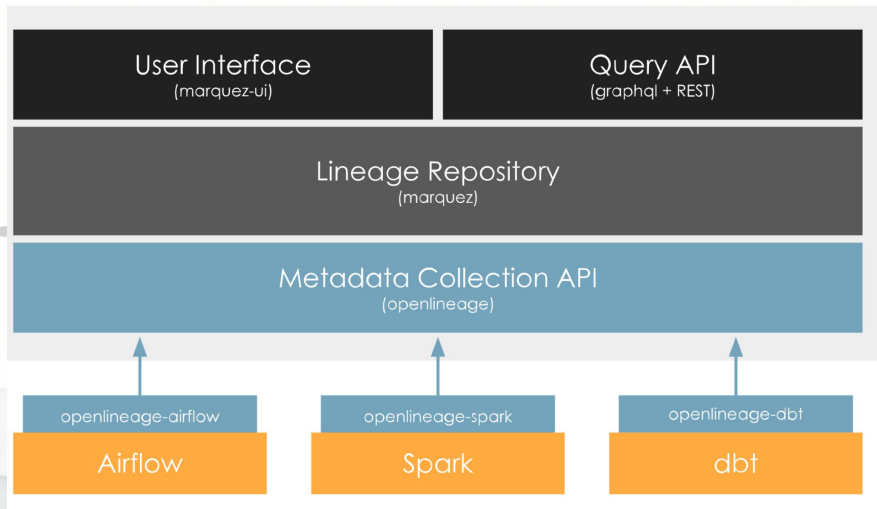
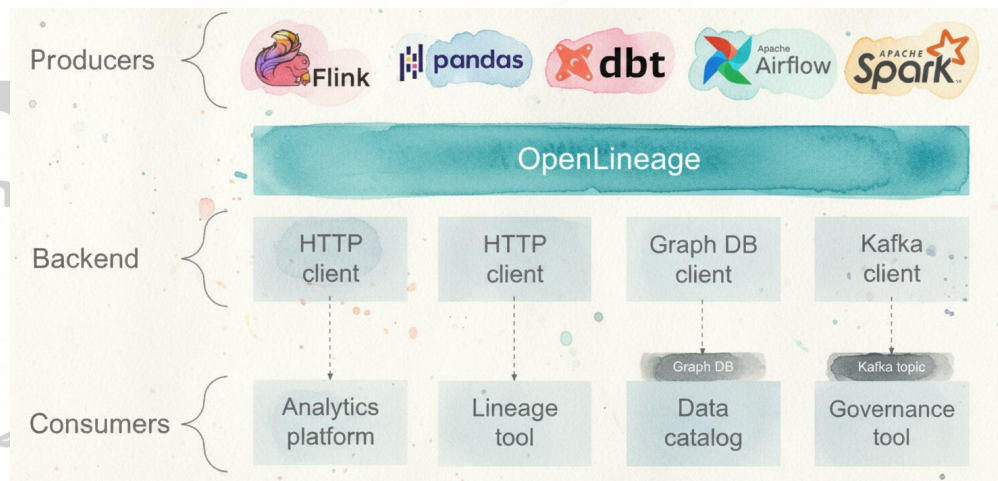
- **Open framework** for data lineage collection and analysis
- Integrations can be *pushed* to the underlying scheduler and/or data processing framework; no longer does one need to play catch up and ensure compatibility





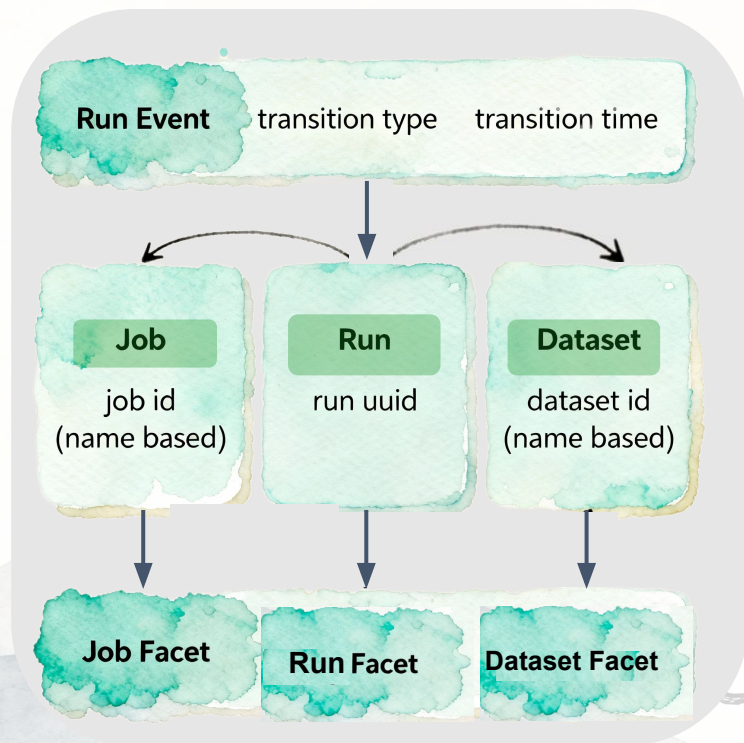
Scope

OpenLineage defines the metadata for running jobs and the corresponding events. A configurable backend allows the user to choose what protocol to send the events to.

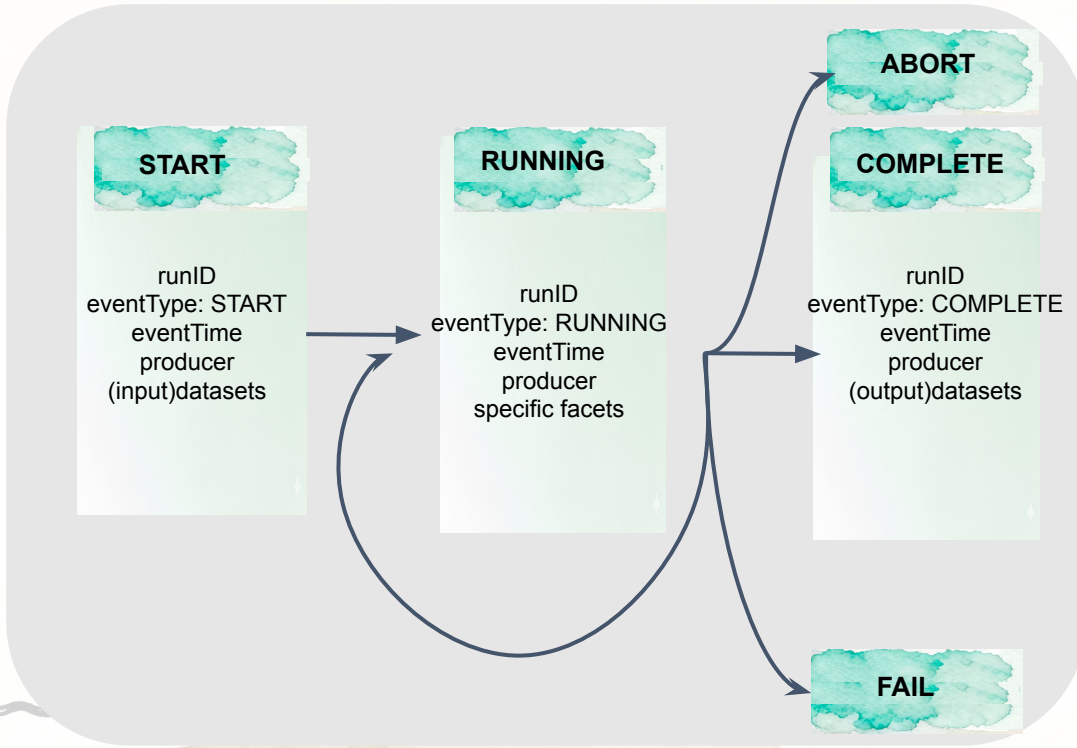




Core Concept

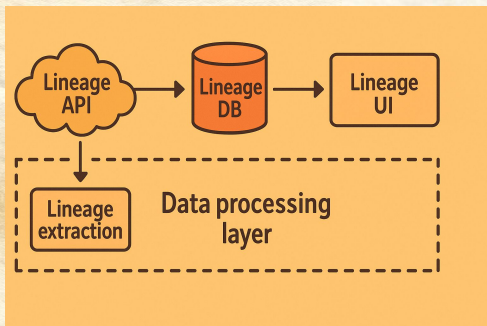


Run Event Lifecycle

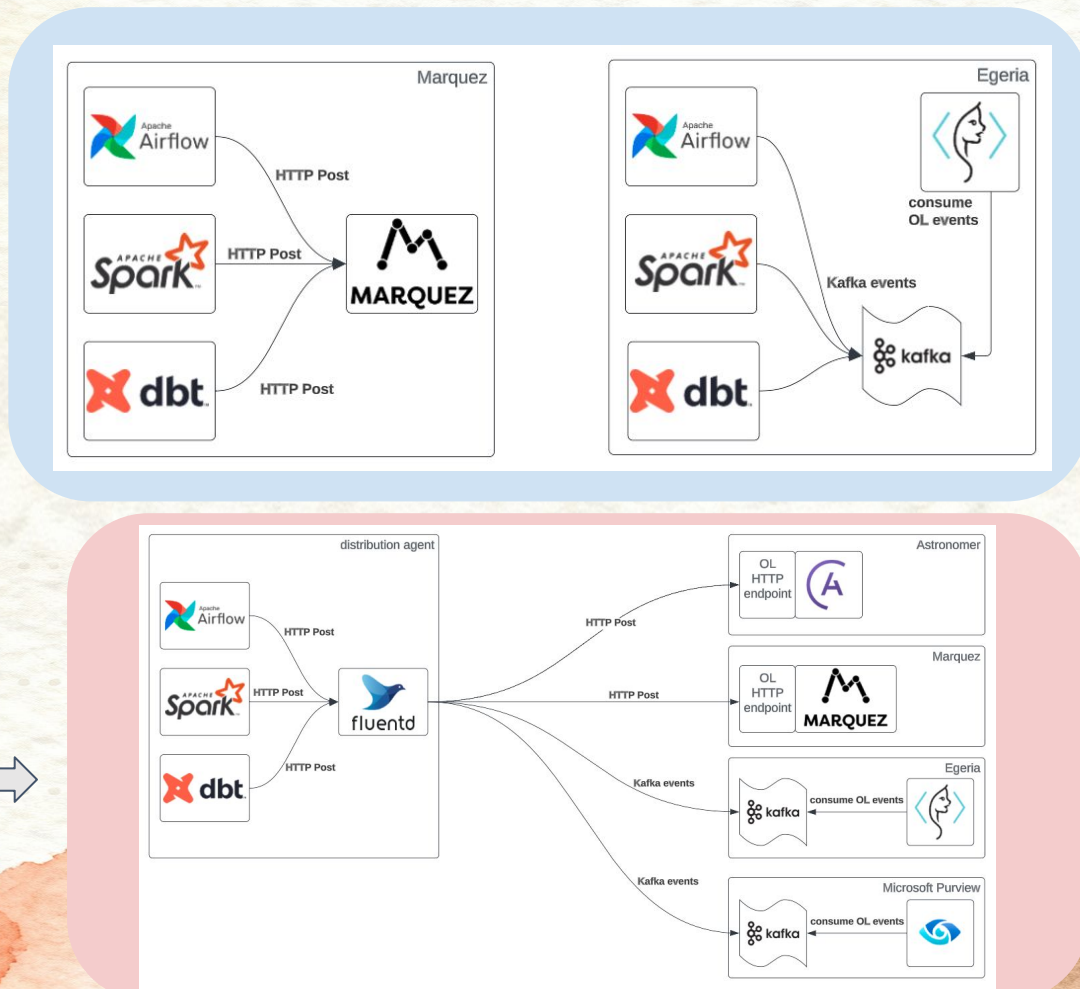


How to Deploy OpenLineage

Simple →



Complex →



Best Practices for implementing OpenLineage

1. Design Principles

- **Start simple** (Job-level then message-level)
- **Prioritize coverage** (basic lineage across all pipelines than detailed lineage)
- **Standardize naming** ([airflow](#))

2. Metadata Strategy

- **Trace ID propagation**
- **Enrich context** (business-layer information)
- **Appropriate granularity** (Row/message level, column/field level, schema level, and change management level)

3. Operational Considerations

- **Performance impact:** eg. Avoid heavyweight processing in `execute()` methods
- **Error handling:** Ensure lineage tracking failures don't affect primary data tasks
- **Retention policy:** Define retention periods for different types of lineage data
- **Access control:** Lineage data may reveal sensitive architecture information

4. Data Governance Integration

- **Connect to data catalogs:** Integrate with OpenMetadata, etc.
- **Support compliance:** Use lineage information to meet GDPR, CCPA requirements
- **Impact analysis:** Assess downstream impacts before changes

Data Observability Design Pattern

Data Detectors

 Flow Interruption Detector

 Skew Detector

Time Detectors

 Lag Detector

 SLA Misses Detector

Lineage Trackers

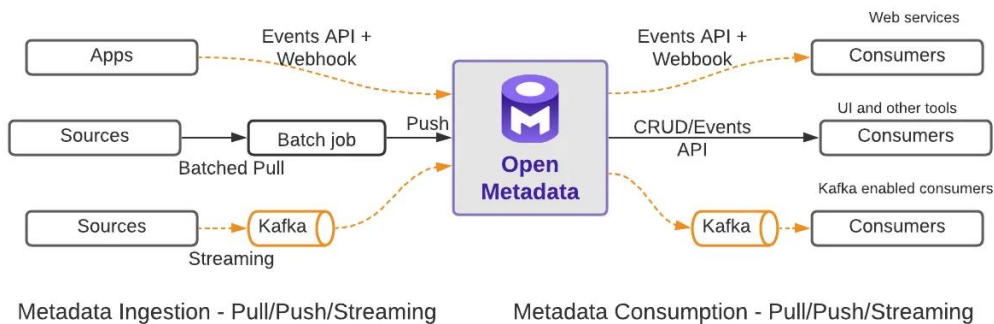
 Dataset Tracker

 Fine-Grained Tracker

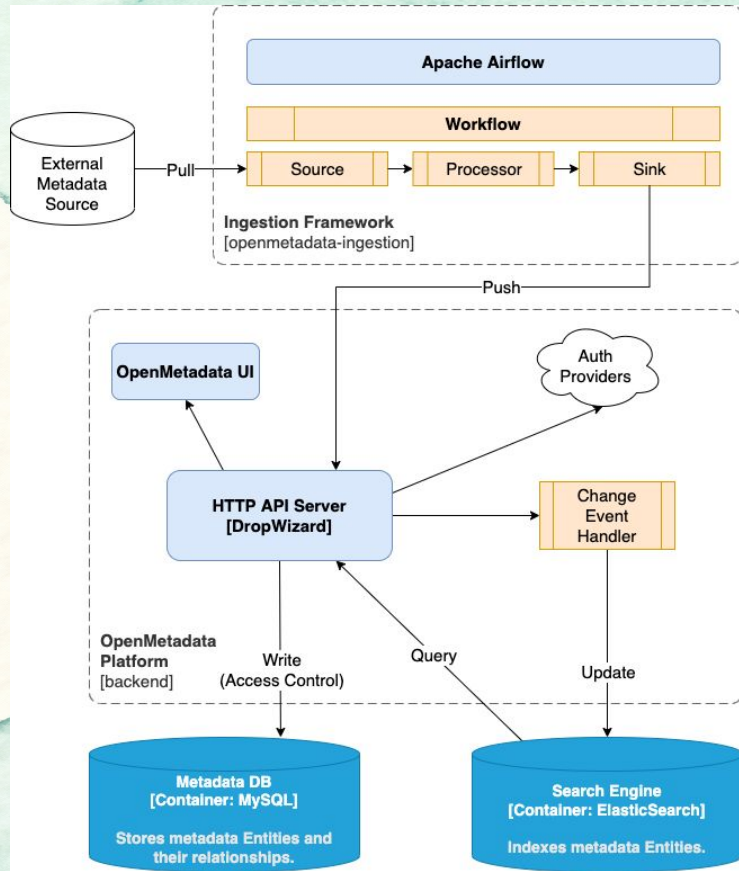




OpenMetadata is a unified platform for **discovery, observability, and governance** powered by a central metadata repository, in-depth lineage, and seamless team collaboration.



<https://blog.open-metadata.org/why-openmetadata-is-the-right-choice-for-you-59e329163cac>



<https://docs.open-metadata.org/latest/developers/architecture>

Observability 1.0

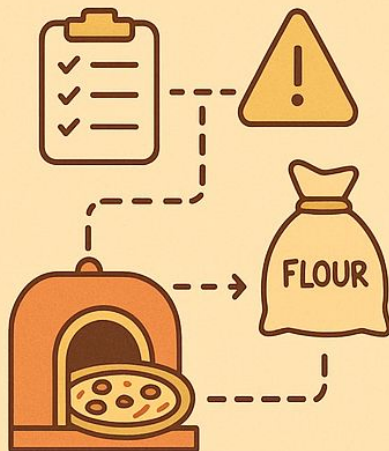


**CHECKING
OVEN
TEMPERATURE**

Prometheus+PromQL, Loki+LogQL,
Tempo+TraceQL



Observability 2.0

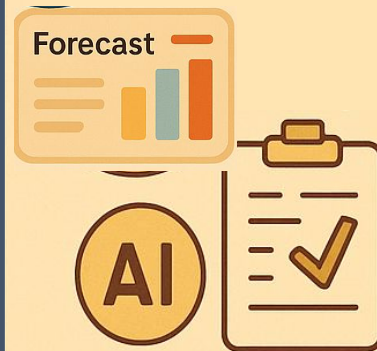


**TRACKING ORDERS,
OVENS,
INGREDIENTS**

OpenTelemetry + OTLP



Observability 3.0



**DERIVING
INSIGHTS,
MAKING
PREDICTIONS**

AI + Observability

The Evolution of Data Observability



Infrastructure Centric

'Is it running?'

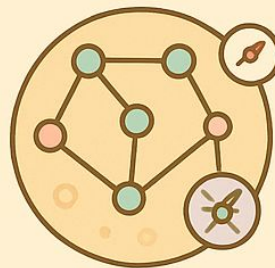
Issue:
Alert fatigue



Shift Left

Catch early
with tests

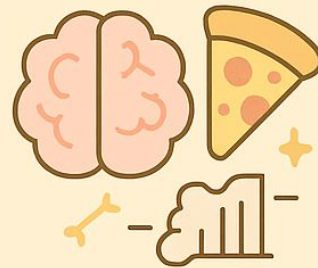
Issue:
Tech-only
responsibility



Lineage & Catalog

We know
the source

Issue: Lacks
business meaning



Context-Aware & Intelligent

Failure → business
impact alert

Issue: Decision-
ready insights

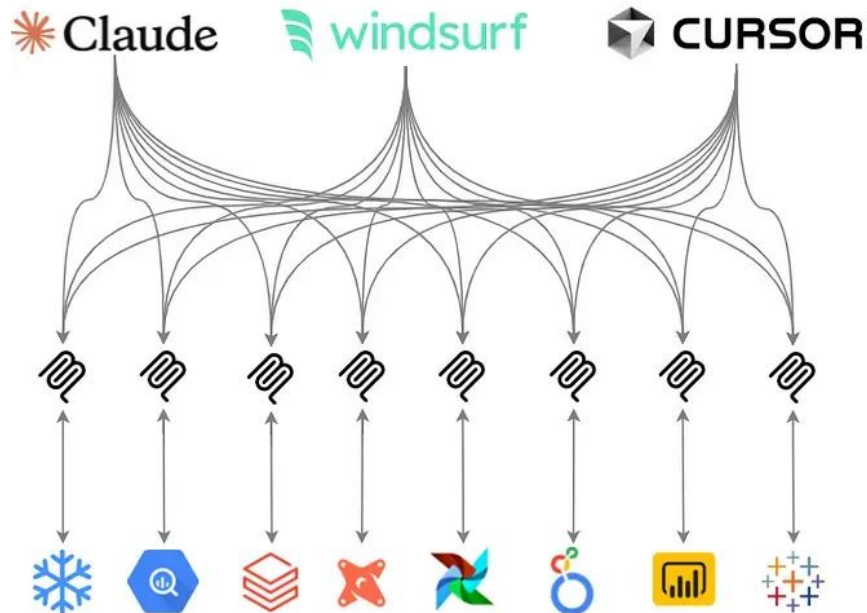
From Monitoring to Intelligent Agents



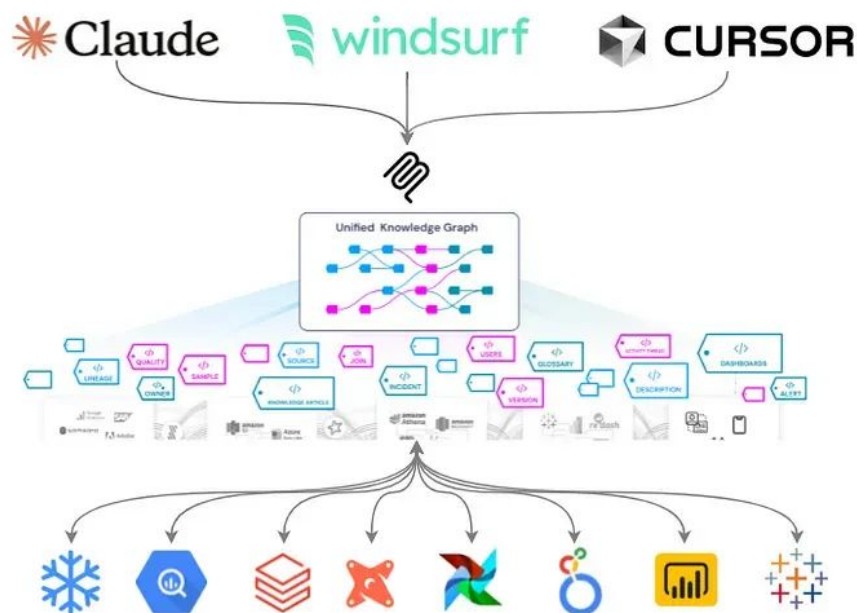
Model Context Protocol (MCP) in OpenMetadata

Context-Aware & Intelligent Observability

Today



OpenMetadata

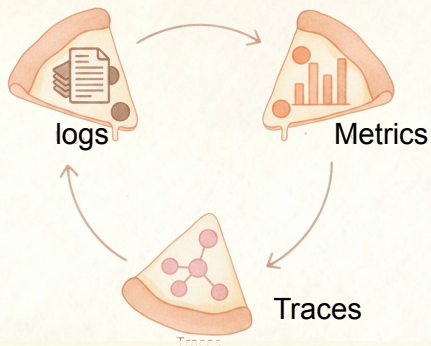
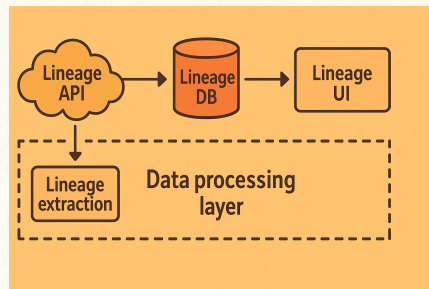


Smart Pizza & AI

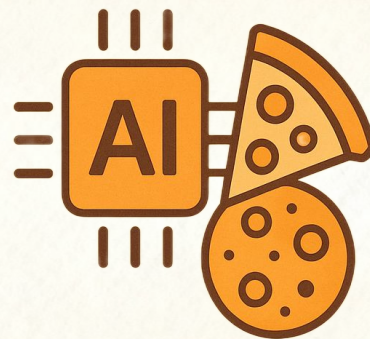
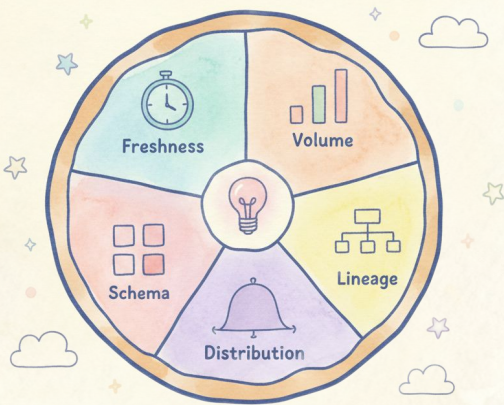
- Smart Pizza & AI
- 3 Observability Pillars
- 5 Data Observability Pillars
- Metadata & Data Lineage
- Openlineage

Takeaway

- Observability Design Patterns
- Context-Aware & Intelligent Observability



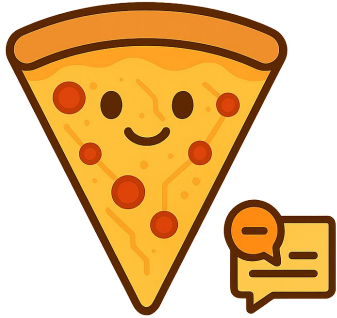
OpenLineage



Let's make data observable & AI trustworthy



Q&A



Data-Centric